VIDTRA: An Efficient and Resilient Video Preloading System

Jiaen Lv, *Graduate Student Member, IEEE*, Yifang Zhang, and Shaowei Wang, *Senior Member, IEEE*

***Abstract*—With the increasing demand for video-based applications, the clarity and fluidity of videos have garnered widespread attention. Current video playback mechanisms either allocate resources to clients with good channel quality or maintain video playback continuity at low bit rates, affecting the user viewing experience. In this work, we propose a video preloading system, namely VIDTRA, which departs from the current paradigm and explores a resilient design scheme. The central insight in VIDTRA is to utilize network situation maps and user trajectory prediction to forecast the serving cell and received signal strength, thereby determining the duration of video preloading. Considering the predictability of public transport route trajectories, our system is primarily designed to enhance the video watching experience for users on public transport, which encompasses functionalities such as route clustering and the identification of users' boarding and alighting. Using real-world data collected from user equipments, we thoroughly evaluate and demonstrate the efficacy of VIDTRA. Results from the experimental evaluations show that VIDTRA can precisely estimate the future signal strength received by users and initiate video preloading before they enter areas with weak signal quality, thus reducing the interruptions of the video while guaranteeing the high definition.**

***Index Terms*—Network situation map, trajectory prediction, video preloading.**

I. Introduction

THE PROLIFERATION of video platforms and the expansion of 5G networks have made it easier for users to consume diverse video content anytime and anywhere [1], [2]. Consequently, ensuring video fluency has become critical for the user experience (QoE) [3], [4]. A common scenario involves users watching videos on mobile devices while on public transport, where playback can be interrupted by weak signal coverage, severely degrading the QoE. Video preloading is an effective technique to mitigate such issues in online media streaming [5]. However, conventional methods often employ a fixed-length preload buffer. This static approach is inefficient: an overly large buffer in a strong network environment wastes bandwidth, while an insufficient buffer in a weak one leads to playback stuttering or buffering events [6]. Therefore, developing an adaptive and precise video preloading mechanism that can dynamically adjust to network conditions remains an open and significant challenge [7].

An effective adaptive preloading mechanism is distinguished by its ability to proactively adjust its strategy based on predicted future network changes, rather than merely reacting to current conditions. The primary objective of this prediction is to forecast future network performance. In mobile environments, the terminal's received signal strength serves as a key and measurable proxy for this purpose, as a stronger signal typically implies a higher signal-to-interference-and-noise ratio (SINR) and less data loss during transmission [8]. Consequently, a feasible approach to ensure uninterrupted playback on public transport is to adjust the preload strategy based on predicted signal strength along the vehicle's route. Specifically, a larger video segment should be preloaded in advance when the vehicle is about to enter an area with poor signal coverage. Conversely, a smaller cache is sufficient when traversing areas with strong signal strength.

Enabling such an adaptive preloading mechanism requires addressing significant technical challenges. A primary challenge lies in obtaining a fine-grained signal strength map for the target routes [9]. Signal strength is influenced by numerous factors, including atmospheric conditions and physical obstacles like buildings and trees, making accurate estimation a complex task [10], [11], [12]. Although various model-based approaches for radio map estimation exist, such as empirical models [13], dominant path models [14], and ray-tracing models [15], they are often either not sufficiently accurate or are too computationally intensive. This makes them impractical for dynamic public transport scenarios.

Furthermore, accurately predicting the vehicle's real-time position and travel time along its route presents another significant challenge [16], [17]. While the route of a public transport vehicle is fixed, its progress is subject to dynamic traffic conditions, including congestion, road work, and accidents [18]. These factors introduce significant variability, making it difficult to forecast the precise time at which the vehicle will enter a future geographical area. Although numerous methods exist for trajectory prediction, such as Kalman filters [19], kinematic models [20], Monte Carlo methods [21], and various machine learning models, applying them effectively to account for the real-time, unpredictable nature of urban traffic remains a complex problem that requires sophisticated, data-driven analysis.

Beyond the technical challenges of signal and trajectory prediction, any system predicated on user location and network data confronts a critical deployment challenge: addressing

Received 27 September 2024; revised 26 July 2025 and 15 September 2025; accepted 27 September 2025. Date of publication 13 October 2025; date of current version 29 December 2025. The associate editor coordinating the review of this article and approving it for publication was R. Birke. *(Corresponding author: Shaowei Wang.)*

The authors are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China (e-mail: dz21230022@smail.nju.edu.cn; mg21230076@smail.nju.edu.cn; wangsw@nju.edu.cn).

Digital Object Identifier 10.1109/TNSM.2025.3620295

inherent privacy concerns. The conventional methodology of centralizing user data collected from sources like GPS and cellular networks poses a significant privacy risk. Such a centralized repository of sensitive travel histories not only reveals user activity patterns but also becomes vulnerable to data breaches and misuse. Therefore, the core challenge is to design a mechanism that can learn from collective experience for adaptive preloading, without necessitating the continuous transmission of sensitive location data to a central server.

In this work, we propose VIDTRA (Video preloading based on Trajectory prediction), a video preloading system designed to improve the viewing experience for public transport passengers. To build its predictive models, VIDTRA constructs a network situation map using crowdsourced, anonymized data collected with explicit user consent. This map reflects the cellular coverage information (e.g., cell ID and signal strength) of different locations on the routes. The system then combines this map with on-device trajectory prediction to anticipate when a user will enter a problem area, executing preloading accordingly. This approach preemptively resolves potential video stuttering while respecting user privacy, thereby improving the overall user experience on public transport.

The main contributions of this paper are summarized as follows:

- We present VIDTRA, an innovative video preloading system aiming at preventing video stalls for users on public transport. To the best of our knowledge, VIDTRA is the first to realize video preloading based on predicted future signal strength in a dynamic network environment.
- We propose a novel method to predict future received signal strength by combining a crowdsourced network situation map with real-time trajectory forecasting. This allows the system to anticipate potential buffering events and schedule preloading accordingly.
- We propose a system architecture that decouples model training from real-time operation to protect user privacy. Our design minimizes data exposure by leveraging on-device trajectory prediction for immediate tasks, while utilizing strictly anonymized and aggregated data for the offline construction of the network situation map [22].
- We implement VIDTRA on mobile terminals and evaluate it with real-world experiments. Our experiments show that VIDTRA constructs high-precision network situation maps, accurately predicts user trajectories, and effectively reduces video playback interruptions in areas with weak signal strength.

II. Related Work

The proliferation of mobile devices and the explosion of user-generated content have driven a significant surge in Internet traffic attributed to video streaming [23]. Nowadays, the video streaming is considered as a major research problem due to the increased demand for high video quality [24]. However, if the bandwidth allocation cannot meet the bitrate requirement, the video streams playing process may be stalled, which deteriorates user experience. To address this issue, a vast number of methods have been proposed which primarily operate along two main directions: (i) scalable codec and (ii) Congestion control.

Scalable codec. A pivotal class of video streaming mechanisms employs the adaptable codecs such as H264/M-PEG-4 SVC [25], [26], which afford both spatial and temporal adaptability. Adaptable codecs facilitate modifications to the resolution and frame rate of images without necessitating the re-encoding of the original video content [27]. In contrast to approaches that depend on transcoding, adaptable codecs confer a benefit in terms of reduced computational demands, given that the video undergoes encoding a single time and can be dynamically adjusted through the scalability features of the codec [28]. Nonetheless, the dependence on particular codecs for adaptation constrains the content provider selection to a limited array of codec options.

Congestion control. To cope with the fluctuating resources of the Internet, rate adaptation adjusts the transmission rate to match the network's varying capacity [29]. A crucial aspect of rate adaptation research involves detecting network congestion and assessing available network capacity [30]. In [31], a rate adaptation algorithm that relies on computing the average rate of downloaded segments is proposed. This algorithm dynamically adjusts the bitrate, scaling it up or down according to the available network bandwidth to sustain an acceptable level of video quality. In [32], a stream-switching algorithm is introduced, which encodes the video stream into various formats with distinct qualities and bitrates. Two control mechanisms are implemented in the method: one to manage the video buffer and another to determine the appropriate video representation level amidst changing temporal conditions. In [33], a method known as the adaptive algorithm for streaming over HTTP is introduced for managing bitrate adjustments based on various factors and scenarios. This method is structured around two main stages: an initial rapid phase aimed at boosting the buffer level to a set threshold, and a subsequent stable phase designed to maintain the buffer level above a critical low point to prevent underflow. However, during poor connectivity, the video buffer tends to accumulate segments of inferior quality. Even when network conditions improve, playback continues with these lower-quality segments until they are exhausted, delaying the transition to higher quality video streams.

The aforementioned approaches, as summarized in Table I, share a fundamental limitation: their inherently reactive nature. They can only mitigate the impact of poor connectivity after it has been detected, forcing an undesirable trade-off between playback continuity and video clarity.

To transcend this trade-off, we shift from a reactive to a proactive paradigm by introducing VIDTRA. Our system pioneers a novel method for predicting future signal strength by combining a network situation map with real-time trajectory forecasting. This foresight allows VIDTRA to anticipate potential buffering events and intervene preemptively, scheduling preloading before a user enters an area with weak signal strength. Consequently, VIDTRA is designed to preserve both high video quality and seamless playback.

TABLE I
COMPARISON OF VIDEO STREAMING OPTIMIZATION TECHNIQUES

Dimension	Scalable Codec	Congestion Control	Our Proposed Method
Decision Mechanism	Adapts bandwidth by dropping data layers based on codec features, without re-encoding.	Passively adjusts the video bitrate based on current or historical network conditions.	Proactively adjusts preloading or handover strategies based on future predictions.
Input Data	Utilizes a single, multi-layered encoded video stream.	Leverages historical/real-time network throughput and playback buffer level.	Employs a predicted user trajectory and a network situation map.
Response Timing	Takes effect instantly by selectively sending data layers when a lower bitrate is needed.	Responds after detecting network congestion or a change in the buffer level.	Intervenes proactively prior to the predicted onset of network degradation.
Problem Solved	Delivers varied quality streams without increasing server computational load.	Copes with unpredictable, random bandwidth fluctuations to prevent stalls.	Handles predictable, severe signal changes in high-speed mobile scenarios.
Method Characteristics	Offers low computational overhead, but depends on specific codec support.	Provides wide applicability, but suffers from reactive lag and potential quality-recovery delays.	Features a proactive, forward-looking design that seamlessly handles predictable network outages.

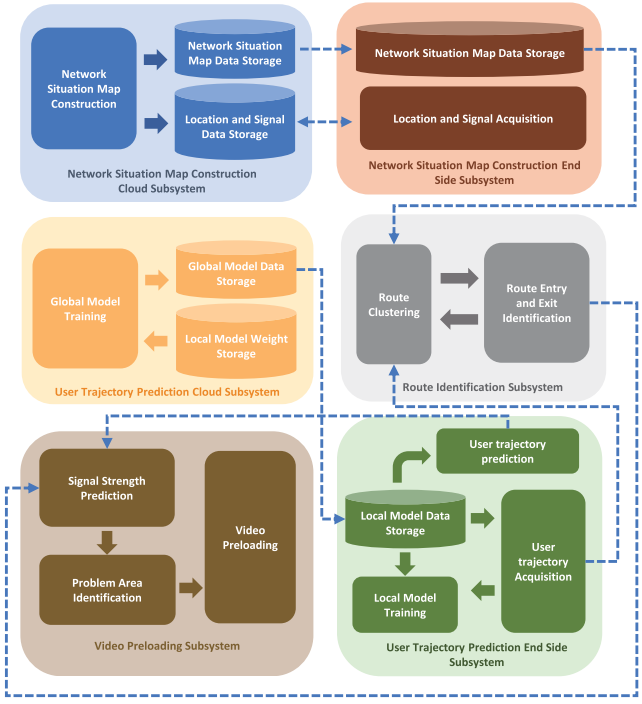


Fig. 1. VIDTRA workflow.

III. SYSTEM ARCHITECTURE

The architecture of VIDTRA is fundamentally designed to protect user privacy. It predicts a user's future signal strength by integrating a network situation map with real-time trajectory prediction, enabling proactive video preloading. As illustrated in Figure 1, the system employs a cloud-side and end-side model but is designed to maximize on-device computation, thereby minimizing sensitive data exposure. The architecture comprises two distinct phases: an offline phase for privacy-preserving map construction and model training, and an online phase where all real-time operations occur exclusively on the end-side.

In the offline phase, the network situation map construction begins, governed by explicit user consent and the principle of data minimization. The end-side application collects the device's geographical location and corresponding cellular signal strength. Before any transmission, this data undergoes a strict on-device anonymization process to remove all personally identifiable information. The resulting anonymized data points are then uploaded to the cloud-side, where they are aggregated from numerous users to construct a comprehensive map of network characteristics across an entire region, ensuring no individual trajectories are stored.

The second offline component, trajectory prediction, employs a Federated Learning (FL) framework to further protect user privacy. This approach is crucial as it ensures a user's raw location history never leaves their device. A local prediction model is trained directly on the end-side using its own data. Subsequently, only the non-sensitive model parameters, not the personal trajectory data itself, are sent to the cloud-side. The cloud-side, in turn, securely aggregates these parameters from many users to build a robust, global prediction model, which is then distributed back to the end-sides for local use.

In the online, real-time phase, all predictive operations are performed on the end-side. The device periodically downloads the latest versions of the global network map and trajectory model from the cloud-side. Using these models, the Route Identification subsystem determines if the user is on a known route, and the Video Preloading subsystem makes its proactive decisions locally. This entirely on-device execution ensures that a user's real-time location and viewing habits are never transmitted to the cloud-side, thus guaranteeing runtime privacy.

IV. NETWORK SITUATION MAP CONSTRUCTION

A network situation map is a representation of signal strengths across different locations, which contains information including GPS, cell ID, frequency bands, received signal strength and route ID [34]. As depicted in Figure 2, the red sections represent areas where the signal strength is weak, resulting in interruptions when users play videos, while the green sections signify areas with strong signal strength, where video playback is relatively smooth.

Typically, network situation maps are acquired through road measurements, which necessitate professional personnel to drive vehicles equipped with signal strength measuring devices along the designated routes, simultaneously recording corresponding GPS data and signal strength information.

Fig. 2. Network situation map. Red indicates weak received signal strength and green indicates strong one.

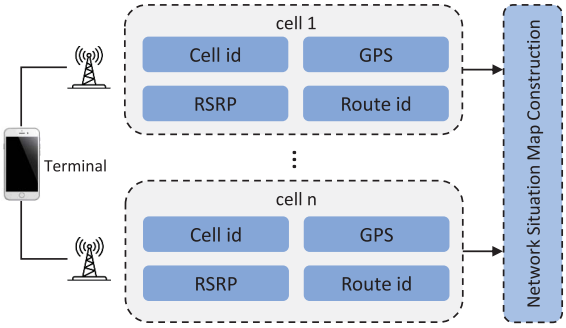


Fig. 3. Network situation map construction workflow.

This measurement approach incurs significant labor and time expenses.

To cope with this issue, we employ a crowdsourcing method to construct network situation maps. Specifically, we need to estimate the received signal strength of all non-measured points based on the measured points. Here, the measured point represents the point where the user provides the information including GPS, cell ID, frequency bands, received signal strength and route ID.

Nowadays, smartphones are equipped with GPS positioning capabilities and have the ability to record the IDs of covering cells and the corresponding reference signal received power (RSRP). The covering cells consist of the currently serving cell tower as well as the IDs of neighboring cells and RSRP is a parameter used to measure the received signal strength. Inspired by this, we utilize the information collected by users to deduce the network situation map along routes as illustrated in Figure 3. Since the terminal can record not only the RSRP of the current serving cell but also of neighboring cells, we construct network situation maps for different cells separately.

Here, we introduce the k-nearest neighbor Gaussian process regression algorithm, referred to as K-GP, to construct the network situation map. Without loss of generality, we assume the route is divided into small segments of five meters each and let N denote the total number of endpoints. The set of endpoints is denoted by $\boldsymbol{X}^* = \{\boldsymbol{x}_n^*\}_{n=1}^N$, where $\boldsymbol{x}_n$ is a two-dimensional coordinate representing the latitude and longitude of the point. Similarly, the set of locations of the sampling points is denoted by $\boldsymbol{X} = \{\boldsymbol{x}_m\}_{m=1}^M$, where M is number of sampling points. Here, the sampling point represents the point at which the user reports the information. The RSRP at location $\boldsymbol{x}_i$ is defined by $R(\boldsymbol{x}_i)$. To reduce the computational overhead, we define $\tilde{\boldsymbol{X}} = \{\boldsymbol{x}_l\}_{l=1}^L$ as the set containing the location of L closest points to the point $\boldsymbol{x}^*$ and $\tilde{\boldsymbol{R}} = \{R(\boldsymbol{x}_l)\}_{l=1}^L$ as the RSRP measurements accordingly. According to the principle of Gaussian process regression, a covariance function can present the correlation between two points. Here we utilize the RBF kernel to represent the covariance function which is expressed as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 exp\left(-\frac{1}{2\beta^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right), \tag{1}$$

where σ_f^2 and β are the hyper-parameters of Gaussian process. The covariance of each endpoints is calculated according to Eq. (1) and we obtain a covariance matrix $K(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}})$, which is given by

$$K = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1L} \\ k_{21} & k_{22} & \cdots & k_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ k_{L1} & k_{L2} & \cdots & k_{LL} \end{bmatrix} \tag{2}$$

The unknown RSRP value R^* of endpoint $\boldsymbol{x}^*$ and signal strength measurements of $\tilde{\boldsymbol{X}}$ satisfy a joint Gaussian distribution, i.e.,

$$\begin{bmatrix} \tilde{\boldsymbol{R}} \\ R^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \tilde{\boldsymbol{\mu}} \\ \mu^* \end{bmatrix}, \begin{bmatrix} K\left(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}\right) + \sigma_n^2 I & K\left(\tilde{\boldsymbol{X}}, \boldsymbol{x}^*\right) \\ K\left(\boldsymbol{x}^*, \tilde{\boldsymbol{X}}\right) & K(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix}\right), \tag{3}$$

where $\tilde{\boldsymbol{\mu}}$ are the signal strength means of sampled points and μ^* are the signal strength mean of the target endpoint. The predicted signal strength of $\boldsymbol{x}^*$ can be obtained by

$$p\left(R^* | \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{R}}, \boldsymbol{x}^*\right) \sim N(u_*, \sigma_*^2), \tag{4}$$

$$u_* = \mu^* + K\left(\boldsymbol{x}^*, \tilde{\boldsymbol{X}}\right)\left[K\left(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}\right) + \sigma_n^2 I\right]^{-1}\left(\tilde{\boldsymbol{R}} - \tilde{\boldsymbol{\mu}}\right), \tag{5}$$

$$\sigma_*^2 = K(\boldsymbol{x}^*, \boldsymbol{x}^*) - K\left(\boldsymbol{x}^*, \tilde{\boldsymbol{X}}\right)\left[K\left(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}\right) + \sigma_n^2 I\right]^{-1} K\left(\tilde{\boldsymbol{X}}, \boldsymbol{x}^*\right), \tag{6}$$

where u_* is the predicted RSRP at location $\boldsymbol{x}^*$.

Considering that each route has a corresponding network situation map, a significant amount of storage space is required to store these maps, which imposes a strain on the user's terminal storage capacity. Consequently, we divide the network situation map subsystem into two parts: the cloud side and the end side. The end-side network situation map subsystem is deployed on the user's terminal device, responsible for constructing the local network situation map for the current route based on the user's GPS data and the RSRP. Whenever the network situation map is updated, the end-side subsystem transmits it to the cloud side. The cloud-side network situation map system, deployed on servers, is tasked with storing the network situation maps of all routes. When the user commences travel on a new route, the cloud-side network status map system transmits the corresponding status map to the user's terminal.

Algorithm 1 Localization Algorithm

```
Initialize α* = 0, S* = ∞, x* = (0,0), C = ∅;
for i = 1 : V do
  for j = 1 : Q do
    if P_i(j,1) = P'(1,1) then
      add P_i to the set C;
    end if
  end for
end for
for (x, P) ∈ C do
  for m = 1 : Q do
    for n = 1 : Q' do
      if P(m,1) = P'(n,1) then
        α = α + 1;
      end if
    end for
  end for
  if α ≥ α* then
    Calculate S as (7);
    α* = α;
    if S < S* then
      S* = S;
      x* = x;
    end if
  end if
end for
return x*
```

V. Trajectory Prediction

For a point p, the network situation map records the RSRP of the Q covering cells, which can be denoted by $\boldsymbol{P} = \{(ID_q, \boldsymbol{R}_q)\}_{q=1}^{Q}$, where ID_q and $\boldsymbol{R}_q$ denote the cell ID and RSRP of the qth cell. There are V points in the map. Based on the network situation map, user location $\boldsymbol{x}'$ can be determined by matching the RSRP measurement $\boldsymbol{P}'$ with the map. Nevertheless, due to the large number of points in the database, an exhaustive search to find the target point is impractical. To address this, we propose a two-stage localization algorithm that utilizes information from the serving cells and neighboring cells.

In the first stage, the algorithm compares the ID of the serving cell with the entries in the network situation map, resulting in a set $\mathcal{C}$ containing potential target points. This step effectively reduces the search space. In the second stage, each $(\boldsymbol{x}, \boldsymbol{P}) \in \mathcal{C}$ is matched with $\boldsymbol{P}'$. Firstly, the number of cells with the same cell ID between $\boldsymbol{P}$ and $\boldsymbol{P}'$ is calculated as α and the maximum α is selected as α^*. Among $\boldsymbol{P}$ with α^*, the sum of RSRP difference between $\boldsymbol{P}$ and $\boldsymbol{P}'$ is calculated as S, given by

$$S = \sum_{k=1}^{\alpha^*} |\boldsymbol{P}(k,2) - \boldsymbol{P}'(k,2)|, \tag{7}$$

and the minimum S is selected as S^*. The element $\boldsymbol{x}$ corresponding to α^* and S^* is identified as the coordinates matched to $\boldsymbol{P}'$. The details of this localization algorithm are summarized in Algorithm 1.

The trajectory of public transport is predictable since its adherence to predefined routes and schedules. Public transport operates on fixed routes, which are carefully planned and publicized, allowing for the prediction of the paths and stopping points. Additionally, public transport follows a schedule to maintain specific times for arrivals and departures from each stop, which adds a level of predictability to the movement.

Neural networks can be effectively used to predict trajectories due to their ability to model complex patterns and relationships within data. Trajectory prediction involves understanding and forecasting the future position of public transport based on various factors such as historical travel times and route specifics. Neural networks are adept at processing sequential and temporal data, making them well-suited for this task [35]. They can learn the intricate dependencies and temporal dynamics inherent in the route data.

DenseNet is an innovative architecture in the field of deep learning, which is introduced to address some of the limitations found in traditional convolutional neural networks [36]. The core of DenseNet is the concept of feature reuse, where each layer receives inputs from all preceding layers and passes its own feature-maps to all subsequent layers. For trajectory prediction, the characteristics of DenseNet ensure that it can learn more complex, nuanced patterns in the data, including temporal dependencies and spatial relationships.

As shown in Figure 4(a), we propose a model based on DenseNet to extract all features from trajectory datasets and is trained on top of the extracted features. The input of the model is the historical trajectory of the user, specifically longitude and latitude. The model begins with two convolutional layers, each followed by a rectified linear unit (ReLU) layer. Convolutional layers are primarily used in processing data with a grid-like topology, which apply a convolution operation to the input, passing the result to the next layer. The ReLU layers introduce non-linearity, allowing the network to handle more complex data patterns. After the ReLU layer are four Dense Block layers. As shown in Figure 4(b), each Dense Block layer has the following internal structure:

Three Convolutional Layers and ReLU Layers: Within each Dense Block, there are three convolutional layers, each followed by a ReLU layer. This structure focuses on extracting and processing complex features from the input data, with each subsequent layer building upon the features identified by the previous ones.

One Concat Layer: After the convolutional and ReLU layers in the Dense Block, there is a concatenation layer. This layer combines the feature maps from the previous layers in the block, preserving feature information and enabling the network to learn more complex patterns.

One Convolutional Layer and One ReLU Layer: Each Dense Block concludes with an additional convolutional layer followed by a ReLU layer, which aims at further refining and processing the features combined in the concat layer. After being added to the input, the tensor is then used as the input of the next Dense Block. Following the Dense Block layers is a concat layer that combines the future of the previous Dense Block layers. Next, several convolution and ReLU layers further process and refine the features. The output of

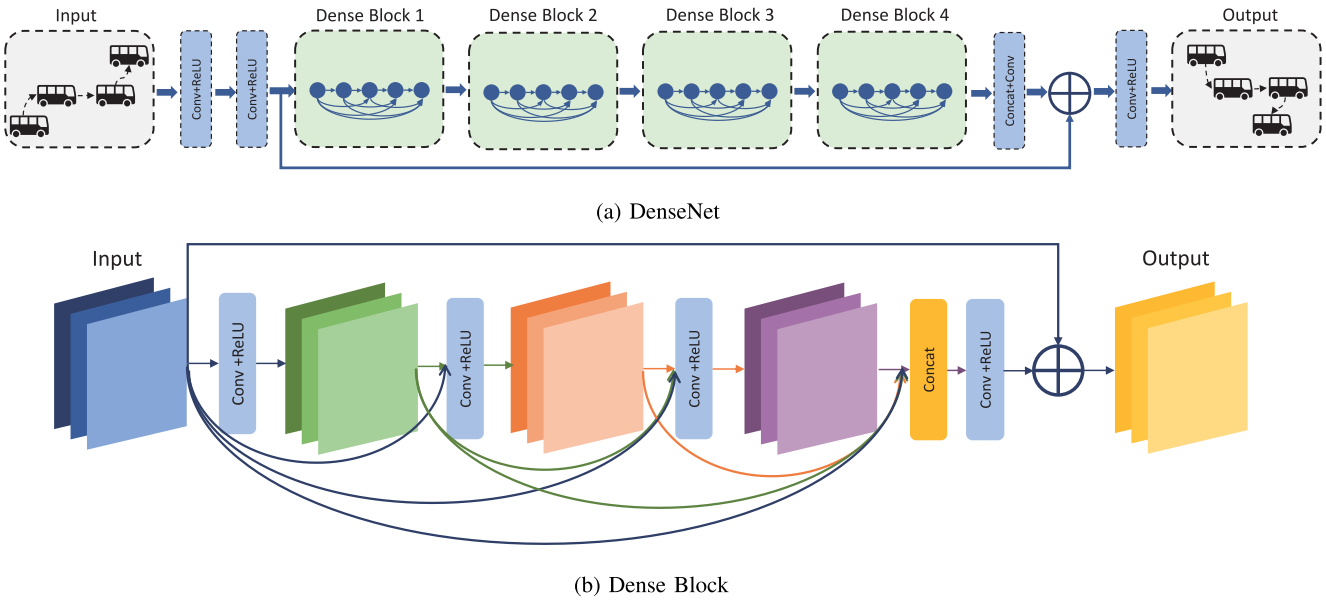


Fig. 4. Architecture of DenseNet.

the model is the future trajectory of the user, represented by longitude and latitude.

We employ the Adam optimizer to tune the parameter of the DenseNet, where the learning rate is 0.01 and the loss function is mean absolute error (MAE). By dynamically adjusting learning rates for each parameter, Adam facilitates efficient convergence, enhancing the model's ability to discern intricate patterns within trajectory data. Additionally, a batch size of 64 is employed to strike a balance between computational efficiency and information richness. The training process spans 500 epochs, ensuring an exhaustive exploration of the predictive capacity. All of these hyperparameters were optimized via a systematic hyperparameter sweep.

The efficient training of DenseNet poses a significant challenge due to the substantial computational resources required. Despite the advancements allowing lightweight neural network training on contemporary mobile terminals, the associated energy consumption remains a concern, resulting in inevitable battery drain. To this end, we propose a bifurcated approach for the trajectory prediction subsystem, segregating it into end (on-device) and cloud components. The end component is deployed on users' smartphones, undertaking preliminary neural network pre-training based on locally collected trajectory data. Subsequently, the learned parameters and trajectory information are transmitted to the cloud component. The cloud, leveraging the amalgamation of diverse users' uploaded parameters and trajectory data, conducts further training to enhance the neural network's predictive capabilities. This dual-sided deployment strategy aims to mitigate energy consumption concerns while optimizing the trajectory prediction performance in a collaborative and distributed manner.

VI. Route Identification

Route identification consists of two main components: (i) Route clustering and (ii) Identification of route entry and exit. Route clustering is employed to assist in constructing the network situational map, while route entry and exit identification contributes to trajectory prediction. When constructing the network situational map, it is essential to consider not only GPS information and RSRP but also knowledge of the route ID. In the context of road testing, obtaining route ID is relatively straightforward. However, when it comes to constructing situational maps for dozens or even hundreds of routes, the human resource cost of road testing becomes excessive. To address this issue, information collection must be facilitated through crowdsourcing. In such scenarios, route ID information is usually missing, thus necessitating the use of route clustering methods to distinguish between different routes.

Density-based spatial clustering of applications with noise (DBSCAN) is a commonly employed density clustering algorithm utilized for identifying clusters within a dataset [37]. Unlike traditional distance-based clustering methods such as K-Means, DBSCAN partitions clusters by seeking density connections among data points, exhibiting robust performance in scenarios characterized by irregular data distributions and higher levels of noise. The core concept of DBSCAN is based on two parameters: ε and ξ, where ε is a distance threshold and ξ represents the minimum number of data points within a neighborhood. Specifically, several important definitions in DBSCAN are shown below: 1) Density: Density refers to how many points are within the radius ε. 2) Core Points: Points that have at least ξ data points within their ε neighborhood. 3) Boundary Points: Points that have fewer than ξ data points within their ε neighborhood but are located within the neighborhood of a core point. 4) Noise Points: Points that are neither core points nor boundary points. 5) Directly Density-reachable: A point P is directly density-reachable from another point Q if P is within the ε-neighborhood of Q and Q is a core point. 6) Density-reachable: A point P is density-reachable from a point Q if there is a chain of points $P_1, P_2, \ldots, P_n$

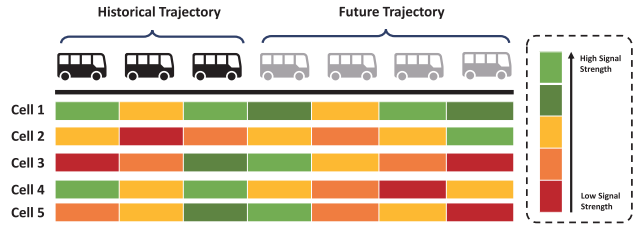


Fig. 5. Low signal strength area identification.

such that $P_1 = Q$ and $P_n = P$, and each point P_{i+1} is directly density-reachable from P_i. This chain effectively forms a bridge from Q to P through dense areas, implying that P belongs to the same cluster as Q.

The procedure for conducting line clustering based on the DBSCAN algorithm is outlined as follows: First, randomly select an unvisited data point and determine whether it is a core point. If the data point is a core one, then designate it as a seed point and extend the cluster through density-reachable relationships. If the point is not a core point, then label it as a noise point. In the case of a boundary point, extend the cluster through density-reachable relationships. The above mentioned process iterates until all data points have been visited.

In the process of trajectory prediction, it is necessary to identify when the user enters and leaves the route, as this informs the activation and deactivation of the trajectory prediction module. Here, we present two methods for route entry and exit identification: one based on cell information and another based on GPS.

GPS based identification: First, we compare the user's GPS data with the GPS information of the network situation map for all routes in the current city. If the distance to a point on the network situation map is less than 10 meters and the service cell of the user matches that at the point, it is inferred that the user is on this route. Subsequently, GPS comparison is conducted based on 200 data points before and after the matched location. If the user's GPS data matches that of the network situation map for 8 seconds within any 10-second window, it is concluded that the user has entered the route. Similarly, if the user's GPS data does not match that of the network situation map for 8 seconds within any 10-second window, it is inferred that the user has exited the route.

Cell information based identification: In scenarios where GPS information is unavailable, it is necessary to utilize cell information such as E-UTRAN cell identifier (ECI), physical cell identifier (PCI) and frequency channel for line identification. First, the ECI of the user's current serving cell is compared with that of the network situation map. Upon successful matching, a retrospective lookup is performed based on M points before and after the matched location. Leveraging the results of the retrospective analysis, we match the cell information of the user's past T seconds with the information of N surrounding points at the retrospective location. Here, a successful match is defined as the PCI and frequency of the cells being identical. Subsequently, we calculate the ratio of successfully matched cells to the total number of user-covered cells, which is defined as the feature coefficient. The user is considered to have entered the route when the feature coefficient becomes greater than I_a, and there are at least T_b instances of the feature coefficient surpassing I_b for the subsequent T_a duration. Similarly, the user is considered to have exited the route when the feature coefficient is less than I_c, and there are at least T_b instances of the feature coefficient equaling 0 during the subsequent T_a period.

VII. Low Signal Strength Area Identification and Video Preloading

The predicted positions are correlated with the network situation map of the route which contains the RSRP for various locations along the route. As shown in Figure 5, by cross-referencing the future positions of the vehicle with the network situation map, the system can efficiently retrieve the expected signal strength for each coverage cell that the vehicle is anticipated to traverse.

Typically, a user position is enveloped by the coverage of two or more base stations, among which the one with the strongest signal strength is often designated as the serving cell. Nonetheless, the prognostications derived from trajectory prediction and the analysis of network situation maps do not yield definitive clarity regarding the identity of the serving cell at a user's prospective position. Thus, we posit an operational definition: If the RSRP of all coverage cells at the location fall below the threshold of –100 dBm, then the signal strength at that point is considered to be weak. The core of this system lies in its ability to preemptively identify areas along the route where the signal strength is predicted to fall below a predefined threshold, indicative of potential video streaming disruptions or video jams. Upon identifying such weak signal strength areas, the system proceeds to calculate the duration for which the user will be within these problem areas, considering the speed and trajectory. In the final stage, leveraging the calculated time intervals during which the user is likely to experience poor signal quality, the algorithm proactively determines the optimal video preloading time. The preloading mechanism is designed to ensure that sufficient video content is buffered on the user's device before the vehicle enters the signal-compromised area, thereby mitigating the impact of weak signal strength on the video streaming experience.

VIII. Evaluation

A. Implementation

We implement VIDTRA on a mobile device with the model OnePlus 10 Pro, NE2210. The device supports both 4G and 5G standards. Given the current trend where 5G is emerging as the mainstream communication system, our tests mainly focus on the 5G communication mode. For our system evaluation, we collected data across two public bus routes in Shenzhen: M562 and M176. We conducted five data collection trips on each route, yielding a total of ten distinct trajectories. This data was used to test all system components, including network situation map construction, user trajectory prediction, route identification, and video preloading.

To deploy the neural network model for user trajectory prediction on the mobile phone, we convert the trained model

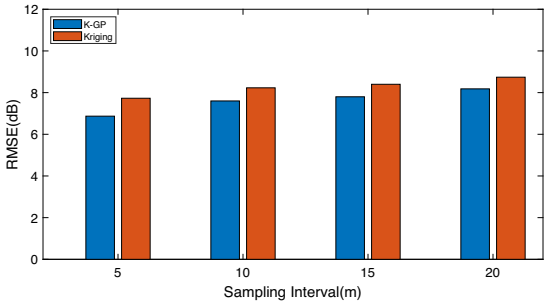
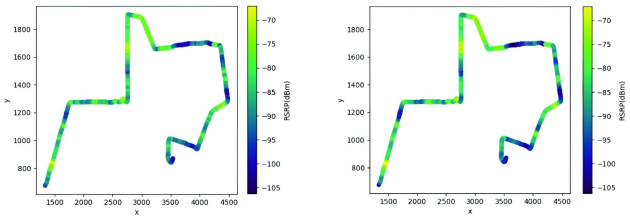


Fig. 6. Accuracy of Network Situation Map.



(a) Original network situation map

(b) Estimated network situation map

Fig. 7. Original and estimated network situation map.

into TensorFlow Lite format, which is optimized for mobile devices. Then we integrate the TensorFlow Lite interpreter into the mobile application, ensuring compatibility with the Android platform. Finally, we package the application with the embedded model for seamless deployment and efficient execution of the neural network directly on the mobile phone.

B. Network Situation Map Results

Figure 6 presents a comparison of the root mean square error (RMSE) for the network situation maps constructed by our proposed K-GP method and the conventional Kriging method [38], under different sampling intervals. While the RMSE for both methods increases with the sampling interval due to sparser data, our K-GP method demonstrates a clear advantage. Across the entire range of sampling intervals, the RMSE of K-GP is consistently 0.6–0.8 dB lower than that of the Kriging method, showcasing its superior interpolation accuracy for building network situation maps. We wish to clarify that the 5 and 20 meters interval in Figure 6 refers to the granularity of the crowdsourced data points used for building our network situation maps, not the real-time frequency at which a user's device actively samples or recalculates its map position.

Our proposed method achieves a sufficiently low RMSE, enabling proactive video preloading. This accuracy is particularly significant given the volatile network conditions on transit routes, where signal strength can fluctuate by 30-40 dB [39]. Crucially, our model captures the overall signal trend while identifying critical transitions between strong and weak connectivity zones.

Specifically, we select a specific bus route to demonstrate the accuracy of the constructed network situation map as depicted in Figure 7. It can be seen from the figure that the estimated network situation map aligns closely with the actual one, effectively capturing variations in signal strength along the bus route. For all subsequent experiments, the sampling interval is set to 5m.

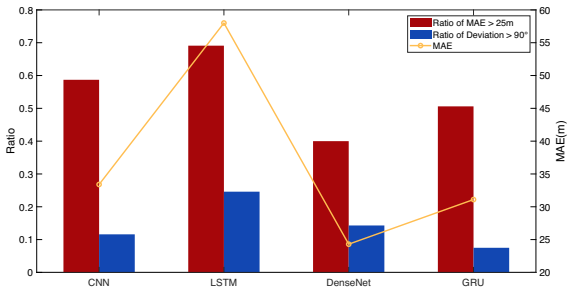


Fig. 8. Trajectory accuracy of different algorithms.

C. Trajectory Prediction Results

In the experiment, we assess the accuracy of user trajectory prediction. The neural network input consists of positional coordinates from the past 12 seconds, while the output comprises positional coordinates for the subsequent 24 seconds. The training dataset is collected from the public bus routes M562 and M176 in Shenzhen, totaling 73,154 instances. Additionally, the validation dataset comprises 4,997 instances gathered from route M176. The predictive accuracy is primarily assessed through the measurement of the mean error. Let N represents the total number of instances in the training dataset, $(\hat{x}_i, \hat{y}_i)$ denotes the predicted positional coordinates for the i-th instance, and (x_i, y_i) represents the true positional coordinates. The mean error can be expressed as follows:

$$MAE = \frac{\sum_{i=1}^{N} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}}{N}. \tag{8}$$

The directional vector for the i-th predicted position can be represented as follows:

$$\hat{d}_i = (\hat{x}_{i+1} - \hat{x}_i, \hat{y}_{i+1} - \hat{y}_i). \tag{9}$$

Similarly, the directional vector for the i-th actual position can be expressed as follows:

$$d_i = (x_{i+1} - x_i, y_{i+1} - y_i). \tag{10}$$

Then the angular deviation between the i-th predicted position and the actual one can be represented as:

$$\theta_i = \cos^{-1}\left(\frac{\hat{d}_i \cdot d_i}{|\hat{d}_i||||d_i|}\right). \tag{11}$$

As shown in Figure 8, DenseNet exhibits the lowest average prediction error, while the GRU demonstrates the lowest direction error rate. The LSTM exhibits the highest average prediction error and direction error rate. It can be observed that DenseNet attains the highest prediction accuracy among the considered models.

Generally, trajectories within steering regions are more challenging to predict than those in straight sections. As illustrated

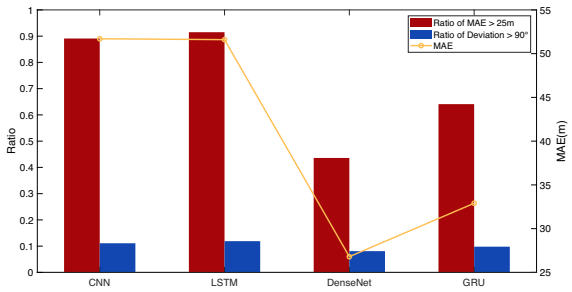


Fig. 9. Steering area prediction accuracy.

TABLE II
NEURAL NETWORK PARAMETERS AND TRAINING TIME

Mode	Training Time for One Epoch	Parameter Quantity
CNN	9s	31570
LSTM	36s	141058
DenseNet	23s	111438
GRU	47s	57090

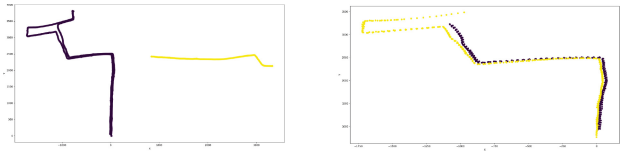
in Figure 9, DenseNet exhibits the lowest average prediction error and direction error rate within the steering regions, while CNN and LSTM show comparatively higher values for both metrics. Notably, DenseNet achieves the highest prediction accuracy in the turning regions.

The parameter quantities and training times for one epoch of different neural networks are illustrated in Table II. The training was conducted on a CPU, specifically the 11th Gen Intel Core i7-11700K, and a GPU, namely the NVIDIA GeForce RTX 3070. As can be seen from the table, CNN exhibits the lowest parameter count and the shortest training duration, whereas LSTM possesses the highest parameter count, and GRU incurs the longest training time.

D. Route Identification Results

We first test the clustering outcomes of routes based on GPS information. ε is set to 20 and ξ is set to 3. The discrimination results for two distinct bus routes are shown in Figure 10(a), from which can be observed that these two routes were successfully differentiated. Figure 10(b) presents the clustering results for the round-trip routes of the same line are illustrated. It can be seen from the figure that the data pertaining to the round-trip directions can be accurately clustered into two categories.

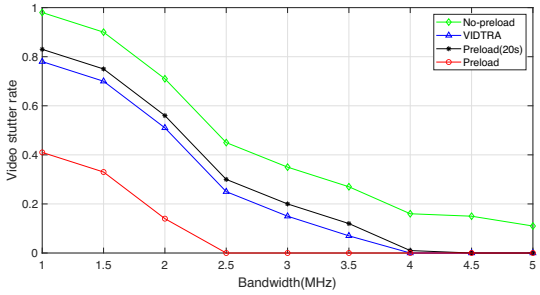
Additionally, we select data spanning approximately 40 minutes from the M176 and M562 bus routes to simulate user entries and exits at various time points. First, we assess the route identification algorithm based on cell information. We set the time window T to 30 seconds, set M to 200 and N to 60. Additionally, we set the feature coefficient threshold I_a for entering the route to 0.5 and the threshold I_b to 0.8. The feature coefficient threshold I_c for exiting the route is set to 0.2. Time window T_a and T_b are set to 10s and 8s respectively. As depicted in scenarios 1 to 5 in Table III, discrepancies of 10-100s are observed in the determination of user entry and exit timings, while location determinations



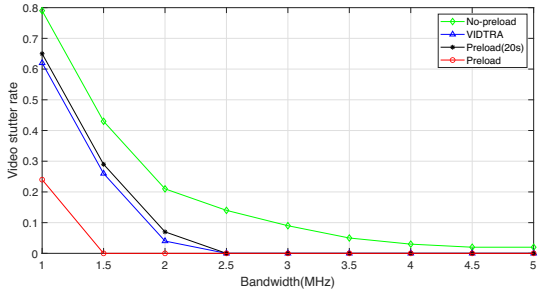
(a) Route clustering of two different routes.

(b) Route clustering of round-trip routes.

Fig. 10. Route clustering based on DBSCAN.



(a) 1440p.



(b) 1080P

Fig. 11. Video stutter rate as a function of bandwidth.

exhibit errors ranging from 10-400m for both entry and exit points on the bus routes.

Scenarios 6 and 7 in Table III present the accuracy of route identification based on GPS information. It can be seen from the table that there is a time error of 0-50 seconds for determining the moments of entering and leaving the bus route, and a location error of 0-50 meters for determining the positions of entering and leaving the bus route. In summary, the accuracy of route identification based on GPS information surpasses that based on cell information. It is noteworthy that in real-world scenarios, situations where GPS information is obtainable are more common than those where it is not. Therefore, GPS information based route identification can exhibit greater generality than the cell information based method.

E. Video Preloading Results

We verify the impact of different video preloading strategies on the reduction of video stuttering rates for bus passengers watching videos. The video stuttering rate is an effective performance indicator that quantitatively measures the frequency and severity of playback disruptions. Figure 11

TABLE III
ROUTE ENTRY AND EXIT IDENTIFICATION

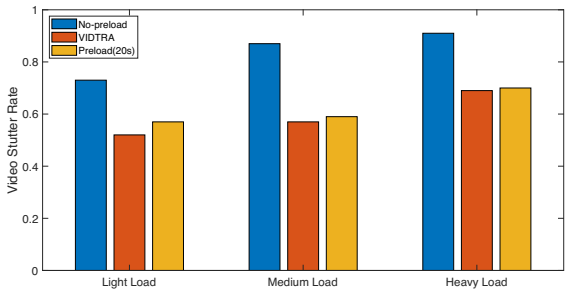
Scenario	Entry Time	Assessed Entry Time	Entry Distance Error	Exit Time	Assessed Exit Time	Exit Distance Error
Scenario 1 (Cell)	101s	124s	90m	2409s	2438s	27m
Scenario 2 (Cell)	201s	180s	1m	2309s	2337s	17m
Scenario 3 (Cell)	301s	291s	59m	2209s	2268s	240m
Scenario 4 (Cell)	401s	291s	228m	2109s	2146s	200m
Scenario 5 (Cell)	501s	454s	178m	2009s	2067s	472m
Scenario 6 (GPS)	101s	101s	0.8m	1979s	2022s	39m
Scenario 7 (GPS)	201s	209s	2m	1879s	1926s	45m

presents a comparative analysis of the stuttering rates for 1440p and 1080p videos under four distinct scenarios: no preloading, the Aggressive Preloading (Preload) strategy, the Constant Buffer (Preload(20s)) strategy, and our proposed system, VIDTRA.

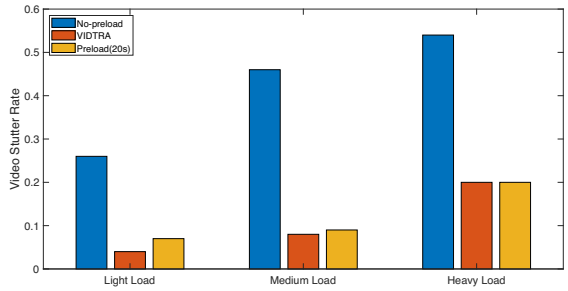
As illustrated, all three preloading strategies yield a substantial reduction in stuttering compared to the baseline with no preloading. Among these strategies, aggressive preloading strategy consistently provides the lowest stuttering rate, effectively serving as the performance benchmark. The constant buffer strategy also offers significant improvements, though it is less robust in low-bandwidth conditions. VIDTRA demonstrates highly competitive performance, achieving a stuttering rate slightly lower than the constant buffer approach and closely matching the performance of the aggressive strategy across the tested bandwidth range. Notably, for 1080p videos, the preloading strategies can achieve zero stuttering when the bandwidth exceeds 2MHz, highlighting the upper limit of performance for preloading systems.

Beyond stuttering performance, we now evaluate the bandwidth efficiency of each strategy. The aggressive preloading strategy, while effective at minimizing stuttering, operates indiscriminately by continuously downloading data regardless of network conditions, which results in the highest potential for bandwidth waste and a constant high overhead. The constant buffer strategy improves upon this by setting a fixed cap on preloaded data, which makes it more efficient by design but less resilient to unpredictable, long-duration network outages. Our proposed method, VIDTRA, achieves the greatest efficiency by employing a targeted preloading logic driven by future predictions. Unlike the aggressive approach, it avoids unnecessary data consumption in good network conditions. It also surpasses the constant buffer strategy by dynamically adjusting the preloading duration based on a forecasted need, thus ensuring the bandwidth cost is directly tied to the necessity of protecting the user experience, rather than being a constant, fixed overhead. This conditional and predictive approach allows VIDTRA to balance the high QoE of aggressive preloading with the efficiency of the constant buffer strategy, offering a superior compromise.

The video stuttering rate under different load conditions is illustrated in Figure 12, where a low load refers to a physical resource block (PRB) load below 20%, a medium load indicates a PRB load above 20% but below 40%, and a high load denotes a PRB load above 40% yet below 60%. While the stuttering rate for both preloading strategies increases with the load, the results confirm that VIDTRA consistently outperforms the constant buffer strategy across all conditions and resolutions. The performance gap between VIDTRA and the constant buffer strategy is most significant under low and high load scenarios. These findings underscore VIDTRA's superior adaptability and its significant effect in reducing playback interruptions for bus passengers, regardless of the network's congestion level.



(a) 1440p.



(b) 1080P

Fig. 12. Video stutter rate under different loads.

IX. CONCLUSION

In this paper, we designed a video preloading system that integrates the network situation map with trajectory prediction to determine the length of video preloading based on the predicted signal strength. The proposed system takes into account both the smoothness and clarity of the video, while avoiding the waste of bandwidth. Numerical results demonstrate the efficacy of VIDTRA in reducing video interruptions and enhancing the user experience of watching videos on public transport.

REFERENCES

[1] "Cisco visual networking index: Forecast and methodology, 2013–2018," White Paper, Cisco, San Jose, CA, USA, 2013.

[2] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. ACM SIGCOMM*, Aug. 2007, pp. 1–14.

[3] S. Chikkerur, V. Sundaram, M. Reisslein, and L. J. Karam, "Objective video quality assessment methods: A classification, review, and performance comparison," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 165–182, Jun. 2011.

[4] Y. Yeznabad, M. Helfert, and G.-M. Muntean, "QoE-driven cross-layer bitrate allocation approach for MEC-supported adaptive video streaming," *IEEE Trans. Netw. Services Manag.*, vol. 21, no. 6, pp. 6857–6874, Dec. 2024.

[5] S. Jabbar et al., "Developing a video buffer framework for video streaming in cellular networks," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–13, Jun. 2018.

[6] F. Dobrian et al., "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 362–373, Aug. 2011.

[7] S. Kumar, R. Devaraj, A. Sarkar, and A. Sur, "Client-side QoE management for SVC video streaming: An FSM supported design approach," *IEEE Trans. Netw. Services Manag.*, vol. 16, no. 3, pp. 1113–1126, Sep. 2019.

[8] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer state is enough: Simplifying the design of QoE-aware HTTP adaptive video streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 590–601, Jun. 2018.

[9] T. Van Chien, T. N. Canh, E. Björnson, and E. G. Larsson, "Power control in cellular massive MIMO with varying user activity: A deep learning solution," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 5732–5748, Sep. 2020.

[10] R. Levie, Ç. Yapar, G. Kutyniok, and G. Caire, "RadioUNet: Fast radio map estimation with convolutional neural networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 4001–4015, Jun. 2021.

[11] Z. Tao and S. Wang, "Improved downlink rates for FDD massive MIMO systems through Bayesian neural networks-based channel prediction," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 2122–2134, Mar. 2022.

[12] L. Shen, Y. Zhang, and S. Wang, "CodeBook based antenna configuration: A new network planning paradigm for mmWave mobile communication systems," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 10368–10379, Aug. 2023.

[13] M. Iskander and Z. Yun, "Propagation prediction models for wireless communication systems," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 3, pp. 662–673, Mar. 2002.

[14] R. Wahl and G. Wolfle, "Dominant path prediction model for urban scenarios," in *Proc. IST MWC Summit*, Sep. 2005, pp. 1–6.

[15] M. Raspopoulos, C. Laoudias, L. Kanaris, A. Kokkinis, C. G. Panayiotou, and S. Stavrou, "3D ray tracing for device-independent fingerprint-based positioning in WLANs," in *Proc. IEEE WPNC*, Mar. 2012, pp. 109–113.

[16] W. Treethidtaphat, W. Pattara-Atikom, and S. Khaimook, "Bus arrival time prediction at any distance of bus route using deep neural network model," in *Proc. IEEE ITSC*, Oct. 2017, pp. 988–992.

[17] H. Tong, T. Wang, Y. Zhu, X. Liu, S. Wang, and C. Yin, "Mobility-aware seamless handover with MPTCP in software-defined HetNets," *IEEE Trans. Netw. Services Manag.*, vol. 18, no. 1, pp. 498–510, Mar. 2021.

[18] J. Ma, J. Chan, G. Ristanoski, S. Rajasegarar, and C. Leckie, "Bus travel time prediction with real-time traffic information," *Transp. Res. C, Emerg. Technol.*, vol. 105, pp. 536–549, Aug. 2019.

[19] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended Kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," in *Proc. IEEE ACC*, Jul. 2007, pp. 1805–1810.

[20] H. Cui et al., "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *Proc. IEEE ICRA*, Paris, France, May 2020, pp. 1–7.

[21] I. Lymperopoulos and J. Lygeros, "Sequential monte carlo methods for multi-aircraft trajectory prediction in air traffic management," *Int. J. Adapt. Control Signal Process.*, vol. 24, no. 10, pp. 830–849, Apr. 2010.

[22] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, Apr. 2017, pp. 1–10.

[23] A. Yaqoob and G.-M. Muntean, "FReD-ViQ: Fuzzy reinforcement learning driven adaptive streaming solution for improved video quality of experience," *IEEE Trans. Netw. Services Manag.*, vol. 21, no. 5, pp. 5532–5547, Oct. 2024.

[24] K.-C. Yang, C. C. Guest, K. El-Maleh, and P. K. Das, "Perceptual temporal quality metric for compressed video," *IEEE Trans. Multimedia.*, vol. 9, no. 7, pp. 1528–1535, Nov. 2007.

[25] C. Krasic, J. Walpole, and W.-C. Feng, "Quality-adaptive media streaming by priority drop," in *Proc. ACM NOSSDAV*, Monterey, CA, USA, Jun. 2003, pp. 112–121.

[26] R. Kuschnig, I. Kofler, and H. Hellwagner, "An evaluation of TCP-based rate-control algorithms for adaptive Internet streaming of H.264/SVC," in *Proc. ACM MMSys*, Feb. 2010, pp. 157–168.

[27] L. Pozueco, X. G. Pañeda, R. García, D. Melendi, and S. Cabrero, "Adaptable system based on scalable video coding for high-quality video service," *Comput. Electr. Eng.*, vol. 39, no. 3, pp. 775–789, Apr. 2013.

[28] T. Laude, Y. G. Adhisantoso, J. Voges, M. Munderloh, and J. Ostermann, "A comprehensive video codec comparison," *APSIPA Trans. Signal Inf. Process.*, vol. 8, pp. 1–16, Nov. 2019.

[29] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Favor: Fine-grained video rate adaptation," in *Proc. ACM MMSys*, Jun. 2018, pp. 64–75.

[30] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. ACM CoNEXT*, Dec. 2012, pp. 109–120.

[31] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. ACM MMSys*, San Jose, CA, USA, Feb. 2011, pp. 169–174.

[32] L. De Cicco and S. Mascolo, "An adaptive video streaming control system: Modeling, validation, and performance evaluation," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 526–539, Apr. 2014.

[33] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Proc. IEEE PV*, May 2012, pp. 173–178.

[34] Y. Zhang and S. Wang, "K-nearest neighbors Gaussian process regression for urban radio map reconstruction," *IEEE Commun. Lett.*, vol. 26, no. 12, pp. 3049–3053, Dec. 2022.

[35] S. Liu, T. Wang, and S. Wang, "Hardware impairment estimation in NB-IoT: A parallel multitask learning method," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6859–6869, Apr. 2023.

[36] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," Apr. 2014, *arXiv:1404.1869*.

[37] E. Schubert, J. Sander, M. Ester, H. Peter Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Jul. 2017.

[38] G. Boccolini, G. Hernandez-Penaloza, and B. Beferull-Lozano, "Wireless sensor network for spectrum cartography based on Kriging interpolation," in *Proc. IEEE PIMRC*, Nov. 2012, pp. 1565–1570.

[39] A. Kulkarni, A. Seetharam, A. Ramesh, and J. D. Herath, "DeepChannel: Wireless channel quality prediction using deep learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 443–456, Jan. 2020.

Jiaen Lv (Graduate Student Member, IEEE) received the B.S. degree from Nanjing University, Nanjing, China, in 2021, where he is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering. His current research interests include mobility management, edge computing and machine learning.

Yifang Zhang received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2021, and the M.S. degree from the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, in 2024. Her research interests include wireless network optimization and machine learning.

Shaowei Wang (Senior Member, IEEE) received the Ph.D. degree from Wuhan University, Wuhan, China, in 2006. He joined the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, as a Faculty Member, in 2006, where he is currently a Full Professor. From 2012 to 2013, he was a Visiting Scholar/a Professor with Stanford University, Stanford, CA, USA, and The University of British Columbia, Vancouver, BC, Canada. His research interests include communications and networking, operations research, and machine learning.