Joint Optimization of Video-based AI Inference Tasks in MEC-assisted Augmented Reality Systems

Guangjin Pan, Heng Zhang, Shugong Xu, *Fellow, IEEE*,
Shunqing Zhang, *Senior Member, IEEE*, and Xiaojing Chen

Abstract—The high computational complexity and energy consumption of artificial intelligence (AI) algorithms hinder their application in augmented reality (AR) systems. However, mobile edge computing (MEC) makes it possible to solve this problem. This paper considers the scene of completing video-based AI inference tasks in the MEC system. We formulate a mixed-integer nonlinear programming problem (MINLP) to reduce inference delays, energy consumption and to improve recognition accuracy. We give a simplified expression of the inference complexity model and accuracy model through derivation and experimentation. The problem is then solved iteratively by using alternating optimization. Specifically, by assuming that the offloading decision is given, the problem is decoupled into two sub-problems, i.e., the resource allocation problem for the devices set that completes the inference tasks locally, and that for the devices set that offloads tasks. For the problem of offloading decision optimization, we propose a Channel-Aware heuristic algorithm. To further reduce the complexity, we propose an alternating direction method of multipliers (ADMM) based distributed algorithm. The ADMM-based algorithm has a low computational complexity that grows linearly with the number of devices. Numerical experiments show the effectiveness of proposed algorithms. The trade-off relationship between delay, energy consumption, and accuracy is also analyzed.

Index Terms—Mobile augmented reality, edge intelligence, mobile edge computing, resource allocation.

I. INTRODUCTION

RECENTLY, the development of networks, cloud computing, edge computing, artificial intelligence, and other technologies has triggered people's infinite imagination of the Metaverse [1]. To enable users to interact between the real world and the virtual world, augmented reality (AR) technology plays a vital role. At the same time, artificial intelligence (AI), due to its learning and inference capabilities, has demonstrated a powerful ability in many fields such as automatic speech recognition (ASR) [2], natural language

processing (NLP) [3], computer vision (CV) [4], and so on. With the assistance of AI technology, AR can carry out deeper scene understanding and more immersive interactions.

However, the computational complexity of AI algorithms, especially deep neural networks (DNN), is usually very high. It is challenging to complete DNN inference timely and reliably on mobile devices with limited computation and energy capacity. In [5], experiments show that a typical single-frame image processing AI inference task takes about 600 ms even with speedup from the mobile GPU. In addition, continuously executing the above inference tasks can only last up to 2.5 hours on commodity devices. The above issues result in only a few AR applications currently using deep learning [6]. In order to reduce the inference time of DNNs, one way is to perform network pruning on the neural network [7], [8]. However, it could be destructive to the model if pruning too many channels, and it may not be possible to recover a satisfactory accuracy by fine-tuning [7].

Edge AI [9]–[11] is another approach to solving these problems. Integrating mobile edge computing (MEC) and AI technology has recently become a promising paradigm for supporting computationally intensive tasks. Edge AI transfers the inference and training process of AI models to the edge of the network close to the data source. Therefore, it can alleviate network traffic load, delay, and privacy problems.

A. Related Works

Many existing studies use MEC's powerful computing capabilities to reduce delay [12], energy consumption [13], or both delay and energy consumption [14]–[16] through offloading. For example, [12] formulated an optimization problem aimed at minimizing the processing delay of eMBB and mMTC users by optimizing the users' transmit power in UAV-Assisted MEC systems. [13] develops a smart pricing mechanism to coordinate the computation offloading of multi-layer devices and reduces energy consumption. [14] uses the Stackelberg game method to optimize the task allocation coefficient, calculation resource allocation coefficient, and transmission power to minimize the energy consumption and delay of the NOMA-based MEC system.

For edge AI inference, existing research has made some progress. The authors in [17] propose a framework for jointly optimizing inference task selection and downlink coordinated beamforming to minimize communication power consumption in wireless networks. Similarly, [18] proposes an IRS-assisted edge inference system and designs a task selection strategy

to minimize the energy consumption of uplink and downlink transmission and calculation. The work in [19] analyzes and models the transmission error probability, inference accuracy, and timeout probability of the AI-powered time-critical services. The work in [20] uses a tandem queueing model to analyze queueing and processing delays of DL tasks in multiple DNN partitions. [21] joint optimizes the service placement, computational and radio resource allocation to minimize the users' total delay and energy consumption. [8] combines model pruning and DNN partitioning to achieve a 4.81x reduction on end-to-end delay. [22] designs the Edgent framework that can jointly optimize DNN partitioning and DNN right-sizing to maximize the inference accuracy while promising application delay requirements. These studies measure the inference time by experiments [8], [22] or assume that the inference task's computational complexity is proportional to the input data size but without derivation and proof [20], [21]. However, these models of computational complexity are not rigorous enough or can not be generalized to different neural network models.

As for the accuracy model, the authors in [23] designs an edge network orchestration algorithm named FACT, which boosts the performance of an edge-based AR system by optimizing the edge server assignment and video frame resolution selection for AR users. However, [23] builds an accuracy model by fitting an accuracy curve for specific tasks, which is not general. The work in [24] compresses image resolution locally and performs inference tasks on edge servers, aiming to maximize learning accuracy under constraints of delay and energy. [24] proposes using an abstract non-decreasing function to describe the relationship between accuracy and input image size, which cannot be used to analyze various AI inference tasks discriminately. Joint optimization is required when different tasks and models are jointly deployed. An insufficiently generalized accuracy model or an overly abstract model can adversely affect joint optimization. A general accuracy model is needed to measure various AI tasks.

Among the above studies, most studies consider optimizing one or two performance metrics among the delay, energy consumption, and accuracy. The authors in [24] jointly considers delay, energy consumption and accuracy in image recognition scenarios. However, it aims at maximizing computational capacity under constraints of delay, energy consumption and accuracy, and the DNN model is only deployed in edge servers. In [6], [23], [25], video analytics scenarios are considered, but they do not jointly consider delay, energy and accuracy.

B. Contributions and Organizations

In this paper, we consider a multi-user MEC system and assume that each device executes the video-based DNN inference task. Each device can be AR glasses, mobile robots, and so on. In order to deepen AR's ability to understand the scene, we need to use time dimension information to improve perception. Therefore, we consider video-based application scenarios.for video-based AI inference tasks, there are two modes, e.g., frame-by-frame recognition mode (the input for each recognition is one frame) and multi-frame recognition

Uplink for real-time captured content
Downlink for inference results

Fig. 1. Multi-user MEC System model. The inference task can be executed on the local or the edge server. When the task is offloaded to the edge server, the uplink transmits the content captured in real-time, and the downlink transmits the inference result.

mode (the input for each recognition is multiple frames). The frame-by-frame inference mode is used to deal with tasks with weak temporal correlation, such as face recognition and target tracking., and has been studied in [6], [23], [25]. In this paper, we focus on multi-frame recognition tasks, such as gesture recognition and action recognition tasks. Since sampling in the spatial dimension brings extra computation [24], we only sample in the temporal domain. At each inference, the device selects the most recent several frames from the history frames for transmission or inference.

As shown in Fig. 1, mobile devices can transmit captured video to the edge server via wireless networks. The edge servers execute inference tasks and send results back to mobile devices. However, when communication and computing resources of the edge server are insufficient, devices can execute the inference task locally. We model the problem as a multi-objective optimization problem to optimize delay, energy consumption, and inference accuracy. The main contributions of this paper are summarized as follows,

- *Multi-dimensional target optimization.* High accuracy, low delay, and low energy consumption are indispensable for AR applications and must be optimized jointly. To explore the trade-off relationship between delay, energy, and accuracy, we formulate the video-based offloading problem as a mixed-integer nonlinear programming problem (MINLP), aiming to reduce service delays, energy consumption and improve recognition accuracy.
- *General computational complexity and accuracy models.* To measure the computational complexity of neural network models with different architectures and different input sizes, we introduce the number of multiply-and-accumulate operations (MACs). We illustrate the main factors affecting DNN inference delay through experiments and show that MAC can be used as a good measure of the computational complexity of DNN inference tasks. We also propose a general model to represent the relationship between the inference accuracy and the number of input frames. This model is suitable for different video-

based recognition tasks and different DNN architectures. We give simple expressions of the inference complexity and accuracy to simplify the optimization problem.

- *Channel-Aware scheduling scheme.* To solve the optimization problem, we decompose the original problem. First, assuming that the offloading decision is given, we solve the resource allocation problems for the device set that completes the inference locally and the device set that offloads the tasks to the edge server, respectively. For edge DNN inference, we propose two algorithms based on search and geometric programming (GP) to solve the problem. Then, to obtain the optimal offloading policy, we propose a Channel-Aware heuristic algorithm. The original problem is solved iteratively through alternating optimization.
- *ADMM-based distributed resource allocation scheme.* To avoid the high complexity of the heuristic algorithm, we propose an algorithm based on the Alternating direction method of multipliers (ADMM). The ADMM-based algorithm decomposes the original problem into parallel and tractable subproblems. Therefore, the total computational complexity of ADMM-based algorithms is more scalable than the heuristic algorithm, especially when the number of devices is large.

The rest of this paper is organized as follows. In Section II, we introduce system models, including delay, energy, and accuracy models. In Section III, we formulate the joint optimization problem and convert the original problem to a more tractable problem. Section IV proposes a Channel-Aware heuristic algorithm to solve the proposed problem. In Section V, we propose another ADMM-based distributed resource allocation algorithm for the proposed problem, and analyze the computational complexity of the solution algorithm. Numerical results and analysis are presented in Section VI. Finally, the paper is concluded in Section VII.

II. SYSTEM MODEL

In this section, we introduce a single-cell MEC system and establish delay, energy consumption, and accuracy models. As shown in Fig. 1, we consider a multi-user MEC system with one base station (BS) and N mobile devices, denoted by the set $\mathcal{N} = \{1, 2, \ldots N\}$. Each device has a camera and needs to accomplish DNN inference tasks. Due to the limitation of device computational resources, DNN inference tasks can be placed on local or edge servers. The limited computational resource will lead to longer computing delay and greater power consumption when the inference task is executed locally. However, when the inference task is executed on the edge server, it will bring additional wireless transmission delay. In addition, accuracy is also a very important optimization target in DNN inference tasks.

A. Offloading Framework

In this paper, we only consider the binary offloading method. Binary offloading requires the DNN inference task to be fully executed either at the device or the MEC server. The overview of the DNN computing offloading system is depicted

Fig. 2. The overview of the video sampling and computing offloading system. The video sampling management module can control the sampling rate of the captured video and determine the number of video frames used for AI inference. Devices can transmit the video to the edge server or perform inference tasks locally based on the wireless channel information and computing capabilities.

in Fig. 2. First, devices sample the video captured in real-time in the temporal dimension to obtain a short video with a certain number of frames. Second, the DNN inference tasks are executed. These inference tasks can be executed locally on devices or the edge server. Therefore, each device's video sampling management module needs to select an appropriate video sampling rate (how many frames need to be input) and choose whether to offload the task to the MEC server. Denote D_n, E_n and ϕ_n to be the total delay, energy consumption and recognition accuracy of the device n, respectively. The total delay and energy consumption of the device n can be given by,

$$D_n = (1 - x_n)D_n^{md} + x_n(D_n^t + D_n^e), \tag{1}$$

$$E_n = (1 - x_n)E_n^{md} + x_n E_n^t, \tag{2}$$

where x_n indicates whether the inference task is executed on local or edge servers. D_n^t is the transmission delay for uplink, D_n^{md} is the local inference delay, and D_n^e is the delay for completing inference at the edge server. E_n^t and E_n^{md} are the transmission and computational energy consumption, respectively. The delay and energy consumption for downloading computation results can be reasonably neglected because of the results' small data sizes.

B. Delay and Energy Models for Inference

The inference delay depends on the DNN model's architecture, the device's or server's computing power, and the input to the model. In this section, we first give a measure of the computational complexity of the DNN model and then give an expression for the inference delay and energy consumption.

Different AI recognition tasks may require different AI model architectures, including classic AI models such as Resnet-18, Resnet-34, Resnet-50, VGG-16, etc. [26], [27]. In order to optimize AI inference tasks more reasonably, different AI models need a common method to evaluate computational

complexity. In this paper, we use the number of MACs [28] to measure the computational complexity of AI inference tasks. MACs calculation methods of layers (such as fully connected (FC) layers, convolutional layers and so on) can be obtained in [28]. Taking 3D Convolutional Neural Network (3DCNN) as an example, the computational complexity (measured by MACs) of the lth layer of the nth device can be expressed as,

$$c_{n,l} = o_l o_{l+1} \prod_{j=0}^{2} K_l^j, \prod_{j=0}^{2} M_{n,l+1}^j, \qquad (3)$$

where o_l is the number of input channels, o_{l+1} is the number of output channels, $\prod_{j=0}^{2} K_l^j$ is the size of the convolution kernel, and $\prod_{j=0}^{2} M_{n,l+1}^j$ is the size of the output feature map. $j = 0$ represents the temporal dimension (the number of frames), $j = 1, 2$ represent spatial dimensions (pixels of one frame). Note that o_l, o_{l+1}, and $\prod_{j=0}^{2} K_l^j$ are all determined by the neural network architecture and $\prod_{j=0}^{2} M_{n,l+1}^j$ depends on the input size. The relation between the output feature size and the input size can be expressed as,

$$M_{n,l+1}^j = \frac{M_{n,l}^j - K_l^j + 2d_l}{r_l} + 1, \qquad (4)$$

where r_l is the stride and d_l is the padding size.

As mentioned above, the computational complexity of a DNN model is determined by the number of layers, the DNN model's architecture, and the input and output size. In this paper, we mainly focus on the impact of the number of input video frames M_n on recognition accuracy and the allocation of communication and computing resources. The inference result will be more accurate with more frames M_n input, but the communication and calculation overhead will be greater. The computational complexity of the nth device's task can be expressed as $C(M_n)$.

Then we give the expression for the inference delay and energy consumption. Denote f^{max} and f_n^{max} (in CPU cycle/s) to be the total computation resource of the edge server and mobile device n, respectively. Let f_n^e and f_n^{md} (in CPU cycle/s) denote the computation resource to device n allocated by the edge server and the device, respectively. Therefore, the computing resources satisfy $\sum_{n \in \mathcal{N}} f_n^e \leq f^{max}$ and $f_n^{md} \leq f_n^{max}$. The computation delay of the device n and MEC can be respectively expressed as,

$$D_n^{md} = \frac{\rho C(M_n)}{f_n^{md}}, \qquad (5)$$

$$D_n^e = \frac{\rho C(M_n)}{f_n^e}, \qquad (6)$$

where ρ (cycle/MAC) represents the number of CPU cycles required to complete a multiplication and addition, which depends on the CPU model.

As for energy consumption, denote κ to be a coefficient determined by the corresponding device [24], and the computational energy consumption of device n can be expressed as,

$$E_n^{md} = \kappa \rho C(M_n) f_n^{md2}. \qquad (7)$$

C. Delay and Energy Models for Transmission

We consider a time-division multiple access (TDMA) method for channel access. Specifically, each radio frame is divided into N time slots for transmission, and each device can only transmit in its own time slot. We assume that the length of each radio frame is ΔT, which is short enough (e.g., 10 ms in LTE or NR system [24]), and the length of a time slot is $\Delta T t_n$.

Denote h_n and p_n to be the channel gain and transmission power of the device n, respectively. According to [21], the achievable data rate of device n can be expressed as,

$$R_n = B_w log_2 \left(1 + \frac{p_n h_n}{B_w N_0} \right), \qquad (8)$$

where B_w and N_0 are the bandwidth and the variance of additive white Gaussian noise (AWGN), respectively.

Let d denote the data size of one video frame. Since we only want to analyze the impact of time dimension information (the number of input frames M_n) on recognition accuracy, d is a constant value. In each radio frame, the data size that can be transmitted is $\Delta T R_n t_n$. Therefore, for each transmission, $\lceil \frac{M_n d}{\Delta T R_n t_n} \rceil$ radio frames are required, where $\lceil \cdot \rceil$ means the ceil function. Considering that the length of the radio frame is much shorter than the transmission delay, the transmission delay for offloading to MEC can be written as,

$$D_n^t = \lceil \frac{M_n d}{\Delta T R_n t_n} \rceil \Delta T \approx \frac{M_n d}{R_n t_n}, \qquad (9)$$

where t_n is the proportion of time that device n transmits. In addition, according to [24], the energy consumption of each device to transmit its video can be expressed as,

$$E_n^t = \frac{M_n d}{R_n} p_n. \qquad (10)$$

D. Inference Tasks Accuracy Model

As mentioned above, we mainly focus on the impact of the number of input video frames M_n on recognition accuracy. We assume that the quality of the input video is the same for different devices. For a certain task and DNN model, the accuracy is only determined by the number of input frames. Therefore, the accuracy of device n can be expressed as $\phi_n = \Phi(M_n)$. According to [29], more frames will lead to better inference accuracy, and as the input frames continue to increase, the performance gain will gradually decrease. Some prior studies also show that the relationship between frame rate and accuracy can be expressed as concave functions [23]. Therefore, we define $\Phi(M_n)$ as a monotone non-decreasing function to describe the relationship between the accuracy and the number of input frames.

III. PROBLEM FORMULATION

In this section, we formulate the optimization problem to reduce the system's delay and devices' energy consumption and improve accuracy. We analyze the difficulty of solving the problem. To simplify the problem, we make a reasonable conversion of the problem.

A. Original Problem Formulation

Based on the above analysis, combining (1), (2), (5)-(7), (9), and(10), the nth device's delay and energy consumption can be expressed as,

$$D_n = (1 - x_n)\frac{\rho C(M_n)}{f_n^{md}} + x_n(\frac{\rho C(M_n)}{f_n^e} + \frac{M_n d}{R_n t_n}), \quad (11)$$

$$E_n = (1 - x_n)\kappa\rho C(M_n)f_n^{md2} + x_n(\frac{M_n d}{R_n}p_n). \quad (12)$$

Given the system model described previously, our goal is to reduce end-to-end delay and energy consumption and improve recognition accuracy. Each device follows the binary offloading policy. The mathematical optimization problem of the total cost (delay, energy consumption, and accuracy) can be expressed as,

Problem $\mathcal{P}1$ (*Original Problem*):

$$\underset{\{M_n,t_n,f_n^{md},f_n^e,x_n\}}{\text{minimize}} \sum_{n\in\mathcal{N}} \left(\beta_1 D_n + \beta_2 E_n - \beta_3\Phi(M_n) \right), \quad (13)$$

$$\text{subject to} \quad \Phi(M_n) \geq \alpha_n, \ \forall n \in \mathcal{N}, \quad (13a)$$

$$M_n \leq M_n^{max}, \ M_n \in \mathbb{Z}, \quad (13b)$$

$$\sum_{n\in\mathcal{N}} x_n t_n \leq 1, \quad (13c)$$

$$\sum_{n\in\mathcal{N}} x_n f_n^e \leq f^{max}, \quad (13d)$$

$$t_n, f_n^e \geq 0, \ \forall n \in \mathcal{N}, \quad (13e)$$

$$0 \leq f_n^{md} \leq f_n^{max}, \forall n \in \mathcal{N}, \quad (13f)$$

$$x_n \in \{0,1\}, \forall n \in \mathcal{N}, \quad (13g)$$

where α_n represents the recognition accuracy requirement, β_1, β_2, β_3 are the weight factors. (13a) represents the recognition accuracy requirement of each device. (13b) indicates the frame limit for the input video, $\mathbb{Z}$ is the set of integers, and M_n^{max} is the maximum number of frames of the input video. (13c) and (13d) represent the communication and computation resource limitation, respectively. (13f) limits the computation resource of each device.

The optimization variables in original problem $\mathcal{P}1$ are the number of input video frames M_n, the proportion of transmission time t_n, the local computation resource f_n^{md}, the edge computation resource allocation f_n^e, and the offloading decision x_n. In addition, the first item in (13) is to reduce the total delay of computation and transmission, the second item is to reduce the device's energy consumption, and the last item is to improve the number of input video frames as well as the recognition accuracy because of the monotone non-decreasing function $\Phi(M_n)$.

Problem $\mathcal{P}1$ is a non-convex MINLP problem and is difficult to be solved. First, the complexity function $C(M_n)$ is discrete and depends on the architecture of the DNN and the size of the input video. As the number of input frames M_n increases, the computational complexity also increases. This kind of increase is irregular because it is affected by the structure of DNN layers, such as the stride and padding size of 3DCNN according to (4). Therefore, $C(M_n)$ cannot be used for optimization directly. Second, as mentioned above, the accuracy function $\Phi(M_n)$ is non-decreasing. However, we

cannot give a deterministic expression for $\Phi(M_n)$, so we can not optimize it. In addition, both M_n and x_n are integers, making the problem difficult to be solved.

B. Problem Conversion

To make the problem $\mathcal{P}1$ more tractable, we convert the problem. First, we give an approximate expression of the computational complexity function $C(M_n)$. According to (3) and (4), the computational complexity of 3DCNN layers is proportional to the size of the input data. We can also obtain a similar conclusion in other types of layers, such as the FC layer [28]. Based on the above conclusion and combined with the experiments in Sec. VI-A, in order to simply express the computational complexity model, $C(M_n)$ can be written as,

$$C(M_n) = m_{c,0}M_n + m_{c,1}, \quad (14)$$

where $m_{c,0} \geq 0$ and $m_{c,1}$ are constants and depend on the network model.

Second, we propose a general model to express the relationship between the accuracy and the number of input video frames. Considering that the function $\Phi(M_n)$ is monotonically non-decreasing and that as the number of input frames increases, the accuracy gain decreases, combining our experiments in Sec. VI-A, we model function $\Phi(M_n)$ as,

$$\Phi(M_n) = -\frac{m_{a,0}}{M_n + m_{a,1}} + m_{a,2}, \quad (15)$$

where $m_{a,0} \geq 0$, $m_{a,2} \geq 0$ and $m_{a,1} > -1$ are constants and depend on the target of inference tasks and the architecture of DNN models.

Finally, we relax the range of the variable M_n. Considering that $\Phi(M_n)$ is a monotone non-decreasing function and depends on the recognition task and network architecture, in order not to lose generality, define $M_n^{min} = \arg\min_{M_n} \Phi(M_n)$, $\Phi(M_n) \geq \alpha_n$, $M_n \in \mathbb{Z}$. We can also relax M_n into a closed connected subset of the real axis, and (13a), (13b) can be written as $M_n \in \left[M_n^{min}, M_n^{max} \right]$. Then $[M_n]$ can be regarded as the number of input video frames, where $[\cdot]$ indicates rounding. We define two sets of devices, i.e. $\mathcal{N}_0 = \{n \mid x_n = 0, n \in \mathcal{N}\}$ and $\mathcal{N}_1 = \{n \mid x_n = 1, n \in \mathcal{N}\}$. $\mathcal{F}_{0,n}$ and $\mathcal{F}_{1,n}$ are the cost function of the device n in sets $\mathcal{N}_0$ and $\mathcal{N}_1$, respectively. The problem $\mathcal{P}1$ can be rewritten as,

Problem $\mathcal{P}2$ (*Converted Problem*):

$$\underset{\{M_n,t_n,f_n^{md},f_n^e,x_n\}}{\text{minimize}} \sum_{n\in\mathcal{N}_0} (1 - x_n)\mathcal{F}_{0,n}(M_n, f_n^{md})$$

$$+ \sum_{n\in\mathcal{N}_1} x_n\mathcal{F}_{1,n}(M_n, f_n^e, t_n), \quad (16)$$

$$\text{subject to} \quad M_n \in \left[M_n^{min}, M_n^{max} \right], \quad (16a)$$

$$(13c) - (13g),$$

where

$$\mathcal{F}_{0,n}(M_n, f_n^{md}) = \beta_1\frac{\rho C(M_n)}{f_n^{md}} + \beta_2\kappa\rho C(M_n)f_n^{md2}$$

$$- \beta_3\Phi(M_n), \quad (17)$$

$$\mathcal{F}_{1,n}(M_n, f_n^e, t_n) = \beta_1\frac{\rho C(M_n)}{f_n^e} + \beta_1\frac{M_n d}{R_n t_n}$$

$$+ \beta_2 \frac{M_n dp_n}{R_n} - \beta_3 \Phi(M_n). \tag{18}$$

IV. OPTIMIZATION PROBLEM SOLVING

In this section, we decompose the problem $\mathcal{P}2$ and propose a Channel-Aware heuristic algorithm to solve it. First, supposing that the offloading decision (i.e., $\{x_n\}$) is given, we solve optimization problems for sets $\mathcal{N}_0$ and $\mathcal{N}_1$, respectively. Second, we propose a Channel-Aware heuristic algorithm to optimize the offloading decision $\{x_n\}$.

A. Optimization Problem Solving for $\mathcal{N}_0$

For set $\mathcal{N}_0$, i.e., when the device executes inference tasks locally, the optimization problem becomes,

Problem $\mathcal{P}_{\mathcal{N}_0}$ (*Problem for $\mathcal{N}_0$*):

$$\underset{\{M_n, f_n^{md}\}}{\text{minimize}} \quad \mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}} \triangleq \sum_{n \in \mathcal{N}_0} \mathcal{F}_{0,n}(M_n, f_n^{md}), \tag{19}$$

$$\text{subject to} \quad \text{(13f), (16a).}$$

The optimization variables in $\mathcal{P}_{\mathcal{N}_0}$ are the number of input video frames M_n and the local computation resource f_n^{md}. Let $\{M_n^*, f_n^{md*}\}$ denote the optimal solution to $\mathcal{P}_{\mathcal{N}_0}$. We can derive the optimal solution to $\mathcal{P}_{\mathcal{N}_0}$ in a closed-form expression.

Theorem 1: The optimal solution to $\mathcal{P}_{\mathcal{N}_0}$ is given by,

$$f_n^{md*} = \min\{\sqrt[3]{(\frac{\beta_1}{2\beta_2 \kappa})}, f_n^{max}\}, \tag{20}$$

$$M_n^* = \min\{\max\{\sqrt{\frac{\beta_3 m_{a,0}}{\frac{\beta_1 \rho m_{c,0}}{f_n^{md}} + \beta_2 \kappa \rho m_{c,0} f_n^{md2}}} - m_{a,1}, M_n^{min}\}, M_n^{max}\}. \tag{21}$$

Proof: Please refer to Appendix A.

From *Theorem 1*, we can see that the optimal local CPU-cycle frequency f_n^{md} is determined by the weight factors β_1, β_2, the coefficient of CPU energy consumption κ, and is limited by its corresponding upper bound f_n^{max}. More specifically, f_n^{md} is proportional to $\beta_1^{\frac{1}{3}}$ and inversely proportional to $\beta_2^{\frac{1}{3}}$ and $\kappa^{\frac{1}{3}}$. As for the number of input video frames, when $\sqrt[3]{(\frac{\beta_1}{2\beta_2 \kappa})} \leq f_n^{max}$, combining (20) and (21), we have,

$$M_n^* = \min\{\max\{3^{-\frac{1}{2}} 2^{\frac{1}{3}} \rho^{-\frac{1}{2}} \kappa^{-\frac{1}{6}} m_{c,0}^{-\frac{1}{2}} \beta_1^{-\frac{1}{3}} \beta_2^{-\frac{1}{6}} \beta_3^{\frac{1}{2}} m_{a,0}^{\frac{1}{2}} - m_{a,1}, M_n^{min}\}, M_n^{max}\}. \tag{22}$$

The optimization results corresponding to each device are only related to the parameters of the device itself and are not associated with the parameters of other devices.

B. Optimization Problem Solving for $\mathcal{N}_1$

Then we solve the optimization problem of $\mathcal{N}_1$. The problem $\mathcal{P}2$ can be written as,

Problem $\mathcal{P}_{\mathcal{N}_1}$ (*Problem for $\mathcal{N}_1$*):

$$\underset{\{M_n, f_n^e, t_n\}}{\text{minimize}} \quad \sum_{n \in \mathcal{N}_1} \mathcal{F}_{1,n}(M_n, f_n^e, t_n), \tag{23}$$

$$\text{subject to} \quad \text{(13c), (13d), (13e), (16a).}$$

Algorithm 1: Algorithm 1: Search-Based Algorithm for solving $\mathcal{P}_{\mathcal{N}_1}$

Input: The offloading policy $\mathcal{N}_1$, the channel gain $\{h_n\}$, and other system parameters.

Output: $\{M_n^\star, f_n^{e\star}, t_n^\star\}$

Initialize the result of cost function $\mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}}^\star$ to a sufficiently large value;

Calculate the achievable data rate $\{R_n\}$ using (8);

foreach $\{M_n\} \in \mathcal{M}$ **do**

    Compute $\mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}}$ using (27);

    if $\mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}} < \mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}}^\star$ **then**

        $\{M_n^\star\} = \{M_n\}$; $\mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}}^\star = \mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}}$;

Calculate $\{f_n^{e\star}\}$ and $\{t_n^\star\}$ using (25) and (26);

return $\{M_n^\star\}$, $\{f_n^{e\star}\}$, and $\{t_n^\star\}$.

The optimization variables in the the problem $\mathcal{P}_{\mathcal{N}_1}$ are the number of input video frames M_n, the edge computation resource f_n^e, and the proportion of transmission time t_n. Let $\{M_n^*, f_n^{e*}, t_n^*\}$ denote the optimal solution to $\mathcal{P}_{\mathcal{N}_1}$. We can obtain the optimal solution to $\mathcal{P}_{\mathcal{N}_1}$ using the method of Lagrange multiplier. The partial Lagrangian function can be written as,

$$\mathcal{L}_{\mathcal{P}_{\mathcal{N}_1}} = \sum_{n \in \mathcal{N}_1} \left(\frac{\beta_1 \rho C(M_n)}{f_n^e} + \frac{\beta_1 M_n d}{R_n t_n} + \frac{\beta_2 M_n dp_n}{R_n} - \beta_3 \Phi(M_n) \right) + \mu_0 \left(\sum_{n \in \mathcal{N}_1} t_n - 1 \right) + \mu_1 \left(\sum_{n \in \mathcal{N}_1} f_n^e - f^{max} \right), \tag{24}$$

First of all, according to (24), supposing that M_n^* is given, we can solve the problem $\mathcal{P}_{\mathcal{N}_1}$ based on the Karush-Kuhn-Tucker (KKT) condition. We can obtain the function expressions of f_n^{e*} and t_n^* relative to M_n, as shown in the following theorem.

Theorem 2: The function expressions of f_n^{e*} and t_n^* relative to M_n^* are given by,

$$f_n^{e*} = \frac{f^{max} \sqrt{C(M_n^*)}}{\sum_{i \in \mathcal{N}_1} \sqrt{C(M_i^*)}}, \tag{25}$$

$$t_n^* = \frac{\sqrt{\frac{M_n^*}{R_n}}}{\sum_{i \in \mathcal{N}_1} \sqrt{\frac{M_i^*}{R_i}}}. \tag{26}$$

Proof: Please refer to Appendix B.

Combining (23), (25) and (26), the problem $\mathcal{P}_{\mathcal{N}_1}$ can be written as an optimized function containing only the variable M_n as follows,

Problem $\widetilde{\mathcal{P}_{\mathcal{N}_1}}$ (M_n *Optimization Problem for $\mathcal{N}_1$*):

$$\underset{\{M_n\}}{\text{minimize}} \quad \mathcal{F}_{\widetilde{\mathcal{P}_{\mathcal{N}_1}}} \triangleq \frac{\beta_1 \rho}{f^{max}} (\sum_{n \in \mathcal{N}_1} \sqrt{C(M_n)})^2 + \beta_1 d (\sum_{n \in \mathcal{N}_1} \sqrt{\frac{M_n}{R_n}})^2 + \beta_2 dp_n (\sum_{n \in \mathcal{N}_1} \frac{M_n}{R_n}) - \sum_{n \in \mathcal{N}_1} \beta_3 \Phi(M_n), \tag{27}$$

Algorithm 2: Algorithm 2: GP-Based Algorithm for solving $\mathcal{P}_{\mathcal{N}_1}$

Input: The offloading policy $\mathcal{N}_1$, the channel gain $\{h_n\}$, and other system parameters.

Output: $\{M_n^\star, f_n^{e\star}, t_n^\star\}$

Calculate the achievable data rate $\{R_n\}$ using (8);

Use the CVX tool to solve (29) and get $\{\hat{M}_n^\star\}$;

$\{M_n^\star\} = \{[e^{\hat{M}_n^\star}]\}$;

Calculate $\{f_n^{e\star}\}$ and $\{t_n^\star\}$ using (25) and (26);

return $\{M_n^\star\}$, $\{f_n^{e\star}\}$, and $\{t_n^\star\}$.

---

subject to   (16a).

Denote $\mathcal{M}_n^{opt} = \{M_n \mid M_n^{min} \leq M_n \leq M_n^{max}, M_n \in \mathbb{Z}\}$ to be the optional video frame number of device n. The optimal solution can be obtained by searching for $\{M_n\} \in \mathcal{M}$, where $\mathcal{M} = \{\{M_i\} \mid M_i \in \mathcal{M}_i^{opt}, i \in \mathcal{N}_1\}$. The detail of the search based algorithm is shown in Algorithm 1.

Considering that the problem $\mathcal{P}_{\mathcal{N}_1}$ is convex when M_n is given, Algorithm 1 is global optimal. However, When the number of devices grows large, the computational complexity of the Search-based algorithm will become very high or even unacceptable. In this paper, we also propose a GP-based sub-optimal algorithm to solve the problem $\mathcal{P}_{\mathcal{N}_1}$. First, we relax the objective function of the problem $\mathcal{P}_{\mathcal{N}_1}$. We introduce the function, $\widehat{\Phi}(M_n) = -\frac{m_{a,0}}{M_n} + m_{a,2}$, and $\mathcal{P}_{\mathcal{N}_1}$ can be rewritten as,

Problem $\mathcal{P}_{GP_{\mathcal{N}_1}}$ (*GP-based Problem for $\mathcal{N}_1$*):

$$\underset{\{M_n, f_n^e, t_n\}}{\text{minimize}} \quad \sum_{n \in \mathcal{N}_1} \left(\beta_1 \frac{\rho C(M_n)}{f_n^e} + \beta_1 \frac{M_n d}{R_n t_n} \right.$$
$$\left. + \beta_2 \frac{M_n d p_n}{R_n} - \beta_3 \widehat{\Phi}(M_n) \right), \qquad (28)$$

subject to   (13c), (13d), (13e), (16a).

It is a non-convex GP problem. Inspired by [30], the GP problem can be transformed into a convex problem by changing variables and transforming the objective and constraints. Therefore, introducing variables, $\hat{M}_n = \ln M_n$, $\hat{f}_n^e = \ln f_n^e$, $\hat{t}_n = \ln t_n$, and the problem can be written as,

Problem $\widetilde{\mathcal{P}_{GP_{\mathcal{N}_1}}}$ (*Converted GP-based Problem for $\mathcal{N}_1$*):

$$\underset{\{\hat{M}_n, \hat{t}_n, \hat{f}_n^e\}}{\text{minimize}} \sum_{n \in \mathcal{N}_1} \left(\beta_1 \rho m_{c,0} e^{\hat{M}_n - \hat{f}_n^e} + \beta_1 \rho m_{c,1} e^{-\hat{f}_n^e} \right.$$
$$\left. + \frac{\beta_1 d e^{\hat{M}_n - \hat{t}_n}}{R_n} + \frac{\beta_2 d p_n e^{\hat{M}_n}}{R_n} + \beta_3 m_{a,0} e^{-\hat{M}_n} \right), \quad (29)$$

subject to $\hat{M}_n \in \left[\ln M_n^{min}, \ln M_n^{max}\right], \forall n \in \mathcal{N}_1,$ (29a)

$$\sum_{n \in \mathcal{N}_1} x_n e^{\hat{t}_n} \leq 1, \qquad (29b)$$

$$\sum_{n \in \mathcal{N}_1} x_n e^{\hat{f}_n^e} \leq f^{max}, \qquad (29c)$$

which is strictly convex problem that can be solved using the CVX tool [31]. Considering that M_n is an integer, the result of CVX optimization needs to be post-processed. Details of the GP-based algorithm are shown in Algorithm 2.

Algorithm 3: Algorithm 3: Channel-Aware heuristic algorithm for Optimizing Offloading Policy $\{x_n\}$

Input: Parameters corresponding to the problem $\mathcal{P}1$.

Output: Offloading policy $\mathcal{N}_0$ and $\mathcal{N}_1$.

Calculate the cost function $\{\mathcal{F}_{0,n}\}$ for the set $\mathcal{N}$ using (20) and (21) ;

Set $\mathcal{N}_0 = \emptyset$, $\mathcal{N}_1 = \mathcal{N}$;

Calculate the cost function $\{\mathcal{F}_{1,n}\}$ corresponding to the set $\mathcal{N}_1$ using Algorithm 1 or Algorithm 2;

Set $Flag = 1$;

while $Flag == 1$ **do**

    $k = \text{argmin}_n h_n, n \in \mathcal{N}_1$;

    $\mathcal{N}_0^* = \mathcal{N}_0 \cup \{k\}$, $\mathcal{N}_1^* = \mathcal{N}_1 - \{k\}$;

    Calculate the cost function $\{\mathcal{F}_{1,n}^*\}$ corresponding to the set $\mathcal{N}_1^*$ using Algorithm 1 or Algorithm 2;

    if $\sum_{n \in \mathcal{N}_0} \mathcal{F}_{0,n} + \sum_{n \in \mathcal{N}_1} \mathcal{F}_{1,n} > \sum_{n \in \mathcal{N}_0^*} \mathcal{F}_{0,n} + \sum_{n \in \mathcal{N}_1^*} \mathcal{F}_{1,n}^*$ **then**

        $\mathcal{F}_{1,n} = \mathcal{F}_{1,n}^*, \forall n \in \mathcal{N}_1^*$;

        $\mathcal{N}_0 = \mathcal{N}_0^*$; $\mathcal{N}_1 = \mathcal{N}_1^*$;

    else

        $Flag = 0$;

return $\mathcal{N}_0$ and $\mathcal{N}_1$.

C. Optimization of Offloading Policy $\{x_n\}$

Considering the complexity of Search-based offloading policy algorithm becomes high when the number of devices N grows large. In this section, we propose a Channel-Aware heuristic algorithm to optimize the offloading decision $\{x_n\}$. Inspired by the *Theorem 1* and *Theorem 2*, when executing inference locally, the cost function $\mathcal{F}_{0,n}$ and optimization variables f_n^{md}, M_n only depend on the device's own parameters. However, for edge set $\mathcal{N}_1$, the cost function is related to the number and parameters of devices in the set $\mathcal{N}_1$. The Channel-Aware heuristic algorithm is shown in Algorithm 3. First, calculate the cost function $\{\mathcal{F}_{0,n}\}$ of set $\mathcal{N}_0$ when each device's task is executed locally. Second, assuming that all devices are offloaded to the edge server for inference and $|\mathcal{N}_1| = N$. In each iteration, the cost function $\{\mathcal{F}_{1,n}\}$ corresponding to each device of $\mathcal{N}_1$ is obtained. We select the device k with smallest channel gain in set $\mathcal{N}_1$. Try to put the device k from the set $\mathcal{N}_1$ into the set $\mathcal{N}_0$ and compute the cost of new sets. If the total cost of new sets is reduced, continue the next iteration. Otherwise, put the device k back to the set $\mathcal{N}_1$.

V. JOINT OPTIMIZATION USING ADMM-BASED METHOD

The complexity of the Channel-Aware heuristic algorithm becomes high when the number of UE grows. In this section, We propose an ADMM-based algorithm. The ADMM-based algorithm can decompose $\mathcal{P}2$ into N parallel sub-problems. Each user only needs to solve one sub-problem, and the average complexity of each device will be reduced.

A. ADMM-based Problem Conversion

To make the original problem tractable, we jointly consider the problem $\mathcal{P}_2$ and problem $\widetilde{\mathcal{P}_{GP_{\mathcal{N}_1}}}$, and we converted the problem into a GP-based problem,

Problem $\mathcal{P}_3$ (*Converted GP-based Problem*):

$$\underset{\{\hat{M}_n, \hat{t}_n, f_n^{\hat{m}d}, \hat{f}_n^e, x_n\}}{\text{minimize}} \sum_{n \in \mathcal{N}} \left[(1 - x_n) \hat{\mathcal{F}}_{0,n}(\hat{M}_n, f_n^{\hat{m}d}) \right.$$

$$\left. + x_n \hat{\mathcal{F}}_{1,n}(\hat{M}_n, \hat{f}_n^e, \hat{t}_n) \right], \quad (30)$$

$$\text{subject to} \quad f_n^{\hat{m}d} \leq \ln f_n^{max}, \forall n \in \mathcal{N}, \quad (30a)$$

$$(13g), (29a) - (29c),$$

where $\hat{M}_n = \ln M_n$, $f_n^{\hat{m}d} = \ln f_n^{md}$, $\hat{f}_n^e = \ln f_n^e$, and $\hat{t}_n = \ln t_n$. $\hat{\mathcal{F}}_{0,n}(\hat{M}_n, f_n^{\hat{m}d})$ and $\hat{\mathcal{F}}_{1,n}(\hat{M}_n, \hat{f}_n^e, \hat{t}_n)$ are given by,

$$\hat{\mathcal{F}}_{0,n}(\hat{M}_n, f_n^{\hat{m}d}) = \beta_1 \rho m_{c,0} e^{\hat{M}_n - f_n^{\hat{m}d}} + \beta_1 \rho m_{c,1} e^{-f_n^{\hat{m}d}}$$

$$+ \beta_2 \kappa m_{c,0} e^{\hat{M}_n + 2f_n^{\hat{m}d}}$$

$$+ \beta_2 \kappa m_{c,1} e^{2f_n^{\hat{m}d}} + \beta_3 m_{a,0} e^{-\hat{M}_n}, \quad (31)$$

$$\hat{\mathcal{F}}_{1,n}(\hat{M}_n, \hat{f}_n^e, \hat{t}_n) = \beta_1 \rho m_{c,0} e^{\hat{M}_n - \hat{f}_n^e} + \beta_1 \rho m_{c,1} e^{-\hat{f}_n^e}$$

$$+ \frac{\beta_1 d e^{\hat{M}_n - \hat{t}_n}}{R_n} + \frac{\beta_2 d p_n e^{\hat{M}_n}}{R_n}$$

$$+ \beta_3 m_{a,0} e^{-\hat{M}_n}, \quad (32)$$

The optimization variables $\{\hat{t}_n, \hat{f}_n^e\}$ are coupled among the devices in the constraints (29b) and (29c). To decompose the problem $\mathcal{P}_3$, we introduce local variables $\{y_n\}$ and $\{z_n\}$. Then, the ADMM-based problem can be written as,

Problem $\mathcal{P}_4$ (*ADMM-based Problem*):

$$\underset{\{\hat{M}_n, \hat{t}_n, f_n^{\hat{m}d}, \hat{f}_n^e, x_n, y_n, z_n\}}{\text{minimize}} \sum_{n \in \mathcal{N}} \hat{\mathcal{F}}_n(x_n, \hat{M}_n, f_n^{\hat{m}d}, y_n, z_n)$$

$$+ g(\hat{f}_n^e, \hat{t}_n), \quad (33)$$

$$\text{subject to} \quad y_n = \hat{f}_n^e, z_n = \hat{t}_n, \quad (33a)$$

$$(13g), (29a), (30a),$$

where,

$$\hat{\mathcal{F}}_n(x_n, \hat{M}_n, f_n^{\hat{m}d}, y_n, z_n) = (1 - x_n) \hat{\mathcal{F}}_{0,n}(\hat{M}_n, f_n^{\hat{m}d})$$

$$+ x_n \hat{\mathcal{F}}_{1,n}(\hat{M}_n, x_n, y_n), \quad (34)$$

$$g(\hat{f}_n^e, \hat{t}_n) = \begin{cases} 0, & \text{if} (\hat{f}_n^e, \ \hat{t}_n) \in \mathcal{G}, \\ +\infty & , \text{otherwise}, \end{cases} \quad (35)$$

and,

$$\mathcal{G} = \left\{ (\hat{f}_n^e, \ \hat{t}_n) | \sum_{n \in \mathcal{N}_1} x_n e^{\hat{t}_n} \leq 1, \sum_{n \in \mathcal{N}_1} x_n e^{\hat{f}_n^e} \leq f^{max} \right\}. \quad (36)$$

B. ADMM-based Problem Solving

The problem $\mathcal{P}_4$ can be effectively solved using the ADMM algorithm. We can write a partial augmented Lagrangian of the problem $\mathcal{P}_4$ as,

$$\mathcal{L}_4(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta}) = \sum_{n \in \mathcal{N}} \hat{\mathcal{F}}_n(x_n, \hat{M}_n, f_n^{\hat{m}d}, y_n, z_n) + g(\hat{f}_n^e, \hat{t}_n)$$

$$+ \sum_{n \in \mathcal{N}} \theta_n^f(y_n - \hat{f}_n^e) + \sum_{n \in \mathcal{N}} \theta_n^t(z_n - \hat{t}_n)$$

$$+ \sum_{n \in \mathcal{N}} \frac{s}{2}(y_n - \hat{f}_n^e)^2 + \sum_{n \in \mathcal{N}} \frac{s}{2}(z_n - \hat{t}_n)^2, \quad (37)$$

where $\boldsymbol{u} = \{x_n, \hat{M}_n, f_n^{\hat{m}d}, y_n, z_n\}$, $\boldsymbol{v} = \{\hat{f}_n^e, \hat{t}_n\}$, $\boldsymbol{\theta} = \{\theta_n^f, \theta_n^t\}$, and s is a fixed step size. Therefore, the dual function is,

$$p(\boldsymbol{\theta}) = \underset{\boldsymbol{u}, \boldsymbol{v}}{\text{minimize}} \, \mathcal{L}_4(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta}) \quad (38)$$

$$\text{subject to} \quad (13g), (29a), (30a),$$

and the dual problem can be given by,

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \, p(\boldsymbol{\theta}), \quad (39)$$

The problem (38) can be solved by iteratively updating $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{\theta}$ [32]. Let $\{\boldsymbol{u}^i, \boldsymbol{v}^i, \boldsymbol{\theta}^i\}$ denote the values in the ith iteration. In the ith iteration, the update strategies of the variables are as follows,

1) Step 1: Local variables update. In this step, we first update the local variables $\boldsymbol{u}$. Given variable $\boldsymbol{v}^i$ and $\boldsymbol{\theta}^i$, we minimize $\mathcal{L}_4(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta})$ by,

$$\boldsymbol{u}^{i+1} = \underset{\boldsymbol{u}}{\text{argminimize}} \, \mathcal{L}_4(\boldsymbol{u}, \boldsymbol{v}^i, \boldsymbol{\theta}^i). \quad (40)$$

The problem (39) can be decomposed into N parallel subproblems. For each subproblem, we consider two cases where $x_n = 0$ and $x_n = 1$, and express the problem as,

$$\begin{cases} \underset{\{\hat{M}_n, f_n^{\hat{m}d}, y_n, z_n\}}{\text{minimize}} \hat{\mathcal{F}}_{0,n}(\hat{M}_n, f_n^{\hat{m}d}) = \theta_n^f y_n + \sum_{n \in \mathcal{N}} \frac{s}{2}(y_n - \hat{f}_n^e)^2 \\ \qquad + \theta_n^t z_n + \sum_{n \in \mathcal{N}} \frac{s}{2}(z_n - \hat{t}_n)^2, & \text{if } x_n = 0, \\ \underset{\{\hat{M}_n, y_n, z_n\}}{\text{minimize}} \hat{\mathcal{F}}_{1,n}(\hat{M}_n, y_n, z_n) = \theta_n^f y_n + \sum_{n \in \mathcal{N}} \frac{s}{2}(y_n - \hat{f}_n^e)^2 \\ \qquad + \theta_n^t z_n + \sum_{n \in \mathcal{N}} \frac{s}{2}(z_n - \hat{t}_n)^2, & \text{if } x_n = 1. \end{cases}$$

$$(41)$$

These problems are both strictly convex problems that can be solved using the CVX tool [31]. Therefore, we can calculate the objective value for $x_n = 0$ and $x_n = 1$ and choose the smaller one as the final result. After solving N parallel subproblems, the optimal solution to (40) is given by $\boldsymbol{u}^{i+1} = \{(x_n)^{i+1}, (\hat{M}_n)^{i+1}, (f_n^{\hat{m}d})^{i+1}, (y_n)^{i+1}, (z_n)^{i+1}\}$.

2) Step 2: Global variables update. In the second step, we update the global variables $\boldsymbol{v}$. By the definition of $g(\boldsymbol{v})$ in (35), $\boldsymbol{v}^{i+1} \in \mathcal{G}$ must hold at the optimum. Therefore, the subproblem can be equivalently written as,

$$\boldsymbol{v}^{i+1} = \underset{\{\hat{f}_n^e, \hat{t}_n\}}{\text{argminimize}} \sum_{n \in \mathcal{N}} (\theta_n^f)^i(-\hat{f}_n^e) + \sum_{n \in \mathcal{N}} (\theta_n^t)^i(-\hat{t}_n)$$

$$+ \sum_{n \in \mathcal{N}} \frac{s}{2}(y_n^{i+1} - \hat{f}_n^e)^2 + \sum_{n \in \mathcal{N}} \frac{s}{2}(z_n^{i+1} - \hat{t}_n)^2, \quad (42)$$

$$\text{subject to,} \quad (29b), (29c).$$

The problem can also be solved by the CVX tool [31]. We propose a low-complexity scheme to solve this subproblem.

Algorithm 4: Algorithm 4: ADMM-Based Algorithm

Input: Parameters corresponding to the problem $\mathcal{P}1$.
Output: $\{x_n, M_n, f_n^{md}, f_n^e, t_n\}$

Initialize $i = 0$, $\{\boldsymbol{u}^i, \boldsymbol{v}^i, \boldsymbol{\theta}^i\} = 0$, $s = 0.5$,
$\mu_f^\star = \mu_t^\star = 10^6$, $\delta = 10^{-4}$;
repeat
    foreach $n \in \mathcal{N}$ **do**
        Update $\boldsymbol{u}^{i+1}$ by solving (41) and choose
        smaller results;
    foreach $n \in \mathcal{N}$ **do**
        Update global variables $\boldsymbol{v}^{i+1}$ using (43) and
        (44);
    foreach $n \in \mathcal{N}$ **do**
        Update multipliers $\boldsymbol{\theta}^{i+1}$ using (45) and (46);
    $i = i + 1$;
until $|\mathcal{F}^i - \mathcal{F}^{i+1}| < \delta$;
$M_n = e^{\hat{M}_n}$, $f_n^{md} = e^{\hat{f}_n^{md}}$, $f_n^e = e^{\hat{f}_n^e}$, $t_n = e^{\hat{t}_n}$;
return $\{x_n, M_n, f_n^{md}, f_n^e, t_n\}$.

Fig. 3. The theoretical delay curve, the experimental delay curve and the fitted curve corresponding to the experimentalal delay. Resnet-18 and Resnet-34 are two classic neural network architectures. The frequency of the CPU is 2.8G and 2.2G.

Considering the constraints (29b) and (29c), let μ_f and μ_t denote the Lagrangian multipliers. The closed-form optimal solution of this subproblem can be expressed as,

$$(\hat{f}_n^e)^{i+1} = y_n^{i+1} + \frac{(\theta_n^f)^i - \mu_f}{s}, \tag{43}$$

$$(\hat{t}_n)^{i+1} = z_n^{i+1} + \frac{(\theta_n^t)^i - \mu_t}{s}, \tag{44}$$

where μ_f can be obtained by the bisection search method over $(0, \mu_f^\star)$, until $\sum_{n \in \mathcal{N}_1} x_n e^{\hat{f}_n^e} \leq f^{max}$ satisfies. $\mu_f^\star$ is a sufficiently large value. It is because when $\mu_f \geq 0$, $(\hat{f}_n^e)^{i+1}$ is non-increasing. Similarly, μ_t can be obtained by the bisection search method over $(0, \mu_t^\star)$, where $\mu_t^\star$ is a sufficiently large value, until $\sum_{n \in \mathcal{N}_1} x_n e^{\hat{t}_n} \leq 1$ satisfies.

3) Step 3: Multipliers update. In this step, we update the multipliers $\boldsymbol{\theta}$ using the obtained global variables $\boldsymbol{v}$ and local variables $\boldsymbol{u}$. The updated method is,

$$(\theta_n^f)^{i+1} = (\theta_n^f)^i + s(y_n^{i+1} - (\hat{f}_n^e)^{i+1}), \tag{45}$$

$$(\theta_n^t)^{i+1} = z_n^{i+1} + s(z_n^{i+1} - (\hat{t}_n)^{i+1}), \tag{46}$$

Repeat the above three steps until the cost function no longer decreases. The cost function is $\mathcal{F}^i = \sum_{n \in \mathcal{N}}[(1 - x_n^i)\hat{\mathcal{F}}_{0,n}((\hat{M}_n)^i, (f_n^{md})^i) + x_n^i \hat{\mathcal{F}}_{1,n}((\hat{M}_n)^i, (\hat{f}_n^e)^i, (\hat{t}_n)^i)]$. We summarize solving steps of the ADMM algorithm as Algorithm 4.

As a distributed iterative algorithm, the ADMM-based scheme performs iterations between devices and BS rather than locally, enabling online optimization during the recognition process. In each iteration, $\boldsymbol{u}^i$ is calculated locally and sent to the MEC. After receiving $\boldsymbol{u}^i$ from all devices, the MEC updates $\boldsymbol{v}^i$ and $\boldsymbol{\theta}^i$, and sends them to the device to complete an iteration. Therefore, the iteration of the ADMM algorithm is an online convergence process that can adapt to slight changes in the channel.

C. Algorithm Computational Complexity Analysis

In this part, we analyze the computational complexity of proposed algorithms. First, the complexity of solving problem $\mathcal{P}_{\mathcal{N}_0}$ is $O(|\mathcal{N}_0|)$. Second, as mentioned above, the complexity of Algorithm 1 is $O(\prod_{n \in \mathcal{N}_1} |\mathcal{M}_n^{opt}|)$, and the complexity of Algorithm 2 is $O((3|\mathcal{N}_1|)^{3.5})$ by the interior-point method according to [33]. When we use Algorithm 1 for solving $\mathcal{P}_{\mathcal{N}_1}$ and use Search-based algorithm for optimizing offloading policy, the computational complexity is $O(2^N \prod_{n \in \mathcal{N}} |\mathcal{M}_n^{opt}|)$. When we use Algorithm 1 for solving $\mathcal{P}_{\mathcal{N}_1}$ and use Algorithm 3 for optimizing offloading policy, the computational complexity is $O(N \prod_{n \in \mathcal{N}} |\mathcal{M}_n^{opt}|)$. In addition, the computational complexity of Algorithm 2 for solving $\mathcal{P}_{\mathcal{N}_1}$ and Algorithm 3 for optimizing offloading policy is $O(N^{4.5})$. For the ADMM-based algorithm, as the complexity of each steps is $O(\mathcal{N})$, the overall complexity of one iteration is $O(\mathcal{N})$.

VI. NUMERICAL RESULTS

In this section, we evaluate the performance of the proposed algorithms via simulations. For all the simulation results, unless specified otherwise, we set the downlink bandwidth as $B_w = 5$ MHz and the power spectral as $N_0 = -174$ dBm/Hz [24]. According to [17], the path loss is modelled as $PL = 128.1 + 37.6 \log_{10}(D)$ dB, where D is the distance between the device and the BS in kilometres. Devices randomly distributed in the area within [500m 500m]. The computational resource of the MEC server and devices are set to be 1.8 GHz and 22 GHz, respectively. The recognition accuracy requirement and the maximum number of input video frames are set to $\alpha_n = 0.86$ and $M_n^{max} = 16$, respectively. The coefficient κ is determined by the corresponding device and is set to be 10^{-28} in this paper according to [24]. The size of the input video is $112 * 112 * M_n$. In addition, the coefficient of computational complexity ρ is set to be 0.12 cycle/MAC, which is obtained through several experiments in Sec.VI-A. Weights β_1, β_2, β_3 are set to be 0.2, 0.2, 0.6, respectively.

A. Model Verification

First, we obtain the complexity coefficient through experimental measurement. The calculation method of the computational complexity coefficient is as follows. First, calculate

Fig. 4. The experimental and fitted curves of gesture recognition task and action recognition task.

Fig. 5. The average cost of proposed schemes and baseline schemes under a different number of devices.

the MACs of the DNN model when the number of input video frames is different, recorded as $\{C\}$. We use the Flops Counter tool [34] for MACs calculation. Second, execute 100 times of inference tasks with a different number of input video frames, and record the average inference delay as $\{t\}$. Finally, calculate the coefficients between the inference delay and MACs by $\rho = \frac{sum(\{C\})}{sum(\{t\})}$. We use Intel(R) Xeon(R) E5-2630 CPU for testing. We use the Resnet-18 and the Resnet-34 for testing and limit the maximum frequency of the CPU to 2.8G and 2.2G. Fig. 3 shows the theoretical (MAC-based) and experimental delay curves and the fitted curve corresponding to the experimental delay. We can observe from Fig. 3 that the theoretical delay is similar to the experimental delay, proving that MACs can be modelled as computational complexity. We also find that the linear fitted curve can approximately represent the computational complexity with 9 ms root mean square error (RMSE) for Resnet-18 and 2.8G, 17 ms RMSE for Resnet-34 and 2.8G, and 11 ms RMSE for Resnet-18 and 2.2G. The inference delay is associated with the number of input frames, DNN model's architecture and the device's capabilities. In addition, the computational complexity coefficients under the three conditions are 0.128, 0.122, and 0.123, respectively. Therefore, in following experiments, we set $\rho = 0.12$ cycle/MAC.

We select the gesture and action recognition tasks to verify the accuracy model. We use the Jester datasets [35], the largest publicly available hand gesture dataset, to test the gesture recognition task. For the action recognition task, we use Kinetics-400 datasets [36]. We choose Resnet-18 and Resnet-101 for testing. As shown in Fig. 4, Under different tasks and different network models, the accuracy curve all conforms to the characteristics of a non-decreasing function. What's more, as the number of input frames increases, the performance gain of accuracy will gradually decrease. This is because the information gain introduced in the temporal domain decreases when the number of input frames increases. The fitted curve can approximately represent the relationship between the accuracy and the number of input frames. In the gesture recognition task with the Resnet-101 model, the gesture recognition task with the Resnet-18 model, and the action recognition task with the Resnet-101 model, the RMSE are 0.0054, 0.0048 and 0.0095, respectively. We take the

Resnet-18 and the gesture recognition task as examples for the following experiments.

B. Simulation Results of Average Cost

In this section, we compare proposed schemes and some baseline schemes. We run 100 tests and can calculate the average cost of each device and the average running time of each test. We compare the following schemes.

1) Search+Search: We use the Search-based algorithm to solve $\mathcal{P}_{\mathcal{N}_1}$ and use the heuristic algorithm to optimize offloading policy.

2) Search+Heuristic: We use the Search-based algorithm to solve $\mathcal{P}_{\mathcal{N}_1}$ and use the Search-based algorithm to optimize offloading policy.

3) GP+Heuristic: We use the GP-based algorithm to solve $\mathcal{P}_{\mathcal{N}_1}$ and use the Channel-Aware heuristic algorithm to optimize offloading policy.

4) ADMM: We use the ADMM-based algorithm to solve the original problem.

5) CCCP [37]: We use the concave-convex procedure (CCCP) algorithm to decide whether to offload inference tasks to edge servers. Then we use *Theorem 1* and the GP-based algorithm for resource allocation.

6) Random: All inference tasks are randomly executed on local or the edge server. We use *Theorem 1* and the GP-based algorithm for resource allocation.

7) Local: All inference tasks are executed locally. We use *Theorem 1* for local resource allocation.

8) Edge: All inference tasks are executed on the edge server. We use the GP-based algorithm for resource allocation

In Fig. 5, we plot the average cost of different schemes under different devices. The Search+Heuristic scheme and Search+Search scheme have the same performance, representing the performance bounds. When the number of devices exceeds 16, the performance bounds are not shown due to their unacceptable computational complexity. It can be seen from Fig. 5 that the proposed schemes are better than the baseline schemes. Compared with the performance bounds, the performance of the GP+Heuristic scheme has a slight decrease due to the relaxation of the accuracy function $\Phi(M_n)$. The performance of the ADMM scheme is worse than that of the GP+Heuristic scheme, and is better than that of the CCCP

Fig. 6. The average running time of proposed algorithms under a different number of devices.

Fig. 7. The curve corresponding to the cost function and the number of iterations.

scheme. For example, when the number of devices is 16, the CCCP, ADMM, and GP+Heuristic schemes have performance losses of 2.1%, 0.24%, and 0.03%, respectively, compared with performance bounds. Moreover, when the number of devices is less than 8, the cost of the scheme that executes tasks only at the edge is almost equal to the cost of the proposed GP+Heuristic scheme. It is because all devices can benefit from performing inference on the edge server when the number of devices is small. If the inference task is only executed locally, the average cost of the device will not change because the local resources among the equipment do not affect each other.

In Fig. 6, we plot the average running time of different schemes under different devices. When the number of devices exceeds 6, the running time of the Search+Heuristic and Search+Search scenarios becomes unacceptable. The GP+Heuristic scheme improves the solution efficiency. The running time of GP+Heuristic is shorter than that of CCCP scheme. However, the complexity of the solution remains unsatisfactory as the number of devices increases. As for the ADMM-based scheme, since the ADMM-based algorithm is a distributed algorithm and the complexity of updating global variables is much smaller than that of updating local variables, we only consider the average running time for each device. The average running time of the ADMM-based scheme does not improve as the number of devices increases. It is worth noting that in the ADMM-based scheme, the iteration stops when $|\mathcal{F}^i - \mathcal{F}^{i+1}| < \delta$, where $\delta = 10^{-5}$. Threshold-based stopping conditions result in a different number of iterations in different cases. When the number of devices is different, the average number of iterations is also different, resulting in different running times. Therefore, the average running time of 18 devices is shorter than that of 14 and 22 devices.

Assuming that the ADMM-based scheme iterates once every time an inference task is performed, we plot the curve corresponding to the cost function and the number of iterations. As shown in Fig.7, the ADMM-based scheme can converge to acceptable performance after completing 3-5 iterations. As the number of iterations increases, the performance will be closer to the optimal performance. It shows that the ADMM algorithm can converge through online iterations. We also test the running time per iteration on each device, and it takes an average of about 278ms.

TABLE I
DELAY, ENERGY CONSUMPTION, AND ACCURACY OF LOCAL DEVICES AND EDGE DEVICES

	Local devices	Edge devices
Number of devices	12.3	12.7
Average delay	0.24 s	0.52 s
Average energy	1.00 J	0.025 J
Average accuracy	0.886	0.866

C. Simulation Results of Delay, Energy, and Accuracy

This section compares the average delay, energy consumption, accuracy, and the offloading rate (the proportion of devices that perform inference on the edge server). We consider the different number of devices, bandwidths, edge computing resources, and weights β_1, β_2, β_3. We use the GP+Heuristic scheme for testing. Table. I shows a comparison of devices that finish inference locally and devices that finish inference at the edge under default experimental settings. On average, 12.7 devices choose to offload to the edge server to perform inference. Compared with edge devices, local devices have a lower delay and higher accuracy but have greater inference energy consumption.

Fig. 8 shows the average delay, energy, accuracy, and offloading rate under different numbers of devices, different bandwidths, and different edge computing resources. In Fig. 8(a), we plot results with different numbers of devices. As shown in Fig. 8(a), when the number of devices is small (less than 10), all devices offload the task to the edge server (the offloading rate is equal to 1). For edge devices, all delay comes from transmission delay and the edge inference delay, and all energy consumption comes from transmission energy. With the number of devices increasing, communication resources and the edge server's computation resources are shared by more devices, decreasing the number of input frames M_n. A decrease in the number of input frames results in a decrease in accuracy. Then as M_n decreases, the transmission data size decreases, and the transmission energy decreases. Meanwhile, Competition from more devices leads to increased delays. Therefore, when the number of devices is small (less than 10), with the number of devices increasing, the average delay increases, the average accuracy and the average energy con-

(a) Different number of devices

(b) Different bandwidth

(c) Different edge computing resource

Fig. 8. The average delay, energy, offloading rate, and accuracy under different numbers of devices, different bandwidths, and different edge computing resources.

Fig. 9. The relationship between the delay, energy consumption, and accuracy.

with the increase of bandwidth and edge computing resources, more edge devices lead to increased delay and decreased energy and accuracy.

We set the minimum number of input frames $M_n^{min} = 1$. We use different weights, β_1, β_2, β_3 to study the trade-off relationship between the average delay, energy consumption, and accuracy. The constraint is $\beta_1 + \beta_2 + \beta_3 = 1$. The performance of the trade-off surface is obtained by the GP+Heuristic scheme. Fig. 9 shows the delay, energy consumption, and accuracy are mutually limited. Higher energy consumption leads to higher accuracy when the delay is constant. From another perspective, in order to improve the accuracy, it is necessary to sacrifice the performance of delay and energy consumption. In addition, with the same accuracy, according to Table. I, higher energy consumption will make the device more inclined to execute inference tasks locally, and the delay decreases.

VII. CONCLUSION

This paper considers optimizing video-based AI inference tasks in a multi-user MEC system. An MINLP is formulated to minimize the total delay and energy consumption, and improve the total accuracy, with the constraint of computation and communication resources. A MAC-based computational complexity model is introduced to model the calculation delay, and a simple approximate expression is proposed to simplify the problem. We also propose a general accuracy model to characterize the relation between the recognition accuracy and the number of input frames. After that, we first assume that the offloading decision is given and decouple the original problem into two sub-problems. The first sub-problem is to optimize the resources of the devices that complete the DNN inference tasks locally. We derive the closed-form solution to this problem. The second sub-problem is optimizing the devices' resources that offload the DNN inference tasks to the edge server. We propose the Search-based and GP-based algorithm to solve the second sub-problem. For the problem of offloading decision optimization, we propose the Channel-Aware heuristic algorithm. We also propose a distributed algorithm based on ADMM. The ADMM-based algorithm reduce computational complexity at the cost of an acceptable performance loss. Numerical simulation and experimental results demonstrate

sumption decrease. When the number of devices exceeds 10, the average energy consumption and accuracy increase, and the average delay and offload rate gradually decrease. Considering different bandwidths and different edge computing resources, we plot Fig. 8(b) and Fig. 8(c). In Fig. 8(b) and Fig. 8(c), as the bandwidth and edge computing resource increase, devices will be more inclined to offload computing to the edge, which increases the offloading rate. According to Table. I, when β_1, β_2 and β_3 are fixed, edge devices have lower energy consumption, lower accuracy and higher delay. More edge devices mean a greater delay and lower power consumption. Meanwhile, when the bandwidth increases, since the edge computing resources are fixed, the number of video frames will decrease to reduce edge computing overhead, resulting in a decrease in accuracy. The same conclusion can also be obtained when edge computing resources increase. Therefore,

the effectiveness of the proposed algorithm. We also provide a detailed analysis of the delay, energy consumption, and accuracy for different device numbers, bandwidths and edge computing resources.

APPENDIX A
PROOF OF THEOREM 1

The partial derivative of $\mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}$ with respect to f_n^{md} is,

$$\frac{\partial \mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}}{\partial f_n^{md}} = -\beta_1 \frac{\rho C(M_n)}{f_n^{md2}} + 2\beta_2 \kappa \rho C(M_n) f_n^{md}, \quad (47)$$

By setting $\frac{\partial \mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}}{\partial f_n^{md}} = 0$, we have,

$$f_n^{md} = \sqrt[3]{\left(\frac{\beta_1}{2\beta_2 \kappa}\right)}, \quad (48)$$

Therefore, f_n^{md} decreases monotonically in the interval $\left(-\infty, \sqrt[3]{\left(\frac{\beta_1}{2\beta_2 \kappa}\right)}\right)$ and increases monotonically in the interval $\left(\sqrt[3]{\left(\frac{\beta_1}{2\beta_2 \kappa}\right)}, +\infty\right)$. Considering the value range of f_n^{md}, the optimal solution can be given by,

$$f_n^{md*} = \min\{\sqrt[3]{\left(\frac{\beta_1}{2\beta_2 \kappa}\right)}, f_n^{max}\} \quad (49)$$

Then we analyze M_n. The partial derivative of $\mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}$ with respect to M_n is,

$$\frac{\partial \mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}}{\partial M_n} = \frac{\beta_1 \rho m_{c,0}}{f_n^{md}} + \beta_2 \kappa \rho m_{c,0} f_n^{md2} - \frac{\beta_3 m_{a,0}}{(M_n + m_{a,1})^2}, \quad (50)$$

By setting $\frac{\partial \mathcal{F}_{\mathcal{P}_{\mathcal{N}_0}}}{\partial M_n} = 0$, we have,

$$M_n = \sqrt{\frac{\beta_3 m_{a,0}}{\frac{\beta_1 \rho m_{c,0}}{f_n^{md}} + \beta_2 \kappa \rho m_{c,0} f_n^{md2}}} - m_{a,1}, \quad (51)$$

Considering the value range of M_n, the optimal solution can be given by,

$$M_n^* = \min\{\max\{\sqrt{\frac{\beta_3 m_{a,0}}{\frac{\beta_1 \rho m_{c,0}}{f_n^{md}} + \beta_2 \kappa \rho m_{c,0} f_n^{md2}}} - m_{a,1}, M_n^{min}\}, M_n^{max}\} \quad (52)$$

which completes the proof.

APPENDIX B
PROOF OF THEOREM 2

According to the KKT conditions, we can obtain the following necessary and sufficient conditions,

$$\frac{\partial \mathcal{L}_{\mathcal{P}_{\mathcal{N}_1}}}{\partial f_n^{e*}} = -\frac{\beta_1 \rho C(M_n^*)}{f_n^{e*2}} + u_1^* = 0, \ f_n^{e*} > 0, \quad (53)$$

$$\frac{\partial \mathcal{L}_{\mathcal{P}_{\mathcal{N}_1}}}{\partial t_n^*} = -\frac{\beta_1 M_n^* d}{R_n t_n^{*2}} + u_0^* = 0, t_n^* > 0, \quad (54)$$

$$\mu_0^* \left(\sum_{n \in \mathcal{N}^*} t_n^* - 1\right) = 0, \quad (55)$$

$$\mu_1^* \left(\sum_{n \in \mathcal{N}^*} f_n^{e*} - f^{max}\right) = 0, \quad (56)$$

$$\mu_0^*, \mu_1^* \geq 0. \quad (57)$$

Because $\frac{\beta_1 \rho C(M_n^*)}{f_n^{e*2}}$ and $\frac{\beta_1 M_n^* d}{R_n t_n^{*2}}$ are positive, μ_0^* and μ_1^* are also positive. We can obtain,

$$\sum_{n \in \mathcal{N}} f_n^{e*} - f^{max} = 0, \quad (58)$$

$$\sum_{n \in \mathcal{N}} t_n^* - 1 = 0, \quad (59)$$

$$f_n^{e*} = \sqrt{\frac{\beta_1 \rho C(M_n^*)}{R_n \mu_1^*}}, \quad (60)$$

$$t_n^* = \sqrt{\frac{\beta_1 M_n^* d}{R_n \mu_0^*}}. \quad (61)$$

Combining (58) and (60), we can get the expression of f_n^{e*} corresponding to M_n^*,

$$f_n^{e*} = \frac{f^{max} \sqrt{C(M_n^*)}}{\sum_{i \in \mathcal{N}_1} \sqrt{C(M_i^*)}}. \quad (62)$$

Similarly, combining (59) and (61), we can get the expression of t_n^* corresponding to M_n^*,

$$t_n^* = \frac{\sqrt{\frac{M_n^*}{R_n}}}{\sum_{i \in \mathcal{N}_1} \sqrt{\frac{M_i^*}{R_i}}}, \quad (63)$$

which completes the proof.

REFERENCES

[1] H. Ning, H. Wang, Y. Lin, W. Wang, S. Dhelim, F. Farha, J. Ding, and M. Daneshmand, "A survey on metaverse: the state-of-the-art, technologies, applications, and challenges," *arXiv preprint arXiv:2111.09673*, 2021.

[2] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 4, pp. 745–777, 2014.

[3] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, 2021.

[4] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2014.

[5] L. N. Huynh, R. K. Balan, and Y. Lee, "Deepsense: A gpu-based deep convolutional neural network framework on commodity mobile devices," in *Proc. ACM WearSys'16*, 2016, p. 25–30.

[6] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE INFOCOM'18*, 2018, pp. 1421–1429.

[7] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE ICCV'17*, Oct 2017, pp. 2736–2744.

[8] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE INFOCOM WKSHPS'19*, 2019, pp. 1–6.

[9] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2167–2191, 2020.

[10] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, 2019.

[11] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun*, vol. 40, no. 1, pp. 5–36, 2022.

[12] S. R. Sabuj, D. K. P. Asiedu, K.-J. Lee, and H.-S. Jo, "Delay optimization in mobile edge computing: Cognitive uav-assisted embb and mmtc services," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 2, pp. 1019–1033, 2022.

[13] P. Wang, B. Di, L. Song, and N. R. Jennings, "Multi-layer computation offloading in distributed heterogeneous mobile edge computing networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 2, pp. 1301–1315, 2022.

[14] K. Wang, Z. Ding, D. K. C. So, and G. K. Karagiannidis, "Stackelberg game of energy consumption and latency in mec systems with noma," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2191–2206, 2021.

[15] M. Qin, N. Cheng, Z. Jing, T. Yang, W. Xu, Q. Yang, and R. R. Rao, "Service-oriented energy-latency tradeoff for iot task partial offloading in mec-enhanced multi-rat networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1896–1907, 2021.

[16] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, 2021.

[17] K. Yang, Y. Shi, W. Yu, and Z. Ding, "Energy-efficient processing and robust wireless cooperative transmission for edge inference," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9456–9470, 2020.

[18] S. Hua, Y. Zhou, K. Yang, Y. Shi, and K. Wang, "Reconfigurable intelligent surface for green edge inference," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 964–979, 2021.

[19] J. Liu and Q. Zhang, "To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: An experimental study," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9357–9371, 2020.

[20] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint DNN partition deployment and resource allocation for delay-sensitive deep learning inference in IoT," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9241–9254, 2020.

[21] Z. Lin, S. Bi, and Y.-J. A. Zhang, "Optimizing AI service placement and resource allocation in mobile edge intelligence systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7257–7271, 2021.

[22] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, 2020.

[23] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE INFOCOM'18*, 2018, pp. 756–764.

[24] Y. He, J. Ren, G. Yu, and Y. Cai, "Optimizing the learning performance in mobile augmented reality systems with CNN," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5333–5344, 2020.

[25] Y. Zhao, Z. Yang, X. He, X. Cai, X. Miao, and Q. Ma, "Trine: Cloud-edge-device cooperated real-time video analysis for household applications," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2022.

[26] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?" in *Proc. IEEE CVPR'18*, 2018, pp. 6546–6555.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR'15*, May 2015, pp. 1–14.

[28] M. Hollemans, "How fast is my model?" https://machinethink.net/blog/how-fast-is-my-model/, accessed Dec. 30, 2021.

[29] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE INFOCOM'20*, 2020, pp. 257–266.

[30] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[31] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[32] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, 2018.

[33] J. Li, X. Li, Y. Bi, and J. Ma, "Energy-efficient joint resource allocation with reconfigurable intelligent surfaces in symbiotic radio networks," *IEEE Trans. Cogn. Commun. Netw.*, pp. 1–1, 2022.

[34] V. Sovrasov, "Flops counter for convolutional networks in pytorch framework," https://github.com/sovrasov/flops-counter.pytorch, accessed Dec. 30, 2021.

[35] J. Materzynska, G. Berger, I. Bax, and R. Memisevic, "The jester dataset: A large-scale video dataset of human gestures," in *Proc. IEEE ICCV Workshop'19*, 2019, pp. 2874–2882.

[36] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[37] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, and L. Hanzo, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2246–2262, 2020.