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

Anselme Ndikumana and Choong Seon Hong*
Department of Computer Science and Engineering, Kyung Hee University, Rep. of Korea
{anselme, cshong}@khu.ac.kr

***Abstract*—In the future, self-driving cars are expected to be involved in public transportation. Once passengers are comfortable with them, the self-driving cars will be new spaces for entertainment. However, getting infotainment contents from Data Centers (DCs) can be perturbed by the high end-to-end delay. To address this issue, we propose caching for infotainment contents in close proximity to the self-driving cars and in self-driving cars. In our proposal, Multi-access Edge Computing (MEC) helps self-driving cars by deploying MEC servers to the edge of the network at macro base stations (BSs), WiFi access points (WAPs), and roadside units (RSUs) for caching infotainment contents in close proximity to the self-driving cars. Based on the passenger's features learned via self-driving car deep learning approach proposed in this paper, the self-driving car can download infotainment contents that are appropriate to its passengers from MEC servers and cache them. The simulation results show that our prediction for the infotainment contents need to be cached in close proximity to the self-driving cars can achieve 99.28% accuracy.**

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

I. Introduction

In the future, self-driving buses are expected to roll down in smart cities [1]. Therefore, the passengers have to ensure that they get the right buses. The self-driving buses also have to ensure that they pick-up the right passengers at the right places and times. To achieve this, self-driving buses need to be equipped with a plethora of high-quality sensors and cameras that help in collecting and analyzing heterogeneous data in real-time. In addition, the absence of a human driver should be replaced by artificial intelligence for guiding and assisting passengers and controlling infotainment contents to play in the self-driving car. To achieve this, the camera of the self-driving car can be used to get passengers' features via facial images. This can help the self-driving car to provide enhanced and customized services such as seating arrangements, infotainment contents, etc. to the passengers [2].

In the coming years, the self-driving car will be considered as a new space for entertainment. Therefore, when the infotainment contents are not in the self-driving car, the car has to download them from the Data Center(DC). However, downloading the contents from DCs can increase backhaul bandwidth consumption and cause delay. To address this issue, we need infotainment contents to be available near the self-driving cars and in the cars. To achieve this, Multi-access Edge Computing (MEC) [3] can assist the self-driving cars by deploying MEC servers at the edges for infotainment content caching. Typically, here in our work, MEC servers are deployed at macro base stations (BSs), WiFi access points (WAPs) and roadside units (RSUs) for caching infotainment contents in close proximity to the self-driving cars. In addition, to reduce MEC-car delay, we need infotainment contents to be cached in the self-driving cars. However, people have different choices for infotainment contents, in which their choices depend on some features such as ages and genders [4]). Therefore, in the self-driving car, the infotainment contents should be cached based on the passenger's features.

To address the above challenges, we propose MEC assistance to a self-driving car for deep learning based caching. The main contributions of this paper are:

- Based on age and gender, people have different choices for the contents [4]. Therefore, we propose a CNN profiling approach [5] for people within a self-driving car, where the CNN is used to predict their ages, and gender. CNN profiling outputs are used by self-driving car for the purpose of deciding on which infotainment contents to cache such as music, video, and game data.
- To get infotainment contents needed to be cached near the self-driving cars at MEC servers, at DC, we propose Multi-Layer Perceptron (MLP) [5] approach to predict the probabilities of infotainment contents need to be requested in areas of MEC servers. MLP outputs are sent to MEC servers. Then, the MEC server downloads and caches the contents that have high probabilities to be requested in its areas of coverage.
- The self-driving car can request MEC server the MLP outputs and apply k-means [6] and binary classification [7] on CNN and MLP outputs for identifying the infotainment contents need to be downloaded and cached.

As related works, in [8]–[10], authors discussed on content caching at BSs, routers, and RSUs. However, in all these works, caching for infotainment contents in a self-driving car was not considered. In [11], the authors presented a cloud-based VANET. In cloud-based VANET, both vehicles and RSUs participate in content caching. Finally, in [12], the authors introduced caching at edge servers (BSs) and at autonomous cars, where the edge server pushes contents to some cars that have enough influences to share them to other cars. However, BSs and cars may belong to the different operators. Therefore, without incentive mechanism, there are no motivations for car's owners to participate in content sharing. The novelty of this paper over related works in [8]–[13] is: this paper proposes new deep learning based caching approach in the self-driving

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2015-0-00557, Resilient/Fault-Tolerant Autonomic Networking Based on Physicality, Relationship and Service Semantic of IoT Devices) *Dr. CS Hong is the corresponding author.

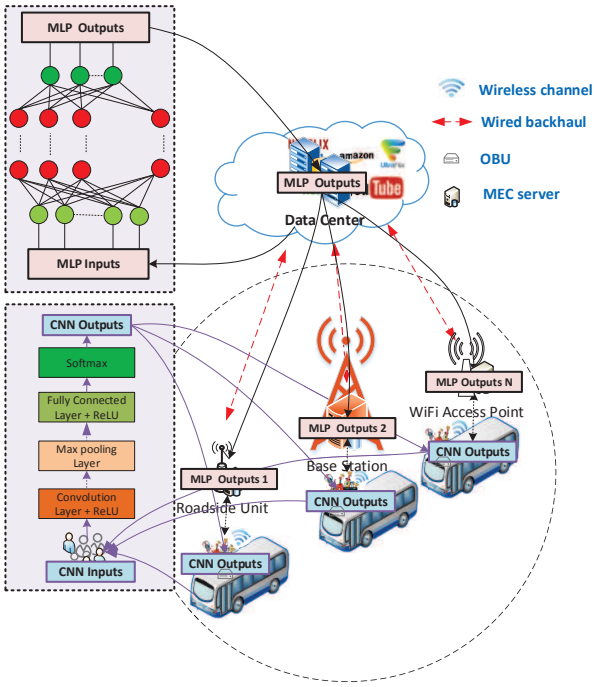

Figure 1: System model.

car assisted by MEC servers.

The rest of this paper is structured as follows. We present system model in Section II, while Section III discusses deep learning-based caching approach. In Section IV, we present performance evaluation. We conclude the paper in Section V.

II. System model

As shown in Fig. 1, in DC, we use MLP and dataset for predicting the probabilities of contents to be requested in a specific area of RSUs, BSs, and WAPs. We consider that each RSU, BS, and WAP have access to the wired backhaul of capacity ω_r^{DC}, ω_b^{DC}, ω_w^{DC} Mbytes/second respectively. We consider the backhaul capacities of RSUs, BSs and WAPs to be limited. In addition, each RSUs, BSs and WAPs have a wireless link of bandwidth ω_v^r, ω_v^b, ω_v^w respectively. Furthermore, each RSU, BS, and WAP has MEC server, where $\mathcal{R}$ is the set of R RSUs, $\mathcal{B}$ is the set of B BSs, and $\mathcal{W}$ is the set of W WAPs. We consider an MEC network composed of a set $\mathcal{S} = \mathcal{B} \cup \mathcal{R} \cup \mathcal{W}$ of MEC servers. Unless stated otherwise, we use also the terms "$\mathcal{B}$", "$\mathcal{R}$", and "$\mathcal{B}$" to denote the set of MEC servers attached to BSs, RSUs, and WAPs, respectively. By using backhaul communication resources, based on the MLP outputs, each MEC server can download and cache predicted contents that have high probabilities to be requested in its region/ areas. We use $\mathcal{N}$ to denote a set of N areas and $\mathcal{I}$ to denote a set of contents, where each content $i \in \mathcal{I}$ has size of $S(i)$ Mb. We consider that the MEC servers are belonged to one Mobile Network Operator (MNO). The MNO has a total storage capacity of C and a total computation capacity of P. The total cache storage pool for the MNO is $C = \sum_{b\in\mathcal{B}} c_b + \sum_{r\in\mathcal{R}} c_r + \sum_{w\in\mathcal{W}} c_w$ and the total computational resource pool is $P = \sum_{b\in\mathcal{B}} p_b + \sum_{r\in\mathcal{R}} p_r + \sum_{w\in\mathcal{W}} p_w$

At the edge, we consider that the self-driving car has OBU that can manage to cache for infotainment contents. We consider $\mathcal{V}$ as a set of self-driving vehicles, where each self-driving car $v \in \mathcal{V}$ has cache store of capacity c_v and computation capability p_v. Furthermore, for deciding on which infotainment content to request and cache in a self-driving car, we use CNN profiling approach to predict ages, and gender of people within a car. We use $\mathcal{U}$ to denote people in the dataset, and $\mathcal{U}_v$ to denote the set of passengers of self-driving car $v \in \mathcal{V}$. We consider that each self-driving car v has a camera system for automatic facial recognition that can be used to provide enhanced and customized services to people on board. After CNN prediction, based on signal strength, the self-driving car can request its nearest RSUs, BSs, or WAPs the MLP prediction. Then, by using k-means and binary classification, the self-driving car can compare CNN prediction and MLP prediction and comes up with the recommendation for the infotainment contents which are appropriate to people on board. Then, self-driving car downloads and caches the recommended contents.

III. Deep Learning Based Caching

In this section, we present MLP, CNN, and recommendation models for deep learning based caching. Finally, we end the section with communication and caching models.

A. Multi-Layer Perceptron (MLP)

As shown in Fig. 1, MLP is deployed at DC, where green circles are for neurons of input and output layers and red circles are for neurons of hidden layers. Furthermore, we use demographical dataset for content demands, where in dataset, we have content names, rating, viewer's age, gender, and ZIP codes [14]. We use these features as an inputs of MLP, where $\boldsymbol{x} = (x_1, x_2, \dots x_M)^T$ is used to denote the input vector and subscripts are used to denote different features. Then, from $\boldsymbol{x}$, we predict $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \tilde{y}_2, \dots \tilde{y}_N)^T$ as the output vector. In $\tilde{\boldsymbol{y}}$, subscripts are used to denote N geographical locations. Furthermore, each geographical location $n \in \mathcal{N}$ is associated with one neuron at the output layers. In addition, $\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(l)}$ is used to denote weight matrices, l is for the number of hidden layers, while $\boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(l)}$ is for bias vectors. Therefore, $\tilde{\boldsymbol{y}} = f(\boldsymbol{W}^{(l)} \dots f(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}) \cdots + \boldsymbol{b}^{(l)})$.

In the training process, we need to adjust $\boldsymbol{w}$ so that we can predict correct outputs $\tilde{\boldsymbol{y}}$ from inputs $\boldsymbol{x}$. To achieve this, in MLP, errors needs to be minimized. Because we need to classify the contents needs to be cached in N geographical locations, our problem is classification problem. Therefore, we use cross-entropy error function $A(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = -\sum_{n=1}^{N} y_n \log \tilde{y}_n$, where $\tilde{\boldsymbol{y}}$ is the estimated class probabilities $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ is the ground truth. Finally, the outputs of MLP are deployed to MEC servers in close proximity to the self-driving cars.

B. Convolutional Neural Network (CNN)

CNN is one of ANN approaches mostly used for analyzing visual imagery [5]. Here, CCN is used to automatically predict age and gender from facial images of people [15] in self-driving cars. We use $\boldsymbol{i}_0^v$ to denote input image of incoming passenger(s) in self-driving car $v \in \mathcal{V}$. The $\boldsymbol{i}_0^v$ has three dimensions space: height, width, color. We consider three color channels: red, green, and blue.

In CNN, first, we use the convolution layer to filter input regions, computes the output of each region of an image and comes out with a feature map $\boldsymbol{i}_l^v$ after convolution layer

l. Second, we use ReLU pooling layer by applying ReLU $\max(0, i_l^v)$ activation function. Third, thus ReLU maintains the size of its previous convolution layer l unchanged which correspond to a high dimensional matrix, we use max pooling layer as matrix downsampling operation by focusing on dense regions. Fourth, we use two fully-connected layers to compute the class scores that passenger could potentially belong to. The first fully-connected layer corresponds to gender classes (male and female), while the second fully-connected layer corresponds to age classes (from 0 to 101). Finally, we use a softmax layer by applying softmax function that helps us to squeeze the outputs as probabilistic values that indicate the classes for gender and age that a face of the passenger could belong to.

C. Recommendation Model

To get the contents that need to be cached in a self-driving car, we consider that each MEC server downloads and caches the contents that have high probabilities to be requested in its area by using the MLP outputs. First, once each self-driving car $v \in \mathcal{V}$ gets connected to MEC server, it downloads MLP output. Second, from MLP outputs, the self-driving car uses k-means algorithm to make an age-based cluster. Inside the age-based cluster, the self-driving car uses binary classification to make gender-based clusters. Therefore, we denote $\tilde{\boldsymbol{y}}_n$ as MLP outputs for geographical location $n \in \mathcal{N}$, where $\mathcal{X} = \tilde{\boldsymbol{y}}_n$ is set as inputs of k-means algorithm. Then, the k-means classify data points $\mathcal{X} = \{x_1, \ldots, x_U\}$ into K clusters $\mathcal{X}_1, \ldots, \mathcal{X}_K$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$, where K sets to be equal to the number of age categories (we have 101).

For gender-based clusters, we use binary classification to make two groups in each age-based cluster j, one group for females $\mathcal{G}_j^{\text{female}}$) and another group for males $\mathcal{G}_j^{\text{male}}$, such that $\mathcal{X}_j = \mathcal{G}_j^{\text{female}} \cup \mathcal{G}_j^{\text{male}}$ and $\mathcal{G}_j^{\text{female}} \cap \mathcal{G}_j^{\text{male}} = \emptyset$.

The $\mathcal{X}_j = \mathcal{G}_j^{\text{female}} \cup \mathcal{G}_j^{\text{male}}$ has the predicted contents that have high probabilities to be requested in area of self-deriving car $v \in \mathcal{V}$. Furthermore, for each incoming passenger $u \in \mathcal{U}_v$, each self-driving car v uses CNN to predict his age and gender and classifies passenger in appropriate cluster. After clustering, the self-driving car downloads the contents and caches them based on probabilistic values in decreasing order.

D. Communication Model

We assume that each self-driving car v moves in the area covered by many network access technologies such as WiFi, RSU, and 4G/5G and it requires at least to have one active connection. Therefore, to select which access technology to use for downloading the contents, in this subsection, we discuss on BS, RSU, and WiFi selection approach.

1) Communication with BS: We assume that BSs are covering most of the N geographical locations and can be used by self-driving cars anytime. When each self-driving car v starts its journey, it can connect to its nearest BS based on signal strength and get the coordinates of RSUs and APs available in its route via Access Network Discovery and Selection Function (ANDSF) [16]. However, we consider the implementation ANDSF server to be out of the scope of this paper. In the self-driving car's request includes its geographic location, speed, and direction. Then, the ANDSF server replies with a list of all APs and RSUs available in self-driving car direction and their coverage radius.

Therefore, for communicating with macro BS $b \in \mathcal{B}$, the spectrum efficiency for each self-driving car $v \in \mathcal{V}$ is expressed as follows:

$$\gamma_v^b = \log_2\left(1 + \frac{\rho_v |G_v^b|^2}{\sigma_v^2}\right),\ \forall v \in \mathcal{V},\ b \in \mathcal{B}, \tag{1}$$

where ρ_v is the transmission power of self-driving car v, $|G_v^b|^2$ is the channel gain between self-driving car v and BS b, and σ_v^2 is the power of the Gaussian noise at self-driving car v.

The instantaneous data rate for self-driving v is given by:

$$R_v^b = a_v^b \omega_v^b \gamma_v^b, \forall v \in \mathcal{V},\ b \in \mathcal{B}, \tag{2}$$

where each self-driving car v connected to BS b is allocated a fraction a_v^b $(0 \le a_v^b \le 1)$ of bandwidth ω_v^b. We assume that the spectrum of the MNO is orthogonal so that there is no interference among self-driving cars and other users using each BS b.

2) Communication with RSU or WAP: We consider that a self-driving car v can perform handover to the WAP or RSU if WiFi or RSU can provide high data rate than BS. Therefore, before completing the handover, we use network selection mechanism described in [16], where the self-driving car can decide to which cache-enabled WAP or RSU to perform handover to before it reaches WAP or RSU area of coverage.

We consider β_v^r as distance between RSU $r \in \mathcal{R}$ and road in which the self-driving car $v \in \mathcal{V}$ goes through, and β_v^w as the distance between WAP $w \in \mathcal{W}$ and road of the self-driving car v. Furthermore, we consider d_v^r as the remaining distance for self-driving car v to reach the area covered by RSU r, while d_v^w as the remaining distance for self-driving car v to reach the area covered by WAP w. Therefore, for RSU, the estimated $\beta_v^r = g_v^r sin\alpha_v^r$, the estimated $d_v^r = g_v^r cos\alpha_v^r$, and α_v^r is the angle between the vector of movement of self-driving car v and the straight line originating from RSU r physical location. In addition, g_v^r is geo-distance between self-driving car v and RSU r, and the calculation of geo-distance is described in detail in [16]. On the other hand, for WIFI, the estimated $\beta_v^w = g_v^w sin\alpha_w^v$, the estimated $d_v^w = g_v^w cos\alpha_w^v$, where α_w^v is the angle between the vector of movement of self-driving car and the straight line originating from WAP w physical location and g_v^w is geo-distance between self-driving car v and WAP w.

Communication with RSU: For connecting to RSU, we consider that each RSU r has wireless channel with capacity $\omega_v^{r'}$. The self-driving car can use one channel at each time slot t, where the channel is shared via time-division multiplexing fashion. In each time slot t, we assume that the channel is not changing. Therefore, $\omega_v^{r'} = \omega_v^r \log_2\left(1 + \varphi_r |G_v^r|^2\right),\ \forall v \in \mathcal{V},\ r \in \mathcal{R}$, where 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.

Communication with WAP: WAP channel is shared to self -driving cars via contention-based model. Therefore, the instantaneous data rate for self-driving v via WAP w is given by:

$$R_v^w = \frac{\varphi_w \tilde{R}_v^w \xi_v^w(|\mathcal{V}_w|)}{|\mathcal{V}_w|}, \forall v \in \mathcal{V}_w,\ r \in \mathcal{R}, \tag{3}$$

where φ_w is WiFi throughput efficiency factor, and $|\mathcal{V}_w|$ is the number of self-driving cars that communicated simultaneously with WAP w, where $\mathcal{V}_w \subset \mathcal{V}$. φ_w is used to determine overhead related to MAC protocol layering such as header, DIFS, SIFS, and ACK. Furthermore, $\tilde{R}_v^w$ is the maximum theoretical data

rate that WAP can handle [17]. Furthermore, $\xi_v^w(|\mathcal{V}_w|)$ is used as decreasing function, which is a function of number of self-driving cars. $\xi_v^w(|\mathcal{V}_w|)$ is used to determine impact of contention over WiFi throughout.

For the top RSU r and WAP w on selection, self-driving car v has to be connected to one of them that satisfy the following constraint:

$$\phi_v^r + \phi_v^w + \phi_v^b = 1, \tag{4}$$

$$\phi_v^r = \begin{cases} 1, \text{ if } d_v^r \geq d_v^w, \ \frac{\beta_v^r}{\nu_r} \geq \frac{\beta_v^w}{\nu_w}, \ \omega_v^{r'} > R_v^b, \\ 0, \text{ otherwise,} \end{cases} \tag{5}$$

$$\phi_v^w = \begin{cases} 1, \text{ if } d_v^w \geq d_v^r, \ \frac{\beta_v^w}{\nu_w} \geq \frac{\beta_v^r}{\nu_r}, \ R_v^w > R_v^b, \\ 0, \text{ otherwise,} \end{cases} \tag{6}$$

where ν_r is coverage radius of RSU r, while ν_w is coverage radius of WAP w. Furthermore, if $\phi_v^w = 0$ and $\phi_v^r = 0$, the self-driving car keeps connected to BS and does not perform handover to WAP or RSU, i.e., $\phi_v^b = 1$. For downloading contents, we define q_v^b as decision variable, where q_v^b

$$q_v^b = \begin{cases} 1, \text{ if car } v \text{ downloads contents from BS } b \in \mathcal{B}, \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

Furthermore, self-driving car can perform handover to the WAP or RSU. Therefore, in order to ensure that self-driving car has at only one active connection for downloading the contents, the following constraint have to be satisfied: $q_v^b + (1 - q_v^b)(\phi_v^r + \phi_v^w) = 1$.

The transmission delay for downloading contents from MEC server to self-driving car is expressed as:

$$\tau_v^{\text{MEC}} = \frac{q_v^b \sum_{i \in \mathcal{I}_b} S(\tilde{i}_f)) + S(\tilde{i}_m)}{R_v^b} + (1 - q_v^b) \left(\phi_v^r \frac{\sum_{i \in \mathcal{I}_r} S(\tilde{i}_f)) + S(\tilde{i}_m)}{\omega_v^{r'}} + \phi_v^w \frac{\sum_{i \in \mathcal{I}_w} S(\tilde{i}_f)) + S(\tilde{i}_m)}{R_v^w} \right), \tag{8}$$

where $\mathcal{I}_b \supset \mathcal{I}$ is the set of contents downloaded via BS, $\mathcal{I}_r \supset \mathcal{I}$ is the set of contents downloaded via RSU, and $\mathcal{I}_w \supset \mathcal{I}$ is the set of contents downloaded via WAP. Furthermore, $\tilde{i}_f \in \mathcal{G}_j^{\text{female}}$ is the most requested content by female and $\tilde{i}_m \in \mathcal{G}_j^{\text{male}}$ is the most requested content by male in each cluster j.

We denote $t_v^r = \frac{2\phi_v^r \nu_r}{\mu_v}$ as a time required by self-driving car $v \in \mathcal{V}$ to leave an area covered by RSU r and $t_v^w = \frac{2\phi_v^w \nu_w}{\mu_v}$ as a time required to leave an area covered by WAP w, and μ_v as the speed of self-driving car v. When $\tau_v^{\text{MEC}} < t_v^r$ or $\tau_v^{\text{MEC}} \leq t_v^w$, the self-driving can maintain its speed μ_v because it will not affect the downloading time. However, when $\tau_v^{\text{MEC}} > t_v^r$ or $\tau_v^{\text{MEC}} > t_v^w$, the self-driving car can reduce its speed μ_v for having more time to download more contents and cache them, only if the speed reduction does not endangering other cars, perturb transport service, or break minimum speed limit.

However, if the content is not cached at the edge of the network, i.e., at MEC server, the self-driving car needs to download the content from DC via the backhaul connection. Therefore the delay from MEC server to DC is expressed as follows:

$$\tau_{\text{MEC}}^{\text{DC}} = \frac{q_v^b \sum_{i \in \mathcal{I}_b} S(\tilde{i}_f)) + S(\tilde{i}_m)}{\omega_b^{DC}} + (1 - q_v^b) \left(\phi_v^r \frac{\sum_{i \in \mathcal{I}_r} S(\tilde{i}_f)) + S(\tilde{i}_m)}{\omega_r^{DC}} + \phi_v^w \frac{\sum_{i \in \mathcal{I}_w} S(\tilde{i}_f)) + S(\tilde{i}_m)}{\omega_w^{DC}} \right). \tag{9}$$

Therefore, the total delay for downloading content $\tilde{i}_f$ and $\tilde{i}_f$ is expressed as follows: $\tau_v^{\text{Total}}(\boldsymbol{q}, \boldsymbol{\phi}) = \tau_v^{\text{MEC}} + \tau_{\text{MEC}}^{\text{DC}}$.

Each self-driving car v has WiFi Router that can provide WiFi connectivity to its passengers U_v, where WiFi channel is shared to the passengers via contention-based model described in [17]. Therefore, the instantaneous data rate for each passenger u can be calculated by applying the Eq. 3.

We define a decision variable q_u^v that indicates whether or not passenger u is connected to WiFi of self-driving v:

$$q_u^v = \begin{cases} 1, \text{ if passanger } u \text{ is connected to WiFi of car } v, \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

For each passenger $u \in \mathcal{U}_v$, the transmission delay τ_u^v for downloading content i via or from self-driving car v is defined as follows: $\tau_u^v = \frac{\sum_{i \in \mathcal{I}_r(n)} q_u^v \left(S(\tilde{i}_f)) + S(\tilde{i}_m) \right)}{R_u^v}$.

E. Caching Model

We assume that the cache storage c_v for each self-driving car i is limited. Therefore, the sizes of the recommended contents need to be downloaded and cached must satisfy the below cache resource constraint:

$$\sum_{j=1}^{k} \left(\sum_{\tilde{i}_f \in \mathcal{G}_j^{\text{female}}} o_v^{\tilde{i}_f} S(\tilde{i}_f) + \sum_{\tilde{i}_m \in \mathcal{G}_j^{\text{male}}} o_v^{\tilde{i}_m} S(\tilde{i}_m) \right) \leq c_v. \tag{11}$$

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 data $\tilde{i}_f \in \mathcal{G}_j^{\text{female}}$, where $o_v^{\tilde{i}_f}$ is given by:

$$o_v^{\tilde{i}_f} = \begin{cases} 1, \text{ if self-driving car } v \text{ caches the data } \tilde{i}_f, \\ 0, \text{ otherwise.} \end{cases} \tag{12}$$

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

$$o_v^{\tilde{i}_m} = \begin{cases} 1, \text{ if self-driving car } v \text{ caches the data } \tilde{i}_m, \\ 0, \text{ otherwise.} \end{cases} \tag{13}$$

Furthermore, in self-driving car, for further 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 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 $i \in \mathcal{I}$ requested by customer $u \in \mathcal{U}$. $h_i^{u \to v} = 1$ if the content $i \in \mathcal{I}$ is cached in the self-driving car v, and 0 otherwise.

$$h_i^{u \to v} = \begin{cases} 1, \text{ if content } i \text{ requested by consumer } u \\ \quad \text{is cached in self-driving car } v, \\ 0, \text{ otherwise.} \end{cases} \tag{14}$$

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 outputs, we consider that the MEC server caches the contents that has high probabilities of being request in its region, where cache allocation has to satisfy the following constraint:

$$\sum_{n=1}^{N} \left(\sum_{i \in \mathcal{I}_b(n)} S(i) + \sum_{i \in \mathcal{I}_r(n)} S(i) + \sum_{i \in \mathcal{I}_w(n)} S(i) \right) \leq C, \tag{15}$$

where $\mathcal{I}_b(n) \supset \mathcal{I}$ is the set of predicted contents that need to be cached in region n of BS b, $\mathcal{I}_r(n) \supset \mathcal{I}$ is the set of

predicted contents that need to be cached in region n of RSU, and $\mathcal{I}_w(n) \supset \mathcal{I}$ is the set of set of predicted contents that need to be cached in region n of WAP. Furthermore, we use $h_i^{\text{MEC}\to v} \in \{0,1\}$ to denote the cache hit indicator at MEC server for content $i \in \mathcal{I}$ requested by self-driving $v \in \mathcal{V}$. $h_i^{\text{MEC}\to v} = 1$ if the content $i \in \mathcal{I}$ is cached at MEC server, and 0 otherwise.

$$h_i^{\text{MEC}\to v} = \begin{cases} 1, \text{ if the content } i \text{ requested by self-dring} \\ \quad \text{car } v \text{ is cached at MEC server,} \\ 0, \text{ otherwise.} \end{cases} \tag{16}$$

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

F. Problem Formulation and Solution

In this work, we formulate an optimization problem that minimizes total delay experienced by passengers in downloading infotainment contents, where total delay $\tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{o})$ is given by:

$$\tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{o}) = \tau_u^v h_i^{u\to v} + (1 - h_i^{u\to v}) \left(\tau_v^{\text{MEC}} + (1 - h_i^{\text{MEC}\to v}) \tau_{\text{MEC}}^{\text{DC}} \right). \tag{17}$$

Furthermore, an optimization problem for minimizing total delay can be expressed as follows:

$$\min_{\boldsymbol{q},\boldsymbol{h},\boldsymbol{o}} \sum_{u=1}^{U} \tau_u^{\text{Tot}}(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{o}) \tag{18}$$

subject to:

$$\sum_{v=1}^{V} q_v^b a_v^b + (1 - q_v^b) \left(\phi_v^r + \phi_v^w \right) \leq 1, \tag{18a}$$

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

$$q_v^b + (1 - q_v^b) \left(\phi_v^r + \phi_v^w \right) = 1, \tag{18c}$$

$$\phi_v^r + \phi_v^w + \phi_v^b = 1, \tag{18d}$$

$$q_u^v h_i^{u\to v} + q_v^b h_i^{\text{MEC}\to v} \leq 1, \tag{18e}$$

The constraint in (18a) ensures that the communication resource allocation to all self-driving cars has to be less or equal to the total available communication resources. The constraints in (18b) ensure that caching resource allocation has to be less or equal to the available caching resources in the self-driving car. The constraints (18c) and (18d) guarantee that the self-driving car needs to have always an active connection. The constraint in (18e) ensures that the cache hit for content i happens in one location, either at the self-driving car or at MEC server.

The above-formulated problem (18) is a mixed integer problem with linear constraints which is a non-convex optimization problem. Therefore, to solve the above problem, we use a Block Successive Majorization-Minimization (BS-MM) and rounding technique. First, in majorization, we need to find a convex surrogate function which is the upper bound of (18) and relax binary variables $\boldsymbol{q}, \boldsymbol{h}$, and $\boldsymbol{o}$ in range $[0, 1]$. Second, in minimization, we need to minimize surrogate function which is a convex optimization problem. Finally, we need to enforce relaxed binary variables $\boldsymbol{q}, \boldsymbol{h}$, and $\boldsymbol{o}$ to be vectors of binary variables by applying the rounding technique. However, for the brevity of this paper, BS-MM and rounding technique to solve (18) are provided in our previous work [2], [8].

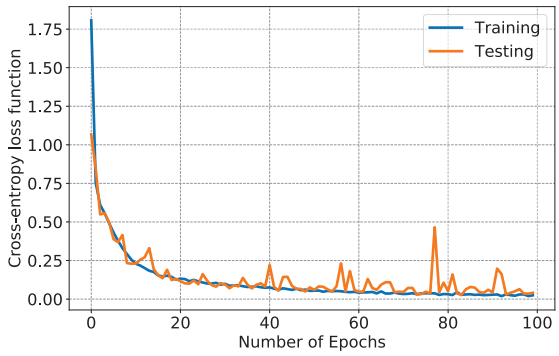

Figure 2: Minimization of cross-entropy loss function.

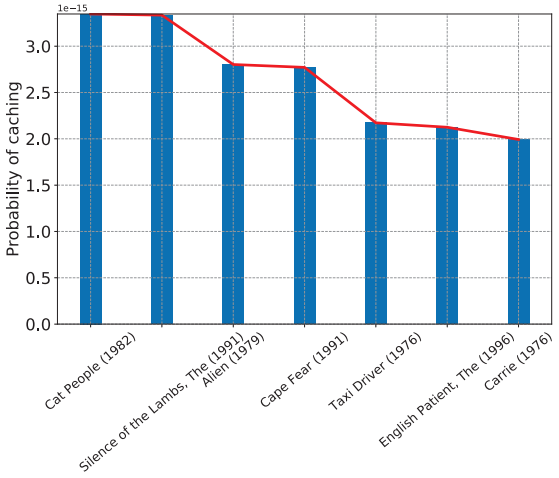

Figure 3: Top seven movies to cache at MEC (WAP 2).

IV. Performance Evaluation

For the performance valuation of our approach, we use Keras with Tensorflow [18].

Simulation Setup: At DC, to predict the probability of contents needed to be cached at MEC servers, we use MovieLens Dataset [14] for infotainment contents. In each movie i, we generate its random size $S(i)$ in the range from $S(i) = 300$ to $S(i) = 750$ Mb. Furthermore, for self-driving car $v \in \mathcal{V}$, we use one self-driving car with generated synthetic data of 37 passengers. In the route of the self-driving car, we use 6 MEC servers attached to 3 *RSU*, 2 WAPs and 1 BS. Furthermore, we set each RSU $r \in \mathcal{R}$, WAP $r \in \mathcal{R}$, and BS $b \in \mathcal{B}$ to be connected to DC with wired backhaul of capacity $\omega_r^{DC} = 60$, $\omega_w^{DC} = 60$, and $\omega_b^{DC} = 120$Mbps, respectively. We consider that each RSU $r \in \mathcal{R}$, WAP $w \in \mathcal{W}$, and BS $b \in \mathcal{B}$ have bandwidth of $\omega_v^r = 20$, $\omega_v^w = 160$, and $\omega_v^b = 20$ MHz respectively. In addition, each MEC server has a cache capacity of 100 terabytes (TB), while the self-driving car has a bandwidth of 160 MHz with maximum theoretical data rate of $\tilde{R}_u^v = 3466.8$ Mbps and cache capacity $c_v = 100$ TB.

Evaluation Results: For predicting the probabilities of the contents to be cached in the areas of RSUs, WAPs, BS, we use MLP of three hidden layers, where the last later has 6 neurons which corresponds to 6 geographical locations of MEC servers. Fig. 2 shows the cross-entropy loss function minimization. Our prediction reaches 99.28% accuracy. Furthermore, Fig. 3 shows

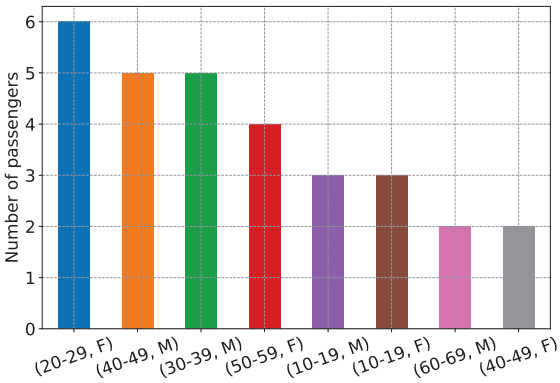

Figure 4: Passengers in the self-driving car.

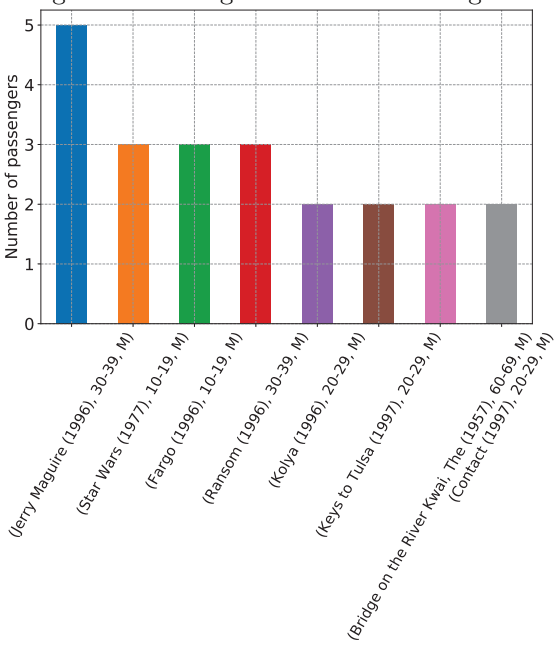

Figure 5: Top 8 movies to cache in the self-driving car.

top seven movies need to be cached in geographical location of WAP 2. The WAP 2 can cache more movies, but here we show the top seven movies to cache based on predicted probability value.

Fig. 4 demonstrates age and gender-based clustering for the passengers in self-driving cars. In addition, based on passengers' features in the self-driving car, Fig 5 shows the contents need to be downloaded from MEC servers and cached in the self-driving car. Furthermore, in a self-driving car, we have a large cache size of 100 TB that can cache more than 133 movies, but for brevity in Fig 5, we only present top eight contents. Each MEC server and self-driving car downloads and caches contents in descending order of predicted probabilistic values until cache storage is full or no more contents need to be cached. However, when content needs to be replaced Least Frequently Used (LFU) or Price Based Cache Replacement Policy (PBCR) can be utilized [9].

V. Conclusion

In this paper, we proposed caching for infotainment contents in self-driving cars and in close proximity to the self-driving cars at MEC servers. In our proposal, MEC servers deployed at RSUs, WAPs, and BSS are used to cache contents which have high predicted probabilistic values for being requested in their areas. Furthermore, based on passengers' features learned via CNN, the self-driving car downloads from MEC servers and caches the infotainment contents that are appropriate to its passengers. The simulation results demonstrate that our approach can be easily implemented in the self-driving car and MEC.

References

[1] BUSBUD, "Will driverless buses be a reality?" https://www.busbud.com/blog/will-driverless-buses-reality/, [Online; accessed August. 11, 2018].

[2] A. Ndikumana, N. H. Tran, and C. S. Hong, "Deep learning based caching for self-driving car in multi-access edge computing," *arXiv preprint arXiv:1810.01548*, 2018.

[3] 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.

[4] Next Analytics, "YouTube video appeal demographics," https://www.nextanalytics.com/excel-youtube-analytic-insights-and-data-mining/page/4/, [Online; accessed Nov. 11, 2018].

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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 *Proceedings of IEEE Globecom Workshops (GC Wkshps)*, 4-8 Dec. 2017 (Singapore), pp. 1–6.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Keras, "Keras: The Python Deep Learning library," https://keras.io/, [Online; accessed Nov. 13, 2018].