FLOW NEURAL NETWORK AND FLOW-STRUCTURED DATA REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

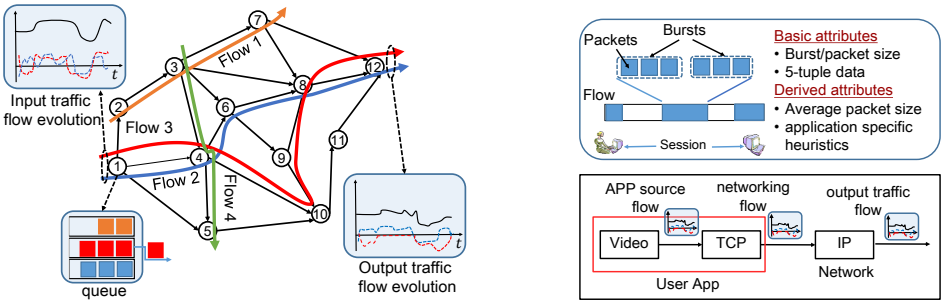
Traffic flows are the most fundamental components in a communication networking system. An accurate understanding of these flows is crucial for many downstream network applications. However, the high nonlinearity, randomness and complicated self similarity (Leland et al., 1994) of these traffic thwart extensive traditional analytical and learning models, particularly in high time resolution. In this paper, we experimentally discover the spatio-temporal induction effect, a universally present property in flow-structured network traffic data. By exploiting the flowing invariance and variance, we propose *FlowNN* and induction operation to learn the representations of flow-structured data in communication networks. Results from regression tasks demonstrate several times better accuracy than the state-of-the-art baselines and extend the existing time resolution from second to millisecond, offering a good testament to the strength of our approach.

1 INTRODUCTION

Deep learning has made major progress in key domains such as vision, language, and relational topology. Normally, these data are organized in the form of three typical data structures (grid, sequence or graph) (Goodfellow et al., 2016). The tremendous efforts on customizing neural network structures to explore the data specific properties underpin the success of modern deep learning in these domains (Krizhevsky et al., 2012; Kiros et al., 2015; Hochreiter & Schmidhuber, 1997). This summarizes the procedures of learning work for the majority applications – 1) refine the data specific properties with domain expertise (e.g., regional correlation in images or local smoothness in time series), 2) and then customize a neural network structure to capture/represent these properties with certain inductive bias (Battaglia et al., 2018). With the same design philosophy, we rollout a radically new data type – *flow-structured data*, as well as the learning model – ***F**low **N**eural **N**etwork (FlowNN)*. Albeit derived from the traffic flows in Internet Protocol (IP) based communication networks, *FlowNN* can be potentially used to learn for any application whose data can be organized in the form of flow structure (e.g., transportation, financial transaction, fluid dynamics, etc).

Specifically, a digital communication network is a group of nodes interconnected by links that are used to exchange messages between the nodes using a common communications protocol e.g., TCP/IP (Hunt, 1992). Network nodes are forwarding/computing devices that originate, route and terminate data communication. They are generally identified by network addresses (IP address), and can include hosts such as personal computers, phones, and servers, as well as networking hardware such as routers and switches. Data transmission between nodes is supported over data links consisting of physical cable media, such as twisted pair or fiber-optic cables, etc. Such kind of networking infrastructure provides us daily Internet services (e.g., watching movies, online gaming, tweets...) and makes nowadays' world all connected. Therefore, it is of paramount importance to understand the underlying networking systems.

Fig. 1 illustrates the common network structure and traffic flow structure. Basically, a traffic flow encapsulates the source messages from user applications into a set of ordered packets, which normally follow a given routing path to traverse the network in the same order before reaching its destination. Each traversed network node would directly forward/retain in a queue/discard the received packets according to its processing capacity/ queuing state/ flow priority, thus leading to different end-to-end transmission experience (e.g., delay/packet loss) for different flows.



(a) (b)

Figure 1: (a) IP network structure; (b) network traffic flow structure and attributes

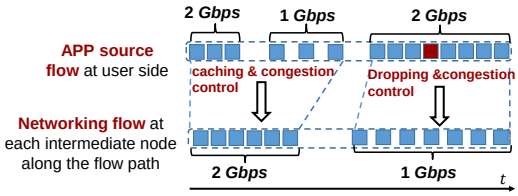
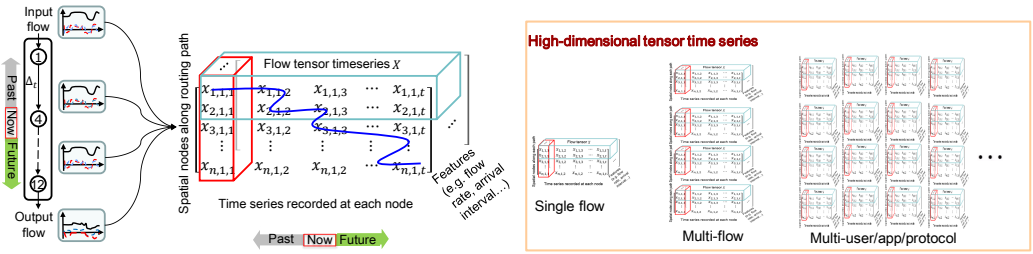


Figure 2: Data transformations by possible network actuation. **APP source flow**: original packet flow representing the user behaviors and message patterns of the source application; **Networking flow**: the packet flow actuated by all possible networking control actions, e.g., congestion control, caching, and packet dropping.

An accurate understanding of these flows is crucial for many downstream network applications, e.g., gaming acceleration, watching higher-resolution video, supporting self-driving Tesla cars, etc. With the growing network dynamics (in possibly subsecond order), concurrent flow scale (more than 1 million flows) and network size (tens to tens thousand of nodes and links), the accurate network-wide flow analysis and modeling in IP networks remains a formidable challenge. Many mathematical modelling approaches, including traditional queuing/ optimization/ control/ network calculus theories (Harchol-Balter, 2013; Ciucu & Schmitt, 2012; Ciucu et al., 2019), have been attempted, but still failed to work efficiently in practical networks. As the advent of many advanced machine learning (particular deep learning), intelligent flow analysis tools are gaining more momentum to proceed. Prominent examples (Rusek et al., 2019; Geyer & Bondorf, 2019; Xiao et al., 2018; Yu et al., 2018; Zheng et al., 2020) have provided a good testament to this data-driven learning principle. However, the current machine/deep learning tools are designed for structure-specific data tasks (e.g., image/audio analysis). They need to be further optimized/customized to capture the spatio-temporal features for flow-structured networking data and applications.

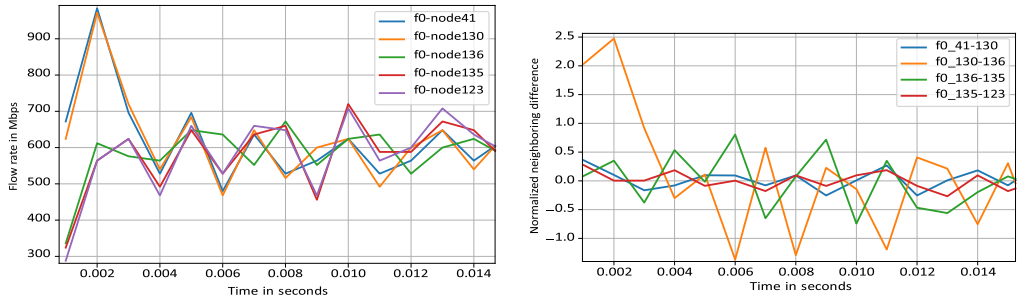
Additionally, in contrast to the learning analysis of formation-agnostic natural objects, the man-made networking system enjoys rich domain expertise to interpret the life-span experiences of traffic flows. Originated from an application client, the network traffic normally experiences a set of ordered control actuation by each traversed network nodes in the routing pathes before reaching their destinations. Each of such actuation imposes physically certain data transformation on the source traffic flows, creating multiple auto-correlated views of original spatio-temporal timeseries. This is exemplified in Fig. 2.

The flow-structured data can be organized compact and complete in the form of high-dimensional tensor time series, as shown in Fig. 3b. In contrast to the spatial/temproal attributes manifested in grid/graph/sequence structured data, such flow-structured data presents many peculiar properties. As depicted in Fig. 3a, the current data at node 4 was originated from previous time window of the data at its predecessor node 1, and will also *flow* to its successor node after some time interval. This connects with strong correlations the current flow state at a node to both the past flow state at its predecessor node and the future flow state at its successor node. Therefore, such time-resolved flow-structured data not only tells us *who* is related to whom but also *when* and in which



(a) (b)

Figure 3: Flow-structured data in the form of flow tensor time series. The S-shaped curve characterizes the message propagation road along the the spatio-temporal domains.



(a) (b)

Figure 4: (better viewed in color) (a) traffic flow evolution; (b) normalized traffic rate difference between neighbouring nodes.

order relations occur. This essentially discriminates against existing grid/sequence/graph-structured data types (e.g., image/audio/topology), for which the message patterns are normally aggregated through orthogonal spatial-temporal domains (i.e., follow either vertical or horizontal direction in Fig. 3a). Instead, in the flow-structured data, the message propagation road is S-shaped with interleaved spatial-temporal correlations.

In what follows, we will disclose the critical *spatio-temporal induction effect* that is universally present in the flow-structured data. Accordingly, a learning model is proposed to represent the network traffic flows with high resolution.

2 Spatio-Temporal Induction Effect

In the path-constrained propagation process as doing in above IP traffic flows, when the time elapses, the messages also flow forward and become the parts of downstream nodes in spatial domain, vice versa. Such synchronized changes in spatio-temporal domains disclose the **S**patio-**T**emporal **I**nduction (STI) effect as follows.

Definition 1. *Spatio-Temporal Induction defines the synchronized map process to produce the future/spatial evolution between neighboring nodes.*

This discloses the flow evolution is *originally*[1] driven by the past of predecessor nodes (i.e., the S-shaped road in Fig. 3a) rather than the past of the flow time series itself (as doing in Recurrent Neural Net and its variants).

An example of such flow tensor time series is given in Fig. 4. The depicted flow f_0 follows a routing path [41→130→136→135→123]. Clearly, a spike at a upstream node in Fig. 4a spurts the flow at

[1]The exotic competition from other co-located flows also drives the flow evolution in part. This can be included by appending the aggregated coflows as an auxiliary feature.

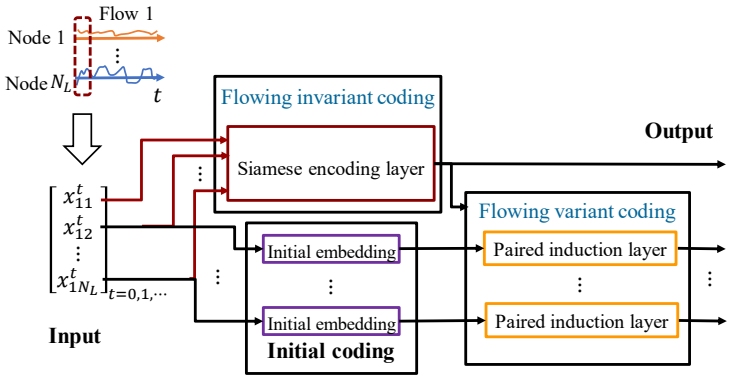


Figure 5: Framework of FlowNN.

downstream nodes in certain subsequent time windows. Further shown in the plot of Fig. 4b is the partially[2] local conservation between the flow rates at neighbouring nodes. That is, the flow rates between two neighbouring or any pair of nodes in the routing path will become larger interchangeably than the other so that the flux is conserved among nodes. This not only experimentally confirms the S-shaped spatio-temporal correlation present in the flow-structured data, but also further confines the correlation with the explicit (partial) flow conservation. The STI effect enjoys the following two properties.

Property 1 (*time-scaling invariance*)**.** *The STI effect holds when the flow tensor time series are time-averaged with any time resolution.*

Property 2 (*space-scaling invariance*)**.** *The STI effect holds when the flow tensor time series are spatially averaged/permutated among path nodes*

Property 1 preserves the STI effect when the traffic flows are time-averaged sampled at any time resolution during raw data measurements. Property 2 makes the STI effect work when the data of parts of spatial nodes are absent. The above two properties preserve the STI effect when the flow tensor time series are sampled at any spatio/temproal resolution, which is of paramount importance for practical usages under imperfect measurements. By permutation/averaging operations, it also makes it possible to enrich the flow dataset without deprecating the STI effect.

Next, we propose what we call *FlowNN* to learn the representation of the flow-structured data in IP networks.

3 Flow Neural Network

As shown in Fig. 5, two key blocks build the frame of *FlowNN–Flowing Invariant coding* and *Flowing variant coding*.

Flowing Invariant Coding: This block constructs a Siamese net (Maron et al., 2020) to extract the shared evolution patterns among all the path nodes. The Siamese encoding layer receives the time series from all path nodes as input. Within this layer, a encoding net with shared weights is learnt to extract the common patterns shared in all received inputs. This reconstructs the evolution patterns of the APP source flows that drive the evolution processes at all nodes along the path. The source flow at APP/user side is beyond the observation of the monitors placed at network side, but the observed evolutions at each network node provide some transformed views of the original source flow. This leaves the chance to reconstruct the source pattern from these correlated views.

Flowing Variant Coding: This block encodes the distinctive transformation patterns at different nodes conditioned on the shared patterns from the source flows. Looking into how flows are transformed in Fig. 4, we can see that the actuation by congestion control, caching can be interpreted as a set of homotopy equivalent operations (e.g., shrinking/expanding) subject to flow conservation

[2]It follows exact flow conservation if dropped packets are compensated.

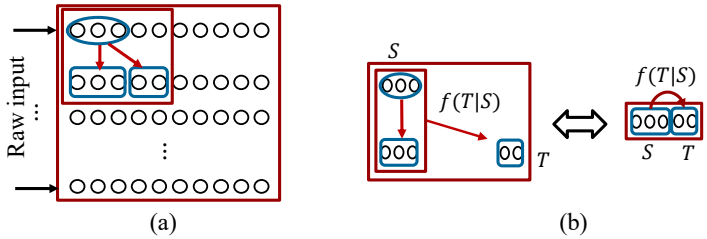


Figure 6: Induction operation with dynamic kernel size. (a) The induction kernel scans every two-row raw data from left to right to extract the paired induction effect; (b) Instead of solely using the source of predecessor node, spatial correlations within same time window are also introduced to better encode the state of S.

(Hereman & et al., 2005). This contrasts with the well-known equivarance/invariance effect by e.g., shift/scale in vision/audio tasks, but can be right captured by the proposed *induction operations* as follows.

As illustrated in Fig. 6, the induction is operated on a dynamic kernel, within which the correlations among the included items are extracted by the paired induction operator. Specifically, the induction operation infers the future of neighboring nodes from the past of its predecessor node by the conditional induction function $f(T|S)$. This technically implements the S-shaped spatial-temporal correlations manifested by flow-structured data.

The kernel size for the induction operation is a dynamic time window pivoted by the turning time point of the neighboring difference signal. For example, to construct the pivoted kernel of Fig. 4b, the sequences above zero are taken as the sources $\{S_1, S_2, \dots\}$, but the right followed sequences below zero are then taken as the targets $\{T_1, T_2, \dots\}$. The conditional induction function $f(T|S)$ can be approximated by a specialized contrastive induction nets as follows.

Contrastive Induction Coding: Given the sequence tuple (S, T), as shown in Fig. 6 (b), the contrastive induction nets learn to approximate the induction functions $f(T|S)$. In our model, we first use a RNN encoder with GRU activations (Chung et al., 2014) to produce the hidden state $\{h_i^t | t = 0, 1, \dots\}$ of sequence S. Conditioned on the outputs and context c of the encoder, a GRU decoder is used to produce the representations of the sequence T. While any encoder and decoder can be used so long as we can backpropagate through it. Inspired by Contrastive Predictive Coding (CPC) (Oord et al., 2019), we learn the mappings from S to T in a way that maximally preserves the mutual information of the source and target sequences defined as follows:

$$I(T; c) = \sum_{T,c} p(T, c) \log \frac{p(T|c)}{p(T)} \tag{1}$$

This enforces the model to learn the induction function rather than fitting an approximation function to map from the source to target. Analogy to CPC, we define the following induction loss to optimize the mutual information in Equation 1:

$$L_I = -\mathbb{E}\left[\log \frac{f_k(T_k, c)}{\sum_{T_k \in \mathcal{T}} f_k(T_k, c)}\right] \tag{2}$$

where $\mathcal{T}$ is all the possible realizations of T_k. $f_k(\cdot)$ is the density ratio function in CPC (more mathmatical work refers to (Oord et al., 2019)).

Connection to GNN/CNN/RNN: Vanilla GNN/CNN/RNN only models separate spatial/temporal correlations (i.e., spatial-spatial or temporal-temporal) in the raw data. In addition to the correlations within the same domain, *FlowNN* uniquely encodes the spatial-temporal correlation with full respect to the properties of flow-structured data.

Connection to CPC (Oord et al., 2019)/CMC (Tian et al., 2019): The predictions in *FlowNN* are inferred by customized induction operation derived from STI effect for flow-structured data. This differs our work from the predictions driven by the direct past of a time series in CPC. CMC models

the view-invariant feature representations by contrastive learning. *FlowNN* inputs multiple views as well, but encodes both the flowing-invariant and variant patterns from these correlated views.

4 EXPERIMENTS

We conduct the following regression tasks to validate the effectiveness of this work.

Dataset. In order to evaluate the effectiveness of our model, we conduct experiments on the publicly available flow dataset *NumFabric*[3] (Nagaraj et al., 2016; Lei et al., 2019) for datacenter network systems. The network topology includes 4 core nodes, 8 leaf nodes and 16 hosts organized in accordance with Fat-Tree network architecture (Al-Fares et al., 2008). We sampled the flow rates as well as the associated aggregated flow rates at each nodes of 50 flows at $1ms$ time resolution for total length $18432ms$. Each flow traverses a fixed routing path with 5 nodes. We found that the flow tensor time series of such length is enough to learn a stable flow pattern at subsecond level.

Baselines We compare the proposed FlowNN with the following baselines: 1) Seasonal Auto-Regression Integrated Moving Average (SARIMA) (Hyndman & Khandakar, 2008; Wang et al., 2006); 2) univariate Gated Recurrent Unit (uniGRU) (Chung et al., 2014); 3) multiple GRUs (multi-GRU); 4) GCN-GRU (Kipf & Welling, 2016; Yu et al., 2018). In particular, uniGRU uses vanilla GRU to encode and predict each time-series sequence stand-alone without any reference to the information from other spatial nodes. In contrast, multiGRU uses separate GRUs to encode each time-series sequence but predicts with Fully-connected Neural Network (FNN) by jointly concatenating the codings of sequences from all spatial nodes. Finally, GCN-GRU uses graph convolutional network (Kipf & Welling, 2016) and GRU to predict the spatial states of all nodes at each time, which are then fed to the same readout layer with FNN as multiGRU does.

Implementation Details. We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $1e^{-4}$. All hidden dimensions are set to 128 and layer size for GRUs is 3. We train on sampled time series window of length 256. A minibatch of 10 examples are used to draw the negative examples in the contrastive induction loss. We implemented the GCN with the DGL library (Wang et al., 2019). Similar to the doings in (Maron et al., 2020), we attempted to use separated GRUs with shared weights and MaxPooling among nodes to construct the Siamese encoding layer in *FlowNN*. Surprisingly, this does not outperform the direct encoding of a GRU with all raw node data as inputs. This stands to the reason that the raw data from other nodes conveys the direct information for next-step future realization of a node. The above GRU and MaxPooling operations actually introduce more distant information but less useful to the prediction of next-step realization. More details are provided in later experiments. Consequently, in the following experiments, we directly encode the flowing-invariant features by a GRU with all raw node data as inputs. The flowing-invariant features are then used to represent the hidden states of the source sequence of S. The initial embedding in *FlowNN* is performed by separated FNNs. We first pre-train the *FlowNN* with the contrastive induction loss until convergence and then fine-tune the model with MSE loss (Eq. 3).

Evaluation metrics. The following metrics are adopted for evaluating quality of prediction $\hat{y}$ with the ground truth y.

$$\text{Mean squared error:} \qquad MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

$$\text{Relative absolute error:} \qquad RAE = \frac{1}{n}\sum_{i=1}^{n} \mid \frac{y_i - \hat{y}_i}{y_i} \mid \tag{4}$$

$$\text{Correlation coefficient:} \qquad Corr = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \hat{\bar{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 \sum_{i=1}^{n}(\hat{y}_i - \hat{\bar{y}})^2}} \tag{5}$$

where $\bar{y}$ and $\hat{\bar{y}}$ are the mean values of y and $\hat{y}$.

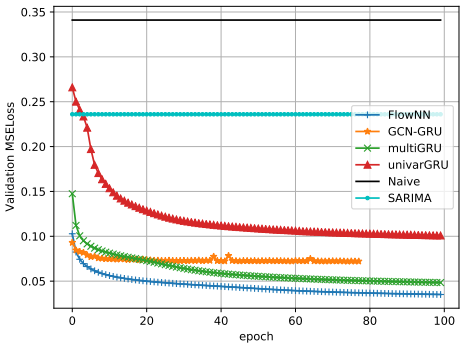


Figure 7: Validation MSELoss

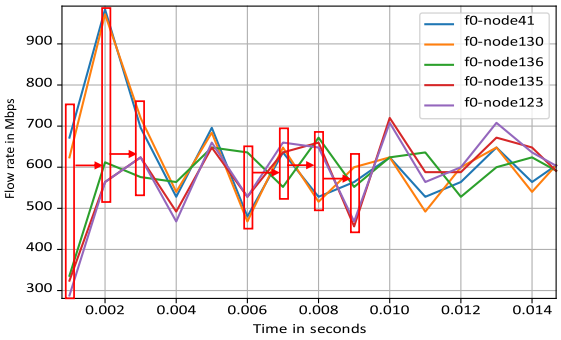


Figure 8: Data transformations

Table 1: Performance comparison of different methods

Method	MSE	RAE	Corr
uniGRU	0.09124	1.3506	0.9521
GCN-GRU	0.074	1.2574	0.9575
multiGRU	0.04086	0.9405	0.9788
FlowNN	**0.02723**	**0.84**	**0.9857**

4.1 Results for 1-Step Prediction Task

Fig. 7 shows the validation MSE loss across training epochs for the next-prediction task. *Naive* solution directly uses the previous observation as the next-step prediction. The results of *Naive* and SARIMA provide a good testament to the prediction difficulty for traditional non-AI solutions in millisecond-level network traffic prediction task. We can see that the finetuned SARIMA only shows a marginal improvement gain (30.8%) against a *Naive* prediction in this task. However, all deep learning based solutions achieve several times to tens times better accuracy. This shows the powerful advantages of the deep learning principle in difficult tasks.

Next, we exam the effectiveness of combing different deep learning components to do the learning tasks. The results of different models are compared in Table 1. uniGRU treats the flow tensor as a set of normal time series and improves 61.34% against SARIMA with the vanilla ability of GRU. Built upon the ability of single GRU, multiGRU predicts by concatenating the multiple GRU codings for all spatial nodes. This improves multiGRU against uniGRU by 55.22%. Instead of encoding the time series from each nodes by multiple independent GRUs, GCN-GRU applies graph convolution (Kipf & Welling, 2016) to exploit the spatial correlation by aggregating the attributes from all nodes. This achieves 18.89% gains against uniGRU, but is inferior to multiGRU (55.22%). Such findings discourage the spatial aggregation operation in GCN as the message propagation road by GCN deviates a lot from the truly S-shaped road for flow-structured data. Finally, the proposed *FlowNN* outperforms all the baselines and achieves 33.36% better accuracy than the best results from multiGRU.

Fig. 8 further explains why multiGRU, to certain extend, also works well for the next-step prediction task. As illustrated by the squared data in Fig. 8, the next-step realizations of all nodes can be directly synthesized by weighting the data at its previous time step. This is benefited from the STI effect, with which, the past states of predecessor nodes directly connect the near future realizations of successor nodes.

We show the comparisons of the predictions by different models against the ground truth in Fig. 9. We can observe that *FlowNN* and multiGRU capture the trend of ground truth accurately due to the

[3]The data can be generated by running the simulation code released in `https://knagaraj@bitbucket.org/knagaraj/numfabric.git`

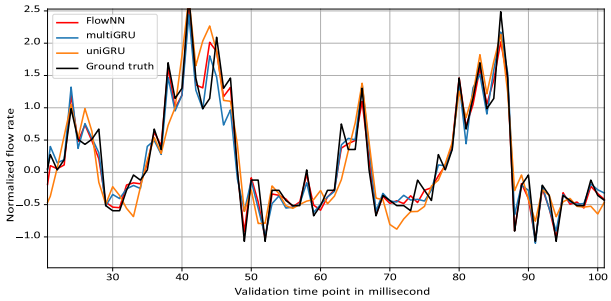


Figure 9: Normalized flow rate predicted by different models.

STI effect in the flow-structured raw data of the network traffic flows. With the induction operation, *FlowNN* shows the fastest response to the dynamic changes among the traffic flows without over-reliance on the far distant history as the vanilla GRU and its variants do.

5 Conclusion

In this paper, we experimentally disclosed the STI effect, a universally present property for flow-structured data. Accordingly, by exploiting the flowing invariance and variance of network traffic flows, we proposed *FlowNN* and the induction operation to learn the representations of flow-structured data. Results from prediction tasks demonstrate the state-of-the-art accuracy and extend the existing time resolution from second to millisecond, offering a good testament to the strength of our approach. Although derived from the IP network traffic flows, this work presents the initial innovation to use customized neural model to learn for the flow-structured data. We hope this will inspire more innovative models in this field in future.

References

Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 38(4):63–74, 2008.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, and Mateusz Malinowski. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS Deep Learning Workshop*, 2014.

Florin Ciucu and Jens Schmitt. Perspectives on network calculus: no free lunch, but still good value. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 311–322, 2012.

Florin Ciucu, Felix Poloczek, and Amr Rizk. Queue and loss distributions in finite-buffer queues. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):1–29, 2019.

Fabien Geyer and Steffen Bondorf. Deeptma: Predicting effective contention models for network calculus using graph neural networks. *IEEE INFOCOMM*, pp. 1009–1017, 2019.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

Willy Hereman and et al. Continuous and discrete homotopy operators and the computation of conservation laws. *Differential Equations with Symbolic Computation*, pp. 249–285, 2005.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

C. Hunt. *TCP/IP Network Administration*. OReilly and Associates: Sebastopol, CA., 1992.

Rob J Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–23, 2008.

Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *NIPS*, 2016.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *NIPS*, pp. 1–9, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, pp. 1–9, 2012.

Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. *IEEE INFOCOMM*, 2019.

Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. *ICML*, 2020.

Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. *ACM SIGCOMM*, pp. 188–201, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748v2*, 2019.

Krzysztof Rusek, Jos Surez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet: Leveraging graph neural networks for network modeling and optimization in sdn. *arXiv preprint arXiv:1910.01508*, pp. 1–12, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *ICML 2019 workshop on Self-Supervised Learning*, 2019.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Xiaozhe Wang, Kate Smith-Miles, and Rob J Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006.

Shihan Xiao, Dongdong He, and Zhibo Gong. Deep-q: Traffic-driven qos inference using deep generative network. *Proceedings of the 2018 Workshop on Network Meets AI & ML*, pp. 67–73, 2018.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *IJCAI*, 2018.

Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. *AAAI*, 2020.