International Journal of Web-Based Learning and Teaching Technologies
Volume 16 • Issue 1 • January-February 2021

Deep Learning Forwarding in NDN With a Case Study of Ethernet LAN

Mohamed Issam Ayadi, Hassan II University, Morocco
Abderrahim Maizate, Hassan II University, Morocco
Mohammed Ouzzif, Hassan II University, Morocco
Charif Mahmoudi, LACL U-PEC, France

ABSTRACT

In this paper, the authors propose a novel forwarding strategy based on deep learning that can adaptively route interests/data packets through ethernet links without relying on the FIB table. The experiment was conducted as a proof of concept. They developed an approach and an algorithm that leverage existing intelligent forwarding approaches in order to build an NDN forwarder that can reduce forwarding cost in terms of prefix name lookup, and memory requirement in FIB simulation results showed that the approach is promising in terms of cross-validation score and prediction in ethernet LAN scenario.

KEYWORDS

Deep Learning, Forwarding Strategy, Named Data Networking, Networking, Stochastic Gradient Descent

INTRODUCTION

Recently, Named Data Networking (NDN) has emerged as a new paradigm that could achieve better performance for the content retrieval and dissemination in a highly dynamic environment; it presented numerous advantages such as a secure scheme based on variable-length, location-independent names to fetch the content, simultaneous multiple interfaces network access without repeated IP address acquirement (D. Saxena et al, 2017), and efficient localization of requested content via in-network caching (IMBRENDA C., 2014).

Yi et al. (C. Yi et al,2013) claimed that while NDN's stateful forwarding plane should provide effective content delivery on the propagated routes, handle network problems such as congestion and short-term link failures, NDN routing only hold a supporting role that provides a starting point for the forwarding plane which explores different multipath opportunities. In return, adaptive forwarding enables a more scalable routing plane in terms of convergence time and completeness (D. Posch et al, 2016), (Rainer B. et al, 2016). We agree that the two mechanisms could not be necessary separated but each must act in its area of responsibility. (D. Posch et al, 2016), (D. Saxena, et al., 2016).

The NDN native forwarding model is based on three major tables: a Content Store (CS), that stores the content, a Pending Interest Table (PIT) that registers the forwarded interests that still waiting for their requested data, and a Forwarding Information Base (FIB) containing prefixes and identified outgoing faces based on forwarding strategy(D. Saxena, et al., 2016)

Furthermore, various surveys on aspects of Information-Centric Network (ICN) point scalability as a major challenge (TROSSEN, D.,2016). Major issues were the great number of control messages

DOI: 10.4018/IJWLTT.2021010101

and the vast size of the content naming space. We suspected that the prefix lookup matching in the F.I.B. and its increasing size influenced the forwarding cost in terms of memory and computational resources requirement as highlighted in (D. Saxena, et al., 2016).

For these reasons we proposed a new forwarding model based on deep learning that should reduce the routers forwarding cost because the FIB information should be used only initially for training our model. Then the latter would predict the corresponding outgoing face for each new incoming packet without the lookup matching process in the FIB, thus reducing the number of control messages and the size of memory used in FIB for new paths towards the same content.

As application field we designed and implemented a star topology based on Ethernet interconnection in a local area network. We assumed that all routes and traffic are initially predetermined. The content could be or not distributed and all nodes might either send or receive a content. Our model was implemented and evaluated only in the NDN router. The objectives were:

1. Design, implementation and evaluation of a new intelligent NDN forwarding strategy based on deep learning.
2. Providing effective forwarding without relying on routing protocols.
3. Improving forwarding plane performance by reducing the forwarding cost related to the F.I.B prefix lookup.

To accomplish those objectives we developed and trained our deep learning model based on the stochastic gradient descent algorithm that would predict the outgoing face for each incoming packet. The training data was obtained from native NDN packets information collected by a traffic sniffer implemented in the NDN router. Our trained model was then implemented in the router and acted at forwarding plane; each incoming packet would be forwarded accordingly based on the prediction computed by our model without relying on the prefix lookup in the F.I.B

The reminder of this paper is as follows: Section 2 considers the state of art in NDN intelligent forwarding strategies. In Section 3 we present our deep learning forwarding model. In Section 4 We give a use case for our model implementation and evaluation. Then conclusion is given in Section 5

BACKGROUND

NDN routing is essentially responsible for topology settings and handling policies changes and forwarding table updates. NDN routing can also help the forwarding plane for interface ranking and probing. Thus the only difference is while routing determines available routes, forwarding makes decisions and preferences about the next hop based on the forwarding strategy and performance measurement (D. Saxena, et al., 2016).

1. Forwarding Scalability

To demonstrate its real worth, NDN needs achievement at large scale deployment level: this is to effectively process tens of thousands of interests/data packets per second, and so the size of the content store should be large which make the prefix name lookup more time consuming (H. Yuan,2012). Thus, we support that fast name lookup is the key to make NDN forwarding scalable. Subsequently many probabilistic and adaptive forwarding strategies have been proposed in the literature in order to improve the forwarding process performance based on machine learning and network conditions (Ayadi, M. I.,2018). The following paragraph gives some examples.

2. Some Existing Intelligent Fowarding StrategieS

(Qian, Haiyang et al.,2013) proposed a Probability-based Adaptive Forwarding strategy (PAF) based on the ant colony algorithm optimization to compute the probability of selecting an outgoing interface by probing the performance of each interface in terms of delay and timeout for the retransmission mechanism using the feedback from interests/data packets. It showed a better result in terms of shortest delay.

(L. Gong et al.,2016) proposed a Probabilistic Binary Tree based Forwarding strategy (PBTF) based on an online machine learning algorithm that predicts for each interest, the average delay in each interface based on the real-time selection probability and the Round Trip Time (RTT) representing the time gap between sending an Interest and receiving the corresponding Data. Then the optimal forwarding path is selected based on that prediction. It showed good results in terms of throughput and drop rate.

(Zhang Y. et al., 2018) proposed an Intelligent Forwarding Strategy Based on Reinforcement Learning (IFS-RL) that trains a neural network model based on reinforcement learning to select the outgoing interfaces for the forwarding of interests based only on observations collected by the routing plane. It showed best throughput and drop rate performance compared to the Bestroute and EPF strategies (Ayadi, M. I., 2018).

DEEP LEARNING FORWARDING MODEL

At routing plane, NDN allows to gather stateful information such as packet size, cache hit ratio, throughput and RTT during data retrieval. The forwarding strategy is responsible for selecting the next hop based on forwarding metrics, routing information and local policies (Ayadi, M. I., 2018).

In this paper we attempt to use initially, available information obtained from traversing traffic and the fully pre-established FIB to train our deep learning model which will then predict the outgoing face for each incoming packet. We aim to minimize the forwarding cost and then improve the overall forwarding performance.

1. Stochastic Gradient Descent (SGD)

A typical approach to an optimization problem in artificial systems is to follow the gradient of the objective function that quantifies the system's performance parameters in the direction of locally ultimate improvement (Werfel J.K. et al,2004)

The gradient descent optimization algorithm aims to minimize the loss function based on that function's gradient. Successive iterations are employed to progressively approach either a local or global minimum of the cost function.

When training weights in a neural network, normal batch gradient descent usually takes the mean squared error that we note J(W,b) of all the training samples when it is updating the weights of the network:

$$W=W-\eta\nabla J(W,b)$$

Where W are the weights, η is the learning rate and ∇ is the gradient of the cost function J(W,b) with respect to changes in the weights and bias b.

We define the cost function J as:

$$J\left(W,b\right)=\frac{1}{m}\cdot\sum_{i=0}^{m}J\left(W,b,x\left(z\right),y\left(z\right)\right)$$

x(z) and y(z)refer to each training sample pair of sample z. As we can see, the overall cost function so the gradient depends on the mean cost function calculated on all of the m training samples.

Stochastic gradient descent updates the weight parameters after evaluation of the cost function after each sample. That is, rather than computing the average of cost function results for all the sample, it updates the weights after every training sample is evaluated. Therefore, the updates look like this:

$$W=W- \eta \nabla J(W,b,x(z),y(z))$$

This weights update can be easily implemented by a minor variation of the batch gradient descent code in Python, by shifting the update component into a loop:

2. OUR MODEL DEFINITION

We design our deep learning model as a feed-forward neural network with one input layer containing the same number of neurons as the inputs number in our dataset. After we add two hidden layers with 8 neurons each, then an output layer is added with Softmax activation to ensure the output values are in between 0 and 1 which can be used as predicted probabilities.

Our neural network topology can be summarized as:

6 inputs -> [8 hidden nodes] -> [8 hidden nodes] -> 4 outputs

The output value with the great value will be taken as the class predicted by the model.

Our neural network used the SGD optimization algorithm with a logarithmic loss function that is justified in paragraph IV.C.a) for training, then we compute the overall cross validation score for performance evaluation.

3. TRAINING DATA

The training data was defined by the native features of the NDN packets in order to use all available information provided by traversing packets without relying on routing protocols and to support the reproducing of our forwarding scheme in other networks. For each incoming packet we omitted the FreshnessPeriod and MustBeFresh fileds because we don't treat the freshness of data in the content store as it is directly retrieved from the corresponding producer, then we considered the following major remaining fields (A. Afanasyev,2015):

The face in, packet size, type, prefix, nonce, lifetime were used as dataset input while the face out was used as output for prediction target.

USE CASE SIMULATION: NDN ETHERNET LAN

Our Deep learning model is designed to be able to predict the correct outgoing face for an incoming packet based on knowledge of its incoming face, its size, type, the prefix name, the nonce, and the interest lifetime.

The designed topology is based on Ethernet links (faces) between NDN nodes and a central NDN router (see fig.1). This makes the forwarding over Ethernet more separated from TCP/IP layer 3 interconnection. Thus, the mapping of the packets information to our model features is essentially based on the NDN native fields which provides the possibility of reutilization of our model on any other local area network architecture based on Ethernet faces/links.

Table 1. Training algorithm

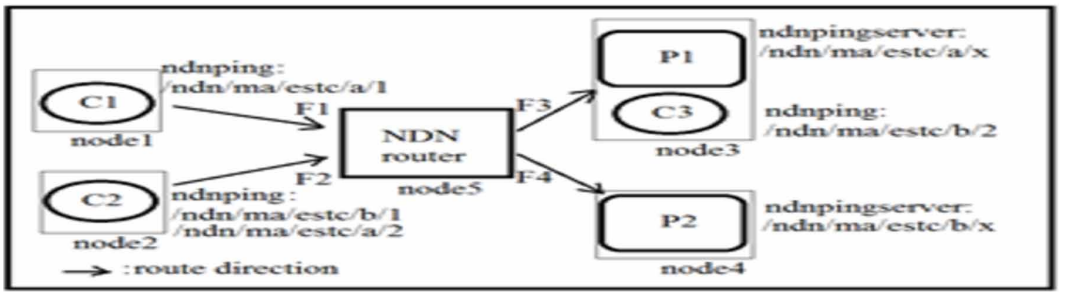
```
We note using the library numpy as np
def train_SGD(nn_topo, X, y, iter_num=2000, eta=0.25,
lamb=0.000):
    W, b = init_weights(nn_topo)
    cnt = 0
    m = len(y)
    avg_cost_func = []
        while cnt < iter_num:
        if cnt%50 == 0:
          tri_W, tri_b = init_tri_values(nn_topo)
         avg_cost = 0
        for i in range(len(y)):
            delta = {}
            # perform the forward propagation
            h, z = forward_propag(X[i, :], W, b)
            # loop from nl-1 to 1 backpropagating the
errors
            for p in range(len(nn_topo), 0, -1):
                if p == len(nn_topo):
                    delta[p] =
compute_out_layer_delta(y[i,:], h[p], z[p])
                    avg_cost += np.linalg.norm ((y[i,:]-
h[p]))
                else:
                    if p > 1:
                delta[p] = compute_hidden_delta(delta[p+1],
W[p], z[p])
                    # triW(p) = triW(p) + delta(p+1) *
transpose(h(p))
                    tri_W[p] =
np.dot(delta[l+1][:,np.newaxis],

np.transpose(h[l][:,np.newaxis]))
                    # trib(p) = trib(p) + delta(p+1)
                    tri_b[p] = delta[p+1]
            # perform the gradient descent step for the
weights in each layer
            for p in range(len(nn_topo) - 1, 0, -1):
                W[p] += -alpha * (tri_W[p] + lamb * W[p])
                b[p] += -alpha * (tri_b[p])
        # final average cost
        avg_cost = 1/m * avg_cost
        avg_cost_func.append(avg_cost)
        cnt += 1
    return W, b, avg cost func
```

Table 2. Training data

Packet size	The packet size in kB where the size of generated interests ranged between 61 and 63 and the size of the generated data packets ranged between 98 and 100
Type	interest or data packet
Prefix	The prefix name
Nonce	The interest nonce
Lifetime	The time remaining before the interest times out
The face in and out	Deduced from the FIB configuration and the architecture design because the dumping tool gave only the source and destination of the captured Ethernet frame

Figure 1. NDN ETHERNET LAN architecture



Network Architecture

- ***NDN Nodes and links description:***

- **NDN router F.I.B configuration:**

The incoming interests are directed to adequate producers following route direction as shown in fig.1:

Each interest requesting /ndn/ma/estc/b/x will be forwarded to face F4
Each interest requesting /ndn/ma/estc/a/x will be forwarded to face F3
Default route is not configured because we don't treat dropped interests

Data Dumping

The dumping tool was placed on the NDN router where the traffic was fully captured because our scenario was built upon a star topology. Our goal was to perform monitoring over Ethernet segments between the NDN router and other nodes. The throughput rate of inter-nodes links was not evaluated due to simulation objective and time restriction.

We chose Tcpdump network sniffer because it is widely used for network monitoring and data acquisition (P. Goyal, A. Goyal, 2017). It can capture interests, data packets on the wire and displays native NDN packet fields which we were essentially looking for. It was mapped to the Ethernet interface

Table 2. NDN ETHERNET LAN Description

NDN router	machine Ubuntu14.04 (2 GHz) running NFD that sends interests and packets data to adequate nodes based on F.I.B
Fx	Ethernet faces that link NDN router to other nodes
Node1, node2	machines Ubuntu14.04 (2 GHz) running customers c1 and c2 respectively
Node3	machine Ubuntu14.04 (2 GHz) running a customer c3 and a producer P1
Node4	machine Ubuntu14.04 (2 GHz) running a producer P2
Px	producers generating prefixes as shown in fig.1 with a frequency of 5 packet/s

used in the creation of the faces F1, F2, F3, and F4 on the NDN router. The output was redirected to a file for data collection. That file was used to create a CSV file containing 3400 records with the attributes defined in the paragraph III-3.3.data training.

Simulation Results

Model Implementation

We implemented our model in the NDN router using keras library. The model was trained and validated using the CSV file obtained from data dumping

The Keras library provides wrapper classes to allow the use of pre-established neural network models in Google scikit-learn. For instance KerasClassifier takes the name of a function as an argument to return the constructed neural network model, ready for training.[*"Keras Documentation" available online* https://keras.io.]

We developed a function that will create a baseline neural network for our model with respect to our definition in paragraph III.B. It creates a simple sequential feed forward network with one input layer containing 6 neurons. After we added two hidden layers with each 8 neurons with Rectifier activation as suggested in Keras documentation. Finally, a 4 neurons ouput layer is added with Softmax activation because we want to predict which one from the 4 interfaces would be likely selected in the forwarding process.

We used the SGD optimization algorithm with a logarithmic loss function called "*categorical_ crossentropy*" in Keras as suggested in Keras documentation to train our model. Also we computed the overall cross validation score for performance evaluation.

Model Validation and Discussion

After many simulations of our model training and computing 5-fold cross validation scores where we changed the number of epochs, and the batch size but fixing the learning rate to 0.01 as suggested by *"Keras Documentation"*. We obtained a result of 80% (+/-9%) with only 30 epochs and a batch size of 128 without indication of overfitting which shows that our model is correctly designed.

The Keras Adam optimizer was also simulated at the same learning rate value as a comparative target, the results showed a less performance (only 76% (+/- 10%)) against our model. We supposed that can be due either to the bad configuration of the decay rates in Adam optimizer or the difficulty of the latter to find a flat minima thus impacting its convergence. Our guess is while the SGD was successful to find a local minima and Adam failed, it can only support the simplicity of our dataset and neural network structure that led SGD to quickly find a local minima which is true. But what of complex structure and dataset?

CONCLUSION

In this paper, we discussed how we can use deep learning in order to exploit content-based information to further improve the forwarding decisions without relying on prefix lookup matching in the F.I.B. We presented our model, methods for gathering data, and the training process. As proof of concept we evaluated our model in an Ethernet LAN that showed good results. Our perspective is to consider more network conditions related parameters for instance, throughput- delay and QoS and develop further our approach.

REFERENCES

Afanasyev, Shi, Zhang, Zhang, Moi-seenko, Yu, Shang, Huang, Abraham, DiBenedetto, Fan, Papadopoulos, Pesavento, Grassi, Pau, Zhang, Song, Yuan, Abraham, Crowley, … Wang. (2015). *NFD Developers Guide.* NDN Project.

Ayadi, M. I., Saadaoui, F. Z., Maizatc, A., Ouzzif, M., & Mahmoudi, C. (2018). Deep Learning for Packet Forwarding with an Application for Real Time IoT. *2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT).*

Gong, L., Wang, J., Zhang, X., & Lei, K. (2016). Intellegent forwarding strategy based on online machine learning in named data networking. Conference Publications in IEEE Trustcom/BigDataSE/-ISPA, 1293-1294.

Goyal, P., & Goyal, A. (2017). Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. *Proceedings of the Ninth International Conference on Computational Intelligence and Communication Networks (CICN)*, 77–81. doi:10.1109/CICN.2017.8319360

Imbrenda, C., Muscariello, L., & Rossi, D. (2014). Analyzing Cacheable Traffic in Isp Access Networks for Micro Cdn Applications via Contentcentric Networking. *Proceedings of the 1st International Conference on Information-centric Networking*, 57-66.

Posch, Rainer, & Hellwagner. (2016). *SAF: Stochastic Adaptive Forwarding in Named Data Networking*. arXiv.org

Qian, H. (2013). Probability-based adaptive forwarding strategy in named data networking. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 1094-1101.

Rainer, B., Posch, D., & Hellwagner, H. (2016). Investigating the Performance of Pull-Based Dynamic Adaptive Streaming in NDN. *IEEE Journal on Selected Areas in Communications*, *34*(8), 2130–2140. doi:10.1109/JSAC.2016.2577365

Saxena. (2016). Named Data Networking: A Survey. *Elsevier Computer Science Review*.

Saxena, D., Raychoudhury, V., & Becker, C. (2017). Implementation and performance evaluation of name-based forwarding schemes in vndn. *Proceedings of the 18th International Conference on Distributed Computing and Networking*, 35.

Trossen, D., Sathiaseelan, A., & Ott, J. (2016). Towards an Information Centric Network Architecture for Universal Internet Access. *SIGCOMM Comput. Commun. Rev.*, *46*(1), 44-49.

Werfel, J. K., Xie, X., & Seung, H. S. (2004). Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Info. Proc. Syst.*, *131*, 57–65.

Yi, C., Afanasyev, A., Moiseenko, I., Wang, L., Zhang, B., & Zhang, L. (2013). A Case for Stateful Forwarding Plane. *Computer Communications*, *36*(7), 779–791. doi:10.1016/j.comcom.2013.01.005

Yuan, H., Song, T., & Crowley, P. (2012). Scalable NDN forwarding: Concepts, issues and principles. *Proceedings of 21st International Conference on Computer Communications and Networks (ICCCN)*, 1-9. doi:10.1109/ICCCN.2012.6289305

Zhang, Y., Bai, B., Xu, K., & Lei, K. (2018). IFS-RL: An Intelligent Forwarding Strategy Based on Reinforcement Learning in Named-Data Networking. *Proceedings of the 2018 Workshop on Network Meets AI & ML - NetAI'18*, 54-59.

Abderrahim Maizate received his Engineering Diploma in Computer Science from the Hassania School of Public Works since 2004 and DESA degree from ENSIAS in 2007. He received with honors the Ph.D. degree in Computer Networks and Telecommunications from Chouaib Doukalli university in El jadida in 2014. Since October 2014 he has been working as a Professor at the Computer Engineering Department at the Higher School of Technology of Casablanca (ESTC-UH2C). He is currently the Deputy Director of the Network, Computer, Telecommunications and Multimedia Laboratory at the same School and member of the research center TIC-DEV. He is also an IEEE member. His research interests include fields such as Wireless communications, WSN, smart cities, NDN, Big-data, cloud computing and security. He is also a TPC member and a reviewer for many international conferences. He is a member of the publishing committee of the Mediterranean Telecommunication Journal and member of the organizing committee of the international conferences SysCo 16 and CMT'2018.

Mohammed Ouzzif is a full professor at ESTC (Higher School of Technology) Hassan II University in Casablanca. He received his PhD in 1999 and his national PhD in 2005 in the field of formal description and verification of distributed systems. Currently, his research interests are in distributed systems, adhoc networks and collaborative systems.

Charif Mahmoudi received the MSc and PhD degrees in computer engineering from the University of Paris-EST (France) in 2009 and 2014, respectively. After His PostDoc at the National Institute of Standards and Technology, he joined Siemens Corporate Technology as an IoT Software Architect. He participated as consultant then software architect to several successful telecommunication projects within France Telecom and Bouygues Telecom and contributed to several research projects with deployments in several countries. His areas of research are on distributed systems, cloud computing, mobile computing and Internet of things.