IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 28, 2021, accepted May 6, 2021, date of publication May 14, 2021, date of current version May 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3080512

DCoL: Distributed Collaborative Learning for Proactive Content Caching at Edge Networks

SUBINA KHANAL, KYI THAR, AND EUI-NAM HUH, (Senior Member, IEEE)
Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, South Korea

Corresponding author: Eui-Nam Huh (johnhuh@khu.ac.kr)

This work was supported by the Institute for Information and Communications Technology Planning and Evaluation (IITP) Grant by the Korean Government through the Ministry of Science and ICT (MSIT) (Service mobility support distributed cloud technology) under Grant 2017-0-00294.

ABSTRACT Caching popular content at the network edge, such as roadside units (RSUs), is a promising solution that enhances the user's quality-of-experience (QoE) and reduces network traffic. In this regard, the most challenging issue is to correctly predict the future popularity of contents and effectively store them in the cache of edge nodes. Thus, in this paper, we propose a distributed proactive caching scheme at the edge to optimize the content retrieval cost and improve the QoE of the mobile users. This proactive content caching scheme, namely Distributed Collaborative Learning (DCoL), is a non-parametric content popularity prediction mechanism in a distributed setting. Next, we show the advantage of DCoL as two folds: (i) it leverages distributed content popularity information to develop local content caching strategy, and (ii) it exploits the regional database using the long short-term memory (LSTM)-based prediction model to capture the dependency between requested contents. Simulation results using real datasets demonstrate that our scheme yields 8.9% and 18% gains, respectively, in terms of the cache hit efficiency and content retrieval cost, compared with a competitive centralized baseline, and outperforms other traditional caching strategies.

INDEX TERMS Proactive content caching, mobile edge computing, distributed learning, collaborative filtering, neural network.

I. INTRODUCTION

In a recent study made by Cisco highlights that mobile traffic generated by videos is expected to reach 75% of the total generated traffic by 2020.[1] This significant rise in mobile traffic causes user latency to increase and puts a heavy burden on backhaul links that connect local base stations and the internet. In this regard, multi-access edge computing (MEC) paradigm has been developed to cater to this unprecedented growth of network traffic, both in terms of communication and computation, at the network edge [1]. In addition, the availability of MEC servers at the network edge has unleashed opportunities for several next-generation services, including edge caching, control, and computation. One of which, edge caching is a promising solution to cope with the exponential growth of data from different internet of things (IoT) devices in edge computing, where content is usually placed on local caches for fast and repetitive access to data [2]. In this regard, it is observed that only a few popular contents are responsible for most of the traffic load, and are requested by several users at different times. Thus, with predicted content popularity, it is advantageous to cache popular contents locally before the requests truly arrive, directly at edge networks, e.g., Roadside Units (RSUs). Here, the popularity of a content is defined as the ratio of the number of requests for a particular content to the total number of requests from users, which is usually obtained for a certain region during a given period of time. Then, such content popularity prediction can be used to minimize the content retrieval cost of edge network, which is the latency incurred for fetching contents from the content server located in the remote cloud, by proactively caching the popular content before the users actually request it. However, developing proactive caching strategies and determining pre-caching contents to serve the selected users before they actually request it is a non-trivial problem as there exist challenge to deal with the dynamic pattern of user requests.

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

[1] http://tubularinsights.com/2020-mobile-video-traffic/

In addition, given the difficulty to appropriately capture user preference profiles over time, the cache space limitation at edge nodes makes it imperative to first predict the popularity of contents, and then cache the most common ones in advance [3]. Thus, it has been a challenge to design the content caching strategy that fulfills the requirements to meet quality-of-experience (QoE) of users. In this regard, traditional caching algorithms such as First-In-First-Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU) [4] do not focus on the potential popularity of the content, which in result, leads to low cache effectiveness and large cache misses. Further, though being implemented as a distributed solution, these approaches avoid the use of beneficial collective information on content requests made by the users to edge nodes; and thus, ignoring the significant notion of local content popularity and regional popularity.

To tackle these issues, we propose a novel proactive content caching scheme, namely Distributed Collaborative Learning (DCoL), where we cache the next most popular content in a distributed manner. DCoL is a non-parametric content popularity prediction mechanism in a distributed setting, where edge networks at first share knowledge about their local popular contents with the macro base station (MBS) to build a regional popularity database. Then, the MBS rework on the constructed regional database to develop proactive content caching strategies at the network edge. In particular, we develop DCoL to solve the formulated optimization problem where the objective is to find a subset of content for caching to minimize content retrieval cost and improve cache hit efficiency.

The main contributions of the paper are summarized as follows:

- We investigate the problem of proactive content caching approach at the network edge with limited cache space. To that end, we formulate an optimization problem with the objective to find contents to be cached proactively at the network edge that minimize the network-wide latency during content request for the given time window.
- We propose a distributed collaborative learning (DCoL) based popularity prediction mechanism to solve the formulated problem that integrates local prediction models to create a regional content popularity database.
- We combine item-based collaborative filtering and long short-term memory (LSTM) to design a proactive content caching algorithm at the network edge. To that end, we propose a non-parametric algorithm to meet the objective of the formulated optimization problem.
- Experimental results based on real-world Movie Lens datasets [5] verify that the DCoL outperforms other well-known reference algorithms in terms of the cache hit efficiency, content retrieval cost, and user-satisfaction level.

The rest of the paper is organized as follows. Section II reviews the related works, where several works on edge caching and its application are discussed. Section III presents our proposed system model of distributed collaborative learning for proactive content caching in which we show the formulated optimization problem, discuss overview of solution approach, and present preliminaries of adopted collaborative filtering and LSTM model. Section IV discuss the details of our proposed approach, and present a non-parametric algorithm to tackle the formulated problem. Section V provides the performance evaluation of the proposed approach and compares with other traditional approaches using real-world datasets. Finally, Section VI concludes this work.

II. RELATED WORKS

It has been a challenge to efficiently manage content caching strategy at the network edge due to two primary reasons: (i) the limited cache space at the network edge, and (ii) the dynamics of content popularity over time. Therefore, it is imperative to acknowledge both the future popularity prediction and the section of contents that are most likely to be requested by the users in the local cache. In this regard, to tackle the problem of limited cache availability, most of the existing caching systems often use simple cache replacement algorithms such as LRU, FIFO, and LFU [4], [6], that update the cache during the content delivery phase. However, these conventional algorithms do not take into account the popularity of potential content, and therefore, are not suitable caching strategies expected to effectively capture the rapidly evolving popularity of content in true sense and make caching decisions. To that end, recent works discuss to develop innovative cache replacement algorithms by learning trends of the popularity of contents [7]–[11]. While recent distributed machine learning approaches such as Federated Learning (FL) [12]–[15] are still in their infancy to learn distributed content popularity trend, particularly due to model complexity and convergence issues, several works opt to low-complexity solutions for jointly addressing cache limitation issues and content replacement policies. In [16], the authors show deep learning-based caching strategies are more efficient and can improve the performance of caching strategy as they can accurately predict content popularity. However, the underlying complexities of deep learning models are still overlooked. Similarly, in [17], [18], the authors consider user's mobility behaviors and social impacts on content preference that distort the performance of content requests. They propose a long-term strategy of proactive caching to minimize the sum of communication costs to get the requested contents. The authors in [19] utilize the concept of sequence prediction algorithm for mobility prediction on vehicle's routes to achieve proactive caching. Furthermore, using the historical popularity of videos of the autonomous vehicle, the authors in [20] use the non-negative matrix factorization (NMF) technique to first predict the preferences of users, which is then used to predict the future demands of the users. The authors in [21] adopt a Q-learning algorithm for proactive edge caching problem to minimize the content retrieval cost at edge nodes. Similarly, the authors in [22] uses methods like deep item-based collaborative filtering to

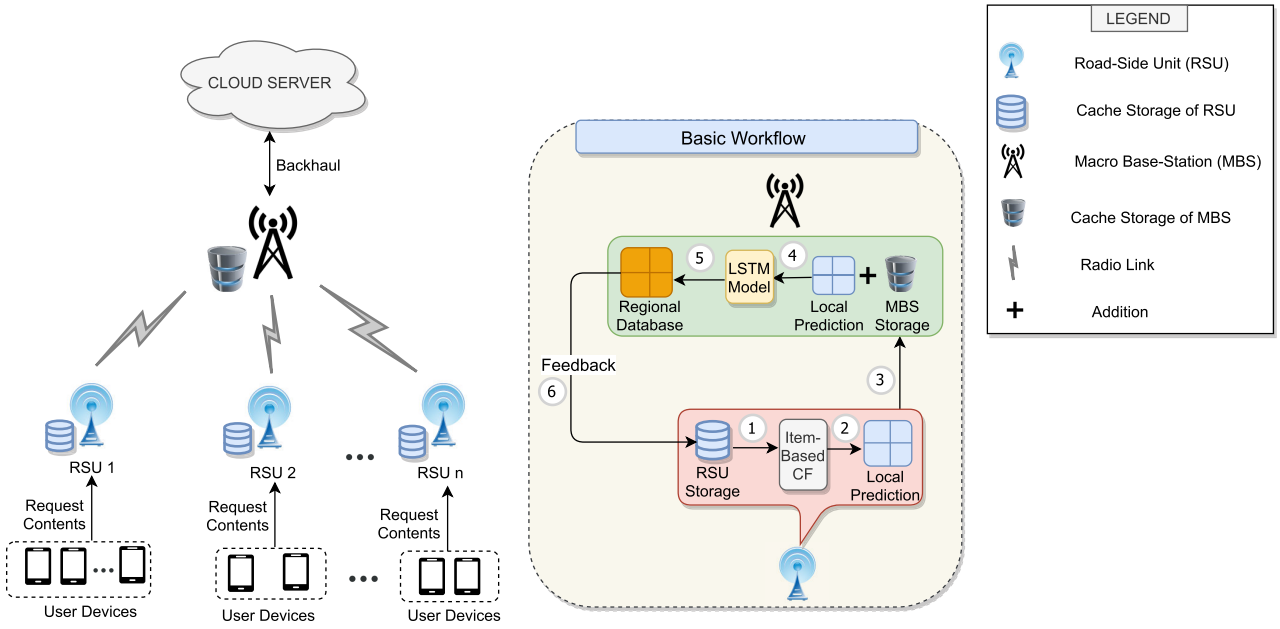

FIGURE 1. System model: DCoL.

model higher-order item relations, where multiple non-linear layers are used to learn those item relations. In line with these works, some approaches adopt personalized quality-of-service (QoS) values prediction for service users using an extended version of collaborative filtering techniques by incorporating additional information about services and users [23], [24]. The authors in [25] focus on mobility-aware hierarchical collaborative caching, where caching agents share their content caching tasks with the base stations. They also propose a vehicle-aided edge caching scheme, where the wireless network edge caching and computing services are jointly scheduled. In [10], [26], the authors use the concept of reinforcement learning for content caching to maximize the long-term cache hit rate. Besides, the authors in [27] exploit a transfer learning-based approach for estimating content popularity. Similar to which, in [28], the authors take user privacy into account to develop proactive content caching strategy. However, unlike the existing works that are mostly parametric and model dependent, we focus on a distributed, non-parametric proactive caching strategy at the network edge (RSUs) with limited cache storage. We take distributed content popularity trends into consideration, develop a regional popularity database, and exploit neural networks to feedback this information to the edge nodes for adjusting local caching scheme.

III. SYSTEM MODEL

The system model of DCoL is shown in Fig. 1 with a basic pictorial workflow. We consider a set of cache-enabled RSUs $\mathcal{R}$ of $|\mathcal{R}| = R$ associated with a macro base station (MBS) M via radio links. We define a set of contents $\mathcal{K}$ of $|\mathcal{K}| = K$, where the size of each content $k \in \mathcal{K}$ is defined as z_k. In the considered model, RSUs acts as an intermediary to serve the content requests made by the users. However, due to limited cache size at the RSUs, not all contents requested by the users are readily available; hence, the RSU will look at the MBS for the missing contents, and similarly fetch the requested content from the content server to meet the users' request. Let C_r denotes the cache capacity of RSU $r, \forall r \in \mathcal{R}$ and $x_k^r\{0, 1\}$ is a binary variable such that

$$x_k^r = \begin{cases} 0, & \text{if content } k \text{ is } \textit{not} \text{ cached at RSU } r, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

In this regard, if $x_k^r = 0$, we consider it as a cache miss, and therefore, the RSU r will bear additional content retrieval cost δ_k^r to serve users fetching contents from the content server located at the remote cloud, or the MBS. Particularly, content retrieval cost is defined as the backhaul usage function. That means, for a missed request of content x at RSU r, the content retrieval cost δ_k^r is the function of the content size z_k and the available link capacity Ω_k^r between the RUS r and the MBS,[2] i.e., $\delta_k^r = z_k / \Omega_k^r$. Mathematically, Ω_k^r can be defined as the standard Shannon rate function [3], [9] which is dependent on the available wireless resources, such as the transmit power and allocated bandwidth, and the channel gain, i.e., $\Omega_k^r = B_k \log_2(1 + \frac{\rho^r |h^r|^2}{n_0 B_k})$, where B_k is the available bandwidth to retrieve content k, p^r is the downlink transmission power, $|h^r|^2$ is the channel gain, and n_0 is the Gaussian noise power density. In fact, the MBS can appropriately determine δ_k^r; and thus, the distributed content caching strategy $x_k^r, \forall k \in \mathcal{K}, \forall r \in \mathcal{R}$, and the number of content requests made dominantly characterize the backhaul usage, i.e., the content retrieval cost. When $x_k^r = 1$, we have

[2]In fact, the content retrieval cost accounts the additional information overhead $\Phi(z_k) \ll z_k$, which be of small size (in bits) and common for all RSUs under the current setting of DCoL; and hence, can further be eliminated in the minimization problem, similar to [9], [13].

a cache hit in which the requested content would be directly delivered from the RSUs to the users. Then, at any time t, we can formally define the cache capacity constraint at each RSU as

$$\sum\nolimits_{k\in\mathcal{K}} x_k^r(t) z_k \leq C_r, \quad \forall r \in \mathcal{R}. \tag{2}$$

In *proactive caching*, at each RSU $r \in \mathcal{R}$, we aim to predict a subset of contents for a time window N in order to jointly improve cache hit ratio, optimize cache utilization, and minimize the content retrieval cost. To that end, we define the network-wide delay associated with cache miss at RSUs for the next $t+N$ time slot as

$$\phi_{\text{delay}}(t+N) = \sum\nolimits_{r=1}^{R} \sum\nolimits_{k=1}^{K} \Big((1 - x_k^r(t)) \delta_k^r \Big) \beta_k^r(t+N), \tag{3}$$

where $\beta_k^r(t+N)$ is the number of request counts for content k at RSU r at a N time spaced window. In (3), we observe the overall network delay associated with the contents $k \in \mathcal{K}$ requested by the users depends on the caching strategy $x_k^r(t)$, which is known to RSU r, and the number of future request counts $\beta_k^r(t+N)$, which is an unknown quantity. Note that $\beta_k^r(t+N)$ can be estimated following historical information about the prior request counts, or we can predict this information leveraging the distributed collaborative learning approach, which will be discussed in the Section IV.

Therefore, we formulate our optimization problem to minimize the network-wide latency during content request for the time window N as

$$\mathbf{P}: \underset{\boldsymbol{x}}{\text{Minimize}} \ \frac{1}{N} \sum\nolimits_{t=1}^{N} \phi_{\text{delay}}(t+N) \tag{4a}$$

$$\text{subject to} \sum\nolimits_{k\in\mathcal{K}} x_k^r(t) z_k \leq C_r, \quad \forall r \in \mathcal{R}, \ \forall t \in \mathcal{T}, \tag{4b}$$

$$\phi(x_k^r(t)) \leq \phi_{\text{th}}, \quad \forall r \in \mathcal{R}, \ \forall t \in \mathcal{T}, \tag{4c}$$

$$x_k^r(t) \in \{0,1\}, \quad \forall k \in \mathcal{K}, \ \forall r \in \mathcal{R}, \ \forall t \in \mathcal{T}, \tag{4d}$$

where $\boldsymbol{x}$ is a matrix which is mapping contents caching strategies for each RSU over the time slots, $\phi(x_k^r(t))$ is a shorthand representation of $\sum_{r=1}^{R} \sum_{k=1}^{K} (1 - x_k^r(t)) \delta_k^r) \beta_k^r(t+N)$, constraint (4b) denotes the cache capacity at each RSU, and respectively, constraint (4c) denotes the constraint imposed on the content retrieval cost to define the QoE for users when requesting contents with their associated RSUs.[3] In this regard, the term QoE equivalently translates into the proportion of users served by the RSU in the formulation, similar to [29]. We observe that the MBS can appropriately determine the content retrieval cost δ_k^r in the downlink, and then, satisfying the objective of minimizing (4a) boils down to first finding the contents request count, and then implementing the caching decision at the RSUs with the available cache space. However, as we have discussed before, it is challenging to solve the optimization problem (4a) for two particular reasons: (i) the request count of contents $\beta_k^r(t+N)$ is unknown, and (ii) the stochastic nature of combinatorial constraint (4d) makes the problem NP-hard. Moreover, the coupled constraints make it difficult to optimally find out the caching strategy beforehand. Hence, we propose a non-parametric distributed collaborative learning scheme, namely DCoL, to solve the formulated optimization problem. DCoL is designed to combine collaborative filtering technique in a distributed fashion at RSUs with LSTM[4] at the MBS to efficiently build local content caching strategy (detailed in Section IV), which obtains a near-optimal solution with low-complexity than the optimal solution to the optimization problem **P**. In particular, we decompose problem **P** into a distributed setting where we first find the local content popularity, and then, jointly determine the content caching strategy at the network edges. This approach facilitates different RSUs to independently find out which contents need to be proactively cached next by using its local historical data and regional database at the MBS, which is built with the collaboration of content request count information between the RSUs. This is inline with the fact that local content popularity follows *Zipf* distribution, and it is imperative to use collaborative approach amongst RSUs to reflect the timeliness of popularity in a region for designing content caching strategies.

Particularly, in our system model, users request contents to the nearby RSU, as in the illustrated cellular network (Fig. 1). These users can be mobile users, or any IoT device, which is a practical consideration. We also note that these requested contents differ with respect to place, time of request, and other features. Besides, the content request patterns may be similar or different depending on the situation of the user. The RSUs store these content request patterns as a historical data. To this end, we observe RSUs are responsible for recognizing these local content request patterns and working accordingly to manage local content caching strategy. Moreover, to recommend the popular contents in RSU level, RSUs exploit the principle of Item-based collaborative filtering, which is used to find the similarity between the items, to help in local content popularity prediction at each RSU. These contents are then cached in the local content space of each RSU. In addition, once getting top popular contents for all RSUs, they are forwarded to MBS so as to build a global estimate of content popularity by using LSTM. In this regard, the LSTM model input will be contents sent by each RSUs, the historical contents of MBS, and popularity count of each content. Here, the LSTM model predicts the next popularity count of each content, which helps MBS to determine the top most

[3]Note that we limit the scope of QoE evaluation at the RSU-level, and not the "user-level", similar to several existing works [3], [9]. This is mostly done to make the problem tractable, and further, minimizing the overall network-wide latency can enforce the formulated optimization problem to obtain caching decisions that satisfies delay requirements on each requested content.

[4]LSTM model is executed to find the popularity count of each item in the popular content lists obtained exploiting the distributed item-based collaborative filtering at the MBS.

popular contents to be cached. Then, MBS stores these top most popular content in its global content space, i.e., regional database. Finally, RSUs reuses the MBS's caching strategy following the global estimate of content popularity to update its caching strategy. And according to the developed caching plan, the n most popular contents are selected for proactive caching in each RSU.

Next, in the following subsections, we will discuss the details of collaborative filtering and LSTM models involved to execute distributed content popularity prediction mechanism in DCoL.

A. PRELIMINARIES: COLLABORATIVE FILTERING

In recommendation systems, collaborative filtering [30], [31] refers to a technology that predicts on its own, based on data obtained from multiple users. Collaborative filtering is focused on the fact that if a user A has the same opinion (interest) on one content as user B, they may share similar opinion on another content as well. Moreover, the availability of timestamp in historical information about the contents watched/requested by the users make it a practical tool to build preference history, behavior patterns, and item-of-interest. Hence, it is widely used in recommendation systems to improve business value of e-commerce platforms in which predicting the next-item for sale matters. In this regard, there exist two popular variants of collaborative filtering: (i) user-based, and (ii) item-based, which follows memory-based and model-based approaches to compute similarity between users or items [32].

In our proposed model, we use the item-based collaborative filtering [33], [34] concept, which is one of the neighborhood algorithms, to capture the degree of similarity between items. In particular, we adopted this method to exploit user's content request history as input and generate prediction of the upcoming requests for the contents. Item-based collaborative filtering approach looks for similar items based on the items preferences build with the information of items users have already liked, or positively interacted with. Then, it suggests items based on the interests of the users, i.e., the contents that have been previously consumed/requested. Accordingly, recommendation system works in this manner. In doing so, the very first step is to build the model by finding similarities between all the item pairs. In this regard, the similarity between item pairs can be found in different ways [32]. One of the most common methods is to use cosine similarity. In this case, two items are thought of as two vectors in the m-dimensional user-space. Then, the similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, considering the $m \times n$ ratings matrix, similarity between items i and j, which is denoted by $sim(i, j)$, is given as

$$sim(i, j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 \times \|\vec{j}\|_2}, \tag{5}$$

where "$\cdot$" denotes the dot-product of two vectors.

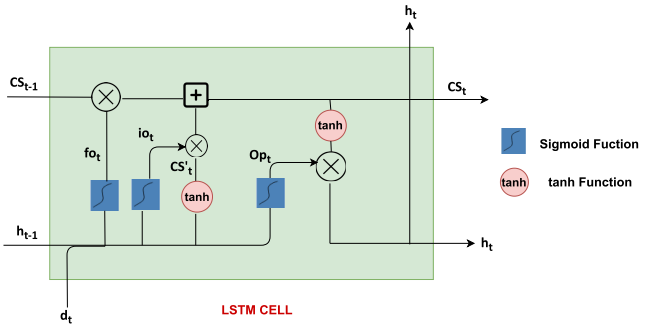

FIGURE 2. An illustration of LSTM cell.

Although collaborative filtering works well in some cases, it has limitation in providing real-time recommendations, primarily due to stale data and inability of collaborative filtering to capture short-term content request patterns [35]. Hence, to overcome this prominent challenge for developing proactive content caching strategies, we combine collaborative filtering with LSTM to determine the list of contents for proactive caching. In doing so, different from existing works, we adopt distributed collaborative filtering in which RSUs share local prediction information with the MBS, and MBS deploys LSTM models to develop proactive content caching at the RSU level.

B. LSTM MODEL

LSTM is a special type of Recurrent Neural Network (RNN) that can learning patterns on data with long dependence periods. RNNs are particularly designed to capture the temporal dynamics of input sequential data [36]. At the core of LSTM is the cell state, as illustrated in Fig. 2, which enables learning from input data sequence with only a small linear interaction. LSTM can add or remove something from the cell state, which is carefully controlled by a gate structure. Each LSTM cell includes gates, which are additional methods by which information can be stored and transferred. These gates consists of a sigmoid layer and point-wise multiplication, and allows to protect and control the cell state. A general LSTM cell has three specific gates: (i) input gate, (ii) output gate, and (ii) forget gate. As discussed, these three gates are intended to allow the flow of information selectively by discarding or retaining information at some point.

The major innovation of LSTM is its memory cell CS_t which essentially acts as an accumulator of the state information. The cell is accessed, written, and cleared by several self-parameterized controlling gates. Every time a new input comes, its information will be accumulated in the cell if the input gate io_t is activated. Also, the past cell status CS_{t-1} could be "forgotten" in this process if the forget gate fo_t is on. Then, whether the latest cell output CS'_t will be propagated to the final state h_t is further controlled by the output gate Op_t.

To that end, the forget gate decides whether to discard past information or not, i.e., to choose what information we're going to throw away from the cell state. As shown in Fig. 2, it takes h_{t-1} and d_t as inputs. Similarly, the *sigmoid* layer

emits the value between 0 and 1, representing how much information each component should convey. If the value is 0, the previous cell state values are all 0, which means "do not pass anything", and that way it does not affect the future result. If output value is 1, which means "hand over everything", we transfer all the values to the next step. Thus, mathematically, we represent fo_t as

$$fo_t = \sigma(W_f \cdot [h_{t-1}, d_t] + b_f), \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid activation function and W_f is the weight matrix to be learned.

The next step is to decide whether new information will be stored in cell state or not, which can be done following two steps. First, the sigmoid layer, also known as the input gate layer, is responsible for deciding which information we want to update. Then, the *tanh* layer creates a new cell state value CS'_t that can be added to the cell state. In the next step, these two values are added together that affect the next state, as follows:

$$io_t = \sigma(W_i \cdot [h_{t-1}, d_t] + b_i), \tag{7}$$

and

$$CS'_t = tanh(W_c \cdot [h_{t-1}, d_t] + b_c). \tag{8}$$

After evaluating these values, we update old cell state CS_{t-1} to a new cell state CS'_t. To do so, we first multiply old cell state by fo_t, where we forget the data that forget gate decided in the first step, and then add the new cell state to it, given as

$$CS_t = fo_t * CS_{t-1} + io_t * CS'_t. \tag{9}$$

This is how new cell state value affect the existing values. And finally, we have to decide which output to print. For this, we first run a sigmoid layer that determines which values need to pass to the output. Then, we will take the cell state through the *tanh* function and extract the value in the range $[-1, 1]$. Next, we will multiply this value by the output of sigmoid gate as

$$Op_t = \sigma(W_o \cdot [h_{t-1}, d_t] + b_o), \tag{10}$$

and

$$h_t = Op_t * \tanh(CS_t). \tag{11}$$

In our proposed model, we feed the timestamped historical content popularity counts as the input in the MBS to predict the sequence of next items, and accordingly develop a proactive content caching strategy at the RSU using DCoL scheme. In the following section, we present our non-parametric solution approach to solve the formulated optimization problem **P**, leveraging collaborative filtering and LSTM model in a distributed manner.

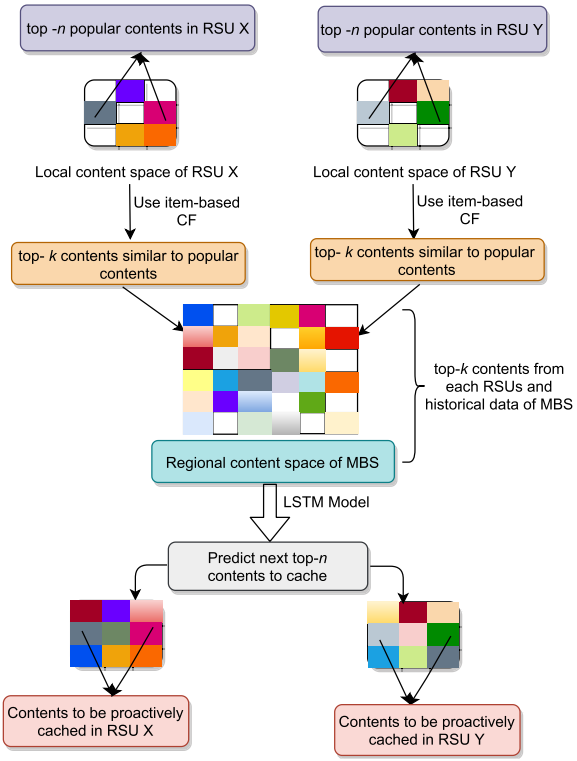

FIGURE 3. An illustration of DCoL for proactive content caching.

IV. PROPOSED METHOD: DCoL FOR PROACTIVE CONTENT CACHING

In this section, we propose the deployment design of distributed collaborative learning mechanism for proactive content caching at the network edge, DCoL. We first present the details of processes involved in finding out the content popularity count in a distributed fashion. Then, we discuss the proposed non-parametric algorithm to solve the presented problem description. In doing so, we consider a small network with one MBS and two RSUs, RSU X and RSU Y, which is usually a practical scenario for simulation [9].

In Fig. 3, we show an illustration of the proposed DCoL mechanism for proactive content caching. The distributed mechanism of DCoL runs in two coupled levels: **Level I**, where collaborative filtering works on the local content space of RSUs to build top-k similar movies of top-n popular contents, and **Level II**, where MBS reuses its own historical information and the lists shared by RSUs to build a global content space, and combines collaborative filtering and LSTM models to build proactive content caching strategies for RSUs. Here, local content space is defined as the local database of historical content request information at RSU level, whereas global content space signifies the regional database at MBS built with shared popular historical content request information from corresponding associated RSUs and its own data.

At first, each RSU X and Y gets content requests from different user devices based on their location. Primarily, RSUs need to have those contents stored to serve the

requesting users. However, due to the dynamic nature of user's preferences and the limited storage capacity of RSUs, not all requested contents are cached at the RSU; and therefore, users might not instantly receive the requested contents. For that reason, RSUs must download these contents from MBS in a proactive manner. Also, because of the high amount of content requests, it is impossible to cache all the replicas of contents in RSUs. Therefore, RSUs must select the contents which may be proactively cached to reduce the high delivery delay.

To that end, using historical data, we first find top-n most popular contents in both RSUs. In our case, popular contents are the contents with a maximum number of request count, i.e., the popularity count. These popularity count determines a list of popular contents in each RSU. Next, after getting the popular contents, each RSU can individually decide to cache them. However, caching only popular contents won't suffice for minimizing content retrieval cost as the requested content depend on user preferences and change over time. That is why we use item-based collaborative filtering to first find out the top-k similar contents of those top-n popular contents to know which contents we can cache proactively, other than popular content. For this purpose, we use widely-adopted *cosine* similarity metric for finding the similarity among contents. Then, considering the local content space, contents are cached based on each content's similarity score, i.e., if the similarity score is high, then these similar contents are likely to be proactively cached. However, given high mobility of the users around RSUs, it is impractical to solely adopt the local content popularity trend as a key metric to design proactive content caching strategy at the RSU level. Therefore, we need to build a regional-level observation instead of a RSU-level local observation in which we exploit the collaboration between RSUs and the MBS to have the distributed local content popularity information and the regional contents, and reuse this knowledge to adjust the local content caching strategies at the RSUs.

For that reason, top-k similar movies are sent to MBS, where MBS first build regional content database exploiting data of historical content requested, and information from corresponding RSUs. Then, MBS uses the LSTM model to predict the popularity count of the next popular contents to be cached, and further ask associated RSUs to update their local content caching scheme with the recommended contents. In this way, leveraging regional information of requested contents over the time period, both RSUs have knowledge of the popular contents of different, but nearby areas. Next, after getting the list of items to be cached proactively from MBS, each RSU update its local content space. In the case of MBS, it checks if these contents are available or not; if yes, it sends these contents directly to each RSU. Else, it requests to nearby content server for those contents to serve user requests. Since MBS's cache space is more significant than RSUs, it can proactively cache more contents in its global content space (i.e., the regional database). In Algorithm 1, we present the pseudocode of our proposed proactive content caching scheme. As aforementioned, each RSU obtains top-k similar contents of top-n popular contents, respectively, using popularity count and item-based collaborative filtering, and share the list with the MBS (line 5–7). Correspondingly, they update their local content space with similar contents based on the obtained similarity scores (line 8). At the MBS, LSTM model is used to predict top-n contents and prepare contents for proactive caching at the RSU level (line 13–24). Based on these information, RSUs update their local content space (line 25), and corresponding rework to determine top-k similar contents of top-n popular contents to share with MBS in the next round.

Algorithm 1 DCoL for Proactive Content Caching

1: **Level I:** Collaborative filtering for obtaining top-k items similar to top-n popular content at each RSU.
2: **Input:** Historical data of requested contents at each RSU.
3: **Output:** Next n contents to be proactively cached.
4: **Initialization:** Initialize local content space.
5: Find top-n popular contents in each RSU using popularity count.
6: Compute top-k similar contents of each popular content, and their similarity scores using (5).
7: Send top-k similar contents to MBS.
8: Update local content space with similar contents based on the obtained similarity scores.
9:
10: **Level II:** Predict next top-n to cache combining distributed collaborative filtering and LSTM model at the MBS.
11: **Input:** top-k similar contents of each RSU, historical data of MBS, popularity count of each content, LSTM configurations [8].
12: **Output:** Predict popularity count of n contents to be proactively cached at each RSU and the MBS.
13: **Initialization:** Initialize regional database of MBS.
14: **for** each contents in regional database **do**
15: Predict next top-n contents using LSTM model.
16: **if** contents available in MBS **then**
17: send n contents list to each RSU.
18: **else**
19: fetch contents from the content server.
20: send n contents to each RSU.
21: **end if**
22: Update regional database.
23: **end for**
24: **return** top n contents to be cached in each RSU.
25: Update local content space.

V. SIMULATION RESULTS

In this section, we present the results of experiments conducted to evaluate the proposed approach's performance.

A. DATASET

We have used the real-world movielens (100k) dataset [5] for our experiments, which includes 100,000 ratings (1-5)

TABLE 1. Summary of key parameters.

Parameters	Value
Number of features	2
Type of feature	Popularity Count of contents, Title of contents
No. of LSTM layer	1
No. of LSTM cell in each layer	90
No. of neurons in dense layer	1
Batch Size	10
Output Activation Function	ReLu
Optimizer	Adam
Window Size	[50, 500]
Cache Size	[10, 50]Mb
Loss Function	Mean Absolute Error (MAE)
Training Epoch	50
Training Data	70%
Validation Data	30%

from 943 users on 1682 movies. It also includes users' demographic information such as *age*, *gender*, *occupation*, and *zip address*. We split the dataset randomly between two different RSUs, namely RSU X and RSU Y, to realize a practical distributed setting. In Table 1, we show the summary of key parameters used when conducting the experiments.

In the following subsection, we present the traditional methods used as baselines to demonstrate the efficiency of our proposed proactive content caching scheme.

B. TRADITIONAL METHODS

We compare our proposed approach with three traditional algorithms [37], and the optimal, which are described below:

1) Optimal Solution (**Optimal**): This approach considers optimal caching scheme where the contents receiving the most requests is cached following an exhaustive search algorithm [38]. Using the contents popularity count at RSUs, and considering the available cache space, the top-n popular content from each RSU is known to the MBS. Then, the MBS combines those contents with its historical data, and prioritized these contents based on popularity count. Finally, the MBS exploits its regional database to compare it with the popularity count of contents at each RSU and obtain the optimal caching strategy for contents that minimizes the overall delay.
2) Centralized Collaborative Filtering (**Centralized**): This approach considers the available of full historical information of content requests at a single place, i.e., at the MBS. The idea of storing the whole dataset of the particular area without sharing the distributed data is basically impractical as it has additional communication overheads, privacy issues, and cannot capture real-time content popularity trend for developing caching strategies.
3) Least Recently Used (**LRU**): When making a caching decision, this strategy discards the least recently used items. The LRU algorithm involves keeping track of what was used from the database for this purpose; therefore, it is a costly method to ensure that the algorithm always discards the least recently used item.

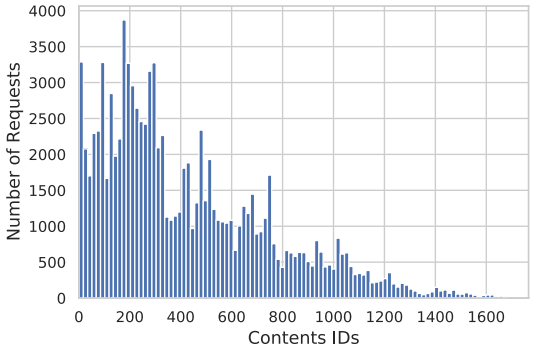

FIGURE 4. An illustration of content popularity distribution.

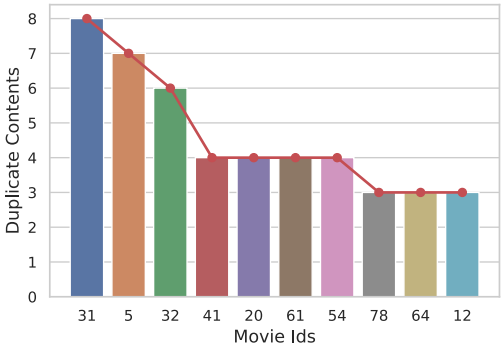

FIGURE 5. Duplicate contents in MBS and RSU X.

4) Random Replacement (**RR**): This strategy selects a candidate item randomly and discards it when necessary to make the cache space available. This algorithm does not need any details about the history of content access. Basically, every other items is replaced with the same probability in this algorithm.

C. PERFORMANCE EVALUATION

In Fig. 4, we show an illustration of the content popularity distribution at a RSU. We observe the number of requests are random and follows a *Pareto* distribution in which most of the requests are confined for certain movies. This also demonstrates the trend of content popularity in a local level. For the simulation purpose, we label the content popularity as the number of request counts for the movies and sort it in a time-sequence data structure. In a distributed setting, once the RSUs share their top-k similar movies of top-n popular movies using the local content space, the MBS build a regional database to capture distributed movie popularity trend. Fig. 5 and Fig. 6 reflects the significance of regional database as it can capture the duplicate requests made for movies and adjust the caching strategy to minimize the content retrieval cost. Next, we show several observations based on cache space utilization and content popularity. In doing so, we use the cache hit ratio as the performance metric to evaluate our proposed approach. Cache hit ratio is a well-known metric to evaluate the efficiency of a caching strategy. Particularly, it is defined as the ratio of the number of requests served due to availability of contents in the cache

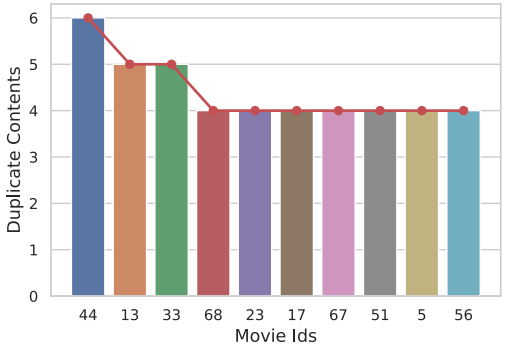

FIGURE 6. Duplicate contents in MBS and RSU Y.

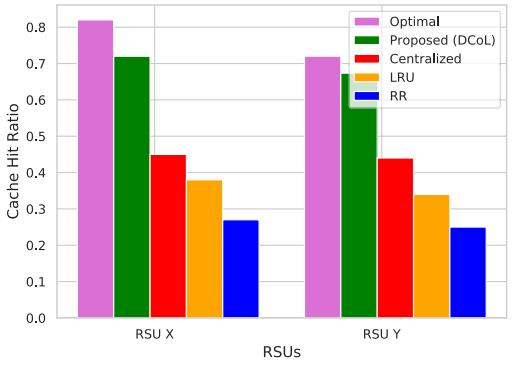

FIGURE 7. Performance comparison between DCoL and baselines: (i) Optimal, (ii) Centralized, (iii) LRU, and (iv) RR, in terms of cache hit ratio.

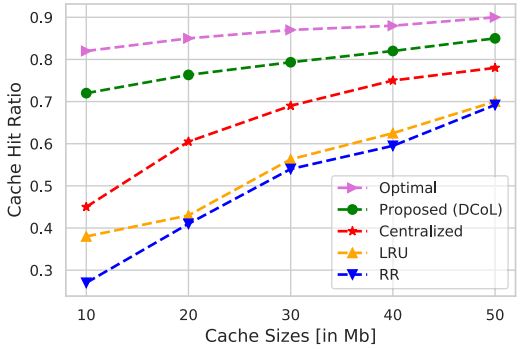

FIGURE 8. Performance comparison between DCoL and baselines: (i) Optimal, (ii) Centralized, (iii) LRU, and (iv) RR, in terms of different cache sizes at the RSUs.

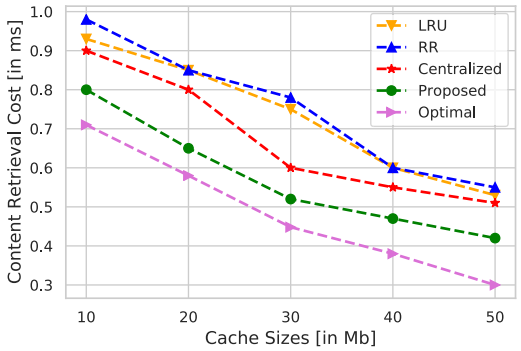

FIGURE 9. Performance comparison between DCoL and baselines: (i) Optimal, (ii) Centralized, (iii) LRU, and (iv) RR, in terms of content retrieval cost.

space to the number of requests for such contents received; and thus, a higher value of cache hit ratio is desirable. Finally, considering the cache space limitation, we show the proposed method can well-capture the content popularity distribution compared with other traditional methods, where we can have several caches misses. In our experiments, we perform Monte Carlo simulations to capture the variability of underlying stochastic process, validate the stability of the obtained solution using Algorithm 1, and for better analysis of the proposed mechanism.

In Fig. 7, we observe DCoL outperforms the baselines Centralized, LRU, and RR in terms of cache hit ratio, and provides a near-optimal, low-complexity solution. In particular, DCoL improves the cache hit ratio by a competitive margin of 8.9% against best performing Centralized approach, which is an offline method to decide content caching strategy. This is quite intuitive as DCoL can well-capture popularity trend over a region to proactively cache the candidate contents that will be requested by the users. In fact, this result is favored by the collaboration amongst RSUs in sharing their historical content popularity data, where we compute similarity scores of movies using cosine-similarity at the local level on local content space.

In Fig. 8, we investigate the cache hit efficiency for varying cache sizes between the range [10, 50]Mb. In this work, cache efficiency is defined in terms of the cache hit ratio, i.e., the number of user requests served by the RSUs following the proactive content caching strategy. Moreover, we assume each movie contents to be cached is of same size, similar to [8], [9]. As shown in Fig. 8, we observe a significant improvement in the cache hit ratio using DCoL as compared to baselines when the local cache size is limited. Furthermore, with the increase in the cache size, we see other baselines performing well in terms of improving cache hit ratio. This is obvious as with increase in cache size, the probability of cache hit improves given more contents proactively cached at the RSUs reusing local content space. Besides, it is interesting to realize our proposed algorithm provides an upper bound on cache hit efficiency, while the RR algorithm performs worst among the traditional baseline algorithms. In this regard, the centralized algorithm shows better performance compared to the LRU and RR algorithm as it is an offline caching scheme that learns from the past requests at the MBS. However, Centralized does not use content information of other RSUs; and thus, results in caching the contents of that specific area only.

In Fig. 9, we evaluate the efficiency of proposed method in terms of content retrieval cost. As we have discussed before, content retrieval cost is correlated with the cache miss and the available cache size at the RSUs. We observe an improvement of 18% in terms of content retrieval cost as compared with the conventional Centralized approach. Notice that DCoL outperforms other baselines with a significant margin as well. Furthermore, with the increase in the available cache size at the RSUs, we see the content retrieval cost also reduces.

TABLE 2. Comparison results of cache hit ratio and content retrieval cost.

Traditional Methods	DCoL: Improvements (%)	
	Cache Hit Ratio	Content Retrieval Cost
Centralized	8.9	18
LRU	23.19	23.63
RR	25	41

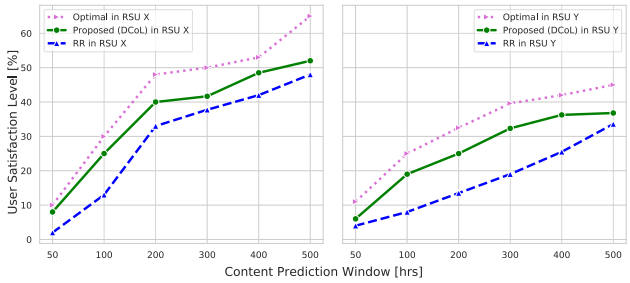

FIGURE 10. User satisfaction level at RSUs.

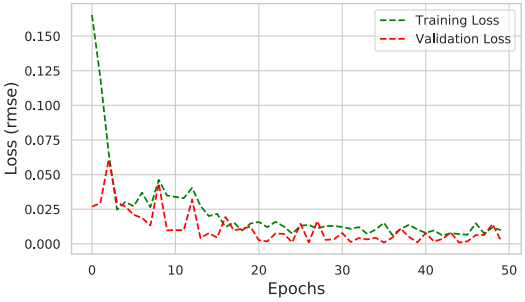

FIGURE 11. Model performance during the training and validation rounds.

This is because of the increase in cache hit with availability of cache space for proactively caching large number of contents. Table 2 summarizes the improvement of our proposed approach in case of both cache hit efficiency and content retrieval cost.

Similarly, in Fig. 10, we measure the user satisfaction level for different values of content prediction window N. Here, user satisfaction is defined as the proportion of the users served by the RSUs following the proposed proactive content caching scheme. Practically, user satisfaction level captures the average of satisfied content requests made by the users with heterogeneous preference over the contents to the RSUs. In particular, considering the dynamics of content requests made by users associated with each RSU, we evaluate the user's satisfaction as the percentage of weighted value of cache hit on the number of contents requested by each user to the total content requested. In this regard, larger value of user's satisfaction accounts for the improved performance of the proposed approach. Also, given the content prediction window, we make sure that the cache size is the same but the timestamp changes in the order of hours. As observed in Fig. 10, we compare the user satisfaction level achieved using DCoL with the baseline RR for different content prediction window. We observe our proposed scheme outperforms the baseline with up to 9% improvement for the use case scenario. Moreover, we see an increase in the user satisfaction level with the increase in content prediction window. This is because DCoL can recommend a larger number of possible contents that will be requested by the users for proactive caching; and thus, increasing the cache hit probability.

Finally, in Fig. 11, we demonstrate the performance efficiency of DCoL with the evaluation of root mean square error (rmse) at the convergence of the algorithm. We observe the validation loss is small (below 0.025) within few epochs. This is significant given the improvements in cache efficiency and content retrieval cost under the proposed distributed mechanism of proactive content caching.

VI. CONCLUSION

Caching at the network edge is a promising solution to cope with the exponential growth of data request from different IoT devices, where contents are usually placed on local caches for fast and repetitive access. However, edge devices have limited cache space and cannot appropriately decide the contents to cache due to dynamic arrival of user requests for random contents. In this work, we have studied the problem of proactive content caching at the network edge. In doing so, we have proposed a distributed collaborative learning mechanism for proactive content caching at the network edge, namely *DCoL*. Particularly, we have combined distributed collaborative filtering technique with LSTM model to exploit information related with local content popularity and determine contents to be cached at the distributed edge nodes proactively. In doing so, we have developed a non-parametric algorithm that first builds the regional database, and then leverages such regional information as a feedback to tune the local content caching strategy. As compared with the traditional cache replacement strategies, simulations have shown DCoL as an efficient proactive caching strategy that can jointly improve cache hit ratio, optimize cache utilization, and minimize the content retrieval cost.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[4] J. Pfender, A. Valera, and W. K. Seah, "Performance comparison of caching strategies for information-centric IoT," in *Proc. 5th ACM Conf. Inf.-Centric Netw.*, 2018, pp. 43–53.

[5] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016.

[6] R. Fares, B. Romoser, Z. Zong, M. Nijim, and X. Qin, "Performance evaluation of traditional caching policies on a large system with petabytes of data," in *Proc. IEEE 7th Int. Conf. Netw., Archit., Storage*, Jun. 2012, pp. 227–234.

[7] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[8] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, "DeepMEC: Mobile edge caching using deep learning," *IEEE Access*, vol. 6, pp. 78260–78275, 2018.

[9] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2021.

[10] L. Li, Y. Xu, J. Yin, W. Liang, X. Li, W. Chen, and Z. Han, "Deep reinforcement learning approaches for content caching in cache-enabled D2D networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 544–557, Jan. 2020.

[11] Y. K. Tun, A. Ndikumana, S. R. Pandey, Z. Han, and C. S. Hong, "Joint radio resource allocation and content caching in heterogeneous virtualized wireless networks," *IEEE Access*, vol. 8, pp. 36764–36775, 2020.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[13] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241–3256, May 2020.

[14] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977

[15] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, Z. Han, and C. S. Hong, "Incentivize to build: A crowdsourcing framework for federated learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[16] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: A deep learning approach with SDN," *IEEE Access*, vol. 6, pp. 5075–5089, 2018.

[17] J. Zhou, X. Zhang, and W. Wang, "Social-aware proactive content caching and sharing in multi-access edge networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1308–1319, Dec. 2020.

[18] S. Manzoor, S. Mazhar, A. Asghar, A. N. Mian, A. Imran, and J. Crowcroft, "Leveraging mobility and content caching for proactive load balancing in heterogeneous cellular networks," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, Feb. 2020, Art. no. e3739.

[19] Q. Wang and D. Grace, "Sequence prediction-based proactive caching in vehicular content networks," in *Proc. IEEE 3rd Connected Automated Vehicles Symp. (CAVS)*, Nov. 2020, pp. 1–6.

[20] Z. Zhang, C.-H. Lung, M. St-Hilaire, and I. Lambadaris, "Smart proactive caching: Empower the video delivery for autonomous vehicles in ICN-based networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7955–7965, Jul. 2020.

[21] X. Xu, C. Feng, S. Shan, T. Zhang, and J. Loo, "Proactive edge caching in content-centric networks with massive dynamic content requests," *IEEE Access*, vol. 8, pp. 59906–59921, 2020.

[22] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for Top-N recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 1–25, Jul. 2019.

[23] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service QoS prediction via collaborative filtering: A survey," *IEEE Trans. Serv. Comput.*, early access, May 18, 2020, doi: 10.1109/TSC.2020.2995571.

[24] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–45, Jul. 2014.

[25] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 80–87, Jun. 2018.

[26] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–6.

[27] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *Int. J. Commun. Syst.*, vol. 31, no. 11, p. e3706, Jul. 2018.

[28] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, "Federated learning based proactive content caching in edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[29] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social-caching in wireless small cell networks," in *Proc. 12th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2014, pp. 569–574.

[30] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating collaborative filtering recommender algorithms: A survey," *IEEE Access*, vol. 6, pp. 74003–74024, 2018.

[31] R. Zhang, Q.-D. Liu, and J.-X. Wei, "Collaborative filtering for recommender systems," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2014, pp. 301–308.

[32] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[33] M. K. Kharita, A. Kumar, and P. Singh, "Item-based collaborative filtering in movie recommendation in real time," in *Proc. 1st Int. Conf. Secure Cyber Comput. Commun. (ICSCCC)*, Dec. 2018, pp. 340–342.

[34] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[35] R. Sharma, D. Gopalani, and Y. Meena, "Collaborative filtering-based recommender system: Approaches and research challenges," in *Proc. 3rd Int. Conf. Comput. Intell. Commun. Technol. (CICT)*, Feb. 2017, pp. 1–6.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[37] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, Mar. 2017.

[38] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Chelmsford, MA, USA: Courier Corporation, 1998.

SUBINA KHANAL received the bachelor's degree in information management from Tribhuvan University, Nepal. She is currently pursuing the M.E. degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. For two years, she worked as a Quality Assurance (QA) Engineer. Her research interests include applied machine learning, federated learning, and the Internet of Things (IoT). She was awarded the prestigious Brain Korea 21st Century (BK21) Scholarship and the Graduate Scholarship, in 2020.

KYI THAR received the Bachelor of Computer Technology degree from the University of Computer Studies, Yangon, Myanmar, in 2007, and the Ph.D. degree in computer science and engineering from Kyung Hee University, South Korea, in 2019. He is currently a Postdoctoral Research Fellow with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include name-based routing, in-network caching, multimedia communications, scalable video streaming, wireless network virtualization, deep learning, and future Internet. In 2012, he was awarded the Scholarship for his Ph.D. degree.

EUI-NAM HUH (Senior Member, IEEE) received the B.S. degree from Busan National University, South Korea, the master's degree in computer science from The University of Texas at Austin, TX, USA, in 1995, and the Ph.D. degree from Ohio University, Athens, OH, USA, in 2002. He is currently with the Department of Computer Science and Engineering, Kyung Hee University, South Korea, as a Professor. His research interests include cloud computing, the Internet of Things, future Internet, distributed real-time systems, mobile computing, big data, and security. He is a Review Board Member of the National Research Foundation of Korea. He has also served many community services for ICCSA, WPDRTS/IPDPS, the APAN Sensor Network Group, ICUIMC, ICONI, APIC-IST, ICUFN, and SoICT, as various types of chairs. He is also the Vice-Chairman of the Cloud/Bigdata Special Technical Group of TTA, and an Editor of ITU-T SG13 Q17.