Deep Learning Based Caching for Self-Driving Cars in Multi-Access Edge Computing

Anselme Ndikumana, Nguyen H. Tran, *Senior Member, IEEE*, Do Hyeon Kim, Ki Tae Kim, and Choong Seon Hong, *Senior Member, IEEE*

Abstract—Without steering wheel and driver's seat, the self-driving cars will have new interior outlook and spaces that can be used for enhanced infotainment services. For traveling people, self-driving cars will be new places for engaging in infotainment services. Therefore, self-driving cars should determine themselves the infotainment contents that are likely to entertain their passengers. However, the choice of infotainment contents depends on passengers' features such as age, emotion, and gender. Also, retrieving infotainment contents at data center can hinder infotainment services due to high end-to-end delay. To address these challenges, we propose infotainment caching in self-driving cars, where caching decisions are based on passengers' features obtained using deep learning. First, we proposed deep learning models to predict the contents need to be cached in self-driving cars and close proximity of self-driving cars in multi-access edge computing servers attached to roadside units. Second, we proposed a communication model for retrieving infotainment contents to cache. Third, we proposed a caching model for retrieved contents. Fourth, we proposed a computation model for the cached contents, where cached contents can be served in different formats/qualities based on demands. Finally, we proposed an optimization problem whose goal is to link the proposed models into one optimization problem that minimizes the content downloading delay. To solve the formulated problem, a block successive majorization-minimization technique is applied. The simulation results show that the accuracy of prediction for the contents that need to be cached is 97.82% and our approach can minimize the delay.

Index Terms—Deep learning based caching, deep learning, self-driving car, multi-access edge computing.

I. Introduction

A. Background and Motivations

RECENTLY, the automobile industries have focused on the next stage of autonomous driving, called "self-driving", where cars will drive themselves without human driver intervention [1]. To make the self-driving cars more intelligent, they need to be equipped with smart sensors and analytics tools that collect and analyze heterogeneous data related to passengers on-board, pedestrians, and the environment in real-time, in which Artificial Intelligence (AI) plays significant roles [2]. Furthermore, AI will be an empathetic companion of passengers for assisting them and providing personalized services. Therefore, AI will need to understand passengers' features [3].

In this work, we choose self-driving cars over human-driven cars because self-driving cars already have On-Board Units (OBUs) with Graphics Processing Units (GPUs), Field Programmable Gate Array (FPGA), and Application Specific Integrated Chip (ASIC) to handle in-car AI. This gives the self-driving cars the capability to observe, think, learn, and navigate in real driving environments [1]. Also, according to a study on the incremental time and what activities people will perform if everyone uses self-driving cars, it is estimated that there will be 22 billions of hours for extra media consumation in the US [4]. Therefore, with AI and OBUs that can handle Computation, Communication, Caching, and Control (4C) in self-driving cars, passengers will spend more time on infotainment services such watching media, playing games, and utilizing social networks. To support this, self-driving cars should be equipped with recent emerging technologies for infotainment services such as AI-based games, Virtual, Augmented, and Mixed Reality [5]. However, retrieving infotainment contents from Data Centers (DCs) can worsen infotainment content delivery services due to the associated end-to-end delay and consumed backhaul bandwidth resource. As an example, watching a video in a car requires three components, namely a video source, screen, and sound system. Therefore, if the source of the video is not in the car, the car needs to download it from DC. Assuming the DC is distantly located, then the infotainment content delivery services will incur a high delay. Therefore, caching in self-driving cars will play an important role in enhancing infotainment services. Furthermore, for retrieving infotainment contents that need to

be cached in self-driving cars, we consider Multi-access Edge Computing (MEC) [6], [7] as a suitable technology to support self-driving cars through caching infotainment contents near self-driving cars. In this work, MEC servers are deployed at RoadSide Units (RSUs).

B. Challenges for Infotainment Caching

- In human-driven cars, drivers choose the infotainment contents to display or play. However, in the absence of the driver, the self-driving car should determine itself the infotainment contents to cache and play that are likely to entertain its passengers.

- Some infotainment contents may not be appropriate for consumption by passengers depending on their age and area. Therefore, the self-driving car should determine itself the infotainment contents to cache that do not violate prohibited and restricted content access policies.

- As shown in Fig. 1 generated from YouTube demographics dataset for one month available in [8], people have different content preferences, in which their choices depend on their features such as age and gender. Therefore, in the self-driving driving cars, caching decisions for the infotainment contents should depend on passengers' features.

- Self-driving cars will eventually deliver more heterogeneous infotainment contents such as movies, TV, music, and games as well as recent emerging technologies such as Virtual, Augmented, and Mixed Reality [5]. However, obtaining infotainment contents from DC can induce high car-DC delay. Therefore, self-driving cars need to be supported by MEC servers by caching infotainment content in close proximity to self-driving cars at RSUs.

- Self-driving cars are sensitive to delay due to their high mobility and connection in-motion. Therefore, to achieve less variation in transmission delay for downloading contents need to be cached, at the beginning of the journey, the self-driving car should select available MEC servers en-route that will be used to download infotainment contents.

C. Related Works

Content caching at macro Base Stations (BSs) and RSUs has gained significant attention [7], [9]. However, there is still a lack of literature on caching infotainment contents in the cars based on passengers' features. To address the above challenges, in [10], the author proposed an auto-control system for the vehicle infotainment system, where the system analyzes the characteristics of passengers, e.g., by listening to conversations between passengers, understanding the atmosphere or ambiance inside the vehicle during the trip, and determining the relationship between passengers. The results of this analysis help the system identify and deliver appropriate infotainment contents to the passengers. However, in [10], there is no caching approach for infotainment contents. Always the cars have to retrieve the infotainment contents from DC. In [11], the authors proposed a cloud-based vehicular ad-hoc network, where both vehicles and RSUs participate in content caching. However, introducing a cloud-based controller into vehicle caching can increase the content retrieval delay. To overcome this issue, the authors in [12] proposed joint communication, caching, and computation. However, the authors did not discuss how to select the contents to cache based on vehicle occupants. Furthermore, for V2X communication, authors in [13] proposed the caching approach which is based on machine learning, where they used different classes of data and class-based cache replacement schemes. Other alternatives have been proposed in [14], where the authors considered two levels of caching at the edge servers (BSs) and autonomous cars. In their proposal, the edge servers inject contents into some selected cars that have enough influence to share these contents with other cars. However, in a realistic network environment, BSs and cars may belong to different entities. Therefore, without an incentive mechanism, there is no motivation for car owners to allow BS operator(s) to inject contents into their cars and participate in content sharing. In [15], the authors proposed a method for caching in an autonomous car. In their proposal, autonomous vehicles have cache storages to cache the data collected by the sensors, including metadata related to driving decisions. From the cache storage, it is possible to generate a driving decision based on similar previous cached driving decisions. In terms of machine learning, the authors in [16] discussed the application of machine learning in vehicular networks for optimizing network performance. To improve the performance of the connected vehicles, using deep reinforcement learning, the authors in [17] proposed the operations scheduling approach. Furthermore, in terms of offloading performance, the authors in [18] introduced a data offloading approach, where their approach uses deep reinforcement learning. Data offloading requires spectrum resources. Therefore, the authors in [19] proposed multi-agent reinforcement learning for spectrum sharing. Also, the authors in [20] highlighted that 5G can satisfy various requirements of connected vehicles in terms of traffic offloading. This enables connected vehicles to communicate with each other autonomously [21]. For the effective data transmission, the authors in [21] introduced data transmission approaches for vehicles using deep learning.

D. Contributions

To address the aforementioned challenges, we propose a deep learning based caching for self-driving cars, where caching decisions depend on passengers' features obtained using deep learning approaches and available communication, caching, and computation (3C) resources. As an extended version of our earlier work published in [22], the main contributions of this paper are summarized as follows:

- We propose deep learning based caching for self-driving cars as a new application of Convolutional Neural Network (CNN), where caching decisions depend on passengers' features obtained using CNN model and facial images of the passengers. Here, we assume the CNN model is trained and tested at DC using dataset. Then, the CNN model is deployed at MEC servers attached to the RSUs in close proximity to the self-driving cars,

Fig. 1. Content preferences based on users' features [8].

Fig. 2. Illustration of our system model.

where the self-driving cars can retrieve model with minimized delay.

- We propose a Multi-Layer Perceptron (MLP) framework at DC to predict the probability of infotainment contents to be requested in specific edge areas of MEC servers. Then, the MLP prediction output is deployed at MEC servers. During off-peak hours, each MEC server uses MLP output to identify the infotainment contents that have high predicted probability values of being requested in its area, downloads and caches them. To identify the infotainment contents that are likely to entertain its passengers and need to be cached in the self-driving car, each self-driving car downloads and stores the CNN model and MLP output from the MEC server. The self-driving car uses the CNN model for predicting passengers' features via facial images captured by its camera. Then, the self-driving car compares the CNN output with the MLP output using classification [23], [24] for identifying the contents that meet passengers' features.
- We propose a communication model that helps the self-driving car select available RSUs en-route. Then, the self-driving car uses these RSUs for retrieving identified infotainment contents that meet passengers' features and need to be cached.
- We propose a computation model for cached infotainment contents, where the cached contents can be served in different formats and qualities depending on demands. Therefore, we consider that MEC servers and self-driving cars have computation resources, which can be used to compute or process cached contents in different formats and qualities.
- We formulate an optimization problem that links the formulated models (deep learning-based caching, communication, and computation models) into one optimization problem whose goal is to minimize the content downloading delay. However, the formulated problem is shown to be non-convex. Therefore, to make it convex, we proposed a convex surrogate problem, which is an upper-bound of the formulated problem. Then, we apply the Block Successive Majorization-Minimization (BS-MM) technique [25] for solving it.

We chose BS-MM over other optimization techniques because BS-MM is a new technique that can decompose the original problem into small subproblems, where each subproblem can be solved separately.

Specifically, the novelties of our proposal over the related works in [7], [11], [14], [26]–[31] are as follows: To the best of our knowledge, we are the first to investigate self-driving car caching for infotainment contents, where caching decisions are based on passengers' features and available communication, caching, and computation resources.

The rest of the paper is organized as follows. We discuss the system model in Section II and present our deep learning based caching approach in Section III. In Section IV, we discuss the problem formulation and solution. We present a performance evaluation in Section V. Finally, we conclude the paper in Section VI.

II. SYSTEM MODEL

The system model of deep learning based caching is depicted in Fig. 2.

Data Center (DC): We assume that DC has higher computation resources than the self-driving car and RSU. Therefore, to minimize computation time, we use DC and dataset to make, train, and test deep learning models (CNN and MLP models) that will be used for predicting passengers features and infotainment contents need to be cached at the RSUs and in self-driving cars. To reduce the communication delay between the self-driving cars and the DC, the trained and tested CNN model and MLP output are deployed at MEC servers attached to the RSUs.

RoadSide Unit (RSU): As defined in 3GPP TS 22.185 V15.0.0 [32], we consider eNB-type RSU as an entity that supports both evolved NodeB (eNB) functionalities and V2X applications. We assume that each RSU $r \in \mathcal{R}$ has access to the DC via a wired backhaul of capacity $\omega_{r,DC}$, where $\mathcal{R}$ is the set of RSUs. Also, each RSU $r \in \mathcal{R}$ has an MEC server. Therefore, unless stated otherwise, we use the terms "RSU" and "MEC server" interchangeably.

Furthermore, as defined in 3GPP specifications in [32], we consider an MEC server as locally application server that serves a certain particular geographic area $n \in \mathcal{N}$, where $\mathcal{N} = \{1, 2, \ldots, N\}$ is a set of geographic areas. Furthermore, each MEC server $r \in \mathcal{R}$ has a cache storage of capacity c_r and computational resource of capacity p_r. Furthermore, during off-peak hours, by using backhaul communication resources, each RSU $r \in \mathcal{R}$ downloads CNN model and MLP output. Then, based on the MLP output, each MEC server downloads and cache infotainment contents that have high predicted probabilities of being requested in its area. We use $\mathcal{I}$ to denote a set of infotainment contents, where each content $i \in \mathcal{I}$ has a size of $S(i)$ Mb. Also, we consider that people from different areas may need different infotainment contents [33]. Therefore, it is more reasonable to cache infotainment contents at RSUs based on probabilities of being requested in particular areas.

Self-Driving Car: We consider $\mathcal{V}$ as a set of self-driving cars, where each self-driving car $v \in \mathcal{V}$ has OBU that can handle 4C to support caching and computation of infotainment contents for passengers. Furthermore, each self-driving $v \in \mathcal{V}$ can get broadband Internet service from RSU $r \in \mathcal{R}$ through a wireless link of capacity $\omega_{v,r}$. Each self-driving car $v \in \mathcal{V}$ has a cache storage capacity of c_v and computation capability of p_v. Furthermore, to predict the passengers' features, we use the CNN model. This helps in deciding which infotainment contents to request and cache in the self-driving car that meet passengers' features. During off-peak hours, each self-driving car $v \in \mathcal{V}$ downloads CNN model and MLP output from MEC server. By using the k-means and binary classification, the self-driving car compares its CNN prediction with the predicted output from MLP. This helps the self-driving car identify the infotainment contents that are appropriate to the passengers' features. Finally, the self-driving car downloads and caches the identified contents that meet passengers' features.

To avoid repetitive delivery of the same contents that require to use backhaul bandwidth, depending on demands, we consider that the computation resources of RSU and the self-driving car can be used to compute cached infotainment contents. As an example, content i' with the H.264 format may not be available in the cache storage. Instead, the cache storage may have content i with the MP4 format of the same content. Therefore, to satisfy the demand, by using the computational resource, cached infotainment content i can be converted to content i' (MP4 to H.264).

III. DEEP LEARNING BASED CACHING IN SELF-DRIVING CARS

In this section, to identify the infotainment contents need to be cached, we discuss the deep learning and recommendation model in Section III-A. For retrieving recommended contents requires communication resources. Therefore, in Section III-B, we discuss the communication model. For caching downloaded contents, we present the caching model in Section III-C. Furthermore, Based on the demands, cached contents can be converted or transcoded to

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Definition
$\mathcal{R}$	Set of RSUs, $|\mathcal{R}| = R$
$\mathcal{V}$	Set of self-driving cars, $|\mathcal{V}| = V$
$\mathcal{I}$	Set of contents, $|\mathcal{I}| = I$
$\mathcal{I}_r(n)$	Set of contents that need to be cached in area n of RSU r, $|\mathcal{I}_r(n)| = I_r(n)$
$\mathcal{U}$	Set of consumers of contents, $|\mathcal{U}| = U$
$\boldsymbol{x}$	Input of MLP
$\tilde{\boldsymbol{y}}$	Output of MLP
$\boldsymbol{y}$	Ground truth for MLP
M	The number of input features
N	The number of geographic areas
c_r	Caching capacity of each RSU $r \in \mathcal{R}$
p_r	Computation capability of RSU $r \in \mathcal{R}$
c_v	Caching capacity of each car $v \in \mathcal{V}$
p_v	Computation capability of car $v \in \mathcal{V}$
$\tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$	Total delay experienced by each passenger $u \in \mathcal{U}_v$
ψ_u^v	Data rate for each passenger u via WiFi of self-driving car v

different formats by using computational resources, where the computation model is described in Section III-D.

A. Deep Learning and Recommendation Model

In this subsection, we discuss MLP for predicting infotainment contents need to be cached at RSUs nearby the self-driving cars, CNN model for predicting passengers' features, and recommendation model for identifying the contents that meet passengers' features and need to be cache in the self-driving cars.

1) Multi-Layer Perceptron (MLP): We propose MLP for predicting probabilities of contents to be requested in particular areas of RSUs. We choose MLP over other prediction methods such as AutoRegressive (AR) and AutoRegressive Moving Average (ARMA) models because MLP can cope with both linear and non-linear prediction problems [34]. We use a demographical dataset that will be described in Section V. The input and output are described as follows:

- *Input:* In the dataset, we have infotainment content names, rating, viewer's age, gender, and location as the input of MLP. Furthermore, for predicting the probabilities of contents to be requested in specific areas, we use $\boldsymbol{x} = (x_1, x_2, \ldots x_M)^T$ to denote the input vector, where the subscripts are used to denote the features such as content names, rating, viewer's age, gender, and location.
- *Output:* From the input, MLP tries to predict $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \tilde{y}_2, \ldots \tilde{y}_N)^T$ as the output vector and the subscripts are used to denote the geographic areas. Also, in the output layer, each area $n \in \mathcal{N}$ corresponds to one neuron, where the output layer predicts the probabilities of contents to be cached in each specific area $n \in \mathcal{N}$.

For MLP, we use l to denote the number of hidden layers, $\boldsymbol{x}$ for the input vector, $\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(l)}$ for the bias vectors, $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(l)}$ for the weight matrices at each hidden layer, and $\tilde{\boldsymbol{y}}$ for the output vector. $\tilde{\boldsymbol{y}}$ can be expressed as follows:

$$\tilde{\boldsymbol{y}} = f(\boldsymbol{W}^{(l)} \ldots f(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}) \ldots + \boldsymbol{b}^{(l)}). \quad (1)$$

where $f(.)$ is the activation function.

In our MLP, we use the Rectified Linear Unit (ReLU) as the activation function in all the layers except at the output layer. We chose ReLU over other activation functions, because it mitigates the vanishing gradient problem experienced by MLP during the training process [35]. Furthermore, in the output layer l, we use the softmax function as an activation function. The purpose of the softmax function is to squeeze the output vector $\tilde{\boldsymbol{y}}$ into a set of probability values, where softmax function is defined as:

$$softmax(\tilde{\boldsymbol{y}})^{(l)} = \frac{e^{\tilde{y}_l}}{\sum_{n=1}^{N} e^{\tilde{y}_n}}, \quad \text{for } l = 1, \ldots, N. \quad (2)$$

The output layer has N neurons that correspond to N areas of RSUs. Furthermore, for the error function, we chose the cross-entropy error function over other error functions since our MLP classifies the contents needs to be cached in N geographic areas of RSUs. This problem can be considered as a classification problem, where we interpret the output as probabilities of the contents to be requested in each specific area $n \in \mathcal{N}$. The cross-entropy error function $A(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ can be expressed as follows:

$$A(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = -\sum_{n=1}^{N} y_n \log \tilde{y}_n. \quad (3)$$

$A(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ calculates the cross-entropy between the estimated class probabilities $\tilde{\boldsymbol{y}}$ and the ground truth $\boldsymbol{y}$.

Finally, to reduce the communication delay between the self-driving car and DC, as the DC may be located far from the self-driving cars, the output of the MLP are downloaded and stored to the RSUs based on their areas.

2) Convolutional Neural Network (CNN): In our proposal, we do not focus on proposing new CNN model. Conversely, we focus on a new application of existing CNN model for automatic age, emotion, and gender prediction from facial images [36] in caching decision. We describe the CNN workflow for automatic age, emotion, and gender extraction as follows:

- *Input:* We consider $\boldsymbol{k}_0$ as the input image with three-dimensional space: height, width, and the number of color channels (red, green, and blue).
- *Convolution Layer:* The convolution layer applies filters to the input regions and computes the output of each neuron. Each neuron is connected to local regions of the input, and using dot products between the weight and local regions, the convolution layer produces a feature map $\boldsymbol{k}_j$. We use $\boldsymbol{k}_j$ to denote the feature map produced after convolution layer j.
- *RELU Layer:* In this layer, we apply the ReLU $(\max(0, \boldsymbol{k}_j))$ as an elementwise activation function. The ReLU keeps the size of its associated convolution layer j unchanged.

- *Max Pooling Layer:* After the convolution and RELU layers, we have a high-dimensional matrix. Therefore, for dimension reduction, we apply a max-pooling layer as a downsampling operation.
- *Fully-Connected Layer:* This layer is fully connected to all previous neurons and is used to compute the class scores that a face could potentially belong to. Here, we have two classes for gender (male and female), 101 classes for age (from 0 to 101), and 8 classes for emotion (anger, anticipation, disgust, fear, joy, sad, surprise, and trust). In other words, we use three fully-connected layers for age, gender, and emotion classification.
- *Softmax Layer and Output:* In this layer, for each facial image, we need to interpret the output as the probability values of classes for gender, emotion, and age that a facial image could potentially belong to. To achieve this, the softmax activation function is applied to the output of the fully-connected layers.

To reduce the communication delay between the self-driving cars and DC, the trained and tested CNN model is deployed to the RSUs. Then, each self-driving car $v \in \mathcal{V}$ downloads CNN model and uses it for predicting age, gender, and emotion of passengers from facial images. Once the facial image of a passenger is captured via a camera. The self-driving car can extract features such as eyes, nose, mouth, and chin and use them for classifying the passengers' faces into different age, emotion, and gender classes. As describe in below recommendation model, this helps the self-driving car identify the infotainment contents that meet passengers' features as recommended contents to cache. Here, we assume that the passengers are aware of the presence of the camera. In other words, the self-driving cars have warning signs that inform passengers on the presence of the cameras. The same techniques were used in the deployment of public video surveillance at streets or public places [37].

3) Recommendation Model: The workflow of the recommendation model for self-driving cars is illustrated in Fig. 3 and described as follows:

- *Step 1:* Each self-driving car $v \in \mathcal{V}$ downloads the MLP output and CNN model from MEC server attached to RSU.
- *Step 2:* By using the k-means algorithm for age and emotion-based grouping and binary classification for gender-based grouping on the MLP output, each self-driving car $v \in \mathcal{V}$ creates age, gender, and emotion-based clusters of content consumers and generates an initial recommendation for the contents that need to be cached and have high predicted probability values for being requested.
- *Step 3:* For each new passenger $u \in \mathcal{U}$, the self-driving car uses the CNN model for predicting its age, gender, and emotion from facial image.
- *Step 4:* The self-driving car uses these passenger's features to calculate the similarity of passenger $u \in \mathcal{U}$ with the existing users (i.e., content consumers) in age, gender, and emotion-based clusters. Then, based on the similarity calculation, each passenger $u \in \mathcal{U}$ will be assigned to a cluster.
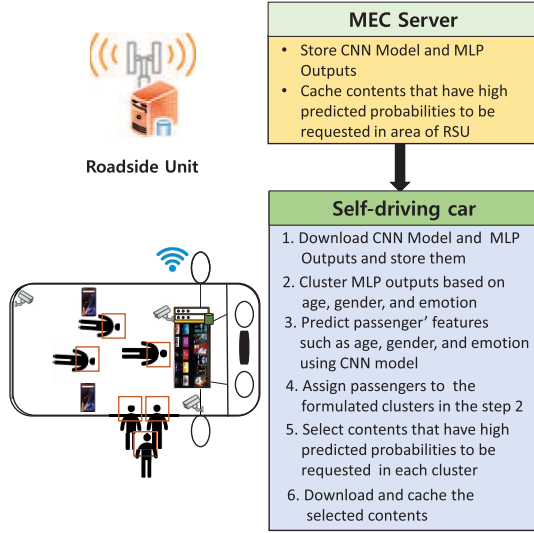
Fig. 3. Recommendation model for self-driving car.

Fig. 4. Communication planning for self-driving car.

• *Step 5:* After clustering the passenger(s), self-driving car $v \in \mathcal{V}$ selects top contents that have high predicted probability values for being requested as recommended contents to cache.

• *Step 6:* Finally, self-driving car $v \in \mathcal{V}$ downloads the recommended contents via RSU and caches them in its cache storage c_v.

For the k-means algorithm, first, we use age as numerical data. We denote $\tilde{\mathbf{y}}_n$ as the MLP output at each area $n \in \mathcal{N}$ and $\mathcal{X} = \tilde{\mathbf{y}}_n$ as the input of the k-means algorithm. The k-means partitions the consumer of the contents $\mathcal{X} = \{x_1, \ldots, x_U\}$ into K age-based clusters $\mathcal{X}_1, \ldots, \mathcal{X}_K$ such that $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$. In k-means, consumers are grouped into clusters based on their age. In addition, the clusters are disjoint $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$, $i \neq j$. The goal of k-means is to assign users to age-based clusters such that the objective function below is minimized:

$$\min_{\{\mathcal{X}_j\}_{j=1}^K} \sum_{j=1}^K \sum_{x_u \in \mathcal{X}_j} \|x_u - \tilde{x}_j\|^2, \tag{4}$$

where $\tilde{x}_j$ is the centroid of cluster $\mathcal{X}_j$, which is defined as

$$\tilde{x}_j = \frac{\sum_{x_u \in \mathcal{X}_j} x_u}{|\mathcal{X}_j|}. \tag{5}$$

In addition to age, consumers in the same age-based cluster can have different choice for contents based on emotion. Therefore, in each age-based cluster j, we use the k-means algorithm to class the consumers of contents in E emotion-based clusters (fear, sad, neutral, angry, disgusted, surprised). Therefore, in each emotion-based cluster e, we group users based on gender. For gender-based grouping, we apply binary classification as described in [24], which results in the formation of two groups, one group for females (denoted $\mathcal{G}_{je}^{\text{female}}$) and another group for males (denoted $\mathcal{G}_{je}^{\text{male}}$) such that $\mathcal{G}_{je}^{\text{female}} \cap \mathcal{G}_{je}^{\text{male}} = \emptyset$. Then, inside $\mathcal{G}_{je}^{\text{female}}$ and $\mathcal{G}_{je}^{\text{male}}$ clusters, which are sub-clusters of age-based cluster j

and emotion-based cluster e, the self-driving car select top infotainment contents that have high predicted probability values of being requested as the recommended contents to cache. Finally, the self-driving car downloads and caches recommended infotainment contents.

In this work, we assume that the self-driving cars and MEC servers download and store the CNN model and MLP output during off-peak hours. Therefore, hereafter, we only focus on recommended infotainment contents downloading, caching, and computing.

B. Communication Model for Retrieving Contents

Using a backhaul link of capacity $\omega_{r,DC}$, each MEC server downloads the infotainment contents that have high predicted probability values for being requested in its area $n \in \mathcal{N}$. The transmission delay for downloading contents from the DC to the MEC server r is:

$$\tau_r^{\text{DC}} = \frac{q^{\text{DC} \to r} \sum_{i \in \mathcal{I}_r(n)} S(i)}{\omega_{r,DC}}, \tag{6}$$

where $\mathcal{I}_r(n)$ for $n \in \mathcal{N}$ denotes the set of predicted contents that have high probability values for being requested in area n of RSU, and $q^{\text{DC} \to r}$ is a decision variable that indicates whether or not MEC server r is connected to the the DC, such that:

$$q^{\text{DC} \to r} = \begin{cases} 1, & \text{if MEC server } r \text{ is connected to the DC,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

As illustrated in Fig. 4, to have less variation in the transmission delay and hand-off before the self-driving car starts its journey, it can select RSUs that will be used to download the top-recommended contents. To discover RSUs located in a route of the self-driving car, Access Network Discovery and Selection Function (ANDSF) implemented in the cellular network and described in 3GPP TS 24.312 V15.0.0 [38] can be utilized. We assume each self-driving car $v \in \mathcal{V}$ moves in an area covered by macro Base Stations (BSs) and RSUs.

Therefore, to obtain RSU information such as coordinate and coverage, the self-driving car sends a request to the ANDSF server via a BS [39]. The request includes a geographic location of the self-driving car, speed, and direction. On the other hand, in the ANDSF server's feedback includes the coordinates and coverage of all RSUs available in the direction of the self-driving car.

Each self-driving car v computes the following distance $\tilde{d}_v^r$ between each RSU r and its route:

$$\tilde{d}_v^r = g_v^r \sin \alpha_v^r, \tag{8}$$

where α_v^r is the angle between the trajectory of movement of self-driving car v and the straight line from RSU $r \in \mathcal{R}$, and g_v^r is the geographical distance between self-driving car v and cache-enabled RSU r. In addition, each self-driving car v computes the following distance d_r^v remaining to reach each area covered by cache-enabled RSU $r \in \mathcal{R}$:

$$d_r^v = g_v^r \cos \alpha_v^r. \tag{9}$$

We defined ρ_v^r as the probability that RSU $r \in \mathcal{R}$ will be selected as a source of infotainment contents to be cached in self-driving car v as follows:

$$\rho_v^r = \begin{cases} 1, & \text{if } \tilde{d}_v^r = 0, \\ \dfrac{\tilde{d}_v^r}{\gamma_r} & \text{if } 0 < \tilde{d}_v^r < \gamma_r, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where γ_r is the radius of the area covered by RSU $r \in \mathcal{R}$. Therefore, we define q_v^r as a decision variable that indicates whether or not the self-driving car is connected to RSU $r \in \mathcal{R}$ as follows:

$$q_v^r = \begin{cases} 1, & \text{if } \rho_v^r > 0 \text{ and } d_r^v = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Equations (10) and (11) ensure that once the self-driving car v reaches an area covered by cache-enabled RSU $r \in \mathcal{R}$, it immediately starts downloading the recommended infotainment contents.

We assume each RSU r has a wireless channel of capacity $\omega_{v,r}$, where $\omega_{v,r}$ can be expressed as follows:

$$\omega_{v,r} = q_v^r B_r \log_2 \left(1 + \varphi_r |G_v^r|^2\right), \quad \forall v \in \mathcal{V}, \ r \in \mathcal{R}, \tag{12}$$

where B_r is the bandwidth for the car to RSU communications, G_v^r is the channel gain between RSU r and self-driving car v, and φ_r is the transmission power of RSU r. Therefore, based on the channel capacity, the transmission delay for downloading contents that meet passengers' features from the MEC server to self-driving car v is expressed as:

$$\tau_v^r = \frac{\sum_{\tilde{i}_f, \tilde{i}_m \in \mathcal{I}_r(n)} q_v^r \left(S(\tilde{i}_f)) + S(\tilde{i}_m)\right)}{\omega_{v,r}}, \tag{13}$$

where $\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}$ is the recommended infotainment content for female passengers and $\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}$ is the recommended infotainment content for male passengers in each age and emotion-based cluster in area n, where $\tilde{i}_f, \tilde{i}_m \in \mathcal{I}_r(n)$.

Based on self-driving car's speed, we consider t_v^r as the time required by self-driving car $v \in \mathcal{V}$ to leave an area covered by RSU r. We can calculate t_v^r as follows:

$$t_v^r = \frac{2q_v^r \gamma_r}{\mu_v}, \tag{14}$$

where μ_v is the speed of self-driving car v. When $\tau_v^r < t_v^r$, the self-driving can easily download the recommended infotainment content in the area coverage by RSU r. However, when $\tau_v^r \geq t_v^r$, the self-driving car can select the next RSU to use for downloading recommended infotainment contents.

Each self-driving car v has a WiFi Router on board that can be used to provide WiFi connectivity to its passengers. However, in the self-driving car, passengers are free to choose their appropriate connections. Here, we aim to minimize delay experienced by the passengers that are inside of the self-driving car and use WiFi connectivity of the self-driving car for getting infotainment contents. Therefore, the instantaneous data rate for each passenger u via the WiFi of self-driving car v is given by:

$$\psi_u^v = \frac{q_u^v \varphi_v \tilde{\psi}_u^v \xi_u^v(|\mathcal{U}_v|)}{|\mathcal{U}_v|}, \quad \forall u \in \mathcal{U}_v, \ v \in \mathcal{V}_v, \tag{15}$$

where φ_v is the WiFi throughput efficiency factor and $|\mathcal{U}_v|$ is the number of passengers that are connected simultaneously to the WiFi of self-driving car v, where $\mathcal{U}_v \subset \mathcal{U}$. We use φ_v to denote the overhead related to the MAC protocol layering. Furthermore, $\tilde{\psi}_u^v$ is the maximum theoretical data rate that the WiFi can handle. Furthermore, $\xi_u^v(|\mathcal{U}_v|)$ is a channel utilization function, which is a function of the number of passengers connected simultaneously to the WiFi [40]. $\xi_u^v(|\mathcal{U}_v|)$ is used to determine the impact of contention over the WiFi throughput. Also, we use q_u^v as a decision variable that indicates whether or not passenger u is connected to the WiFi of self-driving v, specifically:

$$q_u^v = \begin{cases} 1, & \text{if the passenger } u \text{ is connected to the} \\ & \text{WiFi of the self-driving car } v, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

For each passenger $u \in \mathcal{U}_v$, based on its instantaneous data rate ψ_u^v, the transmission delay τ_u^v for downloading content i from self-driving car v is given by:

$$\tau_u^v = \frac{\sum_{i \in \mathcal{I}_r(n)} q_u^v \left(S(\tilde{i}_f)) + S(\tilde{i}_m)\right)}{\psi_u^v}. \tag{17}$$

C. Caching Model for Retrieved Contents

We assume that the cache storage c_v of each self-driving car v is limited. Therefore, the sizes of the recommended infotainment contents that need to be downloaded from the MEC server and cached in the self-driving car must satisfy the cache resource constraint, which is expressed as follows:

$$q_v^r \sum_{j=1}^{K} \left(\sum_{\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f)) + \sum_{\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \right) \leq c_v, \tag{18}$$

where $o_v^{\tilde{i}_f} \in \{0, 1\}$ is the decision variable that indicates whether or not self-driving car v has to cache infotainment content $\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}$, where $o_v^{\tilde{i}_f}$ is given by:

$$o_v^{\tilde{i}_f} = \begin{cases} 1, & \text{if self-driving car } v \text{ caches the content } \tilde{i}_f, \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

On the other hand, we let $o_v^{\tilde{i}_m} \in \{0, 1\}$ be the decision variable that indicates whether or not self-driving car v has to cache infotainment content $\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}$, where $o_v^{\tilde{i}_m}$ is given by:

$$o_v^{\tilde{i}_m} = \begin{cases} 1, & \text{if self-driving car } v \text{ caches the content } \tilde{i}_m, \\ 0, & \text{otherwise.} \end{cases} \quad (20)$$

Furthermore, for analyzing cache storage utilization, which is based on cache hit and cache miss, we assume that $\tilde{i}_f$ and $\tilde{i}_m$ are cached in the same cache storage c_v. Therefore, we omit the subscript and superscript on content, and use i to denote either content $\tilde{i}_f$ or $\tilde{i}_m$.

We use $h_i^{u \to v} \in \{0, 1\}$ to denote the cache hit indicator at self-driving car v for content $i \in \mathcal{I}_r(n)$ requested by customer $u \in \mathcal{U}$:

$$h_i^{u \to v} = \begin{cases} 1, & \text{if content } i \text{ requested by consumer } u \\ & \text{is returned from self-driving car } v, \\ 0, & \text{otherwise.} \end{cases} \quad (21)$$

In the case of a cache miss ($h_i^{u \to v} = 0$), the self-driving car needs to forward the demand for content i to its associated MEC server. Based on the MLP output at the RSU, we assume that the MEC server caches the contents that have high probabilities of being requested in area n, where cache allocation has to satisfy the following constraint:

$$q^{\text{DC} \to r} \sum_{i \in \mathcal{I}_r(n)} o_r^i S(i) \leq c_r, \quad (22)$$

where o_r^i is a decision variable that indicates whether or not MEC server r has to cache content $i \in \mathcal{I}_r(n)$, defined as follows:

$$o_r^i = \begin{cases} 1, & \text{if MEC server } r \text{ caches content } i \in \mathcal{I}_r(n), \\ 0, & \text{otherwise.} \end{cases} \quad (23)$$

Furthermore, we use $h_i^{r \to v} \in \{0, 1\}$ to denote the cache hit indicator at the MEC server for content $i \in \mathcal{I}_r(n)$ requested by self-driving $v \in \mathcal{V}$:

$$h_i^{r \to v} = \begin{cases} 1, & \text{if the content } i \text{ requested by self-dring} \\ & \text{car } v \text{ is cached at MEC server } r, \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

However, when the MEC server does not have content i in its cache storage, the MEC server forwards the demand for content i to the DC via a wired backhaul link.

D. Computation Model for Cached Contents

In self-driving cars, a passenger may request a content format (e.g., H.264) that is not available in the cache storage c_v. Instead, the cache storage may have other content formats (e.g., MP4) for the same content that can be transcoded to the desired format (H.264).

Therefore, to adopt this process of serving cached content after computation, we define the following decision variable:

$$h_{i'}^{v \to u} = \begin{cases} 1, & \text{if content } i' \text{ requested by consumer } u \\ & \text{is returned by car } v \text{ after computation,} \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

To ensure that self-driving car v returns only one format of the requested content, the following constraint should be satisfied:

$$h_i^{u \to v} + h_{i'}^{v \to u} \leq 1. \quad (26)$$

We assume that converting content i to content i' requires computation resource $p_v^{i \to i'}$ of self-driving car v, where the computational resource allocation $p_v^{i \to i'}$ is given by:

$$p_v^{i \to i'} = p_v \frac{h_i^{u \to v} \varrho_v^{i \to i'} z^{i \to i'}}{\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} h_i^{u \to v} \varrho_v^{i \to i'} z^{i \to i'}}, \quad \forall v \in \mathcal{V}, \quad (27)$$

where $z^{i \to i'}$ is the computation workload or intensity in terms of CPU cycles per bit required for converting cached content i to i', while $\varrho_v^{i \to i'}$ is the computation decision variable, which is expressed as:

$$\varrho_v^{i \to i'} = \begin{cases} 1, & \text{if the cached content } i \text{ is converted to the} \\ & \text{desired format } i' \text{ in self-driving car } v. \\ 0, & \text{otherwise.} \end{cases} \quad (28)$$

In (27), for computational resources allocation, we use weighted proportional allocation [41] because it is simple to implement in practical communication systems such as Vehicular Ad-hoc Networks (VANETs) and 4G & 5G cellular networks [7]. Furthermore, computation resource allocation should satisfy the following constraint:

$$\sum_{u=1}^{U} \sum_{i=1}^{I_r(n)} q_u^v h_i^{u \to v} \varrho_v^{i \to i'} p_v^{i \to i'} \leq p_v. \quad (29)$$

In addition, converting content i to content i' requires executing time. Therefore, in self-driving car v, the execution time $\tau_v^{i \to i'}$ is given by:

$$\tau_v^{i \to i'} = \frac{q_u^v h_i^{u \to v} \varrho_v^{i \to i'} z^{i \to i'} S(i)}{p_v^{i \to i'}}. \quad (30)$$

When constraint (29) cannot be satisfied due to insufficient computational resource for converting content i into the requested content i', the self-driving car forwards the demand for content i' to the MEC server.

At the MEC server, to convert cached content i into content i', it requires an execution time of $\tau_r^{i \to i'}$. Thus, the execution time at the MEC server is given by:

$$\tau_r^{i \to i'} = (1 - \varrho_v^{i \to i'}) \left(\frac{q_v^r h_i^{r \to v} \varrho_r^{i \to i'} z^{i \to i'} S(i)}{p_r^{i \to i'}} \right), \quad (31)$$

where $p_r^{i \to i'}$ is the required computation resource of MEC server r for converting content i to content i'. $p_r^{i \to i'}$ can be calculated in the same manner used in (27). We define a $\varrho_r^{i \to i'}$ computation decision variable, where $\varrho_r^{i \to i'}$ is expressed as follows:

$$\varrho_r^{i \to i'} = \begin{cases} 1, & \text{if the cached content } i \text{ is converted to} \\ & \text{; desired format } i' \text{ at MEC server,} \\ 0, & \text{otherwise,} \end{cases} \quad (32)$$

We assume that the computation resources at the MEC server are limited, where computation allocation has to satisfy the following constraint:

$$\sum_{v=1}^{V} \sum_{i=1}^{I_r(n)} q_v^r h_i^{r \to v} \varrho_r^{i \to i'} p_r^{i \to i'} \leq P_r. \quad (33)$$

In addition, we define $h_{i'}^{r \to v}$ as a decision variable that indicates whether or not the MEC server returns the requested content i' to self-driving car v after computation, where $h_{i'}^{r \to v}$ is given by:

$$h_{i'}^{r \to v} = \begin{cases} 1, & \text{if content } i' \text{ requested by car } v \text{ is returned} \\ & \text{by MEC server } r \text{ after computation,} \\ 0, & \text{otherwise.} \end{cases} \quad (34)$$

To ensure that converting cached content i to the requested content i' is performed exactly at one location, either at the self-driving car or at MEC server, and self-driving car or MEC server sends exactly one format of content, we formulate the following constraints:

$$q_u^v (h_i^{u \to v} + h_{i'}^{v \to u}) + q_v^r \eta_v (h_i^{r \to v} + h_{i'}^{r \to v}) \leq 1, \quad (35)$$

$$\varrho_v^{i \to i'} + q_v^r (1 - \varrho_v^{i \to i'}) \leq 1. \quad (36)$$

Here, we use $\eta_v = 1 - (h_i^{u \to v} + h_{i'}^{v \to u})$. However, if the above constraints cannot be satisfied due to limited computation and caching resources, MEC server submits the request for content i' to the DC.

IV. PROBLEM FORMULATION AND SOLUTION

In this section, we present our optimization problem for minimizing delay in downloading the infotainment contents in Section IV-A. Then, in Section IV-B, we present a solution of the formulated optimization problem.

A. Problem Formulation

In the self-driving car, to coordinate deep Learning & recommendation, communication, caching, and computation models, we formulate an optimization problem that links the formulated models into one problem whose goal is to minimize total delay $\tau_u^{\text{Tot}}(\mathbf{q}, \mathbf{h}, \varrho)$ for retrieving infotainment contents, where $\tau_u^{\text{Tot}}(\mathbf{q}, \mathbf{h}, \varrho)$ is given by:

$$\tau_u^{\text{Tot}}(\mathbf{q}, \mathbf{h}, \varrho) = \tau_u^v h_i^{u \to v} + h_{i'}^{v \to u} \varrho_v^{i \to i'} \tau_v^{i \to i'} \\ + \left(1 - (h_i^{u \to v} + \varrho_v^{i \to i'} h_{i'}^{v \to u})\right) \\ \times (\tau_v^r h_i^{r \to v} + \tau_r^{i \to i'} \varrho_r^{i \to i'} h_{i'}^{r \to v}) \\ + (1 - (h_i^{r \to v} + \varrho_r^{i \to i'} h_{i'}^{r \to v})) \tau_r^{\text{DC}}. \quad (37)$$

In the above equation (37), a requested infotainment content can be retrieved in the self-driving car. However, if the requested content can not be retrieved in a self-driving car, the self-driving car sends a request to RSU, where RSU can return the requested content. In the worst case, if the requested content can not be retrieved from self-driving car or RSU, DC can be used. Therefore, our optimization problem can be expressed as follows:

$$\underset{\mathbf{q}, \mathbf{h}, \varrho}{\text{minimize}} \quad \sum_{u=1}^{U} \tau_u^{\text{Tot}}(\mathbf{q}, \mathbf{h}, \varrho) \quad (38)$$

$$\text{subject to: } \sum_{v=1}^{V} q_v^r \leq 1, \ \forall r \in \mathcal{R}, \quad (38a)$$

$$q_v^r \sum_{j=1}^{k} \left(\sum_{\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f) \right) \\ + \sum_{\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \right) \leq c_v, \quad (38b)$$

$$\sum_{u=1}^{U} \sum_{i=1}^{I_r(n)} q_u^v h_i^{u \to v} \varrho_v^{i \to i'} p_v^{i \to i'} \leq p_v, \\ \forall v \in \mathcal{V}, \ \forall n \in \mathcal{N}, \quad (38c)$$

$$q_u^v (h_i^{u \to v} + h_{i'}^{v \to u}) + q_v^r \eta_v (h_i^{r \to v} + h_{i'}^{r \to v}) \leq 1, \quad (38d)$$

$$q_u^v \varrho_v^{i \to i'} + q_v^r (1 - \varrho_v^{i \to i'}) \leq 1. \quad (38e)$$

The constraint in (38a) ensures that the self-driving car has to be connected to RSU $r \in \mathcal{R}$ to download the contents. The constraints in (38b) and (38c) guarantee that the caching and computational resource allocations have to be less than or equal to the available caching and computational resources of the self-driving car. Furthermore, constraint in (38b) is based on CNN output, where the self-driving car caches the contents based on passengers' features such as age, emotion, and gender. The constraint in (38d) ensures that the self-driving car or MEC server returns only one format of the requested content (either cached or computed from the cached content). The constraint (38e) ensures that converting i to i' is only executed at one location, either in self-driving car v or at MEC server r.

The formulated optimization problem in (38) is non-convex problem which makes it complicated to solve. Therefore, in the next Subsection IV-B, we propose a proximal convex surrogate problem of the formulated problem in (38) and apply Block Successive Majorization-Minimization (BS-MM) [25] for solving proximal convex surrogate problem.

B. Proposed Solution: Distributed Algorithm for Deep Learning Based Caching

For solving our optimization problem, we use BS-MM described in [25], [42]. We chose BS-MM over other distributed algorithms such as DC (Difference of Convex) programming, concave-convex, and successive convex approximation because BS-MM is a new approach that allows to partition the problem into blocks and applies MM to one

block of variables while keeping the values of the other blocks fixed [25]. The BS-MM may have computation overhead due to the computation of the best solution at each iteration, especially when the size of the problem is very large. Also, when BS-MM is fast, it may skip the true local minimum. If BS-MM is too slow, it may never converge because it tries to find a local minimum at each iteration. Therefore, to overcome these BS-MM challenges and ensure that all blocks are utilized, as suggested in [43], we use different selection rules such as Cyclic, Gauss-Southwell, and Randomized described in [43]. To apply BS-MM in (38), we consider $\mathcal{Q} \triangleq \{\boldsymbol{q} : \sum_{u=1}^{U} q_u^v \leq 1, \ q_u^v \in [0,1]\}$, $\mathcal{H} \triangleq \{\boldsymbol{h} : \sum_{u=1}^{U} (h_i^{u \to v} + h_{i'}^{v \to u}) + (1 - (h_i^{u \to v} + h_{i'}^{v \to u})) (h_i^{r \to v} + h_i^{r \to v}) \leq 1, h_i^{u \to v}, h_{i'}^{v \to u}, h_i^{r \to v}, h_i^{r \to v} \in [0,1]\}$, and $\mathcal{P} \triangleq \{\boldsymbol{\varrho} : \sum_{i,i' \in \mathcal{I}} \varrho_v^{i \to i'} + (1 - \varrho_v^{i \to i'}) \varrho_r^{i \to i'} \leq 1, \varrho_v^{i \to i'}, \varrho_r^{i \to i'} \in [0,1]\}$ as non-empty and closed sets of the relaxed variables $\boldsymbol{q}$, $\boldsymbol{h}$, and $\boldsymbol{\varrho}$, respectively. Therefore, to simplify our notation, we use $\mathcal{F}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$ to denote (38), where $\mathcal{F}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$ is expressed as follows:

$$\mathcal{F}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho}) = \sum_{u=1}^{U} \tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho}). \tag{39}$$

Both (38) and (39) have the same constraints. Therefore, to solve (39), we use the following steps:

- In the first step, called majorization, we propose a proximal convex surrogate problem $\mathcal{F}_j(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$ (40) of the formulated problem in (39), which is an upper-bound of (39).
- In the second step, called minimization, instead of minimizing (39) which is intractable, we minimize its proximal convex surrogate function $\mathcal{F}_j(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$(40).

The success of BS-MM relies on the surrogate function. Therefore, a surrogate function that is easy to solve and upper-bound of of the formulated problem in (39) is preferable. To achieve this, in the majorization step, we use the proximal upper-bound minimization technique described in [25]. Then, we propose the following proximal convex surrogate problem $\mathcal{F}_j(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$ (40) of the formulated problem in (39) by adding the quadratic term $(\frac{\varrho_j}{2} \|(\boldsymbol{q}_j - \boldsymbol{q}^{(0)})\|^2)$ to (39):

$$\mathcal{F}_j(\boldsymbol{q}_j, \boldsymbol{q}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{\varrho}^{(t)}) := \mathcal{F}(\boldsymbol{q}_j, \boldsymbol{q}^{(0)}, \boldsymbol{h}^{(0)}, \boldsymbol{\varrho}^{(0)}) + \frac{\alpha_j}{2} \|(\boldsymbol{q}_j - \boldsymbol{q}^{(0)})\|^2, \tag{40}$$

where $\boldsymbol{q}^{(0)}$, $\boldsymbol{h}^{(0)}$, and $\boldsymbol{\varrho}^{(0)}$ are the initial feasible points. Furthermore, the surrogate function in (40) can be applied to other vectors $\boldsymbol{h}$ and $\boldsymbol{\varrho}$. In addition, the quadratic term $(\frac{\alpha_j}{2} \|(\boldsymbol{q}_j - \boldsymbol{q}^{(0)})\|^2)$ makes the problem (40) to be convex and upper-bound of (39). In the minimization step, we minimize the surrogate function $\mathcal{F}_j(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$ (40) by taking steps proportional to the negative of the gradient in the direction toward the formulated problem in (39), where $\mathcal{J}^t$ is a set of indexes at each iteration t and α_j is a positive penalty parameter for $j \in \mathcal{J}^t$. At each iteration $t+1$, the solution is updated by solving the following problems:

$$\boldsymbol{q}_j^{(t+1)} \in \min_{\boldsymbol{q}_j \in \mathcal{Q}} \mathcal{F}_j(\boldsymbol{q}_j, \boldsymbol{q}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{\varrho}^{(t)}), \tag{41}$$

$$\boldsymbol{h}_j^{(t+1)} \in \min_{\boldsymbol{h}_j \in \mathcal{H}} \mathcal{F}_j(\boldsymbol{h}_j, \boldsymbol{h}^{(t)}, \boldsymbol{q}_j^{(t+1)}, \boldsymbol{\varrho}^{(t)}), \tag{42}$$

$$\boldsymbol{\varrho}_j^{(t+1)} \in \min_{\boldsymbol{\varrho}_j \in \mathcal{P}} \mathcal{F}_j(\boldsymbol{\varrho}_j, \boldsymbol{\varrho}^{(t)}, \boldsymbol{q}_j^{(t+1)}, \boldsymbol{h}_j^{(t+1)}). \tag{43}$$

To solve our problems in (41), (42), and (43) we use vectors $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ of relaxed variables. Therefore, we need to enforce $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ to be vectors of binary variables. To achieve this, we apply the rounding techniques described in [44]. As an illustration example, for a solution $q_v^{r*} \in \boldsymbol{q}_j^{(t+1)}$, we define the rounding threshold $\varphi \in (0,1)$, such that the enforced binary value of q_v^{r*} is given by:

$$q_v^{r*} = \begin{cases} 1, & \text{if } q_v^{r*} \geq \varphi, \\ 0, & \text{otherwise.} \end{cases} \tag{44}$$

As highlighted in [7], [45], the rounding technique may violate 3C resource constraints. Therefore, to overcome this issue, we solve $\mathcal{F}_j$ in the form $\mathcal{F}_j + \beta_v \Delta_v$ by updating the constrains in (38a), (38b), and (38c) as follows:

$$\sum_{v=1}^{V} q_v^r a_v^r \leq 1 + \Delta_{v_a}, \quad \forall r \in \mathcal{R}, \tag{45}$$

$$\sum_{u=1}^{U} \sum_{i=1}^{I_r(n)} q_u^v h_i^{u \to v} \varrho_v^{i \to i'} p_v^{i \to i'} \leq p_v + \Delta_{v_p}, \quad \forall v \in \mathcal{V}, \tag{46}$$

$$q_v^r \sum_{j=1}^{k} \left(\sum_{\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f) \right) + \sum_{\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \leq c_v + \Delta_{v_c}, \tag{47}$$

where $\Delta_v = \Delta_{v_a} + \Delta_{v_p} + \Delta_{v_c}$ is the maximum violation of the 3C resource constraints and β_v as the weight parameter of Δ_v. Furthermore, the values of Δ_{v_a}, Δ_{v_p}, and Δ_{v_c} are given by:

$$\Delta_{v_a} = \max\{0, \sum_{v=1}^{V} q_v^r a_v^r - 1\}, \quad \forall r \in \mathcal{R}, \tag{48}$$

$$\Delta_{v_p} = \max\{0, \sum_{u=1}^{U} \sum_{i=1}^{I_r(n)} q_u^v h_i^{u \to v} \varrho_v^{i \to i'} p_v^{i \to i'} - p_v\}, \quad \forall v \in \mathcal{V}, \tag{49}$$

$$\Delta_{v_c} = \max\{0, q_v^r \sum_{j=1}^{k} \left(\left(\sum_{\tilde{i}_f \in \mathcal{G}_{je}^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f) \right) + \sum_{\tilde{i}_m \in \mathcal{G}_{je}^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \right) - c_v\}. \tag{50}$$

Therefore, to ensure that the best solution is achieved, we use the integrality gap described in [44].

Definition 1 (Integrality gap): For the problems $\mathcal{F}_j + \beta_v \Delta_v$ and $\mathcal{F}_j$, the integrality gap is expressed as follows:

$$\phi_j = \min_{\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho}} \frac{\mathcal{F}_j}{\mathcal{F}_j + \beta_v \Delta_v}. \tag{51}$$

The best solutions of $\mathcal{F}_j$ and $\mathcal{F}_j + \beta_v \Delta_v$ are obtained when $\phi_j \leq 1$.

We propose a distributed algorithm (Algorithm 1), which is based on BS-MM [25]. We assume that the MLP output and

Algorithm 1 Distributed Algorithm for Deep Learning Based Caching

---

1: **Preconditions:** MLP output and CNN models are deployed to the RSUs and in self-driving car;
2: **Input:** U: A vector of passengers, ω_v^r: wireless link capacities, $\mathcal{X}$: Vector of recommended contents for $\mathcal{G}_{je}^{\text{female}}$ and $\mathcal{G}_{je}^{\text{male}}$ in self-driving car v, ψ_u^v, p_v, and c_v;
3: **Output:** q^*, h^*, ϱ^*;
4: Initialize $t = 0$;
5: Find initial feasible points $q^{(0)}$, $h^{(0)}$, $\varrho^{(0)}$;
6: **repeat**
7:   Choose index set $\mathcal{J}^t$;
8:   Let $q_j^{(t+1)} \in \min\limits_{q_j \in \mathcal{Q}} \mathcal{F}_j(q_j, q^{(t)}, h^{(t)}, \varrho^{(t)})$ (41);
9:   Set $q_k^{t+1} = q_k^t, \forall k \notin \mathcal{J}^t$ and solve $\min\limits_{q_j \in \mathcal{Q}} \mathcal{F}_j(q_j, q^{(t)}, h^{(t)}, \varrho^{(t)})$;
10: For $h_j^{(t+1)}$ and $\varrho_j^{(t+1)}$, restart from step 4, salve (42) and (43);
11: $t = t + 1$;
12: **until** $\lim\limits_{t \to \infty} \inf\limits_{q,h,\varrho} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$;
13: By rounding technique, enforce $q_j^{(t+1)}$, $h_j^{(t+1)}$, and $\varrho_j^{(t+1)}$ to be vectors of binary variables;
14: Solve $\mathcal{F}_j + \beta_v \Delta_v$ and compute ϕ_j until $\phi_j \leq 1$;
15: Then, consider $q^* = q_j^{(t+1)}$, $h^* = h_j^{(t+1)}$, and $\varrho^* = \varrho_j^{(t+1)}$ as a solution.

CNN model are already deployed at RSUs and in self-driving car. We consider a vector of passengers, vector of RSUs, vector of wireless link capacities, vector of recommended contents that need to be cached in self-driving car v, ψ_u^v, p_v, and c_v as the input. First, Algorithm 1 finds the initial feasible points $q^{(0)}$, $h^{(0)}$, and $\varrho^{(0)}$. Then, Algorithm 1 starts an iterative process by choosing an index set $\mathcal{J}^t$ at each iteration t. At each iteration $t + 1$, the solution is updated by solving the problems (41), (42), and (43) until $\lim\limits_{t \to \infty} \inf\limits_{q,h,\varrho} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$, where $\lim\limits_{t \to \infty} \inf\limits_{q,h,\varrho} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$ is the convergence criteria. Therefore, when $\lim\limits_{t \to \infty} \inf\limits_{q,h,\varrho} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$, Algorithm 1 considers $q_j^{(t+1)}$, $h_j^{(t+1)}$, and $\varrho_j^{(t+1)}$ as a solution. Then, Algorithm 1 forces the solution $q_j^{(t+1)}$, $h_j^{(t+1)}$, and $\varrho_j^{(t+1)}$ to be vectors of binary variables via the rounding technique, where Algorithm 1 solves $\mathcal{F}_j + \beta_v \Delta_v$ and computes ϕ_j. Finally, when $\phi_j \leq 1$, Algorithm 1 considers $q^* = q_j^{(t+1)}$, $h^* = h_j^{(t+1)}$, and $\varrho^* = \varrho_j^{(t+1)}$ as a solution which does not violate 3C resource constraints. Furthermore, for the convergence of the proposed algorithm, based on the convergence of MM defined and proved in [25], we make the following remark:

Remark 1 (Convergence of the proposed algorithm): Based on the MM algorithm [25], the proposed Algorithm 1, which is based on BS-MM, converges to coordinate-wise minimum point which is stationary point, when the vectors
$q^* = q_j^{(t+1)}$, $h^* = h_j^{(t+1)}$, *and* $\varrho^* = \varrho_j^{(t+1)}$ *cannot find a better minimum direction, i.e.,* $\lim\limits_{t \to \infty} \inf\limits_{q,h,\varrho} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$.

For complexity analysis of the proposed Algorithm 1, based on complexity analysis described in [43], we make the following remark:

Remark 2 (Complexity of the proposed Algorithm 1): The Algorithm 1, which is based on BS-MM, uses proximal upper-bound minimization technique. This makes it fall under the BSUM framework [43]. Therefore, for the iteration index $j \in \mathcal{J}^t$, the Algorithm 1 has $\mathcal{O}(1/j)$ iteration complexity, which is sub-linear.

V. SIMULATION RESULTS AND ANALYSIS

In this section, we present a performance evaluation of the proposed deep learning-based caching in self-driving cars. We use Google Maps Services [46] for the self-driving car mobility analysis, Keras with Tensorflow [47] for the deep learning simulation, and pandas [48] for data analysis.

A. Simulation Setup

To predict the probabilities of contents to be requested in specific areas of MEC servers, we use a well-known dataset called Movie-Lens Dataset [49]. In the dataset, we have movies with related information such as movie titles, release date, and genre of movies such as comedy, drama, and documentary. We associate the emotion with the genre of movies, where sad users recommended to watch drama movies, disgust users recommended to watch musical movies, anger users recommended to watch comedy movies, anticipate users recommended to watch thriller movies, fear users recommended to watch adventure movies, joy users recommended to watch thriller movies, trust users recommended to watch western movies, and surprise users recommended to watch fantasy movies. However, the dataset does not have movie sizes and formats. Since our deep learning-based caching scheme uses content size, we randomly generate size $S(i)$ for each movie i in the range from $S(i) = 317$ to $S(i) = 750$ Mb and randomly assign each movie i a format. Furthermore, we have user's information such as age (as shown in Fig. 5), gender, rating, and ZIP codes. To identify the areas of users, we convert the ZIP codes into longitude and latitude coordinates and deploy 6 RSUs to the specific areas based on the movie watching counts, rankings, and the locations of users. We use MLP with 2 layers (for input and output) and 2 hidden layers to predict the probabilities of contents to be requested in specific areas of RSUs. In MLP, each layer has 32 neurons except the output layer which has 6 neurons. In the output layer, 6 neurons correspond to the probabilities of contents to be cached in specific areas of 6 RSUs. We use 60% of the dataset for training and 40% for testing. Furthermore, the learning rate is set to be equal to 0.002, while the batch size equals to 32.

With the departure time and locations of the RSUs, the Google Maps service provides the distance and duration to reach each RSU $r \in \mathcal{R}$, where the duration is based on traffic conditions between the source and destination. Based on the distance (in terms of km) and duration (in terms of
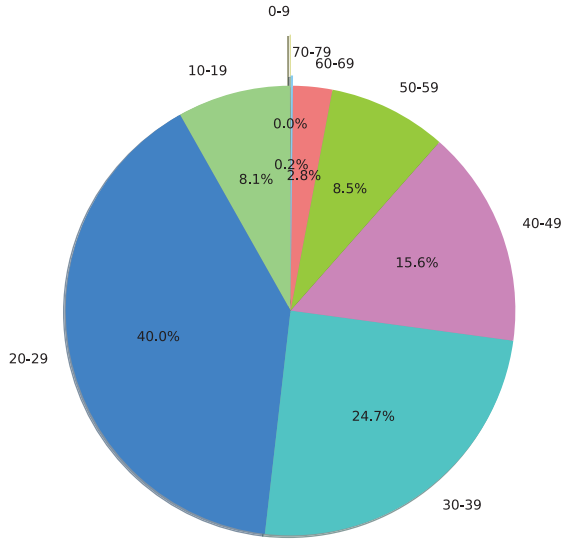
Fig. 5.   Visualization of the used dataset [49] for movie watching based on age.

TABLE II

THE USED ROUTE FOR THE SELF-DRIVING BUS

Route	Distance (Km)	Max. speed (Km/h)	RSUs
1	54.62	109.016	$1-2$
2	53.82	107.34	$2-3$
3	54.02	108.17	$3-4$
4	52.83	105.38	$4-5$
5	55.66	111.33	$5-6$

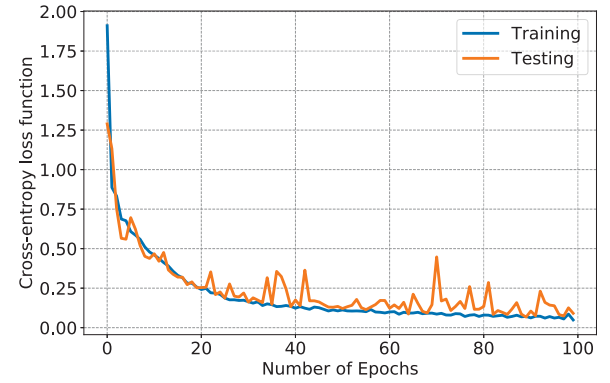Fig. 6.   Visualization of the used passengers' features for the self-driving bus.

Fig. 7.   Minimization of error function for predicting the probability of movies to be requested in the specific areas of RSUs (acc: 97.82%).

hours), we can calculate the speed (in terms of km/h) of the self-driving car and find the RSUs that the self-driving car can connect to for retrieving contents. However, based on Google Maps service [46], the distances between RSUs are very large. Therefore, to have realistic distances between RSUs, we update the RSU locations and create a routing table summarized in Table II, where the self-driving car starts its journey at RSU 1 and ends at RSU 6. We set each RSU $r \in \mathcal{R}$ to be connected to the DC with a wired backhaul of capacity ranging from $\omega_{r,DC} = 60$ to $\omega_{r,DC} = 70$ Mbps. We assume that each RSU $r \in \mathcal{R}$ has a bandwidth of $\omega_{v,r} = 10$ MHz. On the other hand, each MEC server $r \in \mathcal{R}$ has a CPU of capacity $p_r = 3.6$ GHz, while the cache capacity ranges from $c_r = 100$ to $c_r = 110$ terabytes (TB).

For a self-driving car $v \in \mathcal{V}$, as shown in Fig 6, we generated randomly features of 37 passengers (F: Female, M: Male, A: Anger, A*: Anticipation, D: Disgust, E: Joy, S*: Sad, S: Surprise, T: Trust). However, in a realistic implementation, for getting passengers' features, the CNN model described in Section III-A.2 should be used. For emotion-based clustering, we use 8 emotion-based clusters: anger, anticipation, disgust, fear, joy, sad, surprise, and trust as the labels. Furthermore, for age-based clustering, we use 8 age-based clusters: $[0 \rightarrow 9, 10 \rightarrow 19, 20 \rightarrow 29, 30 \rightarrow 39, 40 \rightarrow 49, 50 \rightarrow 59, 60 \rightarrow 69, 70 \rightarrow 79]$ as the labels. We generated randomly demands for contents and the popularity of the

contents follows Zipf distribution described in [50], [51]. Furthermore, the self-driving car has a WiFi bandwidth of 160 MHz (802.11ac) with a maximum theoretical data rate of $\tilde{\psi}_u^v = 3466.8$ Mbps. In addition, the computation capacity of the self-driving car is set to $p_v = 3.6$ GHz, while the cache capacity is set to $c_v = 100$ TB.

B. Evaluation Results

Based on video ratings and users' location information, we select six areas to deploy RSUs by using the k-means algorithm. In the selected six areas, we predict the probabilities of contents to be requested in these areas by using MLP. As shown in Fig. 8, in MLP, we minimize the cross-entropy loss function. An accuracy of 97.82% is achieved for predicting the probabilities of contents to be requested in 6 areas of RSUs. Each RSU $v \in \mathcal{V}$ caches movies by starting with the movies that have high ratings and predicted probabilities to be requested within the RSU area (in descending order) until the cache storage becomes full or there are no more movies to cache. As an example, Fig. 8 shows the top 8 movies that need to be cached at RSU 1 with their predicted probabilities using MLP.

Caching at the RSUs is based on location and movie ratings. However, in addition to location and movie ratings, caching in self-driving cars is based on passengers' features such as age, emotion, and gender. Therefore, when the self-driving
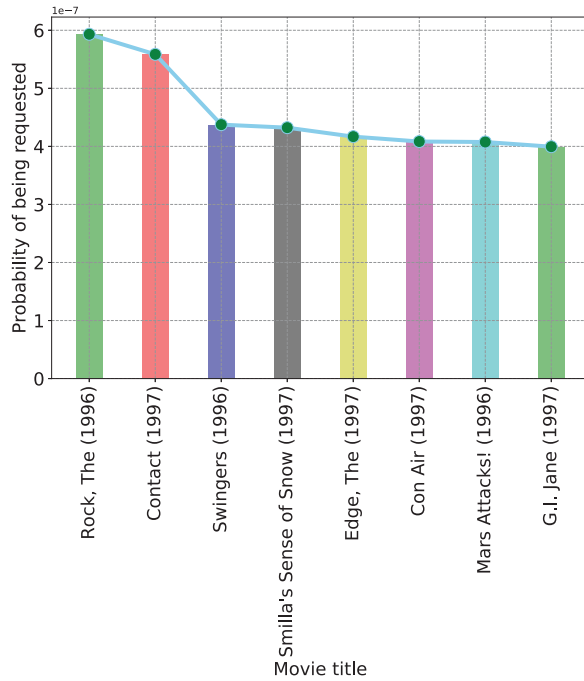
Fig. 8. Some high recommended movies to cache in close proximity of the self-driving cars at RSUs.
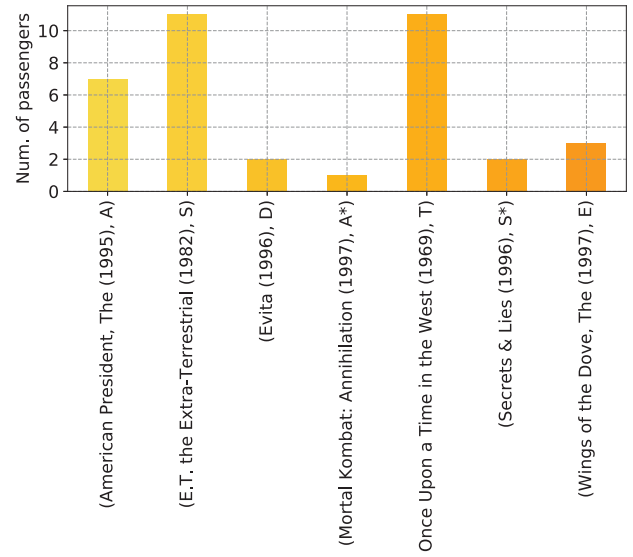

Fig. 9. Some high recommended movies to watch based on passengers' features (age, gender, and emotion).

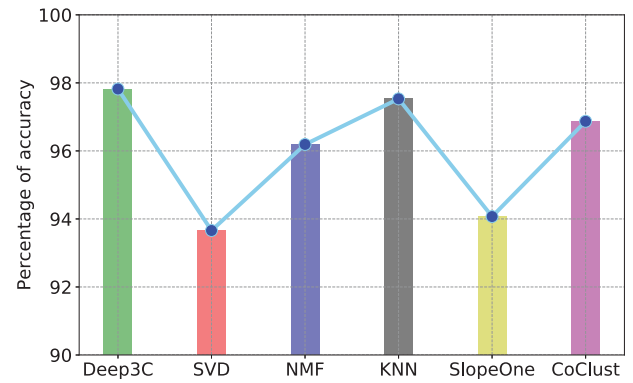Fig. 10. Comparison of various collaborative filtering algorithms and our proposal (Deep3C).

car is connected to an RSU, it downloads the MLP output from the RSU. Then, it groups the MLP output based on age and emotion using the k-means algorithm and on gender using binary classification described in Section III-A.3. Here, we use 8 age-based clusters, 8 emotion-based clusters, and 2 gender-based clusters. Furthermore, for the passengers, we use age, emotion, and gender features described in Fig. 6. However, CNN can be used to predict these features (age, emotion, and gender) using facial images of passengers captured by car's camera. The self-driving car uses k-means and binary classification to classify the passengers in different age, emotion, and gender-based clusters formed using MLP output. Then, inside the formed clusters, the self-driving car finds the movies that have high ratings and predicted probabilities to be requested as recommended movies for the passengers.

Fig. 9 shows recommended movies to watch depending on age, emotion, and gender of the passengers. As shown in this figure, based on these features, passengers may like similar movies (many passengers like Once Upon A Time and Secrets & Lies). Therefore, caching these recommended movies inside the car can prevent repetitive demands of the same movies that need to be sent to RSUs or DC. In other words, we can save bandwidth. Furthermore, we chose CNN and MLP-based recommendation for movies over collaborative filtering approaches because each passenger's features for infotainment contents are not a priori known by the self-driving car. The collaborative filtering approaches, which are described in [52], consist of establishing the relationship between prior known users' preferences and movies' features. However, after identifying passengers' features and movies' features, we compare our proposal denoted Deep3C with the well-known collaborative filtering approaches such as Singular

Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), K-Nearest Neighbors (KNN), and Co-clustering (Coclust). The simulation results in Fig. 10 show that our proposal (Deep3C) achieves better performance over existing collaborative filtering approaches.

We generated randomly demands of passengers for contents, where the popularity of the contents follows Zipf distribution [50]. We use Zipf parameter a with values from $a = 0.5$ to $a = 2.0$. The choice of a = 0.5 to a = 2.0 comes from the results presented in Fig. 11, where the difference in convergence is observed within a range of $a = 0.5$ to $a = 2.0$. Furthermore, based on the demands of the passengers, Fig. 12 shows the normalized cache hits for the cached movies. The movies that are not cached in the self-driving car (cache misses) need to be retrieved at the RSU or DC. In this figure, we present the cache hits for the contents cached at RSUs and self-driving car. In other words, the total cache hits at RSUs and the self-driving car equal to 61% of the whole demands, i.e., 39% of the demands need to be served by DC. Therefore, with edge caching at RSUs and self-driving cars,
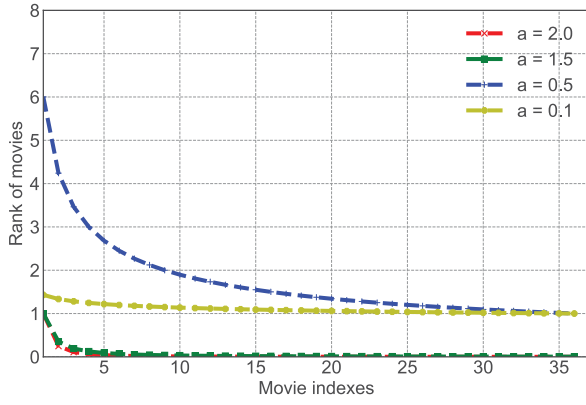
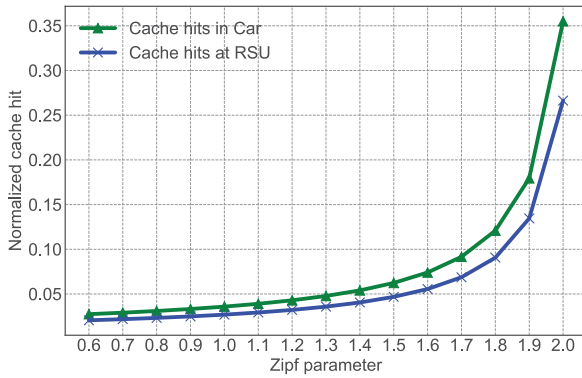Fig. 11. Ranking of movie demands based on Zipf distribution.

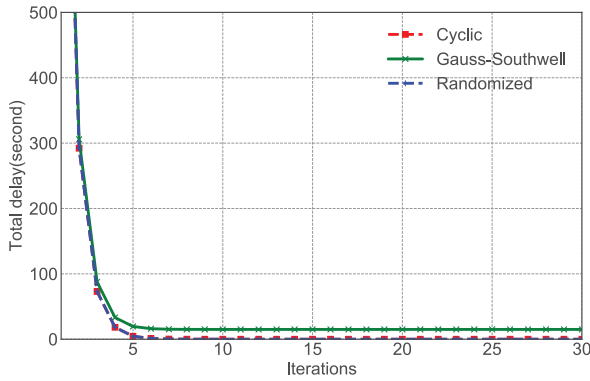Fig. 12. Cache hits for the requested movies.

Fig. 13. The solution of total delay minimization problem (40).

Fig. 14. Computation throughput for the cache contents.

we can significantly save backhaul bandwidth. The results in this figure demonstrate that the cache hits increase with Zipf parameter, i.e., when $a = 2.0$ the small number of movies are very popular and requested by many passengers. In other words, the movies with high demands are characterized by high probabilities of being requested and caching these movies contribute to the high increase of cache hits.

Fig. 13 shows the solution of the surrogate function (40), where (40) minimizes the total delays (transmission delay and computation delay). The surrogate function (40) converges to a coordinate-wise minimum point which is the stationary point through the use of different selection rules such as Cyclic,

Gauss-Southwell, and Randomized. In other words, at a stationary point, the problem (40) cannot find a better minimum direction. Furthermore, in this figure, the self-driving car needs to download the recommended contents first, and then caches these recommended contents; this contributes to high latency at the first iterations. As described in Fig 9, some passengers may need to watch similar movies, i.e., many requests for movies can be satisfied from the cache storage.

In Fig. 14, we present the Cumulative Distribution Function (CDF) of computational throughput in terms of the number of Instruction Per Second (IPS). Here, we define computation throughput as a measurement of how many units of tasks that can be computed by OBU for a given time. In this figure, the simulation results demonstrate that the Cyclic selection rule uses higher computational resource than Gauss-Southwell and Randomized selection rules. Cyclic selection rule has to choose index $j \in \mathcal{J}^t$ cyclically until all indexes in $\mathcal{J}^t$ are used.

VI. CONCLUSION

In this paper, we proposed a novel framework that uses deep learning for content caching in a self-driving car. In the proposed framework, at the DC, we proposed an MLP to predict the probabilities of contents being requested in specific areas. Then, the output is deployed in MEC servers (at the RSUs) close to the self-driving cars, where each MEC server downloads and caches the contents that have high probabilities of being requested in its coverage area. Furthermore, for a self-driving car, to cache infotainment contents that are appropriate regarding the age, emotion, and gender of the passengers, we proposed to use CNN approach for predicting the age, emotion, and gender. Then, the self-driving car downloads the MLP output from the MEC server and combines CNN output with the MLP output using k-means and binary classifications to identify the infotainment contents that meet passengers' features to be downloaded and cached. Therefore, we formulated the deep learning-based caching problem as an optimization problem that minimizes the content-downloading delay. The simulation results demonstrate that our caching approach can reduce 61% of the backhaul traffic, i.e., caching at RSUs and self-driving cars can serve 61% of the whole demands for infotainment contents. Furthermore, our prediction for the

infotainment contents that need to be cached at the RSUs and the self-driving cars reaches 97.82% accuracy.

REFERENCES

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

[2] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 62–70, Mar. 2019.

[3] F. Sullivan. (Mar. 2018). *Global Autonomous Driving Market Outlook, 2018.* Accessed: Jun. 22, 2019. [Online]. Available: https://info. microsoft.com/rs/157-GQE-382/images/K24A-2018%20Frost %20%26%20Sullivan%20-%20Global%20Autonomous %20Driving%20Outlook.pdf

[4] G. Jarvis, "Keeping entertained in the autonomous vehicle," TU-Automotive Detroit, Detroit, MI, USA, Tech. Rep. TU Auto-[Tele.Jarvis.2017.12.08], Jun. 2018. Accessed: Jun. 22, 2019. [Online]. Available: http://pages.tu-auto.com/Detroit18-AVReport

[5] F. Fathi, N. Abghour, and M. Ouzzif, "From big data to better behavior in self-driving cars," in *Proc. 2nd Int. Conf. Cloud Big Data Comput. (ICCBDC)*, 2018, pp. 42–46.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, Sep. 2015.

[7] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, to be published.

[8] Next Analytics. *YouTube Video Appeal Demographics.* Accessed: Jun. 22, 2019. [Online]. Available: https://www.nextanalytics. com/excel-youtube-analytic-insights-and-data-mining/page/4/

[9] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in *Proc. IEEE 19th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Seoul, South Korea, Sep. 2017, pp. 366–369.

[10] L. Divine, J. Kurihara, and D. Kryze, "Auto-control of vehicle infotainment system based on extracted characteristics of car occupants," U.S. Patent 13 192 629, Jan. 31, 2013.

[11] J. Ma, J. Wang, G. Liu, and P. Fan, "Low latency caching placement policy for cloud-based VANET with both vehicle caches and RSU caches," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Singapore, Dec. 2017, pp. 1–6.

[12] S. M. A. Kazmi *et al.*, "Infotainment enabled smart cars: A joint communication, caching, and computation approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8408–8420, Sep. 2019.

[13] V. S. Varanasi and S. Chilukuri, "Adaptive differentiated edge caching with machine learning for V2X communication," in *Proc. 11th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2019, pp. 481–484.

[14] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, Jan. 2018.

[15] I. Raichelgauz, K. Odinaev, and Y. Y. Zeevi, "System and method for caching concept structures in autonomous vehicles," U.S. Patent 15 677 496, Mar. 1 2018.

[16] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 124–135, Feb. 2019.

[17] R. F. Atallah, C. M. Assi, and M. J. Khabbaz, "Scheduling the operation of a connected vehicular network using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1669–1682, May 2018.

[18] Z. Ning, P. Dong, J. J. P. C. Rodrigues, F. Xia, and X. Wang, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, Oct. 2019.

[19] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.

[20] Z. Ning *et al.*, "When deep reinforcement learning meets 5G vehicular networks: A distributed offloading framework for traffic big data," *IEEE Trans. Ind. Informat.*, to be published.

[21] Z. Ning *et al.*, "Deep learning in edge of vehicles: Exploring trirelationship for data transmission," *IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5737–5746, Oct. 2019.

[22] A. Ndikumana and C. S. Hong, "Self-driving car meets multi-access edge computing for deep learning-based caching," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Kuala Lumpur, Malaysia, Jan. 2019, pp. 9–11.

[23] J. J. Whang, I. S. Dhillon, and D. F. Gleich, "Non-exhaustive, overlapping *k*-means," in *Proc. SIAM Int. Conf. Data Mining*, Victoria, BC, Canada, Apr./May 2015, pp. 936–944.

[24] J. Martineau, T. Finin, A. Joshi, and S. Patel, "Improving binary classification on text problems using differential word features," in *Proc. Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, Hong Kong, 2009, pp. 2019–2024.

[25] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, Feb. 2017.

[26] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. S. Shen, "Cost-effective vehicular network planning with cache-enabled green roadside units," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[27] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6321–6334, Oct. 2017.

[28] F. Chen *et al.*, "Distribution-aware cache replication for cooperative road side units in VANETs," *Peer-Peer Netw. Appl.*, vol. 11, no. 5, pp. 1075–1084, Jul. 2017.

[29] A. Ndikumana, N. H. Tran, T. M. Ho, D. Niyato, Z. Han, and C. S. Hong, "Joint incentive mechanism for paid content caching and price based cache replacement policy in named data networking," *IEEE Access*, vol. 6, pp. 33702–33717, 2018.

[30] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 28–35, Jun. 2018.

[31] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[32] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service Requirements for V2X Services; Stage 1(Release 15)*, document 3GPP TS 22.186 version 15.3.0 Release 15, Jun. 2018.

[33] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 3358–3363.

[34] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," in *Proc. NOMS - IEEE/IFIP Netw. Oper. Manage. Symp.*, Taipei, Taiwan, Apr. 2018, pp. 1–5.

[35] M. Zahangir Alom *et al.*, "The history began from AlexNet: A comprehensive survey on deep learning approaches," Mar. 2018, *arXiv:1803.01164*. [Online]. Available: http://arxiv.org/abs/1803.01164

[36] A. Dehghan, E. G. Ortiz, G. Shu, and S. Zain Masood, "DAGER: Deep age, gender and emotion recognition using convolutional neural network," Mar. 2017, *arXiv:1702.04280*. [Online]. Available: http://arxiv.org/abs/1702.04280

[37] L. van Zoonen, "Privacy concerns in smart cities," *Government Inf. Quart.*, vol. 33, no. 3, pp. 472–480, Jul. 2016.

[38] *3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Access Network Discovery and Selection Function (ANDSF) Management Object (MO) (Release 15)*, document 3GPP TS 24.312 V15.0.0, Jun. 2018.

[39] E. Ndashimye, N. I. Sarkar, and S. K. Ray, "A novel network selection mechanism for Vehicle-to-Infrastructure communication," in *Proc. IEEE 14th Int. Conf. Dependable, Autonomic Secure Comput., 14th Int. Conf. Pervasive Intell. Comput., 2nd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Auckland, New Zealand, Aug. 2016, pp. 483–488.

[40] N. Cheng, N. Lu, N. Zhang, X. Zhang, X. S. Shen, and J. W. Mark, "Opportunistic WiFi offloading in vehicular environment: A game-theory approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1944–1955, Jul. 2016.

[41] S. Mosleh, L. Liu, and J. Zhang, "Proportional-fair resource allocation for coordinated multi-point transmission in LTE-advanced," *IEEE Trans. Wireless Commun.*, vol. 15, no. 8, pp. 5355–5367, Aug. 2016.

[42] A. Ndikumana, N. H. Tran, and C. Seon Hong, "Deep learning based caching for self-driving car in multi-access edge computing," Oct. 2018, *arXiv:1810.01548*. [Online]. Available: http://arxiv.org/abs/1810.01548

[43] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, "Iteration complexity analysis of block coordinate descent methods," *Math. Program.*, vol. 163, nos. 1–2, pp. 85–114, Aug. 2017.

[44] U. Feige, M. Feldman, and I. Talgam-Cohen, "Oblivious rounding and the integrality gap," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)* (Leibniz International Proceedings in Informatics (LIPIcs), vol. 60, K. Jansen, C. Mathieu, J. D. P. Rolim, and C. Umans, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 8-1–8-23. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6631

[45] N. Zhang, Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong, and Z.-Q. Luo, "Network slicing for service-oriented networks under resource constraints," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2512–2521, Nov. 2017.

[46] Google. *Python Client Library for Google Maps Api Web Services.* Accessed: Aug. 12, 2018. [Online]. Available: https://github.com/googlemaps/google-maps-services-python

[47] Keras. *Keras: The Python Deep Learning Library.* Accessed: Jun. 22, 2019. [Online]. Available: https://keras.io/

[48] W. McKinney, "Pandas: A foundational python library for data analysis and statistics," *Python High Perform. Sci. Comput.*, vol. 14, no. 9, pp. 1–9, 2011.

[49] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, 2016.

[50] M. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemp. Phys.*, vol. 46, no. 5, pp. 323–351, Sep. 2007.

[51] A. Ndikumana *et al.*, "In-network caching for paid contents in content centric networking," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–6.

[52] V. Vijayakumar, A. Challa, M. Chandrashekhar, V. Subramaniyaswamy, and R. Logesh, "A personalised movie recommendation system based on collaborative filtering," *Int. J. High Perform. Comput. Netw.*, vol. 10, nos. 1–2, pp. 54–63, 2017.
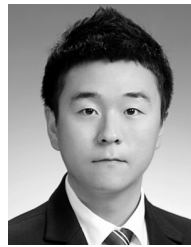
Do Hyeon Kim received the B.S. degree in communication engineering from Jeju National University in 2014 and the M.S. degree from Kyung Hee University in 2017, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include multiaccess edge computing and wireless network virtualization.

Ki Tae Kim received the B.S. and M.S. degrees in computer science and engineering from Kyung Hee University, Seoul, South Korea, in 2017 and 2018, respectively, where he is currently pursuing the Ph.D. degree in computer science and engineering. His research interests include SDN/NFV, wireless networks, unmanned aerial vehicle communications, and machine learning.

Anselme Ndikumana received the B.S. degree in computer science from the National University of Rwanda in 2007 and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in August 2019. His professional experience includes a Chief Information officer, a System Analyst, and a Database Administrator at the Rwanda Utilities Regulatory Authority from 2008 to 2014. Since September 2019, he has been with the Faculty of Computing and Information Sciences, University of Lay Adventists of Kigali, Rwanda, where he is currently a Lecturer. His research interests include deep learning, multiaccess edge computing, information-centric networking, and in-network caching.

Nguyen H. Tran (Senior Member, IEEE) received the B.S. degree in electrical engineering from the Hochiminh City University of Technology in 2005 and the Ph.D. degree in computer engineering from Kyung Hee University in 2011. He was an Assistant Professor with the Department of Computer Science and Engineering, Kyung Hee University, from 2012 to 2017. Since 2018, he has been with the School of Computer Science, The University of Sydney, where he is currently a Senior Lecturer. He has received the Korea NRF Funding for Basic Science and Research for the term of 2016–2023. His research interests are in distributed computing and learning over networks. He received the Best KHU Thesis Award in engineering in 2011 and several best paper awards, including the IEEE ICC 2016, APNOMS 2016, and IEEE ICCS 2016. He has been an Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING since 2016.

Choong Seon Hong (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree from Keio University, Japan, in 1997. In 1988, he joined KT, where he was involved in broadband networks, as a Member of Technical Staff. In 1993, he joined Keio University. He was with the Telecommunications Network Laboratory, KT, as a Senior Member of Technical Staff and as the Director of the Networking Research Team until 1999. Since 1999, he has been a Professor with the Department of Computer Engineering, Kyung Hee University. His research interests include future Internet, ad hoc networks, network management, and network security. He is currently a member of ACM, IEICE, IPSJ, KIISE, KICS, KIPS, and OSIA. He has served as the General Chair, the TPC Chair/Member, or an Organizing Committee Member for international conferences, such as NOMS, IM, APNOMS, E2EMON, CCNC, ADSN, ICPP, DIM, WISA, BcN, TINA, SAINT, and ICOIN. He is also an Associate Editor of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the *International Journal of Network Management*, and the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS, and an Associate Technical Editor of the *IEEE Communications Magazine*.