Deep Learning Based Caching for Self-Driving Car in Multi-access Edge Computing

Anselme Ndikumana, Nguyen H. Tran, *Member, IEEE,* and Choong Seon Hong, *Senior Member, IEEE*

***Abstract*—Once self-driving car becomes a reality and passengers are no longer worry about it, they will need to find new ways of entertainment. However, retrieving entertainment contents at the Data Center (DC) can hinder content delivery service due to high delay of car-to-DC communication. To address these challenges, we propose a deep learning based caching for self-driving car, by using Deep Learning approaches deployed on the Multi-access Edge Computing (MEC) structure. First, at DC, Multi-Layer Perceptron (MLP) is used to predict the probabilities of contents to be requested in specific areas. To reduce the car-DC delay, MLP outputs are logged into MEC servers attached to roadside units. Second, in order to cache entertainment contents stylized for car passengers' features such as age and gender, Convolutional Neural Network (CNN) is used to predict age and gender of passengers. Third, each car requests MLP output from MEC server, and compares its CNN and MLP outputs by using k-means and binary classification. Through this, the self-driving car can identify the contents need to be downloaded from the MEC server and cached. Finally, we formulate deep learning based caching in the self-driving car that enhance entertainment services as an optimization problem whose goal is to minimize content downloading delay. To solve the formulated problem, a Block Successive Majorization-Minimization (BS-MM) technique is applied. The simulation results show that the accuracy of our prediction for the contents need to be cached in the areas of the self-driving car is achieved at** 98.04% **and our approach can minimize delay.**

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

I. Introduction

A. Background and Motivations

According to road traffic accident statistics, in 2016, 235,532 people were killed, and 6,648,078 people were injured in traffic accidents [1]. Furthermore, US National Highway Traffic Safety Administration (NHTSA) data in 2015 shows that, in Georgia, 94% of car accidents caused by human errors and bad decisions [2]. Therefore, in order to save lives, prevent human errors and bad decisions, and releasing human from stressful tasks of controlling car, many research projects for the autonomous car have been introduced [3].

Anselme Ndikumana and Choong Seon Hong are with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Rep. of Korea, E-mail:{anselme, cshong}@khu.ac.kr

Nguyen H. Tran is with the School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia, E-mail: {nguyen.tran}@sydney.edu.au

Recently, the automobile industries have made remarkable improvements by creating autonomous cars that can drive themselves with human driver intervention. Some companies, such as Google, Uber, Samsung, Tesla, Mercedes-Benz, Baidu, etc., have already started to focus on the next stage of autonomous driving called "self-driving", where cars can drive themselves without human driver intervention [4]. Therefore, self-driving cars will have full driving automation in all situations. Furthermore, in order to make the self-driving car more intelligent, the self-driving car needs to be equipped with smart sensors and analytics tools that collect and analyze heterogeneous data related to people on-board, pedestrian, and environment in real time, in which deep learning plays significant roles [5].

Even though a self-driving car has On-Board Unit (OBU) that can handle Computation, Communication, Caching, and Control (4C), we still consider the self-driving car's resources for 4C to be limited, and requires assistance from the remote clouds [6]. For effective self-driving car's data analytics, there is a strong need for low-latency and reliable computations. However, reliance on a cloud can hinder the performance of the self-driving car's data analytics, due to the associated end-to-end delay. Therefore, to reduce end-to-end delay, we consider Multi-access Edge Computing (MEC) [7] as a suitable technology that can support self-driving cars for edge analytics [8]. MEC has been recently introduced by the European Telecommunications Standards Institute (ETSI) to supplement cloud computing, where MEC servers are deployed at the edge of the network for 4C [9]. In this work, MEC servers are deployed at RoadSide Units (RSUs) for edge analytics and content caching in close proximity to the self-driving cars.

We focus on self-driving cars for public transport such as buses, because in the future, self-driving buses are expected to roll down in smart cities [10]. Furthermore, with deep learning and 4C features in the self-driving car, passengers will no longer be limited to onboard radio and TV, instead spending more time on watching media, playing games, and social networks. However, retrieving these contents from Data Centers (DCs) can make content delivery service worse, due to the associated end-to-end delay, and consumed backhaul bandwidth resources. As an example, watching a video in a car requires three components, namely video source, screen, and a 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 in-car services will incur high

delay, caching in self driving car will play an important role to enhance the users' experience.

B. Challenges for Caching in Self-Driving Car

- For traveling people, the self-driving car will be a new entertainment place. Therefore, content providers and game developers need to grasp this new opportunity by providing high-quality of entertainment contents. However, there is still lack of literature on how caching for entertainment contents in self-driving can be performed.
- The self-driving car could deliver more heterogeneous entertainment contents such as movies, TV, music, and games as well recent emerging platforms such as Virtual Reality (VR) [11]. However, the self-driving car's resources for 4C are limited. Therefore, self-driving cars need to be supported by MEC servers.
- The self-driving car is sensitive to delay. Therefore, to reduce car-DC delay and save backhaul bandwidth, communication and caching resources utilization in MEC servers and self-driving cars needs to be reinforced and optimized.

As related works, content caching at BSs, and RSUs has gained significant attention in [9], [12]–[15]. In addition, in [16], [17], the authors proposed deep learning approaches for edge caching (at BS, RSUs, and user equipment). Still, in these works, content caching in the self-driving car was not addressed thoroughly. In [18], the authors proposed the cloud-based vehicular ad-hoc network, where both vehicles and RSUs participate in content caching. However, introducing the cloud-based controller in vehicle caching can increase content retrieval delay. Other alternatives have also been proposed in [19], where the authors considered two levels of caching at edge servers (BSs) and at autonomous cars. In this proposal, the edge server injects contents to some selected cars that have enough influences to share these contents with other cars. However, in a realistic network environment, BSs and cars may belong to the different entities. Therefore, without incentive mechanism, there are no motivations for car owners to allow BS operator(s) to inject the contents in their cars and participate in content sharing. Finally, in [20], self-driving car caching forum was introduced by GEOCACHING in March 2018, but still, there is no proposal on how caching in self-driving car can be implemented.

C. Contributions

In order to address the aforementioned challenges, we propose improving entertainment services in self-driving cars using deep learning based caching and 4C approaches in MEC. The main contributions of this paper are summarized as follows:

- People have different content flavor, in which their choices depend on ages and genders [21]. To fulfill the demands of passengers in self-driving car, we use a Convolutional Neural Network (CNN) approach to predict their ages and genders via facial recognition. Specifically, CNN outputs are used by self-driving car for the purpose of deciding on which entertainment contents, such as music, video, and game data, are appropriate for passengers and thus need to be cached.
- To get the appropriate entertainment content for passengers, the self-driving car needs to be supported by MEC and DC. At DC, we propose a Multi-Layer Perceptron (MLP) framework to predict the probability of content to be requested in a specific area of self-driving car. Then, the MLP prediction outputs are deployed at MEC servers (at RSUs) in close proximity to the self-driving car. During off-peak hours, each MEC server uses MLP outputs to download and then cache the contents that have high probabilities for being requested. We choose MLP over other prediction methods such as AutoRegressive (AR) and the AutoRegressive Moving Average (ARMA) models, because MLP has the capability to cope with both linear and non-linearly prediction problems [22].
- For the contents need to be cached, the self-driving car downloads MLP outputs from MEC server, which is then compared with the CNN outputs. For the comparison, we combine k-means and binary classification. We choose k-means and binary classification over other clustering algorithms due to their computational efficiencies and elegant simplicities in their implementation [23], [24].
- We formulate caching in a self-driving car for entertainment services using deep learning exploiting 4C components in MEC to minimize content-downloading delay. To solve the formulated problem, we use Block Successive Majorization-Minimization (BS-MM) technique [25]. We choose BS-MM over other optimization techniques because BS-MM is a new technique that allows decomposing problem into small subproblems by partitioning the formulated problem into blocks.

Specifically, the novelties of our proposal over related works in [9], [12]–[20] are: to the best of our knowledge, we are the first to investigate self-driving car caching for entertainment contents, where caching decisions are based on MLP, CNN, and available communication, caching, and computation resources.

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

II. System model

The system model is depicted in Fig. 1:

- *At Data Center (DC):* Typically, DC hosts dataset from data market for prediction purpose. In the DC, we use MLP described in section III-A1 for predicting the probabilities of contents to be requested in specific areas. In order to reduce communication delay between the self-driving car and DC, the outputs of the MLP

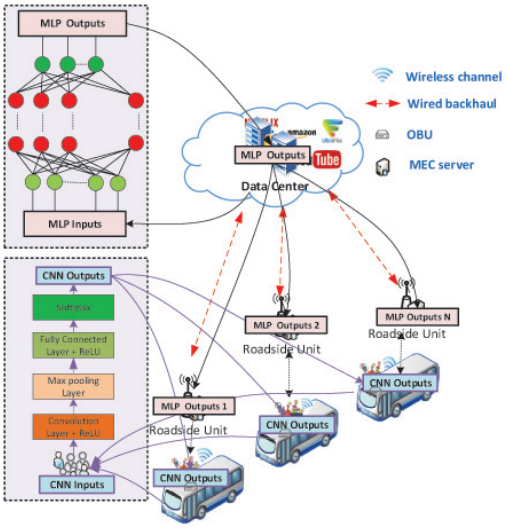

Figure 1: Illustration of our system model.

Table I: Summary of key notations.

Notation	Definition
$\mathcal{V}$	Set of self-driving car, $\lvert\mathcal{V}\rvert = V$
$\mathcal{I}$	Set of contents, $\lvert\mathcal{I}\rvert = I$
$\mathcal{I}_r(n)$	Set of contents need to be cached in area n of RSU r, $\lvert\mathcal{I}_r(n)\rvert = I_r(n)$
$\mathcal{U}$	Set of consumers of contents, $\lvert\mathcal{U}\rvert = U$
$\mathcal{R}$	Set of RSUs, $\lvert\mathcal{R}\rvert = R$
$\boldsymbol{x}$	Input of MLP
$\tilde{\boldsymbol{y}}$	Output of MLP
$\boldsymbol{y}$	Ground truth for MLP
M	The number of inputs 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 $v \in \mathcal{V}$
k_0^v	Input image of passenger in car $v \in \mathcal{V}$
$\tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$	Total delay experienced by each passenger $u \in \mathcal{U}_v$
R_u^v	Data rate for each passenger u via IWR of car v

are deployed at MEC servers attached to RSUs based on their locations. We use $\mathcal{N} = \{1, 2, \ldots, N\}$ to denote the set of geographic locations, where each location $n \in \mathcal{N}$ has one MEC server.

- *At RoadSide Unit (RSU):* We consider that each RSU $r \in \mathcal{R}$ has access to DC via a wired backhaul of capacity $\omega_{r,DC}$, where $\mathcal{R}$ is the set of R RSUs. Furthermore, we consider $\mathcal{V}$ as a set of V self-driving cars, in which each RSU $r \in \mathcal{R}$ can provide broadband Internet service to V self-driving cars via wireless link of capacity $\omega_{v,r}$ at each time slot. In addition, each RSU has one MEC server. Unless stated otherwise, we use also the terms "RSU", and "MEC server" interchangeably. Furthermore, each MEC server $r \in \mathcal{R}$ has cache storage of capacity c_r and computational resource of capacity p_r.
  During off-peak hours, by using backhaul communication resources, based on the MLP outputs, each MEC server can download and cache predicted contents with high probabilities of being requested in its region. We use $\mathcal{I}$ to denote a set of I contents, where each content $i \in \mathcal{I}$ has a size of $S(i)$ Mb. In addition, based on demands for content, each cached content can be served as it is or after being computed. Therefore, we use i to denote the content before computation, and i' to denote the content after computation. As an example, content i' with format .avi may be not available in the cache storage of MEC server. Instead, the cache storage may have content i with format .mpeg of the same content. Therefore, for satisfying the demand, by using the computational resource, MEC server can convert i to i'.
- *At self-driving car:* We consider that the self-driving car has an OBU that can handle 4C with MEC to support caching of entertainment contents for passengers. Each self-driving car $v \in \mathcal{V}$ has cache storage of capacity c_v and computation capability p_v. Furthermore, to decide which entertainment content to request and cache in the self-driving car, we use CNN approach presented in III-A2 to predict age and gender of car passengers, where each self-driving car is equipped with a camera system for capturing incoming passenger. After CNN prediction, the self-driving car can request its nearest RSUs the MLP prediction. Then, by using k-means and binary classification presented in III-A3, self-driving car compares its CNN prediction with the predicted outputs from MLP. This helps to identify the entertainment contents which are appropriate to the passengers. Finally, the self-driving car downloads and caches the identified contents.

III. Deep Learning Based Caching in Self-Driving Car

As described in the previous section, for caching contents, the self-driving car needs to compare its CNN prediction with MLP prediction. Here, we discuss deep learning and recommendation model in Section III-A, where the output of the recommendation model is the contents that should to be cached in the self-driving car. Furthermore, for requesting and downloading the recommended contents, the self-driving car requires communication resources. Therefore, in Section III-B, we will discuss the communication model. For caching downloaded contents, we present the caching model in Section III-C. Based on demands, cached contents can be converted or transcoded to the different formats by using computational resources, where computation model is described in Section III-D.

A. Deep Learning and Recommendation Model

1) Multi-Layer Perceptron (MLP): As depicted in Fig. 1, at DC, we use MLP, where green circles represent neurons of input and output layers, while red circles represent neurons of hidden layers.

For predicting the probabilities of contents to be requested in specific areas, we use a demographical dataset that will be described in Section V. The inputs and outputs are described as follows:

- *Inputs:* In the dataset, we have entertainment content names, rating, viewer's age, gender, and locations as the inputs of MLP. We use $\boldsymbol{x} = (x_1, x_2, \dots x_M)^T$ to denote the input vector, where the subscripts are used to denote the features.
- *Outputs:* From the inputs, MLP tries to predict $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \tilde{y}_2, \dots \tilde{y}_N)^T$ as the output vector and the subscripts are used to denote the geographical areas. In addition, in the output layer, each area $n \in \mathcal{N}$ corresponds to one neuron and predicts the probabilities of contents require to be cached in that area. Furthermore, for predicting the ranking of entertainment contents over time, long short-term memory (LSTM) described in [26] can be used. However, we consider LSTM for predicting content rating to be outside of the scope of this paper.

Before presenting the MLP, let us start with a simple artificial neural network (ANN) of one layer, where we consider the outputs as the weighted sum of the inputs. We use w_{nm} to denote weight from input x_m to output $\tilde{y}_n$. Therefore, the output y_n can be expressed as follows:

$$\tilde{y}_n = f\left(\sum\nolimits_{m=1}^{M} w_{nm} x_m + b_n\right), \tag{1}$$

where $f(.)$ is the activation functiona and b_n is the bias added with a linear combiner ($\sum_{m=1}^{M} w_{nm} x_m$).

As an extension to the above simple ANN, we consider MLP as an ANN which has more hidden layers, where each hidden layers has more units called neurons. For MLP, we use l to denote the the number of hidden layers, $\boldsymbol{x}$ for input vector, $\boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(l)}$ for bias vectors, $\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(l)}$ for the weight matrices at each hidden layer, and $\tilde{\boldsymbol{y}}$ for output vector. $\tilde{\boldsymbol{y}}$ can be expressed as follows:

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

The above equation (2) shows that each neuron received the output from the previous layer and after processing, it sends output to the next neurons of the next layer. In our MLP, we use Rectified Linear Unit (ReLU) as the activation function in all the layers except at the output layer. The ReLU can be mathematically expressed as follows:

$$\tilde{y}_m = \max(0, x_m), \forall m. \tag{3}$$

We choose ReLU over other activation function, because it solves the vanishing gradient problem experienced by MLP during the training process [27]. Furthermore, in the output layer j, we use softmax function as an activation function. The purpose of softmax function is to squeeze the output vector $\tilde{\boldsymbol{y}}$ into a set of probability values, where softmax function is defined as:

$$softmax(\tilde{\boldsymbol{y}})^{(l)} = \frac{e^{\tilde{y}_l}}{\sum_{n=1}^{N} e^{\tilde{y}_n}}, \text{ for } l = 1, \dots, N. \tag{4}$$

The output layer has N neurons that correspond to the number of geographical locations, where the cache-enable RSU will be used for caching the contents.

The aims of our MLP is to compute the output $\tilde{\boldsymbol{y}}$ for each input $\boldsymbol{x}$. Therefore, during the training of our MLP, we need to adjust our $\boldsymbol{w}$ such that the correct output $\tilde{\boldsymbol{y}}$ can be obtained for each input $\boldsymbol{x}$. In other words, we need to adjust the network weights $\boldsymbol{w}$ such that the error function is minimized. As described in [27], weight adjustment can be done through a series of gradient descent weight updates in a backward manner from the output layer, and this technique is called backpropagation.

For error function, we choose cross entropy error function overs other error functions as our MLP aims to classify the contents needs to be cached in N geographical areas. This problem can be considered as classification problem, where we interpret the outputs as probabilities of the contents to be cached in specific geographical area. The cross entropy error function $A(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ can be expressed as follows:

$$A(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = -\sum\nolimits_{n=1}^{N} y_n \log \tilde{y}_n. \tag{5}$$

The above cross entropy error function $A(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ penalizes large deviations from the desired caching locations. Technically, $A(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ calculates the cross-entropy between the estimated class probabilities $\tilde{\boldsymbol{y}}$ and the ground truth $\boldsymbol{y}$.

Finally, in order to reduce communication delay between the self-driving car and DC, as DC may be located in a far distance from the self-driving car, the outputs of MLP are sent to MEC servers attached to RSUs based on their geographical locations/areas.

2) Convolutional Neural Network (CNN): We use CNN for automatic age and gender extraction from facial images. This problem has been extensively studied in [28]. We consider that features like age and gender will play an important role in entertainment content consumption. Once the facial image of passenger is captured via the camera of self-driving car, we can extract features such as location, size, eyes, nose, mouth, chin, etc., and use them for classifying the face into different age and genders classes by using CNN VGG16 described in [29]. We describe CNN workflow for automatic age and gender extraction as follows:

- *Inputs:* In the self-driving car, we consider $\boldsymbol{k}_0^v$ as the input image of incoming passenger(s) with three dimensions space: height, width, and the number of color channels (red, green, and blue).
- *Convolution layer:* Convolution layer applies filters to input regions and computes the output of each neuron. Each neuron is connected to local regions of inputs and by using dot products between weight and local regions, convolution layer comes out with feature map $\boldsymbol{k}_j^v$. We use $\boldsymbol{k}_j^v$ to denote the feature map after convolution layer j in self-driving car $v \in \mathcal{V}$.
- *RELU layer:* In this layer, we apply ReLU $\max(0, \boldsymbol{k}_j^v)$ as an elementwise activation function. The ReLU keeps the size of its associated convolution layer j unchanged.
- *Max pooling layer:* After Convolution and RELU layers, we have a high dimensional matrix. Therefore, for dimension reduction, we apply maxing pooling layer as 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 2 classes for gender (male and female) and 101 classes of age (from 0 to 101). In other words, we use two fully-connected layers, one for age and another one for gender classification.
- *Softmax layer:* In this layer, for each facial image, we need to interpret output as probability values that indicate the classes for gender and age that a face could potentially belong to. To achieve this, the softmax activation function is applied to the outputs of the fully-connected layers.

3) Recommendation Model: Once the self-driving car is connected to RSU, it downloads MLP output. Then, it uses the CNN prediction and MLP output to decide the contents that need to be downloaded and cached in its cache storage. The workflow of the recommendation model for the self-driving car is described as follows:

- *Step 1:* Each self-driving car $v \in \mathcal{V}$ downloads MLP output from MEC server (attached to RSU).
- *Step 2:* By using the k-means algorithm for age-based grouping and binary classification for gender-based grouping on MLP output, each self-driving car $v \in \mathcal{V}$ makes clusters of consumers of contents and generates an initial recommendation for the contents that need to be cached and have high requested probabilities.
- *Step 3:* For each new passenger $u \in \mathcal{U}$, the self-driving car uses CNN for automatically predicting its age and gender. We assume that the self-driving car trains CNN once by using dataset, saves the model, and uses it for predicting age and the gender without always training the model again.
- *Step 4:* The self-driving car uses these passenger's features to calculate the similarity of passenger $u \in \mathcal{U}$ with the existing classified people (consumers of contents). From the result of similarity calculation, the passenger $u \in \mathcal{U}$ will be assigned to a cluster.
- *Step 5:* After clustering of passenger(s), self-driving car $v \in \mathcal{V}$ updates the recommendation for the content that the self-driving car needs to be downloaded and cached.
- *Step 6:* Finally, self-driving car $v \in \mathcal{V}$ downloads the recommended contents via RSUs and caches them in its cache storage c_v.

Let us describe in detail how to use k-means algorithm and binary classification in the recommendation model at each geographical location $n \in \mathcal{N}$ of self-driving car $v \in \mathcal{V}$. For k-means algorithm, first, we use age as numerical data points. We denote $\tilde{\boldsymbol{y}}_n$ as MLP output at each geographical location $n \in \mathcal{N}$ and $\mathcal{X} = \tilde{\boldsymbol{y}}_n$ as inputs of k-means algorithm. The k-means seeks to partition data points $\mathcal{X} = \{x_1, \ldots, x_U\}$ into K clusters $\mathcal{X}_1, \ldots, \mathcal{X}_K$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$. In k-means, people are grouped into clusters based on category of their age, where we choose the number of clusters K equals to the number of age categories. In addition, clusters are disjoint $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$, $i \neq j$. The goal of k-means is to assign data points to the cluster's centroid such that the below objective function is minimized:

$$\min_{\{\mathcal{X}_j\}_{j=1}^{K}} \sum_{j=1}^{K} \sum_{x_u \in \mathcal{X}_j} \|x_u - \tilde{x}_j\|^2, \tag{6}$$

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

$$\tilde{x}_j = \frac{\sum_{x_u \in \mathcal{X}_j} x_u}{|\mathcal{X}_j|}. \tag{7}$$

In addition to the age, people in the same cluster can have different gender. Furthermore, as shown in the processed YouTube demographic dataset from Next Analytics [21], based on gender, people have different choices for contents. Therefore, in each cluster, we need to group data points based on gender. For gender-based grouping, we apply binary classification described in [24] which results in formation of two groups, one group for females (denoted $\mathcal{G}_j^{\text{female}}$) and another group for males (denoted $\mathcal{G}_j^{\text{male}}$), where $\mathcal{X}_j = \mathcal{G}_j^{\text{female}} \cup \mathcal{G}_j^{\text{male}}$ and $\mathcal{G}_j^{\text{female}} \cap \mathcal{G}_j^{\text{male}} = \emptyset$.

Since downloading both MLP outputs and top recommended contents requires communication resources, we propose communication model that is described in below subsection.

B. Communication Model

During off-peak hours, based on the MLP outputs, each MEC server downloads recommended contents by using fiber backhaul link of capacity $\omega_{r,DC}$. The transmission delay for downloading contents from DC to the MEC server r is expressed as:

$$\tau_r^{\text{DC}} = \frac{\sum_{i \in \mathcal{I}_r(n)} q_i^{\text{DC}\to r} S(i)}{\omega_{r,DC}}, \tag{8}$$

where $\mathcal{I}_r(n)$, $n \in \mathcal{N}$, denotes the set of predicted contents via MLP that need to be cached in region n of RSU r and $q_i^{\text{DC}\to r}$ is a decision variable that indicates whether or not MEC server r downloads recommended content i $\in \mathcal{I}_r(n)$ from DC, such that:

$$q_i^{\text{DC}\to r} = \begin{cases} 1, \text{ if MEC server } r \text{ downloads content } i \\ \quad \text{from DC,} \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

In self-driving car, downloading the top recommended contents requires communication resources. We consider each self-driving car $v \in \mathcal{V}$ moves in region covered by macro base stations (BSs) and RSUs. In addition, we assume that in each route of self-driving car v, there can be many RSUs, where some of them are cache-enabled. Therefore, before the self-driving car starts its journey, it can select RSUs that will be used to download the top recommended contents and the speed that will be used for having less variation in time for downloading contents. Therefore, to discover cache-enabled RSUs located in a route of each self-driving car, Access Network Discovery and Selection Function (ANDSF) implemented in cellular

network [30] can be utilized. To get RSU information (their coordinates and coverages), the self-driving car sends a request to ANDSF server via BS. In the request includes self-driving car geographic location, speed, and direction. The ANDSF server's response includes coordinates and coverage radius 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:

$$\tilde{d}_v^r = g_v^r sin\alpha_v^r, \tag{10}$$

where α_v^r is the angle between the trajectory of movement of self-driving car v and the straight line originating from RSU r physical location, and g_v^r is geographical distance between self-driving car v and cache-enabled r. α_v^r and g_v^r can be obtained via Global Positioning System (GPS) [31]. In addition, each self-driving car v computes the following distance d_r^v remaining to reach each area covered by cache-enabled RSU $r \in \mathcal{R}$:

$$d_r^v = g_v^r cos\alpha_v^r. \tag{11}$$

As described in [31], we defined ρ_v^r as a probability that cache-enabled RSU $r \in \mathcal{R}$ will be selected as a source of contents required to be cached in self-driving car v as follows:

$$\rho_v^r = \begin{cases} 1, \text{ if } \tilde{d}_v^r = 0, \\ \frac{\tilde{d}_v^r}{\gamma_r} \text{ if } 0 < \tilde{d}_v^r < \gamma_r, \\ 0, \text{ otherwise}, \end{cases} \tag{12}$$

where γ_r is the radius of area covered by RSU $r \in \mathcal{R}$. The equation (12) ensures that once the self-driving car v reaches in area covered by cache-enabled RSU $r \in \mathcal{R}$, it immediately starts downloading recommended contents. Therefore, we define q_v^r as decision variable that indicates whether or not self-driving car is connected to RSU $r \in \mathcal{R}$ as follows:

$$q_v^r = \begin{cases} 1, \text{ if } \rho_v^r > 0 \text{ and } d_r^v = 0, \\ 0, \text{ otherwise.} \end{cases} \tag{13}$$

We consider each RSU r has wireless channel of capacity $\omega_{v,r}$ and each self-driving car $v \in \mathcal{V}$ can use one channel at each time slot. The channel is shared via time-division multiplexing fashion. As described in [32], at each time slot t, we assume that the channel is not changing. Therefore, $\omega_{v,r}$ can be expressed as follows:

$$\omega_{v,r} = q_v^r B_r \log_2\left(1 + \varphi_r |G_v^r|^2\right), \ \forall v \in \mathcal{V}, \ r \in \mathcal{R}, \tag{14}$$

where B_r is the authorized bandwidth for car to RSU communications, G_v^r is the channel gain between RSU r and self-driving car v, and φ_r is the scalar factor that represents the transmission power of RSU r. Therefore, based on channel capacity, the transmission delay for downloading content i from MEC server to self-driving car v is expressed as:

$$\tau_v^r = \frac{\sum_{\tilde{i}_f, \tilde{i}_m \in \mathcal{I}_r(n)} q_v^r \left(S(\tilde{i}_f)) + S(\tilde{i}_m)\right)}{\omega_{v,r}}, \tag{15}$$

where $\tilde{i}_f \in \mathcal{G}_j^{\text{female}}$ is the most requested content by consumers of the gender female and $\tilde{i}_m \in \mathcal{G}_j^{\text{male}}$ is the most requested content by consumers of the gender male in each cluster j, i.e., $\tilde{i}_m, \tilde{i}_m \in \mathcal{I}_r(n)$.

We consider t_v^r as a time required by self-driving car $v \in \mathcal{V}$ to leave an area covered by RSU r as follows:

$$t_v^r = \frac{2q_v^r \gamma_r}{\mu_v}, \tag{16}$$

where μ_v is the speed of self-driving car v. When $\tau_v^r < t_v^r$, the self-driving doesn't need to reduce μ_v for having more time to download the recommended contents in the region of RSU r. However, when $\tau_v^r \geq t_v^r$, without endangering other cars and breaking minimum speed limit allowed in its lane, the self-driving car can reduce μ_v for having more time to download more contents and cache them.

We consider that each self-driving car v has Integrated WiFi Router (IWR) on board, and it can provide WiFi connectivity to passengers. The IWR channel resources are shared to the passengers via contention-based model described in [33]. Therefore, the instantaneous data rate for each passenger u via IWR of self-driving car v is given by:

$$R_u^v = \frac{q_u^v \varphi_v \tilde{R}_u^v \xi_u^v(|\mathcal{U}_v|)}{|\mathcal{U}_v|}, \forall u \in \mathcal{U}_v, \ v \in \mathcal{V}_v, \tag{17}$$

where φ_v is WiFi throughput efficiency factor, and $|\mathcal{U}_v|$ is the number of passengers that be connected simultaneously to IWR of self-driving car v, where $\mathcal{U}_v \subset \mathcal{U}$. φ_v is used to determine overhead related to MAC protocol layering such as header, DIFS, SIFS, and ACK. Furthermore, $\tilde{R}_u^v$ is the maximum theoretical data rate that IWR can handle, which is assumed to be protocol depended and apriori known [33]. Furthermore, $\xi_u^v(|\mathcal{U}_v|)$ is channel utilization function [33], which is a function of number of passengers connected simultaneously to IWR. $\xi_u^v(|\mathcal{U}_v|)$ is used to determine impact of contention over WiFi throughput. In addition, we use q_u^v as a decision variable that indicates whether or not passenger u is connected to WiFi of self-driving v, specifically:

$$q_u^v = \begin{cases} 1, \text{ if passanger } u \text{ is connected to WiFi of car } v, \\ 0, \text{ otherwise.} \end{cases} \tag{18}$$

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

$$\tau_u^v = \frac{\sum_{i \in \mathcal{I}_r(n)} q_u^v \left(S(\tilde{i}_f)) + S(\tilde{i}_m)\right)}{R_u^v}. \tag{19}$$

C. Caching Model

The aim of caching contents in the self-driving car is to reduce the delay experienced by passengers in downloading content. This helps in improving Quality of Experience (QoE) for consumers (passengers) and quality of service for Content Providers (CPs).

We assume that the cache storage c_v for each self-driving car v is limited. Therefore, the sizes of the recommended

contents need to be downloaded from MEC server v and cached must satisfy cache resource constraint, which is expressed as follows:

$$q_v^r \sum_{j=1}^{K} \left(\sum_{\tilde{i}_f \in \mathcal{G}_j^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f)) + \sum_{\tilde{i}_m \in \mathcal{G}_j^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \right) \leq c_v, \tag{20}$$

where, in each cluster j, we let $o_v^{\tilde{i}_f} \in \{0,1\}$ be the decision variable that indicates whether or not self-driving car v has to cache content $\tilde{i}_f \in \mathcal{G}_j^{\text{female}}$, where $o_v^{\tilde{i}_f}$ is given by:

$$o_v^{\tilde{i}_f} = \begin{cases} 1, \text{ if self-driving car } v \text{ caches the content } \tilde{i}_f, \\ 0, \text{ otherwise.} \end{cases} \tag{21}$$

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 content $\tilde{i}_m \in \mathcal{G}_j^{\text{male}}$, where $o_v^{\tilde{i}_m}$ is given by:

$$o_v^{\tilde{i}_m} = \begin{cases} 1, \text{ if self-driving car } v \text{ caches the content } \tilde{i}_m, \\ 0, \text{ otherwise.} \end{cases} \tag{22}$$

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 any 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 $u \in \mathcal{U}$:

$$h_i^{u \to v} = \begin{cases} 1, \text{ if content } i \text{ requested by consumer } u \\ \quad \text{is returned from self-driving car } v, \\ 0, \text{ otherwise.} \end{cases} \tag{23}$$

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

$$\sum_{i \in \mathcal{I}_r(n)} o_r^i S(i) \leq c_r, \tag{24}$$

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:

$$o_r^i = \begin{cases} 1, \text{ if MEC server } r \text{ caches content } i \in \mathcal{I}_r(n), \\ 0, \text{ otherwise.} \end{cases} \tag{25}$$

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

$$h_i^{r \to v} = \begin{cases} 1, \text{ if the content } i \text{ requested by self-dring} \\ \quad \text{car } v \text{ is cached at MEC server } r, \\ 0, \text{ otherwise.} \end{cases} \tag{26}$$

Due to the limited cache capacity, when the cache storage is full, the self-driving car or MEC server replaces the contents by using Least Frequently Used (LFU) cache replacement policy [34] [35].

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

D. Computation Model for Cached Content

In the self-driving car, a passenger may request a content format (e.g., avi), which is not available in the cache storage c_v. Instead, the cache storage may have other content formats (e.g., mpeg) of the same content which can be transcoded to the desired format.

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

$$h_{i'}^{v \to u} = \begin{cases} 1, \text{ if content } i' \text{ requested by consumer } u \\ \quad \text{is returned by car } v \text{ after computation,} \\ 0, \text{ otherwise.} \end{cases} \tag{27}$$

In order to make sure that self-driving car v returns only one format of content, the following constraint has to be satisfied:

$$h_i^{u \to v} + h_{i'}^{v \to u} \leq 1. \tag{28}$$

We assume that converting content $i \in \mathcal{I}_r(n)$ to content i' requires to use computation resource p_v of self-driving car v, where computational resource allocation $p_v^{i \to i'}$ is given by:

$$p_v^{i \to i'} = p_v \frac{h_i^{u \to v} \varrho_v^{i \to i'} z^{i \to i'}}{\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} h_i^{u \to v} z^{i \to i'}}, \ \forall v \in \mathcal{V}, \tag{29}$$

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 computation decision variable which is expressed as:

$$\varrho_v^{i \to i'} = \begin{cases} 1, \text{ if the cached content } i \text{ is converted to} \\ \quad \text{desired format } i' \text{ in self-driving car } v. \\ 0, \text{ otherwise.} \end{cases} \tag{30}$$

In (29), for computational resources allocation, we use weighted proportional allocation, because it is simple to implement in practical communication systems such Vehicular Ad-hoc Networks (VANETs), 4G and 5G cellular networks [9], [36]. In weighted proportional allocation, each transcoding task receives a fraction of computational resources based on its computation workload requirement.

Furthermore, we assume that the computational resource p_v to be limited, and computation allocation must satisfy the following constraint:

$$\sum_{u=1}^{U} \sum_{i=1}^{I_r(n)} h_i^{u \to v} \varrho_v^{i \to i'} p_v^{i \to i'} \leq p_v. \tag{31}$$

In addition, converting content i to content i' involves the executing time. Therefore, in self-driving car v, as defined in [37], the execution time $\tau_v^{i\to i'}$ is given by:

$$\tau_v^{i\to i'} = \frac{h_i^{u\to v}\varrho_v^{i\to i'} z^{i\to i'} S(i)}{p_v^{i\to i'}}. \tag{32}$$

When constraint (31) cannot be satisfied due to insufficient computational resource for converting content i into requested content i', self-driving car forwards the demand for content i' to MEC server.

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

$$\tau_r^{i\to i'} = (1-\varrho_v^{i\to i'})\left(\frac{q_v^r h_i^{r\to v}\varrho_r^{i\to i'} z^{i\to i'} S(i)}{p_r^{i\to i'}}\right), \tag{33}$$

where $\varrho_r^{i\to i'}$ is a computation decision variable, which is expressed as:

$$\varrho_r^{i\to i'} = \begin{cases} 1, \text{ if the cached content } i \text{ is converted to} \\ \quad \text{desired format } i' \text{ at MEC server,} \\ 0, \text{ otherwise,} \end{cases} \tag{34}$$

where the computational resource allocation $p_r^{i\to i'}$ at RSU r for converting cached content i to content i' can be calculated as follows:

$$p_r^{i\to i'} = p_r \frac{(1-\varrho_v^{i\to i'})\left(\varrho_r^{i\to i'} h_i^{r\to v} z^{i\to i'} S(i)\right)}{\sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{I}} \varrho_r^{i\to i'} h_i^{r\to v} z^{i\to i'} S(i)}, \ \forall r \in \mathcal{R}. \tag{35}$$

In addition, we assume that computation resource at MEC server to be limited, where computation allocation has to satisfy the following constraint:

$$\sum_{v=1}^{V}\sum_{i=1}^{I_r(n)} q_v^r p_r^{i\to i'} \le P_r. \tag{36}$$

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

$$h_{i'}^{r\to v} = \begin{cases} 1, \text{ if content } i' \text{ requested by car } v \text{ is returned} \\ \quad \text{by MEC server } r \text{ after computation,} \\ 0, \text{ otherwise.} \end{cases} \tag{37}$$

Therefore, in order to ensure that converting i to i' is executed only at one location and MEC server r returns only one format of content, we impose the following constraints:

$$q_u^v(h_i^{u\to v} + h_{i'}^{v\to u}) + q_v^r\eta_v(h_i^{r\to v} + h_{i'}^{r\to v}) \le 1 \tag{38}$$

$$\varrho_v^{i\to i'} + q_v^r(1-\varrho_v^{i\to i'}) \le 1 \tag{39}$$

where $\eta_v = 1-(h_i^{u\to v} + h_{i'}^{v\to u})$. However, when the MEC server does not have enough computation resources to satisfy the above constraint (36), it forwards the demand for content i' to DC via backhaul wired link.

IV. Problem Formulation and Solution

To join aforementioned deep Learning, communication, and computation approaches, we formulate an optimization problem in Section IV-A. Finally, we propose a solution of our optimization problem in Section IV-B.

A. Problem Formulation

We formulate a novel deep learning based caching scheme in self-driving car that exploits 4C components of MEC as an optimization problem. The problem aims at minimizing total delay, where total delay $\tau_u^{\text{Tot}}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ for retrieving contents is given by:

$$\tau_u^{\text{Tot}}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}) = \tau_u^v + h_{i'}^{v\to u}\tau_v^{i\to i'} + (1-(h_i^{u\to v} + h_{i'}^{v\to u})) \left((\tau_v^r + \tau_r^{i\to i'} h_{i'}^{r\to v}) + (1-(h_i^{r\to v} + h_{i'}^{r\to v}))\tau_r^{\text{DC}}\right). \tag{40}$$

Therefore, for minimizing delay $\tau_u^{\text{Tot}}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$, the optimization problem can be expressed as follows:

$$\underset{\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}}{\text{minimize}} \quad \sum_{u=1}^{U}\tau_u^{\text{Tot}}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}) \tag{41}$$

subject to:

$$\sum_{v=1}^{V} q_v^r \le 1, \ \forall r \in \mathcal{R}, \tag{41a}$$

$$q_v^r \sum_{j=1}^{k}\Big(\sum_{\tilde{i}_f\in\mathcal{G}_j^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f)) + \sum_{\tilde{i}_m\in\mathcal{G}_j^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m)) \le c_v, \tag{41b}$$

$$\sum_{u=1}^{U}\sum_{i=1}^{I_r(n)} q_u^v h_i^{u\to v}\varrho_v^{i\to i'} p_v^{i\to i'} \le p_v, \ \forall v \in \mathcal{V}, \tag{41c}$$

$$q_u^v(h_i^{u\to v} + h_{i'}^{v\to u}) + q_v^r\eta_v(h_i^{r\to v} + h_{i'}^{r\to v}) \le 1, \tag{41d}$$

$$q_u^v\varrho_v^{i\to i'} + q_v^r(1-\varrho_v^{i\to i'}) \le 1. \tag{41e}$$

The constraint in (41a) ensures that the communication resource allocation for self-driving cars has to be less or equal to the total available communication resources of RSU $r \in \mathcal{R}$. The constraints in (41b) and (41c) guarantee that caching and computational resource allocations have to be less or equal to the available caching and computational resources of the self-driving car. The constraint in (41d) ensures that self-driving car or MEC server returns only one format of the requested content. The constraint (41e) ensures that converting i to i' is only executed at one location, either at MEC server r or at self-driving car v.

The formulated optimization problem in (41) has a non-convex structure, which makes it complicated to solve. Therefore, in order to make it convex and solve it easily, we use a Block Successive Majorization-Minimization (BS-MM) [25] and rounding technique [38], [39] described below in the Section IV-B.

B. Proposed Solution

We solve (41) by using BS-MM. The BS-MM belongs to a family of algorithms called Majorization-Minimization

(MM) algorithms described in [25]. We choose BS-MM over other MM algorithms because BS-MM allows partitioning problem into blocks and applies MM to one block of variables while keeping the values of other block fixed. To ensure that all blocks are utilized, we can use selection rules such as Cyclic, Gauss-Southwell, and Randomized described in [9], [25]. Therefore, with BS-MM, each subproblem can be solved separately using parallel computation. Furthermore, in order to use BS-MM in (41), we consider $\mathcal{Q} \triangleq \{\boldsymbol{q} : \sum_{u=1}^{U} q_u^v + q_v^r \geq 1,\ q_u^v,\ q_v^r \in [0,1]\}$, $\mathcal{H} \triangleq \{\boldsymbol{h} : \sum_{u=1}^{U}(h_i^{u\to v} + h_{i'}^{v\to u}) + (1-(h_i^{u\to v} + h_{i'}^{v\to u}))\,(h_i^{r\to v} + h_{i'}^{r\to v}) \leq 1, h_i^{u\to v}, h_{i'}^{v\to u}, h_i^{r\to v}, h_{i'}^{r\to v} \in [0,1]\}$, 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 relaxed $\boldsymbol{q}$, $\boldsymbol{h}$, and $\boldsymbol{\varrho}$, respectively. Therefore, to simplify our notation, we use $\mathcal{F}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ to denote (41), where $\mathcal{F}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ is expressed as follows:

$$\mathcal{F}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}) = \sum_{u=1}^{U} \tau_u^{\mathrm{Tot}}(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}). \tag{42}$$

To solve (42) by using BS-MM, we apply MM steps summarized below:

- *Step 1:* In the first step, called majorization, we need to find a convex surrogate function denoted by $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ which is the upper-bound of (42).
- *Step 2:* In the second step, called minimization, instead of minimizing (42) which is intractable, we need to minimize the convex surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$.

The success of BS-MM relies on choosing the surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$. Therefore, a surrogate function that is easy to solve and follows the shape of the objective function (42) is preferable. To achieve this, in majorization step, we use proximal minimization technique described in [25] and make the surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ by adding quadratic term $(\frac{\varrho_j}{2}\|(\boldsymbol{q}_j - \tilde{\boldsymbol{q}})\|^2)$ to (42). Therefore, the convex surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ can be expressed as follows:

$$\mathcal{F}_j(\boldsymbol{q}_j,\boldsymbol{q}^{(t)},\boldsymbol{h}^{(t)},\boldsymbol{\varrho}^{(t)}) \coloneqq \mathcal{F}(\boldsymbol{q}_j,\tilde{\boldsymbol{q}},\tilde{\boldsymbol{h}},\tilde{\boldsymbol{\varrho}}) + \frac{\alpha_j}{2}\|(\boldsymbol{q}_j - \tilde{\boldsymbol{q}})\|^2, \tag{43}$$

where $\tilde{\boldsymbol{q}}$, $\tilde{\boldsymbol{h}}$, $\tilde{\boldsymbol{\varrho}}$ are a given initial feasible points. Furthermore, the surrogate function in (43) can be applied to other vectors $\boldsymbol{h}$ and $\boldsymbol{\varrho}$. Due to its quadratic term $(\frac{\alpha_j}{2}\|(\boldsymbol{q}_j - \tilde{\boldsymbol{q}})\|^2)$, $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$ is convex optimization problem.

In minimization step, we minimize surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$, which is convex and upper-bound of the objective function in (42). Therefore, as the surrogate function in (43) can be divided into blocks, we consider $\mathcal{J}^t$ as set of indexes, where at each iteration t and $j \in \mathcal{J}^t$, α_j is used to denote a positive penalty parameter.

For minimizing the surrogate function $\mathcal{F}_j(\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho})$, we propose Algorithm 1, where we can obtain solution of (43) by solving the below optimization problems at each each iteration $t+1$:

$$\boldsymbol{q}_j^{(t+1)} \in \min_{\boldsymbol{q}_j\in\mathcal{Q}} \mathcal{F}_j(\boldsymbol{q}_j,\boldsymbol{q}^{(t)},\boldsymbol{h}^{(t)},\boldsymbol{\varrho}^{(t)}), \tag{44}$$

$$\boldsymbol{h}_j^{(t+1)} \in \min_{\boldsymbol{h}_j\in\mathcal{H}} \mathcal{F}_j(\boldsymbol{h}_j,\boldsymbol{h}^{(t)},\boldsymbol{q}_j^{(t+1)},\boldsymbol{\varrho}^{(t)}), \tag{45}$$

$$\boldsymbol{\varrho}_j^{(t+1)} \in \min_{\boldsymbol{\varrho}_j\in\mathcal{P}} \mathcal{F}_j(\boldsymbol{\varrho}_j,\boldsymbol{\varrho}^{(t)},\boldsymbol{q}_j^{(t+1)},\boldsymbol{h}_j^{(t+1)}). \tag{46}$$

For the obtained solution by using the relaxed vectors $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ that take values in $[0,1]$, we need to ensure that $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ are vectors of binary variables. In order to achieve this, we apply the rounding techniques described in [38]. As an illustration example, for a solution $q_v^{r*} \in \boldsymbol{q}_j^{(t+1)}$, we define rounding threshold $\varphi \in (0,1)$, where enforced binary value of q_v^{r*} is given by:

$$q_v^{r*} = \begin{cases} 1, \text{ if } q_v^{r*} \geq \varphi, \\ 0, \text{ otherwise.} \end{cases} \tag{47}$$

As highlighted in [9], [39], the rounding technique may violate communication, caching, and computation resource constraints. Therefore, as proposed in [9] and [39], to overcome this issue, we solved $\mathcal{F}_j$ in the form $\mathcal{F}_j + \beta_v\Delta_v$ by updating the constrains in (41a), (41b), and (41c) as follows:

$$\sum_{v=1}^{V} q_v^r a_v^r \leq 1 + \Delta_{v_a},\ \forall r \in \mathcal{R}, \tag{48}$$

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

$$q_v^r \sum_{j=1}^{k}\Big(\sum_{\tilde{i}_f\in\mathcal{G}_j^{\mathrm{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f)\Big) + \sum_{\tilde{i}_m\in\mathcal{G}_j^{\mathrm{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m)\Big) \leq c_v + \Delta_{v_c}, \tag{50}$$

where, in self-driving car, we use $\Delta_v = \Delta_{v_a} + \Delta_{v_p} + \Delta_{v_c}$ as maximum violation of communication, caching, and computation 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:

$$\Delta_{v_a} = \max\{0, \sum_{v=1}^{V} q_v^r a_v^r - 1\},\ \forall r \in \mathcal{R}, \tag{51}$$

$$\Delta_{v_p} = \max\{0, \sum_{u=1}^{U}\sum_{i=1}^{I_r(n)} q_u^v h_i^{u\to v} \varrho_v^{i\to i'} p_v^{i\to i'} - p_v\},\ \forall v \in \mathcal{V}, \tag{52}$$

$$\Delta_{v_c} = \max\{0, q_v^r \sum_{j=1}^{k}\Big(\Big(\sum_{\tilde{i}_f\in\mathcal{G}_j^{\mathrm{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f)\Big) + \sum_{\tilde{i}_m\in\mathcal{G}_j^{\mathrm{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m)\Big) - c_v\}. \tag{53}$$

In order to evaluate the quality of our rounding technique, we use the integrality gap defined and proved in [38]. Therefore, for given problems $\mathcal{F}_j + \beta_v\Delta_v$ and $\mathcal{F}_j$, the integrality gap is expressed as follows:

$$\phi_j = \min_{\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}} \frac{\mathcal{F}_j}{\mathcal{F}_j + \beta_v\Delta_v}. \tag{54}$$

For $\mathcal{F}_j$, the solution is obtained by using relaxed vectors $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$. On other hand, for $\mathcal{F}_j + \beta_v\Delta_v$, the solution

Algorithm 1 : Distributed optimization control algorithm for deep learning based caching.

1: **Preconditions:** MLP outputs are deployed to the RSUs;
2: **Input:** $\boldsymbol{U}$: A vector of passengers; $\boldsymbol{V}$: A vector of self-driving cars, $\boldsymbol{R}$: A vector of RSUs, $\boldsymbol{\omega}_r^{\mathrm{DC}}$: A vector of backhaul capacities, $\boldsymbol{\omega}_v^r$: A vector of wireless link capacities, $\tilde{\boldsymbol{y}}$: MLP outputs, $\boldsymbol{\mathcal{X}}$: Vector of recommended contents for self-driving car v, R_u^v p_v, c_v, p_r and c_r;
3: **Output:** $\boldsymbol{q}^*$, $\boldsymbol{h}^*$, $\boldsymbol{\varrho}^*$;
4: Initialize $t = 0$;
5: Find initial feasible points $(\boldsymbol{q}^{(0)}, \boldsymbol{h}^{(0)}, \boldsymbol{\varrho}^{(0)})$;
6: **repeat**
7: Choose index set $\mathcal{J}^t$;
8: Let $\boldsymbol{q}_j^{(t+1)} \in \min_{\boldsymbol{q}_j \in \mathcal{Q}} \mathcal{F}_j(\boldsymbol{q}_j, \boldsymbol{q}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{\varrho}^{(t)})$;
9: Set $\boldsymbol{q}_k^{t+1} = \boldsymbol{q}_k^t, \forall k \notin \mathcal{J}^t$ and solve $\min_{\boldsymbol{q}_j \in \mathcal{Q}} \mathcal{F}_j(\boldsymbol{q}_j, \boldsymbol{q}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{\varrho}^{(t)})$;
10: For getting $\boldsymbol{h}_j^{(t+1)}$ and $\boldsymbol{\varrho}_j^{(t+1)}$, restart from step 4, salve (45) and (46);
11: $t = t + 1$;
12: **until** $\lim_{t\to\infty} \inf_{\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$;
13: By rounding technique, enforce $\boldsymbol{q}_j^{(t+1)}$, $\boldsymbol{h}_j^{(t+1)}$, and $\boldsymbol{\varrho}_j^{(t+1)}$ to be vectors of binary variables;
14: Solve $\mathcal{F}_j + \beta_v \Delta_v$, and compute ϕ_j;
15: Then, consider $\boldsymbol{q}^* = \boldsymbol{q}_j^{(t+1)}$, $\boldsymbol{h}^* = \boldsymbol{h}_j^{(t+1)}$, and $\boldsymbol{\varrho}^* = \boldsymbol{\varrho}_j^{(t+1)}$ as a solution.

is obtained by enforcing the vectors $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ to be the vectors of binary variables. Therefore, the best rounding is achieved, when $\phi_j \leq 1$ ($\phi_j = 1$ for the feasible solution).

1) Proposed Algorithm: We propose a Distributed optimization control algorithm for deep learning based caching in the self-driving car (Algorithm 1), which is based on BS-MM [25].

In Algorithm 1, we assume that the MLP are already deployed at RSUs. We consider vector of passengers, vector of self-driving cars, vector of RSUs, vector of backhaul capacities, vector of wireless link capacities, vector of MLP outputs for recommended contents that need to be cached at RSU in each region n, vector of recommended contents that need to be cached in self-driving car v, R_u^v, p_v, c_v, p_r, and c_r as the inputs. First, the Algorithm 1 finds the initial feasible points $\tilde{\boldsymbol{q}} = \boldsymbol{q}^{(0)}$, $\tilde{\boldsymbol{h}} = \boldsymbol{h}^{(0)}$, and $\tilde{\boldsymbol{\varrho}} = \boldsymbol{\varrho}^{(0)}$. Then, the Algorithm 1 starts an iterative process by choosing index set $\mathcal{J}^t$ at iteration t. After that, the solution of (43) is computed and updated by solving the optimization problems (44), (45), and (46) until $\lim_{t\to\infty} \inf_{\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$. Therefore, as our caching technique is based on content prefetching, in (44), we assume that self-driving car needs to be connected to the RSU first for downloading contents. After establishing connection by solving (45), we can download and serve cached contents based on demands. Based on desired format of the contents, we can solve the problem in (46) related to computation of cached contents. We enforce the solution $\boldsymbol{x}_j^{(t+1)}$, $\boldsymbol{y}_j^{(t+1)}$, and $\boldsymbol{w}_j^{(t+1)}$ to be vector of binary variables via the rounding technique (47), then the Algorithm 1 solves $\mathcal{F}_j + \beta_v \Delta_v$, and compute ϕ_j. Finally, the Algorithm 1 considers $\boldsymbol{q}^* = \boldsymbol{q}_j^{(t+1)}$, $\boldsymbol{h}^* = \boldsymbol{h}_j^{(t+1)}$, and $\boldsymbol{\varrho}^* = \boldsymbol{\varrho}_j^{(t+1)}$ as a solution.

For the converge of the proposed algorithm, based on converged of MM defined and proved in [25], we can make the following remark:

Remark 1 (Convergence of the proposed algorithm). *The proposed Algorithm 1 for deep learning based caching in self-driving car, which based on BS-MM algorithm, converges to minimum points called coordinate-wise minimum, when the vectors $\boldsymbol{q}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{\varrho}_j$ reaches points, where they cannot find a better minimum direction, i.e.,* $\lim_{t\to\infty} \mathit{inf}_{\boldsymbol{q},\boldsymbol{h},\boldsymbol{\varrho}} \|\mathcal{F}_j^{(t+1)} - \mathcal{F}_j^{(t)}\|_2 = 0$.

Table II: The route for self-driving bus.

Route	Distance (Km)	Maximum speed (Km/h)	RSUs
1	218.49	109.016	$1-2$
2	215	107.34	$2-3$
3	216.19	108.17	$3-4$
4	211.10	105.38	$4-5$
5	222.66	111.33	$5-6$

V. Simulation Results and Analysis

In this section, we present the performance evaluation of the proposed deep learning based caching in the self-driving car. We use Google Maps Services [40] for self-driving car mobility analysis, and Keras with Tensorflow [41] for deep learning simulation.

A. Simulation Setup

To predict the probability of contents to be requested in specific areas, we use MLP described in Section III-A1 and a well-known dataset called Movie-Lens Dataset [42]. In this dataset, we have movies with their related information such as movie titles, release dates, and type of movies. However, the dataset doesn't have movie sizes and formats. Since, our deep learning based caching scheme is based on content size, we generate a random size $S(i)$ for each movie i in the range from $S(i) = 317$ to $S(i) = 750$ Mb and assign randomly each movie i the format (.avi or .mpg). In addition, we have users information such as age, gender, their rating for the movies, and ZIP codes. We convert ZIP codes into longitude and latitude coordinates and deploy RSUs to the specific areas based on the location of the consumers of the contents. With departure time and location of RSUs, Google Maps Service provides distance and duration to reach each RSU $r \in \mathcal{R}$, where the duration is based on traffic condition between source and destination. Based on distance (in terms of km) and duration (in terms of hours), we can calculable the speed (in terms of km/h) of the self-driving car and finds the RSU that the self-driving car could potentially connect to for downloading the content.

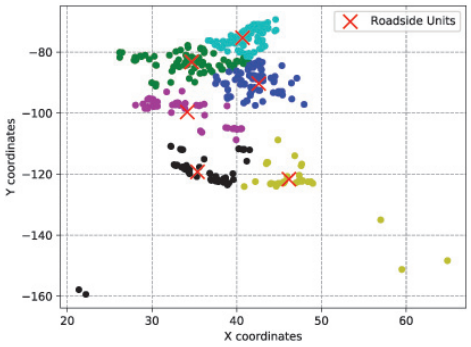

Figure 2: Cache-enabled RSUs deployment.

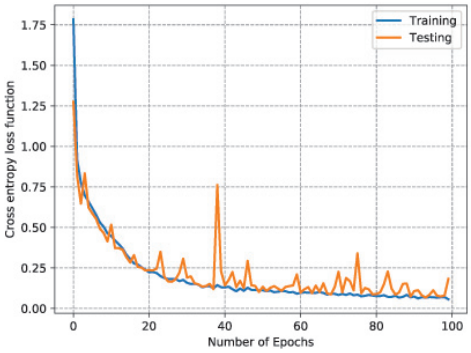

Figure 3: Cross entropy loss function for videos need to be cached at RSUs (acc: 0.9804).

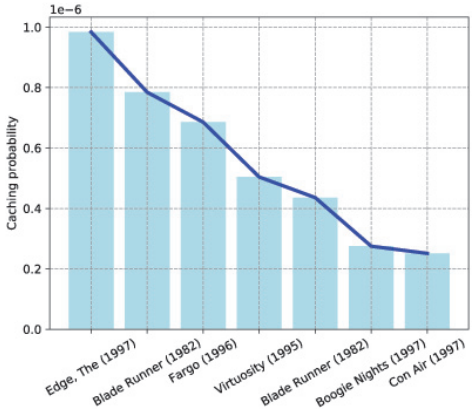

Figure 4: Videos need to be cached at RSUs (RSU 1).

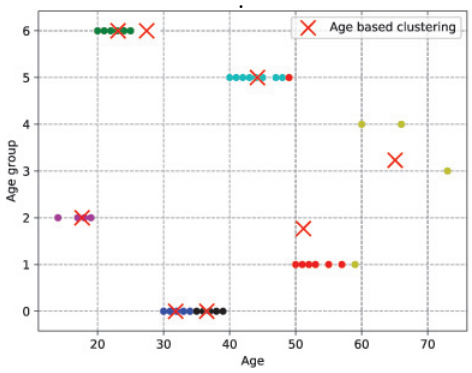

Figure 5: Age based clustering by using k-means.

However, based on Google Maps Services [40], we realize that the distances between RSU units are very high. Then, we update the RSU locations and make a routing table summarized in Table II, where the self-driving car starts its journey from RSU 1 and ends it to RSU 6. We set each RSU $r \in \mathcal{R}$ to be connected to DC with wired backhaul of capacity in the range from $\omega_r = 60$ to $\omega_r = 70$ Mbps. We consider that each RSU $r \in \mathcal{R}$ has the bandwidth of $\omega_{v,r} = 10$ MHz. On the other hand, each MEC server $\in \mathcal{R}$ has CPU of capacity $p_r = 3.6$ GHz, while cache capacity is the range from $c_r = 100$ to $c_r = 110$ terabytes (TB).

For self-driving car $v \in \mathcal{V}$, we consider one self-driving bus with $|\mathcal{U}_v| = 37$. The self-driving car has a bandwidth of 160 MHz (802.11ac) with maximum theoretical data rate of $\tilde{R}_u^v = 3466.8$ Mbps. Furthermore, the computation capacity of self-driving car is set to $p_v = 3.6$ GHz, while cache capacity is set to $p_v = 100$ TB.

B. Evaluation Results

Before starting using MLP, based on video watching counts and location information, we select six areas to deploy cache-enabled RSUs by using the k-means algorithm. In these six areas depicted in Fig. 2, we predict the probabilities of contents to be cached in these regions by using MLP of 3 layers with 2 hidden layer. As shown in Fig. 3, we minimize Cross entropy loss function, an accuracy of 98.04% is achieved for predicting movies that need to be cached at RSUs.

For the videos that need to be cached at RSUs, each RSU $v \in \mathcal{V}$ caches them by starting the ones who have high probabilities to be requested in its area in descending order until the cache storage becomes full or there are no more videos required to be cached. As an example, the Fig. 4 demonstrates the video needs to be cached at the RSU 1 with their predicted probabilities by using MLP.

Caching at RSUs is based on location. However, in addition to location, caching in the self-driving car is based on age and gender. Therefore, when the self-driving car is connected to RSU, it downloads the MLP outputs from the RSU. Then, it groups the MLP outputs based on age by using k-means algorithm and gender by using binary classification described in Section III-A1. Fig. 5 show the age-based clustering in the self-driving bus by using k-means. Even if the self-driving bus can have many passengers, we assume that it starts the journey with 37 passenger on board. The CNN can be used for predicting age and gender of passengers and then self-driving car uses k-means and binary classification for classifying the passengers. The Fig. 6 shows the results of both k-means and binary classification. In other words, each age-based cluster has two sub-clusters: male and female. From both joined MLP and CNN outputs via k-means and binary classification, we infer a recommendation of the contents that require to be cached in the self-driving car. The top recommended contents and their caching order are shown in Fig. 7.

Based on demands of passengers, fig. 8 shows the normalized cache hits for the self-driving car, where the videos not

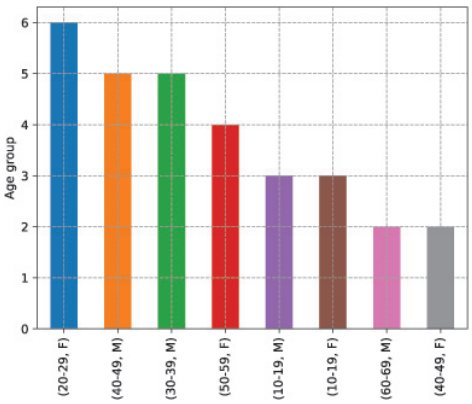

Figure 6: Gender and age of passengers in self-driving bus.

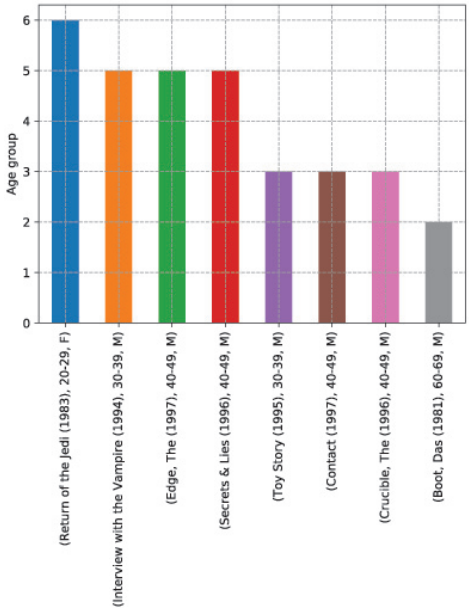

Figure 7: Recommended videos to cache in self-driving bus.

cached in self-driving (cache misses) require to be retrieved at RSU or DC. The results in this figure demonstrate that the cache hits increase with Zipf parameter, i.e., the videos which have high demands are characterized with high cache hit rates.

Fig. 9 shows the solution of surrogate function (43), where (43) minimizes total delays (transmission delay and transcoding delay). In the beginning, the self-driving car and its passengers need to experience with high delay in downloading the contents (video). Once the contents are cached inside the cars, the delay can be minimized and converged to a stationary point. In other words, at stationary point, the problem (43) cannot find a better minimum direction. However, as shown in this figure, the choice of α_j has an impact on size and convergence speed of surrogate function $\mathcal{F}_j(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{\varrho})$.

VI. Conclusion

In this paper, we proposed a novel framework that uses deep Learning for content caching in the self-driving car. In the proposed framework, at DC, we proposed Multi-Layer Perceptron (MLP) for predicting the probabilities of

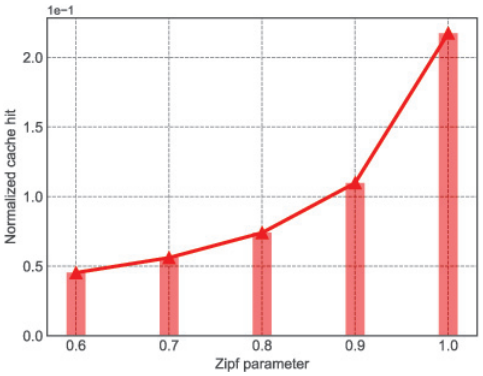

Figure 8: Normalized cache hits for the self-driving bus.

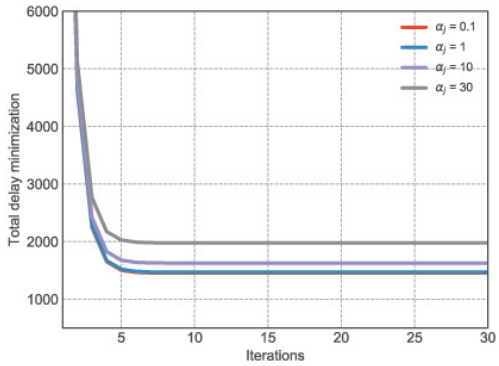

Figure 9: Total delay minimization problem (seconds).

contents to be requested in specific areas. Then, outputs are deployed in MEC servers (at RSUs) in close proximity to the self-driving car, where each MEC server downloads and caches the contents that have high probabilities of being requested in its area of coverage. Furthermore, for a self-driving car, in order to cache entertainment contents that are appropriate to the age and gender of passengers, we proposed convolutional neural network (CNN) for predicting age and gender. Then, the self-driving car downloads MLP output from MEC server and compares its CNN and MLP outputs by using k-means and binary classification for identifying the required contents to be downloaded and cached. Therefore, we formulated deep learning based caching problem as an optimization problem that aims at minimizing content-downloading delay. The simulation results demonstrate that our approach can minimize delay and be easily implemented in the self-driving environment.

References

[1] Icebike, "Real time traffic accident statistics," https://www.icebike.org/real-time-traffic-accident-statistics/, [Online; accessed July. 11, 2018].

[2] Southsideinjuryattorneys, "Georgia personal injury blog," http://southsideinjuryattorneys.com/lawyer/2016/07/21/Personal-Injury/New-Data-Shows-94-Percent-of-Car-Accidents-Caused-by-Human-Error_bl25860.htm, [Online; accessed July. 11, 2018].

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

[4] Business Insider, "These 19 companies are racing to put driverless cars on the road by 2020," https://www.businessinsider.

com/companies-making-driverless-cars-by-2020-2016-8, [Online; accessed September. 4, 2018].
[5] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems," *arXiv preprint :1712.04135*, 12 Dec. 2017.
[6] S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni, "A novel on-board unit to accelerate the penetration of its services," in *Proceedings of IEEE 13th Consumer Communications & Networking Conference (CCNC)*, 9-12 Jan. 2016 (Las Vegas, NV, USA), pp. 467–472.
[7] 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, 5 Sep. 2015.
[8] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in *Proceedings of IEEE 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 27-29 Sept. 2017 (Seoul, South Korea), pp. 366–369.
[9] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *arXiv preprint:1803.11512*, 30 Mar. 2018.
[10] BUSBUD, "Will driverless buses be a reality?" https://www.busbud.com/blog/will-driverless-buses-reality/, [Online; accessed August. 11, 2018].
[11] Hollywood Reporter, "Why Hollywood could make billions from self-driving cars," https://www.hollywoodreporter.com/behind-screen/why-driving-cars-could-be-hollywoods-next-big-thing-1031554, [Online; accessed July. 11, 2018].
[12] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. S. Shen, "Cost-effective vehicular network planning with cache-enabled green roadside units," in *Proceedings of IEEE International Conference on Communications (ICC)*, 21-25 May 2017 (Paris, France), pp. 1–6.
[13] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6321–6334, 2017.
[14] F. Chen, D. Zhang, J. Zhang, X. Wang, L. Chen, Y. Liu, and J. Liu, "Distribution-aware cache replication for cooperative road side units in vanets," *Peer-to-Peer Networking and Applications*, pp. 1–10, 2017.
[15] 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. 33 702–33 717, 2018.
[16] 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 Communications*, vol. 25, no. 3, 2018.
[17] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.
[18] 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 *Proceedings of IEEE Globecom Workshops (GC Wkshps)*, 4-8 Dec. 2017 (Singapore), pp. 1–6.
[19] 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 Network*, vol. 32, no. 1, pp. 80–86, 2018.
[20] Geocaching, "Self Driving Car Caching," https://forums.geocaching.com/GC/index.php?/topic/347757-self-driving-car-caching/, [Online; accessed July. 11, 2018].
[21] Next Analytics, "Why Hollywood could make billions from self-driving cars," https://www.nextanalytics.com/excel-youtube-analytic-insights-and-data-mining/page/4/, [Online; accessed July. 11, 2018].
[22] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," *arXiv preprint:1710.06799*, 2017.
[23] J. J. Whang, I. S. Dhillon, and D. F. Gleich, "Non-exhaustive, overlapping k-means," in *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 30 Apr-2 May, 2015 (British Columbia, Canada, pp. 936–944.
[24] J. Martineau, T. Finin, A. Joshi, and S. Patel, "Improving binary classification on text problems using differential word features," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 2019–2024.
[25] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2017.
[26] E. Erdem, "Predicting movie rating based on tags using machine learning and deep learning," https://github.com/AdvRegProj/MovieLens-ML-LSTM/blob/master/Movie%20Rating%20Prediction%20using%20GloVe%20Word%20Embeddings%20and%20Deep%20Learning.ipynb, [Online; accessed August. 8, 2018].
[27] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint:1803.01164*, 2018.
[28] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 34–42.
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint:1409.1556*, 2014.
[30] V. Shaw and S. Dowlatkhah, "Network edge based access network discovery and selection," Apr. 18 2017, US Patent 9,629,076.
[31] E. Ndashimye, N. I. Sarkar, and S. K. Ray, "A novel network selection mechanism for vehicle-to-infrastructure communication," in *Proceedings of IEEE 14th Intl. Conf. on Pervasive Intelligence and Computing, 2nd Intl. Conf. on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, 8-12 Aug. 2016 (Auckland, New Zealand), pp. 483–488.
[32] T. Wang, L. Song, and Z. Han, "Coalitional graph games for popular content distribution in cognitive radio vanets," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 8, pp. 4010–4019, 2013.
[33] 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 Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1944–1955, 2016.
[34] K. Shah, A. Mitra, and D. Matani, "An o (1) algorithm for implementing the lfu cache eviction scheme," *Technical report*, 2010.
[35] A. Ndikumana, K. Thar, T. M. Ho, N. H. Tran, P. L. Vo, D. Niyato, and C. S. Hong, "In-network caching for paid contents in content centric networking," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 4-8 Dec. 2017 (Singapore), pp. 1–6.
[36] S. Mosleh, L. Liu, and J. Zhang, "Proportional-fair resource allocation for coordinated multi-point transmission in lte-advanced," *IEEE Transactions on Wireless Communications*, vol. 15, no. 8, pp. 5355–5367, 2016.
[37] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
[38] U. Feige, M. Feldman, and I. Talgam-Cohen, "Oblivious rounding and the integrality gap," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 60. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
[39] 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 Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2512–2521, 2017.
[40] Google, "Python client library for google maps api web services," https://github.com/googlemaps/google-maps-services-python, [Online; accessed August. 12, 2018].
[41] Keras, "Keras: The Python Deep Learning library," https://github.com/udacity/self-driving-car-sim, [Online; accessed July. 20, 2018].
[42] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM transactions on interactive intelligent systems*, vol. 5, no. 4, p. 19, 2016.