The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective

Nei Kato, Zubair Md. Fadlullah, Bomin Mao, Fengxiao Tang, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani

Nei Kato, Zubair Md. Fadlullah, Bomin Mao, and Fengxiao Tang are with Tohoku University.

Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani are with Nippon Telegraph and Telephone Corporation (NTT) Network Innovation Laboratories.

Digital Object Identifier: 10.1109/MWC.2016.1600317WC

Abstract

Recently, deep learning, an emerging machine learning technique, is garnering a lot of research attention in several computer science areas. However, to the best of our knowledge, its application to improve heterogeneous network traffic control (which is an important and challenging area by its own merit) has yet to appear because of the difficult challenge in characterizing the appropriate input and output patterns for a deep learning system to correctly reflect the highly dynamic nature of large-scale heterogeneous networks. In this vein, in this article, we propose appropriate input and output characterizations of heterogeneous network traffic and propose a supervised deep neural network system. We describe how our proposed system works and how it differs from traditional neural networks. Also, preliminary results are reported that demonstrate the encouraging performance of our proposed deep learning system compared to a benchmark routing strategy (Open Shortest Path First (OSPF)) in terms of significantly better signaling overhead, throughput, and delay.

Introduction

In recent years, the field of machine intelligence, dictated by deep learning, is flourishing. Technology giants such as Google, Microsoft, Facebook, Amazon, Nvidia, and others are investing heavily with their powerful computing resources to drive machine intelligence research, particularly aiming at deep learning breakthroughs. Machine intelligence, also known as machine learning, is now a thriving field with emerging deep learning techniques in active research topics and relevant applications ranging from speech recognition to driver-less smart cars. In March 2016, AlphaGo, Google's DeepMind Artificial Intelligence (AI) program based on deep learning, managed to beat the champion player of the ancient board game "Go" in four out of five games [1]. The size of the search required for "Go" is larger than chess by more than the number of atoms in the universe determined only recently, in early 2016. AlphaGo demonstrated its ability to look "globally" across a board and find solutions that humans either have been trained not to play or would not consider. This has huge potential for using AlphaGo-like deep learning technology to discover solutions that humans do not necessarily notice in other areas.

In contrast to other machine learning systems shown in Fig. 1 that must be highly tuned to solve specific problems and need a plethora of rules for successful operation, deep learning techniques deal with the use of vast, virtual neural networks to learn and recognize abstract patterns. In this way, deep learning algorithms work similar to how the human brain operates. As data moves from the input of the deep neural network through successive hidden layers, its representation inside the computer becomes more abstract. Increased computing power and advancements in graphics processing units (GPUs) have made it possible to map and process much larger and deeper neural networks than was possible in earlier generations. The available data that can be used to train these systems has also increased dramatically. As depicted in Fig. 1, deep learning systems are being used in speech recognition, computer vision (object perception, visual searches for retail industries, self-driving smart vehicles, robotics, home security, wearables, etc.), and natural language processing.

On the other hand, the application of deep learning in network traffic control has yet to emerge due to the challenge in characterizing appropriate input and output patterns. Network traffic control is an essential component for today's mobile-centric heterogeneous networks to deliver a variety of services and experiences to users as the Internet of Everything continues to take shape. Therefore, next-generation wired and wireless heterogeneous networks need an intelligent mechanism to control the massive growth in network traffic trending from changing networking landscapes such as mobility, cloud computing, big data processing, and machine-to-machine connectivity.

In this article, for the very first time, we show a proof-of-concept to leverage deep learning neural networks to significantly improve heterogeneous network traffic control. In this vein, we envision a supervised deep learning system, and describe the key difference between our propos-

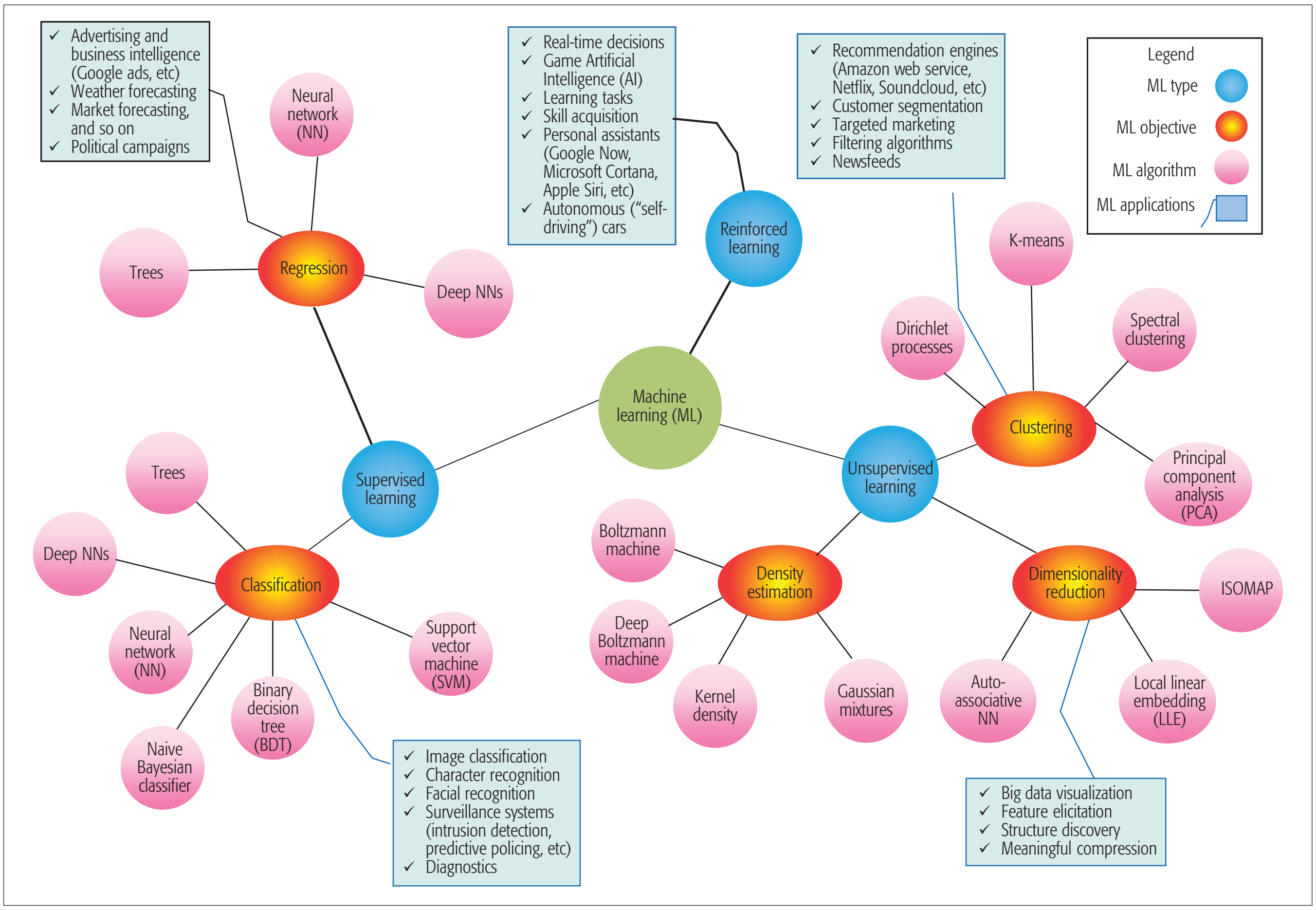


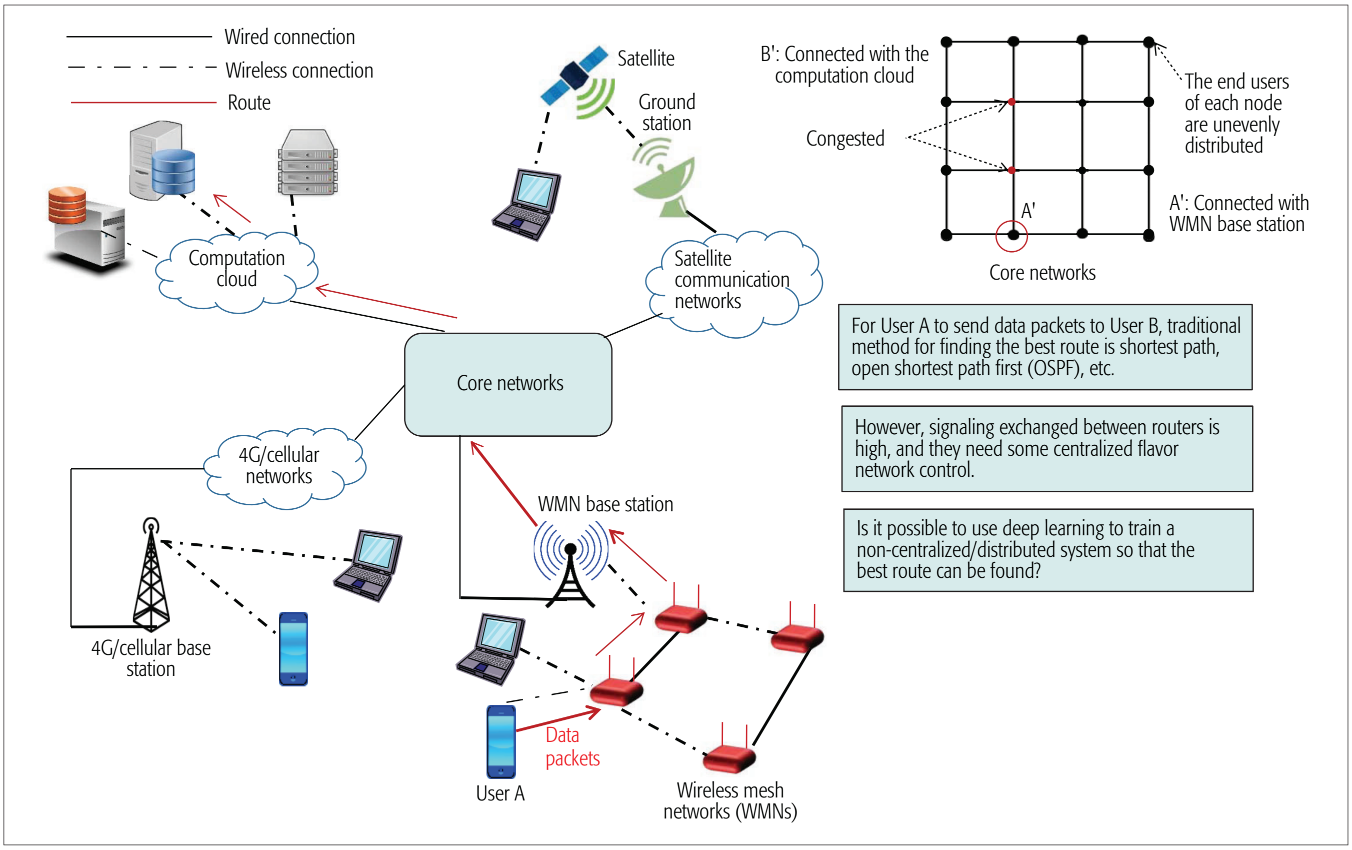
FIGURE 1. Different types and applications of Machine Learning exploited for solving computer science problems. Note that deep learning has only appeared recently and its use is limited to image/character/facial (i.e., patterns) recognition and has not been applied to heterogeneous network control to the best of our knowledge.

al and conventional neural networks. Traditional neural networks, which may also have many hidden layers, usually suffer from poor performance and exhibit similar performance comparable to that of training a shallow neural network. So, it is difficult for traditional neural networks to learn very complex functions or patterns. Therefore, their shortcomings in terms of computational efficiency and scalability become quite evident when used for network traffic control such as routing. As a remedy, we consider a deep learning system comprising multiple hidden layers, each of which computes a non-linear transformation of the previous layer. In addition, we use the greedy layer-wise training method to initialize the deep learning system, and further use the backpropagation algorithm to fine-tune deep learning training. Another contribution of our work is to demonstrate that with appropriate input/output traffic pattern characterizations of a deep learning structure, it is possible to significantly improve network traffic control compared to conventional routing strategies. Our proposed system works in three steps, i.e. the initial, training, and running phases. In addition, preliminary simulation results are provided to demonstrate the viability of our proposed deep learning system to improve heterogeneous network control. Simulation results indicate that our proposal leads to superior performance compared to the benchmark routing protocol (i.e., Open Shortest Path First or OSPF) under supposed conditions, and results in less signaling overhead.

The remainder of the article is organized as follows. The following section includes relevant research work and the state of the art in deep learning research in various disciplines. The problem statement and our considered system model are then described. Our proposed deep learning system model for heterogeneous network control is then presented. Following that, the performance evaluation of our proposal is provided. Finally, we conclude the article.

RELATED RESEARCH WORK

Since the 1990s, many researchers have been using machine learning techniques comprising both statistical and neural network approaches [2, 3] in the field of character recognition. However, due to the lack of computational resources and data samples, the statistical approach was considered to be superior to the neural network method in terms of accuracy of recognition. However, by overcoming the over-fitting problem in the studies of recent years, neural networks driven by deep learning and their applications in various computer science disciplines are gaining renewed interest. In particular, the work in [4] identified these "extreme learning machines" as an emerging learning technique for real-life applications



For User A to send data packets to User B, traditional method for finding the best route is shortest path, open shortest path first (OSPF), etc.

However, signaling exchanged between routers is high, and they need some centralized flavor network control.

Is it possible to use deep learning to train a non-centralized/distributed system so that the best route can be found?

FIGURE 2. Considered network scenario and problem statement.

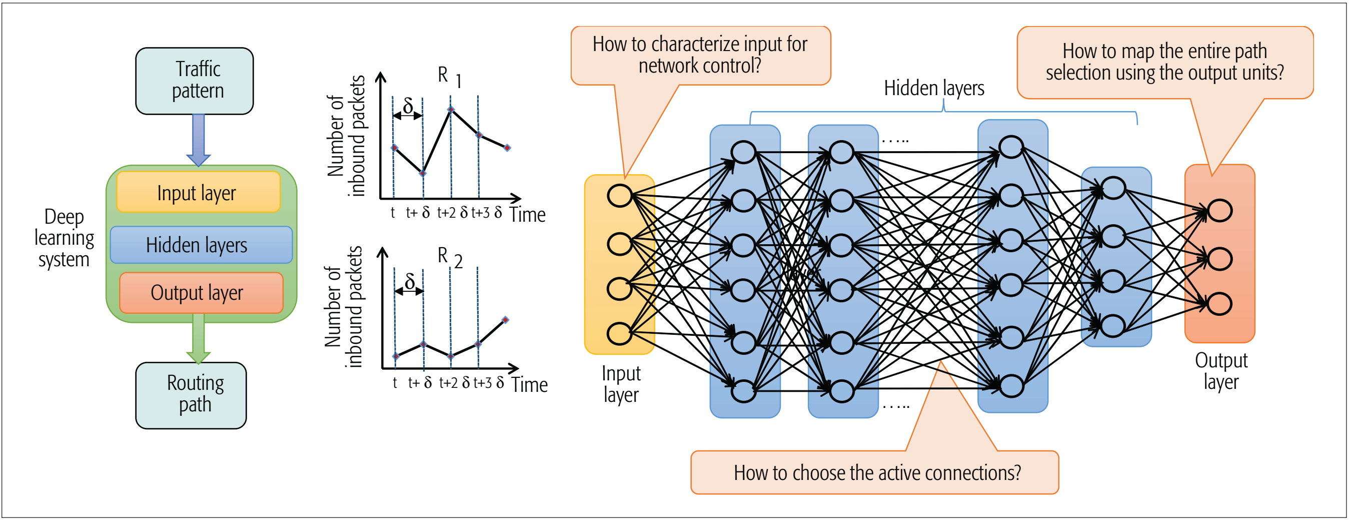
involving big data to provide efficient unified solutions to generalized feed-forward networks such as multi-hidden-layer neural networks, radial basis function (RBF) networks, and kernel learning. Hinton *et al.* in [5] demonstrated that a restricted Boltzmann machine (RBM) and auto-encoders can be utilized for feature engineering that, in turn, can be used to train multiple-layer neural networks, that is, deep networks. These were referred to as the deep belief networks (DBNs). On the other hand, their subsequent work in [6] discussed a new learning algorithm for Boltzmann machines, referred to as the deep Boltzmann machine (DBM), which contains many layers of hidden variables. However, these early deep learning systems, when used for big data, exhibit slow learning speeds. As a remedy to this issue, Kasun *et al.* [7] introduced the extreme learning machine as a feed-forward neural network with a fast learning speed and good generalization capability.

Deep learning is triggering a massive investment opportunity as technology giants such as Apple, Google, Microsoft, Facebook, Nvidia, and others are demonstrating that deep learning is not only a technology itself but also can be leveraged to solve challenges across various industries. Apple stated that their deep learning neural network-based upgrade to the smart assistant "Siri" improved accuracy significantly [8]. Google has adopted deep learning for the speech and image recognition capabilities of Google Translate and Google Photos, respectively. Google's TensorFlow and DeepMind (the initiative that successfully trained the AlphaGo program to beat the world champion of the board game "Go", as described earlier), Microsoft's Computational Network ToolKit (CNTK), Facebook's DeepText, Nvidia's DGX-1, Amazon's Deep Scalable Sparse Tensor Network Engine (DSSTNE), are exploiting GPUs to parallelize deep network training [9]. However, the aforementioned deep learning systems in the literature are limited to specific problems such as character/image recognition, computer vision (visual searches for retail industries, self-driving cars, home security, wearables, etc.), big data visualization, vehicle routing, natural language processing, speech recognition, etc. On the other hand, to the best of our knowledge deep learning systems have not been leveraged for network traffic control and management. Because traditional neural networks and other machine learning techniques are not inherently efficient or scalable enough to cope with the large volume of data, the networking community did not pursue their use in the long run. In this article, we aim to overcome these problems by introducing the first-ever deep learning based network traffic control mechanism.

Problem Statement and Considered Deep Learning System Model

In this section, we present our problem statement and describe our considered deep learning system model.

The mechanism to choose one route from a number of alternative paths to connect each source-destination pair in a communication network is referred to as a routing strategy. The routing strategy in wired/wireless heterogeneous networks as shown in Fig. 2 can be formulated as a classical combinatorial optimization problem, that is, a shortest path routing problem in graphs.



Traffic pattern
Input layer
Hidden layers
Output layer
Deep learning system
Routing path
Number of inbound packets
R 1
δ
t t+ δ t+2 δ t+3 δ Time
R 2
How to characterize input for network control?
Hidden layers
How to map the entire path selection using the output units?
Input layer
Output layer
How to choose the active connections?

FIGURE 3. Considered model of the proposed deep learning system.

From the preceding section, we understand that the machine learning discipline has a rich history with a plethora of techniques that could be used for network control. Conventional machine learning techniques such as artificial neural networks are, however, not without problems. Their shortcomings in terms of computation efficiency and scalability become quite evident when used for network traffic control such as routing. Due to the recent breakthrough in deep learning algorithms and advancements in computing power, now we think it is the time to bring the deep learning method together with network traffic control. While these two originally are totally different areas, given the tremendous advancement in both these disciplines in recent years, now it is the time to propose the interdisciplinary area encompassing deep learning and network traffic control. In this vein, we present our system model in the remainder of this section.

A straightforward representation of our system is portrayed in Fig. 3a. As shown in the figure, for network traffic control, traffic patterns of the routers are served as inputs into the deep learning system. Based on these inputs, the deep learning system is supposed to provide the desired output for network traffic control, namely the routing paths. Figure 3b shows an illustration of the inputs of the deep learning system, that is, the previous history of traffic patterns observed at each router over time-slots of δ. Our considered deep learning system based on a neural network architecture is demonstrated in Fig. 3c. The deep neural network, depicted in this figure, comprises an input layer, multiple hidden layers, and an output layer. Also, as shown in the figure, a key challenge is how to characterize this input for network traffic control. The input layer consists of N input units, where N denotes the number of routers in the target (i.e., considered) network. We use a vector of size N as the input to represent the traffic patterns of the routers in the network. And the i^{th} element of the vector is the number of inbound packets into router i during time-slot δ. Next, the considered deep learning system has multiple hidden layers, each of which computes a non-linear transformation of the previous layer. Thus, the deep learning model can learn significantly more complex functions in contrast with a shallow learning model.

Although the theoretical aspects of deep architectures, such as their compactness and expressive power, have been appreciated over the years, researchers have just recently started to notice success in training deep networks. Previously, researchers were using algorithms to randomly initialize the weights of a deep network, and then train it using a labeled training set $\{(x^{(i)}, y^{(i)}) \mid i = 1, ..., m\}$ by employing a supervised learning objective (e.g., by applying gradient descent) in order to reduce training error. However, this usually did not work well due to a number of reasons such as insufficient training data and diffusion of gradients [10]. In other words, if the entire neural network is trained at the same time with all the layers randomly initialized, it will result in performance comparable to that of training a shallow network.

Finally, in the output layer, the same non-linear activation function adopted in the hidden layers is used. Here, we describe the characterization of the output layer of the model in Fig. 3c. The output for network traffic control is the routing path indicating the next router along the path from the source router to the destination. Based on this, a vector of size N can represent this output such that each of its elements has a binary value. Furthermore, only a single element in this vector can be 1. The position or order of the element in the vector having the value of 1 indicates the next node.

To elucidate the input and output characterization of our considered system model, please refer to Table 1. The table lists the different numbers of input and output units for a network comprising 16 routers (i.e., N = 16). The top row indicates the centralized network control scenario whereby 33792 output units are required in the deep learning system leading to huge complexity. The second row shows a 16 × 16 matrix in the output vector that can provide the entire routing path as output. However, it leads to a significantly high error rate of 70 percent. The third row uses a non-binary output vector to represent the entire path. However, this characterization also suffers from high error rate (i.e., 45%). The fourth row is our chosen structure described in this section which yields an error rate of 5 percent. The final

The objective of the initial phase is to obtain the relevant data for training the deep learning system. One way is to use the traditional routing strategies such as OSPF to simulate the communication between different routers under varying loads and conditions, and record the traffic patterns and paths to be used in the training phase.

Number of input units	Number of output units	The whole path or next router (node)?	Error rate
16	33792	Whole path	–
16	256	Whole path	70%
16	16	Whole path	45%
16	16	Next node	5%
48	16	Next node	5%

TABLE 1. Effect of different input and output characterization strategies on the network control accuracy for $N = 16$.

row uses 48 input units characterizing the input vector having traffic patterns over 3δ. However, its error rate is also similar to our chosen structure at the cost of higher complexity. Now that we have described our considered system model, in the following section, we present our deep learning training and running algorithms for traffic control in heterogeneous networks.

Proposed Deep Learning System for Heterogeneous Network Control

In this section, we describe our proposed deep learning system for heterogeneous network traffic control. Our proposed deep learning system operates in three steps, i.e. initial phase, training phase, and action or running phase. These three phases are described as follows.

Initial Phase

The objective of the initial phase is to obtain the relevant data for training the deep learning system. One way is to use the traditional routing strategies such as OSPF to simulate the communication between different routers under varying loads and conditions, and record the traffic patterns and paths to be used in the training phase. Another approach is to extract the relevant traffic information from a number of available datasets to be utilized during the training phase.

Training Phase

In the training phase, supervised learning is used in order to train the deep learning system based on the collected information in the initial phase. Let N denote the number of routers in the target core/backbone network. Suppose that the number of inner (non-edge) routers is $I < N$. The input of the training algorithm is ($\vec{x}$, $\vec{y}$) where $\vec{x}$ and $\vec{y}$ are vectors of N dimensions. Here, $\vec{x}$ and $\vec{y}$ represent the traffic patterns of N routers and the next router, respectively. The output of the training algorithm is the weight matrix (*WM*). The training algorithm comprises two main parts:

- It employs the greedy layer-wise training method to initialize the deep learning system.
- It uses the backpropagation algorithm to fine-tune the deep learning system.

The training period is executed in each router. In addition, each router needs to train several deep learning systems according to how many destination routers it has. In the considered network, all the data packets are destined for edge routers. Since the number of edge routers in the considered network is $(N - I)$, every edge router needs to train $(N - I - 1)$ deep learning systems since it has $(N - I - 1)$ destination edge routers. On the other hand, the inner routers need to train $N - I$ deep learning systems because they have as many destinations. In the running period, every deep learning system will be used to predict only the next node. Therefore, we need to use the traffic patterns and the corresponding next nodes to train our proposed deep learning systems.

In the training period, labeled data are input to each router to train its deep learning systems. The labeled data comprise two parts, namely $\vec{x}$ and $\vec{y}$. As earlier mentioned, $\vec{x}$'s dimensions are chosen so as to represent the traffic patterns of N routers. Here, the quantity of a router's inbound packets in a given time-slot is used to represent its traffic pattern. Since every deep learning system is just used to predict the next node, $\vec{y}$, another N-dimensional vector is designed in such a manner that only one element in $\vec{y}$ is 1 while the other elements of $\vec{y}$ are all 0s. For instance, if router k is the next node, then the kth element in $\vec{y}$ is 1. We use m sets of data to train the deep learning systems.

Now we describe the two core steps of the training algorithm, namely the greedy layer-wise training method for the initialization, followed by the backpropagation algorithm to perform the fine-tuning [11]. First, using the greedy layer-wise training method, the layers of the neural network are trained one at a time. This means that we first train a deep learning system with one hidden layer. Only after the training of the first hidden layer is accomplished, the training of a deep learning system with two hidden layers may commence, and so forth. At each step, we take the old deep learning system with $(k - 1)$ hidden layers we have just trained as input, and add the kth hidden layer. After this step is finished, the backpropagation algorithm is applied to adjust the weights between the layers to minimize the difference between the output of the deep learning system and the given output, $\vec{y}$ [10]. The training period is not finished until the difference between the two outputs satisfies the requirement or the number of training adjustments reaches a given value [11].

Once the training phase of a router is completed, it transmits its *WM*s to the edge routers. This is done in a multicasting manner whereby the destination nodes are only the edge routers. In other words, each edge router needs to send its $(N - I - 1)$ weight matrices to the other $(N - I - 1)$ edge routers. On the other hand, each of the inner nodes needs to send its $(N - I)$ weight matrices to all the edge routers. In this way, every edge router can obtain all the weight matrices in the network. Let DL_{ij} represent the deep learning system in router i for destination node j. In other words, i and j denote any of the routers and just any of the edge routers, respectively. Also, let WM_{ij} denote the weight matrix of DL_{ij}. Thus, the weight matrices are used to construct the corresponding deep learning system. It is worth noting that the destination node should not be itself, that is, $i \neq j$.

Running Phase

In the remainder of this section, we describe the

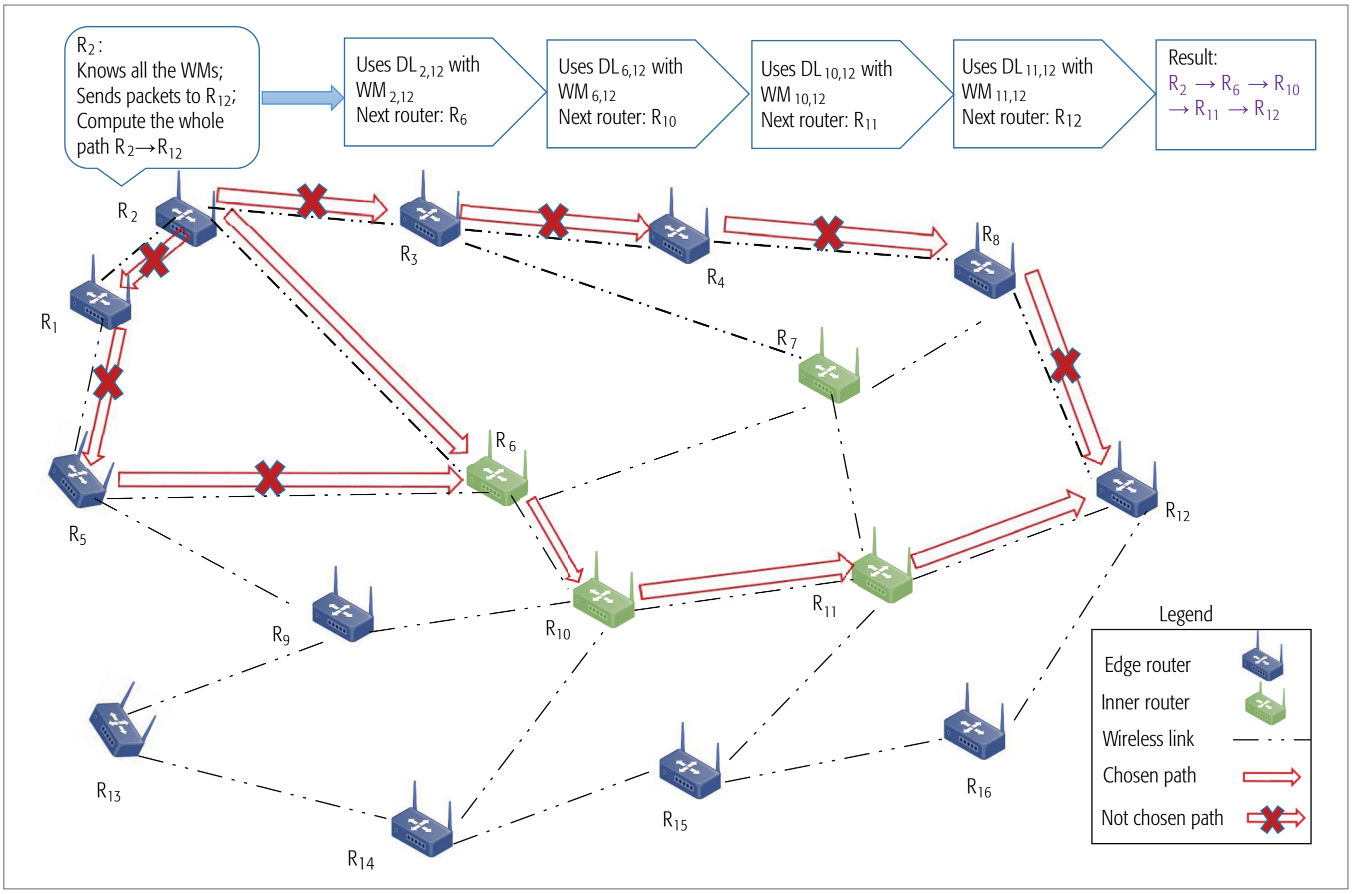

R2: Knows all the WMs; Sends packets to R12; Compute the whole path R2→R12
Uses DL 2,12 with WM 2,12 Next router: R6
Uses DL 6,12 with WM 6,12 Next router: R10
Uses DL 10,12 with WM 10,12 Next router: R11
Uses DL 11,12 with WM 11,12 Next router: R12
Result: R2 → R6 → R10 → R11 → R12
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
Legend
Edge router
Inner router
Wireless link
Chosen path
Not chosen path

FIGURE 4. A simple example in which router R_2 sends packets to router R_{12}.

algorithm used in our deep learning system for the actual running phase. The traffic patterns of the routers are denoted by *TP*s, which are used along with *WM*s as inputs of the running algorithm. The algorithm outputs the whole path to the edge routers. At first, each router transmits its traffic pattern to the edge routers. Then, each router constructs DL_{ij} using WM_{ij}, and predicts the next router with DL_{ij} and *TP*s. In the next step, the deep learning system of the next router for destination edge router *j* is formulated with the weight matrix, and the next router is predicted. This step is repeated until the next router predicted is router *j*. With the weight matrices obtained by every edge router, we can run the deep learning systems to predict the whole path.

To easily demonstrate how the proposed algorithms work, a simple example is presented in Fig. 4. Assume that the router *k* is labeled as R_k. The figure illustrates a situation in which R_2, an edge router, wants to send data packets to R_{12} (another edge router). Then, R_2 needs to run the deep learning systems to compute the whole path (indicated by $R_2 \to \ldots \to R_{12}$). In this vein, R_2 first uses the deep learning system ($DL_{2,12}$) with $WM_{2,12}$, and predicts the next node is R_6 after the traffic pattern $\vec{x}$ is input to $DL_{2,12}$. Then, it uses $DL_{6,12}$ with $WM_{6,12}$ and predicts the third node to be R_{10}. At this stage, the router constructs $DL_{10,12}$ with $WM_{10,12}$, and predicts the fourth node is R_{11}, and so forth. After this process, R_2 evaluates the whole path to be $\{R_2 \to R_6 \to R_{10} \to R_{11} \to R_{12}\}$. Thus, every edge router can predict the whole path to all other edge routers, and attaches the path to the corresponding packets. It is also worth mentioning here that time-slot based synchronization is used in the training phase in our proposed method. This is similar to existing routing algorithms (e.g., OSPF and others) that are executed over fixed time intervals. On the other hand, synchronization between the routers is not required during the running phase even though it uses time-slots.

PERFORMANCE EVALUATION

In this section, the performance of our proposed deep learning system is evaluated. In our conducted simulations based on C++/WILL [12], we conducted all the routers' computations on a workstation with an Intel Core i7 3.60 GHz processor and 16GB random access memory (RAM). Because the computations of all the routers were outsourced to a single machine, the evaluation was restricted to a small size network. Therefore, we considered a medium scale wireless mesh backbone network with 16 routers rather than a full-scale wired/wireless heterogeneous core network topology. It is worth noting that this scale of simulation is sufficient enough in case it demonstrates that the proposed deep learning system outperforms the conventional routing strategies such as OSPF. In the conducted simulations, only the edge nodes are considered to generate data packets, and all the data packets are destined to other edge nodes. The inner nodes are assumed to just forward the packets. The data and control packet sizes are both set to 1 kb. The link bandwidths are set to 8 Mb/s, which is reasonable for

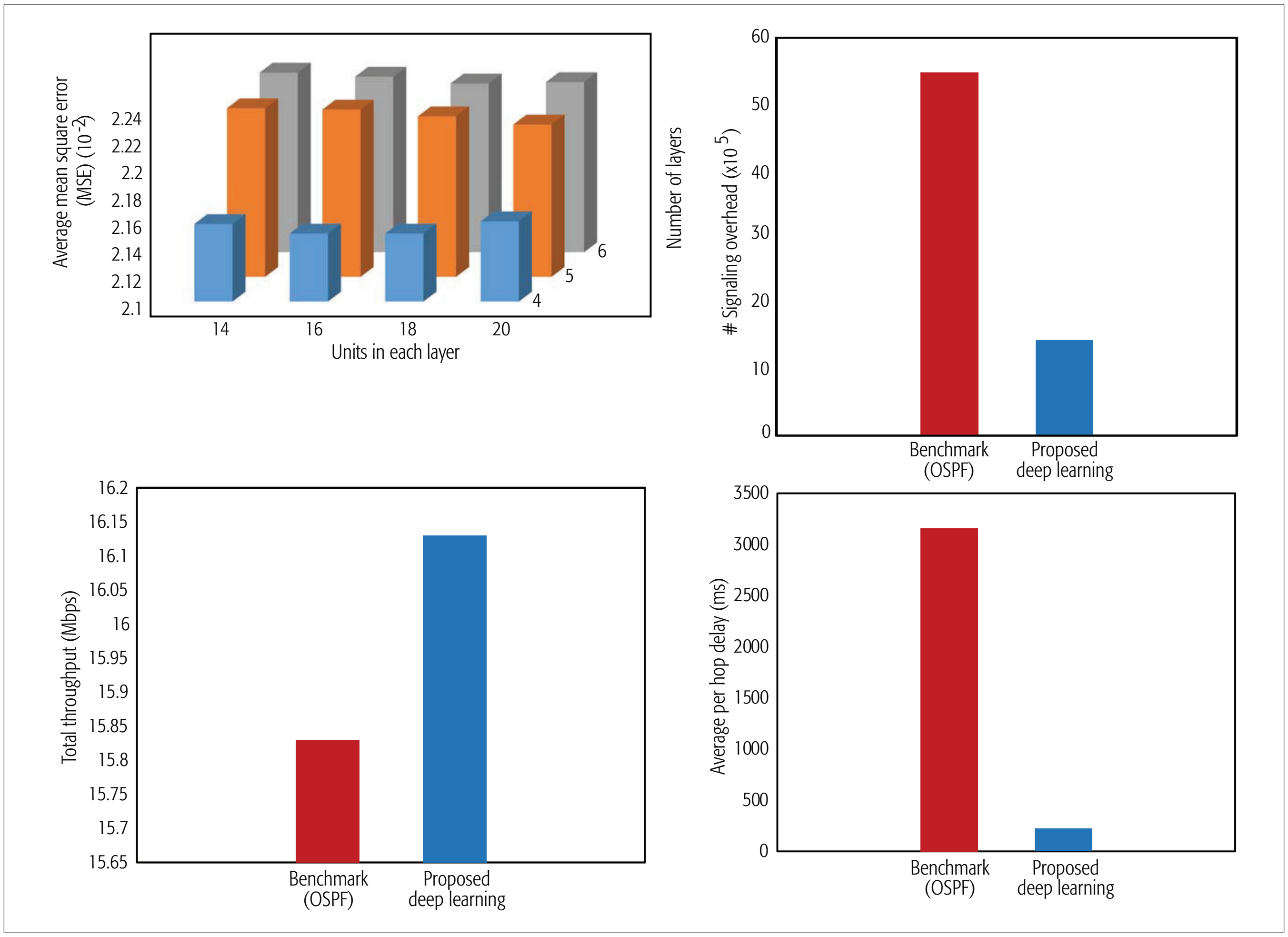


FIGURE 5. Comparison of learning structures for the considered network, and that between the benchmark method (OSPF) and the proposed deep learning system in terms of signaling overhead, throughput, and average per hop delay.

this scale of wireless mesh backbone [13]. Every node is assumed to have an unlimited buffer. The overall data packet generating rate in the considered network is 16.32 Mb/s. For comparison of the proposed deep learning system, OSPF is used as the benchmark method. In the conducted simulations, δ is set to 0.25s during which signaling is exchanged once.

To decide the number of layers and the number of units in every hidden layer, different deep learning structures are compared first. In the training process of the deep learning system, the mean square error (MSE) is often used as the stop criterion of the training. Here, we use this value to measure the performance of different deep learning structures. The result is shown in Fig. 5a. For simplicity, 12 structures in which the numbers of layers are varied from 4 to 6 while the number of units in every hidden layer is varied in the range of {14, 16, 18, and 20}. It can be noticed that when the number of units in the hidden layers is fixed, the value of MSE grows bigger as the number of layers grows. This indicates that four layers are sufficient for our training data. In other words, more layers will cause the problem of over-fitting for the considered scenario. On the other hand, the changing trend of MSE with the number of units in the hidden layers is not the same for different numbers of layers. For instance, for deep learning systems comprising four layers, MSE reaches the minimum value when the number of units in every hidden layer is 16 and 18, which means that these two types of structures can achieve similar performance. However, the training becomes more complex with the increase in units in the hidden layers. Therefore, we choose the deep learning structure of four layers and 16 units in every hidden layer for the evaluation of network performance metrics shown in Figs. 5b, 5c, and 5d. As shown in these figures, three network performance metrics are evaluated between the conventional OSPF and deep learning. First, the signaling overhead is compared with the benchmark method (i.e., OSPF) in Fig. 5b. The results show that the signaling overhead of OSPF is much higher compared to that of the proposed deep learning system. The lower signaling achieved by our proposal can be explained as follows. While the conventional OSPF method needs all the routers to frequently flood their respective neighbors with the routing information, in the proposed deep learning method, only the edge routers' traffic patterns are sufficient to compute the whole path with an accuracy as high as 95 percent. Therefore, in the proposed deep learning method, only the edge routers need to exchange traffic patterns among themselves, and the traffic patterns of the inner routers are arbitrarily set in the running phase. Then, in Fig. 5c the throughputs of the two methods are compared. It can be noticed that the throughput achieved by the proposed deep learning system is almost close to the average data generation rate of the routers since it avoids congestion and packet drops by evaluating routes to the destination much faster than the conventional OSPF. Finally, Fig. 5d demonstrates that the deep learning method finds the appropriate routes quickly enough so that the average per hop delay of the proposed deep learning system is significantly lower than that of the benchmark method.

Conclusion and Future Directions

In this article, we explained why it is important to rethink how the heterogeneous network traffic control method such as routing management can be improved to deal with massive traffic growth. In this vein, we envisioned that deep learning, which has recently emerged as a promising machine learning technique, can substantially improve heterogeneous network traffic control. A supervised deep learning system was proposed that is trained based on uniquely characterized inputs using traffic patterns observed at the edge routers of the considered network. The operations of the proposed algorithms of the learning and running phases of our deep learning system were demonstrated through an example. Also, preliminary simulation results were reported to demonstrate the effectiveness of our proposal, which clearly outperformed the considered benchmark routing method (OSPF). To the best of our knowledge, this is the first work demonstrating the proof-of-concept of applying a deep learning system to improve heterogeneous network traffic control. Future directions include consideration of applying other types of deep networks (e.g., DBMs and so forth) and using the unsupervised deep learning strategy to take into account more input features in addition to the traffic patterns.

References

[1] "Google DeepMind," https://deepmind.com/alpha-go, (accessed Nov. 2016).

[2] N. Kato *et al.*, "A Handwritten Character Recognition System Using Directional Element Feature and Asymmetric Mahalanobis Distance," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 21, no. 3, Mar. 1999, pp. 258–62.

[3] K. Saruta *et al.*, "High Accuracy Recognition of ETL9B Using Exclusive Learning Neural Network-II: ELNET-II," *IEICE Trans. Information Systems* (Special Issue on Character Recognition and Document Understanding), vol. 79, no. 5, May 1996, pp. 516–22.

[4] E. Cambria *et al.*, "Extreme Learning Machines [Trends Controversies]," *IEEE Intelligent Systems*, vol. 28, no. 6, Nov. 2013, pp. 30–59.

[5] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, Jul. 2006, pp. 504–07.

[6] R. Salakhutdinov and G. Hinton, "An Efficient Learning Procedure for Deep Boltzmann Machines," *Neural Computing*, vol. 24, no. 8, Aug. 2012, pp. 1967–2006.

[7] L. L. C. Kasun, H. Zhou, and G.-B. Huang, "Representational Learning with ELMs for Big Data," *IEEE Intelligent Systems*, vol. 28, Nov.-Dec. 2013, pp. 31–34.

[8] "The iBrain is Here and It's Already Inside Your Phone," https://backchannel.com/an-exclusive-look-at-how-ai-and-machine-learning-work-at-apple-8dbfb131932b#.43bf-9cm00, (accessed Nov. 2016).

[9] "Deep Learning Comp Sheet: Deeplearning4j vs. Torch vs. Theano vs. Caffe vs. TensorFlow," http://deeplearning4j.org/compare-dl4j-torch7-pylearn.html, (accessed Nov. 2016).

[10] "Deep Networks: Overview," http://deeplearning.stanford.edu/wiki/index.php/DeepNetworks:Overview, (accessed Nov. 2016).

[11] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," http://www.deeplearningbook.org, (accessed Nov. 2016), book in preparation for MIT Press.

[12] "WILL API," https://scarsty.gitbooks.io/will/content/, (accessed Nov. 2016).

[13] A. Raniwala and T. Chiueh, "Evaluation of a Wireless Enterprise Backbone Network Architecture," *Proc. 12th Annual IEEE Symposium on High Performance Interconnects*, Stanford, CA, USA, Aug. 2004.

Biographies

Nei Kato [M'04, SM'05, F'13] is a professor at the Graduate School of Information Sciences (GSIS), Tohoku University, Japan. He became a strategic adviser to the President of Tohoku University in 2013 and the Director of the Research Organization of Electrical Communication (ROEC), Tohoku University in 2015. He has been engaged in research on computer networking, wireless mobile communications, satellite communications, ad hoc and sensor and mesh networks, smart grid, and pattern recognition. He has published more than 300 papers in peer-reviewed journals and conference proceedings. He currently serves as a member-at-large on the IEEE Communications Society Board of Governors, the Chair of the IEEE Communications Society Sendai Chapter, and IEEE Communications Society Award Committee (2015-2017). He is the editor-in-chief of *IEEE Network Magazine*, associate editor-in-chief of *IEEE Internet of Things Journal*, and an area editor of *IEEE Transactions on Vehicular Technology*. He was a distinguished lecturer of the IEEE Communications Society and the IEEE Vehicular Technology Society. He is also a fellow of the IEICE.

Zubair Md. Fadlullah [M'11, SM'13] is serving as an assistant professor at GSIS. His research interests are in the areas of 5G, smart grid, network security, intrusion detection, game theory, and quality of security service provisioning mechanism. He was a recipient of the prestigious Dean's and President's Awards from Tohoku University in March 2011, the IEEE Asia Pacific Outstanding Researcher Award in 2015, and the NEC Tokin award for research in 2016. He has also received several best paper awards at conferences including IWCMC'09 and GLOBECOM'14.

Bomin Mao [S'15] received the B.E. degree in telecommunications engineering and the M.E. degree in electronics and telecommunications engineering at Xidian University, China, in 2012 and 2015, respectively. Currently, he is pursuing a Ph.D. degree at the GSIS, Tohoku University, Japan. His research interests are focused on various areas of wireless networks, particularly with applications of machine intelligence and deep learning.

Fengxiao Tang [S'15] received the B.E. degree in measurement and control technology and instruments from the Wuhan University of Technology, Wuhan, China, in 2012, and the M.E. degree in software engineering from the Central South University, Changsha, China, in 2015. Currently, he is pursuing a Ph.D. degree at the GSIS, Tohoku University, Japan. His research interests are unmanned aerial vehicle systems, game theory optimization, and machine learning algorithms.

Osamu Akashi received B.Sc. and M.Sc. degrees in information science from Tokyo Institute of Technology, in 1987 and 1989, respectively. He received his Ph.D. degree in mathematical and computing sciences from Tokyo Institute of Technology in 2001. He joined the Nippon Telegraph and Telephone Corporation (NTT) Software Laboratories in 1989, and is a senior research scientist at the NTT Network Innovation Laboratories. His research interests are in the areas of distributed systems, multi-agent systems, and network architectures. He is a member of ACM, IEICE, IPSJ, and JSSST.

Takeru Inoue is a distinguished researcher at NTT Network Innovation Laboratories. He was an ERATO researcher at the Japan Science and Technology Agency from 2011 through 2013. His research interests widely cover algorithmic approaches in computer networks. He received the B.E., M.E., and Ph.D. degrees from Kyoto University, Japan, in 1998, 2000, and 2006, respectively.

Kimihiro Mizutani [M] is a researcher at NTT Network Innovation Labs. He received the M.S. degree in information systems from the Nara Institute of Science and Technology in 2010. His research interest is focused on future Internet architectures. He received the Best Student Paper Award at the International Conference on Communication Systems and Application (ICCSA) in 2010. He also received research awards from IPSJ and IEICE in 2010 and 2013, respectively. He is a member of IEICE and the IEEE Communication Society.

Future directions include consideration of applying other types of deep networks (e.g., DBMs and so forth) and using the unsupervised deep learning strategy to take into account more input features in addition to the traffic patterns.