

Real-Batch: Real-Time Adaptive Batch Processing for Accurate Object Detection in Autonomous Driving

Tianen Liu¹, Shuai Wang², *Member, IEEE*, Borui Li³, *Member, IEEE*, Zheng Dong⁴, *Member, IEEE*, Guang Wang⁵, Wei Gong⁶, *Member, IEEE*, and Tian He⁷, *Fellow, IEEE*

Abstract—Video object detection stands as a pivotal element within the burgeoning landscape of autonomous driving systems. The exigency to fulfill stringent real-time requisites, while upholding both precision and efficiency in detection, underscores its significance. Although extant methodologies enhance either accuracy or efficiency through the exploitation of spatio-temporal inter-dependencies within the video context, their propensity to conduct detection on discrete frames begets superfluous computations and curbed real-time efficacy. This paper introduces a pioneering approach, called Real-Batch, tailored explicitly to redress this quandary. Real-Batch ingeniously processes batches of video frames uniformly, effectually winnowing out repetitive object detection occurrences. Our methodology is rigorously evaluated on the real-world datasets, scrutinizing four key metrics: accuracy, efficiency, informational value, and adherence to timing constraints. The comprehensive findings substantiate that Real-Batch yields an unparalleled maximal surge in accuracy and efficiency, increasing of 4.2%-13.2% and 24.7%-43.9%, respectively, offering promising advancements for autonomous driving systems.

Index Terms—Real-time scheduling, video object detection, autonomous driving.

I. INTRODUCTION

WITH the rapid development of sensing technology and mobile computing platform, autonomous driving and other advanced driving assistance system (ADAS) receive great attention from smart cities [1]. Among various components of autonomous driving, vision-based object detection is definitely an important building block. It serves as the foundation of

obstacle avoidance and route planning and is more cost-effective than other sensing technologies such as mmWave, LiDAR, and ultra-sonic. For example, Tesla achieves full self-driving (FSD) based on object detection with eight cameras [2]. Moreover, in autonomous driving, object detection has a strict deadline to make sure the vehicle could make decisions and react to the emergency as quickly as possible. However, the computing resources of vehicles are limited, making it difficult to process video streams accurately in real-time [3].

Existing works leverage optical flow [4], [5] or object matching [6], [7] to improve the timelines of video object detection. However, these approaches take video streams as continuous frames and perform single-frame detection on them, resulting in a large number of redundant detection of the repeated objects in video streams. Another way to reduce the detection latency is to offload some of the computation to a nearby edge device [8], [9]. Nevertheless, in the autonomous driving scenario, existing edge computing infrastructure is not mature enough to support real-time offloading due to the high mobility of vehicles [10].

In order to process enough frames for more information on the surrounding environment while satisfying the decision deadline, *batching* the input frames and *eliminating* the redundant objects being detected seem to be a straightforward solution. However, applying batching and object elimination encounters three formidable questions, especially in autonomous driving scenario. (i) For the batched inference, how can we determine the batch size in such a highly variable environment of autonomous driving? To meet the decision deadline, our system should consider the tradeoff between the waiting latency to form a batch and the inference latency. (ii) For the redundant detection elimination, how should we set up the appropriate elimination quantity under different situations? The prevailing object detection algorithm first generates abundant detection proposals and then performs detection for each proposal. The more proposals eliminated, the faster detection will be. However, filtering too many proposals will lower the detection accuracy. (iii) Moreover, how should we select the proposals inside a batch to be eliminated? We should keep the unique objects and try to filter the identical objects in different frames in a batch.

To answer the above three challenging questions, we propose *Real-Batch* – an adaptive batch processing system for object detection in autonomous driving. To be more specific, Real-Batch incorporates several novel design elements: (1) To ensure that

Received 18 December 2023; revised 8 December 2024; accepted 13 October 2025. Date of publication 23 October 2025; date of current version 4 February 2026. This work was supported by the National Natural Science Foundation of China under Grant U24B20152, Grant 62272098, and Grant 62302096. Recommended for acceptance by Z. Li. (*Corresponding author: Shuai Wang.*)

Tianen Liu, Shuai Wang, Borui Li, and Tian He are with the Key Laboratory of Computer Network and Information Integration, School of Computer Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: toliutianen@seu.edu.cn; shuaiwang@seu.edu.cn; libr@seu.edu.cn; tianhe@seu.edu.cn).

Zheng Dong is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: dong@wayne.edu).

Guang Wang is with the Department of Computer Science, Florida State University, Tallahassee, FL 32306 USA (e-mail: guang@cs.fsu.edu).

Wei Gong is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230052, China (e-mail: weigong@ustc.edu.cn).

Digital Object Identifier 10.1109/TMC.2025.3625072

video frames are processed within the deadline, we introduce a *Deadline-driven Adaptive Batch Scheduling* mechanism. This mechanism predicts the number of proposals and the latency for the next batch of video frames, enabling the system to select an appropriate batch in a timely manner. (2) To recognize the regularity of object changes between consecutive video frames, we propose a *Batch Proposal Density-based Discriminator*. It automatically classifies the driving scenario into sparse, normal, or dense categories, taking into account the density of objects in the frames. (3) Given the significant number of proposals generated by the Region Proposal Network (RPN), we propose a *Determinant Point Process-based Proposal Filter*, which models the selection process of a subset from all proposals using the Determinantal Point Process (DPP). Additionally, we propose the Greedy Selection Batch Proposal (GSBP) algorithm, which expedites the selection process by adding multiple proposals to the candidate set simultaneously. It is worth noting that we design Real-Batch to be more adaptive and can serve as a *plug-and-play* system for all two-stage object detection algorithms.

We implement Real-Batch with five object detection algorithms and compare with three scheduling approaches on the UA-DETRAC (University at Albany DETection and TRACKing) dataset [11]. Experiments show that Real-Batch is superior to the other three scheduling approaches in accuracy, efficiency, information value, and missing deadline ratio. Compared with traditional networks, Real-Batch with Faster RCNN gains $1.53 \times$ efficiency improvement when an autonomous driving car has five cameras with an acceptable accuracy decrease.

- We first propose batch processing to improve the efficiency of video object detection in the autonomous driving scenario. Real-Batch is fundamentally different from previous object detectors that detect each proposal. Real-Batch is a middleware for video object detection, which is suitable for region-based video object detection frameworks.
- We pack video frames into a batch for input to the detection network to avoid detecting each frame. Accounting for the complex variability of autonomous driving environments, we utilize the batch proposal density to distinguish scenarios. In order to improve detection efficiency, we use the relationship of proposals between adjacent frames to filter out different batch objects and reduce redundancy calculation.
- We use real-world autonomous driving datasets to verify the performance of our method and other state-of-the-art algorithms. Experimental results show that our proposed Real-Batch is effective in ensuring detection accuracy and also greatly improves the detection efficiency at the same time.

Section II presents the motivation behind Real-Batch, offering insights into the challenges it addresses. Section III provides an overview of the framework, outlining its key components and design principles. Section IV delves into the methodology and technical design of Real-Batch in detail. Section V describes the experimental setup, including the datasets, platform specifications, evaluation metrics, and baselines. Section VI evaluates the results and provides a comprehensive analysis of the performance compared to existing methods. Section VII

discusses open issues of Real-Batch. Section VIII discusses the work and summarizes the differences between Real-Batch and state-of-the-art (SOTA) methods, while Section IX concludes the paper.

II. MOTIVATION

The video object detection problem is usually formulated as follows: given continuous frames, it outputs bboxes (bounding boxes) and categories of the objects contained in these frames separately and evaluates the accuracy of the network model by comparing the detection results of each frame with the ground truth. For example, in an autonomous driving environment, vehicles are bound to encounter sparse and congested roads during a complete travel route. A camera on an autonomous driving car captures 30 frames in a second, and we need to detect each frame within 33 ms. Crucially, the autonomous driving car's decision-making system updates its decisions (braking, accelerating, or decelerating) every 100 ms based on the detection results from video frames [12], [13], [14].

However, current video object detection algorithms achieve detection efficiency of less than 30 FPS (Frame Per Second). For video object detection algorithms based on optical flow, FGFA [4] and DFF [5] have detection efficiency of 1.36 FPS and 20.25 FPS at K40 GPU, respectively. For video object detection algorithms based on tracking, D&T Loss [7] and Track assisted [6] have detection efficiency of 7.8 FPS and 30 FPS at Titan XP GPU, respectively. Therefore, in order to meet the real-time requirements of autonomous driving, many researchers use skipping schedule [15], [16], [17] or keyframe schedule [18], [19] to process video frames before the deadline.

Employing scheduling approaches to improve detection efficiency: An ideal schedule is that each video frame is processed in real-time (less than 33 ms), but the current video object detection algorithms cannot achieve this on mobile GPUs. A naive schedule is to process each video frame regardless of processing time and deadline. Although the video frame is not discarded in this approach, each detection result is from previous remote video frames, therefore the freshness of information is poor. Skipping schedule ensures the freshness of information, but many frames are discarded. If the discarded frame contains important information, this is fatal to security. Keyframe schedule is the same as the skipping schedule, both of which discard many video frames. The difference is that the keyframe schedule first picks out video frames that contain important information.

However, the features of keyframes cannot represent the features of all frames. Not all objects may be included in keyframes. The above approaches [15], [16], [17], [18], [19] are single-frame processing and need to detect each selected frame. Our idea is to package selected frames in a buffer pool together for processing and to improve detection efficiency.

Reducing the redundant object repeated detection to improve detection efficiency: Existing video object detectors usually include two networks: a feature extraction network and a detection network. The feature extraction network uses CNN (Convolutional Neural Network) to extract features from the input frame to generate feature maps. The detection network uses feature

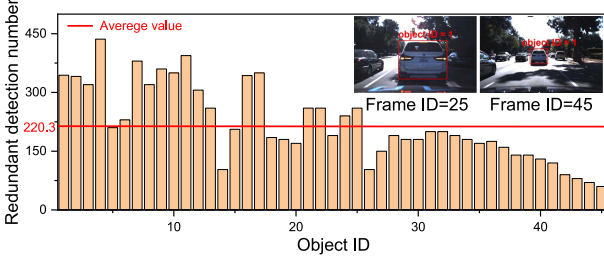


Fig. 1. The number of same detected object ID repeatedly.

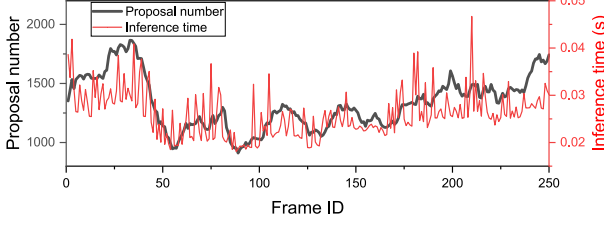


Fig. 2. Relevance of proposal number and inference time.

maps generated by the feature extraction network to detect the category and location of objects. The video consists of continuous frames, with strong continuity between each frame, which causes the network to detect a large number of similar objects in consecutive frames. Actually, this leads to the detection of a lot of redundant information and slows down calculation efficiency.

As shown in the upper right corner of Fig. 1, the same object ID appears in 25th frame and 45th frame. Based on this observation, we count the redundancy detection times of all object IDs in a video, and the average redundancy detection times in this video is 220.3. In addition, we observe the correlation between the number of proposals and inference time. As shown in Fig. 2, the two trends are basically the same, which indicates that there is a certain correlation between the number of proposals and inference time. This is the key motivation behind Real-Batch by filtering a large number of proposals to improve computing efficiency.

Our idea is to pack a series of frames into a batch and extract different batch proposal results uniformly, which reduces the computational cost of the detection network. At the same time, it does not affect the accuracy of video object analysis results. Specifically, Real-Batch first predicts the latency of batch frames and selects a suitable batch size, then Real-Batch processes the frames at once and outputs the proposal of selected frames. Second, Real-Batch does not directly match or detect objects like other detection algorithms, but judges the congestion of the road in a batch of video frames according to the proposal density. Our system handles sparse, normal, and dense cases separately to maximize the balance between accuracy and efficiency. Finally, by uniformly filtering the candidate proposals of batches of video frames, Real-Batch reduces the detection of duplicate objects, thereby greatly improving computational efficiency.

III. SYSTEM OVERVIEW

Fig. 3 shows the architecture of Real-Batch in detail. As we mentioned before, we design Real-Batch as *plug-and-play* building blocks for existing two-stage object detection systems. It is worth noting that Real-Batch can also work with one-stage object detection (OD). We will discuss the adaptivity of Real-Batch in Section VII.

Traditional two-stage object detection algorithms consist of a region proposal stage and a detection stage. Based on the insights in Section II, to achieve better timeliness with batching mechanism and proposal filtering, Real-Batch includes the following three parts: (i) *Deadline-driven Adaptive Batch Scheduling* module, (ii) *Batch Proposal Density-based Discriminator* module, and, (iii) *Determinant Point Process-based Proposal Filter* module.

Before sending video frames to region proposal network, Real-Batch leverages a *Deadline-driven Adaptive Batch Scheduling* module to select the most appropriate batch size of a video frame, so that enough images can be used without missing the decision deadline. The video frames are taken as a whole and input into the region proposal network [20] to obtain the proposal coordinates and scores of video frames. More explanations can be found in Section IV-A.

After proposal generation, Real-Batch perceives the complexity of detection scenario with a *Batch Proposal Density-based Discriminator* by judging the region proposal density and divides the proposal set into three cases: sparse, normal, and dense. The judging proposal coordinates and scores are input into the feature extraction network to generate the feature map of proposals. More explanations can be found in Section IV-B.

Finally, Real-Batch filters out less-important proposals to avoid repeated detection of the same objects to enhance detection efficiency via a *Determinant Point Process-based Proposal Filter* module. We predict bounding box coordinates and object categories by bounding box prediction network and category prediction network. More explanations can be found in Section IV-C.

It is worth noting we do not modify the deep network's layer architecture itself. Instead, we introduce several modules between the existing layers, showcasing the *plug-and-play* nature of Real-Batch. This design ensures that our approach seamlessly integrates with existing two-stage detection frameworks without altering the underlying network structure.

IV. MAIN DESIGN

For describing the detailed design of each module, we define some necessary notations. We use decision timeline $\dots, t, t+1, t+2, \dots$ to denote periodic decisions in autonomous driving cars. Between the two decision intervals, cars capture m video frames. Video frames generate all proposals and scores set through backbone network and RPN network. We define proposal set $P_1^t = \cup_{i=1}^m p_i^t$, where p_i^t represents the proposals of i th video frame in the t th decision. We define score set $S_1^t = \cup_{i=1}^m s_i^t$, which is the foreground probability of proposal, where s_i^t and p_i^t correspond one to one.

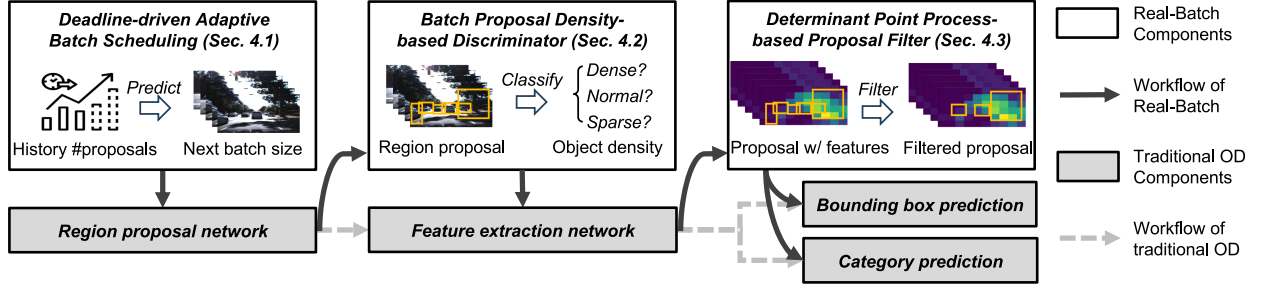


Fig. 3. Architectural overview of Real-Batch.

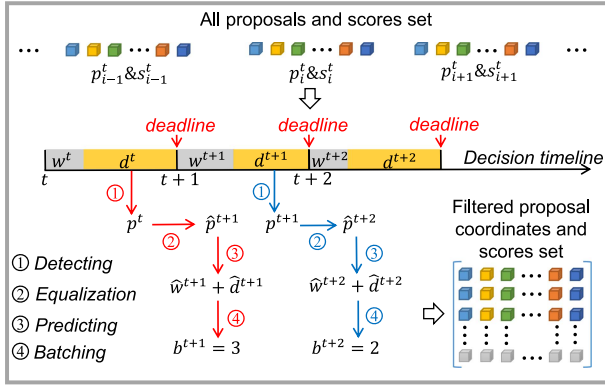


Fig. 4. The framework of the Deadline-driven Adaptive Batch Scheduling module. A gray square represents the filtered proposal and score. Arrows of the same color represent a prediction process.

A. Deadline-Driven Adaptive Batch Scheduling

When the sensing data arrives according to the camera's capture frequency, the submodule is responsible for adaptively batching the video frames and sending them to the detection network. Compared with traditional methods, the adaptive batching video frame is more suitable for autonomous driving scenarios. Traditional object detection systems [8], [10] do not batch video frames and rely on single-frame processing. The method is suitable for the case of low camera acquisition frequency. However, the capture frequency of the car camera is large (at least 30~60FPS) [11], [21]. For regular external driving environments, some autonomous driving systems [22], [23] consider batch video frames into fixed batch sizes. However, the autonomous driving environment is highly dynamic, and the use of fixed-size batches can cause larger time variations, further exacerbating the task timeout.

The key insight of the adaptive batching video frame is the relevance of proposal number and inference time, as is shown in Fig. 2. We predict the batch size of the next decision timeline based on the number of proposals generated in the previous decision timeline. Moreover, we balance the waiting delay and inference delay of video frames by dynamically adjusting the batch size, further reducing the task timeout.

The framework of this module is shown in the Fig. 4. First, we input proposals P_1^t and scores S_1^t generated by video frames captured by the self-driving car between two decision intervals.

Second, we decide which frames in the next decision are eliminated to avoid missing the deadline through the next steps.

- In step 1 (Detecting), the number of proposals $|p_i^t|$ for each frame i in t decision is detected by the RPN network and is defined as ground truth.
- In step 2 (Equalization), we compute the number of proposals for each frame in the $t + 1$ decision using history proposals. Although the number of proposals between adjacent frames is similar, the complex autopilot environment causes the number of proposals for each batch to fluctuate. The number of proposals for each frame in the $t + 1$ decision can be expressed as $|\hat{p}_i^{t+1}| = \sum_{i=1}^m \frac{|p_i^t|}{m}$, where $|p_i^t|$ and $|\hat{p}_i^{t+1}|$ represent the ground truth and predicted number of proposals respectively for each frame. For notational simplicity, we omit the index t and consider the decision process only once.
- In step 3 (Predicting), polynomial regression is used to model K relationships between the batch size and inference delay, where $K = 1, 2, \dots, k, \dots$. The polynomial regression curve is expressed as $\text{PR}_k(\tau_k, P_k)$, where τ_k is k th regression parameters and $P_k = \sum_{i=1}^k |p_i|$ is the ground truth of proposal number when batch size is k . This means that for each batch size, we can construct a real inference delay regression curve. Therefore, we can dynamically batch video frames by selecting a suitable regression curve. We then predict the delay for each batch size by using known regression curve PR_k , i.e., $\hat{d}_i \leftarrow \text{PR}_k(\tau_k, \hat{P}_k)$, where $\hat{P}_k = \sum_{i=1}^k |\hat{p}_i|$. In addition, different batch sizes will also lead to different waiting delays $\hat{w}_i$.
- In step 4 (Batching), we select a batch size from K in this decision timeline. Because the batch processing of Real-Batch inevitably causes video frames to wait, we model the following optimization problem to balance wait delay and inference delay. We model this optimization problem as (1), where ω represents the number of cameras and γ represents the interval of decision deadline. γ is usually set to 100ms in experiments [12], [13], [14].

$$\frac{\text{FPS}}{1000 \times \omega} \times (k - 1) + \text{PR}_k(\tau_k, \hat{P}_k) \leq \gamma. \quad (1)$$

Equation (1) ensures that the total processing time for a batch, including the waiting delay and inference delay, remains within the decision deadline. The key to achieving this lies in the

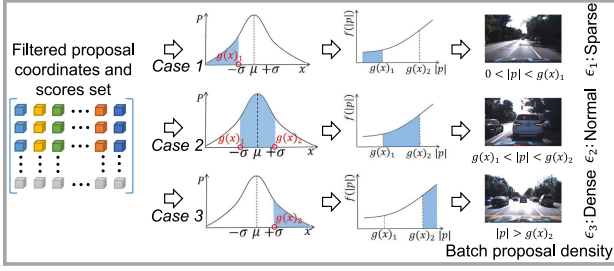


Fig. 5. The framework of the *Batch Proposal Density-based Discriminator* module.

dynamic adjustment of batch size, which is informed by the relationship between the number of proposals and the inference time. This relationship is modeled using a polynomial regression curve PR_k , which predicts the inference delay for different batch sizes based on historical data.

The regression curve plays a crucial role in the batch size selection process. By analyzing the correlation between the proposal count $\hat{P}_k$ and inference latency, $PR_k(\tau_k, \hat{P}_k)$ enables the system to dynamically predict the delay for any given batch size. This predictive capability allows us to optimize the batch size k such that the overall delay, including waiting and inference time, satisfies the deadline constraint γ .

Finally, this module outputs filtered proposals and score set of the next decision timeline. We define filtered proposal set $P_2 = \cup_{i=1}^{m*} p_i$, where p_i represents the proposal of i th video frame, $m*$ represents the number of filtered video frames. In addition, we define filtered scores set $S_2 = \cup_{i=1}^{m*} s_i$, where s_i and p_i correspond one to one.

B. Batch Proposal Density-Based Discriminator

As shown in Fig. 5, *Batch Proposal Density-based Discriminator* module input proposal coordinates set and scores set. The module divides vehicle density into three types according to different driving environments: sparse, normal, and dense. Each type determines the filtering threshold of the next module, which is: ϵ_1 , ϵ_2 , and ϵ_3 .

First, based on an objective fact, the number of vehicles presents a quasi-normal distribution in the real world, that is, too few or too many vehicles have only a small probability of occurrence. Therefore, we compute the critical values of the number of objects and their occurrence probabilities under three types. We assume that the number of objects x in the real world follows a quasi-normal distribution $X \sim N(\mu, \sigma^2)$ with the position parameter μ and the scale parameter σ . This assumption holds when a dataset is sufficiently complete because events that occur with extremely small or large x are low-probability events based on the nature of the normal distribution.

Therefore, the critical values of sparse, normal, and dense objects in video frames can be obtained according to the probability of quasi-normal distribution of object numbers, as shown in (2).

$$g(x)_{1,2} = \begin{cases} g(x)_1 \leftarrow P(0 < x < \mu - \sigma) \\ g(x)_2 \leftarrow P(x > \mu + \sigma) \end{cases}, \quad (2)$$

where $g(x)_1, g(x)_2$ respectively represent the critical values of sparse and dense objects. $P(0 < x < \mu - \sigma)$ and $P(x > \mu + \sigma)$ are known and represent the sparse and dense probability of object x in the real world, respectively.

Second, we model the relationship between the number of proposals and objects. Proposals are generated when video frames pass through the RPN network and the object is proposed according to the intermediate results of proposals in the IoU (Intersection Over Union) process. For the same IoU process, the more objects in the video frame, the more proposals are generated.

Therefore, we model batch proposal density by building the relationship between objects and proposals, as shown in (3).

$$f(|p|)_{1,2,3} = \begin{cases} f(|p|)_1 \leftarrow 0 < |p| < g(x)_1, \text{ sparse} \\ f(|p|)_2 \leftarrow g(x)_1 \leq |p| < g(x)_2, \text{ normal} \\ f(|p|)_3 \leftarrow |p| \geq g(x)_2, \text{ dense} \end{cases} \quad (3)$$

where $|p|$ represents the number of batch proposals in the video frames. $f(|p|)_1, f(|p|)_2$ and $f(|p|)_3$ respectively represent the range of number of batch proposals in the sparse, normal, and dense cases.

Finally, the three types correspond to thresholds ϵ_1, ϵ_2 and ϵ_3 in the *Determinant Point Process-based Proposal Filter* module, which determine how many proposals are filtered respectively in the next module. The larger thresholds ϵ_1, ϵ_2 and ϵ_3 are, the more proposals are selected from the ground set, while the fewer proposals are filtered. When the number of filtering proposals increases, although the efficiency is improved, the accuracy is decreased. This means that when the number of objects is sparse, Real-Batch needs to retain as many proposals as possible, thus improving accuracy. When the number of objects is dense, the proposal number is much larger than proposal number when the number of objects is sparse, therefore Real-Batch needs to delete as many proposals as possible to improve the efficiency.

C. Determinant Point Process-Based Proposal Filter

As shown in Fig. 6, this module contains three submodules: the *Modeling Kernel Matrix*, the *Determinant Point Process*, and the *Greedy Selection Batch Proposals Algorithm*.

1) *The Modeling Kernel Matrix*: The submodule inputs filtered proposals and scores set and outputs the kernel matrix of the ground set. The kernel matrix represents the relationship between each proposal. The construction of the kernel matrix takes into account both the quality of proposals and their diversity to each other. The quality of a proposal is defined as the foreground score of this proposal. Only high-quality proposals work in the subsequent detection process. Diversity measures how different the batch proposal is. We define diversity as the similarity between proposal features in the same batch. The similarity is inversely proportional to diversity in the batch proposal set.

Kernel matrix $\mathbf{L}$ is equivalent to a relationship table used to describe the relationship between any two individuals in the ground set Z , as shown in (4). We use the determinant of kernel matrix to simultaneously measure diversity and quality

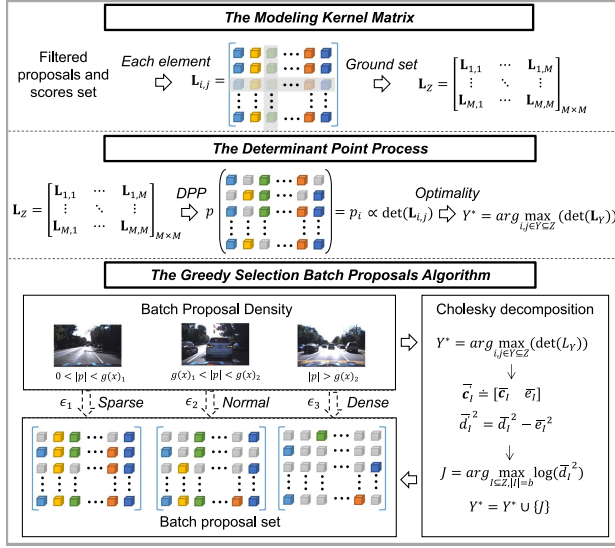


Fig. 6. The framework of the *Determinant Point Process-based Proposal Filter* module. The color definitions for all squares in this module are the same as those in Fig. 4. In the submodule of *The Modeling Kernel Matrix*, the rectangle with a gray background indicates that i row and j column are selected.

relationship between any two individuals in one process. The purpose of constructing the kernel matrix in this way is that we select high-quality and diverse subsets at the same time in the process of one selection.

$$\mathbf{L}_Z = \begin{bmatrix} \mathbf{L}_{1,1} & \mathbf{L}_{1,2} & \dots & \mathbf{L}_{1,M} \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} & \dots & \mathbf{L}_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{M,1} & \mathbf{L}_{M,2} & \dots & \mathbf{L}_{M,M} \end{bmatrix}_{M \times M}. \quad (4)$$

Equation (4) represents kernel matrix $\mathbf{L}_Z$ of the ground set Z , where M represents the number of proposal features in ground set Z . $\mathbf{L}_{i,j}$ represents the relationship between proposal features i and j .

To allow the DPP model to select high-quality and high-diversity proposal features, in our proposed algorithm, the element $\mathbf{L}_{i,j}$ of kernel matrix is constructed in the form of (5).

$$\mathbf{L}_{i,j} = \langle \mathbf{B}_i, \mathbf{B}_j \rangle = \langle s_i \mathbf{f}_i, s_j \mathbf{f}_j \rangle = s_i s_j \langle \mathbf{f}_i, \mathbf{f}_j \rangle = s_i s_j \mathbf{S}_{i,j}, \quad (5)$$

where $\mathbf{S}_{i,j}$ is the similarity between proposal features i and j , $\langle \mathbf{f}_i, \mathbf{f}_j \rangle$ is the inner product of two vectors $\mathbf{f}_i$ and $\mathbf{f}_j$.

Combining (4) and (5), the kernel matrix of ground set Z can be simplified to:

$$\mathbf{L}_Z = \prod_{i \in M} s_i^2 \cdot \mathbf{S}_M, \quad (6)$$

where $\mathbf{S}_M$ represents the difference of all proposals in the ground set Z .

To further improve the efficiency, we perform the log function operation on (6) to convert multiplication operation into addition

operation. It can be expressed as (7):

$$\log \det(\mathbf{L}_Z) = \underbrace{\sum_{i \in M} \log(s_i^2)}_{\text{foreground-score}} + \underbrace{\log \det(\mathbf{S}_M)}_{\text{diversity-score}}, \quad (7)$$

where $\det(\cdot)$ represents matrix determinant. $\sum_{i \in M} \log(s_i^2)$ is expressed as the foreground score of proposal. $\log \det(\mathbf{S}_M)$ is expressed as the diversity among proposal features.

It can be seen from (7) that calculating the kernel matrix can comprehensively evaluate the diversity and quality of proposal features. A set of good diversity among proposal features is selected while ensuring a high foreground score.

2) *The Determinant Point Process*: The goal of the submodule is to select different batch proposals in video frames to avoid too many redundant calculations of the same objects. We take all proposal features in the batch of video frames as the ground set Z , and the features of different proposals as a subset Y . The submodule inputs the kernel matrix $\mathbf{L}_Z$ of the ground set Z and selects a subset of the largest determinant from it. This selection process is modeled as the determinant point process (DPP).

We would like to borrow the existing DPP to solve our problem. Unfortunately, the traditional DPP is not feasible in our problem. In a complex driving environment, the number of objects in a batch frame changes dynamically. Therefore, in a selection process, the number of different objects should also be different. However, the traditional DPP selects a fixed number of objects in a selection process [24], [25]. Real-time Batch dynamically determines the object number based on changes in the environment.

The determinant point process $\mathcal{P}$ is a probabilistic model used to extract a subset Y from the ground set Z . $\mathcal{P}$ describes the probability of occurrence of each subset in a discrete set $Z = \{1, 2, \dots, 2^Z\}$. DPP converts complex probability calculations into simple determinant calculations, and calculates the probability of each subset through the determinant of the kernel matrix. This probability can be understood as the probability of maximizing the quality and diversity of batch proposals in the detection process, which is affected by two factors, foreground score and the diversity of proposal features. The probability that the subset is selected is equal to the ratio of the determinant of subset Y kernel matrix to the determinant of the ground set Z kernel matrix, as shown in (8),

$$\mathcal{P}(Y) = \frac{\det(\mathbf{L}_Y)}{\det(\mathbf{L} + \mathbf{I})}, \quad (8)$$

where $\mathbf{L}_Y$ denotes a submatrix of matrix $\mathbf{L}$ formed by row and column subscripts belonging to Y . $\det(\mathbf{L})$ is the determinant of $\mathbf{L}$. $\mathbf{I}$ is an identity matrix whose rows and columns are equal to $\mathbf{L}$.

When $\mathcal{P}$ gives nonzero probability to the empty set, there exists a positive semi-definite matrix $\mathbf{L} \in \mathbb{R}^{M \times M}$. Since the denominator $\det(\mathbf{L} + \mathbf{I})$ is the same in every selection process, for every subset $Y \subseteq Z$, the probability of Y is shown as (9), which describes the probability of subset Y being selected proportional to the determinant of a subset Y .

$$\mathcal{P}(Y) \propto \det(\mathbf{L}_Y). \quad (9)$$

Algorithm 1: The GSBP Algorithm Based on the Cholesky Decomposition.

Input: Kernel matrix L , stopping selection threshold

 $\epsilon_{i,i=1,2,3}$
Output: Batch proposal set Y^*

1: Take out the diagonal elements of the kernel matrix

$$d_i^2 = \mathbf{L}_{i,i}$$

2: Calculate the average of the cholesky decomposition,

$$\bar{c}_I = (\sum_{i=1}^h \sum_{j=1}^{|c_i|} c_{i,j}) / |c_i|, \bar{d}_I = (\sum_{i=1}^h d_i) / h$$

3: **while** $\bar{d}_I^2 < \epsilon_{i,i=1,2,3}$ **do**

4: Select h proposals by $J = \underset{I, |I|=h}{\operatorname{argmax}} \log(\bar{d}_I^2)$

5: Update batch objects set $Y^* = Y^* \cup \{J\}$

6: **for** $i \in Z \setminus Y^*$ **do**

7: $\bar{e}_I = (\sum_{n=1}^h \mathbf{L}_{n,i} - \langle \bar{c}_I, c_i \rangle) / \bar{d}_I$

8: Update $\bar{c}_I$ based on (20)

9: Update $\bar{d}_I^2$ based on (21)

10: **end for**

11: $J = \underset{I \in Z, |I|=h}{\operatorname{argmax}} \log(\bar{d}_I^2), Y^* = Y^* \cup \{J\}$

12: **end while**

13: **return** Batch proposal set Y^*

The batch proposal set Y is to select the subset that maximizes quality and diversity from the ground set Z in the detection process. Therefore, we can formulate the following optimization problem, as shown in (10). We need to select an optimal subset Y^* with the largest determinant by traversing the ground set Z . There are 2^Z power sets in the ground set Z , and Y^* is the one with the largest determinant in 2^Z .

$$Y^* = \arg \max_{Y \subseteq Z} (\det(\mathbf{L}_Y)). \quad (10)$$

3) *The GSBP Algorithm:* Finding a subset Y^* of the largest determinant from the ground set Z is a complex process due to the vastness of the solution set. Traditional DPP selects one element at a time from the ground set, which depends heavily on the size of the ground set. *Real-Batch speeds up this selection process by expanding the number of selected elements.* Different from the traditional DPP, Real-Batch can greedily select multiple proposals at a time to add to the candidate set, further improving selection efficiency.

In this submodule, we solve the optimal subset Y^* to get the proposal features of high quality and diversity by maximizing $\det(\mathbf{L}_Y)$. We propose the Greedy Selection Batch Proposals algorithm based on the cholesky decomposition to achieve a fast solution. This solution process is simply to greedily select a proposal set that can maximize the marginal benefit from the candidate set each time and add it to the final result subset until the stopping condition ϵ_1, ϵ_2 or ϵ_3 is satisfied.

Combining $\mathcal{P}(Y) \propto \log \det(\mathbf{L}_X)$, at each iteration, we greedily select a proposal subset I of size h that maximizes the marginal benefit from the candidate set and add it to final batch proposal set, as shown in (11).

$$J = \underset{I, |I|=h}{\operatorname{argmax}} \log \det(\mathbf{L}_{X \cup \{I\}}) - \log \det(\mathbf{L}_X). \quad (11)$$

Due to the high complexity of directly solving $\det(\mathbf{L}_X)$, we use cholesky decomposition to subtly simplify the calculation process. The cholesky decomposition of $\mathbf{L}_X$ is defined as $\mathbf{L}_X = \mathbf{V}\mathbf{V}^T$, where $\mathbf{V}$ is an invertible lower triangular matrix. For any subset I , the Cholesky decomposition of $\mathbf{L}_{X \cup \{I\}}$ is shown in (12):

$$\begin{aligned} \mathbf{L}_{X \cup \{I\}} &= \begin{bmatrix} \mathbf{L}_X & \mathbf{L}_{X,I} \\ \mathbf{L}_{I,X} & \mathbf{L}_{I,I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_I & \bar{d}_I \end{bmatrix} \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_I & \bar{d}_I \end{bmatrix}^T \\ &= \begin{bmatrix} \mathbf{V}\mathbf{V}^T & \mathbf{V}\bar{\mathbf{c}}_I^T \\ \bar{\mathbf{c}}_I\mathbf{V}^T & \bar{\mathbf{c}}_I\bar{\mathbf{c}}_I^T - \bar{d}_I^2 \end{bmatrix}, \end{aligned} \quad (12)$$

where $\bar{\mathbf{c}}_I = (\sum_{i=1}^h \sum_{j=1}^{|c_i|} c_{i,j}) / |c_i|$, $\bar{d}_I = (\sum_{i=1}^h d_i) / h$. According to (12) and the principle of matrix multiplication, we can get:

$$\mathbf{V}\bar{\mathbf{c}}_I^T = \mathbf{L}_{X,I}, \quad (13)$$

$$\bar{d}_I^2 = \mathbf{L}_{I,I} - \|\bar{\mathbf{c}}_I\|^2. \quad (14)$$

Further solve the determinant of (12):

$$\begin{aligned} \det(\mathbf{L}_{X \cup \{I\}}) &= \det(\mathbf{V}\mathbf{V}^T (\bar{\mathbf{c}}_I\bar{\mathbf{c}}_I^T - \bar{d}_I^2) - \mathbf{V}\mathbf{V}^T \bar{\mathbf{c}}_I\bar{\mathbf{c}}_I^T) \\ &= \det(\mathbf{V}\mathbf{V}^T \bar{d}_I^2) = \det(\mathbf{L}_X) \bar{d}_I^2. \end{aligned} \quad (15)$$

We simplify (11) by (15):

$$J = \underset{I, |I|=h}{\operatorname{argmax}} \log(\bar{d}_I^2). \quad (16)$$

When the J th proposal feature set is added to the candidate set, we can get $\mathbf{L}_{X \cup \{J\}}$ according to (12):

$$\mathbf{L}_{X \cup \{J\}} = \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_J & \bar{d}_J \end{bmatrix} \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_J & \bar{d}_J \end{bmatrix}^T = \mathbf{V}'\mathbf{V}'^T. \quad (17)$$

For the J th proposal feature set, $\bar{\mathbf{c}}_J$ and $\bar{d}_J$ can also be updated incrementally in the last iteration. Define $\bar{\mathbf{c}}'_J$ and $\bar{d}'_J$ as the new vector and scalar. According to (13) and (17), we have

$$\mathbf{V}'\bar{\mathbf{c}}'^T_I = \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_J & \bar{d}_J \end{bmatrix} \bar{\mathbf{c}}'^T_I = \mathbf{L}_{X \cup \{J\}, I} = \begin{bmatrix} \mathbf{L}_{X,I} \\ \mathbf{L}_{J,I} \end{bmatrix}. \quad (18)$$

Inversely transform (18) to further obtain:

$$\begin{aligned} \bar{\mathbf{c}}'^T_I &= \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \bar{\mathbf{c}}_J & \bar{d}_J \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{L}_{X,I} \\ \mathbf{L}_{J,I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{V}^{-1}\mathbf{L}_{X,I} \\ (\mathbf{L}_{J,I} - \bar{\mathbf{c}}_J\mathbf{V}^{-1}\mathbf{L}_{X,I}) / \bar{d}_J \end{bmatrix}, \end{aligned} \quad (19)$$

where $\bar{\mathbf{c}}'^T_I = \mathbf{V}^{-1}\mathbf{L}_{X,I}$ is calculated by (13). Equation (19) can be further simplified by substituting $\bar{\mathbf{c}}'^T_I$ into:

$$\bar{\mathbf{c}}'_I = \begin{bmatrix} \bar{\mathbf{c}}_I & (\mathbf{L}_{J,I} - \langle \bar{\mathbf{c}}_J, \bar{\mathbf{c}}_I \rangle) / \bar{d}_J \end{bmatrix} \doteq \begin{bmatrix} \bar{\mathbf{c}}_I & \bar{e}_I \end{bmatrix}. \quad (20)$$

TABLE I
DETAILED DESCRIPTION OF THE DATASETS

	UA-DETRAC	D ² -City	MOT20
Object density	33.87	41.94	115.52
Train: test	8200: 5610	5250: 1500	8931: 4479
# of categories	4	12	13
Workload(req/s)	15	25	25

Then, we have

$$\overline{d_I}^2 = \mathbf{L}_{I,I} - \left\| \overline{\mathbf{c}_I'} \right\|_2^2 = \mathbf{L}_{I,I} - \left\| \overline{\mathbf{c}_I} \right\|_2^2 - \overline{e_I}^2 = \overline{d_I}^2 - \overline{e_I}^2. \quad (21)$$

Therefore, we solve for the maximum probability $\det(\mathbf{L}_X)$ by updating $\overline{\mathbf{c}_I'}$ and $\overline{d_I}^2$. In each iteration, a proposal feature set that maximizes the marginal benefit is greedily selected from the candidate set and added to the final batch proposal set.

V. EXPERIMENTAL SETUP

A. Dataset and Evaluation Setup

We deploy the Real-Batch in a real-world development environment and empirically verify the merit by conducting experiments on the UA-DETRAC [11], D²-City [26] and MOT20 [27] datasets. UA-DETRAC and D²-City datasets are large-scale driving video datasets, particularly featuring extremely complex and diverse traffic scenarios. MOT20 dataset is a large benchmark dataset for multi-object tracking and detection, focusing more on densely populated scenes. The details about the dataset are shown in Table I. In addition, we set different video frame workloads to simulate the perception environment of autonomous driving.

Training Details: We train and inference the whole architecture on an Nvidia RTX 2060 GPU with SGD (learning rate=0.005, momentum=0.9, weight decay=0.0005). We adjust the learning rate by designing a route that multiplies the learning rate by 0.33 every 3 steps. During the training process, we first froze the pre-trained weights of the backbone, and then use the pre-trained weights to fine-tune RPN and for the final prediction network part. Next, we unfreeze the backbone weights, and then train the entire network. Training the network is not the primary focus of our research. Incorporating various optimizers into Real-Batch is a seamless process that does not compromise its core functionality. As a result, we conduct experiments using the Adam optimizer, known for its robustness in hyperparameter selection, and employ the default parameters provided by the PyTorch framework. We find that the Adam optimizer converges faster than SGD and demands less memory. While there are differences in training performance, the selection of optimizer does not affect the inference capabilities of Real-Batch.

Testbed: The platform is based on a high-performance workstation equipped with an Nvidia RTX 2060 GPU, which has a total memory of 16 GB. The GPU performance (FP16: 12.9 TFLOPS) is comparable to Nvidia Jetson AGX Xavier (FP16: 11 TFLOPS), which is often used in autonomous driving cars [28], [29], [30]. The workstation has a 22.3 GHz Intel processor with 32 CPU cores and 64 GB of CPU memory. On the software

side, the platform is running Ubuntu 20.04, which provides a stable and efficient environment for our experiments. We utilize Python 3.8 as the primary programming language and PyTorch to support inference models on GPUs.

B. Evaluation Metrics and Baselines

We use four metrics to evaluate Real-Batch and baselines: *Accuracy* (mean Average Precision), *FPS* (Frames Per Second), *MIV* (Mean Information Value), and *MDR* (Missing Deadline Ratio).

Accuracy represents the mean Average Precision (mAP) contained in all video frames. *FPS* is how many frames per second the system can process (detect). *MIV* represents the mean of information value for all decisions during detection. Information value contains two metrics: information completeness and information age. Information completeness evaluates how many video frames are lost during a decision. Information age evaluates whether the inference video frame is freshly captured during a decision. *MDR* represents missing deadline ratio during all decisions. This metric evaluates whether the system can make real-time decisions based on inference results.

In order to compare the superior performance of Real-Batch, we used four SOTA methods that are closely related to the autonomous driving object detection system. AdaPyramid [31] balances accuracy and inference latency by adjusting specific combinations of knobs in the video analysis system (such as resolution, sampling rate, etc.) and eliminating redundant background in video frames. Batch-MOT [32] combines multiple input images into a batch for DNN inference during batch execution, reducing execution time at the expense of tracking accuracy. RT-MOT [33] designs a scheduling algorithm that combines offline and online methods, selecting different detection models and target matching models based on changes in target confidence, balancing accuracy and latency. BAS [34] is a Batching-Aware Scheduling algorithm that assigns high priority to key targets and increases detection frequency. Additionally, this algorithm submits multiple regions of the same size to the GPU for processing simultaneously, enhancing computational efficiency.

VI. PERFORMANCE EVALUATION

A. Main Result

Performance comparison of Real-Batch on different datasets: As shown in Table II, the Real-Batch outperforms other SOTA across UA-DETRAC, D²-City and MOT20 datasets on four metrics. Specifically, the Real-Batch achieves a 4.2%–13.2% higher average accuracy than the SOTA methods. In terms of efficiency, the Real-Batch processes frames 4.3-7.9 FPS faster than SOTA methods. This is due to its optimized batch processing and elimination of similar objects, enabling higher throughput while maintaining detection deadlines. In addition, the Real-Batch ensures minimal information loss of frames during decision-making, and delivers 3.2%-15.1% higher Mean Information Value (MIV) compared to others. At the same time, the Real-Batch reduces the deadline miss rate by 3%-10.6%,

TABLE II
PERFORMANCE COMPARISON OF REAL-BATCH ON DIFFERENT DATASETS

Metrics Dataset	Accuracy (% , $\uparrow$ better)			FPS (frames/s, $\uparrow$ better)			MIV (% , $\uparrow$ better)			MDR (% , $\downarrow$ better)		
	UA- DETRAC	D ² - city	MOT- 20	UA- DETRAC	D ² - city	MOT- 20	UA- DETRAC	D ² - city	MOT- 20	UA- DETRAC	D ² - city	MOT- 20
SOTA												
AdaPyramid	76.9	62.4	60.8	17.5	17.2	16.8	80.8	64.5	60.7	10.6	23.9	26.1
Batch-MOT	77.3	72.2	68.4	16.7	21.7	23.8	81.6	77.2	74.7	9.7	14.3	16.5
RT-MOT	76.4	62.9	51.6	16.9	17.5	17.6	79.8	60.2	57.9	9.8	25.2	28.4
BAS	75.6	71.8	67.2	15.4	19.4	22.4	80.3	76.8	72.6	11.2	15.8	16.8
Real-Batch	78.5	76.2	75.7	18.3	24.3	32.6	84.2	80.7	78.3	8.4	10.5	12.6

ensuring that all frames are processed within the required deadline by dynamically adjusting the batch size.

On the low-workload UA-DETRAC dataset, both Real-Batch and SOTA process video frames in real-time, indicating that both exhibit satisfactory scheduling and processing capabilities in lightweight task scenarios. Consequently, they perform well across all four metrics. This validates that, under relaxed task conditions, performance bottlenecks are less pronounced for these algorithms.

However, as the workload increases, particularly on high-density datasets such as MOT20, Real-Batch and SOTA decline in Accuracy, MIV, and MDR. This occurs because higher workloads result in fewer video frames being processed, leading to decreased detection accuracy (Accuracy) and information value (MIV), as well as increased missed deadline ratios (MDR). Despite the overall decline in metrics, Real-Batch demonstrates significantly smaller losses in precision compared to SOTA algorithms. On MOT20, Real-Batch's average precision loss is only 2.8%, while AdaPyramid, Batch-MOT, and RT-MOT suffer losses of 16.1%, 8.9%, and 24.8%, respectively. This highlights Real-Batch's robustness in handling complex, high-density tasks.

As the workload increases, batch-based scheduling algorithms experience an improvement in FPS due to mechanisms designed to process video frames more quickly under high input rates. Compared to the UA-DETRAC dataset, batch-based scheduling algorithms, Batch-MOT, BAS, and Real-Batch, achieved improvements of 7.1, 7, and 14.3 FPS, respectively, on the MOT20 dataset. However, non-batch based scheduling algorithms, such as AdaPyramid and RT-MOT, exhibited little to no improvement. In scenarios with higher workloads, Real-Batch achieved an average improvement of 5.2% compared to other batch-based scheduling algorithms, and an average improvement of 7.8% compared to non-batch scheduling algorithms. This fully demonstrates the effectiveness of Real-Batch in processing video frames through dynamic adjustment of batch sizes.

The comparison of visualization results: We conduct an additional experiment to visualize the object detection results for Real-Batch alongside other SOTA methods. We specifically chose an IoU threshold of 0.7 to highlight the ability of Real-Batch in filtering out similar candidate objects, showcasing its effectiveness in reducing redundancy. For a clearer comparison, we select three typical frames from a batch of video frames for visual comparison. To ensure a fair comparison, we select the largest detected object in each video frame as the ground truth (GT) for comparison. The rationale behind this choice is that

larger targets are typically easier to detect, providing a clearer demonstration of detection performance.

From the visualizations in Fig. 7, we observe that Real-Batch consistently detected objects that are closest to the ground truth. This is largely due to the batch processing and the ability to filter out similar candidate targets, thereby reducing redundancy in the detection process.

AdaPyramid and RT-MOT, which do not employ batch processing, are observed to have undetected video frames, indicating their limitations in handling the redundancy in candidate targets. While BAS and Batch-MOT use batch processing to accelerate detection, we find that these methods still result in a significant amount of redundant detection, as they do not filter out similar targets as effectively as Real-Batch.

Comparative analysis with respect to real-time analysis: We compare the real-time performance of Real-Batch with SOTA methods across different datasets. The results are presented in a new figure that shows the Cumulative Distribution Function (CDF) of latency for both Real-Batch and SOTA.

Fig. 8(a) demonstrates that on the UA-DETRAC dataset, which has a lighter workload, the latency CDFs of Real-Batch and SOTA are closely aligned. Despite this similarity, Real-Batch achieves a reduction in inference latency by $1.12 \times$. This indicates that even under lighter workloads, Real-Batch is effective in optimizing latency.

As the workload increases, the divergence between the latency CDFs of Real-Batch and SOTA becomes more pronounced. This trend is evident in the results from the D²-City and MOT20 datasets, where Real-Batch reduces the inference latency by $1.57 \times$ and $1.5 \times$, respectively, as shown in Fig. 8(b) and (c). The enhanced performance of Real-Batch under heavier workloads can be attributed to its ability to efficiently eliminate redundant detection of similar targets within video frames. This feature becomes increasingly beneficial as the number of similar targets grows with the workload.

B. Comparative Analysis of Various Performance Parameters

First, we analyze Real-Batch's performance on the UA-DETRAC dataset under five different workloads (requests/s): 10, 15, 20, 25, and 30. As shown in Fig. 9, we observe that at lower workloads, Real-Batch exhibits relatively stable performance in terms of Accuracy, FPS, MIV, and MDR. However, as the workload increases, accuracy decreases slightly (by 0.5% to 1.26%), but this is offset by significant improvements in FPS (up to 58.3%). This speed improvement is due to Real-Batch's dynamic adjustment of batch sizes, which helps accelerate target

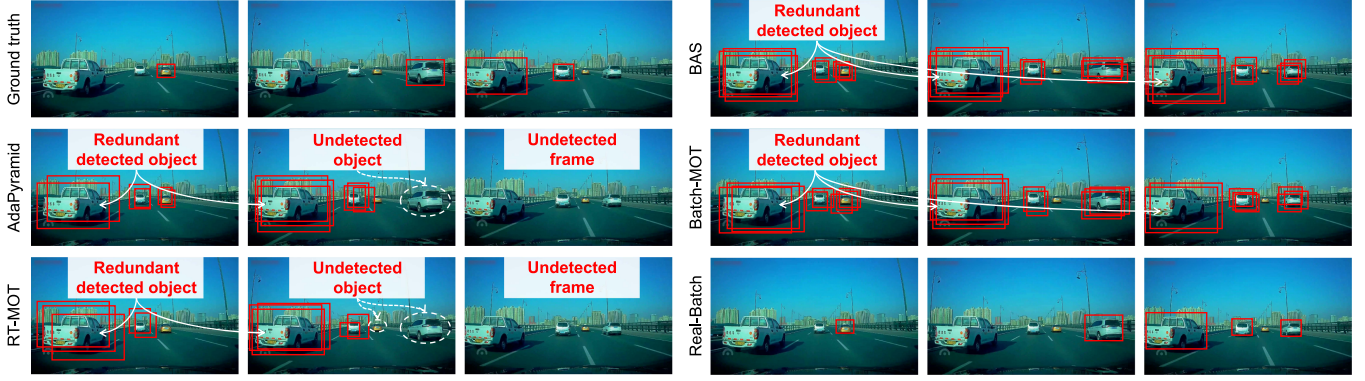


Fig. 7. Visualize results for object detection by comparing Real-Batch and various SOTA methods.

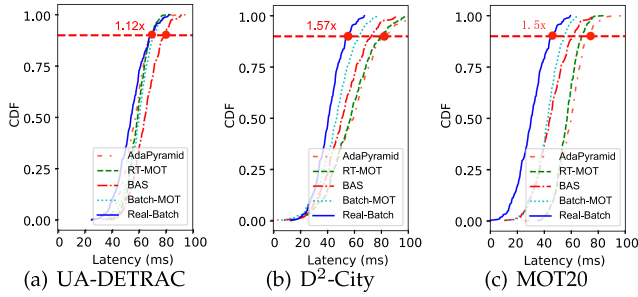


Fig. 8. The latency CDF of Real-Batch and SOTA methods under different datasets.

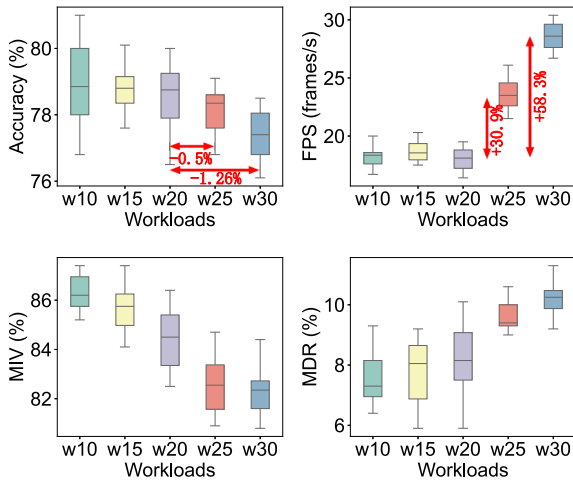


Fig. 9. Performance analysis of the Real-Batch on the UA-DETRAC dataset when workload changes.

detection with minimal accuracy loss. Nonetheless, when the workload becomes too high, MIV increases and MDR decreases, as more video frames cannot be processed in the available time.

In addition, we evaluate Real-Batch's performance under varying deadlines on the UA-DETRAC dataset, setting the deadlines at 100 ms, 300 ms, and 500 ms. As depicted in Fig. 10, the deadline appears to have a more significant impact on MIV and MDR than on Accuracy and FPS. A shorter deadline results

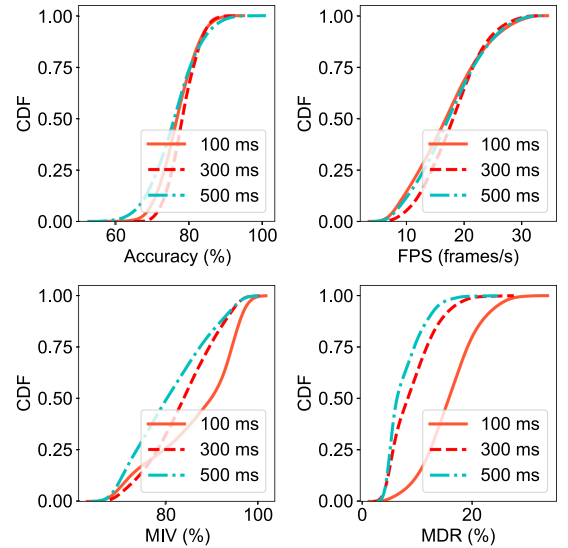


Fig. 10. Performance analysis of the Real-Batch on the UA-DETRAC dataset when deadline changes.

in newer information being processed, leading to an increase in MIV, as the model is able to deliver more accurate detection with fresher data. However, when the deadline becomes shorter, the system may not have sufficient time to process all frames, which causes lower MDR.

C. Confusion Matrix and Precision-Recall Curves of Real-Batch on Different Datasets

We analyze the Precision-Recall (PR) curves and confusion matrices for the Real-Batch algorithm across different datasets. We visualize the confusion matrix, where the axes represent the predicted and true labels, respectively. The label "0" denotes the background, while the other labels follow the order specified in the public datasets. As illustrated in Fig. 11(a), Real-Batch demonstrates superior accuracy on the UA-DETRAC dataset. This can be attributed to the UA-DETRAC dataset's simpler scenes, fewer labels, and a lower number of targets compared to the D²-City and MOT20 datasets. Furthermore, Fig. 11(b),

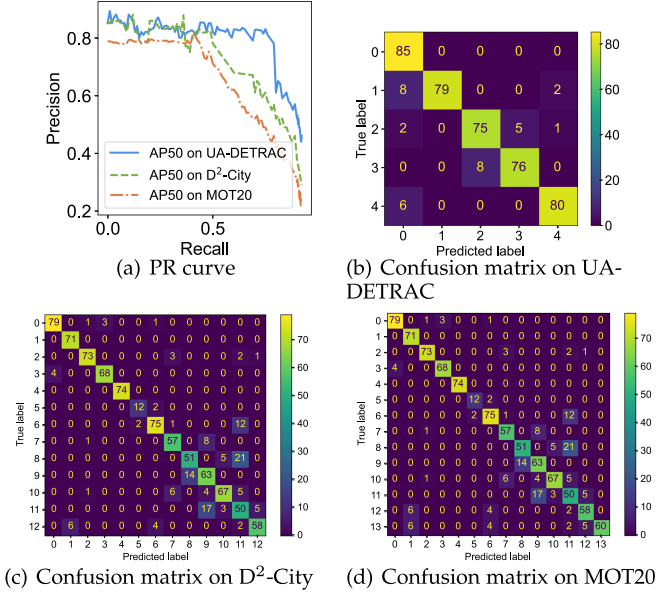


Fig. 11. PR curve and confusion matrix of the Real-Batch.

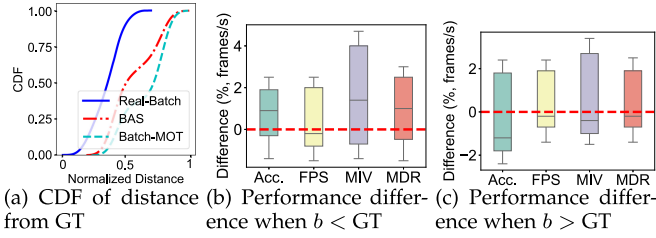


Fig. 12. Comparative analysis of predicted batch size by Real-Batch and GT.

(c) and (d) highlight that Real-Batch performs well in detecting larger objects, such as trucks. However, its accuracy decreases when detecting smaller objects, like pedestrians. These findings underscore the algorithm's effectiveness in various scenarios and its potential areas for improvement.

D. Optimality of Dynamic Batch Size and Risk From Wrong Size

We conduct analytics experiments to validate the effectiveness and robustness of the dynamic batch size selection in Real-Batch, comparing it with ground truth and other state-of-the-art methods.

To verify the optimality of dynamic batch sizes, we calculate the normalized euclidean distance between the batch sizes selected by Real-Batch, BAS, and Batch-MOT, and the corresponding ground truth (GT). As shown in Fig. 12(a), the batch sizes selected by Real-Batch exhibit the smallest distance to the GT compared to the other methods, demonstrating that its dynamic batch size selection is closer to optimal. This indicates that Real-Batch consistently makes better batch size decisions than other state-of-the-art methods, validating its adaptability and precision in real-world scenarios.

We further analyze the impact of incorrect batch sizes by comparing performance metrics when the dynamically selected batch size by Real-Batch deviates from the ground truth. In Fig. 12(b) and (c), we plot the differences in performance metrics between the batch sizes chosen by Real-Batch and the GT. The results reveal only minor fluctuations (approximately -5% to +5%) across all metrics when the batch size deviates slightly from the GT. These fluctuations indicate that even when batch sizes are not perfectly aligned with the GT, the system maintains stable performance without significant degradation.

The experiments demonstrate that Real-Batch achieves near-optimal performance while being robust to small deviations in batch size selection. The dynamic adjustment mechanism enables Real-Batch to adapt to varying conditions in real-time without introducing significant risks or performance loss. These findings highlight the strength of Real-Batch in balancing accuracy and efficiency under diverse operational scenarios.

E. Scalability of Real-Batch

We implement and evaluate the scalability of Real-Batch with five state-of-the-art object detection algorithms and different camera numbers to show how Real-Batch performs with different models and the generosity of our framework.

Faster RCNN [35] is the most classical two-stage object detector in the field of object detection. Compared with the one-stage object detector, this algorithm has better accuracy and worse efficiency. *DFF* [5] and *ATEN* [36] use a similar feature map of two adjacent frames to reduce redundant calculations based on optical flow. Real-Batch utilizes similar proposals to reduce redundant calculations and has a finer granularity. *RDN* [37] is a video object detection algorithm based on an attention mechanism and *D&T* [38] is a tracking-based video object detection algorithm.

We use two scheduling approaches to compare with Real-Batch. These two scheduling approaches are the most common in autonomous driving scenarios. *Skipping schedule approach* [15], [16], [17] always detects frames the camera has just captured and discards existing frames. Although the deadline can be met in time, this approach loses a lot of information. Among these existing works, we use the dynamic frame detection [17], as a representative of the skipping schedule approach in our evaluation. *Keyframe schedule approach* [18], [19] selects keyframes that contain important information to detect. Compared to the skipping schedule approach, this approach reduces information loss, but it depends heavily on the choice of keyframes. Among these existing works, we use the EC² Detect approach [19], as the keyframe schedule approach in our evaluation.

The main experimental results are shown in Table III and drawn by testing all video frames in the UA-DETRAC test dataset. In Table III, we assumed that an autonomous driving car has only one forward-looking camera and five cameras (one forward-looking camera and four circumferential cameras), respectively. One camera is a must for an autonomous driving car to sense its environment, while five cameras are a basic configuration for the L2 stage [39].

TABLE III
PERFORMANCE COMPARISON OF REAL-BATCH AND OTHER SCHEDULE APPROACHES WITH DIFFERENT TRADITIONAL NETWORKS

		Faster RCNN		DFF		ATEN		RDN		D&T	
Camera's number		1	5	1	5	1	5	1	5	1	5
Acc.	Skipping schedule [17]	55.7	53.4	68.4	67.7	65.7	64.2	76.6	75.4	75.1	73.8
	Keyframe schedule [19]	56.2	55.4	69.8	68.2	66.3	65.4	77.5	76.2	76.2	75.4
	Real-Batch	57.4	56.8	70.4	69.5	67.5	66.9	78.9	77.4	77.2	76.5
FPS	Skipping schedule [17]	10.9		13.5		8.4		7.2		5.8	
	Keyframe schedule [19]										
	Real-Batch	18.3	27.6	20.7	29.1	13.5	17.8	12.1	17.5	9.8	15.2
MIV	Skipping schedule [17]	0.64	0.51	0.68	0.52	0.54	0.42	0.48	0.33	0.38	0.31
	Keyframe schedule [19]	0.68	0.54	0.74	0.58	0.59	0.48	0.52	0.41	0.43	0.36
	Real-Batch	0.72	0.59	0.79	0.61	0.64	0.52	0.58	0.51	0.47	0.40
MDR	Skipping schedule [17]	0.13	0.10	0.08	0.03	0.14	0.10	0.18	0.13	0.29	0.18
	Keyframe schedule [19]	0.14	0.11	0.09	0.05	0.19	0.15	0.26	0.21	0.43	0.27
	Real-Batch	0.12	0.08	0.07	0.03	0.16	0.12	0.21	0.16	0.32	0.24

From Table III, we can see that Real-Batch achieves better *Accuracy*, *FPS*, *MIV*, and *MDR* than other scheduling approaches. We find that when the number of cameras increases (that is, more video frames arrive at the same time), Real-Batch loses more *Accuracy* and *MIV*, but gains better *FPS* and *MDR*. This means that Real-Batch will be more real-time as more frames are captured.

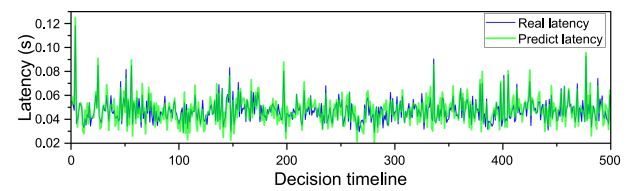
For *Accuracy*, Real-Batch gets the best metrics compared to the other two approaches, as shown in Table III. This is mainly because Real-Batch processes more video frames by batching and eliminating similar targets. In addition, the performance of Real-Batch is related to the performance of the traditional network. RDN has the highest performance among the five traditional networks, so Real-Batch with RDN is also the best.

For *FPS*, Real-Batch with five traditional networks achieves the highest efficiency compared to the other three scheduling approaches. This is because as the number of cameras increases, Real-Batch adapts to change batch size (Section IV-A) to process more video frames.

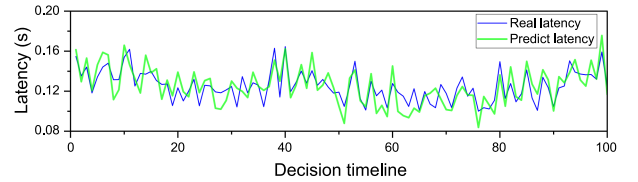
For *MIV*, when the number of cameras increased, the performance of all three scheduling approaches decreased. This is simply because when the number of captured video frames increases, the detection capability of the three scheduling approaches with traditional networks is limited, resulting in more newly arrived video frames not being detected. Skipping schedule approach detects video frames closest to the deadline, but the information integrity of this approach is less than that of Real-Batch.

For *MDR*, Real-Batch with Faster RCNN and DFF achieves the best performance. Skipping schedule approach with other traditional networks achieves the highest performance. When the decision deadline arrives, skipping schedule approach skips undetected video frames to avoid missing the deadline. Further, by combining the *FPS* comparison, we found that faster traditional networks get smaller *MDR*. This is mainly because faster traditional networks are able to process video frames more promptly before the deadline, thus avoiding missing the deadline.

Generosity of Real-Batch: We can also see from the results that Real-Batch is general enough to cooperate with different two-stage object detection algorithms. More importantly, our



(a) The number of cameras is 1



(b) The number of cameras is 5

Fig. 13. Comparison of real latency and predict latency.

method is designed to seamlessly integrate with these existing dynamic frame selection algorithms. They can be employed as modules to determine the adaptive batch size and frame selection criteria, serving as a foundation for Real-Batch to further enhance detection efficiency through proposal filtering. By combining these complementary techniques, Real-Batch can achieve a holistic optimization of both batch composition and proposal selection.

F. Experimental Analysis of Multiple Cameras

In this section, we analyze Real-Batch with Faster RCNN under two conditions (one camera and five cameras). The test set is all frames from a whole video in the UA-DETRAC dataset.

Prediction latency and accuracy of proposals: We analyze the prediction accuracy of the *Deadline-driven Adaptive Batch Scheduling* module, including latency prediction and proposal number prediction, as shown in Figs. 13 and 14 respectively.

In Fig. 13, the abscissa is the decision timeline and the ordinate is latency. As shown in Fig. 13(a), when the number of cameras is 1, the curves of real latency and predicted latency almost coincide. This means that latency prediction is relatively accurate when the number of video frames is small in a decision

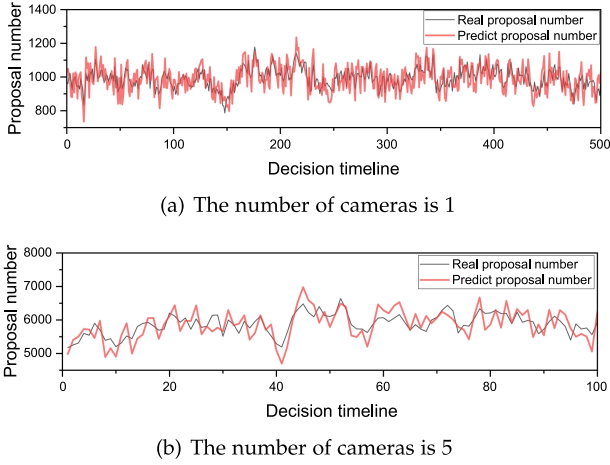


Fig. 14. Experimental comparison of the number of real proposal and predict proposal.

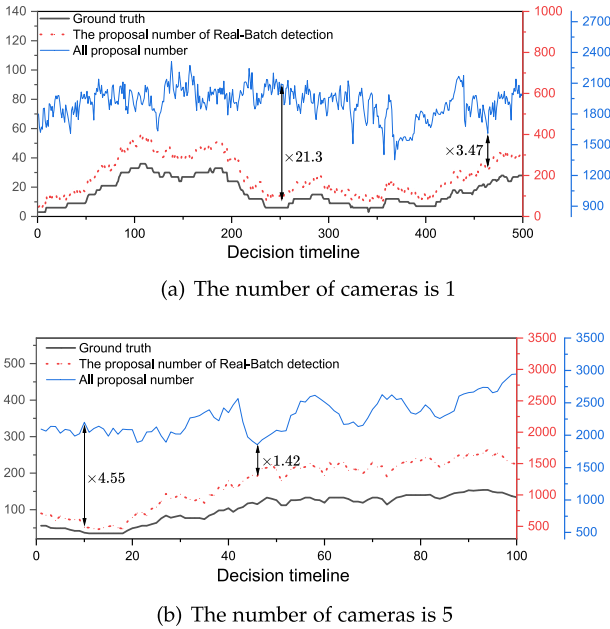


Fig. 15. Experimental comparison of proposal number.

process. As shown in Fig. 13(b), when the number of cameras is 5, there is a certain deviation between the curves of real latency and predicted latency. This is mainly because when there is a large number of video frames in a decision process, the batch selected by Real-Batch will change dynamically to avoid missing the deadline. In Fig. 14, the abscissa is the decision timeline and the ordinate is the number of proposals. Figs. 14 and 13 have similar predictions, and this further verifies the correlation between proposal number and inference time.

Elimination of proposal number: We analyze the validity of the *Determinant Point Process-based Proposal Filter* module, as shown in Fig. 15, where ground truth represents the number of different object IDs in the decision process.

In Fig. 15(a), the abscissa is the decision timeline, and the ordinate corresponds from left to right to ground truth, the

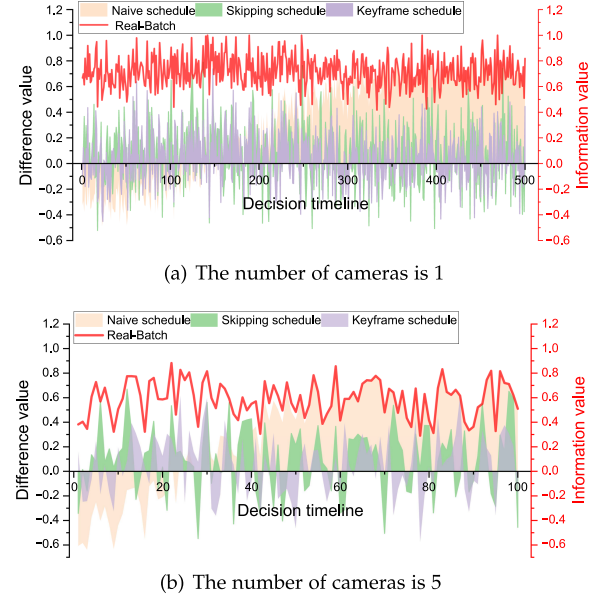


Fig. 16. Experimental comparison of information value.

proposal number of Real-Batch and all proposal numbers. As shown in Fig. 15(a), the proposal number of Real-Batch and ground truth shows a similar trend. This verifies that Real-Batch predicts the proposal number of the next decision and adjusts the batch size in time. More importantly, Real-Batch can filter the same objects in batch frames, and greatly reduce the number of proposals to be detected, thus increasing the detection efficiency. Compared with the original proposal number, the proposal number of Real-Batch reduces by 3.47x at the minimum and 21.3x at the maximum. This means that the *Determinant Point Process-based Proposal Filter* module improves the detection efficiency by filtering proposals.

Compared with Fig. 15(a), when the number of cameras is increased to 5, the efficiency improvement is small. In Fig. 15(b), compared with original proposal number, the proposal number of Real-Batch reduces by a minimum of 1.42x and a maximum of 4.55x. This is mainly because the larger batch has led to an increase in the proposal number of Real-Batch. Although there is a small efficiency gain, Fig. 15(b) has a larger batch than Fig. 15(a). Therefore, as the number of cameras increases, so does FPS.

The information value of Real-Batch: As shown in Fig. 16, we analyze Real-Batch schedule approach with Faster RCNN in every decision process. In Fig. 16, the abscissa is the decision timeline, and the ordinate on the left corresponds to the difference value between naive, skipping, keyframe schedule approach and Real-Batch schedule approach under information value. The right ordinate corresponds to the information value of Real-Batch schedule approach. As shown in Fig. 16(a), naive, skipping and keyframe schedule approaches have large areas above the abscissa. This proves that Real-Batch schedule approach has greater information value compared to the other three approaches. This situation can be summarized more easily in Fig. 16(b).

G. Comparison of Time and Space Computational Complexity

Time complexity analysis: In the Deadline-driven Adaptive Batch Scheduling module, we use polynomial regression to predict the number of proposals in the next batch and dynamically adjust the batch size. In detail, predicting the number of proposals uses polynomial regression, with a complexity of $O(kn)$, where k is the degree of the polynomial regression, and n is the amount of historical proposal data. For the optimization problem of dynamically adjusting the batch size, selecting K batch sizes incurs a constant complexity of $O(K)$. Therefore, the total time complexity of this module is $O(n)$.

In the Batch Proposal Density-based Discriminator module, we analyze the density of proposals in a batch using a normal distribution approximation and classify them into sparse, normal, or dense categories. In detail, this module simple linear relationships to construct the process, with a complexity of $O(m)$. Therefore, the total time complexity of this module is $O(m)$.

In the Determinant Point Process-based Proposal Filter module, we construct a similarity matrix for proposals and select a subset of proposals using DPP, accelerated by Cholesky decomposition. In detail, first, during the matrix construction phase, we compute the similarity matrix L for the proposals, with a size of $m \times m$, resulting in a complexity of $O(m^2)$. When using the DPP to select proposals, a greedy algorithm is applied to select subsets, with a theoretical complexity of $O(m^3)$, as multiple determinant calculations are required for $m \times m$ matrix. However, with Cholesky decomposition introduced, the complexity is reduced to $O(m^2)$ for each iteration. The total complexity becomes $O(hm^2)$, where h is the number of iterations of the greedy selection, which is generally much smaller than m . Therefore, the time complexity of this module is $O(m^2)$.

The overall time complexity of Real-Batch is $O(m^2)$, where m is the number of proposals in a batch.

Space complexity analysis: In the Deadline-driven Adaptive Batch Scheduling module, we require storing the number of proposals n over a period of time for polynomial regression, with a complexity of $O(n)$. Additionally, we also store the batch size used for the next decision, which incurs a space complexity of $O(1)$. Therefore, the space complexity for this module is $O(n)$.

In the Batch Proposal Density-based Discriminator module, for each proposal, we need to store its coordinate information and score, which takes up $O(m)$ space, where m is the number of proposals per frame in current times. In addition, storing the parameters of normal distribution and density classification results (where proposals are classified as sparse, normal, or dense scenes) only requires constant space. Therefore, the space complexity for this module is $O(m)$.

In the Determinant Point Process-based Proposal Filter module, we construct and store a similarity matrix for proposals, with size $m \times m$, resulting in a space complexity of $O(m^2)$. Each proposal's feature vector requires storage. For instance, if each vector has d dimensions (e.g., 1024), the feature storage cost is $O(md)$. The score of proposals also needs storage, contributing $O(m)$. During the proposal filtering process, the selected proposal set Y^* (size h , generally $h \ll m$) and updates

to the matrix during Cholesky decomposition add negligible overhead, contributing $O(mh)$. Therefore, the space complexity for this module is $O(m^2 + md + mh)$.

By combining the space complexity of all modules, the overall space complexity of the Real-Batch is $O(n + m + m^2 + md + mh)$. Simplifying under the assumption d and h are fixed, the overall space complexity approximates to $O(m^2 + md)$.

Based on all the previous analyses, the time and space complexity of the Real-Batch are $O(m^2)$ and $O(m^2 + md)$, respectively, where m is the number of proposals in a batch and d is the feature dimension of each proposal.

VII. DISCUSSION

In this section, we discuss important open issues of Real-Batch.

Incomplete adaptability of one-stage object detection algorithm: One-stage object detection algorithms, such as YOLO and SSD, do not generate proposals, making them incompatible with the full Real-Batch framework. To address this, we design a limited version of Real-Batch that incorporates only the Deadline-driven Adaptive Batch Scheduling module. While this adaptation demonstrates some improvement in real-time performance, it cannot leverage the full potential of proposal-based filtering. However, it is worth noting that one-stage object detection algorithms are inherently designed for scenarios where speed is prioritized over accuracy. Compared with the one-stage detection algorithm, the two-stage detection algorithm with higher accuracy is more suitable for safety-critical scenarios. Thus, the absence of full compatibility with one-stage algorithms is less impactful in these critical scenarios.

The industrial significance of the Real-Batch: The Real-Batch demonstrates significant potential in the domain of autonomous driving, particularly in real-time object detection for Advanced Driver Assistance Systems (ADAS). For example, it can be integrated into collision avoidance systems to enhance real-time responsiveness and reduce redundant detection, ensuring the vehicle can react swiftly in high-density traffic environments. Studies [40], [41], [42], [43] have highlighted the critical need for efficient object detection modules in these systems, and our method addresses these requirements by improving computational efficiency without compromising accuracy.

Additionally, Real-Batch can be adapted for use in video-based traffic monitoring systems deployed in smart cities. By processing video frames in batches and filtering redundant objects, it can streamline the detection pipeline, enabling the system to analyze high-traffic zones more effectively. This integration aligns with the goals of improving urban traffic flow and reducing congestion, as discussed in [44], [45], [46]. Our method could, for instance, enhance the module responsible for detecting and predicting vehicle movements, resulting in better traffic management outcomes.

The risk factors of Real-Batch: One of the primary risk factors is the possibility of filtering out important candidate bounding boxes during the processing of a single video frame. While this adaptive approach enhances efficiency, it may occasionally result in missing key targets within that specific frame. However,

due to the nature of video systems, the same target is often visible in adjacent frames. As a result, our system has the capability to detect and recover these missed targets from subsequent frames, thereby reducing the overall impact of this issue on detection performance. This temporal redundancy inherent in video data helps to mitigate the risks associated with filtering errors.

Hyperparameters setup of Real-Batch: The hyperparameters $\epsilon_1, \epsilon_2, \epsilon_3$ determine the number of selected proposals in *Determinant Point Process-based Proposal Filter* module. The larger the threshold is, the more proposals are filtered out for efficiency. When the number of filtering proposals increases, although the efficiency is improved, the accuracy is decreased. We adjust the thresholds by leveraging under different scenarios (i.e., sparse, normal, and dense real-world scenarios), which makes the thresholds more robust and can be adapted to different two-stage object detection algorithms. In our experiments with Real-Batch, we obtained the best results using $\epsilon_1 = 0.08$, $\epsilon_2 = 0.05$, and $\epsilon_3 = 0.01$, after systematically tuning ϵ_1, ϵ_2 , and ϵ_3 within the range $[0.01, 0.1]$ with a step size of 0.01.

VIII. RELATED WORK

Object detection algorithms and optimization: Existing object detection algorithms rely on deep convolutional neural networks. According to their network structure, they are divided into two categories, one-stage and two-stage. The former is faster because it directly calculates the image to obtain the object. Representative work includes SSD [47], YOLO [48] and RetainNet [49], etc. The latter is pioneered by RCNN [50], which is a region-based network and has a higher accuracy, such as Fast RCNN [51], Faster RCNN [35]. In vision-based autonomous driving, perception data is typically captured in the form of video frames, so video object detection [7], [37], [52] is also commonly used in autonomous driving environments. This method improves the accuracy and efficiency of object detection by fusing features from multiple video frames.

The real-time performance of one-stage object detection algorithms and the high accuracy of two-stage object detection algorithms are both essential in autonomous driving environments. Therefore, in recent years, a significant amount of work has focused on improving detection accuracy while meeting real-time requirements [53], [54], [55], [56]. WAVE [53] shortens inference time through Region of Interest encoding to adapt to resource-constrained edge devices. Ma et al. [54] enhance the generalization capability of object detection algorithms for unseen objects under various scenarios. RT-DETR [55] and YOLOH [56] modify model architectures and loss functions to ensure both efficiency and accuracy in object detection.

The aforementioned works require modifications to loss functions and retrain model architecture when balancing efficiency and accuracy, reducing their plug-and-play adaptability across different systems. In contrast, the Real-Batch seamlessly integrates with existing two-stage detection frameworks as middleware, providing a flexible and scalable solution. Additionally, Real-Batch optimizes the demands for both efficiency and accuracy by adaptively selecting batches and eliminating similar objects within a batch of video frames.

Object detection systems scheduling optimization: To improve the efficiency of object detection on resource-constrained autonomous vehicles, some studies offload video frames to edge servers for processing [46], [57], [58], [59]. AdaptSLAM [57] and STAC [58] select a subset of key frames to adapt to video content in mobile devices and edge servers under resource constraints. Edgeduet [46] enhances the edge device collaboration framework for small object detection through tile-level parallelism to achieve high accuracy and low latency. However, these methods rely on a stable and low-latency connection between autonomous vehicles and edge servers, which may not always be feasible in highly dynamic autonomous driving environments.

A hard deadline is established for object detection in autonomous driving. To avoid missing this deadline, existing work accelerates the efficiency of object detection by optimizing scheduling algorithms. Batch-MOT [32] supports two types of pipeline execution, providing strict timing guarantees for batch DNN execution across multiple tasks. RT-MOT [33] considers various combinations of various detection and tracking models to reduce execution time. Skipping [15], [16], [17] and keyframe scheduling [18], [19], [34] optimize frame selection strategies by discarding certain video frames, thereby avoiding missed deadlines and ensuring efficient object detection. Although these methods avoid missing deadlines to some extent, they do not account for significant fluctuations in object density and workload, making it difficult to adapt to the variability of autonomous driving.

Unlike the aforementioned works, Real-Batch operates independently of edge infrastructure and runs entirely on the vehicle, eliminating reliance on external infrastructure. This makes it more robust in scenarios where network connections are unstable or unavailable. Additionally, Real-Batch allows to dynamically adjust batch size based on real-time workload changes. This adaptability ensures efficient detection and timely responses, even in highly variable autonomous driving environments.

IX. CONCLUSION

This work proposes a novel and fast video frame batch detection method by reducing redundant computation for the same object. Real-Batch aims to effectively package a batch of video frames by intelligently scheduling frames and filtering redundant objects based on the contextual relationships between video frames. In order to overcome the complex environment of autonomous driving, we differentiate video frames and perform object detection by cases. We model the relationship between proposals, and propose a fast solution algorithm to extract the optimal subset of proposals. We evaluate Real-Batch based on real-world datasets. Experimental results show that Real-Batch achieves better accuracy and real-time performance.

ACKNOWLEDGMENT

The authors thank all the reviewers for their insightful feedback to improve this paper.

REFERENCES

- [1] M. Colley, J. Britten, and E. Rukzio, "Scalability in external communication of automated vehicles: Evaluation and recommendations," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 7, no. 2, pp. 1–26, 2023.
- [2] Tesla, "Autopilot and full self-driving capability," 2023. [Online]. Available: <https://www.tesla.com/support/autopilot>
- [3] C. Urmson et al., "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
- [4] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 408–417.
- [5] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2349–2358.
- [6] W. Yang, B. Liu, W. Li, and N. Yu, "Tracking assisted faster video object detection," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2019, pp. 1750–1755.
- [7] C. Guo et al., "Progressive sparse local attention for video object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3909–3918.
- [8] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2019, Art. no. 25.
- [9] W. Zhang et al., "Elf: Accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 201–214.
- [10] Z. Zhai, X. Chen, Y. Zhao, L. Zhao, J. Qian, and J. Wu, "SmartCamera: Realtime video stream-oriented action recognition platform in edge environment," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. 2021 ACM Int. Symp. Wearable Comput.*, 2021, pp. 88–89.
- [11] L. Wen et al., "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Comput. Vis. Image Understanding*, vol. 193, 2020, Art. no. 102907.
- [12] A. Furda and L. Vlacic, "Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making," *IEEE Intell. Transp. Syst. Mag.*, vol. 3, no. 1, pp. 4–17, Spring 2011.
- [13] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst.*, 2013, pp. 2063–2067.
- [14] R. Goldstein and S. Smith, "Expressive real-time intersection scheduling," in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 757.
- [15] Y. Gaobo and Y. Shengfa, "Modified intelligent scissors and adaptive frame skipping for video object segmentation," *Real-Time Imag.*, vol. 11, no. 4, pp. 310–322, 2005.
- [16] S. Jenni, G. Meishvili, and P. Favaro, "Video representation learning by recognizing temporal transformations," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 425–442.
- [17] U. Ahmed, J. C.-W. Lin, and G. Srivastava, "Unmanned aerial multi-object dynamic frame detection and skipping using deep learning on the internet of drones," *IEEE Internet Things Mag.*, vol. 4, no. 4, pp. 36–39, Dec. 2021.
- [18] Y. Luo, H. Zhou, Q. Tan, X. Chen, and M. Yun, "Key frame extraction of surveillance video based on moving object detection and image similarity," *Pattern Recognit. Image Anal.*, vol. 28, pp. 225–231, 2018.
- [19] S. Guo, C. Zhao, G. Wang, J. Yang, and S. Yang, "EC² Detect: Real-time online video object detection in edge-cloud collaborative IoT," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20382–20392, Oct. 2022.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [21] W. Shi and L. Liu, *Computing Systems for Autonomous Driving*. Berlin, Germany: Springer, 2021.
- [22] M. S. Nosrati et al., "Towards practical hierarchical reinforcement learning for multi-lane autonomous driving," in *Proc. NeurIPS*, Oct. 2018, pp. 1–7. [Online]. Available: <https://openreview.net/pdf?id=SyeFABuTcm>
- [23] B. Mirchevska, M. Blum, L. Louis, J. Boedecker, and M. Werling, "Reinforcement learning for autonomous maneuvering in highway scenarios," in *Proc. Workshop Driving Assistance Syst. Auton. Driving*, 2017, pp. 32–41.
- [24] A. Kulesza and B. Taskar, "k-DPPs: Fixed-size determinantal point processes," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 1193–1200.
- [25] L. Chen, G. Zhang, and E. Zhou, "Fast greedy MAP inference for determinantal point process to improve recommendation diversity," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5627–5638.
- [26] Z. Che et al., "D²-City: A large-scale dashcam video dataset of diverse traffic scenarios," 2019, *arXiv: 1904.01975*. [Online]. Available: <https://api.semanticscholar.org/CorpusID:102486513>
- [27] P. Dendorfer, "MOT20: A benchmark for multi object tracking in crowded scenes," 2020, *arXiv: 2003.09003*.
- [28] A. Elavarasan, "Human re-identification using Siamese convolutional neural network on NVIDIA GeForce RTX 2060," PhD dissertation, Sch. Elect. Eng., Universiti Teknologi Malaysia, Johor Bahru, Malaysia, 2021.
- [29] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz, and P. Mader, "BEVDet-Net: Bird's eye view LiDAR point cloud based real-time 3D object detection for autonomous driving," in *Proc. 2021 IEEE Int. Intell. Transp. Syst. Conf.*, 2021, pp. 2809–2815.
- [30] R. Xu, J. Lee, P. Wang, S. Bagchi, Y. Li, and S. Chaterji, "LiteReconfig: Cost and content aware reconfiguration of video object detection systems for mobile GPUs," in *Proc. 17th Eur. Conf. Comput. Syst.*, 2022, pp. 334–351.
- [31] X. Shi et al., "AdaPyramid: Adaptive pyramid for accelerating high-resolution object detection on edge devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 8, pp. 8208–8224, Aug. 2024.
- [32] D. Kang, S. Lee, C.-H. Hong, J. Lee, and H. Baek, "Batch-MOT: Batch-enabled real-time scheduling for multi-object tracking tasks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 43, no. 11, pp. 3539–3550, Nov. 2024.
- [33] D. Kang et al., "RT-MOT: Confidence-aware real-time scheduling framework for multi-object tracking tasks," in *Proc. IEEE Real-Time Syst. Symp.*, 2022, pp. 318–330.
- [34] S. Liu et al., "Self-cueing real-time attention scheduling in criticality-aware visual machine perception," in *Proc. IEEE 28th Real-Time Embedded Technol. Appl. Symp.*, 2022, pp. 173–186.
- [35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
- [36] Q. Zhou, X. Liang, K. Gong, and L. Lin, "Adaptive temporal encoding network for video instance-level human parsing," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1527–1535.
- [37] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei, "Relation distillation networks for video object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7023–7032.
- [38] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3057–3065.
- [39] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11618–11628.
- [40] S. Feng et al., "Dense reinforcement learning for safety validation of autonomous vehicles," *Nature*, vol. 615, no. 7953, pp. 620–627, 2023.
- [41] S. Shi et al., "VIPS: Real-time perception fusion for infrastructure-assisted autonomous driving," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 133–146.
- [42] L. Liu, Z. Dong, Y. Wang, and W. Shi, "Prophet: Realizing a predictable real-time perception pipeline for autonomous vehicles," in *Proc. IEEE Real-Time Syst. Symp.*, 2022, pp. 305–317.
- [43] C. Cao, X. Chen, J. Wang, Q. Song, R. Tan, and Y.-H. Li, "SGDCL: Semantic-guided dynamic correlation learning for explainable autonomous driving," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, 2024, pp. 596–604.
- [44] A. Boukerche and Z. Hou, "Object detection using deep learning methods in traffic scenarios," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–35, 2021.
- [45] A. Aboah, "A vision-based system for traffic anomaly detection using deep learning and decision trees," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4207–4212.
- [46] Z. Yang et al., "EdgeDuet: Tiling small object detection for edge assisted autonomous mobile vision," *IEEE/ACM Trans. Netw.*, vol. 31, no. 4, pp. 1765–1778, Aug. 2023.
- [47] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
- [48] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv: 2004.10934*.
- [49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
- [50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

- [51] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
- [52] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9217–9225.
- [53] L. Dong, Z. Yang, X. Cai, Y. Zhao, Q. Ma, and X. Miao, "WAVE: Edge-device cooperated real-time object detection for open-air applications," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4347–4357, Jul. 2023.
- [54] Z. Ma, Y. Yang, G. Wang, X. Xu, H. T. Shen, and M. Zhang, "Rethinking open-world object detection in autonomous driving scenarios," in *Proc. 30th ACM Int. Conf. Multimedia*, 2022, pp. 1279–1288.
- [55] Y. Zhao et al., "DETRs beat YOLOs on real-time object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 16965–16974.
- [56] S. Wang, R. Chen, H. Wu, X. Li, and Z. Feng, "YOLOH: You only look one hourglass for real-time object detection," *IEEE Trans. Image Process.*, vol. 33, pp. 2104–2115, 2024.
- [57] Y. Chen, H. Inaltekin, and M. Gorlatova, "AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
- [58] X. Xiao, J. Zhang, W. Wang, J. He, and Q. Zhang, "DNN-driven compressive offloading for edge-assisted semantic video segmentation," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1888–1897.
- [59] J. Xu et al., "Edge assisted mobile semantic visual SLAM," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1828–1837.



Zheng Dong (Member, IEEE) received the PhD degree from the Department of Computer Science, University of Texas at Dallas, in 2019. He is an assistant professor with the Department of Computer Science, Wayne State University. His research interests include real-time cyber physical systems and mobile edge computing. He received the Outstanding Paper Award at the 38th IEEE RTSS.



Guang Wang received the PhD degree in computer science from Rutgers University, New Brunswick, NJ, USA. He is an assistant professor with the Department of Computer Science, Florida State University. Before that, he was a postdoctoral research associate with the Massachusetts Institute of Technology. He is interested in mobile computing, cyber-physical systems, Big Data analytics, and machine learning. His technical contributions have led to more than 60 peer-reviewed publications in premium venues.



Tianen Liu received the BS degree from the School of Computer and Control Engineering, Yantai University. He is currently working toward the PhD degree with the School of Computer Science and Engineering, Southeast University. His research interests include autonomous driving, real-time scheduling, and edge computing.



Wei Gong (Member, IEEE) received the BS degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology, and the MS and PhD degrees from the School of Software and Department of Computer Science and Technology, Tsinghua University. He is currently a professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include backscatter networks, edge systems, and the IoT applications.



Shuai Wang (Member, IEEE) received the BS and MS degrees from the Huazhong University of Science and Technology, China, and the PhD degree from the Department of Computer Science and Engineering, University of Minnesota, in 2017. He is currently a professor with the School of Computer Science and Engineering, Southeast University. He has been serving as the head of the Computer Engineering Department since November 2022. His research interests include the wireless networks and sensors, cyber-physical systems, data science.



Tian He (Fellow, IEEE) is currently a professor with the School of Computer Science and Engineering, Southeast University. He is a fellow of ACM. He was a recipient of the NSF CAREER Award in 2009, the McKnight Land-Grant Chaired Professorship in 2011, the George W. Taylor Distinguished Research Award in 2015, the China NSF Outstanding Overseas Young Researcher I and II in 2012 and 2016, and the eight best paper awards in international conferences, including MobiCom, SenSys, and ICDCS. His research interests include wireless networks, networked



Borui Li (Member, IEEE) received the PhD degree in computer science from Zhejiang University, China, in 2022. He is now an assistant professor with Southeast University. His research interests include IoT system, edge computing, and WebAssembly. He is a member of the ACM.

sensing systems, cyber-physical systems, real-time embedded systems. He has served as the few general/program chair positions for multiple conferences and on many program committees. He also has been an editorial board member for six international journals, including the *ACM Transactions on Sensor Networks*, *IEEE Transactions on Computers*, *IEEE/ACM Transactions on Networking* and so on.