Deep-learning assisted Cross-Layer Routing in Multi-hop Wireless Network

Paulo Alexandre Regis*, Suman Bhunia†, Amar Nath Patra‡, and Shamik Sengupta§
* Department of Computer Science, Southeastern Louisiana University–Hammond, Louisiana
Email: pregis@southeastern.edu
† Department of Computer Science and Software Engineering, Miami University–Oxford, Ohio
Email: bhunias@miamioh.edu
‡ School of Computing and Information Sciences, Radford University–Radford, Virginia
Email: apatra@radford.edu
§ Department of Computer Science and Engineering, University of Nevada–Reno, Nevada
Email: ssengupta@unr.edu

***Abstract*—Wireless *ad hoc* networks rely on exchanging information among the nodes to find a feasible route between source and destination. Routing protocols require periodically flooding the entire system with the updated state of the network. Routing protocols define the rules and semantics of how this information is spread throughout the network and can be classified into proactive and reactive protocols, both of which eventually sends out control packets to the network. However, with the increase of the scale of the network, the amount of packets exchanged uses resources that could be otherwise used for application messages. In this paper, we explore the use of machine learning techniques to restrict the flooding radius of control packets. We define the problem objective as a regression problem and use neural networks to model it. The model can be used in networks that share similar attributes, suggesting that transfer learning can be used to re-purpose the initial model. The results show that using machine learning reduces the amount of resources used to maintain the network state up to date, which could be subsequently used to improve the overall performance of the networked system. We anticipate machine learning to be the starting point for more sophisticated models of cross-layer routing mechanisms.**

***Index Terms*—internet of things, ad hoc network, machine learning**

I. Introduction

The continuous advance in wireless *ad hoc* networks enables a variety of new application possibilities. Unmanned Aerial Vehicle networks can perform autonomous tasks such as monitoring and surveillance of areas of interest without the need to deploy human resources [1]. First responders may also benefit of such application, where UAVs establish a temporary backbone network where infrastructure is not available, such as in areas affected by natural disasters [2]. The dynamic aspect of such networks coupled with the versatility of UAVs can facilitate applications that before would have been impossible.

However, these networks also face complex challenges. Due to the mobility aspect of the nodes comprising the system, the topology of the network cannot be fixed. Communication paths in wireless *ad hoc* networks must react and adapt due to the frequent changes in topology of the network. This task is commonly achieved by the use of decentralized protocols where there is no single entity that manages the network routing. Decentralized protocols rely on the constant exchange of information among the participating nodes in order to maintain an accurate view of the state of the network. However, the frequent control messages and the time needed to compute the new routes can disturb the service that the network provides. In other words, the resources that could have been used to pass actual information (sensor data, voice, image) is used to maintain the system itself. Moreover, the amount of control packets being exchanged takes up resources that could be used for transmitting application messages as the scale of the network increases in size.

By itself, the field of machine learning has expanded dramatically in recent years. From support vector machines, random forests, to neural networks, the subject is in active development with new theoretical models as well as applications. Machine learning can reduce the complexity of a system since it can be considered a black box. This is achievable due to the models being trained in accurate and curated data. Although machine learning in networking applications is not a novel application *per se*, there is still huge potential to its usage [3]. In this paper, we explore the use of machine learning techniques to restrict the flooding radius of control packets.

We perform a preliminary study on machine learning assisted routing in wireless networks. We define our objective as a regression problem and use neural networks to model it. We found that the model can be used in networks that share similar attributes, suggesting that transfer learning can be used to re-purpose the initial model. Our results demonstrate that using machine learning reduces the amount of resources used to maintain the network state up to date, which subsequently can be allocated to actual data traffic, improving the overall performance of the networked system. We anticipate machine learning to be the starting point for more sophisticated models of cross-layer routing methods. For example, complex network systems could leverage this technique as well, such as in multi-radio multi-channel heterogeneous networks, to decide with radio interface and channel to use. Furthermore, cross-layer optimization is a major area of study in networking, and a

well-defined machine learning model can be relevant for the development of such area. The contributions of these paper can be summarized as follows:

- Preliminary study of machine learning-assisted routing in wireless *ad hoc* networks.
- Neural network model for cross-layer routing protocol.
- Analysis of the impact of local neighborhood radius.

The remainder of this paper is organized as follows. Section II provides a literature review related to the field. Section III describes the system model and assumptions. Then, Section IV illustrates the proposed system. Section V discusses the system system performance. Finally, Section VI concludes the paper and provides future directions.

II. Background

The versatility of mobile network architectures has caught the attention of investors and researchers. The complexity of such systems make the task of monitoring and managing the network components and resources a challenging task [3], [4], [5]. Machine learning solutions have resonated these issues, from anomaly detection and intrusion prevention to medium access and channel selection [6], [7], as well as networked devices to support the application of machine learning [8], [9], [10], [11]. Machine learning has been successfully applied to solve complex problems otherwise thought to be impossible for a computer to perform. Some of the leading applications of deep learning are natural language processing [12] and computer vision [13], and, more recently, as an artificially intelligent player of the Go board game, defeating top human experts thought to be unbeatable by a machine a few years ago [14]. Researchers are further investigating the applicability of deep learning in networking problems, acknowledging its importance in the field [15].

Due to the vast amount of data generated by mobile network devices, coupled with the variety of types of data, deep learning can be an efficient tool in the next generation of mobile networks (5G). It has the potential to distill relationship in highly dimensional data at each layer of the neural network, which are extremely difficult for traditional machine learning methods [16]. However, the performance of deep learning models is directly affected by the amount and quality of the data, which given the increasingly more heterogeneous forms of data, makes it a valuable choice [17], [18], [19]. In addition, the advances in specialized machine learning hardware further enables deep learning training f and inference operations on edge and even IoT devices [10], [11]. However, current deep learning solutions to networking problems are spread through a variety of research areas, and cross-layer optimizations involving deep learning is an open research area.

One aspect of networking in which deep learning has been applied is in routing in mobile networks. In [20], [21], authors used a deep belief network to decide the next hop in routing decisions, achieving 95% accuracy compared to the Open Shortest Path First protocol. However, their system requires complete knowledge of the network state and its use is limited by only wired networks where the topology is more stable. The work in [22] makes use of Hopfield neural networks to improve the performance of and survivability of mobile *ad hoc* networks. However, they system works on top of the AODV routing protocol, thus, cannot be used to optimize other protocols, and also requires control packets flooding. In [23], authors use neural network to classify the node degree, given information from the routing nodes. The results are then used for route generation using the Viterbi algorithm. Authors in [24] modeled the network as a graph, and designed the custom Graph-Query neural network to address distributed routing problem. Although it tackles the distributed routing problem, it still requires message flooding between nodes. In [25], authors combined the concept of software-defined networks with neural networks to tackle the routing problem in knowledge-defined networks. They used a deep deterministic policy gradient algorithm based on reinforcement learning that takes traffic conditions as input and introduce Quality-of-Service into the reward function.

III. System model

In our network model, we assume that lower layers of the network stack have access to information from the upper layers. In other words, when deciding the next-hop neighbor to send a packet, the node has access to the type of application data being sent. For our purposes, we assume that the nodes are running a video streaming application. Common video compression algorithms generate different types of picture frames; some contain more information and are, therefore, larger, while others contain less information and are smaller in byte size. We reproduce this behavior through a network trace file, commonly utilized in network simulations [26]. We also adopt as edge weight (link cost) the total number of data flows going through the destination node of the link in a directed graph. In this study, we also assumed nodes were static when generating the data because, at each transmission, the samples collected reflect a snapshot of the network at that instant, regardless of the movement of the nodes. Each node in the network starts with a predefined amount of energy, which the simulation decreases at each step by the amount of energy used to transmit the packet (smaller packets need less energy than larger ones).

A. Neighborhood

One of the goals of this work is to limit the amount of control messages that need to be disseminated in the network. Thus, we constrain the field-of-view of each node to a fixed number of hops away. In Figure 1 we can easily visualize it. By fixing the number of hops that a node can obtain information from, we also limit the flooding of the network with control messages. It is important to note that a node can only be in one of the neighborhood sets, for instance it cannot be both in the 1-hop neighborhood as well as in the 2-hop neighborhood. Otherwise, samples would be counting the state of a node more than once.

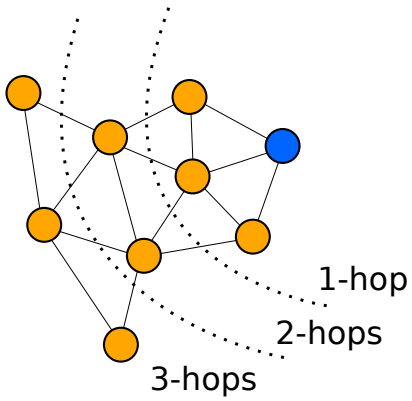

Fig. 1. Neighborhood radius.

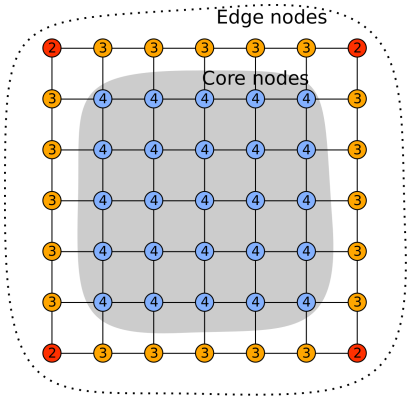

Fig. 2. Grid topology for training data collection.

B. Data description

We define a few generic features that can expand depending on the neighborhood radius. We also use application layer information in this cross-layer approach, namely the type of video frame being transmitted. The features used in this study are as follows.

- **frame type**: common video streaming applications codecs use different types of picture frames in order to reduce the amount of bytes to transmit. In our case, we use frame types of I P and B.
- **frame size**: each type of frame has different average byte size. I-frames are usually larger than P-frames, which in turn contains more information than I-frames. However, frames of the same type do not necessarily have the same size, thus, we select it as a feature as well as the frame type.
- **size h-hop neighborhood =** S_h: the total number of neighbors in the h-hop neighborhood.
- **average current charge (h-hop)**: for each neighborhood (1 to h) calculate the current charge of the power supply per node: $\frac{\sum_1^{S_h} cc_i^h}{S_h}$, where cc_i^h is the remaining energy fraction of node i in the h^{th}-hop neighborhood.
- **average energy fraction (h-hop)**: for each neighborhood (1 to h) calculate the average remaining energy fraction per node: $\frac{\sum_1^{S_h} ef_i^h}{S_h}$, where ef_i^h is the remaining energy fraction of node i in the h^{th}-hop neighborhood.

The features are purposely designed to be generic enough to accommodate any number of nodes to be present in each neighborhood. Thus, making this solution adaptable to any network topology, an asset when considering the volatile topology change that may occur in mobile *ad hoc* networks. The number of features increases linearly with the radius of the neighborhood. The attributes value for each node are: initial energy ($150J$), up current when transmitting ($1.7A$), down current when idle ($0.8A$).

1) Training data: We obtain the ground truth data from a simple grid network topology, as illustrated by Figure 2, with 7 nodes per side (7x7 grid). This will give an approximate similar number of edge nodes as well as inner ones. The ground truth values are calculated with complete knowledge of the graph, using Dijkstra to compute the route for each packet, and updating the weights after each route computation. We collected a total of 163474 training samples by varying the number of traffic flows in the network (1 to 5).

2) Testing data: We generated two distinct testing datasets, both of them randomly generated using a larger scale graph with 1000 nodes. First, we replicate the same node degree distribution as the training dataset, the frequency count of each degree is listed in Table I. We generated 10 random graphs, resulting in 9025384 samples from a network with similar node degree distribution.

TABLE I
NODE DEGREE DISTRIBUTION.

Degree	Distribution (%)
2	8.2%
3	41.0%
4	82.0%

The second test dataset was generated using the Erds-Rnyi graph model (also known as binomial: $G_{n,p}$), where n is the number of nodes and p is the probability of edge creation. This type of graph does not yield the same degree distribution since nodes are not limited to 4 neighbors. We generated 5128780 samples from 10 randomly generated graphs.

IV. Deep learning assisted routing

Our proposed system for a generic routing mechanism leverages from machine learning models used in regression problems, instead of using it as a classifier. The input data of the ML model is an aggregated set of values from all of the neighborhood set, allowing the input data to grow more complex as the neighborhood radius increases, but not with the number of neighbors in each neighborhood. The output of the model is then used as a multiplier, which will become the index of a sorted list of the 1-hop neighborhood. An overview of the system can be seen in Figure 3, where $|N| = S_1$ (the size of the neighborhood). The result of the multiplication is then rounded to the nearest integer, which is the index used to find the next-hop neighbor to send the packet.

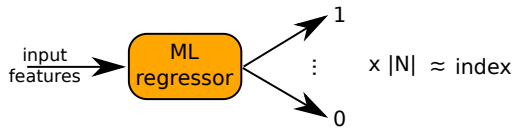

Fig. 3. Role of regression in the system.

In our study, we used a sequential neural network architecture with two densely connected layers containing 64 nodes each and the rectified linear unit (ReLU) activation function, and finally, the last layer outputs the index weight ($[0, 1]$). The model was implemented, trained, and tested using TensorFlow and Keras [27].

V. Results and Discussion

To evaluate the proposed system we compare two distinct machine learning approaches: deep learning and support vector regression. The neural network that comprises the deep learning model is composed of two densely connected layers, each with 64 nodes, and a final single-node layer that outputs the weight. The network is trained over 100 epochs using tensorflow [27]. In Figure 4 we can see the mean squared error (used as the loss function), and the mean absolute error, throughout the training phase. Only the first 30 epochs are shown for better visualization. We compare the neural network to a support vector regression model with radial basis function as the kernel function, ϵ value of 0.1, and tolerance of 0.001, implemented using scikit-learn [28]. The target feature of the system, for both ML models, is the index weight of the sorted list of the 1-hop neighborhood set, according to the remaining energy left.

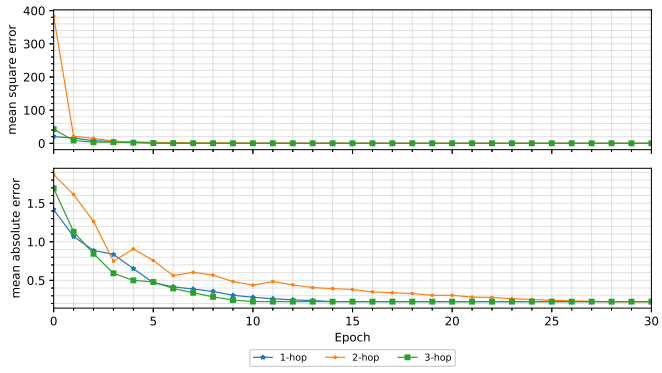

Fig. 4. Training phase metrics.

We compare both solutions using two distinct datasets as previously described: one with similar node degree distribution as the training dataset, and another with completely random degree distribution. In Table II we compare the performance of the models in the test data with similar distribution. Both the SVR and DL models performed relatively the same, with a slightly better outcome from the DL (Deep Learning) model. However, the SVR models require a much longer time to perform the prediction operations, thus, the deep learning approach is more adequate for use in production.

Table III on the other hand lists the performance of the ML models using the binomial random graph model. We can see that for the 3-hop neighborhood case, the DL model produced a much larger error than the SVR (Support Vector Regression) model. This can indicate that during the training phase the model encountered a local optimal solution and was not able to escape. However, the same DL model slightly outperformed the SVR when the radius was 1 and 2 hops. Overall, DL achieved a better metrics than SVR.

Given that both machine learning models were trained on a small dataset limited to 49 nodes in a single topology while tested on a variety of different topologies, the proposed scheme could be potentially improved by updating the model with new data. With new hardware dedicated and optimized for deep learning operations, the model would not necessarily require re-training from scratch.

TABLE II
Comparison with similar degree distribution.

Neighborhood radius	ML model	MSE	MAE
1-hop	DL	0.0717	0.2265
	SVR	0.0805	0.2415
2-hops	DL	0.0706	0.2240
	SVR	0.0760	0.2379
3-hops	DL	0.0583	0.2023
	SVR	0.0772	0.2416

TABLE III
Comparison with random degree distribution.

Neighborhood radius	ML model	MSE	MAE
1-hop	DL	0.1368	0.3322
	SVR	0.1382	0.3372
2-hops	DL	0.1368	0.3302
	SVR	0.1380	0.3401
3-hops	DL	4.6221	2.0064
	SVR	0.1387	0.3417

A. Limitations

Currently, the deep learning model performs relatively well when predicting the index multiplier weight. On the other hand, this approach might not yield the best results when considering the ground truth for the next-hop neighbor. However, there is a need to measure its effects applied to a networked system in order to identify potential benefits, of gain insights on where it can be further improved.

VI. Conclusion and Future Work

In this paper, we performed an exploratory analysis of machine learning routing models in wireless *ad hoc* networks with cross-layer architecture in mind. We trained a machine learning model using deep learning neural network in order to assist the routing of packets in a wireless *ad hoc* network. The model applied regression to determine the next-hop neighbor to which forward packets to. The model can also be generalized to any network topology. Preliminary results show

that our generic approach can be used in such networks. By limiting the amount of control packets flooding the network it can potentially enable large scale systems to be deployed, since control packets are limited to a bounded region surrounding each node.

Deep learning assisted networking is a relatively new research area, especially in the routing layer of the network stack. There are several different situation we envision that machine learning can assist in networking tasks. In the future, we want to investigate the need to retrain the routing model in mobile networks, since the mobility can severely affect the performance of such systems. Another research opportunity lies on the security aspect of this approach with many unanswered questions. This application of deep learning is still in its infancy and is worth investigating.

REFERENCES

[1] B. Galkin, J. Kibilda, and L. A. DaSilva, "Coverage analysis for low-altitude uav networks in urban environments," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[2] P. A. Regis, A. N. Patra, and S. Sengupta, "Unmanned aerial vehicles positioning scheme for first-responders in a dynamic area of interest," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, IEEE, 2018.

[3] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[4] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, "Big data-driven optimization for mobile networks toward 5g," *IEEE Network*, vol. 30, pp. 44–51, 1 2016.

[5] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, pp. 98–105, 4 2017.

[6] D. D. Nguyen, H. X. Nguyen, and L. B. White, "Reinforcement learning with network-assisted feedback for heterogeneous rat selection," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 6062–6076, 9 2017.

[7] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, pp. 343–357, 1 2016.

[8] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, (Boston, MA), pp. 629–647, USENIX Association, 2017.

[9] W. Xiao, J. Xue, Y. Miao, Z. Li, C. Chen, M. Wu, W. Li, and L. Zhou, "Tux^2: Distributed graph computation for machine learning," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, (Boston, MA), pp. 669–682, USENIX Association, 2017.

[10] "Edge-tpu." https://cloud.google.com/edge-tpu.

[11] "Coral." https://coral.withgoogle.com.

[12] R. Socher, Y. Bengio, and C. D. Manning, "Deep learning for nlp (without magic)," in *Tutorial Abstracts of ACL 2012*, pp. 5–5, Association for Computational Linguistics, 2012.

[13] C. Zhang, P. Zhou, C. Li, and L. Liu, "A convolutional neural network for leaves recognition using data augmentation," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 2143–2150, IEEE, 2015.

[14] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[15] H. Zhu, Y. Zhang, M. Li, A. Ashok, and K. Ota, "Exploring deep learning for efficient and reliable mobile sensing," *IEEE Network*, vol. 32, pp. 6–7, 7 2018.

[16] M. A. Alsheikh, D. Niyato, S. Lin, H. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE Network*, vol. 30, pp. 22–29, 5 2016.

[17] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, pp. 92–99, 3 2018.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[20] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, pp. 1946–1960, 11 2017.

[21] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A tensor based deep learning technique for intelligent packet routing," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, 12 2017.

[22] H. Yang, Z. Li, and Z. Liu, "Neural networks for manet aodv: an optimization approach," *Cluster Computing*, vol. 20, pp. 3369–3377, 12 2017.

[23] Y. Lee, "Classification of node degree based on deep learning and routing method applied for virtual route assignment," *Ad Hoc Networks*, vol. 58, pp. 70 – 85, 2017. Hybrid Wireless Ad Hoc Networks.

[24] F. Geyer and G. Carle, "Learning and generating distributed routing protocols using graph-based deep learning," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, Big-DAMA '18, (New York, NY, USA), pp. 40–45, ACM, 2018.

[25] T. A. Q. Pham, Y. Hadjadj-Aoul, and A. Outtagarts, "Deep reinforcement learning based qos-aware routing in knowledge-defined networking," in *Quality, Reliability, Security and Robustness in Heterogeneous Systems* (T. Q. Duong, N.-S. Vo, and V. C. Phan, eds.), (Cham), pp. 14–26, Springer International Publishing, 2019.

[26] F. H. Fitzek and M. Reisslein, "Mpeg-4 and h. 263 video traces for network performance evaluation," *IEEE network*, vol. 15, no. 6, pp. 40–54, 2001.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[28] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.