



ChatCam: Embracing LLMs for Contextual Chatting-to-Camera with Interest-Oriented Video Summarization

KAIJIE XIAO, College of Computer Science, Zhejiang University, China

YI GAO, College of Computer Science, Zhejiang University, China

FU LI, College of Computer Science, Zhejiang University, China

WEIFENG XU, School of Software Technology, Zhejiang University, China

PENGZHI CHEN, School of Software Technology, Zhejiang University, China

WEI DONG, College of Computer Science, Zhejiang University, China

Cameras are ubiquitous in society, with users increasingly looking to extract insights about the physical world. Current human-to-camera interaction methods, while advanced, still need to support an intuitive, conversational interaction as one would expect in human-to-human communication. To achieve a more natural interaction between humans and cameras, we proposed a novel contextual chatting-to-camera paradigm. This paradigm allows users to interact with the camera using natural language including raising interests and questions. In response, the camera can customize specific tasks tailored to these interests and attempt to provide answers to the questions asked. We designed ChatCam, embracing LLMs for contextual chatting-to-camera with interest-oriented video summarization. With a novel prompt with the actor-critic LLMs approach, ChatCam can understand users' interests and translate them into some tasks and objects. ChatCam can also customize relevant models with the help of the multi-modal large language model and deep reinforcement learning on the resource-constrained edge and maintain high accuracy. Results show that ChatCam achieves an improvement up to 43.9% in understanding user interests and 21.1% in model accuracy compared to state-of-the-art methods in multiple settings. Various examples and the user study also prove the effectiveness of ChatCam in practice.

CCS Concepts: • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*.

Additional Key Words and Phrases: Video summarization, Large language model, Deep reinforcement learning, Vector database

ACM Reference Format:

Kaijie Xiao, Yi Gao, Fu Li, Weifeng Xu, Pengzhi Chen, and Wei Dong. 2024. ChatCam: Embracing LLMs for Contextual Chatting-to-Camera with Interest-Oriented Video Summarization. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 4, Article 168 (December 2024), 34 pages. <https://doi.org/10.1145/3699731>

1 Introduction

Video cameras have become ubiquitous in modern society, with their presence spanning across smart homes, traffic monitoring systems, retail environments [15, 21, 23, 32, 44, 49, 62], etc. The extensive video data they capture offers users the opportunity to conduct queries that provide a more profound understanding of the

Yi Gao and Wei Dong are the corresponding authors.

Authors' Contact Information: **Kaijie Xiao**, College of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China, xiaokj@zju.edu.cn; **Yi Gao**, College of Computer Science, Zhejiang University, China, gaoyi@zju.edu.cn; **Fu Li**, College of Computer Science, Zhejiang University, China, lileaf@zju.edu.cn; **Weifeng Xu**, School of Software Technology, Zhejiang University, China, xuwf@zju.edu.cn; **Pengzhi Chen**, School of Software Technology, Zhejiang University, China, chenpz@zju.edu.cn; **Wei Dong**, College of Computer Science, Zhejiang University, China, dongw@zju.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2474-9567/2024/12-ART168

<https://doi.org/10.1145/3699731>

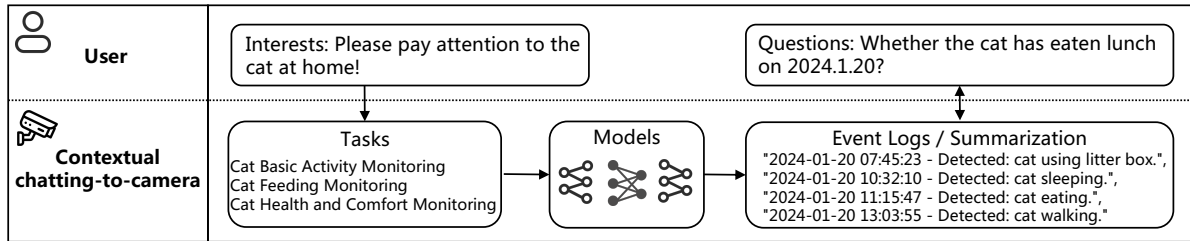


Fig. 1. Two key components in contextual chatting-to-camera paradigm. Users can use natural language to interact with the camera what they need to pay attention to, and then the camera needs to generate relevant tasks and automatically customize these models. After these models analyze the video and record several event logs, users can also ask relevant questions in natural language and cameras need to achieve this contextual QA.

physical world, as well as insights into their personal lives and relationships [13, 28, 44, 48, 50, 79]. For instance, FMT [44] allows video captured from a camera to help the elderly determine if they have completed certain actions.

Considering that users have different preferences for video, intelligent algorithms for query-focused video summarization [13, 36, 38–40, 48, 56, 64, 71, 82] have thus emerged as a pressing need. Traditional approaches [36, 39, 40] are designed to address “after the fact” queries on large-scale recorded video datasets. These solutions are typically cloud-based and, as such, require extensive resources, including network bandwidth and computational power. As the volume of video data continues to grow exponentially, these systems face increasingly unsustainable bandwidth costs and substantial query latency due to data transmission demands.

With the development of deep neural networks (DNNs) as well as edge computing, recent researches [13, 38, 48, 71, 82] have suggested deploying some long-running tasks that predefined by users directly on cameras or at the edge. The video summarizations generated from these tasks are designed to address users’ detailed queries more efficiently, reducing the need for extensive bandwidth and computational resources by processing content closer to the source of data generation. Since edge models need to be customized on demand, these tasks are often limited to requests from a fixed set of predefined concepts, which limits the subjectivity of users. In addition, the human-to-camera interaction methods in these works, while advanced, still need to support an intuitive, conversational interaction as one would expect in human-to-human communication.

Recent advancements have also seen the emergence of open-vocabulary detection systems [27, 46, 51] and multi-modal large language models (MLLM, e.g., GPT-4V [5], Gemini [4], CogVLM [72], CogAgent [33]) as a leading trend. There are also some works [43, 54, 59, 86, 88] that use “large multimodal models (LMM)” to represent large models that process multimodal data. Several works [58, 66, 76] try to generate descriptions of video frames and then answer user-specific questions through retrieval. VidIL [76] employs the pretrained image captioner in the image-language model BLIP [46] to obtain frame captions. PDL [66] generates captions from individual video frames using a pre-trained image captioning model GIT [68]. These video summarizations are often generic and fail to consider user interests. Despite the challenge posed by the large volume of video data, the strength of these large models lies in their ability to accurately classify user-abstracted queries during the inference phase, providing a more natural interaction.

Combining the advantages of cloud-edge collaboration query-focused video summarization and LLM, we propose a novel human-to-camera interaction paradigm, i.e. *contextual chatting-to-camera*. As Figure 1 shows, this paradigm first seeks to revolutionize the way users *interact* with cameras, rendering the experience as natural as conversing with another human. This paradigm allows users to express their specific interests to the camera, and the camera is then responsible for creating appropriate models to address these interests. For instance, a user

might request a home camera to monitor his cat at home, which could encompass specific tasks (e.g., cat activity monitoring, cat feeding monitoring). Second, this paradigm must facilitate *contextual* dialogues between users and cameras, enabling users to query incidents related to their expressed interests. For example, the owner of the cat may ask whether the cat has eaten lunch on a specific day and the cameras should retrieve some related event logs to answer the user.

To realize the contextual chatting-to-camera paradigm, we present *ChatCam* with *interest-oriented video summarization*. Here, these summarizations have the same meaning as the event logs. ChatCam will first translate the user's interests into some feasible tasks and generate some models for solving these tasks. By deploying sophisticated models at the edge, ChatCam can efficiently generate video summarizations, which give ChatCam a *memory* capability that allows it to use contextual information to answer user questions. ChatCam must overcome the following two challenges concerning how to understand users and customize models at the edge.

Challenge 1: How to translate the user's interests into some feasible tasks correctly? The importance of scenes in a video is often subjective, and users should be able to customize their "interests" using natural language to specify topics that are of concern to the users [56, 63, 64, 79]. To better cater to user needs, ChatCam must possess the capability to comprehend user intentions and subsequently generate suitable tasks to complete the video summary.

Leveraging the advanced natural language processing capabilities of large language models (LLMs), we propose a *Prompt with Actor-Critic LLMs* method to enhance the system's understanding of users' interests. This method is a novel application of LLMs, designed to understand user interests through the interaction of the two LLMs that ensures both accuracy and relevance.

At the outset of our method, the ActorLLM is employed to decode the user's interest and subsequently devise a suite of tasks presumed to be in concordance with the user's expressed interest. Following the generation of tasks by ActorLLM, another CriticLLM will enter the workflow to critically assess the tasks produced by the ActorLLM. With the critical feedback from CriticLLM in hand, ActorLLM will refine its task output. This prompt with actor-critic LLMs approach functions as a quality assurance mechanism, capitalizing on the LLM's capacity to emulate complex decision-making processes to enhance the quality of output.

Challenge 2: How to generate and deploy models at resource-constrained edge with high model accuracy? Upon the generation of tasks by ActorLLM, the edge server is tasked with creating the corresponding models. However, due to the unpredictable nature of user interests, it is impractical to prepare an exhaustive array of models and their associated datasets. Additionally, resource constraints at the edge significantly limit the deployment of multiple models [12, 48, 57]. For instance, a typical edge server Jetson Xavier NX [6] with a memory capacity of 8 GB is capable of supporting just 2 instances of the more resource-intensive Faster RCNN [61].

To address this challenge, we first propose *MLLM-based Model Generation* which will utilize multi-modal large language models to label the pertinent unlabeled datasets about the users' interests. This process results in the creation of deployable, lightweight models that are suitable for edge servers. While weight sharing among models [57, 65] has been identified as a potential solution for enabling real-time video analytics on edge servers, indiscriminate weight sharing can result in suboptimal performance. We then introduce a novel methodology, termed *DRL-based Model Deployment*. Our method harnesses the capabilities of deep reinforcement learning (DRL) to implement a sophisticated decision-making mechanism that intelligently determines the most effective strategy for sharing weights among different models with guaranteed model accuracy.

Subsequently, ChatCam also incorporates a *VectorDB-based Query System*. When a user poses a question to ChatCam, the system can employ vector-similarity techniques in the vector database to retrieve the most relevant video summarizations and answer the questions with the assistance of LLM. We evaluated ChatCam extensively and results show that ChatCam significantly outperforms existing approaches. We further conducted a case study that shows the feasibility of ChatCam. We summarize the contributions of ChatCam as follows:

- We propose a novel paradigm known as *contextual chatting-to-camera* which realizes a more intelligent interaction with cameras.
- We design *ChatCam* system with *interest-oriented video summarization* that can understand the users' interests correctly, *Prompt with Actor-Critic LLMs* to improve LLM performance in understanding users' interests, *MLLM-based Model Generation* to generate lightweight models and *DRL-based Model Deployment* to customize these models at the edge.
- We implemented a video summarization query system based on the vector database and evaluated ChatCam under different settings. Results show that ChatCam can understand users' interests well, which improves the rating by 23.1%-43.9% and the ISR by 42.9%-132%. In addition, ChatCam can improve the model accuracy by 11.9%-21.1% under different memory constraints.
- We provided various examples and conducted a user study to highlight the effectiveness of ChatCam.

2 Related Work

There are mainly several existing approaches that are closely related to ChatCam. Incorporating user queries enhances answer relevance, leading to the development of query-focused summarization. In addition, with the development of language models, many works have proposed open-vocabulary detection and multi-modal LLMs, which can meet the personalized needs of users. Some LLM-based multi-agent approaches have also improved the ability to solve complex problems. We will briefly describe these works and summarize their major differences from ChatCam in Table 1. Simultaneously, there has been considerable research exploring strategies for deploying models in resource-constrained environments.

Table 1. Difference between existing approaches and ChatCam.

Approaches	Edge Computing Resource	Network Resource	Initial Detailed Query
Cloud Query Focused Video Summarization [36, 39, 40]	Low	High	Yes
Edge-Cloud Query Focused Video Summarization [13, 48, 71]	High	Medium	Yes
Open-vocabulary Detection [27, 46, 51] and MLLM [5, 33, 59]	Low	High	No
ChatCam (Ours)	Medium	Low	No

2.1 Query-Focused Video Summarization

Video is one of the richest sources of sensory information, and numerous applications aspire to harness this wealth of data to aid in real-world scenarios [28, 44, 71]. Integrating the user query in video summarization has been a hot topic. Traditional approaches [36, 39, 40] are designed to address “after the fact” queries on large-scale recorded video datasets. NoScope [40] and Blazeit [39] use different models to form a multilevel filter to screen out most of the video frames from full analysis during offline analysis. Focus [36] uses a model to construct an index when storing videos and then analyses only the indexed video frames with the full model when processing subsequent tasks. These solutions are typically cloud-based and require extensive resources, including network and edge computing resources. Since all computing is done in the cloud, these approaches only require minimal resources on the edge side. However, as the volume of video data continues to grow exponentially, these systems face increasingly unsustainable bandwidth costs and significant query latency due to data transmission demands.

Previous approaches [13, 48, 71] employ a more cost-effective and specialized classification model at the edge, which is trained to discern whether an object of interest relevant to the current query appears in the current video frame. Reducto [48] dynamically adjusts its filtering decisions according to some metrics including the difference between each frame, the accuracy of the query, and so on. These edge-cloud hybrid approaches for query-focused

video summarization leverage high edge computing resources. By performing substantial processing on local edge devices, they reduce reliance on cloud servers and mitigate latency issues. Network resource consumption is reduced to a moderate level, as only processed and relevant video frames are transmitted to the cloud.

In addition, when users are unable to articulate their specific needs through a limited set of fixed textual queries at the outset of the summarization process, these query-focused video summarization systems may face difficulties in comprehension, which can lead to suboptimal performance.

2.2 Open-vocabulary detection and Multi-modal LLMs

Recent advancements have seen the emergence of open-vocabulary detection systems [27, 46, 51]. ViLD [27] distills knowledge from a pretrained image classification model into an RCNN-like detector into a two-stage detector. BLIP [46] is pretrained in two stages including vision-and-language representation learning and vision-to-language generative learning. Grounding DINO[51], the state-of-the-art open-vocabulary detection system, marries the transformer-based detector DINO with the grounded pre-training detector.

Some multi-modal large language models (MLLM, e.g., GPT-4V [5], Gemini [4], CogVLM [72], CogAgent [33]) also lead the forefront of video analysis. There are also some works [43, 54, 59, 86, 88] that use “large multimodal models (LMM)” to represent large models that process multimodal data. Several works [58, 66, 76] try to generate descriptions of video frames and then answer user-specific questions through retrieval. VidIL [76] employs the pretrained image captioner in the image-language model BLIP [46] to obtain frame captions. PDL [66] generates captions from individual video frames using a pre-trained image captioning model GIT [68]. These video summarizations are often generic and fail to consider user interests.

In addition, the substantial size of these large models imposes limitations, rendering their deployment at the edge impractical. Similar to the cloud-based approach, these approaches also consume high network resources due to the video transmission. Despite the challenge posed by the large volume of video data, the strength of these large models lies in their ability to accurately classify user-abstracted queries during the inference phase, providing a more natural interaction.

2.3 Multi-Agent Systems

Large language models (LLMs) have achieved remarkable success across a wide array of tasks. Recently, based on the development of using one LLM as a single planning or decision-making agent, LLM-based multi-agent systems [9, 30, 45, 81, 83] have achieved considerable progress in complex problem-solving. CAMEL [45] proposes a cooperative agent framework known as roleplaying, enabling agents to cooperate on complex tasks. Peer-Review [83] also proposes a multi-agent collaboration method that emulates the peer review process – “creation, review, and revision”. To evaluate LLMs’ communication and decision-making abilities in multi-agent setups, Sahar et.al. [9] create a testbed of complex multi-agent, multi-issue, and semantically rich negotiation games.

Following the mainstream multi-agent solution, ChatCam designs the “Prompt with Actor-Critic LLMs” approach. ActorLLM first generates relevant tasks and objects based on user interests. Immediately afterward, CriticLLM evaluates ActorLLM’s output and provides feedback to guide ActorLLM in producing better results. Unlike existing work that relies on simple textual interactions between multiple LLMs, ChatCam innovatively introduces MLLM to pre-evaluate the output of ActorLLM, a method validated by subsequent evaluation.

2.4 Model Deployment in Resource-constrained Environments

Common methods for deploying models on edge devices include techniques such as model compression, quantization, and weight sharing, each aiming to reduce the model’s resource requirements while maintaining performance.

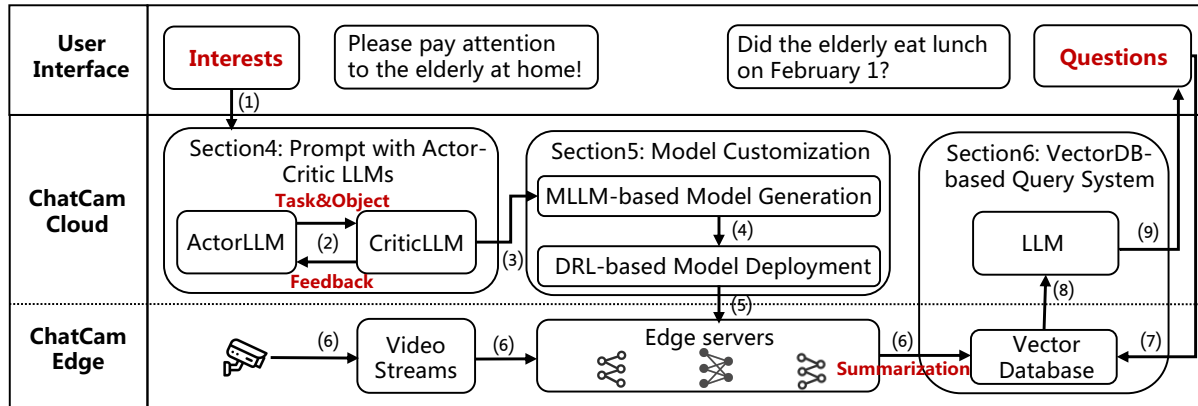


Fig. 2. Overview of ChatCam. The main components are the User interface, which receives interests and questions from users, and the ChatCam, which is distributed in the cloud and at the edge. The user initiates the process by expressing their interests to ChatCam, which then prompts the system to engage LLM and MLLM in generating relevant tasks and models. Over time, the user can pose specific questions based on these summarizations to obtain the desired information. The sequence in the figure represents the workflow of the ChatCam.

Model parameter pruning and quantization techniques [31, 60, 80] are employed to create lighter model variants that require less memory and computational resources, thereby reducing the inference time without significantly compromising model performance. Upon incorporating readily available compressed variants such as Tiny YOLO into ChatCam, weight sharing emerges as another viable solution to diminish memory overhead during the execution of multi-model inference tasks.

In recent years, several advanced techniques have emerged to mitigate the performance loss associated with weight sharing among models. Some works [53, 78] try to merge models simply by averaging their weights while some works [52] use fisher information to perform weighted averaging. Gemel [57] efficiently carries out model merging by quickly finding and retraining accuracy-preserving layer-sharing configurations. ChatCam’s implementation of model weight sharing is grounded in the principles established by this line of research. The primary objective is to employ deep reinforcement learning to derive a more efficient strategy for sharing weights among models.

3 System Overview

ChatCam embraces the LLMs for contextual chatting-to-camera with interest-oriented video summarization. Figure 2 depicts the ChatCam system’s architecture, encompassing the user interface that allows for interaction with the system, the integration of Actor-Critic LLMs to understand users’ interests, the utilization of MLLM for model generation, a DRL-based Model Deployment method to establish the optimal weight sharing policy, and a VectorDB-based query system designed to address users’ specific questions. We will start by going over some of the descriptions that come up in ChatCam, and then go over the overall workflow.

3.1 Five distinct descriptions in ChatCam

ChatCam employs a structured hierarchy comprising five principal descriptions to systematically address users’ interests. Detailed descriptions of them are listed below.

Interest. This level encapsulates the vague, overarching themes that concern the user. The focus on the situation of older relatives living at home is a good example and represents a vague thematic focus.

Task. At this stage, the user’s interest is translated into specific, actionable objectives. For instance, this could involve monitoring the elderly individual’s daily routines or detecting any occurrences of falls, thereby providing tangible objectives for the summarization system to achieve.

Object. Reflecting the principles of query-focused video summarization, the user is required to input precise keywords relevant to their interest. These objects serve as targeted directives for the system and could encompass terms such as “elderly eating”, “elderly using a smartphone” or “elderly falling” which act as focal points for extracting relevant video summarization. Once ChatCam ascertains the viability of the current task and the associated division of objectives, it proceeds to generate a specific model tailored to address and resolve the designated task.

Summarization. Video summarization is an automated technique that distills lengthy video content into a shorter version, capturing the most critical events or objects of interest. In ChatCam, video summarizations are crafted using models deployed on edge servers. These models are tasked with the efficient processing and analysis of video frames to identify and emphasize particular objects corresponding to users’ interests and tasks. For example, the summarization could be “An elderly man is eating at 11:30 on October 1, 2023”.

Question. This description pertains to the direct and specific questions posed by the user after reviewing the video summaries. An example of such a query might be, “Did the elderly eat at 11:30 on October 1, 2023?” These questions facilitate a deeper dive into the summarized content for detailed information.

3.2 Workflow of ChatCam

As shown in Figure 2, the user will initiate the process by articulating their “Interests” which are some vague, high-level textual descriptions of the content they wish to extract or monitor from a camera. Leveraging Actor-Critic LLMs, ChatCam processes these interests, dissecting them into discrete “Tasks” and their corresponding “Objects”. Each Task is specifically designed to address a particular facet of the interests. ActorLLM will first generate relevant tasks as well as objects based on the users’ interests from a semantic point of view. Immediately after that, CriticLLM evaluates the results of ActorLLM’s generation and gives some feedback accordingly. ActorLLM also updates the generated tasks as well as objects again to better meet the users’ interests after receiving feedback from CriticLLM.

After the tasks and objects have been identified, we adopt multi-modal large language models as assistants to obtain the corresponding models. Subsequently, through the DRL-based Model Deployment, an array of models could be deployed at the resource-constrained edge with guaranteed model accuracy.

Upon deployment, these models analyze incoming video frames in real time, producing summarizations of notable events or activities. These summarizations are then converted into vector embeddings, which are subsequently stored in a vector database for quick follow-up queries. When the user presents a “Question”, the system is primed to swiftly consult the vector database, extracting and presenting summarizations that precisely address the user’s questions. Immediately thereafter, the system encapsulates the user’s question along with the related summarizations into a pre-defined prompt that interacts with the LLM to get an accurate answer.

4 Prompt with Actor-Critic LLMs

To more closely align Large Language Models (LLMs) with human values, several works [19, 89] have implemented the integration of human feedback. This process fine-tunes LLMs by incorporating human judgments, effectively refining alignment with criteria such as helpfulness, truthfulness, and harmlessness. However, this fine-tuning LLM method typically demands a considerable amount of human feedback and extensive training time.

Recent studies [45, 47, 83, 87] have proposed the multi-agent method and tried to employ another LLM to evaluate the generated content. Learning from these multi-agent frameworks and classical actor-critic reinforcement learning [17, 20, 26], we try to better understand the users’ interests through the interaction between an actor

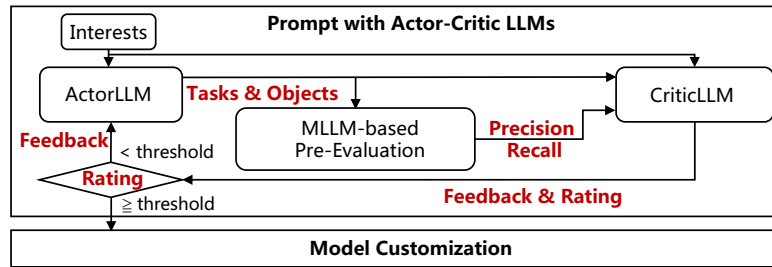


Fig. 3. Prompt with Actor-Critic LLMs. There are three main components: (1) ActorLLM translates interests to some tasks, (2) the Multi-modal Large Language Model calculates the precision and recall of tasks for pre-evaluation, (3) the CriticLLM analyzes the result and rates it.

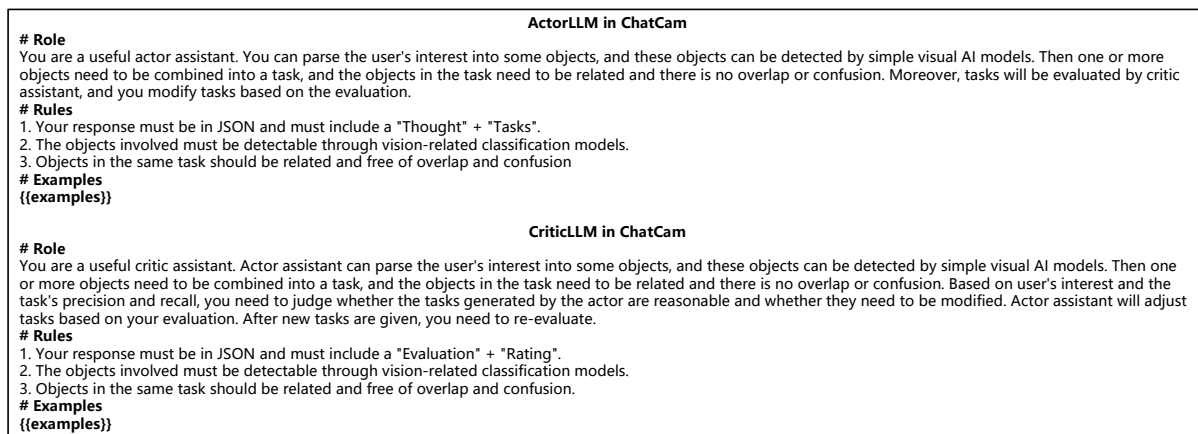


Fig. 4. Prompt template of ActorLLM and CriticLLM. The complete prompt template is given in the appendix.

(ActorLLM) as well as a critic (CriticLLM). Simultaneously, to verify the rationality of the results obtained by LLM, semantic-level analysis by another LLM is not enough. To this end, we introduce MLLM to analyze the feasibility of tasks. Figure 3 shows the overview of the Prompt with Actor-Critic LLMs method in ChatCam.

4.1 LLM as Actor

In the realm of query-focused video summarization, the specified user queries play a pivotal role in ascertaining the pertinence and quality of the resulting video summarizations. To mitigate the challenges posed by users who may supply interests that are either nebulous or lack precision, the advanced capabilities of Large Language Models (LLMs) are leveraged.

Upon receiving users' inputs that articulate their interests, an LLM acts as an *Actor* which follows the pre-prepared prompts to understand these interests. This involves distilling clear or vague interests into a delineated set of specific tasks and objects. In the pre-prepared prompts, we delineate the role that ActorLLM is required to fulfill along with the specific tasks. Concurrently, by establishing a set of rules, we facilitate the optimization of its outcomes, thereby enhancing performance. Figure 4 shows how to guide LLM to act as an actor assistant and better understand the user's interest.

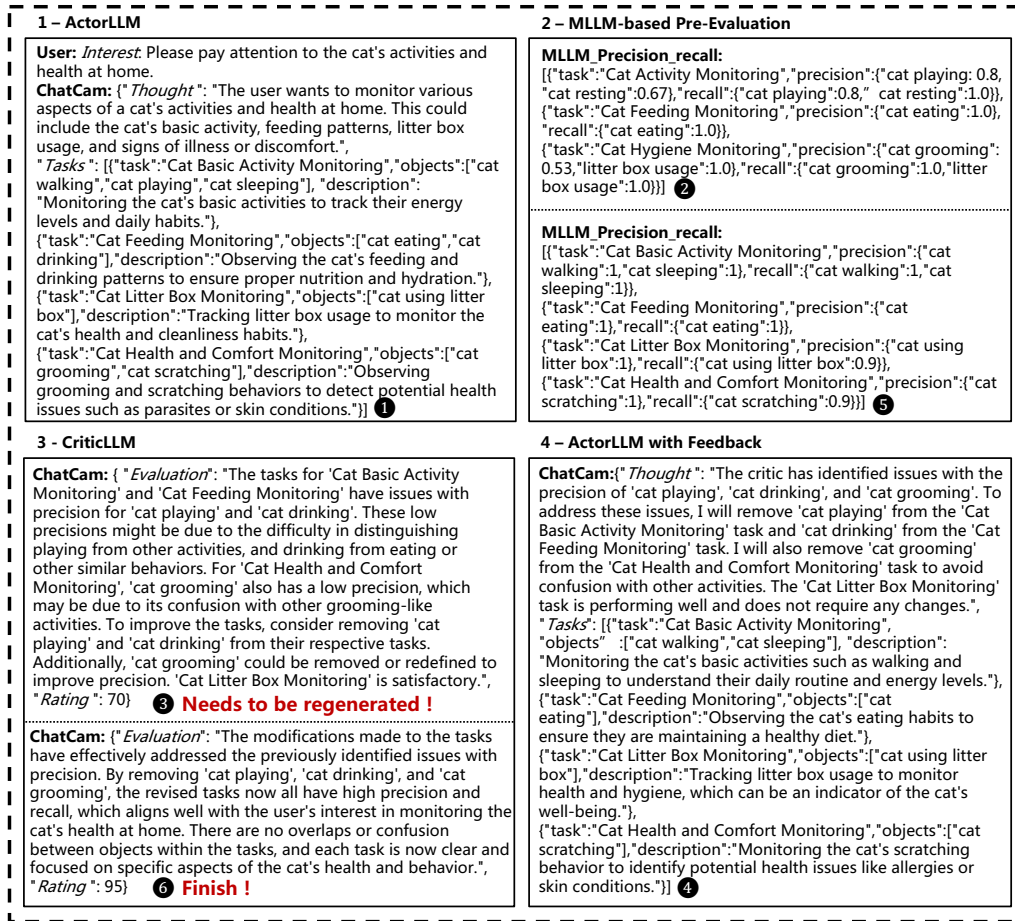


Fig. 5. An example of the interaction between ActorLLM, MMLM, and CriticLLM. The example shows that the appropriate tasks and objects are generated after two rounds of interaction based on the user's interest.

As previously indicated, the ActorLLM system is designed not only to interpret users' interests but also to dynamically refine its output in response to feedback from CriticLLM. This iterative refinement method between two LLMs is instrumental in augmenting the accuracy of the video summarization process, thereby guaranteeing that the resultant summaries are not only precise but also truly representative of the users' interests. Figure 5(1) shows how the ActorLLM understands the user's interest. ActorLLM can understand the user's interest in the cat activity at home and can translate it to some specific tasks including basic activity, feeding, litter box, and health and comfort monitoring. In addition, Figure 5(4) shows how to modify its outputs based on the feedback like removing some objects while maintaining the overall tasks.

4.2 MLLM-based Pre-Evaluation

Furthermore, our approach extends beyond the conventional direct evaluation of the textual output produced by ActorLLM, which is typical in previous research. We have innovatively incorporated an assessment of the task's feasibility using a multi-modal large language model (MLLM, e.g., GPT-4V [5], CogVLM [72], CogAgent [33]). By

introducing MLLM in a pre-evaluation role, we can simulate the performance of the lightweight models on the intended tasks before actual deployment at the edge.

To effectively evaluate the tasks and objects generated by ActorLLM, we will curate a dataset tailored to these outputs in the process of pre-evaluation. Utilizing an MLLM, we analyzed the prepared dataset to calculate the corresponding *Precision*, *Recall* and *F1* score for the tasks generated by ActorLLM. *Precision* assesses the model's accuracy in making positive predictions, contributing to user trust by ensuring relevant responses. *Recall* gauges the model's capability to identify all pertinent instances, which is crucial to avoid missing significant information in surveillance applications. *F1* score will be provided to CriticLLM below with a quantitative measure of the ActorLLM's performance, ensuring that the tasks it generates are effectively executable. If these metrics estimates are low, adjustments to task partitioning may be required. Figure 5(2) illustrates the pre-evaluation process which is conducted with the support of MLLM. It can be seen that "cat playing" and "cat resting" objects have low precision which can guide the CriticLLM to suggest changes so that ActorLLM can redetermine the task partitioning.

4.3 LLM as Critic

Some studies [45, 47, 83, 87] have proposed the multi-agent method and tried to employ another LLM to evaluate the generated content. Drawing inspiration from this concept, we propose an innovative approach that involves the use of a secondary LLM to critically assess the output of the primary LLM. This secondary LLM functions as a "critic", providing ratings and suggestions for modifications to the initial output, thereby streamlining the optimization process and reducing the need for extensive user interaction.

CriticLLM undertakes a thorough review of the tasks and objectives formulated by ActorLLM, alongside an evaluation of the metrics obtained from the MLLM. This review process involves a critical assessment of how well the tasks correspond with the user's articulated interests and the performance of the models involved. By examining these aspects, CriticLLM delivers feedback to guide ActorLLM, intending to ensure that the outputs from ActorLLM are pertinent and dependable. The cyclical exchange between the two LLMs is designed to incrementally improve the tasks generated, continuing until they achieve a standard of quality that aligns with the users' varying interests.

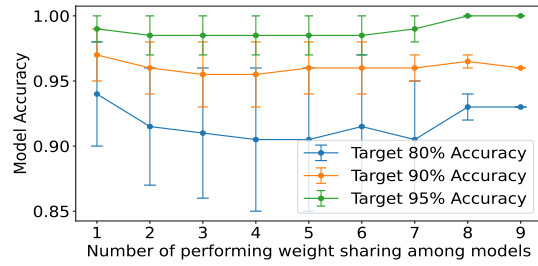
Figure 5(3) shows an example of Critic LLM. From the outcomes provided by MLLM, it is apparent that the tasks "cat playing" and "cat resting" exhibit low precision. CriticLLM, upon reviewing these metrics, will suggest potential reasons for this underperformance and propose solutions, such as considering eliminating the object requirement for detecting "cat playing".

5 Model Customization

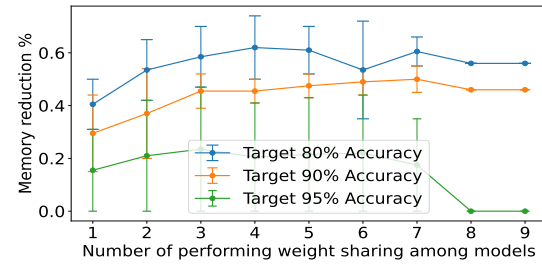
Through the prompt with actor-critic LLMs method, ChatCam can understand users' interests and generate some related tasks and objects. Since there are no models that can be generalized to solve these individualized tasks, ChatCam needs to generate these models and deploy them at the edge. Based on the two observations below, ChatCam adapts MLLM and DRL to customize these models.

5.1 Observation

5.1.1 Trade-off between model size and accuracy. We first conducted using the widely recognized object detection framework, Faster RCNN [61] to solve the tasks. Stemming from its complex network structure, Faster RCNN was adept at accurately identifying pertinent objects which achieved a mean Average Precision (mAP) of 0.467 on the coco dataset. However, this sophistication came at the cost of significant memory overhead. For instance, a common edge server Jetson Xavier NX [6] with a memory capacity of 8 GB is capable of supporting just 2 instances of the Faster RCNN model.



(a) Relationship between model accuracy and number of models participating in weight sharing



(b) Relationship between model memory reduction and number of models participating in weight sharing

Fig. 6. The performance of different weight sharing methods in terms of model accuracy and memory reduction percentage. There is a trade-off between model accuracy and memory reduction. The x-axis represents the number of models sharing weights with the initially selected model. For example, “x=2” means that, in addition to the randomly selected model, two models from the remaining models share weights with this model.

Subsequently, we directed our experimental efforts toward the utilization of popular lightweight object detection models designed for edge servers, such as Tiny YOLO [10]. Tiny YOLO makes it possible to use it in resource-limited environments by narrowing down the model size but also leads to a loss of the nuanced feature detection necessary for identifying a diverse range of objects with high precision which just achieved an mAP of 0.237 on the coco dataset.

This discrepancy in performance metrics illustrates the trade-off between the computational complexity of a model and the efficiency of a streamlined model. Based on the observation, it needs to propose a suitable method that guarantees both the accuracy of the model and the possibility of reducing the memory overhead.

5.1.2 Varying memory reduction under different settings. In addition, when the number of models is large, there still exists the problem of resource constraints at the edge, especially the memory overhead. Inspired by some model compression method [57, 65], performing weight sharing method among models can significantly reduce memory overhead. In the context of weight sharing, there is a critical need to strike a balance between maintaining model accuracy and achieving memory reduction.

The resource constraints present on the edge side influence the number of models that can be deployed, impacting the implementation of the corresponding models produced by the MLLM. Recently, the technique of weight sharing stands out as a viable strategy to curtail the computational burden imposed on memory resources. As can be seen in Figure 6, as the number of merged models increases, we find that the amount of memory that can be reduced changes. Additionally, the memory reduction achieved through weight sharing is influenced by the desired threshold of accuracy for the final task. In scenarios where high accuracy is imperative, the potential for memory savings via weight sharing becomes constrained. The retention of a substantial number of model parameters is often necessary to preserve the level of precision demanded by such tasks. Conversely, when the accuracy threshold is more lenient, there is a greater opportunity to compress the model through weight sharing without significantly impacting performance.

This relationship indicates a trade-off between the accuracy of a model and the memory efficiency gains from weight sharing. Based on the above experiments, it requires an optimal approach to decision-making, with a smart trade-off between efficiency and accuracy.

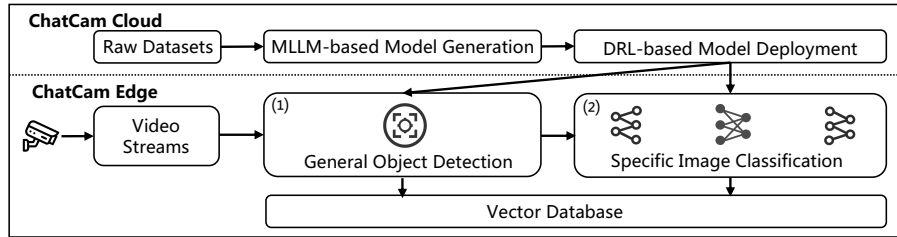


Fig. 7. Model customization. After ActorLLM generates the required tasks, ChatCam first uses MLLM to generate corresponding models, then uses DRL to implement appropriate weight sharing strategies on these models, and then deploys them to the edge for video analysis.

5.2 MLLM-based Model Generation

To maintain high model accuracy under memory constraints, we generate lightweight models with the help of multi-modal large language models.

Like existing work [16, 71], ChatCam initially deploys a lightweight object detection model to identify more general categories, followed by a lightweight image classification model that classifies the specifics of the detected objects. Figure 7 shows our MLLM-based model generation approach. For example, our approach modifies this process by just utilizing lightweight object detection models to detect broader categories specifically (e.g., people and vehicles). Subsequently, we will apply a compact yet efficient image classification model (e.g., MobileNetV3 [35]), to further classify the type of objects detected (e.g., cars, ambulances, fire trucks).

When the specific tasks generated by ActorLLM do not correspond to existing datasets with pre-labeled data, the training process for the model becomes inherently complex. This complexity is further amplified in tasks that require precise annotations, such as the localization of objects within images. In scenarios lacking annotated datasets, multi-modal large language models offer an efficient solution. Similar to knowledge distillation [18, 25, 70], it allows a compact “student” model to learn from a larger “teacher” model that has been pre-trained on extensive datasets. The student model absorbs the teacher’s knowledge, gaining advanced capabilities without the need for extensive labeled data, thus bypassing the challenge of dataset annotation scarcity. Therefore, we would like to leverage the powerful image recognition capabilities of MLLM to generate some customized models.

5.3 DRL-based Model Deployment

To address the challenge of balancing model accuracy with memory efficiency through weight sharing, employing deep reinforcement learning (DRL) is a rational approach to solve the combinatorial optimization problem [11, 14, 41]. DRL is well-suited for this task as it involves an agent who can learn optimal policies through trial and error, receiving feedback in the form of rewards based on the performance of the actions it takes.

5.3.1 Overview of D3QN-based model deployment. For the reinforcement learning component, we opted for Dueling-Double-DQN (D3QN). D3QN amalgamates the benefits of Double DQN [67], which mitigates the overestimation of Q-values often seen in standard DQN, with the enhanced action space representation provided by Dueling DQN. This combination allows for more accurate value estimation and effective decision-making across a broader range of possible actions.

5.3.2 Design of D3QN. We summarize the state space, action space, done, and reward function that are used in our D3QN-based model deployment framework. Table 2 lists the relevant notations and their corresponding descriptions.

Table 2. Summary of notations

Notation	Description
\mathcal{M}	Set of models
\mathcal{P}	Accuracy of model
\mathcal{V}	Memory of model
V_{max}	Constrained memory
$x_{i,j}$	Action refers to how to handle two models m_i and m_j
\mathcal{S}	State space
\mathcal{A}	Action Space
r	Reward

State Space: The state space is primarily utilized to describe the environmental information including all models, their accuracy, and memory usage. Indeed, in the context of employing reinforcement learning for model weight sharing, it is imperative to factor in the internal dependencies and interactions of the model to enhance the robustness of the reinforcement learning algorithm. In our experiments, we found that merging is mainly merged in several layers including classifier and FC layer weights (e.g., classifier.0.weight, features.14.block.2.fc2.weight). Therefore, we introduce a new model similarity metric “multi-layer similarity”, which is an intermediate state between “model similarity” and “layer similarity”. To estimate the similarity, we use SSIM [34] to calculate the “multi-layer similarity” similarity of the two models.

We employ the notation \mathcal{S} to represent the state space, where each $s \in \mathcal{S}$ is denoted as a five-tuple $[\mathcal{M}, \mathcal{P}, \mathcal{V}, \mathcal{L}, \mathcal{K}]$. Here, $m \in \mathcal{M}$ represents a model that will be deployed at the edge for solving some specific task, $p \in \mathcal{P}$ represents the accuracy of each model, $v \in \mathcal{V}$ represents the memory required by each model for inference, \mathcal{L} represents the status of weight sharing among models, and \mathcal{K} represents the average similarity of several layers of each model.

Action Space: The action space pertains to the actions performed by the policy module for weight sharing among models. Let \mathcal{A} represent the action space and each $a = x_{i,j} \in \mathcal{A}$ is a variable that indicates performing model sharing between model m_i and m_j with different target accuracy percentage.

$$x_{i,j} = \begin{cases} 0, & \text{Do nothing;} \\ 1, & \text{Merging } m_i \text{ and } m_j \text{ with a target accuracy percentage of 80;} \\ 2, & \text{Merging } m_i \text{ and } m_j \text{ with a target accuracy percentage of 90;} \\ 3, & \text{Merging } m_i \text{ and } m_j \text{ with a target accuracy percentage of 95.} \end{cases} \quad (1)$$

When a decision necessitates the involvement of multiple models, and some of those models have previously engaged in weight sharing, it becomes imperative for all the involved models to partake in the weight sharing process. This ensures consistency and maintains the integrity of the decision-making process.

During the reinforcement learning training phase, we have integrated a masking mechanism. This mechanism serves to determine the model from taking repetitive actions, thereby fostering exploratory behavior that can enrich the policy with diversity and effectiveness. By embedding this masking mechanism into the agent’s decision-making function *choose action*, we enable the agent to select actions more judiciously. It functions by screening out actions that are potentially less beneficial or superfluous, steering the agent towards decisions that are more conducive to achieving the best possible results.

Done: Once the resource constraint is satisfied by sharing the weights of several models, we consider that a termination state has been reached once the required memory satisfies the resource constraint.

$$\sum_{i=0}^{|\mathcal{P}|-1} v_i \leq V_{max} \quad (2)$$

Reward Function: In real-world implementations, deploying models often requires a balance between computational performance and resource utilization. Generally, the DRL agent aims to maximize the reward in the training process. Therefore, the reward function we employ incorporates two key parameters: the accuracy of the model and the resource demand scenario where the model's requirements exceed the available resources.

Furthermore, the integration of a weighted parameter within the reward function affords us the flexibility to calibrate the reinforcement learning algorithm for varying objectives. Adjusting this weighted parameter allows for the modulation of the algorithm's focus, balancing resource utilization against model accuracy.

In the context of deploying models, it is essential to guarantee the minimum level of accuracy that the integrated model must meet. The minimum model accuracy ensures that the model's performance is satisfactory for practical applications in various scenarios. Therefore, to guide the model deployment process toward this objective, we introduce a reward function as below:

$$r_{min} = \min_{i=0}^{|\mathcal{P}|-1} p_i - \alpha \cdot \max \left\{ 0, \sum_{i=0}^{|\mathcal{V}|-1} v_i - V_{max} \right\}. \quad (3)$$

In addition, we also set the reward to balance the average accuracy of the model which is measured as follows:

$$r_{avg} = \frac{\sum_{i=0}^{|\mathcal{P}|-1} p_i}{|\mathcal{M}|} - \alpha \cdot \max \left\{ 0, \sum_{i=0}^{|\mathcal{V}|-1} v_i - V_{max} \right\}. \quad (4)$$

5.3.3 Interaction between agent and environment. Within the realm of our DRL-based model deployment, the agent is tasked with selecting an action from the action space \mathcal{A} based on its current perception of the environment's state. Each action $a = x_{i,j}$ is a discrete decision that determines the extent of weight sharing between models m_i and m_j , aimed at a specific accuracy target. Upon the agent's action execution, the environment responds by effectuating the weight sharing between the chosen models, which inherently alters the state space \mathcal{S} . This alteration reflects the updated model accuracy $p \in \mathcal{P}$, memory usage $v \in \mathcal{V}$, the new status of weight sharing \mathcal{L} , and potentially the multi-layer similarity \mathcal{K} due to the shared weights. The state space is dynamically updated to encapsulate these changes, providing the agent with the latest information to inform subsequent actions and strategize future interactions with the environment.

Upon completion of the training phase for the reinforcement learning network, it is capable of handling several models derived from the MLLM mentioned earlier. By executing an appropriate weight sharing strategy among these models, the system can be adapted to comply with the resource constraints inherent to edge computing environments. Subsequently, models that share layers, as determined by the reinforcement learning policy, are deployed to the edge servers, where they carry out the analysis of video streams.

6 Implementation

ChatCam is a contextual chatting-to-camera system that is based on interest-oriented video summarization, i.e., some questions can be answered from the summarizations since the system started recording video.

Therefore, in this section, we will present the implementation of the video summarization system and a vectorDB-based query system. Firstly, we first introduce some details about the interest-oriented video summarization. Subsequently, we delve into the vectorDB-based query system.

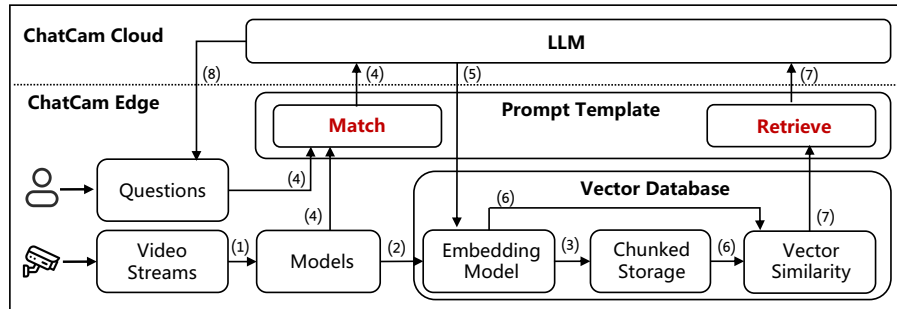


Fig. 8. VectorDB-based query system. ChatCam employs LLM to match the user’s questions with appropriate models and then uses vector database similarity to extract relevant video summaries. These summaries are processed by LLM to deliver the final answer to the user.

6.1 Interest-oriented video summarization

In the prompt with actor-critic LLMs framework, the system’s ability to precisely interpret user interests has been enhanced through the implementation of *GPT-4-1106* in the dual capacities of ActorLLM and CriticLLM. Referring to some previous works [74, 75, 84], we set the temperature parameter to 0.7 and the maximum token to 1024. We chose GPT-4 for both the actor and critic components because it is currently regarded as the state-of-the-art (SOTA) in large language models. Alongside the utilization of CriticLLM to evaluate tasks generated by ActorLLM concerning user interests, a multi-modal large language model is used to determine the feasibility of the tasks produced by ActorLLM. Given our resource and cost constraints, it was essential to choose a MLLM that balances performance with computational feasibility. We selected CogAgent [33] as our MLLM due to its high accuracy and ability to run efficiently on a single NVIDIA 4090 GPU. CogAgent meets these criteria, providing excellent accuracy while being resource-efficient, making it a practical choice for our research.

Through the prompt with Actor-Critic LLMs method, ChatCam can understand users’ interests and generate some related tasks and objects. As described in Section 5, ChatCam utilizes MLLM to generate some lightweight models deployed at the edge for object detection and image classification tasks. GPT-4V, while proficient in various tasks, exhibits limitations in spatial comprehension. Therefore, ChatCam utilizes Tiny YOLO [10] as the student object detection model while GroundingDINO-B [51] as the teacher model. For the classification aspect, we adopt MobileNetV3-Large [35] as the student model while CogAgent is responsible for classifying the state or behavior of objects.

As for the MLLM-based model generation, upon receiving the requests (e.g., classify the vehicle type), the MLLM sets out to collect unlabeled data from the internet, leveraging the abundant information available online. Utilizing its sophisticated classification capabilities, the MLLM annotates this data, effectively converting unstructured information into a structured, labeled dataset. These labeled datasets then serve as critical assets for training the classification models deployed at the edge, providing them with quality, pre-labeled data. In addition, appropriate weight sharing is performed on these models with the help of DRL decision-making and then deployed to the edge. These models continuously analyze incoming video streams and derive corresponding video summarizations.

6.2 VectorDB-based query system

Figure 8 shows the workflow of the VectorDB-based summarization query system. Similar to some typical real-time video analytics pipelines [37, 71], we use an edge server that runs some long-running models customized by the DRL-based weight deployment method. These models will process the incoming video frames and then generate some summarizations to answer the user’s query questions.

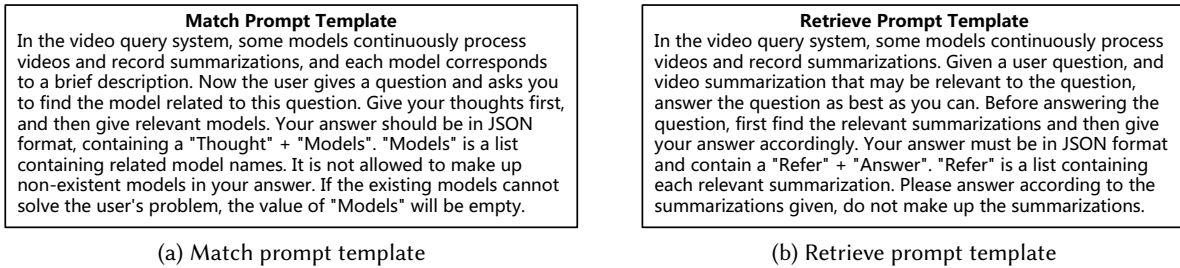


Fig. 9. Prompt Template of LLM-based query system.

6.2.1 Vector database. Vector databases (e.g., Qdrant [7], Milvus [29, 69], Chroma [2]) are specialized repositories designed for the storage and retrieval of vector embeddings. These databases are integral for preserving embeddings along with associated metadata, which can subsequently be utilized by large language models. This functionality is leveraged by employing the embedding function offered by VectorDB to execute the embedding of video summarizations. Since we need to store some video summarizations at the edge, we use a lightweight vector database, Chroma [2]. We chose the Sentence Transformers model “all-MiniLM-L6-v2” [73], which is trained on more than 1 billion data points and can create sentence and document embeddings for a wide variety of tasks. We have performed experiments with different queries in Section 7.4.2, and the results show that “all-MiniLM-L6-v2” can achieve the similar retrieval results as “all-mpnet-base-v2” in a shorter time.

6.2.2 Chunked summarization storage. To enhance the efficiency and accuracy of summarization queries, the ChatCam system has adopted a chunked storage architecture, organizing the outputs from various models into distinct chunks. This method plays a pivotal role in improving data retrieval speeds and minimizing the search space necessary for addressing queries [3, 8, 22, 24].

6.2.3 LLM-based query system. During the query summarization procedure employed by ChatCam, an initial interpretation of the user’s question is conducted by an LLM. Figure 9(a) shows the prompt template for matching user’s questions and models. The LLM fulfills a critical function by mapping the user’s raw question to the specialized capabilities of the diverse models integrated into the system. After pinpointing the models that are pertinent to the query, the system employs a pre-trained embedding model which will convert the user’s question into a vector embedding.

The embedding obtained from the user’s questions is subsequently utilized to conduct a semantic similarity search within a filtered repository of video summarization vectors. In our system, we have chosen the cosine as the metric for assessing similarity. By calculating the cosine distances, the system can effectively gauge the level of relevance between the user’s questions and the available video summarizations, enabling the identification of the most pertinent content in response to the user’s query questions.

Following summarization retrieval, the extracted information is integrated with the original user query to form an enriched contextual prompt as shown in Figure 9(b). This enriched prompt is then fed back into the LLM, which synthesizes and articulates the final response answers.

7 Evaluation

In this section, we will first introduce the setup, including the datasets, weight sharing settings, baselines, and evaluation metrics. Then we will evaluate the performance of ChatCam through a comparative study in terms of multiple metrics. Finally, to show the effectiveness of ChatCam, we report a case study.

Table 3. Example of the customized dataset that contains a series of real users' interests.

Domain	Clear interests	Vague interests
Smart Home	Please pay attention to whether the elderly at home has fallen.	Please pay attention to the elderly.
	Please pay attention to the child's eating and sleeping patterns.	Please pay attention to the child.
	Please pay attention to whether there is anyone at the door.	Please pay attention to the safety situation at home.
Traffic Surveillance	Please pay attention to the vehicle type on the road.	Please pay attention to the traffic condition.
	Please pay attention to whether there are cracks on the road surface.	Please pay attention to the behavior of drivers.
	Please pay attention to whether there is a car accident.	Please pay attention to the presence of any road hazards.
Mall Surveillance	Please pay attention to whether there is any violence in the store.	Please pay attention to the customers in the store.
	Please pay attention to any signs of fire or smoke.	Please pay attention to the customers at the store entrance.
	Please pay attention to any customer disputes or fights.	Please pay attention to customer shopping behaviors.

7.1 Setup

7.1.1 Datasets. To evaluate the performance of the prompt with Actor-Critic LLMs, we developed a customized dataset that designed to simulate real-world scenarios. In our dataset, we categorize user interests into two distinct types: clear interests, such as monitoring for instances where the elderly may have fallen, and vague interests, like keeping an eye on pets.

Our customized dataset mainly involves three scenarios: smart home, traffic surveillance, and mall surveillance. Table 3 shows some example interests in our customized dataset. There are 10 interests in each scenario with 5 being clear and 5 being vague. The dataset was collected through an online survey distributed via various social media platforms, where participants were asked to list their interests related to three scenarios in text form. We have also collected several video segments across various location (e.g., crossroads, supermarket shelves, escalators), with an average duration of approximately 17.66 minutes. We evaluated ChatCam's ability to understand users by assessing real user interests, and evaluated its vectorDB-based query system using the collected real-world videos.

As for our DRL-based model deployment method, we employed the CIFAR-10 [42] dataset, which is comprised of ten discrete object classes, including a range of animals. To augment this dataset, we incorporated additional categories from CIFAR-100 [42], specifically "sea", "house", and "road" to act as background classes. This expansion of the dataset yielded a collection of ten models, which were then leveraged to execute and evaluate various model weight sharing strategies.

7.1.2 Weight sharing settings. During the process of determining how to share weights among models, if the number of classifications between two models is not aligned or if the accuracy falls below a certain threshold, retraining becomes necessary which introduces a significant time overhead. Therefore, in our experimental setup, we have chosen to only consider merging models that have an identical number of classifications. This can be achieved by subdividing the background classes to maintain consistency in the number of classifications across all models.

We impose a constraint to prevent models from undergoing re-training during the merging process, thereby avoiding the substantial time costs associated with retraining sessions. This also explains why we used the CIFAR-10, and CIFAR-100 datasets to construct 10 binary classification models when evaluating our DRL-based model deployment. In addition, we set three different target percentage thresholds of model accuracy, 80%, 90%, and 95%. Different thresholds result in different numbers of layers being used for weight sharing.

7.1.3 Baselines and metrics. To evaluate the performance of the prompt with Actor-Critic LLMs, we compare ChatCam with three types of baselines.

- Zero-shot: General LLM analyzes users' interests directly and generates corresponding tasks and objects.

- ActorLLM: Based on the chain of thought (CoT) [77] method, ActorLLM directly analyzes users' interests with some examples and generates corresponding tasks and objects.
- ChatCamLite: Ignore the assessment of model precision and recall. Use this algorithm to evaluate the effectiveness of our proposed MMLM pre-evaluation.

Utilizing our tailored dataset, ChatCam's comprehension of user interests is assessed. Initially, the *ratings* provided by CriticLLM are employed to assess ChatCam's interpretive capacity regarding interests. This evaluation reflects the task's completeness, namely, the system's ability to respond to the user's detailed questions.

In addition, we also define a new metric called interest satisfaction ratio (*ISR*), which is designed to evaluate the effectiveness of a system in generating tasks. *ISR* leverages the F1 score and only objects with an F1 score exceeding the threshold of 0.8 are considered. By summing the F1 scores above this threshold, *ISR* quantifies the system's performance in creating high-quality tasks. This sum is then divided by the number of user interests to normalize the result, providing an average satisfaction level per interest. The *ISR* can be represented as follows:

$$ISR = \frac{\sum_{i=1}^n F1_i \cdot \mathbb{I}(F1_i > 0.8)}{N} \quad (5)$$

where $F1_i$ is the F1 score of the i -th object, $\mathbb{I}(F1_i > 0.8)$ is an indicator function that equals 1 when the F1 score is greater than 0.8 and 0 otherwise, n is the total number of objects in the task, and N is the number of user interests.

Similarly, to evaluate the performance of DRL-based model deployment, we also compare ChatCam with the following three algorithms.

- Random: Randomly perform weight sharing between two models and ensure that the memory constraint is eventually met.
- Greedy: Memory availability is established as a primary condition to guarantee that the model can be deployed in its entirety. We repeatedly select the two models with the highest model accuracy for weight sharing, so that the minimum accuracy and average accuracy can be guaranteed.
- DQN: DQN [55] is a classic policy-based deep reinforcement learning algorithm, and D3QN is an improvement based on it.

We compare the results of ChatCam with the three baselines above in terms of both minimum as well as average accuracy metrics under different memory constraints. In addition, we also verified the effect of different reward function settings on the results.

We configured the DRL training to consist of 500 epochs. The discount factor, *gamma*, is set to 0.99, allowing us to quantify the present worth of future rewards. The parameter *tau* is set to 0.005 and is used to manage the speed of the soft update for the target network's parameters. The exploration strategy commences with *epsilon* at an initial value of 1 while the decrement rate is set to "5e-7". We use the mean of the last two epochs to converge to reflect the performance of the algorithm.

7.2 Performance of Prompt with Actor-Critic LLMs

In this section, we evaluate the performance of the prompt with actor-critic LLMs. Initially, we assessed the overall performance by examining ratings and *ISR*. Subsequently, we observed the trends of these metrics throughout the interaction between ActorLLM and CriticLLM.

7.2.1 Rating and ISR. The performance of the algorithms was systematically evaluated based on two principal metrics: the qualitative *rating* obtained from the unified CriticLLM which evaluates effectiveness in generating tasks that align with user interests and the *ISR*, which evaluates the actual performance in solving problems on these tasks.

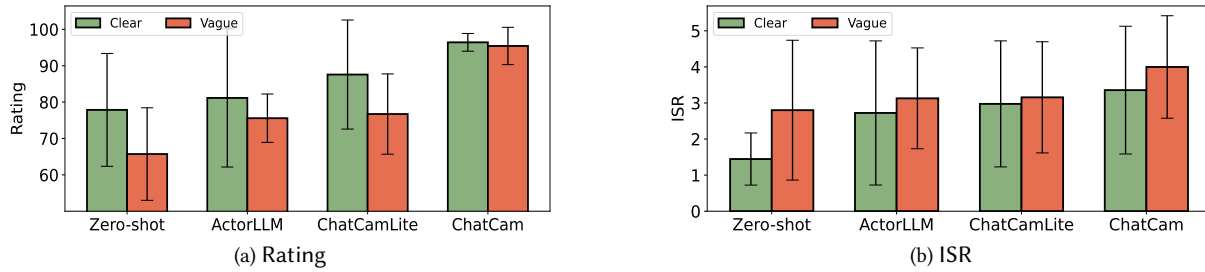


Fig. 10. Performance of different algorithms on *Rating* and *ISR*. *Rating* evaluates the effectiveness in generating tasks that align with user interests and the *ISR* evaluates the actual performance in solving problems on these tasks.

We commenced by scrutinizing the average *rating* performance of each algorithm using a clear interest dataset. Figure 10(a) depicts that the zero-shot algorithm trails with a *rating* of 78. In contrast, the ActorLLM, which adopts the CoT methodology, fosters example-based learning, prompting the LLM to tackle problems sequentially and unveil the steps of intermediate reasoning. This strategy not only brings the LLM's operation into closer resemblance with human cognitive functions but also manifests in an augmented *rating* of 81. Moreover, the synergy of CriticLLM's evaluative proficiency with ActorLLM's problem-solving strategy has culminated in a marked elevation in output quality, as reflected by an impressive *rating* of 88. The introduction of the MLLM as a preliminary assessment tool in the dialogue interplay between ActorLLM and CriticLLM has noticeably boosted the system's performance metrics, with a 23.1% enhancement over the baseline and a superior *rating* of 96. Corresponding experiments on datasets typified by ambiguous interests yielded similar trends, with ChatCam outshining its rivals and amplifying the *rating* by 43.9% when compared to the baseline.

The *ISR* performance of each algorithm reflects trends similar to those observed in the rating evaluations. Detailed analysis was conducted to assess the average *ISR* performance of each algorithm against a dataset with clearly defined parameters. The zero-shot algorithm lagged behind, registering an *ISR* of 1.45. The ActorLLM, in comparison, demonstrated progress with an *ISR* of 2.72. Building upon this, ChatCamLite, which integrates the evaluative prowess of CriticLLM, accomplished an even higher *ISR* of 2.97. Notably, ChatCam achieved the apex *ISR* score of 3.36, signifying a 132% enhancement relative to the baseline figure.

When examining the performance on datasets characterized by vague user interests, a notable increase in *ISR* was observed for each algorithm compared to their performance on the clear dataset. This is largely attributable to the nature of vague interests, which inherently span a wider range of task-related aspects, thereby contributing to an elevated *ISR*. Specifically, ChatCam recorded a substantial 42.9% improvement in *ISR* when benchmarked against the baseline.

7.2.2 Rating and ISR across rounds. Our evaluation aims to scrutinize the gradual enhancement of *ratings* and *ISR* achieved through successive rounds of dialogue between the LLMs. Figure 11 demonstrates that the algorithm's performance is not fixed but progressively enhances with each subsequent dialogue round between the respective models.

As illustrated in Figure 11(a), it becomes apparent that the rounds of interaction exert a considerable influence on the *rating* performance. As the number of interaction rounds increases, ChatCam evaluated with the help of MLLM can significantly improve the rating which achieves a 15.7%-39.7% improvement over round 1. As for ChatCamLite, since it only uses semantics to adjust the output, its *rating* can be improved on clear datasets while remaining almost unchanged on vague datasets.

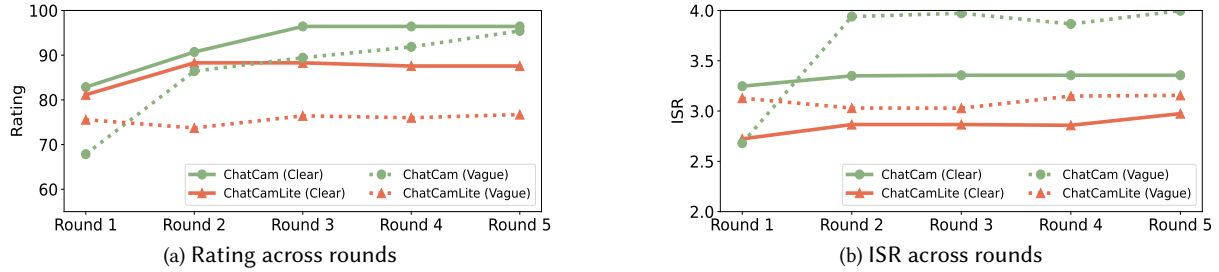


Fig. 11. *Rating* and *ISR* vary with the number of interaction rounds. As the number of interaction rounds between ActorLLM and CriticLLM increases, the MLLM-assisted ChatCam can improve *rating* as well as *ISR*.

Parallel experiments focused on the *ISR* metric were conducted. Figure 11(b) clearly illustrates that the number of interaction rounds significantly influences the *ISR* performance, specifically for ChatCam’s vague dataset. This effect is attributed to ChatCam’s ability to dynamically refine the quantity and segmentation of tasks in response to feedback, resulting in multiple tasks exhibiting elevated F1 scores, thereby enhancing the overall *ISR*.

7.2.3 Iterations for the different interests. To illustrate the impact and feasibility of MLLM on iterative processes, we present detailed information on different iterations tailored to various user interests. As shown in Table 4, for the clear interests (e.g., monitoring elderly falls and observing children’s eating and sleeping patterns), ChatCam can achieve convergence in just one iteration. In these cases, the tasks generated align perfectly with the user’s initial interests, demonstrating the system’s efficiency and effectiveness.

When dealing with more ambiguous user interests, ChatCam requires multiple rounds of iterations. During these iterations, the system adjusts the division of tasks based on the feedback from MLLM. By evaluating the model’s feasibility through MLLM, tasks that exceed the model’s capacity can be identified and either removed or modified accordingly. These necessary adjustments are then communicated back to ActorLLM, which in turn re-plans the task division to better align with the model’s capabilities and user requirements. For example, ActorLLM retains “smoke” but removes “fire” due to its low precision for “Smoke and Fire Detection” task.

By continuously refining tasks based on MLLM feedback, ChatCam effectively adapts to the user’s interests, ensuring higher precision and recall in object detection tasks. Table 9 in the Appendix B.2 shows the complete results.

Table 4. Information on the different iterations for some interests.

Interest	Iteration	Task and Objects	Precision	Recall	Rating
Elderly fallen	1	Elderly Fall Detection: “elderly falling”	1	1	100
Child eating and sleeping patterns	1	Child Eating Monitoring: “child eating”	1	1	100
		Child Sleeping Monitoring: “child sleeping”	1	1	
Home safety situation	1	Smoke and Fire Detection: “smoke”, “fire”	0.71, 0.56	1, 1	65
		Intrusion Detection: “door forced open”, “window broken”	0.5, 0.59	0.9, 1	
		Occupant Safety Monitoring: “person falling”, “person lying still”	0.71, 0.88	1, 0.7	
		Smoke Detection: “smoke”	1	1	
Home safety situation	2	Intrusion Detection: “unknown person”, “window broken”	1, 1	0.7, 1	85
		Occupant Safety Monitoring: “person falling”	1	1	
		Smoke Detection: “smoke”	1	1	
Home safety situation	3	Intrusion Detection: “window broken”	1	1	100
		Occupant Safety Monitoring: “person falling”	1	1	

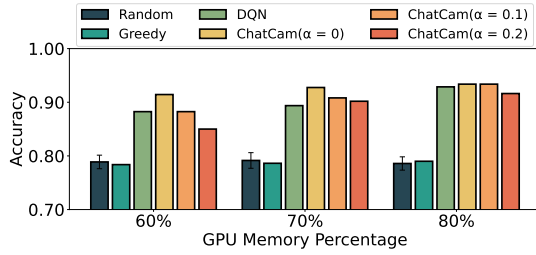


Fig. 12. Minimum Model Accuracy. We repeated the random algorithm 10 times, while reinforcement learning uses the weights file saved by training 500 epochs for inference. It can be seen that ChatCam can outperform other algorithms on minimum accuracy and different weighting parameter settings can affect its performance.

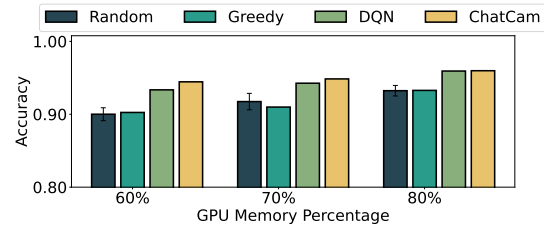


Fig. 13. Average Model Accuracy. We repeated the random algorithm 10 times, while reinforcement learning uses the weights file saved by training 500 epochs for inference. It can be seen that ChatCam can still outperform other algorithms on average accuracy.

7.3 Performance of DRL-based Model Deployment

In this section, we report the performance of the DRL-based model deployment framework against two baselines using two different evaluation metrics.

7.3.1 Minimum Model Accuracy. Figure 12 provides a comprehensive performance analysis, juxtaposing the efficacy of various algorithms with a focus on the pivotal metric of minimum accuracy achieved. In this key performance indicator, our proposed algorithm, characterized by strategic weight sharing, exhibits superior performance, yielding the highest minimum accuracy in comparison to its counterparts. Both deep reinforcement learning algorithms performed better than the heuristic algorithms on the minimum model accuracy. In particular, ChatCam markedly enhances the minimum accuracy by 11.9%-21.1% in different settings, relative to the heuristic algorithms. Compared with DQN, ChatCam can improve the minimum accuracy by 0.54%-3.78%.

In our experimental setup, we three different GPU memory usage percentages (60%, 70%, and 80%) compared to deploying all models directly. The empirical data corroborates the hypothesis that alleviating memory limitations is directly proportional to an enhancement in model accuracy. This trend aligns seamlessly with our theoretical projections, positing that the availability of increased memory capacity permits the execution of a larger ensemble of models or the maintenance of a more extensive set of parameters, both of which are instrumental in boosting model precision.

Moreover, it is evident from the results that the relaxation of memory constraints enables our algorithm, ChatCam, to surpass the performance improvements achievable by a heuristic algorithm. Specifically, when the GPU memory occupancy is configured to 60%, ChatCam achieves a minimum accuracy boost ranging from 11.9% to 18.8%. This enhancement extends to a range of 13.5%-20.3% when the occupancy is increased to 70%, and it further escalates to a span of 15.5%-21.1% as the occupancy reaches 80%. These findings underscore the efficiency of ChatCam in optimizing performance under varying memory availability.

Our evaluation extends to the investigation of the influence that varying weighted penalty parameters within the reward function exert on the algorithm's performance. It is evident from Figure 12 that these parameters are pivotal in determining the final efficacy of the model. By setting different parameters, ChatCam achieves different performances after the same epochs for training.

7.3.2 Average Model Accuracy. In addition to minimum accuracy, we also evaluated our algorithm's performance on average accuracy, where the reward function was formulated to resemble Equation 4. This reward function was specifically designed to capture the overall effectiveness of the model across a range of inputs, rather than

Table 5. Comparative study for object detection task in traffic surveillance scenario.

Method	Weight	Inference Time	F1 Score			
			Car	Truck	Bus	Person
BLIP-base	944M	149.3ms	92.86%	18.49%	28.45%	19.92%
BLIP-large	1.8G	209.3ms	92.92%	28.45%	44.56%	15.05%
ChatCam	34M	2.9ms	99.71%	98.03%	94.56%	94.01%

just the worst-case scenarios. Subsequent experiments replicated the assessment of each algorithm against this metric of average model accuracy.

Both deep reinforcement learning algorithms performed better than the heuristic algorithms on the average model accuracy. Our approach still achieves an improvement in optimizing the average accuracy of the model when compared to strategies that aim to balance the average accuracy of the model. ChatCam can improve the average accuracy of 0.9%-6.2% compared to the two heuristic algorithms. Compared with DQN, ChatCam can improve the average accuracy by 0.05%-1.18%.

7.4 Case Study

In our case study, we implemented our vector-based query system within a realistic setting, concentrating on its configuration and the outcomes of its principal operations. We first compared ChatCam with existing methods on the object detection task in traffic surveillance scenarios. Then, we used a detailed example to show the workflow of the Vector-DB-based query system and measure the embedding model. Next, we added more examples to demonstrate ChatCam's performance across multiple scenarios. Finally, we conducted a real user study to validate the practicality of ChatCam.

7.4.1 Comparative study. There are several methods that apply large language models (LLMs) to video summarization. PDL [66] generates captions from individual video frames using a pre-trained image captioning model GIT [68]. VidIL [76] employs the pretrained image caption model in the image-language model BLIP [46] to obtain frame captions and use GPT to answer questions as well.

To evaluate the performance of ChatCam in practice, we used BLIP (SOTA for image caption) with different weight sizes and employed GPT-4 to analyze the summarizations generated. The following are the two baselines and ChatCam.

- BLIP-base: BLIP with Vision Transformer (ViT) base backbone for image caption.
- BLIP-large: BLIP with ViT-large backbone for image caption.
- ChatCam: Tiny-YOLO model for each interest to detect user interests.

Taking traffic scene as an example, we selected two interests from our customized dataset as follows: "Please pay attention to the vehicle type on the road" and "Please pay attention to whether there are pedestrians on the road". We chose the BDD100K [85] dataset for evaluation, selecting validation video clips relevant to the interests. For the evaluation, we measured the weight size and time cost of three different methods used to generate the summarizations. Then, we evaluated the F1 score of each object based on the video summarizations.

Table 5 shows that ChatCam excels in performing user-required detection tasks more efficiently by utilizing lighter models. Specifically, ChatCam is capable of processing a frame in just 2.9ms, whereas the BLIP model requires over 100ms. While larger models can enhance accuracy, they also come with increased computational time usage. For instance, BLIP-large takes 209.3 ms to generate captions, which is 1.4 times that of BLIP-base.

Results have shown that ChatCam can complete the detection tasks required by users in a shorter time by deploying lighter models. As for BLIP, the larger model can contribute to the F1 score of three objects ("car",

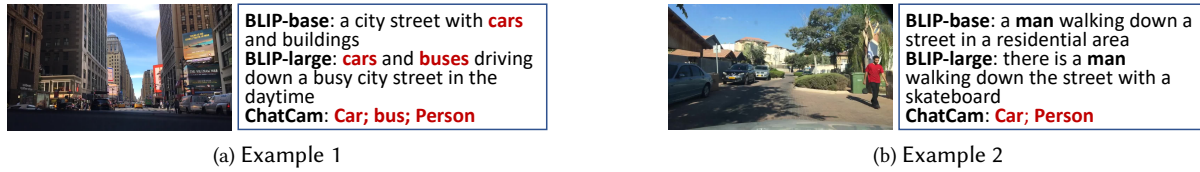


Fig. 14. Visualization of comparative study results. The general caption model typically focuses on the main parts of the image while ChatCam can focus on users’ different interests. The example figures come from the BDD100K [85] dataset.

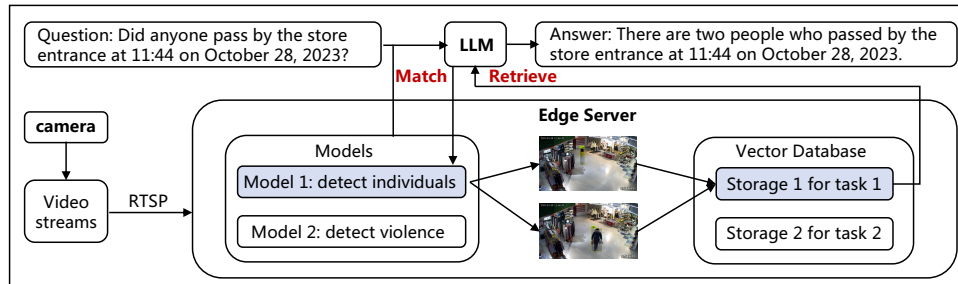


Fig. 15. An example in VectorDB-based query system. The user’s interest is to pay attention to the pedestrians outside the store, and the specific question is to check whether pedestrians are passing by at a specific time. ChatCam first matches to model 1 and then retrieves video summarizations from the storage corresponding to model 1.

“truck” and “bus”). The general caption model typically focuses on the main parts of the image, such as the car in Figure 14(a) and the person in Figure 14(b). However, it often fails to consider the user’s specific interests, which may be different. For instance, a user might be more interested in the person in Figure 14(a) or the car in Figure 14(b), rather than the elements that the model highlights. ChatCam, on the other hand, focuses on users’ interests through a lightweight object detection model, thereby significantly improving the F1 score. In summary, ChatCam can improve F1 scores by 7.30-524.47% compared to state-of-the-art approaches by customizing the model based on user interests.

7.4.2 Workflow of VectorDB-based Query System. We utilize a Jetson Xavier NX [6] as the edge server and it deploys some relevant models based on user interests. The edge server will continuously analyze the incoming video streams and then record the results generated from models. We conducted an experiment centered on the area by the entrance of a store with user interest “Please pay attention to the pedestrians outside the store” to show the workflow of the VectorDB-based query system in detail. Guided by the Actor-Critic LLM process, ChatCam identified several pertinent tasks that the system could perform. These included detecting individuals passing by the store’s entrance and monitoring for any occurrences of violent incidents. The “individual detection” model further distinguishes men and women by a classification model. The “violence detection” model focuses on classifying whether a fight has occurred.

We use three different user queries “Did anyone pass by at 11:44 on October 28, 2023?”, “Did a man pass by at 11:44 on October 28, 2023?” and “Did a fight happen at 14:06 on October 28, 2023?” to demonstrate how well they match the stored vectors and compare the performance of two embedding models.

Retrieve video summarizations. ChatCam employs a two-stage process to efficiently handle user questions. Initially, the system encapsulates both the user’s questions and the descriptions of all models deployed on the

Table 6. Match performance for different queries.

Embedding model	Weight Size	Metric	Query1	Query2	Query3
all-mpnet-base-v2 [1]	420MB	Similarity	A man passed by. (0.4792)	A man passed by. (0.7627)	A fight happens. (0.8046)
		Time	A woman passed by. (0.4481)	A woman passed by. (0.5869)	0.0121s
all-MiniLM-L6-v2 [73]	80MB	Similarity	A man passed by. (0.4432)	A man passed by. (0.7475)	A fight happens. (0.8510)
		Time	A woman passed by. (0.3793)	A woman passed by. (0.4058)	0.0063s

edge server into a structured prompt as delineated in Figure 9(a). Upon receiving the prompt, the LLM analyzes the user’s question in conjunction with the capabilities of the various models deployed on the edge server and then selects some relevant ones. For “Query1” and “Query2”, ChatCam selected the individual detection model, whereas for “Query3” it opted for the violence detection model. Additionally, ChatCam extracts the time specified in the user’s query and filters out irrelevant logs based on this time information.

With the appropriate model identified, the vector database at the edge is responsible for embedding the user’s question into a searchable vector format. The system then performs a vector similarity match, comparing the embedded user query with the stored video summarizations to retrieve the higher relevance ones. Table 6 shows the match performance of three queries under two different embedding models. Among these, all-mpnet-base-v2 [1] is the model with the highest accuracy in Sentence-Transformer. It can be observed that the two embedding models yield similar results in measuring the similarity between logs and queries including “Query1” and “Query2”. When the user’s query is about whether a man is present, all-MiniLM-L6-v2 [73] can achieve a larger similarity difference compared to all-mpnet-base-v2. This ability to accurately distinguish user queries for different classification tasks within the same context is exactly what ChatCam needs. Additionally, the retrieval time for the all-MiniLM-L6-v2 model is only half that of all-mpnet-base-v2, and its weight size is just 20%. This performance validates our choice of using “all-MiniLM-L6-v2” as the embedding model.

Generate answer. Upon successfully retrieving the relevant video summarizations from the vector database, our system then proceeds to the next critical phase. We encapsulate both the extracted video summaries and the user’s original question into another carefully predefined prompt as outlined in Figure 9(b). The LLM in the cloud is then engaged to process this composite input. For the given example in Figure 15, the system produces a remarkably clear and direct answer to the user’s question, “There are two people who passed by the store entrance at 11:44 on October 28, 2023.”, which is consistent with the ground truth. The accuracy of this answer is a testament to the efficacy of our approach in leveraging the LLM’s natural language processing strengths to interpret and communicate complex data in a user-friendly manner.

7.4.3 Examples in Multiple Scenarios and User Study. To further validate the effectiveness of ChatCam, we expanded the case study across three domains: smart home, traffic surveillance, and mall surveillance. For each domain, we listed some user interests and the corresponding tasks generated by ChatCam, as shown in Table 7. It can be seen that ChatCam is capable of effectively generating relevant tasks for user interests across different domains.

In addition, we conducted a real user study with 15 participants to evaluate ChatCam’s performance in a practical setting. Specifically, each user asked several questions related to different interests, including both clear and vague ones. We collected a total of 600 matching questions and results show that ChatCam can successfully address 70.33% of the questions. Specifically, it can solve 72.33% of the questions about clear interests and 68.33% of the questions about vague interests. These findings demonstrate ChatCam’s robust capability in handling a wide range of user queries, effectively addressing both explicit and ambiguous user interests.

Table 7. Examples of interests and their corresponding tasks.

Domain	Interests	Tasks
Smart Home	Please pay attention to whether the elderly at home has fallen.	Fall Detection
	Please pay attention to the elderly.	Basic Activity Monitoring Health Monitoring Safety Monitoring Social Interaction Monitoring
Traffic Surveillance	Please pay attention to the vehicle type on the road.	Vehicle Type Identification
	Please pay attention to the traffic condition.	Vehicle Type Monitoring Traffic Flow Monitoring Road Incident Monitoring
Mall Surveillance	Please pay attention to whether there is any violence in the store.	Physical Altercation Detection Vandalism Detection
	Please pay attention to the customers at the store entrance.	Individuals Monitoring Violence Detection

8 Discussion and Future Work

In this section, we discussed some issues of ChatCam to enhance the understanding of the implications and limitations of the proposed paradigm and application.

8.1 User interests increase and update.

When a user expresses a new interest, ChatCam follows a structured set of processes to address it. Each interest is treated equivalently, and the system first generates the corresponding tasks. Following this, the MLLM is used to generate the required model, and the DRL agent is used to determine the weight sharing strategy among all models. Additionally, once the model is deployed at the edge, users can set multiple queries related to multiple interests at any time. ChatCam will then retrieve the relevant summarizations to provide answers.

Regarding the update of user interests, ChatCam has not yet implemented this feature. If a user's interests change significantly from their initial preferences, ChatCam could address this by regenerating some related models. If the user's interests become more specific, some models might be undeployed. Conversely, if the interests become more general, additional models may need to be deployed. This update mechanism raises an interesting question and is an area we hope to explore in future work.

8.2 Power consumption.

ChatCam deploys all user interest-related models on the edge server. In typical video surveillance scenarios, we assume that edge servers are powered by an active power source, so power consumption is not currently a consideration in ChatCam's design. Considering that edge devices may need to be power-efficient, a smaller number of multi-category classification models may be more power-efficient than a larger number of binary classification models.

In future work, we aim to incorporate energy consumption constraints into the DRL decision process to better accommodate specific scenarios. Additionally, we plan to use a combination of multiple multi-category classification models and some weight-sharing binary classification models. This approach aims to balance the requirements of power consumption and accuracy, ensuring that ChatCam remains efficient and effective across various deployment environments.

8.3 Interaction paradigm.

Unlike existing commercial off-the-shelf (COTS) cameras with predefined tasks, we aim to achieve a more natural and extensible dialogue mechanism between users and cameras. Users can express their interests to the

camera for subsequent monitoring, and the camera needs to understand these interests and record relevant video summarizations. Additionally, after some time, users should be able to ask the camera specific questions, much like communicating with a person. To achieve this, we proposed ChatCam, leveraging the powerful language understanding capabilities of LLMs.

ChatCam allows users to express their specific interests to the camera, breaking the boundaries of predefined tasks. ChatCam also enables users to query events related to their expressed interests. When users pose specific queries, ChatCam utilizes LLMs to summarize the information retrieved from a vector database, providing precise and contextually relevant answers. This conversational paradigm feels natural and makes users feel as though they are interacting with a real person.

8.4 Real-world scenarios.

Regarding the application of ChatCam in real-world scenarios, we discussed its current capabilities and areas for improvement. On the one hand, similar to existing work in the video surveillance field [16, 71], ChatCam adopts a two-stage approach to object detection tasks. ChatCam utilizes Tiny YOLO as the object detection model and MobileNetV3-Large for classifying the state or behavior of objects. In general, the task that ChatCam aims to accomplish is the most common object detection task in video surveillance, and it supports clearer classification of objects. However, when users need to pay attention to counts or relationships between objects, ChatCam currently cannot fulfill these requirements due to limitations in the model's capabilities. We hope to improve performance on these tasks by incorporating some lightweight image captioning models in future work.

On the other hand, ChatCam does not require additional user participation to understand user interests but has the ability to interact with users. Alternatively, once ChatCam reaches convergence, asking users to proofread the tasks and suggest adjustments is also a feasible solution. In addition, in situations where users may struggle to monitor certain aspects on their own like "Please pay attention to the diversity of customer demographics", ChatCam can act as an auxiliary tool by suggesting tasks and iterating with the user.

9 Conclusion

In this paper, we proposed a novel human-to-camera interaction paradigm known as contextual chatting-to-camera which allows users to interact with the camera in a more natural way. To realize this, we designed ChatCam with interest-oriented video summarization that can understand the users' interests correctly. Major contributions of ChatCam include the prompt with Actor-Critic LLMs method to improve LLM performance in understanding users' interests, MLLM-based model generation, and the DRL-based model deployment method to deploy several models at the resource-constrained edge. We also implemented a video summarization query system based on a vector database. Our evaluation of ChatCam under different settings shows that it excels in understanding users' interests while maintaining model accuracy at the edge. We also conducted a real user study, and ChatCam can solve 70.33% questions collected from 15 participants. Additionally, we have discussed various issues encountered during the deployment of ChatCam in the real-world scenarios and proposed directions for future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 62272407 and 62072396, the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under grant No. 2023C01033, and the National Youth Talent Support Program. The work is also supported by Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

- [1] 2021. all-mpnet-base-v2. <https://huggingface.co/sentence-transformers/all-mpnet-base-v2>.
- [2] 2023. Chroma. <https://github.com/chroma-core/chroma>.
- [3] 2023. Evaluating the ideal chunk size for a RAG system using LLaMaindex. <https://www.llamaindex.ai>.
- [4] 2023. Gemini. <https://deepmind.google/technologies/gemini>.
- [5] 2023. GPT-4V(ision). <https://openai.com/research/gpt-4v-system-card>.
- [6] 2023. Jetson Xavier NX. <https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-xavier-nx/>.
- [7] 2023. Qdrant. <https://github.com/qdrant/qdrant>.
- [8] 2023. Recursively split by character. <https://python.langchain.com/docs>.
- [9] Sahar Abdelnabi, Amr Goma, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. 2023. Llm-deliberation: Evaluating llms with interactive multi-agent negotiation games. *arXiv preprint arXiv:2309.17234* (2023).
- [10] Pranav Adarsh, Pratibha Rathi, and Manoj Kumar. 2020. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In *2020 6th international conference on advanced computing and communication systems (ICACCS)*. IEEE, 687–694.
- [11] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).
- [12] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 119–135.
- [13] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya Dulloor. 2019. Scaling video analytics on constrained edge nodes. *Proceedings of Machine Learning and Systems 1* (2019), 406–417.
- [14] Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A Cire. 2021. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3677–3687.
- [15] Kabalan Chaccour, Rony Darazi, Amir Hajjam El Hassani, and Emmanuel Andres. 2016. From fall detection to fall prevention: A generic classification of fall-related systems. *IEEE Sensors Journal* 17, 3 (2016), 812–822.
- [16] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 155–168.
- [17] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. 2022. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3852–3878.
- [18] Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4794–4802.
- [19] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017).
- [20] Zhou Fan, Rui Su, Weinan Zhang, and Yong Yu. 2019. Hybrid actor-critic reinforcement learning in parameterized action space. *arXiv preprint arXiv:1903.01344* (2019).
- [21] Aleksandr Fedorov, Kseniia Nikolskaia, Sergey Ivanov, Vladimir Shepelev, and Alexey Minbaleev. 2019. Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data* 6 (2019), 1–15.
- [22] Paulo Finardi, Leonardo Avila, Rodrigo Castaldoni, Pedro Gengo, Celio Larcher, Marcos Piau, Pablo Costa, and Vinicius Caridá. 2024. The Chronicles of RAG: The Retriever, the Chunk and the Generator. *arXiv preprint arXiv:2401.07883* (2024).
- [23] Sven Fleck and Wolfgang Straßer. 2008. Smart camera based monitoring system and its application to assisted living. *Proc. IEEE* 96, 10 (2008), 1698–1714.
- [24] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).
- [25] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129 (2021), 1789–1819.
- [26] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)* 42, 6 (2012), 1291–1307.
- [27] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. 2021. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921* (2021).
- [28] Anhong Guo, Anuraag Jain, Shomiron Ghose, Gierad Laput, Chris Harrison, and Jeffrey P Bigham. 2018. Crowd-ai camera sensing in the real world. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–20.
- [29] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, et al. 2022. Manu: a cloud native vector database management system. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3548–3561.

- [30] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680* (2024).
- [31] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* (2015).
- [32] Kyota Higa and Kota Iwamoto. 2018. Robust estimation of product amount on store shelves from a surveillance camera for improving on-shelf availability. In *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 1–6.
- [33] Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. CogAgent: A Visual Language Model for GUI Agents. [arXiv:2312.08914](https://arxiv.org/abs/2312.08914) [cs.CV]
- [34] Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th international conference on pattern recognition*. IEEE, 2366–2369.
- [35] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.
- [36] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 269–286.
- [37] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 115–131.
- [38] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 253–266.
- [39] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. Blazelt: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046* (2018).
- [40] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529* (2017).
- [41] Minsu Kim, Junyoung Park, and Jinkyoo Park. 2022. Sym-nco: Leveraging symmetry for neural combinatorial optimization. *Advances in Neural Information Processing Systems* 35 (2022), 1936–1949.
- [42] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.
- [43] Chunyuan Li. 2023. Large multimodal models: Notes on cvpr 2023 tutorial. *arXiv preprint arXiv:2306.14895* (2023).
- [44] Franklin Mingzhe Li, Di Laura Chen, Mingming Fan, and Khai N Truong. 2019. FMT: A wearable camera-based object tracking memory aid for older adults. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–25.
- [45] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems* 36 (2023), 51991–52008.
- [46] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597* (2023).
- [47] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470* (2023).
- [48] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 359–376.
- [49] Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. 2015. Customer behavior recognition in retail store from surveillance camera. In *2015 IEEE international symposium on multimedia (ISM)*. IEEE, 154–159.
- [50] Shinan Liu, Tarun Mangla, Ted Shao Wang, Jinjin Zhao, John Paparrizos, Sanjay Krishnan, and Nick Feamster. 2023. AMIR: Active Multimodal Interaction Recognition from Video and Network Traffic in Connected Environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–26.
- [51] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023).
- [52] Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems* 35 (2022), 17703–17716.
- [53] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
- [54] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roi Herzig. 2024. Compositional chain-of-thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14420–14431.
- [55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fiedjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015),

- 529–533.
- [56] Medhini Narasimhan, Anna Rohrbach, and Trevor Darrell. 2021. Clip-it! language-guided video summarization. *Advances in neural information processing systems* 34 (2021), 13988–14000.
- [57] Arthi Padmanabhan, Neil Agarwal, Anand Iyer, Ganesh Ananthanarayanan, Yuanhao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. 2023. Gemel: Model Merging for {Memory-Efficient}, {Real-Time} Video Analytics at the Edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 973–994.
- [58] Junting Pan, Ziyi Lin, Yuying Ge, Xiatian Zhu, Renrui Zhang, Yi Wang, Yu Qiao, and Hongsheng Li. 2023. Retrieving-to-answer: Zero-shot video question answering with frozen large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 272–283.
- [59] Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S Khan. 2024. Glamm: Pixel grounding large multimodal model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13009–13018.
- [60] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*. Springer, 525–542.
- [61] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015).
- [62] Andrzej Ruta, Fatih Porikli, Shintaro Watanabe, and Yongmin Li. 2011. In-vehicle camera traffic sign detection and recognition. *Machine Vision and Applications* 22 (2011), 359–375.
- [63] Aidean Sharghi, Boqing Gong, and Mubarak Shah. 2016. Query-focused extractive video summarization. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VIII 14*. Springer, 3–19.
- [64] Aidean Sharghi, Jacob S Laurel, and Boqing Gong. 2017. Query-focused video summarization: Dataset, evaluation, and a memory network based approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4788–4797.
- [65] George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. 2023. ZipIt! Merging Models from Different Tasks without Training. *arXiv preprint arXiv:2305.03053* (2023).
- [66] Tomoya Sugihara, Shuntaro Masuda, Ling Xiao, and Toshihiko Yamasaki. 2024. Language-Guided Self-Supervised Video Summarization Using Text Semantic Matching Considering the Diversity of the Video. *arXiv preprint arXiv:2405.08890* (2024).
- [67] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.
- [68] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. 2022. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100* (2022).
- [69] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data*. 2614–2627.
- [70] Lin Wang and Kuk-Jin Yoon. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence* 44, 6 (2021), 3048–3068.
- [71] Shibo Wang, Shusen Yang, and Cong Zhao. 2020. SurveilEdge: Real-time video query based on collaborative cloud-edge deep learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2519–2528.
- [72] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. CogVLM: Visual Expert for Pretrained Language Models. *arXiv:2311.03079* [cs.CV]
- [73] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *arXiv:2002.10957* [cs.CL]
- [74] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).
- [75] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747* (2022).
- [76] Zhenhailong Wang, Manling Li, Ruochen Xu, Luwei Zhou, Jie Lei, Xudong Lin, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Derek Hoiem, et al. 2022. Language models with image descriptors are strong few-shot video-language learners. *Advances in Neural Information Processing Systems* 35 (2022), 8483–8497.
- [77] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
- [78] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*. PMLR, 23965–23998.

- [79] Guande Wu, Jianzhe Lin, and Claudio T Silva. 2022. Intentvizer: Towards generic query guided interactive video summarization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10503–10512.
- [80] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4820–4828.
- [81] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).
- [82] Ronghua Xu, Seyed Yahya Nikouei, Yu Chen, Aleksey Polunchenko, Sejun Song, Chengbin Deng, and Timothy R Faughnan. 2018. Real-time human objects tracking for smart surveillance at the edge. In *2018 IEEE International conference on communications (ICC)*. IEEE, 1–6.
- [83] Zhenran Xu, Senbao Shi, Baotian Hu, Jindi Yu, Dongfang Li, Min Zhang, and Yuxiang Wu. 2023. Towards reasoning in large language models via multi-agent peer review collaboration. *arXiv preprint arXiv:2311.08152* (2023).
- [84] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).
- [85] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2636–2645.
- [86] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490* (2023).
- [87] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).
- [88] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. 2024. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289* (2024).
- [89] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).

A Prompt Details

In this section, we provide the complete prompt template that includes the role that the LLM is required to play, along with specific rules and examples to help the LLM respond more effectively.

Figure 16 shows the complete prompt template of the ActorLLM. We configure the LLM to function as ActorLLM by defining its Role, instructing it to analyze the user's interests, and generate corresponding tasks. Concurrently, ActorLLM is also tasked with refining its output in response to feedback received from CriticLLM. Additionally, we establish specific rules to guide the behavior of ActorLLM, including the output format, as well as providing examples and setting requirements to ensure that the tasks generated are consistent and aligned with the intended objectives.

ActorLLM in ChatCam

Role
You are a useful actor assistant. You can parse the user's interest into some objects, and these objects can be detected by simple visual AI models. Then one or more objects need to be combined into a task, and the objects in the task need to be related and there is no overlap or confusion. Moreover, tasks will be evaluated by critic assistant, and you modify tasks based on the evaluation.

Rules
1. Your response must be in JSON and must include a "Thought" + "Tasks". Tasks is an array of tasks. Each task contains 3 fields: "task": the name of the task; "objects": the objects contained in the task; "description": a brief description of the task. For example, {"task": "Elderly Health Monitoring", "objects": ["elderly eating", "elderly sleeping"], "description": "Observing eating and sleeping patterns to ensure the elderly's health and well-being."}. The object itself should be clear, for example, the object should be "elderly standing" instead of "standing". And Object should be general and should not contain personal things, such as strangers, hobbies, etc.
2. The objects involved must be detectable through vision-related classification models. The critic assistant's evaluation will include the precision of each object. If the recall is lower than 80%, it means that the object cannot be implemented currently and needs to be deleted.
3. Objects in the same task should be related and free of overlap and confusion. The critic assistant's evaluation will include the recall of each object. If the recall is higher than 80%, but the precision is lower than 80%, it means that this object overlaps or is confused with other objects. You should consider deleting one of them, or moving one of them to other tasks or a new task.

Examples
USER: Interest: Please pay attention to the state of the elderly at home
ASSISTANT: {"Thought": "The user wants to monitor various aspects of an elderly person's activities at home. This could include their basic activity, health-related activities, safety, and entertainment activities.", "Tasks": [{"task": "Elderly Basic Activity Monitoring", "objects": ["elderly walking", "elderly sitting", "elderly standing"], "description": "Monitoring the elderly's basic movements to track their physical activity levels."}, {"task": "Elderly Health Monitoring", "objects": ["elderly eating", "elderly sleeping"], "description": "Observing eating and sleeping patterns to ensure the elderly's health and well-being."}, {"task": "Elderly Safety Monitoring", "objects": ["elderly falling"], "description": "Detecting falls to ensure immediate response in case of emergencies."}, {"task": "Elderly Entertainment Activities Monitoring", "objects": ["elderly watching TV"], "description": "Monitoring entertainment activities watching TV."}]}}
USER:
Evaluation: The tasks are aligned with the user's interest in monitoring the state of the elderly at home. However, within 'Elderly Basic Activity Monitoring', the objects 'elderly walking' and 'elderly standing' may have some overlap for image-process AI model, you can delete 'elderly walking'. 'Elderly Health Monitoring' is reasonable with 'elderly eating' and 'elderly sleeping' as these activities are distinct and important for health tracking. The 'Elderly Safety Monitoring' task is clear with the single object 'elderly falling', which is crucial for emergency response systems. The 'Elderly Entertainment Activities Monitoring' with the object 'elderly watching TV' is specific and relevant for assessing leisure activities.
Precision_recall: [{"task": "Elderly Basic Activity Monitoring", "precision": {"elderly walking": 0.56, "elderly sitting": 0.83, "elderly standing": 0.41}, "recall": {"elderly walking": 1.0, "elderly sitting": 1.0, "elderly standing": 0.9}}, {"task": "Elderly Health Monitoring", "precision": {"elderly eating": 0.9, "elderly sleeping": 1.0}, "recall": {"elderly eating": 0.9, "elderly sleeping": 1.0}}, {"task": "Elderly Safety Monitoring", "precision": {"elderly falling": 1.0}, "recall": {"elderly falling": 1.0}}, {"task": "Elderly Entertainment Activities Monitoring", "precision": {"elderly watching TV": 1.0}, "recall": {"elderly watching TV": 1.0}}]
Rating: 75
ASSISTANT: {"Thought": "The critic has pointed out that 'elderly walking' and 'elderly standing' may have some overlap for image-processing AI models. To address this issue, I will remove 'elderly walking' from the 'Elderly Basic Activity Monitoring' task. The other tasks seem to be well-received and aligned with monitoring the state of the elderly at home.", "Tasks": [{"task": "Elderly Basic Activity Monitoring", "objects": ["elderly sitting", "elderly standing"], "description": "Monitoring the elderly's basic postures to track their physical activity levels and ensure they are comfortably engaging in daily activities."}, {"task": "Elderly Health Monitoring", "objects": ["elderly eating", "elderly sleeping"], "description": "Observing eating and sleeping patterns to ensure the elderly's health and well-being."}, {"task": "Elderly Safety Monitoring", "objects": ["elderly falling"], "description": "Detecting falls to ensure immediate response in case of emergencies."}, {"task": "Elderly Entertainment Activities Monitoring", "objects": ["elderly watching TV"], "description": "Monitoring entertainment activities to assess leisure time engagement and ensure the elderly are entertained."}]}}

Fig. 16. Prompt template of ActorLLM.

Figure 17 shows the complete prompt template of the CriticLLM. We configure the LLM to function as CriticLLM by defining its Role, instructing it to analyze the output of ActorLLM and MLLM, and then generate corresponding feedback. In addition, we have established specific rules to guide the behavior of CriticLLM, including output formats, providing examples, and requiring it to give appropriate ratings.

CriticLLM in ChatCam

Role
You are a useful critic assistant. Actor assistant can parse the user's interest into some objects, and these objects can be detected by simple visual AI models. Then one or more objects need to be combined into a task, and the objects in the task need to be related and there is no overlap or confusion. Based on user's interest and the task's precision and recall, you need to judge whether the tasks generated by the actor are reasonable and whether they need to be modified. Actor assistant will adjust tasks based on your evaluation. After new tasks are given, you need to re-evaluate.

Rules

1. Your response must be in JSON and must include a "Evaluation" + "Rating". In "Evaluation", based on user interest and the given precision and recall, you need to evaluate whether the tasks are reasonable and comprehensive and whether the tasks need to be modified. After providing "Evaluation", in "Rating", you can rate tasks overall on a scale of 1 to 100.
2. The objects involved must be detectable through vision-related classification models. The object itself should be clear, for example, the object should be "elderly eating" instead of "eating". And Object should be general and should not contain personal things, such as strangers, etc. If the recall is lower than 80%, it means that the object cannot be implemented currently and needs to be deleted.
3. Objects in the same task should be related and free of overlap and confusion. If the recall is higher than 80%, but the precision is lower than 80%, it means that this object overlaps or is confused with other objects. You should consider deleting one of them, or moving one of them to other tasks or a new task.
4. If the rating is higher than 95, it means that you think the current tasks can well satisfy the user's interest and can be performed well. Therefore, if there is an object whose precision and recall are lower than 80, the rating should not be higher than 95. If the score is lower than 95, you need to give further suggestions for revision.

Examples
USER: Interest: Please pay attention to the state of the elderly at home.
Thought: The user wants to monitor various aspects of an elderly person's activities at home. This could include their movement, health-related activities, safety, and entertainment activities.
Tasks: [{"task": "Elderly Activity Monitoring", "objects": ["elderly walking", "elderly sitting", "elderly standing"], "description": "Monitoring the elderly's basic movements to track their physical activity levels."}, {"task": "Elderly Health Monitoring", "objects": ["elderly eating", "elderly sleeping"], "description": "Observing eating and sleeping patterns to ensure the elderly's health and well-being."}, {"task": "Elderly Safety Monitoring", "objects": ["elderly falling"], "description": "Detecting falls to ensure immediate response in case of emergencies."}, {"task": "Elderly Entertainment Activities Monitoring", "objects": ["elderly watching TV"], "description": "Monitoring entertainment activities watching TV."}]
Precision_recall: [{"task": "Elderly Basic Activity Monitoring", "precision": {"elderly walking": 0.56, "elderly sitting": 0.83, "elderly standing": 0.41}, "recall": {"elderly walking": 1.0, "elderly sitting": 1.0, "elderly standing": 0.9}}, {"task": "Elderly Health Monitoring", "precision": {"elderly eating": 0.9, "elderly sleeping": 1.0}, "recall": {"elderly eating": 0.9, "elderly sleeping": 1.0}}, {"task": "Elderly Safety Monitoring", "precision": {"elderly falling": 1.0}, "recall": {"elderly falling": 1.0}}, {"task": "Elderly Entertainment Activities Monitoring", "precision": {"elderly watching TV": 1.0}, "recall": {"elderly watching TV": 1.0}}]
ASSISTANT: {"Evaluation": "In 'Elderly Activity Monitoring', 'elderly walking' and 'elderly standing' have low precision, this may be caused by the similarity between 'walking' and 'standing', you can remove 'elderly walking'. 'Elderly Health Monitoring', 'Elderly Safety Monitoring' and 'Elderly Entertainment Activities Monitoring' are ok", "Rating": 75}
USER:
Thought: Based on the evaluation, 'elderly walking' and 'elderly standing' are causing precision issues due to their similarity. It is recommended to remove 'elderly walking' and keep 'elderly standing' as it is more general and can encompass walking.
Tasks: [{"task": "Elderly Basic Activity Monitoring", "objects": ["elderly sitting", "elderly standing"], "description": "Monitoring the elderly's basic movements to track their physical activity levels."}, {"task": "Elderly Health Monitoring", "objects": ["elderly eating", "elderly sleeping"], "description": "Observing eating and sleeping patterns to ensure the elderly's health and well-being."}, {"task": "Elderly Safety Monitoring", "objects": ["elderly falling"], "description": "Detecting falls to ensure immediate response in case of emergencies."}, {"task": "Elderly Entertainment Activities Monitoring", "objects": ["elderly watching TV"], "description": "Monitoring entertainment activities watching TV."}]
Precision_recall: [{"task": "Elderly Basic Activity Monitoring", "accuracy": {}, "precision": {"elderly sitting": 0.83, "elderly standing": 0.82}, "recall": {"elderly sitting": 1.0, "elderly standing": 0.9}}, {"task": "Elderly Health Monitoring", "accuracy": {}, "precision": {"elderly eating": 1.0, "elderly sleeping": 1.0}, "recall": {"elderly eating": 0.9, "elderly sleeping": 0.9}}, {"task": "Elderly Safety Monitoring", "accuracy": {}, "precision": {"elderly falling": 1.0}, "recall": {"elderly falling": 1.0}}, {"task": "Elderly Entertainment Activities Monitoring", "accuracy": {}, "precision": {"elderly watching TV": 1.0}, "recall": {"elderly watching TV": 1.0}}]
ASSISTANT: {"Evaluation": "The modification to 'Elderly Basic Activity Monitoring' by removing 'elderly walking' and keeping 'elderly standing' has improved the precision of the task. All tasks now have acceptable precision and recall rates, aligning well with the user's interest in monitoring the elderly's activities, health, safety, and entertainment at home.", "Rating": 90}

Fig. 17. Prompt template of CriticLLM.

B Appendix for evaluation

In this section, we provide the complete customized dataset and the complete results in the evaluation section.

B.1 Customized dataset Details

Table 8. Customized dataset that contains a series of real users' interests.

Domain	Clear interests	Vague interests
Smart Home	Please pay attention to whether the elderly at home has fallen.	Please pay attention to the elderly.
	Please pay attention to the child's eating and sleeping patterns.	Please pay attention to the child.
	Please pay attention to whether there is anyone at the door.	Please pay attention to the safety situation at home.
	Please pay attention to whether the pet is on the sofa.	Please pay attention to the situation in the living room at home.
Traffic Surveillance	Please pay attention to whether someone is cooking in the kitchen.	Please pay attention to the dog at home.
	Please pay attention to the vehicle type on the road.	Please pay attention to the traffic condition.
	Please pay attention to whether there are cracks on the road surface.	Please pay attention to the behavior of drivers.
	Please pay attention to whether there is a car accident.	Please pay attention to the presence of any road hazards.
	Please pay attention to whether there is a traffic jam.	Please pay attention to the overall traffic flow.
Mall Surveillance	Please pay attention to whether there are pedestrians on the road.	Please pay attention to the parking lot.
	Please pay attention to whether there is any violence in the store.	Please pay attention to the customers in the store.
	Please pay attention to any signs of fire or smoke.	Please pay attention to the customers at the store entrance.
	Please pay attention to any customer disputes or fights.	Please pay attention to customer shopping behaviors.
	Please pay attention to any customers falling.	Please pay attention to the diversity of customer demographics.
	Please pay attention to whether there are pregnant women.	Please pay attention to the pedestrians outside the store.

B.2 Iterations for the different interests

To illustrate the impact and feasibility of MLLM on iterative processes, we present detailed information on different iterations tailored to various user interests. Table 9 shows the complete results.

Table 9. Information on the different iterations for four interests (2 clear and 2 vague).

Interest	Iteration	Task and Objects	Precision	Recall	Rating
Elderly fallen	1	Elderly Fall Detection: "elderly falling"	1	1	100
Child eating and sleeping patterns	1	Child Eating Monitoring: "child eating"	1	1	100
		Child Sleeping Monitoring: "child sleeping"	1	1	
Home safety situation	1	Smoke and Fire Detection: "smoke", "fire"	0.71, 0.56	1, 1	65
		Intrusion Detection: "door forced open", "window broken"	0.5, 0.59	0.9, 1	
		Occupant Safety Monitoring: "person falling", "person lying still"	0.71, 0.88	1, 0.7	
	2	Smoke Detection: "smoke"	1	1	85
		Intrusion Detection: "unknown person", "window broken"	1, 1	0.7, 1	
		Occupant Safety Monitoring: "person falling"	1	1	
3	Smoke Detection: "smoke"	1	1	100	
	Intrusion Detection: "window broken"	1	1		
	Occupant Safety Monitoring: "person falling"	1	1		
Traffic situation	1	Vehicle Type Monitoring: "light vehicles", "truck", "motorcycle", "bicycle", "bus"	0.2, 0.71, 1, 1, 0.91	0.8, 1, 1, 1, 1	70
		Traffic Flow Monitoring: "traffic jam", "traffic light"	1, 0.91	1, 1	
		Road Incident Monitoring: "car accident", "road construction"	1, 1	1, 0.9	
	2	Vehicle Type Monitoring: "car", "truck", "motorcycle", "bicycle", "bus"	0.83, 1, 1, 1, 1	1, 1, 1, 1, 1	85
		Traffic Flow Monitoring: "traffic jam", "traffic light"	1, 0.91	1, 1	
		Road Incident Monitoring: "car accident", "road construction"	1, 0.89	1, 0.8	
	3	Vehicle Type Monitoring: "car", "truck", "motorcycle", "bicycle", "bus"	0.83, 1, 1, 1, 1	1, 0.9, 1, 1, 1	95
		Traffic Flow Monitoring: "traffic jam", "traffic light"	1, 0.91	1, 1	
		Road Incident Monitoring: "car accident"	1	1	

To illustrate the impact and feasibility of MLLM on iterative processes, we present detailed information on different iterations tailored to various user interests. As shown in Table 9, for two well-defined interests—namely,

monitoring elderly falls and observing children's eating and sleeping patterns—ChatCam can achieve convergence in just one iteration.

In these cases, the tasks generated align perfectly with the user's initial interests, demonstrating the system's efficiency and effectiveness. When dealing with more ambiguous user interests, ChatCam requires multiple rounds of iterations. During these iterations, the system adjusts the division of tasks based on the feedback from MLLM. By evaluating the model's feasibility through MLLM, tasks that exceed the model's capacity can be identified and either removed or modified accordingly. These necessary adjustments are then communicated back to ActorLLM, which in turn re-plans the task division to better align with the model's capabilities and user requirements.

For a user's vague interest in "Please pay attention to the home safety situation", ChatCam generated three tasks in its first iteration: "Smoke and Fire Detection", "Intrusion Detection" and "Occupant Safety Monitoring". Based on the critic's feedback on feasibility from the MLLM, ActorLLM adjusts the tasks in its second iteration to improve the precision of object detection. For "Smoke and Fire Detection", ActorLLM retains "smoke" but removes "fire" due to its low precision. In "Intrusion Detection", ActorLLM removes both "door forced open" because of its low precision and potential confusion with other situations. ActorLLM considers alternative objects or different approaches to detect intrusion, such as adding "unknown person". For "Occupant Safety Monitoring", since "person lying still" has a low recall, ActorLLM removes it and evaluates whether "person falling" can be improved. In the third iteration, while the "Smoke Detection" and "Occupant Safety Detection" tasks are performing well according to the critic's feedback, the "Intrusion Detection" task requires further refinement. The object "unknown person" has high precision but low recall, indicating inconsistent detection. Since MLLM is used for general summarization and cannot judge the existence of strangers, ActorLLM removed the "unknown person".

For a user's vague interest in "Please pay attention to the traffic situation", ChatCam generated three tasks in its first iteration: "Vehicle Type Monitoring", "Traffic Flow Monitoring", and "Road Incident Monitoring". Based on the critic's feedback on feasibility from the MLLM, ActorLLM adjusts the tasks in its second iteration. For "Vehicle Type Monitoring", ActorLLM changes "light vehicles" to "car" due to its low precision. To meet the diversity of scenarios, ChatCam uses different pictures for pre-evaluation each time using MLLM, so the results obtained will also vary. Therefore, for the "Road Incident Monitoring" task, it was found that the precision of "road construction" in the second iteration differed from the first iteration. In real-world scenarios, it is sufficient to monitor road conditions by determining whether a traffic accident has occurred, so ActorLLM removed "road construction". In the third iteration, all three tasks are performing well according to the critic's feedback.

By continuously refining tasks based on MLLM feedback, ChatCam effectively adapts to the user's interests, ensuring higher precision and recall in object detection tasks. This iterative process, guided by MLLM evaluations and ActorLLM adjustments, enhances the system's ability to meet user needs accurately and efficiently.