Flow Neural Network and Flow-Structured Data Representation

Removed for anonymous review

Abstract

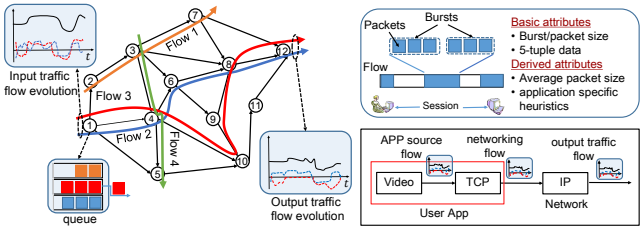
Traffic flows are the most basic components in a communication networking system. An accurate understanding of these flows is crucial for many downstream network applications. However, the high nonlinearity, randomness and complicated self similarity (Leland et al. 1994) of these traffic thwart extensive traditional analytical and learning models, particularly in high time resolution. In this paper, we experimentally discover the universal flow-structured data properties within network traffic flows. By exploiting the flowing invariance and variance of network traffic, we propose Contrastive Inductive Coding to learn their representations in a self-supervised manner. Results from both regression and classification tasks demonstrate the 2-3x better accuracy and extend the state-of-the-art time resolution from second to millisecond, offering a good testament to the strength of our approach.

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). The tremendous efforts on customizing neural structures to explore the data specific properties underpin the success of this powerful tool in these domains. This summarizes the mainstream learning work for the majority applications – 1) refine the data specific properties with the domain knowledge (e.g., regional correlation in images or local smoothness in time series), 2) customize a neural network to capture these properties with certain inductive bias (Peter W. Battaglia 2018). With the same design philosophy, we present a radically new data type – *flow-structured data*, as well as the learning model – ***F**low **N**eural **N**etwork (FlowNN)*. Although originated from the traffic flows in IP 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 telecommunications network is a network for sharing messages between nodes using a common communications protocol e.g., TCP/IP. 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.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) (b)

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

As illustrated in Fig. 1, 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.

An accurate understanding of these flows is crucial for many downstream network applications. With the growing network dynamics (in possibly subsecond order), concurrent flow scale ($>$ 1 million flows) and network size (tens to tens thousand of nodes), 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 (Note:add citations), have been invented, 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. However, the current machine/deep learning tools are designed for structure-specific data tasks

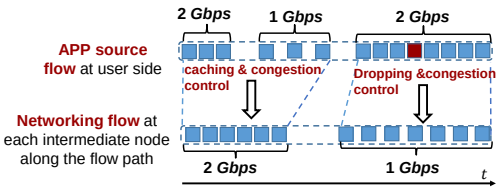


Figure 2: Data transformations by possible network actuation. **APP source flow**: original packet flow representing the user behaviors and message patterns of the source application; **Networking flow**: the packet flow actuated by all possible networking control actions, e.g., congestion control, caching,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

(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 lifespan 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 certain data transformation on the source traffic flows, creating multiple auto-correlated views of original spatio-temporal timeseries. This is exemplified in Fig. 2.

Manifested in high-dimensional tensor time series, network traffic data exhibits specific flow-structured data properties. As depicted in Fig. 3, the current data at node 4 is 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 *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. 3). 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 self-supervised learning model is proposed to represent the network traffic flows with high resolution.

Spatio-Temporal Induction Effect

In the path-constrained propagation process as above, 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 dis-

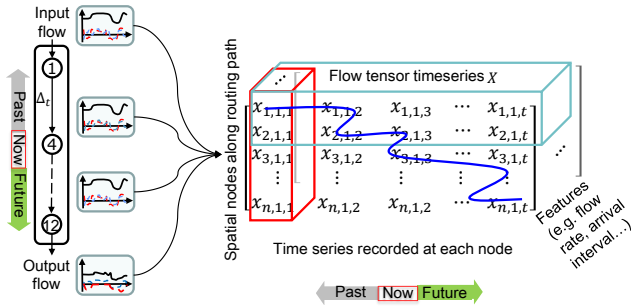


Figure 3: Flow-structured data in the form of flow tensor time series. The S-shaped curve characterizes the message propagation road within the the spatio-temporal domain.

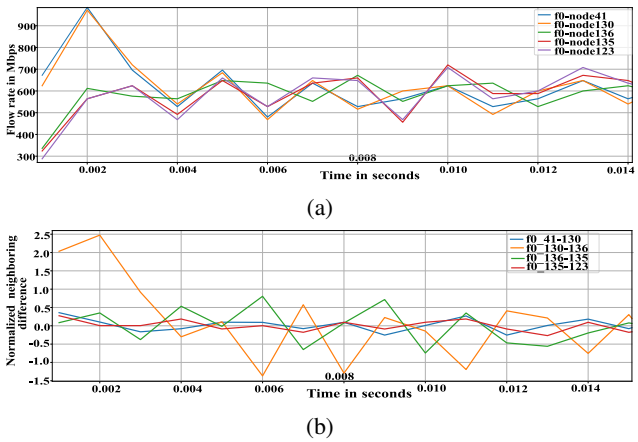


Figure 4: (better viewed in color) (a) traffic flow evolution; (b) traffic rate difference between neighbouring nodes.

close the **S**patio-**T**emporal **I**nduction (STI) effect as follows.

Definition 1. *Spatio-Temporal Induction defines the synchronized map process to product 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. 3) 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 drives the increase of 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 will become larger interchangeably than the other so that the flux is conserved among nodes. This not only experimentally confirms

[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.

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

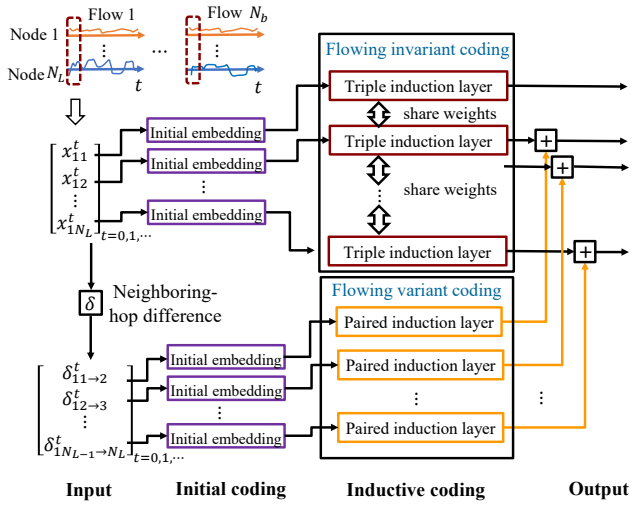


Figure 5: Framework of flow neural network.

the S-shaped spatio-temporal correlation present in the flow-structured data, but also further confines the correlation with the explicit (partial) flow conservation.

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*

The above two properties preserve the STI effect when the flow tensor time series are sampled at any rate/resolution, which is of paramount importance for practical usages under partial measurements.

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

Flow Neural Network

Two key building blocks frame the *FlowNN–Flowing Invariant coding* and *Flowing variant coding*.

Flowing Invariant Coding: This block constructs a Siamese net with the triple induction layer to extract the shared evolution patterns among all the path nodes. This reconstructs the evolution patterns of the APP source flows that drive the evolution processes at all the nodes of 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.

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 (Hereman and *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.

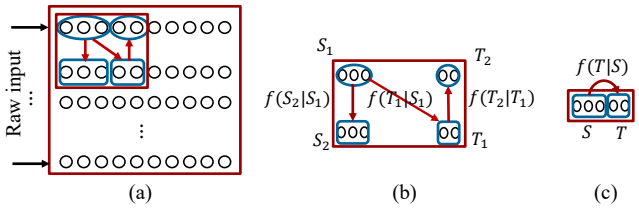


Figure 6: Induction operation with dynamic kernel size. (a) The induction kernel scans every two-row raw data from left to right to extract the paired induction effect; (b) 2D kernel with triple induction operation for raw signal; (c) 1D kernel with paired induction operation for the neighboring-hop difference signal.

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 triple/paried induction operations. Specifically, the triple induction operation 1) connects the spatial attributes of two neighboring nodes at the same time window by $f(S_2|S_1)$; 2) infers the future of neighboring nodes by the past of its predecessor node by $f(T_1|S_1)$; and then 3) infers the future of a node from the predicted future of its predecessor node by $f(T_2|T_1)$. This technically implements the S-shaped spatial-temporal correlations manifested by flow-structured data. Likewise, the paired induction operation in Fig. 6 (c) builds the structural relations for the difference signals.

Flowing Variant Coding: This block encodes the distinctive transformation patterns at different nodes after the shared ingredients are removed from the source flows. The interchangeably turning behaviors within the neighboring-hop difference time series can be extracted by the 1D kernel with the paired induction operation in the form of $f(T|S)$.

The kernel size is a dynamic time window pivoted by the turning time point of the difference signal. For example, in the pivoted kernel of Fig. 4b, the sequences above zero are taken as the sources $\{S_1, S_2\}$, but the followed sequences below zero are then taken as the targets $\{T_1, T_2\}$. The correlations among the source and target sequences are formulated by conditional kernel functions $f(S_2|S_1)$, $f(T_1|S_1)$ and $f(T_1|T_2)$, which can be approximated by a specialized contrastive induction nets as follows.

Contrastive Induction Coding: Given the sequence tuple (S_1, S_2, T_1, T_2), 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 of each element in sequence S. Conditioned on the outputs 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 CPC–Contrastive Predictive Coding (van den Oord, Li, and Vinyals 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:

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

Analogy to CPC, we define the following induction loss to optimize the mutual information in Equation 1:

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

The loss is the categorical cross-entropy of classifying the positive sample correctly, with $\frac{f_k}{\sum_T f_k}$ being the prediction of the model. The probability that sequence sample T are drawn from the conditional distribution $p(T|S)$ rather than the data distribution $p(T)$ can be derived as follows:

$$\begin{aligned} p(d=i|T,S) &= \frac{p(T|S)\prod_{l\neq i} p(T)}{\sum_{j=1}^{N} p(T|S)\prod_{l\neq j} p(T)} \\ &= \frac{\frac{p(T|S)}{p(T)}}{\sum_{j=1}^{N} \frac{p(T|S)}{p(T)}} \end{aligned} \tag{3}$$

Consequently, optimizing this loss will result in $f_k(T,c)$ estimating the density ratio.

Connection to GNN/CNN/RNN: reduces to RNN when the kernel size is 1

Connection to CPC (van den Oord, Li, and Vinyals 2019)/CMC (Tian, Krishnan, and Isola 2019):

Inductive NCE loss: InduNCE 1. segment the whole flow tensor time series with sampled windows of length 1024
2. coding per neighboring pair with the contrastive inductive coding upon the coding results of previous pair coding until the last neighbouring pair, using same seq2seq net with shared weights. new induction operation only update the codes positioned for target sequences, leading to relative encoding. Therefore, the core neural network only consists of one seq2seq model with shared weights for all pair induction. this enable the model to process different flows with varying path length by recursively invoking same triangular (seq2seq+fully connected net) model at each iteration.
3. calculate contrastive inductive loss and Siamese loss

Experiments

aaaa

Implementation Details.

Dataset.

Evaluation metrics.

1-Step Prediction Task

bbb

Multi-Step Prediction Task

ccccc

Ablation Study

dddd

Conclusion

eeee

Thank you for reading these instructions carefully.

References

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS Deep Learning Workshop* .

Hereman, W.; and *et al.* 2005. Continuous and Discrete Homotopy Operators and the Computation of Conservation Laws. *Differential Equations with Symbolic Computation* 249–285.

Leland, W. E.; Taqqu, M. S.; Willinger, W.; and Wilson, D. V. 1994. On the Self-similar Nature of Ethernet Traffic. *arXiv preprint arXiv:2004.08378* 2(1): 1–15.

Peter W. Battaglia, t. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* .

Tian, Y.; Krishnan, D.; and Isola, P. 2019. Contrastive Multiview Coding. *ICML 2019 workshop on Self-Supervised Learning* .

van den Oord, A.; Li, Y.; and Vinyals, O. 2019. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748v2* .