Proactive Content Caching at Self-Driving Car Using Federated Learning with Edge Cloud

Subina Khanal, Kyi Thar, Md Delowar Hossain, Eui-Nam Huh
Department of Computer Science and Engineering, Kyung Hee University,
Yongin 17104, South Korea
Email: {subinakhanal, kyithar, delowar, johnhuh}@khu.ac.kr

***Abstract*—Proactive content caching in self-driving cars poses several challenges, particularly because of the dynamic nature of content popularity, heterogeneity in user preferences, and privacy issues for data sharing. To tackle these issues, in this paper, we study the significance of proactive content caching strategy in self-driving cars for optimizing content retrieval cost and quality-of-experience (QoE) with the edge cloud infrastructure. To that end, we propose a low-complexity content popularity prediction mechanism in a federated setting where we extract local content popularity patterns in the self-driving cars using long short-term memory (LSTM)-based prediction mechanism. Then, we leverage the privacy-preserving distributed model training paradigm of Federated Learning (FL) to create a global model by applying the Federated Averaging (FedAvg) algorithm on local LSTM models to create a regional content popularity prediction model. With extensive simulations on real-world datasets, we show the obtained global model helps to improve the local cache hit ratio, cache space utilization, and correspondingly minimize latency overhead at the self-driving cars.**

***Index Terms*—proactive content caching, edge computing, federated learning, recurrent neural network**

I. Introduction

As self-driving car technology has advanced, the likelihood of autonomous public vehicles running on roads is no more a far-fetched reality. Consequently, passengers will find themselves with a good deal of free time within the driverless car [1]. In this regard, taking advantage of the in-vehicle infotainment service provided by on-board unit (OBU) installed in self-driving cars [2], passengers can now spend their time working and being entertained, or just relaxing. However, OBU can only cache a subset of contents; thus, not every passenger will instantly receive the requested contents. For that reason, self-driving cars must download these contents from the nearby edge cloud which is also known as Road Side Unit (RSU), or the core cloud. If the requested content is cached in an exceedingly connected RSU, it may be downloaded directly from the RSU. Otherwise, the self-driving cars have to request the contents from other (nearby) self-driving cars or RSUs, or cloud until the content is found and retrieved, leading to a higher content delivery delay. One approach to mitigate the possible delay is use OBU to proactively cache the contents following the historical content popularity patterns. However, the popularity of content is dynamic, and relying only on historical contents of a particular OBU will not be sufficient to guarantee Service Level Agreements (SLAs) and Quality-of-Experience (QoE) for seamless media streaming. That is why we need to consider the content popularity patterns of other OBUs as well. However, due to privacy concerns, OBUs will not share their data. Thus, considering the limited cache capacity of OBU, efficient proactive content caching strategy is imperative [3], [4]. In fact, proactively caching the expected content to be requested at the selected OBUs can significantly reduce the peak load on edge networks, i.e., when the OBU receives the request from passengers, the content can be retrieved directly from the cached memory of OBU instead of accessing from the wireless edge network. Fairly, the prediction paradigm is based on the assumption that the demand trend of an infotainment content of a self-driving car is predictable to some degree.

There are several ongoing research works on proactive content caching strategies [4]–[9]. The authors in [6] concentrated primarily on the relationship between the content and its recent access pattern. However, they don't use any prior knowledge of content popularity distribution or any dedicated model training phase, which they believe is obsolete or biased. The authors in [10] configured the number of chunks to be cached by using the popularity count of contents. Similarly, the authors of [11] concentrate on context-aware proactive caching, in which they learn context-specific content popularity online by constantly analyzing context details of linked users, updating the cached content, and then monitoring cache hits. Similarly, a large amount of data is used in [12] to estimate contents popularity, and then, strategic contents are cached at the base stations to improve user's cost for content retention and backhaul offloading. In [4], the authors use passenger's features, which are obtained using deep learning techniques to obtain caching decisions for infotainment contents in self-driving cars.

The recent research works, and those mentioned above, mainly focus on centralized content caching. However, the traditional centralized approach is privacy sensitive, and not all users may want to share their data with the core cloud, or the edge cloud. And even if some users share data, the model may only work for a certain number of users, which leads to generalization issues, i.e., the trained model will not satisfy all user's preferences. To tackle these issues, in this work, we leverage the distributed privacy-preserving model training paradigm of federated learning [13] to develop a proactive content caching scheme, where self-driving cars will train their local dataset using LSTM models and share model parameters to the server to get a generalized global model.

The main contributions of the paper are summarized as follows:

- We propose a federated learning-based popularity prediction mechanism in self-driving cars for proactive content caching, where data (i.e., content popularity patterns) will not be directly shared among OBUs, but rather predicted, following the decentralized model training approach of federated learning.

- We build a global model for content popularity prediction exploiting LSTM models. Particularly, LSTM models are trained first to capture local content popularity prediction at the self-driving cars; and then, the local parameters of LSTM models are shared with the RSUs to build a global model in an iterative fashion.
- We show the proposed methodology captures personalized preferences of passengers in building a content popularity distribution.
- We show extensive experimental results based on real-world Movie Lens datasets [14] to verify that our proposed approach 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 presents our proposed system model of proactive content caching at self-driving car using federated learning, discuss overview of solution approach, and present preliminaries of adopted federated learning and LSTM model. Section III discuss the details of our proposed approach, and present a low-complexity algorithm. Section IV provides the performance evaluation of the proposed approach and compares it with other traditional approaches using real-world datasets. Finally, Section V concludes this work.

II. System Model

The system model of our proposed method is shown in Fig. 1. In our system model, we consider self-driving cars as public vehicles connected to RSUs via radio links. The passengers in the car request various infotainment contents, such as movies, music videos, and so on. Particularly, these requested contents differ according to the location, time of the request, and other features. Besides, the content request patterns may be similar or different depending on the passenger's situation. The OBU of the car store these content request patterns as historical data. As a matter of fact, we observe OBU is responsible for recognizing these content request patterns. And to utilize these patterns, we first use the concept of LSTM, whose input be the sequential historical contents of OBU and the popularity count of each content. Here, the LSTM model predicts each content's next popularity count, which helps OBU determine the top most popular content recommended to the passengers. However, content popularity changes over time; that is why, using the historical content of a particular OBU is not be sufficient for satisfying passenger's requests. Therefore, we consider content request patterns of other self-driving cars as well to appropriately characterize popularity patterns and address content request delivery. However, content request patterns include passenger's personal information, and taking privacy into account, other OBUs will not share their popularity request pattern directly. To address this challenge, we exploit the concept of federated

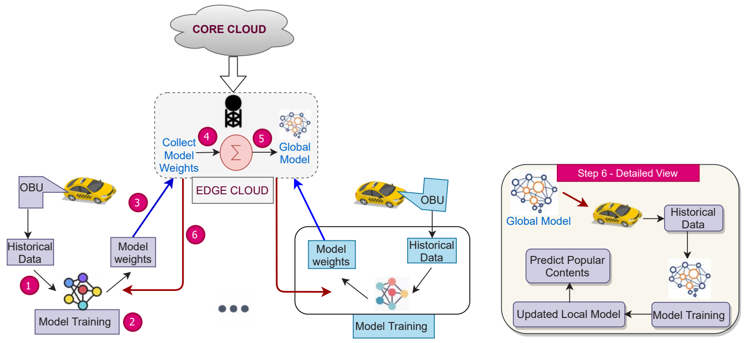

Fig. 1: System Model

learning in which each OBU only shares its model parameters rather than the raw data itself; and hence, significantly eliminating the privacy concern. Finally, the global model built after decentralized training over LSTMs help each OBUs to predict the popular contents of the overall area. Then, OBUs can recommend these top most popular contents. However, given limited storage space of OBUs, it is impossible for OBUs to cache all of the recommended contents; thus, it only caches the popular contents in a proactive manner according to its cache size.

In the following subsections, we will discuss the details of federated learning and LSTM models involved to execute the proposed method.

A. Preliminaries: Federated Learning

Federated Learning (FL) [15]–[17] is a distributed machine learning approach that enables training on a large corpus of decentralized data residing on devices like mobile phones. FL is a promising approach for privacy preserved edge intelligence in distributed scenarios. While in conventional machine learning, all training data is collected at a centralized curator, federated learning addresses the privacy concerns to a large extent and reduces data transmission cost by distributing the training work to users themselves. Particularly, the local training is executed by users on their data, which usually adopts the gradient descent optimization algorithm. In a federated learning framework, users keep their data with themselves but send the server parameters for aggregation. This provides a parallel scheme for users to learn a global model collaboratively concerning their data privacy. The general optimization problem is as follows [18], [19]:

$$F(w) = \frac{1}{K}\sum\nolimits_{k=1}^{K} F_k(w); F_k(w) = \frac{1}{n_k}\sum\nolimits_{(x_i,y_i)\in\mathcal{D}_k} l_i(w), \quad (1)$$

where $l_i(w)$ is the loss of prediction for some input-output pairs (x_i, y_i) in the training data samples x_i and labels y_i, respectively, w is the model parameter, K is the total number of clients, $\mathcal{D}_k$ is the set of indices of data points on client k with $n_k = |\mathcal{D}_k|$ and D the total data samples. Each client k updates local model w^k following stochastic gradient descent to solve 1 in a distributed manner.

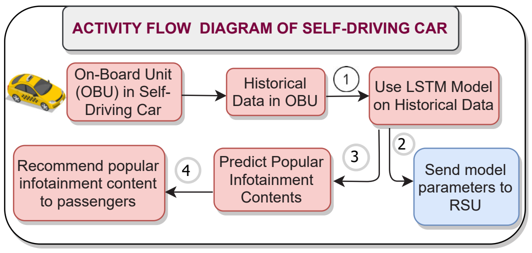
ACTIVITY FLOW DIAGRAM OF SELF-DRIVING CAR
On-Board Unit (OBU) in Self-Driving Car
Historical Data in OBU
1
Use LSTM Model on Historical Data
2
Send model parameters to RSU
3
Predict Popular Infotainment Contents
4
Recommend popular infotainment content to passengers

Fig. 2: An illustration of proposed approach: Client-Side.

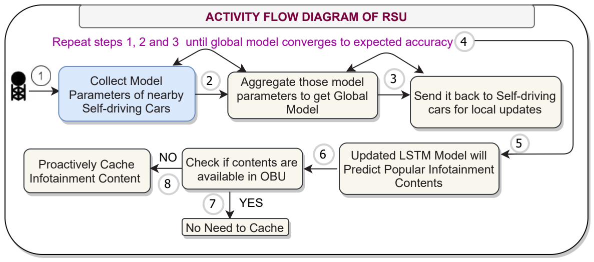
ACTIVITY FLOW DIAGRAM OF RSU
Repeat steps 1, 2 and 3 until global model converges to expected accuracy
4
1
Collect Model Parameters of nearby Self-driving Cars
2
Aggregate those model parameters to get Global Model
3
Send it back to Self-driving cars for local updates
5
Updated LSTM Model will Predict Popular Infotainment Contents
6
Check if contents are available in OBU
NO
8
Proactively Cache Infotainment Content
7
YES
No Need to Cache

Fig. 3: An illustration of proposed approach: Server Side.

B. LSTM Model

LSTM is a Recurrent Neural Network (RNN) that can perform learning patterns on data with long dependent periods [20]. RNNs are mainly designed to capture the temporal dynamics of sequential input data. In our proposed approach, we feed the timestamped historical content popularity counts of each self-driving car as the input of the LSTM model to predict the sequence of next popular contents to be proactively cached for recommending and serving those infotainment contents to the passengers.

In the following section, we present our proposed solution approach.

III. Proposed Method for Proactive Content Caching

This section proposes the deployment design of federated learning-based proactive content caching in self-driving car. We first present the details of processes involved in predicting the content popularity count in our proposed system using LSTM model. Then, we discuss how self-driving car will proactively cache the popular content exploiting the global model obtained after model averaging.

At first, self-driving cars get content requests from various passengers based on their location. Primarily, the car's OBUs need to have those contents stored to serve requested contents to passengers. However, the dynamic nature of the user's preferences and the limited storage capacity of OBUs, not all requested contents are cached at the OBU; therefore, passengers will not instantly receive the requested contents. For that reason, OBUs must download these contents from RSU proactively. Also, because of the high amount of content requests, it is impossible to cache all the replicas in OBUs. Therefore, OBUs must select the contents which may be proactively cached to reduce the high delivery delay. In Fig. 2, we show an illustration of the proposed mechanism

Algorithm 1 LSTM-embedded Federated Averaging for Proactive Content Caching Strategy.

```
1:  **RSU executes:**
2:  **Initialization:** initialize model parameter $w_0^g$;
3:  **Output:** global model $w^g$, content popularity count;
4:  $V_t$: total associated cars with the RSU at time $t$;
5:  **for** each round $t = 1, 2, \ldots$ **do**
6:      **for** $\forall k \in V_t$ **in parallel do**
7:          $w_{t+1}^k \leftarrow \text{LOCALUPDATE}(k, w_t^g)$;
8:      **end for**
9:      $w_{t+1}^g \leftarrow \sum_{k=1}^{V_t} \frac{n_k}{D} w_{t+1}^k$;
10: **end for**
11:
12: LOCALUPDATE$(k, w)$ :    ▷ *Execute local model updates
    on car $k$.*
13: **for** each local epoch $E$ **do**
14:     $n_k$: is the training data;
15:     $B$: is the local minibatch size used for the car updates;
16:     $\eta$: is the learning rate;
17:     $\mathcal{B} \leftarrow$ (split $n_k$ into mini-batches of size $B$);
18:     **for** $b \in \mathcal{B}$ **do**
19:         $w_{t+1}^k \leftarrow w_{t+1}^k - \eta \nabla \ell(w; b)$;    ▷ Update LSTM
    model with global model $w$.
20:     **end for**
21:     return $w_{t+1}^k$ to RSU;
22: **end for**
```

for infotainment content recommendation at the client-level. The OBU of self-driving cars (clients) have historical contents stored in them. These sequential historical contents have a specific content popularity pattern in the form of content request count. Using these historical data and considering the limited storage of each OBUs, we find top-n most popular contents in all clients. In our case, popular contents are the contents with a maximum number of request count, i.e., the popularity count. This popularity count determines popular content in each client. In the first step, we use the historical contents and their popularity count as an input feature, and then train our LSTM model to predict the popularity count of the next popular content; each client's OBU may choose to cache and recommend it to the passengers. However, caching only popular content won't be enough as the contents requested depend on passenger's preferences and changes over time. In addition to this observation, given the clients and the passengers' high mobility, it is impractical to solely adopt content popularity patterns in an individual OBU as a critical metric to design proactive content caching strategy at the client-level. Hence, we need to build a generalized observation of popular contents from other client's OBU. However, due to privacy reasons, clients may not share their content request pattern. For that reason, we use the concept of federated learning, as discussed before, to share the local model parameters of OBUs for more generalized view of content request patterns. To that end, in the second step, after local model training in each client, model parameters

Algorithm 2 Local Content Caching Strategy in Self-Driving Cars.

```
    //local content caching strategy//;
2:  $w^g$: Final global model;
    $V_t$: total associated cars with the RSU at time $t$;
4:  **Output:** local content popularity count, top- $n$ contents to be proactively cached;
    **for** $\forall k \in V_t$ **do**
6:      Use $w^g$ to obtain local content popularity count;
        **for** each contents predicted by the model **do**
8:          **if** contents are not available in OBU **then**
                fetch contents from RSU and cache them;
10:         **end if**
        **end for**
12: **end for**
```

are sent to the server[1], i.e., RSU for model aggregation. In our proposed approach, the purpose of LSTM is to predict next popular content using the content popularity count, and hence, the third step is to predict the popular infotainment contents from the local data. Finally using these predicted popularity patterns, in the fourth step, OBU can recommend the infotainment contents to the passengers.

Fig. 3 shows the server-side illustration of the proposed mechanism. Here, in the first step, RSU aggregates and averages the model parameters obtained from the clients using FederatedAveraging (FedAvg) algorithm [18]. After FedAvg, a global model is obtained in the second step, which is sent to each client for local updates in the third step. This process in step four continues until the global model converges to expected model accuracy. After downloading the global model from the server, each client in the fifth step updates the LSTM model to predict the next popular contents to be cached proactively. In this manner, we obtain a generalized viewpoint of popular infotainment contents. In the sixth step, each client's OBUs check if these contents are available or not; if yes, it serves the content requested by passengers. Else sends a request to nearby RSUs for proactively caching those contents to serve passenger's requests. The details of the sever-side, client-side mechanism are presented in Algorithm 1 and Algorithm 2 respectively.

IV. Simulation Results

We implemented the proposed algorithms in Tensorflow [21]. We have used the real-world movielens 25M (ml-25m) dataset [14] for our experiments, which contains 25000095 ratings and 1093360 tag applications across 62423 movies.

In Fig. 4, we compare our proposed approach with the traditional centralized approaches. The first is a centralized approach without sharing data (Centralized w/o sharing data), in which local data is not shared with other clients' OBUs or the server. Clients will use their local dataset to train the model, and the local model obtained after training will predict the next popular content from the local dataset itself. The

[1]Note that the term "server" and "Edge cloud" means the same and are used interchangeably in this work.

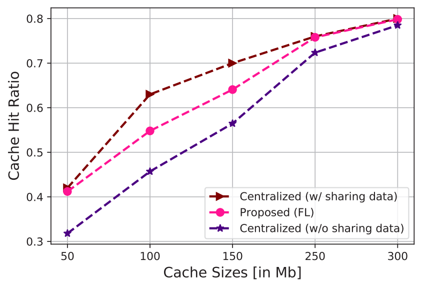

Fig. 4: Performance comparison of proposed approach with: (i) Centralized w/o sharing data ,and (ii) Centralized w/ sharing data in terms of cache hit ratio.

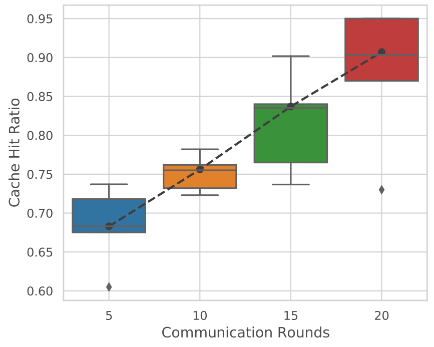

Fig. 5: Performance evaluation of proposed approach in terms of cache hit ratio by changing communication round between five random clients and a server.

second is a centralized approach with sharing data (Centralized w/ sharing data), where clients share their data with the server, and the server stores the data centrally. The server will use these data to train the model, and the final model will predict the next popular contents from the pool of stored contents. That way, each client will get the requested content instantly. For doing so, we set a small vehicular network of density 5 that share their local data with the computing server (nearby RSUs). As all the data is stored in the RSUs, this approach outperforms **Centralized w/o sharing data method**, which performs poor due to limited data. However, this approach poses some privacy issues due to data sharing requirements. In this regard, we observe our proposed approach outperforms the centralized approach without sharing data, and shows competitive performance in terms of cache hit ratio against **centralized w/ sharing data**. Furthermore, it is intuitive that relaxing cache space constraint improves cache hit ratio as more contents are cached.

Fig. 5 shows the performance of our approach for the number of communication rounds, i.e., the global iteration. Here, we evaluated the cache hit ratio for the cache size of 300MB for each client's OBU. The local model training epoch in each client is 20. We observe the cache hit ratio increases with the increase in communication round, as more rounds of communication improves the model performance.

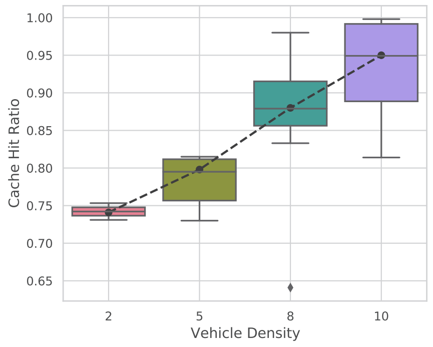

Fig. 6: Performance evaluation of proposed approach in terms of cache hit ratio by changing vehicle density.

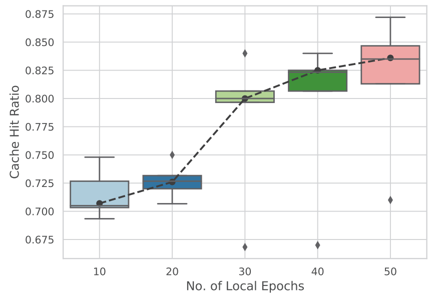

Fig. 7: Performance evaluation of proposed approach in terms of cache hit ratio by changing number of local epochs.

Similarly, in Fig. 6, we evaluate the proposed method in terms of a cache hit ratio for varying vehicle density in range [2,10], and with fixed communication rounds. Here, vehicle density is the number of clients participating in the federated learning process. We can observe with the increase in vehicle density, the cache hit ratio also increases. This is expected as improved participation increases the global model accuracy which eventually minimizes the content prediction error.

In Fig. 7, we observe the cache hit ratio increases with the increase in the number of local model training epochs for fixed number of communication rounds. This is a typical characteristic in the FL setting: as the number of local epochs increases, the accuracy of local model also increases, which results in the increase of cache hit ratio due to high-quality global model. In Fig. 8 and Fig. 9, respectively, we observe our proposed approach outperforms the traditional baselines [22] Least Recently Used (**LRU**) and Random Replacement (**RR**) in terms of a cache hit ratio by 50% and 61% respectively, and content retrieval cost by 37.78% and 46% respectively, for different cache sizes at the client's OBU. In fact, clients can make caching decisions based on the knowledge of a global model and local data; and thus, achieves a lower content retrieval cost as compared with the conventional approaches that make content replacement decision.

Finally, in Fig. 10, we measure the user satisfaction level for different cache sizes of the client's OBU. Here, the user satisfaction level captures the heterogeneous preference of

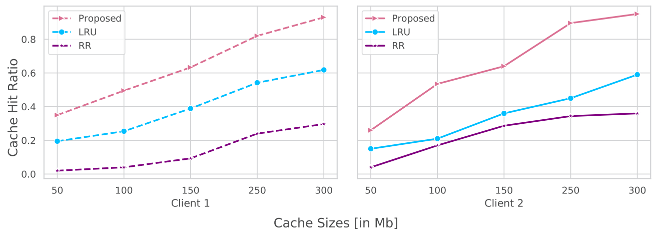

Fig. 8: Performance comparison between Proposed and baselines: (i) LRU, (ii) RR, in terms of different cache sizes at the client's OBU.

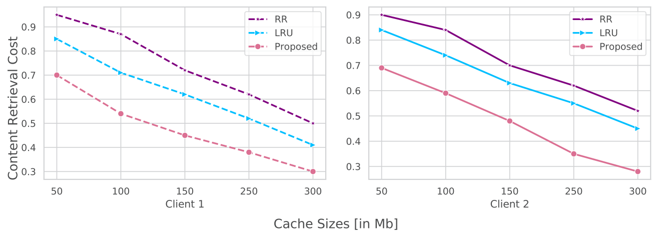

Fig. 9: Performance comparison between Proposed and baselines: (i) LRU, (ii) RR, in terms of content retrieval cost.

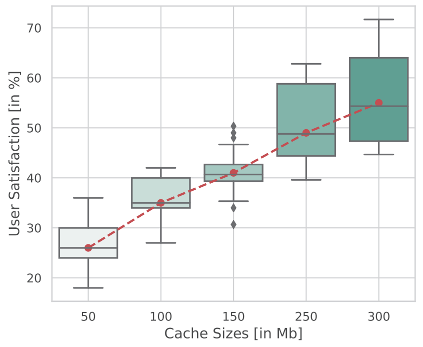

Fig. 10: User satisfaction level for three Clients.

passengers for infotainment contents and is defined as the number of satisfied passengers served by the OBU following the proposed proactive content caching scheme. We measure the user satisfaction ratio by the sum of requests served for passengers to the total number of requests. We observe the user satisfaction levels increases for large cache sizes because the OBU can cache more contents proactively. Furthermore, as observed in Fig. 11, the average user satisfaction level using the proposed method outperforms the baselines (LRU and RR) for different cache sizes in the client's OBU. This is because the proposed method can efficiently predict content popularity; and thus, we observe an increase in the cache hit probability.

V. Conclusion

Content Caching in the self-driving car is a promising solution to cope with the exponential growth of content requests from the diverse set of passengers, where contents are usually placed on OBUs for fast and repetitive access. However, OBUs

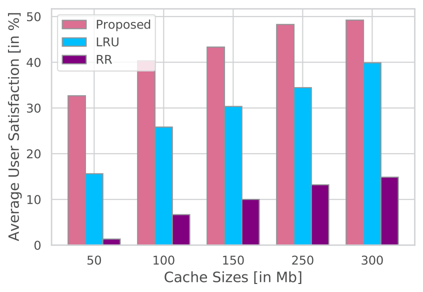

Fig. 11: Performance comparison between Proposed and baseline LRU and RR, in terms of different user satisfaction at the client level.

have limited cache space and cannot appropriately decide the contents to cache due to the dynamic arrival of passengers' requests for random contents. In this work, we have studied the problem of proactive content caching at the self-driving car. In doing so, we have proposed a federated learning-based mechanism for proactive content caching in the self-driving car. Particularly, we have used the LSTM-based mechanism for predicting local content popularity patterns, and sending model parameters to the edge cloud for creating a global model via model aggregation. With extensive simulations on real-world datasets, we have shown that the proposed mechanism can jointly improve cache hit ratio, optimize cache utilization, and minimize the content retrieval cost as compared with the traditional cache replacement strategies.

Acknowledgment

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00294, Service mobility support distributed cloud technology).

References

[1] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[2] S. A. Kazmi, T. N. Dang, I. Yaqoob, A. Ndikumana, E. Ahmed, R. Hussain, and C. S. Hong, "Infotainment enabled smart cars: A joint communication, caching, and computation approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8408–8420, 2019.

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

[4] A. Ndikumana, N. H. Tran, K. T. Kim, C. S. Hong *et al.*, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[5] S. Khanal, K. Thar, and E.-N. Huh, "Dcol: Distributed collaborative learning for proactive content caching at edge networks," *IEEE Access (in press), doi: 10.1109/ACCESS.2021.3080512*, 2021.

[6] S. Li, J. Xu, M. Van Der Schaar, and W. Li, "Popularity-driven content caching," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[7] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, "Deepmec: Mobile edge caching using deep learning," *IEEE Access*, vol. 6, pp. 78 260–78 275, 2018.

[8] Q. Chen, W. Wang, F. R. Yu, M. Tao, and Z. Zhang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 623–636, 2020.

[9] C. Li, Y. Zhang, M. Song, X. Yan, and Y. Luo, "An optimized content caching strategy for video stream in edge-cloud environment," *Journal of Network and Computer Applications*, p. 103158, 2021.

[10] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 2012, pp. 316–321.

[11] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2016.

[12] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 36–42, 2016.

[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 Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3241–3256, 2020.

[14] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[15] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, 2019.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[17] 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 *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[19] S. R. Pandey, M. N. Nguyen, T. N. Dang, N. H. Tran, K. Thar, Z. Han, and C. S. Hong, "Edge-assisted democratized learning towards federated analytics," *IEEE Internet of Things Journal*, 2021.

[20] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

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

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