Optimizing Bandwidth Reservation Decision Time in Vehicular Networks using Batched LSTM

Abdullah Al-khatib[1], Klaus Moessner[2], Holger Timinger[3]
Institute for Data and Process Science, Landshut University of Applied Sciences, Germany[1,3]
Professorship for Communications Engineering, Technical University Chemnitz, Germany[2]

***Abstract*—Time-sensitive and safety-critical networked vehicular applications, such as autonomous driving, require deterministic guaranteed resources. This is achieved through advanced individual bandwidth reservations. The efficient timing of a vehicle decision to place a cost-efficient reservation request is crucial, as vehicles typically lack sufficient information about future bandwidth resource availability and costs. Predicting bandwidth costs often using time-series machine learning models like Long Short-Term Memory (LSTM). However, standard LSTM models typically require longer durations of multiple input data sets to achieve high accuracy. In certain scenarios, quick decisions must be made, even if the vehicle means sacrificing some accuracy. We propose a batched LSTM model to assist vehicles in placing bandwidth reservation requests within a limited data for an upcoming driving path. The model divides data during training to enhance computational efficiency and model performance. We validated our model using historical Amazon price data, providing a real-world scenario for experiment. The results demonstrate that the batched LSTM model not only achieves higher accuracy within a short input data duration but also significantly reduces bandwidth costs by up to 27% compared to traditional time-series machine learning models.**

***Keywords*—Networked vehicular application; time-sensitive networking; network reservation; batched LSTM**

I. Introduction

In recent years, significant efforts have been made by academia and industry to leverage powerful onboard computing resources for applications such as self-driving [1], [2]. These applications are typically computation-intensive, safety-critical, and time-sensitive, requiring immediate action and reaction to ensure safety. However, the limited onboard computing resources of a single vehicle may not be sufficient to handle the demands of these applications. To address this, application data can be offloaded to cloud servers or edge cloud servers via 5G Vehicle-to-Infrastructure (V2I) connections [3], [4]. A prominent challenge is the execution of computation-intensive tasks within strict time constraints, often with a maximum latency threshold of 100 ms $\sim 1s$ [5].

To ensure the essential network resource requirements (i.e., bandwidth) between vehicles and fog/edge networks for safety, a reservation approach is employed. This approach guarantees timely access to scarce bandwidth resources. The conventional approach, network-side reservation [6], [7], [8], involves the MNO allocating bandwidth for various Quality of Service (QoS) classes. However, this approach provides probabilistic rather than deterministic guarantees for individual vehicles accessing network bandwidth. Vehicle-side and individual bandwidth reservation has proven to be a more efficient solution, where vehicles reserve the necessary resources in advance and make reservations based on their specific requirements and resource costs rather than relying on the MNO to allocate resources on their behalf [9], [10], [11], [12], [13]. Another focus of this approach is the economical aspect, which is being minimized by the expenditure for guaranteed access to the network upon reservation.

From the viewpoint of business, MNOs have various traditional pricing options for allocating resources. These involve network service reservation (i.e., subscription) [14] and the Pay-As-You-Go (PAYG) option [15]. However, these pricing strategies may not always effectively manage peak-time and real-time network conditions while ensuring sufficient QoS. Recently, dynamic pricing has emerged as a promising solution for resource management in edge computing [16], [17]. This method dynamically adjusts prices based on network conditions, aiding in managing congestion and ensuring QoS. However, many individual reservation methods overlook dynamic pricing set by MNOs [9], [10], [11], leading to overpriced reservations or insufficient resources.

As a result, vehicles face various challenges, such as the timing of place reservation requests, leading to higher costs or missed opportunities for cost savings in this dynamic environment. Companies like Amazon Web Services (AWS), MTN, China Telecom, and Uninor utilize this method, adjusting prices in response to supply and demand [18], [19], [20]. In a prior study [12], we introduced a concept called a smart request, which is a strategically placed reservation request that allows vehicles to optimize bandwidth costs and mitigate potential risks associated with dynamic pricing and resource unavailability from MNOs. We employed machine learning techniques, specifically LSTM and Transformers, which have proven effective in temporal prediction tasks. However, the conventional LSTM model required a long time interval of input data to achieve high precision, which is not always feasible in some scenarios where vehicles often need to make quick decisions, even if it means sacrificing some accuracy.

To address this, we proposed the batched LSTM model. This model divides data into batches during the training process, which can help improve computational efficiency and model performance. It optimizes the decision time for bandwidth reservation requests within a multiple time interval for predicting bandwidth costs for an upcoming driving path. The major contributions of our article can be summarized as follows:

- We formulate the optimization problem for bandwidth reservation with the primary objective of minimizing the comprehensive cost over a specified time interval.

- We propose a batched LSTM model, which is utilized to optimize decision time by efficiently handling multiple requests concurrently within a short time.
- Through simulations based on the historical dataset of Amazon [18], we show that the proposed model achieves higher accuracy for a given time interval of data and also reduces bandwidth costs compared to traditional models.

The remaining sections of the paper have been arranged as follows: Section II reviews relevant work on cost-effective resource reservation. Section III covers the system model. Section IV provides a formulation for the bandwidth reservation problem. In Section V, we propose to optimize bandwidth reservation decision time and the implementation of the batched LSTM model in this work. In Section VI, we carry out a comparison of the performance of the proposed model with state-of-the-art methods. Section VII concludes the article.

II. Related Work

Many studies have explored problems in resource reservation, focusing on network-side resource reservation in mobile networks[6], [7]. However, few studies have considered the economic implications of vehicle-side reservations with a focus on minimizing resource consumer expenditure. This is becoming a growing area of interest in edge computing [9], [10], [11], [12], [13].

Generally, most studies related to reservation requests mainly put emphasis on the onsite competition [21],[22] or immediate request mode. The main difference between those two types of requests is that in competition requests, users compete for the resource through various game theoretic ways, such as auctions, Stackelberg game, etc. [21], [23], [22]. This results in only a limited number of winners acquiring the resources, leading to a risk of failure for some users and a violation of QoS. Furthermore, onsite requests frequently exhibit volatile pricing and inherent inequity due to the stochastic nature of resource availability and demand. In contrast, immediate requests, as discussed in [10], the authors developed an approach based on meta-learning to assist in reserving resources for computing with the goal of minimizing the cost of using edge services. Zang et al. proposed a smart online reservation framework to minimize the cost of reserving resources for an individual user [9] or multiple users [11]. Based upon their settings, the approaches discussed above for reservations mainly operate on an immediate request basis. As a result of limited resources, the corresponding vehicles need to carry out the schedule reservation well in advance in order to ensure they are able to acquire the necessary resources on time. As a consequence, the immediate requests entail high costs and low guarantees. Planning reservations efficiently is a challenge as users lack knowledge about cost trends and available resources, making it difficult to ensure cost-effectiveness.

Motivated by challenges incurred by competition and immediate requests, an advanced reservation solution [12] has been introduced. This solution enables the advanced reservation of mobility locations at specific time intervals, achieving commendable cost-effectiveness and time efficiency. However, this study lacks an in-depth discussion of the challenges and complexities associated with making reservation decisions within short time intervals. To enhance the reservation decision time, we leverage batched LSTM, which is often more effective for quick decisions with limited data than standard LSTM due to its ability to process multiple input sequences concurrently.

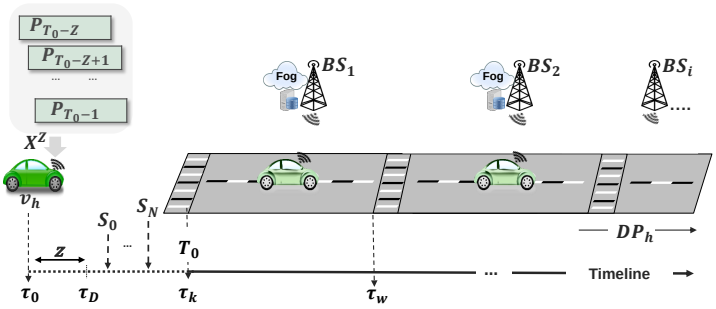


Fig. 1. System model.

III. System Model and Formulation Problem

A. System Model

In this paper, the urban vehicular network is composed of vehicles and Base Stations (BSs), which include Macro Base Stations (MBSs) and Road Side Units (RSUs). The driving path (DP) is partitioned into road segments (RS), each BS associated with a single MNO. This MNO establishes wireless connections between edge routers and participating vehicles within the core network. In each segment of the road, a BS is strategically located (Fig. 1).

In order to meet the strict latency demands of vehicular applications, a Fog/Edge server node is intricately integrated into the infrastructure of each BS. At the initial time τ_0, vehicle v_h $(h = 1, 2, ..., V)$ initiates a reservation request to the MNO for a specific RS along the DP_h. This request provides detailed information about the bandwidth time period Δt_h required to successfully traverse the intended DP_h.

The specified time duration, denoted as Δt_h, is carefully divided into reservation time intervals represented by the range $[\tau_k, \tau_w]$. Each of these intervals corresponds to a specific RS_i, which is entirely covered by a BS_i and the i^{th} road segment. τ_k denotes the entry time into the designated RS_i, while τ_w marks the exit time from that specific RS_i. The determination of these intervals depends on the length of DP_h and the speed of the vehicle v_h, both of which are computed based on data obtained from the navigation system.

Vehicle v_h divides its desired departure interval $[T_0, T_R]$ into $|St|$ small intervals, each representing a discrete time step. For each time step, vehicle v_h requests the cost associated with its designated route DP_h. The resulting list of costs, p_{T_0},..., p_{T_R}, forms a session (S_0). This process is repeated at regular intervals of Δt_{s_v}, up to a maximum of N times, resulting in a sequence of pricing sessions (S_0, S_1,..., S_N), as illustrated in Fig. 1. The session validity time (Δt_{s_v}) is the duration for which the pricing sessions remain valid. This is a predetermined parameter set by the MNO.

B. Formulation Problem

The optimization problem for bandwidth reservation aims to minimize the comprehensive cost associated with reserving bandwidth over the time interval $[\tau_k, \tau_w]$ for all RS_i in D_h. In this approach, the RS_i is transformed into discrete cost areas using a function φ. Each cost area, denoted as A, is associated with a time interval $[\tau_k, \tau_w]$.

$$\varphi : R_+^2 \rightarrow R_+^{N \times R} : (\tau_k, \tau_w) \mapsto A \tag{1}$$

where $R = T_R - T_0$ and A is a matrix holding the information about the prices:

$$A = \begin{bmatrix} \begin{pmatrix} p_{11} & \cdots & p_{1R} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NR} \end{pmatrix} \end{bmatrix} \tag{2}$$

This matrix has S_N rows (representing sessions) and T_R columns (representing the desired time departure $[T_0, T_R]$), where $p_{n,r}$ denotes the price corresponding to the n-th session and the r-th desired departure time. The θ function is utilized to establish a relationship between a designated area A and its comprehensive reservation cost:

$$\theta : R_+^{N \times R} \rightarrow R_+ \tag{3}$$

Additionally, the function $\omega = \theta \circ \varphi$, where θ and φ are distinct functions, is used to map:

$$\omega : R_+^2 \rightarrow R_+ \tag{4}$$

Ultimately, the objective function J is formulated as follows:

$$J = \sum_{\tau_k, \tau_w} \omega(\tau_k, \tau_w) \tag{5}$$

The sum operates over all pairs (τ_k, τ_w) of start and end times. The function ω assigns these time intervals to an overall reservation cost based on the previously defined composite function, incorporating both φ and θ. The objective is to minimize this overarching objective function.

$$\min_{\tau_k, \tau_w} J(\tau_k, \tau_w) \tag{6}$$

The implementation of an optimization technique requires a sophisticated prediction model for accurate decision-making. The process begins with a proficient time-series model, ensuring high prediction accuracy. This forms the basis for effectively applying the optimization technique, using the insights from the prediction model to guide decisions. The model uses historical average prices, X^Z, from the interval $[T_0 - Z, T_0 - 1]$ (as per Eq. 7) to predict the price at time t. Here, Z is the number of time steps before the first price request (τ_D). Optimal Z values can be found by searching for a δt that yields satisfactory accuracy. Hence, the prediction model can be expressed as:

$$(\widehat{x}_{Z+1} = f\left(X^Z; \theta\right)) \tag{7}$$

where,

$$X^Z = [P_{T_0 - Z},\ P_{T_0 - Z + 1},\ \cdots\ P_{T_0 - 1}]$$

f is the model with trainable parameters θ, which returns the expected future price, $\widehat{P}_{T_0}$, for reserving in the following t time steps. The expected price $\widehat{P}_{T_0}$ is then pushed into X^Z, forming a constant size buffer as follows: $X^Z = \left[P_{T_0 - Z + 1}\ \cdots\ \widehat{P}_{T_0}\right]$. Subsequently, the value $\widehat{P}_{T_0+1}$ is obtained by repeating the process of predicting and updating the buffer. This process continues until the expected future price $\widehat{P}_{T_R}$ is obtained, as shown in Fig. 2. The method for finding the minimum expected price is as follows:

$$(\widetilde{p} = min(\widehat{P}_{T_0},\ \widehat{P}_{T_0+1},\ \ \cdots\ \widehat{P}_{T_R})) \tag{8}$$

In order to calculate the optimal reservation time, we introduce the concept of a decision threshold function, denoted as DE. This function uses the output (the expected minimum future price) to advise a vehicle on its reservation strategy. Specifically, it determines whether the vehicle should reserve at the minimum price in the current session (reserve) or consider postponing the reservation by requesting a new session (wait). The decision action is computed using Eq. (9):

$$DE = \begin{cases} reserve: & \widetilde{p}_{S_n} \leq \widetilde{p} - \varepsilon(n) \\ wait: & \text{otherwise} \end{cases} \tag{9}$$

where,

$$\varepsilon(n) = \frac{c}{n}\ ,\ n \in [1,\ N]$$

In our model, N represents the maximum number of sessions that can be requested. The hyper-parameter c should be tuned to maximize benefits. The expected minimum future price, denoted as $\widetilde{p}$, is derived from Eq. (8). Our model anticipates session prices for the time interval $[T_0, T_R]$. The term $\widetilde{p}_{S_n}$ represents the lowest price at the current requested session, which includes prices $\widetilde{p}^{S_n}_{T_0}, \widetilde{p}^{S_n}_{T_0+1}, \ldots, \widetilde{p}^{S_n}_{T_R}$ in the time interval $[T_0, T_R]$. This is defined in Equation (10):

$$(\widetilde{p}_{S_n} = min(\widetilde{p}^{S_n}_{T_0},\ \widetilde{p}^{S_n}_{T_0+1},\ \ \cdots\ \widetilde{p}^{S_n}_{T_R})) \tag{10}$$

The proposed model suggests the optimal reservation time for the vehicle by returning the time associated with the lowest price. The steps of this algorithm are detailed in Algorithm 1. In the context of Formula 9, determining the optimal risk level is of substantial importance. This is because the proposed strategy aims to minimize overall costs while reducing the risk of rejecting valid sessions. The goal is to find an optimal balance that maximizes efficiency and minimizes negative outcomes. This strategy is based on the concept that the risk is lower at the beginning, but as the maximum number of sessions is approached, the risk begins to increase. The

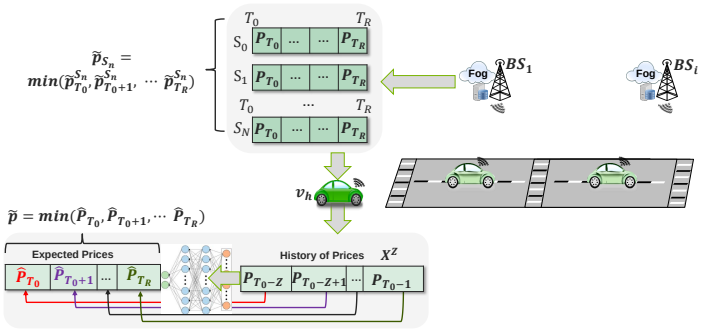


Fig. 2. Prediction model bandwidth reservation request timing.

decision function uses $\varepsilon(n)$ because the model predicts the expected price of reserving in the next time step δt, not the minimum price. Statistically, if the vehicle waits, it will receive a price that is at least equal to the expected price. However, there is a high chance of getting a lower price, and the amount of reduction depends on the variance in the prices. Therefore, the constant c should be proportional to the variance in the prices of DP_h. As the value of N is approached, $\varepsilon(n)$ should decrease. Here, N represents the maximum number of sessions that can be requested per time step δt, which is determined by the MNO. The value of $\varepsilon(n)$ also depends on the time the vehicle started searching for the minimum price prior to takeoff.

In our model, the constant c is crucial. Its optimal values are determined empirically, balancing cost savings and risk reduction. The choice of c also prevents failing to reserve resources due to reaching the maximum number of sessions. Further investigation of these values is conducted in our experiments.

IV. Bandwidth Reservation Decision Time

First, a brief overview of the approach is provided. Following that, appropriate time series prediction models are selected and applied. This enables vehicles to reserve bandwidth on a specific future path at a designated time, thereby minimizing costs. Subsequently, the prediction model, which utilizes the batched LSTM algorithm, is explained in detail.

A. Overview on the Proposed Approach

From the vehicle's perspective, the challenge lies in efficiently timing decisions for bandwidth reservation requests due to insufficient future price prediction data. The proposed approach involves vehicle v_h dividing its desired departure interval $[T_0, T_R]$ into Q small δt intervals. It then requests the cost for each area A of DP_h at each future time.

The proposed method uses a time-series deep neural network to predict future costs and continues to request pricing sessions until the minimum price surpasses a dynamic threshold. This threshold, determined by a parameter c, adjusts with each new session to balance risk. The prediction model, trained on models like LSTM and Transformers, leverages recurrent load patterns along DP_h. Despite the high computational cost of training, it's infrequent and can be offloaded to an external server for efficient onboard cost prediction. Retraining is only necessary if major road events significantly alter local traffic.

B. Machine Learning Model

In this section, the details of the primary families of candidate time-series deep neural network models (LSTM, batched LSTM, and Mix) are explored to assess their suitability for the prediction task.

1) LSTM: The LSTM model is the classical LSTM architecture without batches. LSTM is an artificial recurrent neural network (RNN) architecture extensively used for sequence modeling and prediction tasks. LSTM networks excel at handling problems where inputs possess long-term dependencies or temporal relationships. Traditional RNNs are susceptible to the "vanishing gradient" problem, which impedes their ability to capture extended dependencies in sequences. LSTM networks tackle this challenge by introducing memory cells and gating mechanisms that control the information flow within the network. The key components of an LSTM network are:

- Cell State: It serves as the memory of the network and is responsible for capturing long-term dependencies. The cell state C can selectively forget or store information using gate units.
- Forget Gate: It decides which information in the cell state should be forgotten or discarded. In the time step t, the decision is made using a sigmoid function σ of the current input vector x_t and the current hidden state h_t. The output, called f_t, is a weight value between 0 and 1: 0 means "let nothing through", 1 means "let everything through".
- Input Gate: It determines which information from the input should be stored in the cell state. In the time step t, using always a sigmoid function σ, it is decided which values will be updated (i_t). A tahn layer creates a vector of new candidate values $\tilde{C}_t$ and, combing i_t with $\tilde{C}_t$, an update C_t is created.
- Output Gate: It controls the output of the LSTM unit and determines what information should be output based on the current input and the previous cell state. In time step t, a sigmoid layer decides what parts of the cell state will go to output. This part is called o_t. The output h_t is obtained by multiplying o_t with the cell state C_t transformed by the $tanh$ function. The formulas outlined within it are:

$$f_t = \sigma(W_f(h_{t-1}, x_t) + b_f) \tag{11}$$

$$i_t = \sigma(W_i(h_{t-1}, x_t) + b_i) \tag{12}$$

$$\tilde{C}_t = tanh(W_C(h_{t-1}, x_t) + b_C) \tag{13}$$

$$C_t = f_t C_t + i_t \tilde{C}_t \tag{14}$$

$$o_t = \sigma(W_o(h_{t-1}, x_t) + b_o) \tag{15}$$

$$h_t = o_t tanh(C_t) \tag{16}$$

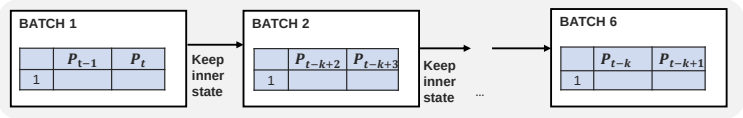


Fig. 3. Structure of batch step

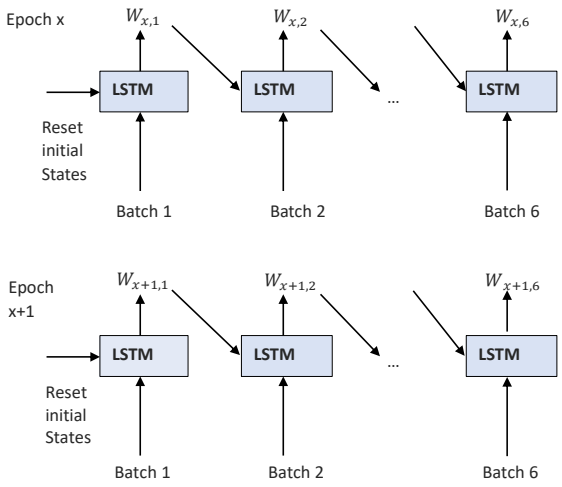


Fig. 4. The structure of a stacked batched LSTM network.

2) Batched LSTM: Batched LSTM is an LSTM model with batches. In LSTM networks, the data is typically processed in batches during training and inference. Let's discuss LSTM with different batches:

- Batch Processing: In LSTM, batch processing refers to dividing the input data into multiple batches. Each batch consists of a subset of the training or test dataset. Batch processing offers several benefits, including improved computational efficiency, parallel processing, and better generalization.
- Variable Batch Sizes: LSTM networks can handle variable batch sizes. It means that each batch can have a different number of sequences or time steps. This flexibility allows for processing sequences of varying lengths within the same batch, which is particularly useful when working with time series data of different lengths.
- Training with Different Batch Sizes: During training, LSTM networks are typically trained on multiple batches, where each batch contains a fixed number of sequences or time steps. The batch size can be chosen based on factors such as computational resources, memory constraints, and the specific characteristics of the dataset. Common batch sizes range from a few samples to several hundreds or even thousands.
- Impact on Training: The choice of batch size can influence the training process. Smaller batch sizes provide more frequent weight updates, which can lead to faster convergence but may result in noisy gradients due to a smaller sample size. Larger batch sizes, on the other hand, offer better gradient estimation but may slow down the training process and require more memory.
- Testing and Inference with Batches: During testing or inference, LSTM networks can process input data in batches as well. This allows for efficient evaluation of multiple samples simultaneously. The batch size for testing can be different from the batch size used during training.
- Handling Remaining Data: When the total number of samples is not divisible by the chosen batch size, there may be a smaller "remainder" batch at the end of each epoch. Some approaches include discarding the remaining data, padding it to match the batch size, or using dynamic batching techniques that can handle variable batch sizes more gracefully.

Overall, LSTM networks can handle different batch sizes, allowing for efficient processing of time series data in parallel. The choice of batch size depends on various factors, such as computational resources, memory constraints, and the characteristics of the dataset. Careful consideration should be given to selecting an appropriate batch size to balance computational efficiency and model performance. Two types of LSTM with batches exist:

- Stateless batches LSTM: In a stateless LSTM with batches, the internal states of the LSTM cells are reset at the beginning of each batch. This means that the LSTM does not retain any memory of the previous batch when processing the next batch. The LSTM treats each batch as an independent entity.
- Stateful LSTM with batches: In a stateful LSTM with batches, the internal states of the LSTM cells are preserved between batches. The LSTM maintains the hidden states and memory states from the previous batch and uses them as the initial states for processing the next batch. The LSTM carries information from one batch to another within a sequence.

Stateless LSTM with batches is suitable for independent sequences, while stateful LSTM with batches is useful for capturing sequential dependencies and handling variable-length sequences within a batch. For this reason, batched LSTM is a stateful model.

The structure of LSTM with 6 batches is similar to that of LSTM, but the parts of the batches are described in Fig. 3 and 4. While in LSTM in each epoch, only 1 set W is obtained, in batched LSTM in each epoch, 6 sets W are obtained. During each epoch, both LSTM models update their internal parameters (weights and biases) based on the training data and the computed gradients. This parameter update aims to minimize the difference between the model's predictions and the actual targets, effectively improving the model's ability to capture patterns and make accurate predictions. The relationship between different epochs is that each epoch builds upon the progress made by the previous epochs. As the training progresses through multiple epochs, the model's performance typically improves, and the learned representations become more refined. However, it's important to note that the relationship between epochs is not strictly linear or guaranteed to continually improve.

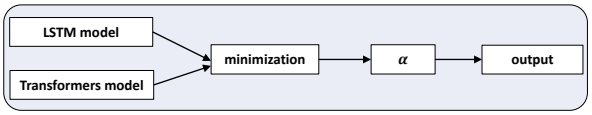


Fig. 5. Structure of mix model.

3) Mixed model: The mixed model, where transformers [24] and LSTMs are run separately, is considered. The key components of a mixed model are:

- Transformer Model: Start by training a transformer model on the time series data. The transformer model captures global dependencies and long-range patterns effectively by utilizing self-attention mechanisms.
- LSTM Model: Train a separate LSTM model on the same time series data. The LSTM model focuses on capturing local dependencies and temporal patterns within the time series.
- Prediction Combination: Once both the transformer and LSTM models are trained, combine their predictions to obtain the final output. This combination is a weighted average, where the weights are calculated to minimize the MSE of the prediction.

Overall, this mixed model combining transformer and LSTM models separately provides a way to leverage the strengths of each architecture and capture both global and local dependencies in the time series. However, it comes with increased complexity and potential challenges in integrating the two models effectively. The structure of the Mix model is described in Fig. 5.

C. Prediction Model Algorithm with Batched LSTM

The algorithm presented is a predictive model that employs a batched LSTM technique to forecast future prices within a time series. Its main goal is to estimate the upcoming price ($\hat{P}_{T_0+1}$), determine the minimum expected price ($\widetilde{p}$) over a specified time interval, and make a decision based on a predefined criterion. To initiate the process, the algorithm takes into account the average prices (X^Z) within a specified time range, namely $[T_0 - Z, T_0 - 1]$. It requires certain parameters, Z, and operates on a model with trainable parameters (θ). The algorithm unfolds in multiple steps. Step 1 represents the initialization and the model training. The algorithm commences by initializing the sequence of average prices (X^Z) and configuring a batched LSTM model with parameters (θ). A crucial step involves defining a batch size, and subsequently, X^Z is reshaped into batches for streamlined processing. The training of the model takes place within a loop, persisting until the successful prediction of $\hat{P}_{T_0+1}$. This training iteration involves updating the LSTM with batches, and the trainable parameters (θ) undergo adjustments as dictated by the model's training procedure. Simultaneously, X^Z is continually updated to incorporate the newly predicted values. The steps from 11 to 18 represent the decision making. Moving forward, the algorithm proceeds to calculate the minimum expected price ($\widetilde{p}$) over the predefined time interval. The decision-making process ensues, determining whether to reserve or wait. This decision hinges on a condition integrating $\widetilde{p}_{S_n}$ and a decreasing function $\epsilon(n)$. The culmination of this phase involves the algorithm providing as output the determined minimum expected price ($\widetilde{p}$) and the corresponding time. Utilizing LSTM for time series prediction ensures the algorithm's ability to capture intricate temporal dependencies. The incorporation of batch processing in the model training phase enhances computational efficiency. Decision making involves a dynamic strategy, employing a threshold ($\epsilon(n)$) and the minimum expected price, guiding the choice between reserving or waiting.

Algorithm 1 Prediction Model Algorithm with Batched LSTM

Input:

- Average prices, X^Z, between time interval $[T_0 - Z, T_0 - 1]$
- Parameters Z
- Model with trainable parameters θ

Output:

- Expected future price, $\widehat{P}_{T_0+1}$, of reserving in the following t
- Minimum expected price, $\widetilde{p}$, for the time interval $[T_0, T_R]$
- Time associated with the lowest price

1: **procedure** PREDICTION MODEL BATCHED LSTM(X^Z, Z, θ)
2: $X^Z \leftarrow [P_{T_0-Z}, P_{T_0-Z+1}, ..., P_{T_0-1}]$
3: Initialize batched LSTM model with parameters θ
4: $batch\_size \leftarrow b$ ▷ Set your desired batch size
5: $X^Z\_batches \leftarrow$ reshape_into_batches($X^Z, batch\_size$)
6: **while** $\widehat{P}_{T_0+1}$ not obtained **do**
7: **for** $batch$ in $X^Z\_batches$ **do**
8: $\widehat{P}_{T_0+1} \leftarrow$ LSTM_Model($\theta, batch$)
9: Update θ using the model's training procedure
10: $X^Z \leftarrow [P_{T_0-Z+2}, ..., \widehat{P}_{T_0+1}]$
11: **end for**
12: **end while**
13: $\widetilde{p} \leftarrow \min(\widehat{P}_{T_0}, \widehat{P}_{T_0+1}, ..., \widehat{P}_{T_R})$
14: **if** $\widetilde{p}_{S_n} \leq \widetilde{p} - \varepsilon(n)$ **then**
15: $DE \leftarrow$ "reserve"
16: **else**
17: $DE \leftarrow$ "wait"
18: **end if**
19: $\varepsilon(n) \leftarrow \frac{c}{n}$ for $n \in [1, N]$
20: $\widetilde{p}_{S_n} \leftarrow \min(\widetilde{p}_{T_0}^{S_n}, \widetilde{p}_{T_0+1}^{S_n}, ..., \widetilde{p}_{T_R}^{S_n})$
21: **Return** $\widetilde{p}$ and associated time
22: **end procedure**

V. PERFORMANCE EVALUATION

A. Dataset Description

To assess the effectiveness of our methodology, we utilized a historical dataset of Amazon spot prices, which are subject to fluctuations influenced by factors such as capacity, demand, geographic location, and specific instance types [18]. Given the time-sensitive nature of various applications, vehicles require both computing instances and communication links, i.e., bandwidth. Our assumptions are that the pricing for setting up computing and communication resources aligns with the Amazon spot pricing model, as previously referenced in [11],

[9]. For this study, we collected pricing data from all available instances and two specific regions, namely us-west-1b and us-west-1c. This data was collected from April 17, 2021, to May 2, 2021, for training purposes, and from May 3, 2021, to May 8, 2021, for the testing phase of the model.

B. Experimental Results

In this section, the experimental results of the study are presented, encompassing model evaluation and metrics, a comparison of prediction errors among different models, an analysis of the level of risk, and the performance evaluation of cost. The study includes four models: LSTM, batched LSTM, Transformers, and Mix. The experiments were conducted using the PyTorch framework in Python and trained on an NVIDIA GeForce RTX 2080 Ti GPU.

1) Model evaluation and metrics: Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are three commonly used metrics for evaluating the performance of prediction models and assessing their forecasting accuracy. These metrics provide valuable insights into the effectiveness and reliability of the models in capturing the underlying patterns and making accurate predictions.

Table I presents a comparison of the time consumption among four models, measured in seconds. Batched LSTM and Transformer models demonstrate significantly shorter processing times compared to LSTM and the mix model, thanks to their ability to leverage parallel processing. It is noteworthy that the mix model is the most time-consuming, which is a recognized drawback of this particular approach due to its hybrid nature. Additionally, the table includes a comparison of the time taken for a single forward pass among the three models. Batched LSTM takes advantage of parallel processing by batching sequences together, resulting in improved performance during a forward pass. This parallelization allows for more efficient utilization of hardware resources like GPUs, enabling batched LSTM to achieve faster forward pass times compared to processing each sequence individually, as in LSTM. Transformers are known for their exceptional parallel processing capabilities. By employing self-attention mechanisms, Transformers can simultaneously process all positions in the input sequence, enabling parallelization across different positions. Consequently, Transformers achieve fast forward pass times, especially for long sequences. Interestingly, Transformers and batched LSTM show similar forward pass times, which is advantageous for resource-constrained environments like vehicles, where computational resources are limited and model training need not be performed onboard. The mix model is not considered in this comparison due to its hybrid nature, which makes it challenging to fit into the batch processing paradigm. As a result, the focus is on analyzing the performance of models that fully embrace parallel processing for more efficient and faster computations. In addition, the parameters settings of models are as shown in Table II.

2) Model prediction error comparison: In this section, the error analysis provides valuable insights into the strengths and weaknesses of each model in capturing different aspects of prediction accuracy. The three error metrics, MAE, MAPE, and RMSE, play distinct roles in assessing prediction accuracy.

TABLE I. TIME CONSUMPTION OF TRAINING PER EPOCH AND PER SINGLE FORWARD PASS TIME ON THE AMAZON SPOT PRICING DATASET

Training Time per epoch [s]				Time Single Forward-Pass [ms]		
LSTM	Tranf.	Mix	B.LSTM	LSTM	Tranf.	B.LSTM
0.034	0.025	0.188	0.025	2.412	1.889	1.886

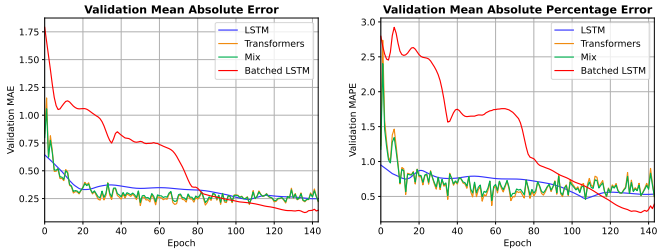
TABLE II. PARAMETERS SETTINGS

Model	Layer	Hidden layers	Dropout percentage
LSTM	1	100	
Transformers	2	10	0.2
Mix model	3	110	0.2
Batched LSTM	1	100	

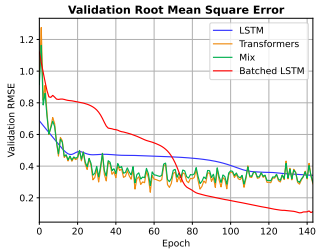
MAE emphasizes the magnitude of errors; MAPE provides a relative measure of the prediction error as a percentage; and RMSE takes both the magnitude and direction of errors into account, giving more weight to larger errors. The choice of which metric to use depends on the specific context and requirements of the problem at hand. Based on these metrics, the batched LSTM model emerges as the top performer, achieving a lower error rate as demonstrated in Fig. 6. However, for a more comprehensive evaluation, the errors of other models are compared as well. At epoch 140, the Transformers and Mix models show similar errors to the LSTM model when considering MAE and RMSE. On the other hand, taking MAPE into account, the LSTM model exhibits a smaller error compared to Transformers and Mix, though still higher than the Batched model. The lower error rate with MAPE for the LSTM model can be attributed to its effective handling of outliers, as MAPE is less influenced by extreme values compared to RMSE and MAE. Consequently, the LSTM model demonstrates superior robustness in producing predictions when faced with extreme data points.

Fig. 7 clearly shows that the accuracy of the model improves with a longer time interval provided as input. The length of time (in the analyzed case, an hour) that the vehicle has to determine the input's duration directly affects the model's accuracy, with longer input intervals leading to better performance and a lower MAE. Consistently, the LSTM models outperform the Transformer model. They demonstrate better performance in capturing temporal dependencies and patterns in the time series data, consistently yielding lower MAE values compared to the Transformer model across different input matrix lengths. This indicates that the LSTM models are more effective in modeling the sequential nature and capturing relevant information for accurate predictions.

On the other hand, the Transformer model benefits from longer sequences, exhibiting a larger reduction in MAE as the length of the input matrix increases. Although the Transformer model has a higher MAE than the LSTM models at all input lengths, it shows a more significant reduction in MAE with longer sequences. This suggests that the Transformer model can leverage the additional information present in longer sequences to learn more complex patterns. While the MAE remains higher than that of the LSTM models, the relative improvement in performance for the Transformer model is substantial. Based on these observations, it can be concluded that the LSTM models consistently outperform the Transformer model in terms of MAE. The LSTM models'

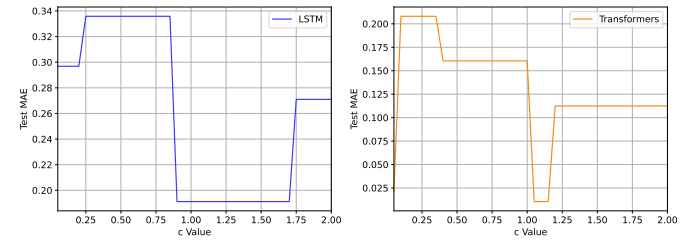


(a) (b)

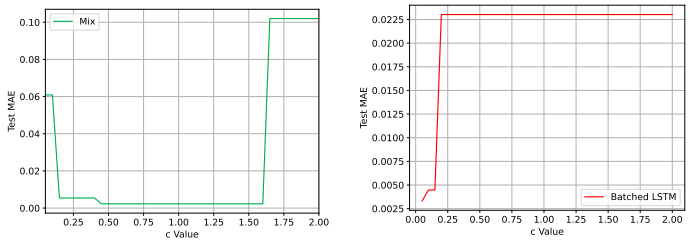


(c)

Fig. 6. Prediction errors comparison between models through epochs.



(a) LSTM (b) Transformers



(c) Mix (d) Batched LSTM

Fig. 8. Comparison of risk levels among different models.

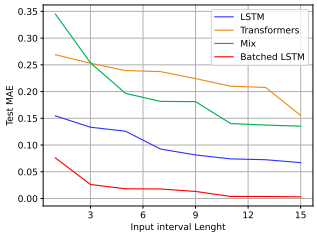


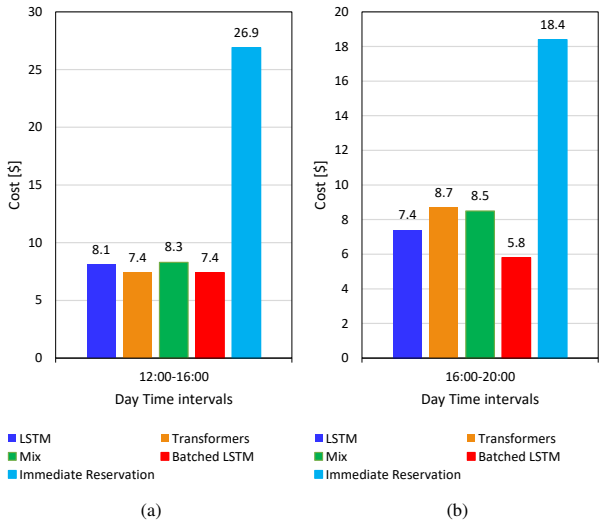
Fig. 7. Input request price time interval.

ability to model sequential dependencies and capture relevant information gives them an advantage over the Transformer model. However, the Transformer model shows promise in handling longer sequences and exhibits a larger reduction in MAE, indicating its potential for capturing more complex patterns and dependencies. In the context of the mix model, if either the transformer or LSTM model poorly estimates the true price, the decision threshold step in the bandwidth cost problem may lead to worse performance of the mix model compared to both the transformer and LSTM models, even if it outperforms them in the estimation step.

Finally, the batched LSTM outperforms the standard LSTM due to its consideration of batches, enabling it to produce more generalized results. LSTM models trained with batches tend to exhibit improved generalization performance. By exposing the model to multiple sequences within each batch, it gains exposure to a diverse range of patterns and can capture broader temporal dependencies. This increased variety of examples enhances the model's ability to make predictions on unseen data, resulting in improved generalization and more accurate forecasts, even with shorter input to the model. As observed in Fig. 7, the vehicle only needs a small number of hours (i.e., 1-3 hours) as input to the model to achieve better and more accurate predictions of bandwidth cost. This indicates that the model's performance remains consistent even with limited input, closely resembling real-world scenarios. The advantage of using a shorter input interval is that the vehicle does not need to wait for an extended period to find the best price for bandwidth.

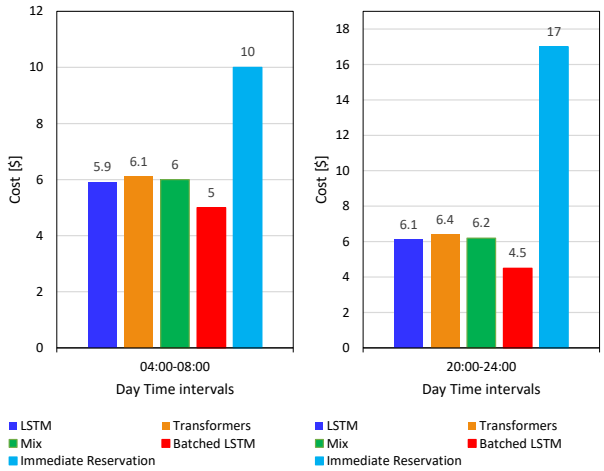
3) Level of risk analysis: The analysis of risk levels among different models sheds light on their precision and helps identify the best c values for optimizing prediction accuracy. When comparing the optimal value of risk achieved by various models in Fig. 8, the batched LSTM demonstrates the lowest value of c, indicating higher precision, while Transformers exhibit the highest value of c. It's important to note that as c increases, precision decreases. For instance, when $\tilde{p}^{S_n} = 1$ and $\tilde{p}$ equals 1.1, the algorithm reserves with a c of 0, whereas a c of 0.2 would prompt the algorithm to wait wrongly. Consequently, batched LSTM, being the most precise algorithm, has the lowest MAE associated with the smallest c, and the change in MAE associated with varying c is minimal. However, all the best c are very small.

4) Cost performance: In this subsection, a comprehensive evaluation of the cost performance of the prediction models is conducted by comparing them with the benchmark immediate reservation request scheme. Three distinct benchmark scenarios are explored, each characterized by different time intervals during which the vehicle searches for the most cost-effective option. As mentioned earlier, the time interval represents the desired departure time for the vehicle. Furthermore, time steps are utilized to divide the aforementioned time interval into smaller segments, such as hours or half-hours.



(a) (b)

Fig. 9. Comparison of cost performance across benchmark with time intervals $[T_0\ , T_R\ ] = 4$ hr and time steps $\delta t = 1$ hr.



(a) (b)

Fig. 10. Comparison of cost performance across benchmark with time intervals $[T_0\ , T_R\ ] = 4$ hr and time steps $\delta t = 0.5$ hr.

Subsequently, the total cost is calculated by determining the number of BSs that the vehicle reserves along its driving path to reach the final destination. The calculation enables an effective assessment of the performance of prediction models in comparison to the established benchmark. The analysis of cost performance highlights the effectiveness of the proposed method in securing reservations at lower prices. Fig. 9 and Fig. 10 illustrate the comparison between the costs of various approaches and immediate reservations. In all cases, the prediction methods derived from the different approaches outperform the immediate reservation scheme in obtaining a better price. This is due to the fact that the proposed approach identifies the optimal price, which may differ significantly from the immediate reservation price.

Among the estimation methods, batched LSTM exhibits the smallest Mean Absolute Error (MAE), resulting in an estimated value that closely approximates the true best price, deviating distinctly from the immediate price. Nevertheless, all four estimation methods capture this difference, making the proposed method valuable for securing reservations at lower prices.

Comparing Fig. 9 with Fig. 10, it is evident that as the time steps (δt) increase from 0.5 hr to 1 hr, the price of immediate reservations may rise, while the cost savings from employing the proposed approach increase. For example, comparing the savings obtained using the estimated price from batched LSTM with the immediate reservation for the interval 12.00-16.00, the saving is 72.49% for 1 hr (Fig. 9.a) and time interval 20.00-24.00 for 0.5 hr (Fig. 10.b) is 73.52%.

VI. Conclusions

In conclusion, this research effectively addresses the challenges in decision time for bandwidth reservation, particularly within the context of safety-critical vehicular applications. The paper introduces an optimized approach using batched LSTM to predict bandwidth costs within a short duration. By organizing data into batches during the training phase, the model enhances both computational efficiency and prediction accuracy. This approach has proven highly effective, through the utilization of real price data, resulting in significant cost reductions by 27% compared to traditional time-series machine learning models, as we have provided in the experimental results. In future work, we aim to explore dynamic BS ranges and varying MNO numbers for more realistic and intelligent reservation request strategies.

References

[1] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10 843–10 856, Jul. 2021.

[2] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, "Multiagent ddpg-based joint task partitioning and power control in fog computing networks," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 104–116, Jan. 2022.

[3] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[4] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, Aug. 2019.

[5] Ieee spectrum, 6 key connectivity requirements of autonomous driving. [Online]. Available: https://spectrum.ieee.org/6-key- connectivity-requirements-of-autonomous-driving

[6] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5g network slicing resource utilization," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May. 2017, pp. 1–9.

[7] A. A. Al-Khatib and A. Khelil, "Priority- and reservation-based slicing for future vehicular networks," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Aug. 2020, pp. 36–42.

[8] A. A. Al-khatib, A. Khelil, and M. Balfaqih, "Bandwidth slicing with reservation capability and application priority awareness for future vehicular networks," in *Proc. - Int. Conf. Adv. Inf. Netw. Appl. (AINA)*. Springer, Apr. 2021, pp. 681–691.

[9] S. Zang, W. Bao, P. L. Yeoh, B. Vucetic, and Y. Li, "Filling two needs with one deed: Combo pricing plans for computing-intensive

multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1518–1533, May. 2019.

[10] D. Chen, Y.-C. Liu, B. Kim, J. Xie, C. S. Hong, and Z. Han, "Edge computing resources reservation in vehicular networks: A meta-learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5634–5646, May. 2020.

[11] S. Zang, W. Bao, P. L. Yeoh, B. Vucetic, and Y. Li, "Soar: Smart online aggregated reservation for mobile edge computing brokerage services," *IEEE Trans. Mob. Comput.*, vol. 22, no. 1, pp. 527–540, Jan. 2023.

[12] A. A. Al-Khatib, F. Al-Khateeb, A. Khelil, and K. Moessner, "Optimal timing for bandwidth reservation for time-sensitive vehicular applications," in *Proc. IEEE Int. Conf. Fog and Edge Comput. (ICFEC)*, May. 2022, pp. 94–99.

[13] A. A. Al-khatib, M. U. Hassan, and K. Moessner, "Heuristic optimization of bandwidth reservation cost for vehicular applications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 4909–4915.

[14] At&t wireless plans. [Online]. Available: https://www.att.com/plans/wireless/

[15] Aws pricing - how does aws pricing work. [Online]. Available: https://aws.amazon.com/pricing/?nc1=h ls

[16] N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Data collection and wireless communication in internet of things (iot) using economic analysis and pricing models: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2546–2590, Jun. 2016.

[17] Y. Liao, X. Qiao, Q. Yu, and Q. Liu, "Intelligent dynamic service pricing strategy for multi-user vehicle-aided mec networks," *Future Gener. Comput. Syst.*, vol. 114, pp. 15–22, Jan. 2021.

[18] Amazon ec2 spot instances pricing. [Online]. Available: https://aws.amazon.com/ec2/spot/pricing/

[19] L. Lin, L. Pan, and S. Liu, "Backup or not: An online cost optimal algorithm for data analysis jobs using spot instances," *IEEE Access*, vol. 8, pp. 144 945–144 956, Aug. 2020.

[20] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "A survey of smart data pricing: Past proposals, current plans, and future trends," *ACM Comput. Surv.*, vol. 46, no. 2, pp. 1–37, Nov. 2013.

[21] Y. Cao, C. Long, T. Jiang, and S. Mao, "Share communication and computation resources on mobile devices: A social awareness perspective," *IEEE Wirel. Commun.*, vol. 23, no. 4, pp. 52–59, Aug. 2016.

[22] I. Bajaj, Y. H. Lee, and Y. Gong, "A spectrum trading scheme for licensed user incentives," *IEEE Trans. Commun.*, vol. 63, no. 11, pp. 4026–4036, Nov. 2015.

[23] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 273–287, Jul. 2020.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017.