# IGNNITION: Fast Prototyping of Graph Neural Networks for Communication Networks

David Pujol-Perich*, José Suárez-Varela*, Miquel Ferriol-Galmés*, Bo Wu†,

Shihan Xiao†, Xiangle Cheng†, Albert Cabellos-Aparicio*, Pere Barlet-Ros*

contactus@bnn.upc.edu

*Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain

†Network Technology Lab., Huawei Technologies Co., Ltd.

## ABSTRACT

*Graph Neural Networks* (GNN) have recently exploded in the Machine Learning area as a novel technique for modeling graph-structured data. This makes them especially suitable for applications in the networking field, as communication networks inherently comprise graphs at many levels (e.g., topology, routing, user connections). In this demo, we will present *IGNNITION*, an open-source framework for fast prototyping of GNNs applied to communication networks[1]. This framework is especially designed for network engineers and/or researchers with limited background on neural network programming. IGNNITION comprises a set of tools and functionalities that eases and accelerates the whole implementation process, from the design of a GNN model, to its training, evaluation, debugging, and integration into larger network applications. In the demo, we will show how a user can implement a complex GNN model applied to network performance modeling (*RouteNet*), following three simple steps.

## 1 INTRODUCTION

Recent years have seen the great emergence of Graph Neural Networks (GNN) [1] onto the Machine Learning (ML) scene. Similarly to Convolutional Neural Networks for spatially arranged data (e.g., images) or Recurrent Neural Networks for sequences (e.g., Natural Language Processing), GNN is a novel Deep Learning tool especially conceived for modeling graph-structured data (i.e., relational information). Nowadays, we have witnessed a plethora of groundbreaking GNN-based applications in fields where data is inherently represented as graphs (e.g., molecules in chemistry, gravitational systems in physics, proteins in biology, or user-item interactions in recommendation systems) [12].

In the last few years, the networking community has shown growing interest in the potential applications of GNN to complex networking problems, *as graphs are a fundamental data type present at many levels in networks* (e.g., topology, routing, flow interactions, user connections). To date, GNN has been applied to a broad variety of use cases in communication networks [4], such as routing optimization in wired networks [6, 9], resource allocation in wireless networks [5, 10], Virtual Network Function placement [8], or job scheduling in data center networks [7]; while there is arguably a long way ahead to investigate many other potential applications –involving graphs– yet unexplored.

[1]https://ignnition.net

*SIGCOMM '21 Demos and Posters, August 23–27, 2021, Virtual Event, USA*
2021. ACM ISBN …$
https://doi.org/

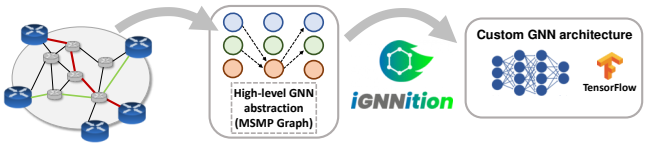


**Figure 1: Overview of IGNNITION.**

Nowadays, creating a GNN-based solution for networking is a complex process that requires to unify a deep knowledge and expertise in ML and networking. Standard GNN models (e.g., Graph Attention Networks [11]) are not directly suitable for most network applications. Often, it is needed to make an analysis on the target network scenario, identify the elements involved and their relationships, and implement a custom GNN architecture adapted to the problem (e.g., heterogenenous graphs with various relation types [5, 6, 9, 10]). In this context, implementing a non-standard GNN model is a cumbersome task that entails complex mathematical formulation and dealing with purpose-specific ML libraries, such as TensorFlow or PyTorch. This represents a critical entry barrier for network engineers that aim to explore the potential of GNN in their particular problems, but lack the needed ML expertise to implement these models.

In this demo, we introduce IGNNITION, an open-source framework for fast prototyping of GNNs (see Fig. 1). This framework mainly targets networking experts with little background in neural network programming. IGNNITION is developed by network engineers, following a *top-down approach*, from the requirements of network applications to the intricacies behind the implementations of their corresponding GNN models.

With IGNNITION, users can easily design their own GNN models — including non-standard GNN architectures – via an intuitive high-level abstraction called the *Multi-stage Message Passing (MSMP) graph*. This makes them oblivious to the mathematical formulation and the underlying tensor-based code behind their model implementations.

As a result, a typical network engineer and/or researcher should be able to produce a functional GNN prototype tailored to his/her specific problem without prior knowledge on ML programming libraries (e.g., TensorFlow, PyTorch).

The final publication is available at ACM via http://dx.doi.org/10.1145/3472716.3472853

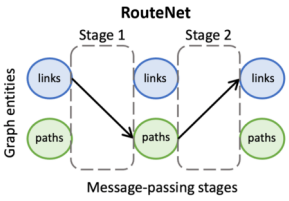


**Figure 2: MSMP graph of RouteNet [9].**

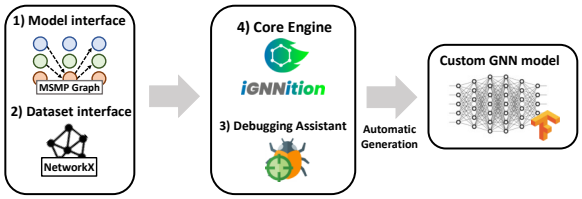


**Figure 3: High-level architecture of IGNNITION.**

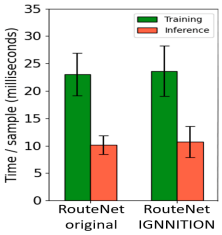


**Figure 4: Evaluation of execution cost.**

## 2 MSMP GRAPH ABSTRACTION

GNN models applied to networking typically require to define heterogeneous graphs to describe the input network scenarios, including various element types (e.g., forwarding devices, links, paths), and relations between them sequentially arranged [5, 6, 9, 10]. This involves a set of features not supported by standard GNN architectures [12]. The goal of IGNNITION is to provide a user-friendly interface to support this type of models.

This leads to the definition of the *MSMP graph abstraction*, a novel high-level representation of GNNs that covers the common requirements of state-of-the-art GNN models applied to networking. With MSMP graphs, users can define their custom GNN architectures through a visual graph representation, abstracting them from the complex mathematical formulation behind their designs. As an example, Figure 2 shows the MSMP graph of RouteNet [9]. We refer the reader to [3] for a detailed description on MSMP graphs.

At the same time, the MSMP graph abstraction covers a broad definition of GNN, which provides flexibility to implement a broad variety of existing GNN architectures, and combine individual components of them to create new custom designs (e.g., Message Passing Neural Networks, Graph Convolutional Networks, Gated Neural Networks, Graph Attention Networks, Graph Recurrent Networks, and many more) [12].

## 3 IGNNITION OVERVIEW

Figure 3 depicts an overview of the internal architecture of IGNNITION. The framework comprises four main modules:

1) The *Model interface* implements the MSMP graph abstraction described in Section 2. With this interface, users can easily describe the MSMP graph of their GNN models –like the one of Fig. 2– via a simple YAML file.

2) The *Dataset interface* is based on the well-known NetworkX library. This Python library implements a plethora of functions that automatize the definition of graphs from datasets, as well as offering advanced tools for visualizing the data. This enables network engineers to easily feed their GNN models with data in various formats (e.g., network monitoring logs) and –more importantly– to remain completely oblivious to the internal TensorFlow data pipeline, which requires to deal with complex tensor definitions.

3) The *Debugging assistant* incorporates advanced error-checking mechanisms that guide the user through the entire GNN design process, as we consider this a key aspect to achieve a good user experience. Moreover, it produces an interactive visual representation of implemented GNN models to help users understand the internal NN architecture and ease troubleshooting.

4) Lastly, the *Core engine* implements the main logic behind the framework. Once the GNN model is described via the MSMP graph (Fig. 2) and the dataset is adapted with NetworkX, the core engine automatically generates an efficient implementation of the GNN model in TensorFlow. The resulting implementations of IGNNITION have similar performance to models directly coded by experts in TensorFlow. As an example, Figure 4 shows a comparison of execution time (during training and inference) between the original implementation of RouteNet in TensorFlow [9] and the equivalent model implemented with IGNNITION.

## 4 DEMO SCRIPT

We provide a short video outlining the content of the demo[2]. We will show how a user can implement RouteNet [9] with IGNNITION, following three simple steps. Participants will be able to see the results in an interactive dashboard and a visual representation showing the model architecture.

Demo users will also have access to a quick start tutorial on how to design a simple GNN model that computes the shortest path between two nodes in a network[3], and all of them will be invited to participate in the Graph Neural Networking challenge [2].

## ACKNOWLEDGEMENTS

This open-source project has received funding from the European Union's Horizon 2020 research and innovation programme within the framework of the NGI-POINTER Project funded under grant agreement No. 871528. This article reflects only the authors' view; the EC is not responsible for any use that may be made of the information it contains. The work was also supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE) and the Catalan Institution for Research and Advanced Studies (ICREA).

## REFERENCES

[1] Peter W. Battaglia et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).
[2] BNN-UPC. 2021. The Graph Neural Networking challenge. https://bnn.upc.edu/challenge/. (2021). Accessed on 23/06/2021.
[3] BNN-UPC. 2021. Multi-stage Message Passing (MSMP) graphs. https://ignnition.net/doc/model_description/#multi-stage-message-passing. (2021). Accessed on 23/06/2021.
[4] BNN-UPC. 2021. Must-read papers on GNN for communication networks. https://github.com/BNN-UPC/GNNPapersCommNets. (2021). Accessed on 08/06/2021.
[5] Mark Eisen and Alejandro Ribeiro. 2020. Optimal wireless resource allocation with random edge graph neural networks. *IEEE Transactions on Signal Processing* 68 (2020), 2977–2991.

[2] https://youtu.be/H86JfOsMdss
[3] https://ignnition.net/doc/quick_tutorial

[6] Fabien Geyer and Georg Carle. 2018. Learning and generating distributed routing protocols using graph-based deep learning. In *Proceedings of the ACM SIGCOMM BigDAMA Workshop*. 40–45.
[7] Hongzi Mao et al. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of ACM SIGCOMM*. 270–288.
[8] Pham Tran Anh Quang, Yassine Hadjadj-Aoul, and Abdelkader Outtagarts. 2019. A deep reinforcement learning approach for VNF Forwarding Graph Embedding. *IEEE Transactions on Network and Service Management* 16, 4 (2019), 1318–1331.
[9] Krzysztof Rusek et al. 2019. Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN. In *Proceedings of ACM SOSR*. 140–151.
[10] Yifei Shen, Yuanming Shi, Jun Zhang, and Khaled B Letaief. 2020. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 101–115.
[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *Proceedings of ICLR* (2018).
[12] Jie Zhou et al. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.