FedABR: A Personalized Federated Reinforcement Learning Approach for Adaptive Video Streaming

Yeting Xu[1], Xiang Li[1], Yi Yang[1], Zhenjie Lin[2], Liming Wang[2], Wenzhong Li[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China
[2]China Southern Power Grid Digital Platform Technology Company, Shenzhen, 518053, China
MF20330097@smail.nju.edu.com

***Abstract*—Modern video streaming applications apply adaptive bitrate (ABR) algorithms to enhance user quality of experience (QoE). The existing model-based ABR algorithms failed to generalize to diverse network conditions and personalized QoE objectives due to their fixed control rules. The learning-based ABR algorithms required significant tuning to learn a well-performed model which can cause a QoE degradation during the model testing phase. In this paper, we propose FedABR, a novel ABR algorithm based on personalized federated learning to address the above challenges. To enable clients' local model dealing with network environment changes, we introduce a federated learning approach to train a global model using all the clients' local model without gathering their data together to protect clients' privacy. We also introduced an adaptation phase to train a personalized model for each client to maximize their individual QoE. By jointly training multiple learning tasks with a global model, it has the ability to provide transferable knowledge to supervise bitrate selection, and can be efficiently adapted to a new task in unseen environment with much fewer data samples and training epochs. We implement the proposed FedABR based on an emulation platform which connects to the Linux network protocol stack through a virtual network interface to send real data packets for evaluation. Extensive experiments based on real-world traces show that FedABR achieves the best comprehensive QoE compared with the state-of-the-art ABR algorithms in a variety of network environments.**

***Index Terms*—Federated learning, Adaptive video streaming**

I. Introduction

With the development of global network infrastructure and Internet devices, more and more video streaming applications appeared in recent years. Internet users can watch whatever they want anytime through these Video on demand (VoD) services, which contributes to the rapid increasing of video streaming traffic and the steady rise of user's demands on video quality. According to previous study [1], Internet users have less and less patience with startup delay, rebuffering time and low video quality when watching video. A small amount of interruptions can decrease a great amount of average video play time of a consumer, which is unacceptable for video content providers. To win customers and make them watch videos as much time as possible, content providers are making a great effort to provide high-quality video streaming services to their users.

HTTP adaptive streaming (HAS) is the dominant approach for video streaming. It chunks the media content into small pieces and delivers them like regular web content over HTTP protocal. There are a plenty of commercial solutions for HAS such as Microsoft's Smooth Streaming, Apple's HTTP Live Streaming (HLS), Adobe's HTTP Dynamic Streaming (HDS) and several open-source solutions. Dynamic Adaptive Streaming over HTTP (DASH) [2] is the first adaptive bitrate HTTP-based streaming solution that is an international standard. In DASH system, a multimedia file is partitioned into one or more segments and the segment information are described in a media presentation description (MPD). Each segment has one or more representations that represent versions at different resolutions or bit rates. DASH leaves implementation of Adaptive bitrate (ABR) algorithms to the third parties. Adaptive bitrate (ABR) algorithm is the essential components that content providers use to optimize users' quality of experience (QoE). It dynamically selects a bitrate for each multimedia segment on client-side video players based on the network conditions, device capabilities, and user preferences. However, selecting the appropriate bitrate in dynamic network is challenging because of the limited network bandwidth and trade-off of conflicting video QoE requirements [3].

ABR algorithm has been studied for a long time and many works have been proposed. Some works [4]–[6] builded mathematical models for network conditions based on past network bandwidth and made bitrate decisions based on the estimated network throughput. Some works [7], [8] selected bitrate only based on user playback buffer occupancy and didn't take network conditions into consideration. These algorithms failed to achieve optimal performance for they didn't use all the useful information to make bitrate decisions. MPC [9] developed a model predictive control algorithm that combined both network throughput estimates and buffer occupancy information to select bitrates to maximize QoE over a horizon of several future chunks. However, the fixed control rules makes MPC unable to adapt to a broad set of network conditions that exist in real world.

Recent works [10]–[12] consider to apply machine-learning algorithms to generate ABR policy. The machine-learning based ABR algorithms use raw observations (e.g., throughput

This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61972196, 61832008, 61832005), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Sino-German Institutes of Social Computing. The corresponding author is Wenzhong Li (lwz@nju.edu.cn).

ISBN 978-3-903176-57-7© 2023 IFIP

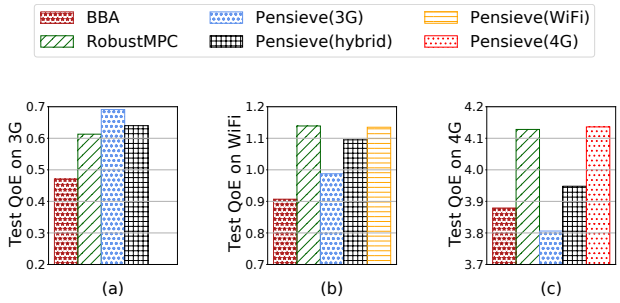


Fig. 1. The performance of Pensieve [11] DRL agents trained with different datasets, where hybrid means the dataset combining 3G, WiFi and 4G traces (details are found in Sec. 5).

samples, playback buffer occupancy, video chunk sizes) as neural network inputs, and the output is the predicted bandwidth or bitrate for the next video chunk. For example, the works of [10], [11], [13] used Deep Reinforcement Learning (DRL) that trains an agent by interacting with the environment to select the right bitrate. Fugu [12] used a neural network to train a network throughput predictor to make a more precised bandwidth prediction.

Despite the flexibility and effectiveness of the DRL-based ABR algorithms, there remain a variety of challenges to apply them for real-world video streaming systems. On the one hand, collecting training data in video streaming sessions is difficult and privacy sensitive. It requires the users to upload their video watching information from various network environments to a central server for DRL model training, which will raise users' serious privacy concerns. On the other hand, due to the complexity and diversity of network conditions, training a uniform DRL model for heterogeneous clients is infeasible and hard to deal with user behaviors heavy-tailed changing with time.

We demonstrated the limitations of the DRL-based ABR algorithm called Pensieve [11] through experiments. First, we trained four Pensieve models using the 3G, 4G, WiFi, and a hybrid dataset (which combines training samples form 3G+4G+WiFi datasets), denoted by $Pensieve(3G)$, $Pensieve(4G)$, $Pensieve(WiFi)$, and $Pensieve(hybrid)$ respectively. The datasets are described in detail in Sec 5. Figure1(a), Figure1(b), and Figure1(c) show the performance of Pensieve and two model-based ABR algorithms called BBA [8] and RobustMPC [9] testing in 3G, 4G and WiFi network environments respectively. From Figure1(a), we can see that $Pensieve(3G)$ outperforms other ABR algorithms in 3G environment, which means that the DRL-based ABR algorithm is customized for a specific environment. $Pensieve(hybrid)$ performs worse than $Pensieve(4G)$ and $Pensieve(WiFi)$ in 4G and WiFi environments, which means mixing the training data samples cannot improve performance or generality. Also, we tested the performance of $Pensieve(3G)$ in 4G and WiFi network environments. As can be seen in Figure1(b), $Pensieve(3G)$ performs worse than the model-based algorithms if transferring to unseen environment. In Figure1(c), $Pensieve(3G)$ performs the worst. These results show that a personalized Pensieve model can perform better than a uniform model in the corresponding environment. However, if the network environment changes to an unseen scenario, the specialized model can lead to severe performance degradation even worse than that of the simple heuristic.

In this paper, we propose FedABR, a novel ABR algorithm based on personalized federated-learning to address the above challenges. To enable model training without privacy leakage, we introduced a federated-learning method to train a global model across all participating clients without uploading the personal data to the central server to protect users' privacy. To deal with the heavy-tail nature of network environments and user behaviors, we propose to train a personalized local model for individual client. Specifically, we treat the participating clients as a number of learning tasks that learn ABR policies on different network environments (e.g., WiFi, 4G and Ethernet). Unlike conventional DRL methods that train the tasks separately, the proposed federated learning approach trains all the tasks jointly to learn a global model, and adapt the global policy to the local environment at the adaptation phase with a few data to get a personalized model (local model in FL). With the proposed personalized federated learning, the heavy-tail nature of network environment and user behavior can be addressed with local training, and the training of new clients can be accelerated by downloading the global model from the FL server and adapting it to the local environment with a few data samples.

The contribution of our work are summarized as follows.

1) We proposed a novel federated reinforcement-learning based adaptive bitrate selection algorithm. We discuss the weakness of DRL-based ABR algorithms that makes it challenging to deploy them in real-world video streaming systems.
2) We proposed an adaptation phase to train a personalized model for each client to maximize clients' QoE. Our study indicates that a personalized model can achieve higher performances compared with global model in different network environments.
3) We implement the proposed FedABR based on an emulation platform which connects to the Linux network protocol stack through a virtual network interface to send real data packets for evaluation. Extensive experiments based on real-world traces show that FedABR achieves the best comprehensive QoE compared with the state-of-the-art ABR algorithms in a variety of network environments.

II. Related Work

We summarize the related works in the aspects of ABR algorithms for video streaming and personalized federated learning.

A. ABR Algorithms for Video Streaming

There has been a lot of recent work on ABR algorithms that use bandwidth estimation or playback buffer occupancy

or both of them to make ABR decisions. The model-based methods builded mathematical models to describe network conditions to make ABR decisions. FESTIVE [6] used the harmonic mean of download speed over recent chunks to predict the throughput and proposed a stateful bitrate selection to compensate for the biased interaction between bitrate and estimated bandwidth, and tried to find a tradeoff between efficiency and fairness when there is competition between clients. BBA [8] was a buffer-based approach. It simply picked a bitrate based on playback buffer occupation, which causes the bitrate to change frequently during long-term bandwidth fluctuations resulting in QoE degradation. Buffer Occupancy based Lyapunov Algorithm (BOLA) [7] used an online control algorithm to maximize users' QoE adapting to network changes. It considered average bitrate and rebuffering time as a measure of QoE. MPC [9] proposed a model predictive control algorithm that can optimally combine throughput and buffer occupancy information to outperform traditional approaches, and provided a flexible QoE model which is widely used in subsequent studies. Oboe [14] pre-computes a config map that maps different network conditions to different ABR algorithms, then dynamically adapts ABR algorithms at run-time for the current network conditions.

Though model-based algorithms improved users' QoE, they failed to achieve optimal performance across a broad set of network conditions and QoE objectives because of their fixed control rules. The learning-based methods [11]–[13], [15], [16] were proposed to learn personalized ABR strategies for various conditions. Pensieve [11] generated an ABR algorithm using Deep Reinforcement Learning (DRL). It does not rely on pre-programmed models but learns to make ABR decisions solely through observations (i.e., throughput estimation and buffer occupancy) of the resulting performance of past decisions. DRL algorithms trains agent through trial-and-error, resulting in low sample efficiency. Comyco [13] trained an ABR policy via imitating expert trajectories to avoid redundant exploration. Stick [10] used DRL to adjust the hyperparameters of the traditional buffer-based method BBA [8] to maximize users' QoE. Fugu [12] argued that in the real world, it was difficult for complex or machine learning control schemes to outperform simple schemes (i.e., BBA [8]). It used supervised learning in situ to train a probabilistic predictor of upcoming chunk transmission times, then informed the predicting time to improve MPC [9] with a harmonic mean based throughput prediction scheme.

B. Personalized Federated Learning

Federated learning is a distributed machine learning architecture, in which data is stored on distributed local devices instead of central servers [17]. Generally, federated learning aims to aggregate multiple users' experiences while preserving privacy, and also to reduce communication overhead [18], [19].

For devices whose data is non-IID distribution, global model may not perform better than the local models that trained with their local private data. To improve the performance for individual clients, several personalized federated learning (PFL) techniques has been proposed. The PFL approaches are basically divided into two types: data-based and model-based approaches. Data-based approaches include data augmentation and client selection. The idea is to smooth the statistical heterogeneity through sampling techniques to either data or client [20], [21]. Model-based personalized approaches aim to enable FL models to adapt to the diverse data distributions among clients [22]. In federated learning, model regularization can be applied to achieve convergence stability and improve model generalization. FedProx [23] not only averages the parameters of each local model, but also adds a penalty term to reduce the impact of the data distribution difference of local training samples. One personalization method is meta-learning. Inspired by Model-Agnostic Meta-Learning (MAML) [24], Per-FedAVG [25] learns an initial meta-model that can perform well after one more gradient update. Other personalization methods include multi-task learning and transfer learning. There are also some works [26], [27] use transfer learning to train personalized model by integrating global model and local models.

III. Problem Formulation

Reinforcement learning is a process in which an agent learns to make decisions through trial and error. This problem is often modeled mathematically as a Markov decision process (MDP), where an agent at every timestep is in a state s, takes action a, receives a scalar reward r and transitions to the next state s' according to environment dynamics $p(s'|s,a)$. In this work, we consider ABR algorithm as a Markov decision process (MDP), which can be solved using deep reinforcement learning (DRL).

A. ABR as a DRL Problem

We use the A3C [28] method, an actor-critic method as the basic DRL algorithm in our system. The actor-critic algorithm involves training two deep neural networks (DNN), namely the actor-network and the critic-network, which refer to the policy function and the value function respectively. The policy function $\pi(a|s)$ generates the expected candidate action a for the current state s, and the value function $V(s)$ approximates the expected reward of the current state s. We use $\pi_\theta(s_t, a_t)$ to represent the policy function with parameters θ, which is refer to as policy parameters. We represent the critic network with parameters θ_v, and denote the value function by $V_{\theta_v}(s_t)$.

For ABR algorithm, the state is the observations of the network environment and video playback status. The observations include: the network throughput measurements for the past k video chunks $\vec{x_t}$; the download time of the past k video chunks $\vec{\tau_t}$, which also represents the time interval of the throughput measurements; the available sizes for the next video chunk $\vec{n_t}$; the current buffer level b_t; the number of chunks remaining in the video c_t; the bitrate at which the last chunk was downloaded l_t. Thus state $s_t = (\vec{x_t}; \vec{\tau_t}; \vec{n_t}; b_t; c_t; l_t)$.

The agent determine the bitrate for the next video chunk from a set of candidate bitrates through policy function

$\pi_\theta(a|s)$. $\pi_\theta(a|s)$ is defined as a probability distribution over actions $\pi_\theta : \pi_\theta(s_t, a_t) \rightarrow [0, 1]$, where $\pi_\theta(s_t, a_t)$ is the probability that action a_t is taken in state s_t. Usually, we choose the action a with largest probability. After applying the action, the environment provides the learning agent with a reward r_t for that chunk. Here r_t is calculated using a comprehensive QoE metric introduced in section 5.

B. Policy Gradient Training

The policy network is trained using a policy gradient method [29], which optimizes a performance objective by finding a good policy to variants of stochastic gradient ascent with respect to the policy parameters [30]. The update rule [31] of the policy network is as follows:

$$\theta \leftarrow \theta + \alpha(R_t - V(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t) + \beta\nabla_\theta H(\pi_\theta(\cdot|s_t)), \quad (1)$$

where $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ is the expected discounted reward at time t and α is the learning rate; γ is a factor discounting future rewards; $H(\cdot)$ is an entropy term that encourages diversity. The parameter β is set to a large value at the start of training to encourage exploration and decreases over time to emphasize improving rewards [28].

The aim of the critic network is to minimize the Bellman error:

$$\theta_v \leftarrow \theta_v - \alpha'\nabla_{\theta_v}(r_t + \gamma V(s_{t+1}; \theta_v) - V(s_t; \theta_v))^2, \quad (2)$$

where α' is the learning rate.

IV. Federated Reinforcement Learning based ABR Method

A. FedABR Framework

The overall framework of FedABR is illustrated in Figure 2. The central server first initialises the model and distributes it to the clients. The clients train their local model with their own local data for several epochs and send the trained model back to central server. The central server then aggregates the model to get a global model. The central server then send the global model to each clients for next round's local training. When the global model is trained to convergence, it can perform well on each type of network environment. If a new user starts to watch the video, it can download the well-trained global model first, and won't encounter the QoE degradation at the bootstrapping phase. By utilizing federated learning, we can average the parameter of the models trained in each user's device to get a global model without gathering the data together to protect users' privacy, while the global model is equivalent to the model trained with data gathered from all the users' devices [17].

B. Federated Reinforcement Learning for Model Aggregation

Since the Internet is complex and diverse, individual client only observes a noisy piece of the system dynamics, and its behavior is often heavy-tailed and changed with time. Different clients exhibit different levels of network dynamics, and different network dynamics are possible to happen to

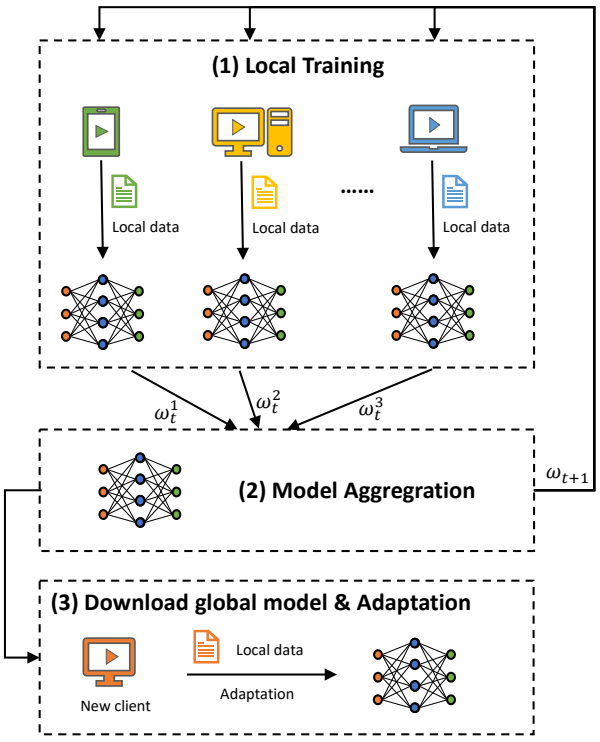


Fig. 2. The framework of FedABR.

the same client when it moves and changes from different network connections. To make use of all the available data and build an efficient model so that to have good performance in all kinds of network environments, we introduce a federated learning approach in our system. We use the classical federated learning method FedAvg [17] to conduct model aggregation. Specifically, suppose we have a total number of K users, and each user has its neural network model denoted with ω^k. The aggregate phase of FedAvg algorithm is as follow:

$$\omega = \frac{1}{K}\sum_{k=1}^{K} \frac{n_k}{n}\omega^k, \quad (3)$$

where n_k is the number of local data for user k and $n = \sum_{k=1}^{K} n_k$.

C. Personalized Federated RL Algorithm

As mentioned above, each client in federated learning can be viewed as a task to learn ABR policy in a particular network environment. Experiment results in Fig. 1 showed that a uniform Pensieve model trained in a combined dataset including 3G, 4G and WiFi network traces failed to perform better than a personalized Pensieve model. The goal of federated learning is to learn a global model that can perform well on all kinds of network environments. Thus, to better improve the performance of the global model, a personalize phase is needed.

A common personalization method is local finetune. After learning a converged global model, the client can perform

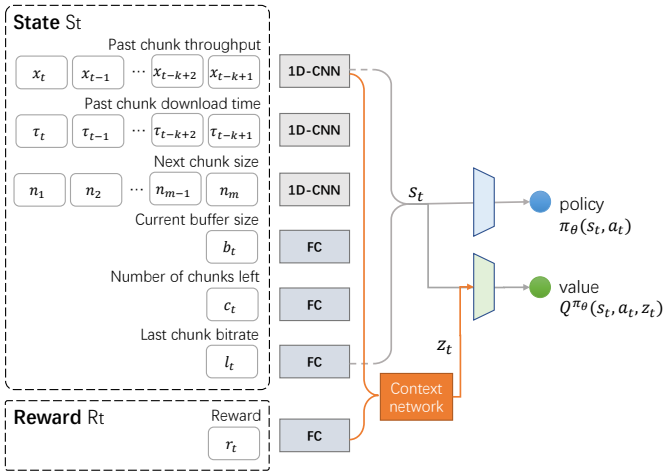


Fig. 3. The neural network architecture of FedABR.

several rounds of local SGD on top of the global model using its local data. In addition to local finetune, we added a context network to the conventional actor-critic networks [32], [33] that only train an actor neural network and a critic neural network. The input of the context network is an episode

$$\tau_t = (s_{i,t-k}, a_{i,t-k}, r_{i,t-k}, ..., s_{i,t}, a_{i,t}, r_{i,t})$$

that is collected in the past k trials when actor interacts with the environment at time t. The output of the context network is a low-dimensional embedding vector z_t. The τ records the state transitions of the past k trials and the rewards under different states and actions. Through τ, we can infer the dynamics and reward function of the current environment and abstract some specific characterization about it. Thus z_t is a latent representation of the current network environment and can capture the characteristics of different learning tasks. Since τ is a sequence of triplets (s_t, a_t, r_t), we design the context network as RNN and implement it with Long-Short Term Memory (LSTM), whch is denoted as θ_c.

The proposed neural network architecture of FedABR is illustrated in Fig. 3. FedABR passes the past chunk throughput, next chunk sizes and past chunk donwload time to three 1D convolution layer (CNN) respectively. Results from these layers are then concatenated with other inputs as a tensor s_t. The policy network then takes the state s_t as input into a hidden layer, while the value network takes the concatenation of s_t and z_t as input. The context network takes the concatenation of the state s_t and reward r_t as input into a recurrent neural network (RNN) layer to produce an embedding z_t.

We train the context networks to learn a latent representation of the task. All value functions $\theta_v(s_t, a_t)$ are conditioned on the context and implemented as $\theta_v(s_t, a_t, z_t)$. The update rules for the policy network and the critic network with context network are as follows:

$$\theta \leftarrow \theta + \alpha(R_t - V(s_t, z_t))\nabla_\theta \log \pi_\theta(a_t|s_t, z_t) + \beta\nabla_\theta H(\pi_\theta(\cdot|s_t, z_t)), \quad (4)$$

Algorithm 1 Federated-learning based ABR algorithm

{// Run on the server}
ServerUpdate:
Initialize global model ω_0
for each round $t = 1, 2, \dots$ **do**
  Distribute ω_t to the sampled clients.
  for each client $c \in C$ in parallel **do**
    $\omega_t^c \leftarrow$ **ClientUpdate**(ω_t)
  end for
  Update algorithm parameters $\omega_{t+1} \leftarrow \frac{1}{|C|}\sum_{c\in C}\frac{n_c}{n}\omega_t^c$
end for

{// Run on client c}
ClientUpdate(c, ω):
for each local epoch from 1 to E **do**
  Sample an episode τ using ω
  Update ω using Eqn. 4 and Eqn. 5
end for
Return ω to server

$$\theta_v, \theta_c \leftarrow argmin_{\theta_v,\theta_c}(r_t + \gamma V(s_{t+1}, z_{t+1}; \theta_v, \theta_c) - V(s_t, z_t; \theta_v, \theta_c))^2. \quad (5)$$

When a new client joins or when the client's network environment changes, the context network can detect the change from the latest episode τ and thus output a new embedding vector z, which affects the critic's output of expected reward of the current state on the current task and thus guides the update of the actor. The pseudo-code of the FedABR algorithm is described in Algorithm 1. The local finetune process is the same as ClientUpdate in Algorithm 1.

V. Evaluation

In this section, we conduct extensive experiments to evaluate the performance of the proposed FedABR method.

A. Experiment Setup

Evaluation Platform. We conduct the experiments on a PC (CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz; Memory: 32GB DDR4 2400Mhz*4; OS: 64-bit Ubuntu 16.04). We use Google Chrome and chrome driver as client video player and deployed a video server using Apache. We use Mahimahi [34] to emulate network environments from the network traces between the client and the server with 80ms RTT. We play the video on dash.js and the player is configured to have a playback buffer capacity of 60 seconds. We train FedABR use the "Envivio-Dash3" video of the DASH-246 JavaScript reference client. The video has a total length of 193 seconds and is divided into 48 chunks, each chunk represents approximately 4 seconds of video playback. In addition, the video is encoded by the H.264 codec at bitrates in the range of 300,750,1200,1850,2850,4300kbps respectively.

Network Traces. We generated a set of network traces in different network environments to train and test the ABR algorithms. The traces are generated from real-world network environment datasets: HSDPA [35] is a 3G network trace

TABLE I
DATASETS STATISTICS.

Dataset	Time Granularity	Average Bitrate (Mbps)	Scenario
3G	10s	0.93 ± 0.18	Video streams on public transportation.
WiFi	5s	1.51 ± 0.53	Web browsing with WiFi
4G	10s	8.77 ± 0.66	Speed test APP

dataset with a throughput scope of $0.1 \sim 1$Mbps; FCC [36] is a WiFi network trace dataset with a throughput scope of $0.5 \sim 6$Mbps; Sydney [37] is a 4G network trace with a throughput scope of $5 \sim 10$Mbps. For each network environment dataset listed above, we generated 1000 traces. Each trace has a duration of 320 seconds. Each type of traces are partitioned into training set and testing set, where 800 traces form the training set and 200 traces form the testing set. We use 3G, WiFi and 4G traces to indicate each type of traces for convenience. Moreover, we combined the training set of 3G, WiFi and 4G traces as a hybird dataset to simulate real world scenario where users dynamically change their network environments.

The basic information of the datasets are listed in Table I.

QoE Metrics. Video consumers have different preferences for video streaming QoE. Some may want better video quality, while others want less rebuffering time or better video quality smoothness. However, these preferences are conflicting from each other. ABR algorithms need to find the best trade-off for different user preferences. We use the widely used QoE metric introduced in [9] to evaluate ABR algorithms:

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \tau \sum_{n=1}^{N} |q(R_{n+1}) - q(R_n)| \tag{6}$$

where N is the total number of chunks of current video; R_n represents the video bitrate of each chunk n; T_n represents the rebuffering time for each chunk n; $q(R_{n,v})$ is a function that maps the bitrate R_n to the video quality perceived by the user; μ, τ are non-negative weighting parameters corresponding to rebuffering time and video quality smoothness respectively. A relatively small μ indicates that the user is not particularly concerned about rebuffering time.

To evaluate ABR algorithms for different user preferences, we considered three different QoE metrics, each of them has different choices of the combination of $q(R_n)$, μ and τ.

- QoE_{std} : $q(R_n) = R_n, \mu = 4.3, \tau = 1$. This is the standard QoE metric used in MPC [9] and Pensieve [11].
- QoE_{fluent} : $q(R_n) = R_n, \mu = 8, \tau = 1$. This metric favors fluent video where longer rebuffering time will result in lower video QoE.
- QoE_{hd} : $\mu = 4.3, \tau = 1$. This metric favors high-definition (HD) video where HD bitrates have significantly higher quality scores than non-HD bitrates. It maps video bitrate to $q(R_n)$ where $0.3 \to 1, 0.75 \to 2, 1.2 \to 3, 1.85 \to 12, 2.85 \to 15, 4.3 \to 20$.

Baseline Algorithms. We compare FedABR with three state-of-the-art ABR algorithms:

- BBA [8]: a buffer-based ABR algorithm that selects bitrates based on playback buffer occupation. In this paper, we set the buffer bound $B = 5, 10$ as suggested by the authors.
- RobustMPC [9]: a model predictive control ABR algorithm that combines both throughput estimates and buffers occupancy information to select bitrates.
- Pensieve [11]: a DRL-based ABR algorithm. Pensieve trains a neural network model that selects bitrates for future video chunks based on observations collected by client video players.

Implementation We setup 3 clients that play video and train local model in 3G, 4G and WiFi network environment respectively. Within a communication round, each client first downloads the parameters of the global model from the central server as the initial parameters of the local reinforcement learning model, and then updates the local model for 10 times with a batch size of 48, which means the length of an episode is 48. Then the client uploads the current local model to the central server for global aggregation to obtain a new global model. For adaptation phase, we train the global model with local data for 100 times to get a personalized local model. The network architecture is as Fig 3. Following the parameter settings of Pensieve [11], FedABR adopts $k = 8$ past bandwidth measurements, and passes the measurements and next chunk size to a 1D convolution layer (CNN) with 128 filters, each of size 4 with stride 1. Then, aggregates the results from these layers with other inputs in a hidden layer that uses 128 neurons to apply the softmax function. The critic network uses the same neural network structure, but its final output is a linear neuron (with no activation function). The output of context network is a vector with a length of 128. During training, we use a discount factor $\gamma = 0.99$, learning rates for the actor, critic and context network are configured to be $5 * 10^{-5}$, $5 * 10^{-4}$ and $5 * 10^{-3}$, respectively. We keep all these hyper-parameters fixed throughout our experiments. We implemented this architecture using TensorFlow [38].

B. Comparison with Baseline Algorithms

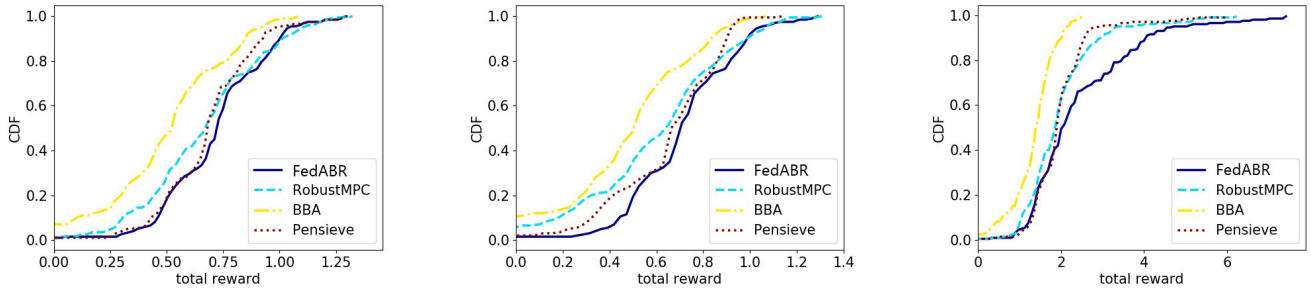
In this section, we compare the performance of the proposed FedABR with the baseline algorithms on different networks. Fig. 4, Fig. 5, and Fig. 6 provide results of each ABR algorithms in the form of Cumulative Distribution Functions (CDFs) for each QoE metric and network environments. The detailed results are shown in Table II. There are two key takeaways from these results.

First, it is shown that our algorithm either matches or exceeds the performance of the best existing ABR algorithm on each QoE metric and network environments. The closest competing scheme is Pensieve; this proves that our method can have superior performance compared to the other learning-based ABR methods. For QoE_{std}, a widely considered metric in [9], [11], the average QoE for our system is 5% higher than that of Pensieve on the WiFi traces, and $3\% \sim 15\%$ higher

TABLE II

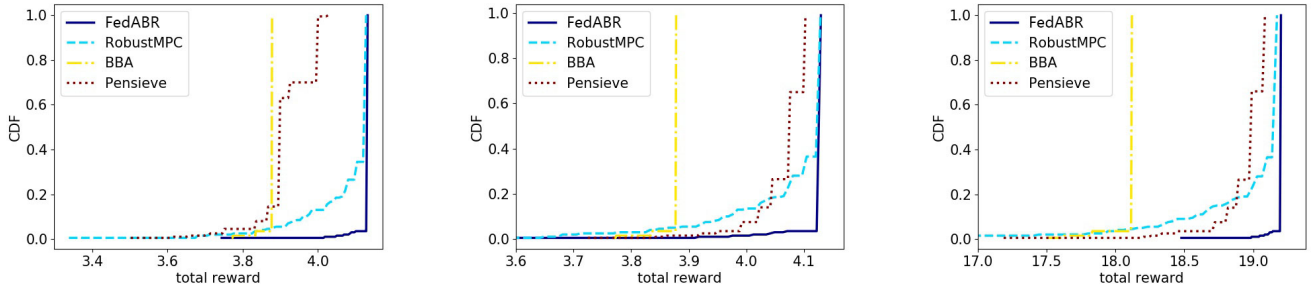
COMPARISON OF AVERAGE BITRATE (MBPS), REBUFFERING TIME (SECOND), BITRATE VARIATION, AND THEIR CORRESPONDING QOE METRICS ON DIFFERENT NETWORK ENVIRONMENTS.

Trace	Method	Bitrate	Reb.	Var.	QoE_{std}	Bitrate	Reb.	Var.	QoE_{fluent}	Bitrate	Reb.	Var.	QoE_{hd}
3G	FedABR	0.823	**0.002**	**0.067**	**0.731**	0.781	**0.002**	**0.041**	**0.724**	0.78	**0.001**	**0.041**	**3.278**
	Pensieve	0.841	0.012	0.071	0.695	0.749	0.003	0.064	0.66	0.787	0.002	0.086	3.187
	BBA	0.826	0.01	0.275	0.508	0.825	0.01	0.275	0.471	0.826	0.01	0.275	1.409
	RobustMPC	**0.849**	0.02	0.084	0.68	**0.846**	0.019	0.086	0.613	**0.848**	0.019	0.085	2.035
WiFi	FedABR	1.076	**0.002**	**0.076**	**0.992**	1.076	**0.002**	**0.076**	**0.985**	1.123	**0.0009**	**0.054**	**4.096**
	Pensieve	1.132	0.014	0.095	0.979	1.132	0.014	0.095	0.928	1.091	0.0073	0.085	3.70
	BBA	1.153	0.008	0.334	0.787	1.153	0.008	0.334	0.759	1.153	0.0076	0.334	3.217
	RobustMPC	**1.184**	0.026	0.104	0.968	**1.182**	0.027	0.105	0.864	**1.184**	0.0267	0.105	3.923
4G	FedABR	**4.21**	0.0009	**0.074**	**4.096**	**4.21**	0.0009	**0.073**	**4.129**	4.143	**0**	0.074	18.972
	Pensieve	4.007	**0**	0.088	3.70	4.143	**0**	0.074	4.069	**4.21**	0.0009	**0.073**	**19.201**
	BBA	3.97	**0**	0.093	3.217	3.969	**0**	0.093	3.877	3.97	**0**	0.093	18.111
	RobustMPC	4.168	**0**	0.078	3.922	4.141	**0**	0.079	4.063	4.116	**0**	0.079	19.018



(a) QoE_{std} (b) QoE_{fluent} (c) QoE_{hd}

Fig. 4. QoE Comparison with existing ABR algorithms in 3G network environment.



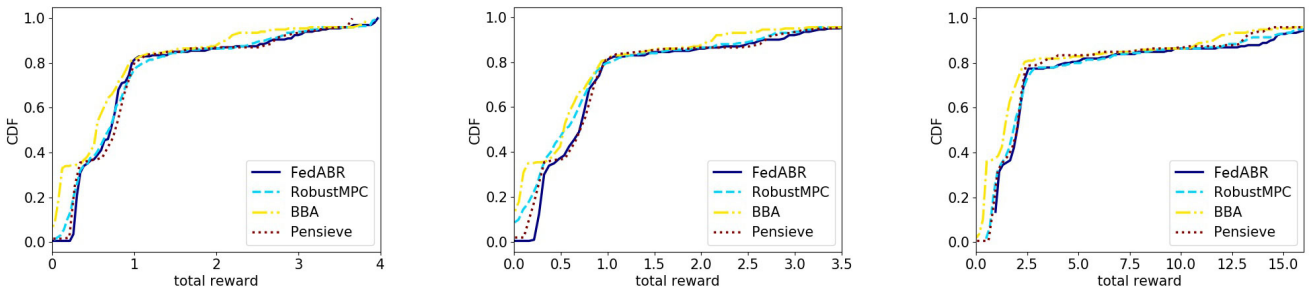
(a) QoE_{std} (b) QoE_{fluent} (c) QoE_{hd}

Fig. 5. QoE Comparison with existing ABR algorithms in 4G network environment.

on other network traces. The gaps between FedABR and other methods are also found in QoE_{fluent} and QoE_{hd}.

Second, it is observed that the existing model-based ABR algorithms struggle to optimize the performance on different QoE objectives. The reason is that these algorithms employ fixed control laws, even though optimizing for different QoE objectives requires different ABR policies. Even Pensieve trains different models for each QoE objective. Our proposed FedABR can automatically learn these policies with personalized federated learning to adapt different QoE objectives, thus the performance of our system maintains consistently high in all conditions.

Model transfer to unseen scenario. We then compare the performance of the FedABR model with a local Pensieve model trained on the 3G training data to imitate a scenario that a client has only been watching videos in a 3G network and have no experience of streaming videos in Wifi and 4G networks. We apply the pretrained models to the WiFi and 4G test data. Figure 7(a) shows the results of the FedABR model and the Pensieve model. According to the figure, both our system and Pensieve perform well in the 3G network. After applying to other unseen networks, Pensieve performs even worse than the model-based approaches BBA and RobustMPC, while our system still achieves the best performance among



(a) QoE_{std} (b) QoE_{fluent} (c) QoE_{hd}

Fig. 6. QoE Comparison with existing ABR algorithms in WiFi network environment.

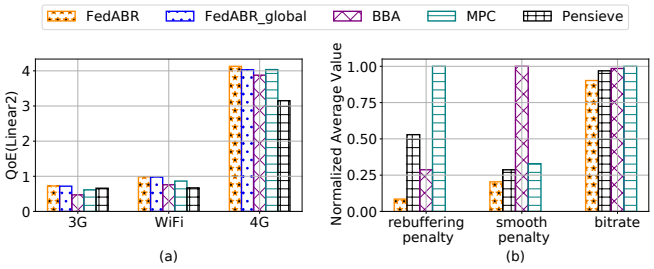


Fig. 7. (a) Testing FedABR on unseen scenarios. (b) Comparison of different QoE metrics for different ABR algorithms.

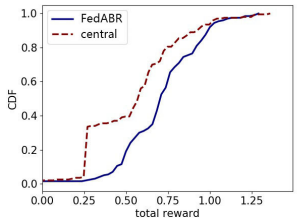


Fig. 8. Comparing FedABR with the centrally-traied model with insufficient training data.

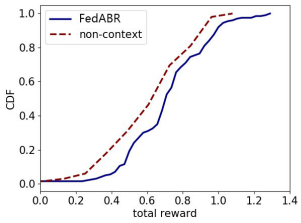


Fig. 9. Comparison of context and non-context models.

all the scenarios.

Trade off between QoE metrics. We study the trade-off between different conflicting QoE metrics, and the normalized results are shown in Fig. 7(b). It is shown that RobustMPC achieves the highest bitrate and the largest rebuffering time. BBA has the second high bitrate and has the largest smooth penalty. Pensieve has a modest bitrate, rebuffering time, and variation. Compared to other algorithms, our system achieves the lowest rebuffering time and smooth penalty by sacrificing the bitrate.

Local training model vs. federated model. We demonstrate the gain of federated learning by comparing the performance of the federated model against a model centrally-trained using 10% of 3G training data from scratch. We limit the number of training data to imitate a real scenario when clients don't have enough data. We test the models with the 3G test data. The results are in Fig 8. The results show that with limited training data, the QoE of the local model is 10% lower than that of the federated model. And the local model can't perform well in a specific network condition, for the federated model achieves higher QoE than the local model when QoE is between 0.25 and 1.00.

C. Ablation Study

We conduct two ablation studies to analyze the impact of each design module in the proposed framework separately.

Context variable. We evaluate the effect of context variables on the performance of FedABR. Context is a powerful structure [39]. We trained the FedABR model without the context network (non-context model) under the condition of keeping other factors unchanged, and tested the context model and non-context model on the 3G test data. The results in Fig 9 show that the QoE of the non-context model is 9.5% lower than that of the context model.

Adaptation phase. We also compared the performance improvements in adaptation phase. The results are shown in Fig 7(a). The personalized model performs 1.5% better than the global model on the 4G network. While on Wifi and 3G networks, the performance gap is minor.

VI. Conclusion

We proposed FedABR, a novel ABR algorithm based on personalized federated learning to address the challenges of adaptive video streaming applications. To protect clients' data privacy and enable local model dealing with network environment changes, we introduced a federated learning approach to train a global model without gathering their data together. We also applied an adaptation phase to train a personalized model for each client to maximize their individual QoEs. By jointly training multiple learning tasks with a global model, the proposed method had the ability to provide transferable knowledge to supervise bitrate selection, which could be efficiently adapted to learn a new task in unseen environment with much fewer data samples and training epochs. We implemented the proposed FedABR based on an emulation platform which connected to the Linux network protocol stack through a virtual network interface to send real data packets for evaluation. Extensive experiments based on real-world traces showed that FedABR achieved the best comprehensive QoE compared with the state-of-the-art ABR algorithms in a variety of network environments.

REFERENCES

[1] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

[2] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[3] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 internet measurement conference*, 2012, pp. 225–238.

[4] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[5] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "pistream: Physical layer informed adaptive video streaming over lte," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 413–425.

[6] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 97–108.

[7] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[8] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[9] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[10] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1967–1976.

[11] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[12] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 2020, pp. 495–511.

[13] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Comyco: Quality-aware adaptive video streaming via imitation learning," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 429–437.

[14] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.

[15] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[16] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

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

[18] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[19] X. Yao, T. Huang, C. Wu, R.-X. Zhang, and L. Sun, "Federated learning with additional mechanisms on clients to reduce communication costs," *arXiv preprint arXiv:1908.05891*, 2019.

[20] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2020.

[21] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[22] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *MLSys*, 2020, pp. 429–450.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[25] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[26] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[27] H. Yang, H. He, W. Zhang, and X. Cao, "Fedsteg: A federated transfer learning framework for secure image steganalysis," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1084–1094, 2020.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[29] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[30] P. R. Montague, "Reinforcement learning: an introduction, by sutton, rs and barto, ag," *Trends in cognitive sciences*, vol. 3, no. 9, p. 360, 1999.

[31] Y. Wu and Y. Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," 2016.

[32] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[33] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[34] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for {HTTP}," in *2015 {USENIX} Annual Technical Conference ({USENIX}{ATC} 15)*, 2015, pp. 417–429.

[35] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.

[36] "Raw data - measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/, 2021.

[37] A. Bokani, M. Hassan, S. S. Kanhere, J. Yao, and G. Zhong, "Comprehensive mobile bandwidth traces from vehicular networks," in *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, pp. 1–6.

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

[39] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-q-learning," *arXiv preprint arXiv:1910.00125*, 2019.