This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2912365, IEEE Wireless Communications Letters

Distributed Deep Learning at the Edge: A Novel Proactive and Cooperative Caching Framework for Mobile Edge Networks

Yuris Mulya Saputra, Dinh Thai Hoang, Diep N. Nguyen, Eryk Dutkiewicz, Dusit Niyato, and Dong In Kim

***Abstract*—We propose two novel proactive cooperative caching approaches using deep learning (DL) to predict users' content demand in a mobile edge caching network. In the first approach, a content server (CS) takes responsibilities to collect information from all mobile edge nodes (MENs) in the network and then performs the proposed DL algorithm to predict the content demand for the whole network. However, such a centralized approach may disclose the private information because MENs have to share their local users' data with the CS. Thus, in the second approach, we propose a novel distributed deep learning (DDL) based framework. The DDL allows MENs in the network to collaborate and exchange information to reduce the error of content demand prediction without revealing the private information of mobile users. Through simulation results, we show that our proposed approaches can enhance the accuracy by reducing the root mean squared error (RMSE) up to 33.7% and reduce the service delay by 47.4% compared with other machine learning algorithms.**

***Index Terms*—Mobile edge caching, distributed deep learning, proactive and cooperative caching.**

I. Introduction

MOBILE edge caching (MEC) aims to distribute popular contents closer to the mobile users via mobile edge nodes (MENs) [1] to reduce the service delay for the mobile users. To efficiently cache the popular contents, proactive caching [2] has emerged as one of the most effective methods to increase the cache hit rate and reduce the operational as well as service costs for the MEC service providers [3].

In [4], a learning based proactive caching using singular value decomposition (SVD) was investigated. In this work, the data is first collected from the base stations and then trained in a big data platform. Nevertheless, the SVD sets all missing values to be undefined, leading to a poor prediction, especially when a dataset is extremely sparse. Furthermore, the SVD observes approximated ranks of elements and thus may produce negative numbers which provide no information about users' demands. To address this problem, the authors in [5] adopted the non-negative matrix factorization (NMF) to predict content request probabilities of mobile users based on the implicit feedback on their social connections. As such, the NMF applies the additive parts-based representation with non-negative elements to enhance the interpretability of the elements. However, the NMF is a linear model which considers only two-factor correlation (i.e., the user-content relationship) without learning multi-level correlation. Given that, deep learning seems a suitable solution that relies on deep neural networks (DNN) to learn multiple levels of processing layers. Each layer of the DNN provides nonlinear transformations of the complex hidden features to obtain correlations between the mobile users and the content demands hierarchically [6].

In this letter, we introduce two novel proactive cooperative caching approaches using deep learning (DL) algorithms to improve the accuracy of content demand prediction for the MEC network and deal with the dynamic users' demands based on the most up-to-date information. In the first approach, we develop a model utilizing the content server (CS) as a centralized node to collect information from all the MENs. We then use the DL to predict the demands for the whole network. This approach is especially useful when MENs have limited computing resources and cannot perform the DL algorithms by themselves. However, such an approach may raise the concerns on information privacy and communication overhead. To address these problems, we propose the novel approach using distributed deep learning (DDL) based framework. In this framework, the CS only needs to collect the trained models from MENs and update the global model accordingly [7]. After that, the global model will be sent back to the MENs for further updates. Through simulation results, we demonstrate that both proposed approaches can improve the accuracy of prediction up to 33.7% and reduce the service delay by 47.4% compared with other proactive caching algorithms at MENs.

II. System Model

A. Network Architecture

The proposed network architecture is illustrated in Fig. 1. Mobile users are connected to MENs within their service area. All MENs are also connected to the CS through the backhaul links by using either wireless (i.e., cellular networks) or wired connections. When a user sends a content request to an MEN, the content will be sent to the user instantly if the content is stored locally at the MEN. Otherwise, the MEN downloads the content from the CS or from one of its directly connected MENs, and sends it to the requesting user. Let $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ denote the set of MENs and $\mathcal{U} = \{1, \ldots, u, \ldots, U\}$ denote the set of mobile users in

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) under grant 2017R1A2B2003953.

Y. M. Saputra and D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz are with University of Technology Sydney, Australia (email: yurismulya.saputra@student.uts.edu.au, Hoang.Dinh, Diep.Nguyen, and Eryk.Dutkiewicz@uts.edu.au).

D. Niyato is with Nanyang Technological University, Singapore (email: dniyato@ntu.edu.sg).

D. I. Kim is with Sungkyunkwan University, South Korea (e-mail: dikim@skku.ac.kr).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2912365, IEEE Wireless Communications Letters

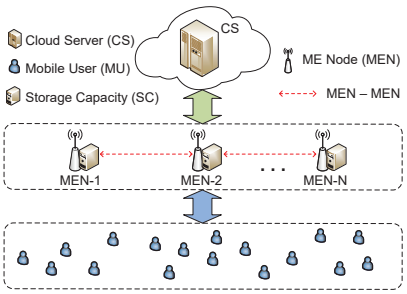


Fig. 1: Network architecture.

the network, where U is the total number of mobile users. Each MEN-n is equipped with a finite storage capacity S_n to cache popular contents locally according to the decision of caching framework. Additionally, each MEN-n has a vector $\boldsymbol{\mu}_n$ to monitor the users who did visit and download contents at MEN-n. Specifically, $\boldsymbol{\mu}_n = [\mu_n^1, \ldots, \mu_n^u, \ldots, \mu_n^U]$, where μ_n^u is a binary variable with $\mu_n^u = 1$ indicating that user u did visit MEN-n to download contents, and $\mu_n^u = 0$ otherwise. To obtain the total number of users who did visit and download contents at MEN-n, we define $U_n = \sum_{u=1}^{U} \mu_n^u$. Then, we denote $\mathcal{I} = \{1, \ldots, i, \ldots, I\}$ as the set of contents.

B. Proactive Cooperative Caching Mechanism

To cache popular contents, each MEN-n collects information from mobile users in its coverage area and sets up a dynamic log file, i.e., $\mathbf{X}_n$, containing a table with user ID and requested content ID in each column and row, respectively. Each element of the table, denoted by $f^i_{\mu_n^u=1}$, represents the number of times (i.e., popularity factor) that user u downloads content i at MEN-n. If the user u visits MEN-n to download the content i, the value of $f^i_{\mu_n^u=1}$ will be increased by one. This information will be captured and updated locally in the log file of each MEN.

For the first approach (i.e., using DL in the CS), the CS collects $\mathbf{X}_n, \forall n \in \mathcal{N}$ from the MENs cooperatively and then concatenates them into dataset $\mathbf{X}_{cs}$ vertically with popularity factor f_u^i of the user u over the content i. In this way, MENs can share the model information to improve the prediction accuracy for the whole network. We use $\mathbf{X}_{cs}$ to predict the content demands and then generate dataset $\hat{\mathbf{Y}}_{cs}$ containing predicted popularity factors $\hat{f}_u^i$ at the CS. This $\hat{\mathbf{Y}}_{cs}$ is then sent back to the MENs for content placement decision. Specifically, each MEN-n obtains $\hat{f}_n^i = \sum_{u \in \mathcal{U}} \hat{f}_u^i$ as the predicted popularity factor aggregation of content i.

In the second approach with the DDL, each MEN-n can predict the demands locally using $\mathbf{X}_n$. Then, the CS only needs to collect the trained models from MENs and update the global model cooperatively (explained in Section IV) and create $\hat{\mathbf{Y}}_n$ which contains predicted popularity factors $\hat{f}^i_{\mu_n^u=1}$. To perform the content placement decision, each MEN-n aggregates the predicted popularity factors of content i as $\hat{f}_n^i = \sum_{u \in \mathcal{U}} \hat{f}^i_{\mu_n^u=1}$. Based on $\hat{f}_n^i$ of the first and second approaches, we can obtain specific largest numbers of $\hat{f}_n^i$ at MEN-n in descending order. In particular, we select the contents with top-R of $\hat{f}_n^i$ which are likely to be cached at MEN-n.

III. DL-Based Proactive Cooperative Caching

In this approach, the CS needs to learn $\mathbf{X}_{cs}$ through the DNN by partitioning $\mathbf{X}_{cs}$ into smaller subsets (referred to as mini-batch size β). For DNN, each layer ℓ produces an output matrix containing global weight matrix $\mathbf{W}_\ell$ to control how strong the influence of a layer's each neuron to the other, and global bias vector $\mathbf{v}_\ell$ to fit the dataset as follows:

$$\mathbf{Y}^\ell_{cs} = \alpha_{cs}\big(\mathbf{W}_\ell \mathbf{X}^\ell_{cs} + \mathbf{v}_\ell\big), \tag{1}$$

where $\mathbf{X}^\ell_{cs}$ is the input matrix (i.e., training dataset) of layer ℓ in the CS (with $\mathbf{X}^1_{cs} = \mathbf{X}_{cs}$) and α_{cs} is the *rectified linear unit (ReLU)* activation function to transform the input of the layer into a nonlinear form for learning more complex feature interaction patterns. As the DNN contains several layers including the hidden layers, we can express $\mathbf{X}^{\ell+1}_{cs} = \mathbf{Y}^\ell_{cs}$. To prevent the overfitting problem and the generalization error [8], we augment a dropout layer ℓ_{drop} just after the last hidden layer. This additional layer randomly drops the input $\mathbf{X}^{\ell_{drop}}_{cs}$ by a fraction rate r, and thus the rest of the input elements are scaled by $\frac{1}{1-r}$. Then, the output layer L will generate $\mathbf{Y}^L_{cs}$ which is used to find the prediction loss for each mini-batch iteration τ. In particular, if we consider $\omega = (\mathbf{W}, \mathbf{v})$, where $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}\ell, \ldots, \mathbf{W}_L]$ and $\mathbf{v} = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \ldots, \mathbf{v}_L]$, as the global model for all DNN layers, the prediction loss $p(\omega_\tau)$ for one τ in the CS is expressed by the mean-squared error (MSE) $p(\omega_\tau) = \frac{1}{\beta}\sum_{u=1}^{\beta} p_u(\omega_\tau)$, where $p_u(\omega_\tau) = (y^u_{cs} - x^u_{cs})^2$. Here, x^u_{cs} and y^u_{cs} are the elements of matrices $\mathbf{X}^1_{cs}$ and $\mathbf{Y}^L_{cs}$, respectively. Then, we can compute the global gradient of using DL by $G_\tau = \nabla\omega_\tau = \frac{\partial p(\omega_\tau)}{\partial \omega_\tau}$.

After G_τ is obtained, the CS updates the global model ω_τ with the aim to minimize the prediction loss function, i.e., $\min\limits_{\omega} p(\omega)$. As such, we adopt the adaptive learning rate optimizer *Adam* to provide fast convergence and profound robustness to the model [9]. Consider η_τ and δ_τ to be the exponential moving average (to estimate the mean) of the G_τ and the squared G_τ to predict the variance at τ, respectively. Then, the update rules of $\eta_{\tau+1}$ and $\delta_{\tau+1}$ can be expressed by:

$$\eta_{\tau+1} = \gamma^\tau_\eta \eta_\tau + (1-\gamma^\tau_\eta) G_\tau, \text{and } \delta_{\tau+1} = \gamma^\tau_\delta \delta_\tau + (1-\gamma^\tau_\delta) G^2_\tau, \tag{2}$$

where γ^τ_η and $\gamma^\tau_\delta \in [0, 1)$ represent the exponential decay steps of η_τ and δ_τ at τ, respectively. To update the global model, we also consider the learning step λ to decide how fast the global model will be updated at each τ. In particular, the update rule for λ follows this expression $\lambda_{\tau+1} = \lambda \frac{\sqrt{1-\gamma^{\tau+1}_\delta}}{1-\gamma^{\tau+1}_\eta}$. Then, the global model $\omega_{\tau+1}$ for the next $\tau+1$ is updated by:

$$\omega_{\tau+1} = \omega_\tau - \lambda_{\tau+1} \frac{\eta_{\tau+1}}{\sqrt{\delta_{\tau+1}} + \epsilon}, \tag{3}$$

where ϵ indicates a constant to avoid zero division when the $\sqrt{\delta_{\tau+1}}$ is almost zero. For this approach, $\omega_{\tau+1}$ is used to learn the dataset for the next $\tau+1$ in the CS. The same process is repeated until each sample u of $\mathbf{X}_{cs}$ has been observed referred to as epoch time t. Then, the process terminates when the prediction loss converges or the certain number of epoch time T is reached. In this case, we can obtain the final global model ω^* to predict $\hat{\mathbf{Y}}_{cs}$ of training dataset $\mathbf{X}_{cs}$ and new dataset $\hat{\mathbf{X}}_{cs}$ using Eq. (1). The algorithm for proactive cooperative caching

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2912365, IEEE Wireless Communications Letters

using DL is shown in Fig. 2 in which the process inside the dotted block (A) is executed at the CS.

IV. DDL-Based Proactive Cooperative Caching

In this approach, each MEN distributedly implements the DL technique to learn from its dataset $\mathbf{X}_n$ locally. The $\mathbf{X}_n$ is then divided into smaller subsets with mini-batch size $\frac{\beta}{N}$. For DNN, each MEN-n generates the output matrix

$$\mathbf{Y}_n^{\ell} = \alpha_n\big(\mathbf{W}_{\ell}\mathbf{X}_n^{\ell} + \mathbf{v}_{\ell}\big), \tag{4}$$

where $\mathbf{X}_n^{\ell}$ is the input matrix of layer ℓ at MEN-n (with $\mathbf{X}_n^1 = \mathbf{X}_n$) and α_n is the *ReLU* activation function at MEN-n. We also drop the input $\mathbf{X}_n^{\ell_{drop}}$ in the dropout layer by a fraction rate r. In the output layer, we can generate $\mathbf{Y}_n^L$ and find the prediction loss for each τ by $p_n(\omega_\tau) = \frac{N}{\beta}\sum_{u=1}^{\frac{N}{\beta}} p_n^u(\omega_\tau)$, where $p_n^u(\omega_\tau) = (y_n^u - x_n^u)^2$. Here, x_n^u and y_n^u are the element of matrices $\mathbf{X}_n^1$ and $\mathbf{Y}_n^L$ at MEN-n, respectively. Next, we can compute the local gradient by $g_n^\tau = \nabla\omega_\tau = \frac{\partial p_n(\omega_\tau)}{\partial \omega_\tau}$. When g_n^τ computation is completed for each τ, each MEN will send this local gradient to the CS for global gradient aggregation G_τ. Specifically, the CS acts as a parameter server to aggregate the gradients of the models from all connected MENs and then update the global model ω_τ by using Eq. (3) before sending back to the MENs. Doing so allows all MENs to collaborate by sharing local model information to each other to further improve the prediction accuracy through the CS. To guarantee that the gradient staleness is 0, the gradient averaging process is enabled right after N local gradients, i.e., g_n^τ are received by the CS synchronously. Here, the gradient staleness happens when local gradients are computed using an obsolete/non-latest global model. Then, the global gradient G_τ of the DDL is $G_\tau = \frac{1}{N}\sum_{n=1}^{N} g_n^\tau$.

To minimize the prediction loss function, i.e., $\min_{\omega} p_n(\omega)$, at each MEN-$n$, we also adopt the Adam optimizer and update the global model $\omega_{\tau+1}$ as expressed in Eqs. (2)-(3). This $\omega_{\tau+1}$ is then sent back to the MENs for the next local learning process. The aforementioned process continues until the prediction loss converges or T is reached. We then can predict $\hat{\mathbf{Y}}_n$ of training dataset $\mathbf{X}_n$ and new dataset $\hat{\mathbf{X}}_n$ at each MEN using ω^* through Eq. (4). The algorithm for proactive cooperative caching using DDL is summarized in Fig. 2. The process inside the dotted block (B) is implemented at the CS.

Overall, for centralized DL-based caching algorithm, all dataset is sent to the CS for the learning process, and thus MENs can cache the globally most popular contents based on the global knowledge in the network. On the other hand, for DDL-based caching algorithm, the dataset is learned locally with some exchanged information shared among MENs. Thus, the MENs can cache locally most popular contents based on the global information sharing.

V. Performance Evaluation

A. Experimental Setup

We evaluate the performance with one CS and six MENs using *TensorFlow CPU* in Intel Xeon Gold 6150 2.7GHz 18 cores with 180GB RAM. We compare our proposed methods

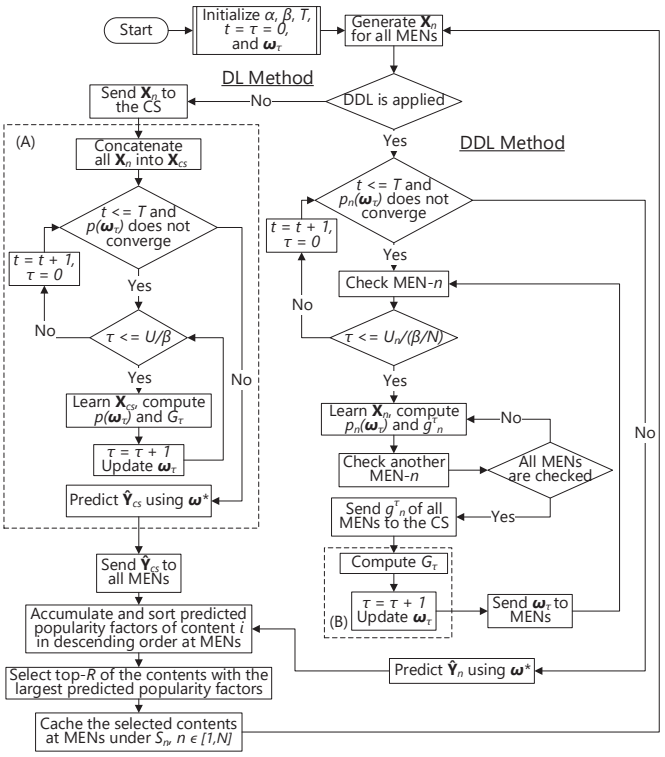


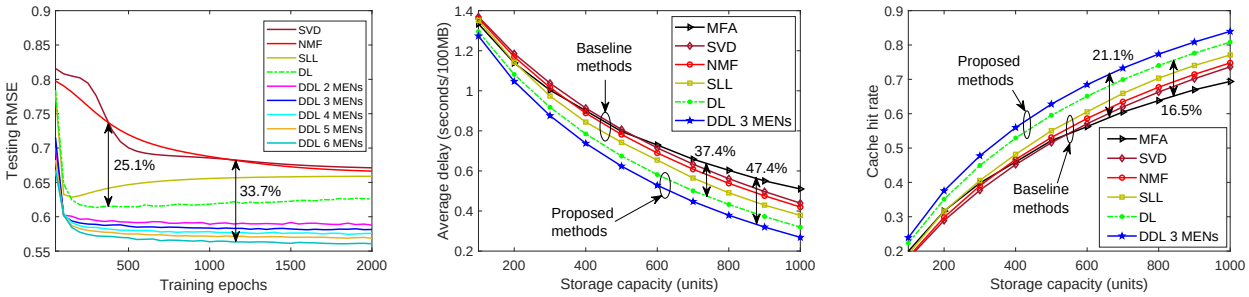
Fig. 2: Flowchart of DL and DDL approaches.

with baseline methods including non-prediction method, i.e., most frequency of access (MFA), as well as three well-known machine learning methods including SVD [4], NMF [10], and single layer neural network learning (SLL). We use Movielens 1M-dataset with more than 1M ratings from 6040 users with 3952 movies. Then, we split the dataset into 80% training dataset and 20% testing dataset. From the training dataset, we divide the number of samples equally with respect to the number of MENs when DDL is implemented. Each MEN runs the testing dataset for the popularity factor prediction. For DNN, we use two hidden layers with 64 neurons per layer and one dropout layer with a fraction rate 0.8. We also apply the adaptive learning rate Adam optimizer with initial step size 0.001 and 2000 epoch time. Furthermore, we consider the same size for each content at 200MB. The bandwidth between an MEN and the CS is set at 60Mbps.

B. Simulation Results

Fig. 3 shows the comparison between baseline and proposed methods. We first evaluate the prediction accuracy, i.e., RMSE, as the learning epoch increases in Fig. 3(a). In particular, the RMSE obtained by the DDL is 33.7% lower than those of SVD and NMF, and 17.5% lower than that of SLL. The reason is that DDL can deeply learn the meaningful features from the subset of the whole dataset independently at different MENs, and thus the sensitivity to learn new testing dataset becomes better when the local models obtained by the MENs are aggregated together. In other words, the average prediction of all MENs will produce less variance and lower error regarding the number of MENs [11]. Finally, the RMSE will converge at 0.56 when the number of MENs keeps increasing. In contrast,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2912365, IEEE Wireless Communications Letters



(a) RMSE vs prediction methods (b) Average delay vs storage capacity (c) Cache hit rate vs storage capacity

Fig. 3: The performance comparison of the baseline and proposed methods.

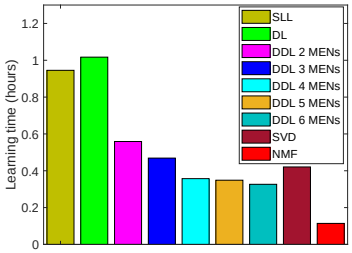


Fig. 4: Learning 2000 epoch time for various methods.

SVD and NMF only generate linear assumptions of two factors based on the low-rank approximation [10], while SLL learns at the absence of hidden layers. As such, without deeply learning the representations, those learning methods cannot minimize RMSE properly. For the DL, although the RMSE is higher than that of the DDL, the DL can improve the RMSE by as much as 25.1% compared with those of the SVD and NMF as well as 5.3% compared with that of the SLL.

We then observe the average communication delay (i.e., the delay when requested contents are not cached at the MENs) and cache hit rate when the storage capacity increases in Figs. 3(b) and 3(c), respectively. Align with the trend of the RMSE, the DL and DDL approaches can reduce average delay up to 37.4% and 47.4% and increase the cache hit rate by 16.5% and 21.1%, respectively, compared with those of other baseline methods. The reason is that the proposed approaches can optimize the use of hyperparameter settings to improve the accuracy of content demand prediction. Examples of the hyperparameters settings include the number of hidden layers and neurons, the regularization methods, the activation functions, and the size of mini-batch. Furthermore, the use of DDL method with 3 MENs and MovieLens 1M-dataset can reduce the communication overhead up to 87.2% compared with the SVD, NMF, SLL, and DL. The reason is that, in DDL method, the CS only needs to collect the trained models from MENs without considering any raw dataset transmission. This benefit aligns with the reduction of user's private information disclosure.

We also observe in Fig. 4 that the learning time for NMF is very short. However, its performance in terms of the prediction accuracy is very poor compared with all other learning methods. Furthermore, the DDL can learn the dataset faster than the DL as the number of the MENs increases. The learning time then will converge to 0.3 hours if we keep increasing the number of MENs. This interesting trend can provide useful information for MEC service providers to tradeoff between the learning time of the users' demands and the implementation costs in the MEC network.

VI. Summary

In this letter, we have presented two novel proactive cooperative caching approaches leveraging deep learning (DL) algorithms for the MEC network. In the first approach, the CS collects the information from all MENs and uses the DL technique to predict the users' demands for the network. Then, we proposed the distributed DL (DDL)-based scheme in which the DL can be executed at the edge. This scheme allows MENs to only exchange the gradient information of the users, and perform the DL to predict users' demands without revealing the private information of the mobile users.

References

[1] Y. Mao *et al.*, "A survey on mobile edge computing: the communication perspective", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, Fourth Quarter 2017, pp. 2322-2358.

[2] S. Shukla *et al.*, "Proactive Retention-Aware Caching With Multi-Path Routing for Wireless Edge Networks", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, Jun. 2018, pp. 1286-1299.

[3] D. T. Hoang *et al.*, "A dynamic edge caching framework for mobile 5G networks", *IEEE Wireless Communications*, vol. 25, no. 5, Oct. 2018, pp. 95-103.

[4] E. Zeydan *et al.*, "Big data caching for networking: moving from cloud to edge", *IEEE Communications Magazine*, vol. 54, no. 9, Sept. 2016, pp. 36-42.

[5] J. Ahn, S. H. Jeon, and H. Park, "A novel proactive caching strategy with community-aware learning in CoMP-enabled small-cell networks", *IEEE Communications Letters*, vol. 22, no. 9, Sept. 2018, pp. 1918-1921.

[6] S. Zhang *et al.*, "Deep learning based recommender system: a survey and new perspectives", *ACM Computing Surveys*, vol. 1, no. 1, Jul. 2018, pp. 1-35.

[7] J. Dean *et al.*, "Large scale distributed deep networks", *ACM NIPS 2012*, Dec. 2012, pp. 1223-1231.

[8] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, Jan. 2014, pp. 1532-4435.

[9] D. Kingma and J. Ba, "Adam: a method for stochastic optimization", *ICLR 2015*, May 2015.

[10] Y. Mao, L. K. Saul, and J. M. Smith,"IDES: an internet distance estimation service for large networks", *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, Dec. 2006, pp. 2273-2284.

[11] Y. Guo and J. Sutiwaraphun, "Probing knowledge in distributed data mining", *PAKDD 1999: Methodologies for Knowledge Discovery and Data Mining*, vol 1574, pp. 443-452, Springer, Berlin, 1999.