

Responsive DNN Adaptation for Video Analytics against Environment Shift via Hierarchical Mobile-Cloud Collaborations

Maozhe Zhao

Shanghai Jiao Tong University
Shanghai, China
larval_roc@sjtu.edu.cn

Fan Wu

Shanghai Jiao Tong University
Shanghai, China
fwu@sjtu.edu.cn

Shengzhong Liu*

Shanghai Jiao Tong University
Shanghai, China
shengzhong@sjtu.edu.cn

Guanghai Chen

Shanghai Jiao Tong University
Shanghai, China
gchen@sjtu.edu.cn

Abstract

Mobile video analysis systems often encounter various deploying environments, where environment shifts present greater demands for responsiveness in adaptations of deployed “expert DNN models”. Existing model adaptation frameworks primarily operate in a cloud-centric way, exhibiting degraded performance during adaptation and delayed reactions to environment shifts. Instead, this paper proposes MOCHA, a novel framework optimizing the responsiveness of continuous model adaptation through hierarchical collaborations between mobile and cloud resources. Specifically, MOCHA (1) reduces adaptation response delays by performing on-device model reuse and fast fine-tuning before requesting cloud model retrieval and end-to-end retraining; (2) accelerates history expert model retrieval by organizing them into a structured taxonomy utilizing domain semantics analyzed by a cloud foundation model as indices; (3) enables efficient local model reuse by maintaining onboard expert model caches for frequent scenes, which proactively prefetch model weights from the cloud model database. Extensive evaluations with real-world videos on three DNN tasks show MOCHA improves the model accuracy during adaptation by up to 6.8% while saving the response delay and retraining time by up to 35.5 \times and 3.0 \times respectively.

CCS Concepts

- Human-centered computing → Mobile computing;
- Computer systems organization → Embedded systems.

Keywords

Video Analytics, Mobile Computing, Continuous Learning.

*Shengzhong Liu is the corresponding author.

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.

SenSys '25, May 6–9, 2025, Irvine, CA, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1479-5/25/05
<https://doi.org/10.1145/3715014.3722044>

ACM Reference Format:

Maozhe Zhao, Shengzhong Liu, Fan Wu, and Guihai Chen. 2025. Responsive DNN Adaptation for Video Analytics against Environment Shift via Hierarchical Mobile-Cloud Collaborations . In *The 23rd ACM Conference on Embedded Networked Sensor Systems (SenSys '25), May 6–9, 2025, Irvine, CA, USA*. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3715014.3722044>

1 Introduction

Video analytics [3, 49, 67, 75] with deep neural networks (DNN) have been extensively applied in Internet of Things (IoT) applications, especially autonomous driving [21, 22, 66] and low-cost intelligent robots [10, 32, 77]. Mobile video analytics perform DNN-based inference on mobile devices to deliver real-time results [2, 80, 86]. Due to the resource constraints on mobile devices, dedicated *expert models*, with compressed size and distilled knowledge from the large-scale *teacher model*, are deployed to balance the prediction fidelity and processing throughput [17, 47, 69]. This makes the analytics system responsive and accurate to data that matches the expert model’s training distribution.

However, numerous tasks are not confined to a single scene. They are subject to various unpredictable domain shifts [52, 73], which alter the data distribution of collected video streams. In this paper, we focus on environment shifts, mainly referring to shift factors related to the natural environment, which is a type of domain shift and can easily be recognized in real-world applications. For example, light conditions can change dramatically when vehicles pass through consecutive tunnels, and backgrounds can shift significantly when re-entering elevators or buildings. In these environment shifts, while teacher models exhibit robustness, they cannot be directly applied to lightweight mobile devices. Expert models, on the other hand, can only memorize limited video scenes (*i.e.*, domain), resulting in poor generalization in environments deviating from their training domain [13]. Furthermore, the safety-critical nature of these tasks requires analytics systems to maintain high accuracy throughout their operations. A slight performance degradation at any time can impact the quality of service (QoS). Unlike regular working conditions, environment shifts put additional pressure on the systems’ accuracy and must be managed precisely.

To sustain stable DNN accuracy against environment shifts, continuous expert model adaptations are conducted in previous frameworks [16, 26, 33, 45], where two strategies are commonly exploited

: (1) *Model retraining* [5, 15, 59] updates the expert model parameters with freshly collected samples; (2) *Model reuse* [34] compares and selects one best candidate from a history model zoo to replace the obsolete one. However, due to resource-intensive computation or storage needs, existing solutions typically host model adaptation services on the cloud [6, 34, 73]. This produces delayed responses (*i.e.*, process of completing the adaptation and returning to high accuracy) to environment shifts: First, cloud-based model adaptation requires frequent data and model weight exchange (communication overhead time) between mobile and cloud [42, 50, 72], leading to considerable communication delays and bandwidth consumption; Second, centralized adaptation paradigms make the cloud a single point of congestion, leading to poor scalability to the number of mobile devices with long cloud queuing delays. In tasks involving multiple environments, environment shifts are inevitable and such shifts make QoS unacceptable with delayed responses. If DNN-based analytics systems can recover more quickly with stronger responsiveness by reducing the aforementioned overheads, they will achieve higher overall accuracy during environment shifts, agnostic to the model optimization algorithms used.

This paper optimizes the *response efficiency* of continuous model adaptation in mobile video analytics against environment shifts by efficiently exploiting mobile resources through a novel mobile-cloud collaborative paradigm, thereby improving performance in autonomous driving. The key intuition is to leverage lightweight mobile adaptation actions to reduce the frequency of requesting cloud adaptation services and avoid unnecessary mobile-cloud communications. Two technical challenges are raised: First, considering the cloud network delays and mobile resource constraints, how to design a mobile-cloud collaboration algorithm taking advantage of both the low latency on the mobile and rich resources on the cloud is complex. We need to rethink a brand-new arrangement of inference and various adaptations across mobile devices and cloud servers, one that differs from all previous systems. The integration of these design components is, in itself, a tough challenge. Second, forcibly moving intact model reuse and retraining functions from cloud servers to mobile devices induce excessive computation and storage load to onboard resources. We need to identify suitable meta-level information to reduce the algorithm's computational demand while ensuring accuracy and responsiveness.

We introduce MOCHA, a mobile-cloud collaborative model adaptation framework for mobile video analytics, which achieves the objective without introducing a new algorithm. It distributes hierarchical adaptation functions between mobile and cloud resources. We serve most model reuse and retraining requests locally on mobile through retrieval from a small expert model cache and updating the expert model with single-layer LoRA (Low-Rank Adaptation) fine-tuning while offloading intensive requests to the cloud server which hosts a full-fledged expert model database and runs end-to-end expert model retraining. The cloud only works as a backup to the mobile when lightweight onboard services do not suffice to achieve the adaptation objective.

The design of MOCHA includes three key components. First, MOCHA offloads lightweight model adaptations to the mobile devices for better system responsiveness, which composes a two-tier adaptation hierarchy. It applies on-device model reuse and fast

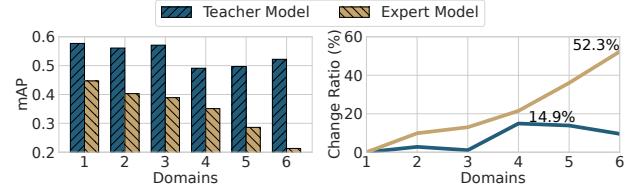


Figure 1: Teacher Model vs. Expert Model. The change ratio denotes the ratio of accuracy drop in the current domain over accuracy in the optimal domain, reflecting model generalizability in different environments.

LoRA single-layer fine-tuning as immediate actions against environment shifts, before data transmission to the cloud for cloud model retrieval and end-to-end retraining. Second, it constructs a structured model database organized by domain semantics, which can be analyzed by a foundation model (FM) on the cloud, to retrieve the fittest model candidate among history models in fast adaptations. Third, MOCHA maintains onboard model caches to store model weights for popular scenes, enabling seamless model switching with minimal onboard transmission delays. The mobile cache proactively interacts with the cloud model database for prefetching to ensure uninterrupted inference.

We evaluate MOCHA with two large-scale real-world video datasets on three analytics tasks: object detection with YOLO [29], image classification with ResNet [20], and semantic segmentation with Deeplabv3+ [9]. We extensively compare it against state-of-the-art (SOTA) model adaptation systems. The results show that MOCHA supports more devices than SOTA methods in lower response delays with the same cloud resources. As the mobile device number scales, MOCHA improves the accuracy during adaptation by up to 6.8%, and saves the response delay and retraining time by 35.5× and 3.0×. Ablation results validate that MOCHA's superior performance mainly comes from the onboard adaptations of fine-tuning and reuse with mobile model cache.

In summary, the main contributions of this paper are:

- We propose a mobile-cloud collaborative continuous adaptation framework MOCHA for effective mobile video analytics against runtime scene changes.
- We design an adaptation hierarchy of onboard model reuse, fine-tuning, and cloud retraining for better responsiveness.
- We introduce an automated semantic model taxonomy for real-time domain metadata-based model retrieval used within adaptation strategies.
- We implement MOCHA and perform extensive evaluations with a real-world video dataset, where MOCHA saves the adaptation latency by up to 35.5×.

2 Background and Motivations

Here, we introduce the background of continuous adaptations against frequent environment shifts in mobile video analytics (§2.1), analyze the advantages of the hierarchical model adaption (§2.2), and give intuitions of semantics-indexed model retrieval (§2.3).



Figure 2: Example frames of the domains in Figure 1.

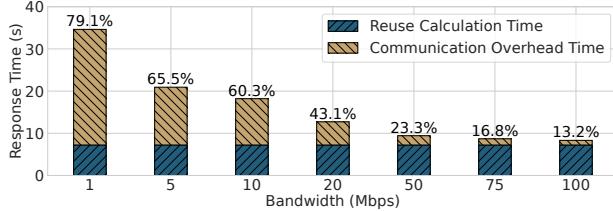


Figure 3: Communication overhead ratio comparison. The system first calculates the model to reuse and then transmits model weights to the mobile device.

2.1 Continuous Adaptation against Frequent Environment Shift

Environment shifts in video analytics are common, particularly in mobile systems. In the datasets we collected, we find that numerous environment shifts occur, lowering the system's performance. Compared to the teacher model, expert models generalize poorly against environment shifts. As Figure 1 and Figure 2 show, the accuracy degradation of the teacher model remains within 15% across different domains, while the expert model experiences up to a 52% accuracy drop. Therefore, during frequent environment shifts, continuous adaptations are needed to stabilize DNN accuracy.

However, the two adaptation strategies (*i.e.*, reuse and retraining) face distinct challenges. Reuse requires a trade-off between response speed and accuracy by making different choices of model selection within it while retraining needs more compute resources and cannot provide a responsive solution at once [4]. To guarantee DNN accuracy, most existing solutions [6, 34, 35, 73] completely offload these adaptations to the remote cloud server, leading to suboptimal response efficiency during environment shifts, due to mobile-cloud communication delays and cloud queuing delays [71]. As shown in Figure 3, the communication overhead can account for over 60% of the total response time when the bandwidth is 10 Mbps, highlighting the inadequacy of cloud-based strategies to handle frequent environment shifts. To improve the responsiveness, our intuition is to deploy partial actions to mobile devices and design a hierarchical adaptation system [44, 82].

2.2 Hierarchical Model Adaptation

The “hierarchy” can be interpreted from two orthogonal perspectives: First, model adaptation tasks can be defined hierarchically: model reuse, fine-tuning a few layers, and end-to-end retraining, with increasing adaptability but decreasing efficiency. Second, the model adaptation could happen collaboratively between mobile devices and the cloud for balanced response efficiency and adaptation effectiveness.

Table 1: Resource metrics of different tasks. Bandwidth is 10 Mbps and model weight size is 14 MB.

Task	Speed (FPS)	Total Time (s)	Memory (GB)
Inference	8.3	/	2.2
Onboard Fine-tune	9.7	120	2.6
Cloud Retrain	303	160	14.4
Onboard Reuse	/	0.47	/
Cloud Reuse	/	22.8	/

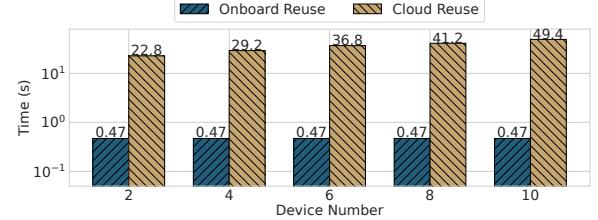


Figure 4: Onboard Reuse v.s. Cloud Reuse.

2.2.1 Mobile Fine-tuning vs. Cloud Retraining. The benefits of model reuse in reducing model retraining have been highlighted in RECL [34], while we further address the importance of lightweight model fine-tuning that only selectively updates a few layers of a DNN. It is more computationally efficient than end-to-end retraining and attains better adaptability than reuse when no suitable history expert model can be directly picked. As can be seen from Table 1, although onboard fine-tuning takes longer per iteration, its total update time is shorter due to the reduced training iterations with fewer data and the absence of communication and queuing delays encountered in cloud retraining tasks. Based on the extent of environment shift, the combination of reuse, fine-tuning, and retraining constitutes a more fine-grained action space for model adaptation.

2.2.2 Mobile Reuse vs. Cloud Reuse. The limited mobile storage and computation capacity have not been sufficiently explored in model adaptations, where model reuse can demonstrate its value. Intuitively, we make an analogy between mobile-vs-cloud resources and the hierarchical storage of computer systems. We have limited but fast-accessing resources (“cache”) on mobile devices, and abundant but distant resources (“main memory”) on the cloud server. From the storage aspect, saving a few frequent expert models on the mobile can save remote access delays. From the computation aspect, distributing lightweight fine-tuning on mobile devices while centralizing heavyweight retraining on the cloud can balance the response efficiency and processing throughput in expert model updates.

For example, in object detection tasks, a YOLOv5-s model (17M FLOPs) weights file is about 14MB, thus we can store multiple expert models on the mobile device. As shown in Table 1, if the expert model is cached onboard, we can directly load it in 0.47s, without waiting for the cloud dispatch, thus improving the overall response speed [81]. We further analyze the scalability of mobile device numbers between local reuse and cloud reuse. As Figure 4 shows, onboard retrieval exhibits magnitudes shorter response delays than cloud model retrieval, which gap is further pronounced when more mobile devices need to be served by the cloud.

Table 2: Examples of foundation model (FM)-based domain semantic discrimination.

Prompt	Identify the <i>location</i> , the <i>weather</i> , and the <i>time</i> in the image. Respond in three words in the order of location, weather, and time, separated by commas, all lowercase.			
Images				
Responses	<table border="1"> <tr> <td>street, rainy, night</td> <td>highway, snowy, daytime</td> <td>residential, clear, daytime</td> </tr> </table>	street, rainy, night	highway, snowy, daytime	residential, clear, daytime
street, rainy, night	highway, snowy, daytime	residential, clear, daytime		

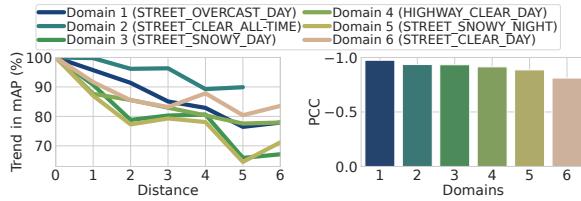


Figure 5: Reuse on similar domains. Pearson correlation coefficient (PCC) is calculated from the distance and mAP for each validation domain.

2.3 Semantic-Indexed Model Retrieval

One critical step to hierarchical adaptations is the efficient retrieval of the best candidate for the new domain from history expert models. Besides iterating over all models with a benchmark dataset, recent studies have employed unsupervised clustering [73] or maintained a gate network [34] for expert model selection. However, these learnable methods require frequent updates when the model set changes. We argue that they all overlook recognizable but critical semantic information of the deployed environments [24], which can be used to organize history expert models into a structured database, instead of a disorganized model zoo, to improve the model retrieval efficiency.

2.3.1 Domain Semantics Recognition. Domain semantics refer to the environmental attributes that describe and differentiate domains in environment shifts.¹ This includes factors like time, weather or location. These domain semantics can be easily recognized by general foundation models (FM) in a zero-shot way [8, 36, 70], which has been pre-trained on massive data and can be used for diverse downstream tasks. Existing FMs, like LLaVA [48], have been pre-trained to comprehend multi-modal input by connecting image input with a text prompt [51]. In Table 2, we use LLaVA to recognize the time, weather, and location of an image with a dedicated prompt. It achieves promising accuracy in discriminating domains. The computation for a 640×640 image takes 500ms on an NVIDIA RTX4090 GPU. We request human operators to provide a subset of domain semantic dimensions with value options as a one-time effort to facilitate expert model management, while other uninterpretable factors are implicitly handled in model continual learning.

¹The selection of attributes’ dimensions is orthogonal to this paper, which is hard to analyze quantitatively. We assume users provide parts of these parameters and the system can complete the others and construct the taxonomy offline using a few-shot validation dataset.

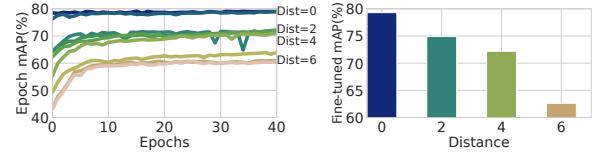


Figure 6: Fine-tuning on similar domains. Models from domains at varying distances are used as starting points for fine-tuning on the same dataset.

2.3.2 Domain Semantics Utilization. The semantic dimensions can build a hierarchical taxonomy and a distance space on different domains, which can be directly used to retrieve the fittest model candidate in fast adaptations. We hypothesize that the semantic-level distance can work as an indicator of cross-domain model reuse and retraining effectiveness. We conduct a few experiments to analyze the cross-domain gaps for empirical validation. Intuitively, two domains are similar if they share most attributes but differ in a few. If all attributes match, the domains are identical.

Direct Reuse: We use data from BDD100K [83] and a set of YOLOv5-s pre-trained models to perform object detection on more than 10 validation domains divided by three attributes “location”, “weather”, “time_of_day”, and record the mean Average Precision (mAP) as detection accuracy. We use a distance (introduced in §5.1 of MOCHA) as domain similarities. We list 6 domains of model performances in Figure 5. More similar domains tend to have higher cross-domain accuracy, and the highest accuracy is achieved in the training domain (*i.e.*, distance=0). The semantic similarity exhibits a strong negative correlation with model performance, with the overall average Pearson correlation coefficient (PCC) exceeding -0.82. With more shared attributes, domain distributions become closer or even overlapped. In the overlapped scenes, models from both domains produce similar outputs and close accuracy.

Lightweight Fine-tuning: We also test whether models from similar domains are more suitable for fast fine-tuning. We use the same data and models to conduct a fine-tuning task with fixed epochs while freezing all but a single layer of the model. In Figure 6, we fine-tune models from all domains on the “STREET_CLEAR_DAY” domain and test mAP on its validation data. The results show that more similar domains can also obtain faster convergence and better accuracy improvement after fine-tuning. Thus, semantic similarity can also be used as an indicator for cross-domain fine-tuning.

3 MOCHA Framework

3.1 Overall Architecture

As Figure 7 shows, MOCHA is a mobile-cloud collaborative model adaptation framework for continuous mobile video analytics with resilient prediction fidelity. Upon an environment shift, the system should calibrate the mobile-deployed model to recover high accuracy within a short response latency. We consider a cloud server with powerful compute and storage resources, and a set of distributed mobile devices with constrained compute and communication resources (e.g., NVIDIA Jetson Nano, TX2). MOCHA offloads low-latency adaptation to mobile devices through lightweight computations (*i.e.*, local reuse and fast fine-tuning), with background support from the cloud server through heavyweight computations (*i.e.*, end-to-end retraining).

Optimization Objective: The core tasks of continuous modal adaptation include “*detecting when the environment shift happens*” and “*adapting the mobile-deployed expert model*”, optimizing the responsiveness (*i.e.*, finish adaptation in a short time) and scalability (*i.e.*, support more mobile devices) of adaptations, under the constraints of mobile resources and mobile-cloud network delays. MOCHA performs environment shift detection on mobile devices, implements model-cache-based reuse and LoRA single-layer fine-tuning as immediate onboard reactions, and leaves accurate model retrieval from a full-fledged model database and end-to-end retraining on the cloud as backup strategies. MOCHA’s response efficiency benefits from its mobile-cloud collaborative nature on reuse and update strategies, as defined below.

- **Model Reuse:** If the exact or a similar domain has appeared in the past, we directly retrieve and use the previously trained model from the mobile model cache or the cloud model database as a response.
- **Model Update:** If no previous model can fit in the current domain, we update part or all of a model’s parameters through lightweight onboard fine-tuning or end-to-end retraining on the cloud server to respond with delays.

MOCHA’s performance highly relies on effective and efficient history model retrieval from past domains. MOCHA leverages meta-level domain semantics identified by a multi-modal foundation model, as indices to construct a domain taxonomy for real-time expert model retrieval. Below, we list the main mobile and cloud components in MOCHA.

3.1.1 Mobile Components. Besides regular model inference, the mobile device periodically detects the occurrence of environment shifts and performs fast onboard model reuse and fine-tuning, through the following four modules.

- **Executor:** It performs inference on the real-time video stream. Besides, a lightweight environment shift detector is integrated to detect potential distributional drifts.
- **Mobile Model Cache:** It stores a few expert models that could be reused shortly. At the meta-level, it keeps a copy of the semantic model taxonomy (synchronized with the cloud server) to assist in model selection.
- **Mobile Data Buffer:** Two data buffers are used to store the real-time data stream awaiting inference and the labeled data prepared for fine-tuning.

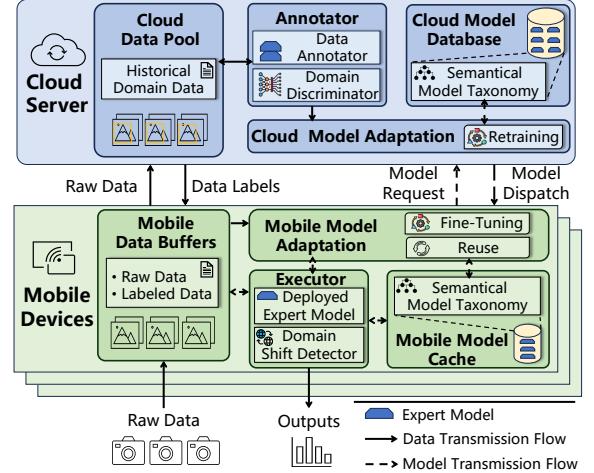


Figure 7: Architecture of MOCHA.

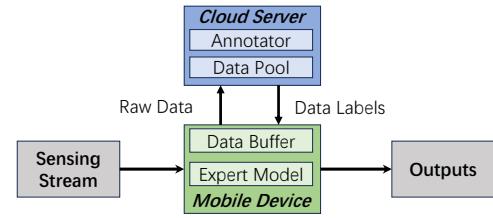


Figure 8: Data transmission flow.

- **Mobile Model Adaptation:** Its frontend performs direct model reuse or fine-tuning based on the mobile model cache. If needed, its backend interacts with the cloud for retrieval from the cloud model database or end-to-end retraining for ultimate adaptation.

3.1.2 Cloud Components. The cloud server stores history models collected from its connected mobile devices and performs computation tasks in the background to assist the mobile device components, with the following four modules.

- **Annotator:** A large-scale teacher model is deployed to annotate the mobile-uploaded samples and a multi-modal foundation model is used to recognize the meta-level domain semantics (*i.e.*, time, weather, location).
- **Cloud Model Database:** It stores expert models in historical domains based on predefined semantic dimensions, enabling efficient model retrieval. A taxonomy updating algorithm is designed along with retraining tasks.
- **Cloud Data Pool:** The labeled samples of historical domains are stored for future domain retraining.
- **Cloud Model Adaptation:** It conducts end-to-end retraining for new domains using data from the cloud data pool and dispatches the trained model to mobile devices.

3.2 Mobile-Cloud Collaboration Workflow

With the introduced mobile and cloud modules, we elaborate on their collaboration procedure during adaptations. A standard data transmission flow is shown in Figure 8. Although the frequency of environment shifts is much lower than the sensor sampling rate (e.g., 25 FPS), previous works [6, 34, 35] require continuous data uploading for cloud-based environment shift detection. In

contrast, MOCHA avoids data transmission during periods without environment shifts by hosting onboard environment shift detection.

This leads to two system states: during “*regular inference time*”, only inference tasks and environment shift detection are conducted; during “*potential shift time*”, continuous data uploading and annotation are launched to confirm actual environment shifts and to prepare data for adaptations. Following existing practices [6, 34], we set fixed-length *model-update windows* (by default, 30 seconds) to ensure adequate time for fine-tuning data collection on mobile devices and to provide a baseline for waiting times in specific system states.

As a mobile-oriented framework, most MOCHA actions are triggered and performed on the mobile, while the cloud passively serves the submitted mobile requests. Within the collaboration, the **mobile activities** include:

- (1) It performs regular inference with the deployed expert model on video frames and conducts environment shift detection based on extracted sample features (§4.1). If distribution drifts are detected, it enters potential shift time.
- (2) In potential shift time, it uploads subsampled frames to the cloud for (i) domain semantic discrimination to confirm the environment shift, and (ii) sample annotations for potential adaptations. If the cloud confirms no environment shift happens, the false alarm is resolved and the mobile quits potential shift time. Otherwise, the below adaptation actions are performed.
 - (a) It immediately performs model reuse from the most similar domain(s) to recover accuracy (§4.2) and evaluates the reused model with cloud-annotated samples. If an accuracy threshold is missed, a LoRA fine-tuning task starts accordingly (§4.3).
 - (b) After step (a), if the new domain has not appeared in the model taxonomy (*i.e.*, an unseen domain), its end-to-end training is needed. The mobile continuously uploads samples to the cloud while using the expert model from step (a) for inference. Once enough data is aggregated, the new model is trained and dispatched from the cloud to finish the adaptation.
 - (c) After the adaptation sequence, the mobile quits potential shift time and performs model replacement (§4.4) in the background for the mobile model cache.

To assist the mobile adaptations, the **cloud activities** in each time window include:

- (1) Upon receiving mobile-uploaded samples, it uses the teacher model to annotate task labels and uses the FM to discriminate domain semantics with dedicated prompts. It sends the domain semantics back and stores the annotated samples in its domain buffer.
- (2) It serves model retrieval requests from mobile devices by dispatching the expert model to them.
- (3) When one window ends, it updates the sample count of each domain and launches end-to-end retraining if enough data is accumulated.
- (4) If an end-to-end retraining is finished, the new expert model is stored in its model DB and indexed with domain semantics. The model taxonomy (§5.1) is accordingly updated and synchronized with all mobile devices.

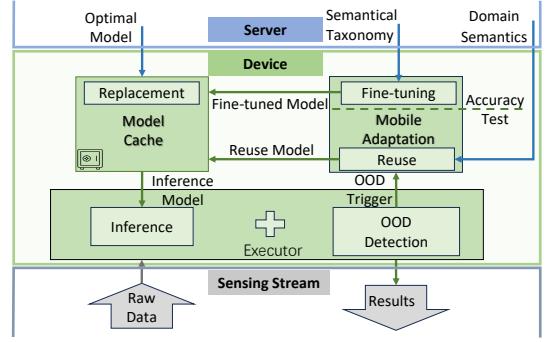


Figure 9: Responsive model adaptation design.

4 Mobile Model Adaptation

This section answers four questions related to mobile adaptation components: (1) how to detect environment shift? (2) how to load the optimal reuse model? (3) how to arrange onboard fine-tuning tasks? (4) how to update the cache for future adaptations? An overview of the mobile adaptation components is summarized in Figure 9.

4.1 Environment Shift Detection

The end-to-end environment shift detection is achieved through a two-step process. First, we perform onboard environment shift detection to avoid unnecessary mobile-cloud data transmission during regular inference time. Specifically, MOCHA applies a light-weight out-of-distribution (OOD) approach [41] to determine the likelihood of an environment shift without overconfidence bias. If an alarm is fired, we switch from regular inference time to potential shift time. Second, in potential shift time, the mobile device regularly uploads samples to the cloud to analyze the domain semantics to confirm the actual environment shift. If the environment shift happens, mobile model adaptation strategies are triggered.

We use a multi-dimensional Gaussian distribution to regress the training distribution. Upon a model training finishes, we choose features from a fixed layer, perform dimensionality reduction with pooling, and record the mean $\hat{\mu}_c$ and variance $\hat{\Sigma}$ of the training features. At runtime detection, the OOD score of a sample is calculated by feeding the extracted sample feature x_i into the equation to get $S(x_i)$. The environment shift indicator of a window $S_{win}(x)$ is based on the aggregated sample OOD scores, as shown below.

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} f(x_i), \quad \hat{\Sigma} = \frac{1}{N} \sum_c \sum_{i:y_i=c} f(x_i - \hat{\mu}_c)(f(x_i - \hat{\mu}_c))^T,$$

$$S(x_i) = \max_c [-(f(x_i) - \hat{\mu}_c)^T \hat{\Sigma}^{-1} (f(x_i) - \hat{\mu}_c)],$$

$$S_{win}(x) = \frac{1}{N_{win}} \sum_{0 \leq i < N_{win}} S(x_i),$$

where N_c is the sample count of class c , N is the training sample count, and N_{win} is the sample count in a window.

We use the OOD score of point “ $mean + k \times std$ ” as the threshold of environment shift detection, with k as a hyper-parameter. In §7.3 we select a k value that is relatively sensitive to distributional drift, to ensure a high recall on the environment shift. The false positive alarms fired by the mobile environment shift detector could be resolved by the cloud domain discriminator (*i.e.*, FM), without

initiating meaningless adaptations. If the cloud FM indicates no environment shift, we exit the potential shift time. Such collaborative environment shift detection well balances the environment shift detection efficiency and quality.

4.2 Mobile Model Reuse

Once an environment shift is confirmed on the cloud, MOCHA performs efficient model reuse as the immediate reaction. Technically, the fitness of a history expert model should be defined by its accuracy in the new domain. However, annotating samples for the new domain on the cloud and iterating over all history expert models can be time-consuming, we instead use the semantical proximity between domains as an efficient proxy of the model fitness, which is calculated by a distance measure defined on the domain semantical features recognized by the cloud foundation model (as detailed in §5.1). Since the semantical model taxonomy is synchronized on each mobile device, model lookup can be performed locally on each mobile device.

We first identify a “global optimal model” on the cloud model database that is expected to achieve the best reuse accuracy. If it is stored in the mobile model cache (*i.e.*, cache hit), we directly load it for reuse; otherwise if the “global optimal model” is not cached onboard (*i.e.*, cache miss), the reuse finishes in two steps. We identify another “local optimal model” from the mobile model cache for temporary reuse before the “global optimal model” is downloaded from the cloud model database. In cache hit cases, MOCHA eliminates the time of transmitting data and waiting for the cloud to identify the optimal model. Otherwise, it involves cloud-based model dispatch but still saves cloud queueing delays for environment shift detection. Overall, the mobile model cache and onboard environment shift detector optimize the responsiveness of MOCHA’s model reuse.

4.3 Mobile Model Fine-Tuning

Model reuse, although being most responsive, may lead to suboptimal adaptation quality in some cases. Thus, we further test the reused expert model with a subset of labeled samples annotated by the cloud. If the accuracy misses a threshold (by default 35% mAP in object detection), at the end of one window, we launch an onboard LoRA fine-tuning task with accumulated labeled samples in the background, without interrupting the inference task.

Even though mobile fine-tuning takes longer for each iteration than cloud retraining, it achieves completion faster overall as updating single-layer parameters requires magnitudes fewer training samples. Moreover, fine-tuning avoids cloud queueing delays, while retraining time could increase multiplicatively as more mobile devices are connected to the cloud server. In our ablation studies (§7.4), fine-tuning is shown to be one critical factor in optimizing the accuracy during the adaptation. Notably, MOCHA does not upload fine-tuned models to the cloud as it exhibits limited generalizability into other mobile devices.

4.4 Mobile Model Cache Replacement

Similar to how the CPU cache prefetches memory pages after a load, we proactively replace expert models in the mobile model cache after reuse to ensure a higher likelihood of cache hits in the next environment shift. Although future environment shift

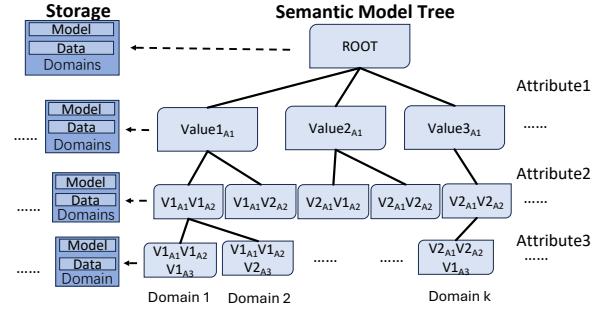


Figure 10: Semantic model taxonomy on the cloud.

patterns can be difficult to predict, we choose simple intuitions in cache replacement.

First, the recurrence of environment shift indicates locality in model reuse. For instance, in real-world driving, phenomena such as entering and exiting tunnels or merging onto and off highways are commonly observed. This indicates a pattern of frequent iteration between two domains. Thus, we store the unloaded expert model in the cache for potential use in the next fetch. Second, within a continuous video stream, the domain can not shift abruptly to semantically distant domains. Instead, similar domains are more likely to happen in the future. Therefore, we prefetch the expert model for the most similar domain to the current domain with the most shared attributes (*e.g.*, from “daytime_clear_city-street”) to “daytime_clear_highway” or “daytime_rainy_city-street”). The cache replacement happens in the background through a best-effort manner without strict deadlines or interrupting the inference task.

5 Cloud Model Adaptation

In addition to performing data annotations and domain discrimination upon receiving uploaded samples from mobile, and managing model dispatch in model reuse, the cloud is responsible for maintaining a model taxonomy and serving end-to-end retraining requests.

5.1 Semantic Model Taxonomy

Meta-level domain semantics are used as indices to perform efficient model retrieval and estimate the fitness of cross-domain expert model reuse. Our intuition is to organize extensive history expert models into a hierarchical taxonomy rather than a disorganized zoo, such that semantical comparison can be used to replace repetitive model evaluations during selection. Below we introduce how the domain semantics are organized and utilized.

We request the human operator to specify only the dimensions of semantic attributes in advance (*e.g.*, time, weather, and location) while leaving their values open². Based on these dimensions, MOCHA generates the domain semantic taxonomy as a tree as visualized in Figure 10, where each layer corresponds to an attribute. The child node is more concrete in domain semantics compared to its parent node. The leaf nodes specify values in all attributes and represent most fine-grained domains, while domains at non-leaf nodes cover all their child domains. Each node on the semantic model tree has an expert model in the cloud model DB and related

²The foundation model (FM) possesses the capability of recognizing their values along the given dimensions. Note the semantic dimensions are sorted in the decreasing order of their impact on the model performance, which could be profiled offline with a limited set of scenes.

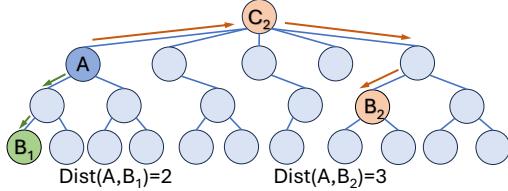


Figure 11: Distance algorithm examples.

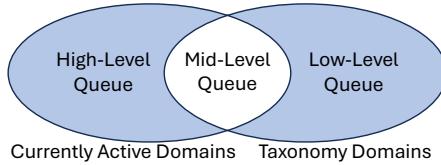


Figure 12: Multi-level queue.

labeled data in the cloud data pool. Besides, we use a meta-level taxonomy table to record the semantical model tree. It is synchronized between the cloud and all mobile devices for model retrieval.

A semantic distance function $Dist$ is defined to measure the similarity of two domains as an indicator of cross-domain expert model reuse accuracy. It also assists the expert model prefetch in the mobile model cache management. The domain semantical distance is defined below.

$$Dist(A, B) = \begin{cases} |layer(A) - layer(B)|, & A \in S_B \vee B \in S_A \\ Dist(A, C) + Dist(B, C), & A \notin S_B \wedge B \notin S_A \end{cases}$$

where A and B are the nodes of the tree, S_A means the subtree of A , C is the lowest common ancestor node of A and B . We empirically show its effectiveness in proxying the cross-domain performance in §7.3. Its computation is lightweight (Figure 11 as an example) compared to the deep neural network (DNN) based gate networks for expert model selection [34]. Meanwhile, with a new domain, the taxonomy table can be easily extended without time-consuming learning. Once enough samples are accumulated for a new domain, its model can be end-to-end trained and archived on the cloud. We add this new domain to the model taxonomy and synchronize the new taxonomy to all mobile devices.

5.2 Cloud Model Retraining

In most situations, it is impossible to prepare training data for all domains during the offline training, so we only train models for a subset of domains as the initial point. At runtime, as we aggregate enough data for a new domain, we initiate end-to-end training and dynamically expand the model taxonomy. Meanwhile, an existing domain may meet with plenty of new data shifting from the original data distribution. Although fast fine-tuning can partially resolve the in-domain environment shift in some cases, it raises the necessity of end-to-end retraining in other cases with significant shift³.

We store samples annotated by the teacher model and domain semantics recognized by the FM for new domains and existing domains currently under distribution drift. For storage efficiency, we set an upper limit on the sample count per domain (by default, 1000). A non-leaf node in the taxonomy consolidates multiple sub-domains, whose (re)training only begins when the node has data

³This is regarded as distributional drift caused by unexplainable factors where conventional continuous learning algorithms can fit in to tackle the catastrophic forgetting problem, which is beyond the scope of this paper.



Figure 13: Cloud server and mobile devices.

from at least two subdomains or a new subdomain has appeared. Moreover, the data for the non-leaf domain is sampled in a balanced manner between subdomains for better generalizability.

5.3 Retraining Task Scheduling.

With the number of mobile devices rising, the cloud faces many training tasks to run at a time and each task corresponds to a domain with a set of labeled frames. We treat this as a single-processor, non-preemptive scheduling problem [53, 87] and apply a multi-level queue scheduling algorithm on the cloud, as shown in Figure 12. Different queues are scheduled with priority scheduling, where retraining tasks in lower-level queues only start when higher-level queues are empty.

Three levels of queues are maintained, and different scheduling policies are applied within each level of task queues. At the end of each window, two meta-information—the *currently active domain* and *model accuracy in the last window*—are uploaded for each retraining task from mobile devices to determine the corresponding queues they belong to.

- **High-level queue:** Training tasks of *currently active domains* that have not appeared in the model taxonomy, where an expert model from a similar domain is temporally reused but with sub-optimal accuracy. Its tasks are scheduled with a First-In-First-Out (FIFO) policy.
- **Mid-level queue:** Retraining tasks of currently active domains that are already in the taxonomy. Its tasks are scheduled in the increasing order of *accuracy in the last window*.
- **Low-level queue:** Retraining tasks of domains in the model taxonomy but not currently active, scheduled with FIFO policy. Non-leaf nodes are in the low-level queue because they are only used for temporary reuse.

6 Implementation

We implemented MOCHA with 6,000 lines of Python code and implemented all neural networks under PyTorch [60]. The hardware and software setups are specified below.

Hardware Configurations. As shown in Figure 13, we use a workstation equipped with an Intel Xeon Gold 5420+ CPU, 128 GB memory, and an NVIDIA RTX 4090 GPU (24 GB RAM) as the cloud server. For mobile devices, we use multiple NVIDIA Jetson TX2, Jetson Nano, and Orange Pi (smartphone-class) boards [61]. NVIDIA Jetson devices are configured to MAXN mode to ensure stable performance. Orange Pi boards are set to maximum operating frequency, with RKNN-toolkit [62] boosting inference speed. Each mobile device runs MOCHA’s components alongside a separate input thread to simulate the video stream.

Software Implementation. We use wondershaper [14] to simulate different network conditions (by default, 10Mbps). We use

llava-v1.6-vicuna-7b as our FM to discriminate domain semantics. The network data transmission is implemented via TCP Socket APIs. We launch a socket server on the cloud with a fixed port. Each mobile device connects to the port and is allocated an ID stored on the cloud. When transmitting data, the socket API automatically adds its ID to the TCP packages and the cloud can correctly process data from different devices. For retrieval requests, a temporary socket connection is established to transmit model weights without an allocated ID. On the mobile side, model inference, model adaptation, and data transmission are executed in separate threads, and the model cache has 3 slots. On the cloud server, data annotation, socket connections, and database updates are managed as distinct threads.

7 Experiments

In this section, we report the evaluation results of MOCHA on two video-analytics tasks. We first present the methodology (§7.1) and end-to-end results (§7.2), then show micro-performance analysis (§7.3), ablation studies (§7.4) and mobile overhead (§7.5).

7.1 Methodology & Setup

7.1.1 Datasets. We evaluate MOCHA under 2 different data: domain sequences from BDD dataset [83], and the real-world datasets. **BDD** dataset consists of 100K 1280×720 one-minute-long colored videos collected with dashcams from diverse environments. Ten object classes, along with 108 domains and 3 semantic attributes, are included. We randomly generate 3 domain sequences, and in each domain sequence, we concatenate one video with another video from the same domain when a video ends and change the domain after a fixed number of videos like past frameworks [34, 73]. By continually changing domains, we simulate different environment shifts and evaluate MOCHA’s performance. For image classification, we create the dataset by cropping images from the BDD dataset, using YOLO model annotations to obtain the desired size ($1.4\times$ to the bounding box size) of images with both the domain background and target objects. In these domain sequences, we have the ground truth for when the environment shifts occur, with which we can precisely evaluate the performance of MOCHA’s components. The **real-world datasets** consist of videos collected from YouTube and videos collected from our dashcams. All videos are at least an hour long and include various driving environments. Their resolution is 1280×720 . We use the teacher model for labeling and the label classes are consistent with BDD. We run MOCHA on them to help us evaluate the performance when deployed in natural environments.

7.1.2 Models. We leverage YOLOv5-5.0 implementation for object detection, with YOLOv5-s and YOLOv5-x as expert and teacher models respectively.⁴ For image classification, we use ResNet implementation in torchvision [11], with ResNet18 and ResNet152 as expert and teacher models respectively. For semantic segmentation, we use DeepLabv3+MobileNet and DeepLabv3+ResNet101 as expert and teacher models respectively. We pre-train expert models of 12 random domains and store them in the model database according to configurations to simulate the real-time adaptations.

⁴The version of Python pre-installed on Nvidia Jetson TX2 is 3.6.9, and higher versions of YOLO are not supported in this environment.

7.1.3 Metrics. To evaluate the system accuracy during continuous adaptation, we compare inference results with labels annotated by the more accurate and more expensive teacher model and ground-truth labels manually annotated from BDD. We use mean Average Precision (mAP) for object detection tasks, classification accuracy (ACC.) for image classification tasks and Mean Intersection over Union (MIoU) for semantic segmentation tasks. In **end-to-end evaluations**, the following metrics are compared:

- **Recovery accuracy:** To evaluate the adaptation performance of dynamic frameworks during environment shifts instead of several static models with non-adaptation situations, we measure the model accuracy during the *recovery period* from detecting environment shifts to the completion of adaptation. For a fair comparison, we regard the time when the slowest framework, RECL, deploys a new model as the end of recovery, such that all compared frameworks share the same recovery period.
- **Response delay:** It is defined as the duration from when environment shifts are detected to when the first model adaptation action is performed.
- **Retraining time:** It is defined as the duration from starting an end-to-end retraining on the cloud to deploying the retrained model on the device.

7.1.4 Baselines. The compared baselines include:

- **No Adaptation:** Pre-trained models are deployed on edge devices without any adaptation.
- **ODIN** [73]: It detects and recovers from data drifts based on the similarity of video data and an autoencoder-based model selection. During runtime, the encoder generates an embedding vector for the input data, and ODIN selects the model corresponding to the cluster whose centroid is closest to the embedding vector. In our evaluation, we use the L2 distance as the similarity measure.
- **Ekya** [6]: It enables retraining and inference to co-exist on the edge device without model reuse. We utilize Ekya’s microprofiler and thief scheduler to manage model retraining jobs for a fair comparison. Despite advanced resource-sharing mechanisms, Ekya experiences out-of-band profiling overhead, which diminishes overall responsiveness.
- **RECL** [34]: It enables both retraining and reuse purely on the cloud. We implemented RECL as described in their paper (model selector, safety checker, teacher labeler, and update modules released in RECL). It maintains a gate network for initial model selection, followed by a safety check with a validation dataset. Despite its effective model reuse, RECL overlooks the network overhead and focuses much on the update of the gate network.

7.2 End-to-end Evaluation

7.2.1 Recovery Accuracy. This is the end-to-end quality assessment of all frameworks upon environment shifts. The recovery accuracy results of MOCHA and the baselines on domain sequences are presented in Figure 14, with 12 domains included and 3 attributes of semantics used. The recovery accuracy results of four adaptation frameworks on real-world datasets are presented in Figure 15. MOCHA consistently outperforms all baselines under different numbers of connected mobile devices and different datasets, by up to 6.8% in the object detection task, up to 4.2% in the image

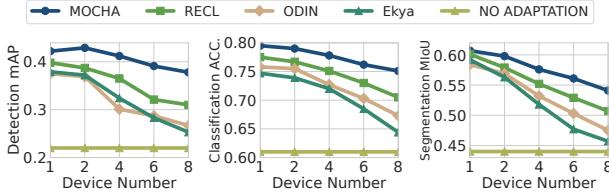


Figure 14: End-to-end evaluation of the recovery accuracy in three tasks.

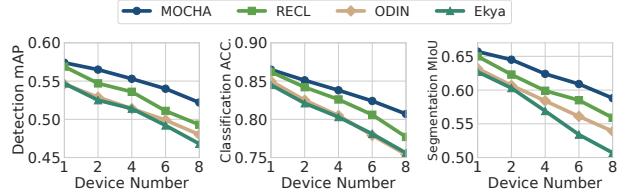


Figure 15: End-to-end evaluation on real-world datasets in three tasks.

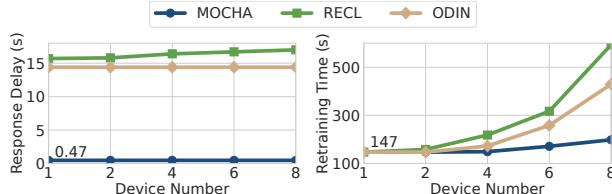


Figure 16: Response delay and retraining time comparison. Object detection task is used.

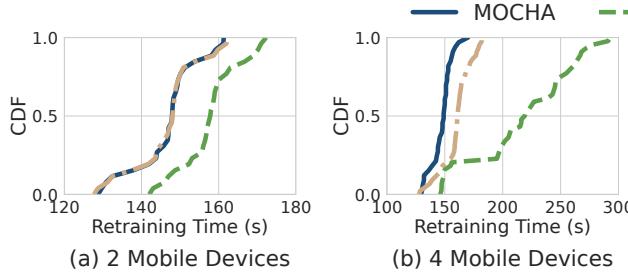


Figure 18: CDF of retraining time across different numbers of devices.

classification task, and up to 3.8% in the semantic segmentation task, due to its advanced hierarchical adaptation strategy, allowing faster reuse and efficient fine-tuning. In domain sequences, frameworks will encounter environment shifts with larger variation gaps compared to the real-world datasets we collected. In such more complex scenarios, MOCHA demonstrates a greater advantage over other baselines.

7.2.2 Response Efficiency. We next compare the response delay between different adaptation algorithms with model reuse in Figure 16 and Figure 17. MOCHA achieves the shortest response delay, 35.5× shorter than the second-best approach. Moreover, MOCHA exhibits better scalability in retraining time when more mobile devices are connected to the cloud. In Figure 18, MOCHA demonstrates superior robustness in retraining time compared to baselines. It experiences only a modest increase in retraining time as the number of devices grows, resulting in a 1.34× increase. In contrast, RECL shows a 4.03× increase due to the accumulation of ineffective retraining tasks, and ODIN shows a 2.91× increase because of its inefficiency in managing real-world data drift.

7.2.3 Adaptation Analysis. We use example traces to analyze the adaptation collaborations in MOCHA. The same domain sequences are evaluated for all frameworks with object detection, and 4 devices are connected to the cloud.

In the **reuse process** (Figure 17), upon an environment shift, MOCHA’s on-device reuse quickly recovers the accuracy through onboard model reuse, while both ODIN and RECL need to wait for

the dispatched model from the cloud, creating delays in response. “Cache Hit” and “Cache Miss” in MOCHA only differ in whether the expert model for the new domain is cached onboard. “MOCHA Cache Hit” finishes with onboard model reuse, providing the best overall accuracy, while “MOCHA Cache Miss” partially resolves with onboard cache and goes through the second reuse when the best expert model is dispatched, which still surpasses RECL on the recovery accuracy. Although ODIN finishes the check and dispatch quickly, it leads to suboptimal recovery accuracy due to the less effective embedding manifold.

In Figure 19, we plot the inference accuracy starting from an empty cloud model database, including MOCHA, ODIN, RECL, and the Oracle framework that has sufficient cloud computing resources and bandwidth for multiple retraining tasks. The system ingests 3 hours of video featuring 15 domains. MOCHA exhibits a rapid increase in accuracy due to its superior semantic taxonomy and fine-tuning mechanisms, whereas RECL shows slower improvement. Both MOCHA and RECL can approach Oracle’s performance given enough time to expand the model database, while ODIN falls short for a less effective embedding manifold.

7.3 Micro Experiments

7.3.1 OOD performance. We compare the precision and recall under different k of the boundary point in environment shift detection (§4.1) to choose the best threshold and test the OOD detection accuracy with the data in §7.2. In Figure 20, when $k \geq 0.4$, the

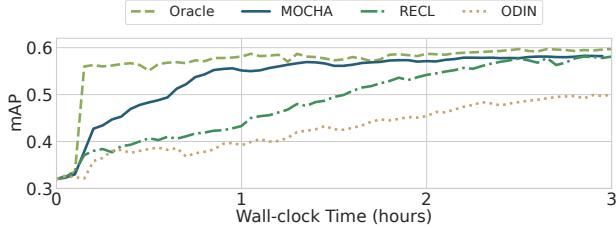


Figure 19: Practical adaptation example over time.

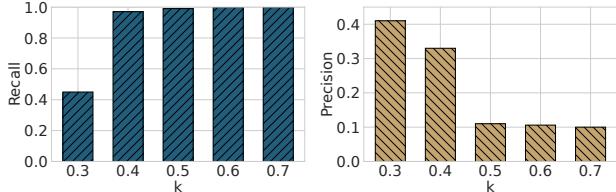
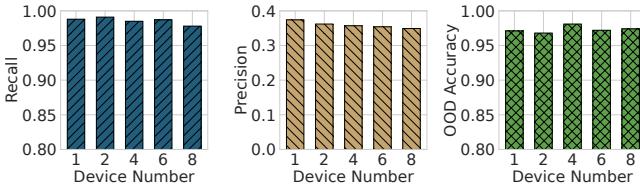
Figure 20: OOD threshold performances by the hyper-parameter k .

Figure 21: Practical OOD performances on real-world data. OOD accuracy is measured on the §7.2 data with ground truth for environment shifts.

recall exceeds 0.95, which means it is inclusive in detecting potential environment shifts. With k increases, the precision decreases to 0.1, which means OOD triggers false positives without actual environment shifts and MOCHA discriminates these with FM. Thus, “ $k = 0.4$ ” turns out to be the best configuration. Using the ground truth for when the environment shift occurs, we evaluate the OOD accuracy with the threshold “ $k = 0.4$ ” in Figure 21. The results show that for all environment shifts, an average of 96% are correctly raised in MOCHA across different numbers of devices, as detailed in the OOD accuracy. The recall and precision values are consistent with those observed in the previous threshold test.

7.3.2 Adaptation ratio vs. Mobile device number As more mobile devices are connected, the adaptation ratios for reuse, fine-tuning, and retraining change, as illustrated in Figure 22. With more devices, each device can leverage models in the model DB trained from other devices. This boosts the probability of reuse while reducing the need for fine-tuning and retraining, enhancing the scalability of MOCHA.

7.4 Ablation Studies

7.4.1 Cloud Retraining Scheduler. When handling multiple retraining tasks on the cloud, the scheduler performs better using a “multi-level queue (MLQ)” algorithm compared to a simple First-In-First-Out (FIFO) approach. We show the accuracy ratios (MLQ/FIFO) in Figure 23. With an increasing number of devices, more retraining tasks are parallelized, and the scheduler can optimize task ordering, leading to a better overall performance by up to 1.23 \times in object detection and 1.12 \times in image classification.

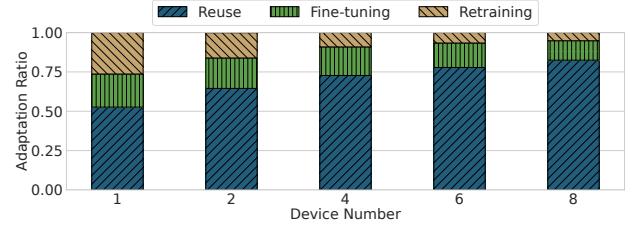


Figure 22: Adaptation ratio changes across different numbers of devices.

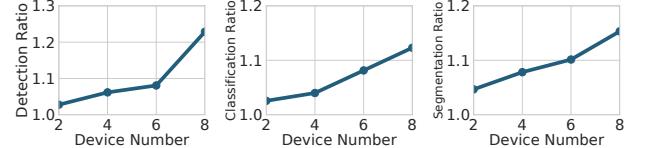


Figure 23: Cloud retraining scheduler comparison. Both ratios represent the comparison of Multi-Level Queue (MLQ) over First-In-First-Out (FIFO).

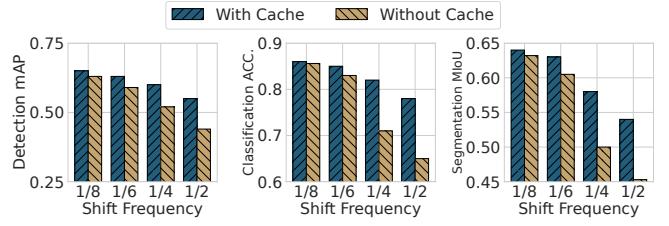


Figure 24: Impact of mobile model cache. Shift frequency represents the reciprocal of the environment shift period, measured in the number of windows.

7.4.2 Mobile model cache. In Figure 24, we compare the MOCHA’s accuracy with and without the mobile model cache in sequences including frequent environment shifts. The sequences generate environment shifts of existing domains in model DB with fixed numbers of windows and we only launch reuse in MOCHA. Eliminating the model cache, MOCHA experiences an accuracy drop of up to 11% and 13% respectively in two tasks. The average reuse time increases to 15.2s and is slightly influenced by the queuing delay on the cloud, which makes MOCHA less competitive in responsiveness. It confirms that the device model cache is the key factor in optimizing the responsiveness of MOCHA.

7.4.3 Model semantic-based selection and fine-tuning. In a model switch process, the traditional approach involves mobile-cloud collaborative model selection and model reuse with a gate network. In contrast, MOCHA implements all model selection, reuse and fine-tuning at the edge. Thus, it is not fair to directly compare MOCHA’s lightweight semantic taxonomy with the gate network in model selection. To illustrate the contributions of each step within MOCHA, we compare the performance of four methods: MOCHA, RECL, NoFT(MOCHA without fine-tuning), and NoRU (Ekyu without reuse). Results are presented in Figure 25. It is evident that NoRU, utilizing only lightweight model selection, does not have an accuracy advantage over RECL (though it demonstrates significant improvements in responsiveness and scalability). However, the incorporation of reuse allows NoFT to achieve better performance than NoRU. Once fine-tuning is included, MOCHA surpasses RECL

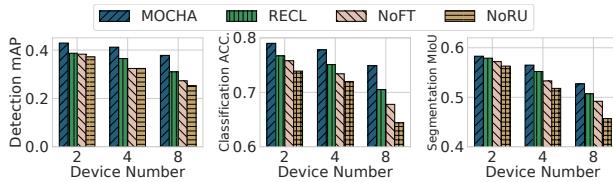


Figure 25: Impact of mobile fine-tuning and mobile selection. Both MOCHA and NoFT use the semantic taxonomy for model selection while RECL uses a gate network. NoFT lacks fine-tuning in MOCHA. NoRU has no model selection or reuse.

in accuracy as well. This is because we are willing to accept a less precise model selection method than the gate network in exchange for responsiveness and lightweight performance while fine-tuning can ensure the accuracy of MOCHA.

7.5 Mobile Overhead

We measure the inference performance and mobile overhead of MOCHA, RECL, and the “Inference Only” baseline in Table 3. Compared to RECL, during most non-domain-shifting periods without adaptations (without Adap.), MOCHA consumes less power by 7% and reaches faster speed by 19% due to the reduced need for frequent data transfers, but extra memory by 10% because of detection. In potential shift time with no fine-tuning, MOCHA consumes the same overhead as RECL, while its power consumption and memory consumption increase by 2.4× and 1.78× respectively, and its inference speed decreases 32% during the fine-tuning (with Adap.). The resource overhead of MOCHA is generally within an acceptable range.

8 Related Work

Video Analytics System. Video analytics systems aim to achieve high inference accuracy while minimizing energy use and response time through techniques like model distillation [6, 31, 35, 43], architecture pruning [57, 58, 76, 79], configuration adaptation [25, 28, 55, 84, 85], frame selection [12, 46] and DNN feature reusing [18, 23, 27, 78]. The closest to MOCHA is model distillation, where lightweight models are generated to perform on a specific video scene [30, 31, 40]. It designs strategies to choose suitable experts based on model retraining and model selection. Retraining techniques train the lightweight models on the latest video frames [13, 35, 56] or on the most relevant images from the training set [65]. Selection techniques maintain and then select a model from a collection of history models [73] or a cascade of models with increasing capacities [7, 65]. Unlike traditional methods, MOCHA offloads tasks to devices, improving responsiveness and scalability with an on-device model cache and fine-tuning.

Model Selection under data drifts. Model selection in a collection of expert models has gained considerable attention, particularly in Mixture-of-Experts (MoE) and autoencoders (AE). Traditional MoE methods [37, 38, 63, 64, 68], which rely on a gate network, reduce compute costs but necessitate enough memory for loading models and frequent retraining for the gate network. While AE techniques [1, 73, 74] project input data to a latent space and map new models to a region in the latent space with limited retraining but perform poorly facing environment shifts. In contrast, MOCHA

Table 3: Mobile inference speed and overhead

Methods	Speed (ms/img)	Throughput (FPS)	Memory (GB)	Power (W)
Inference Only	181	5.5	1.8	5.37
RECL	243	4.1	2.0	6.15
MOCHA	no Adaptation	208	4.9	2.2
	with Adaptation	354	2.8	10.96

employs an FM and a semantics-based model taxonomy for on-device selection, excelling in few-shot or zero-shot scenarios with minimal updates (§2.3.2).

Continuous learning on the devices. In ML literature, CL for mobile devices has been explored in notable works [19, 39] in trade-offs between performance, storage, memory, and compute costs. Recent studies [6, 54] highlight its benefits, motivating offloading adaptations to leverage their natural high response advantages. In MOCHA, on-device reuse and fine-tuning ensure responsiveness and scalability, aligning with our research objectives.

9 Conclusion

With cloud offloading being the mainstream solution for continuous mobile video analytics against environment shifts, this paper demonstrated that hierarchical model adaptation through mobile-cloud collaboration offers significant potential in optimizing the responsiveness of adaptation, which proves promising in autonomous driving. We presented MOCHA, a mobile-cloud collaborative continuous model adaptation framework, which organically integrates efficient onboard model reuse and fine-tuning with backend model retrieval and retraining on the cloud, achieving improved mobile responsiveness and cloud scalability in our evaluations.

Still, our current work has limited optimization for replacement strategies of the mobile model cache. And model management on the cloud with numerous devices connected can be further developed to reduce memory overhead and adaptation delays. We hope our findings can stimulate further research into harnessing the full potential of the synergy between model reuse and model retraining in mobile video analytics systems.

Acknowledgments

This work was sponsored in part by the National Key R&D Program of China (No. 2022ZD0119100), in part by China NSF grant No. 62472278, 62025204, 62432007, 62441236, 62332014, and 62332013, in part by Alibaba Group through Alibaba Innovation Research Program, and in part by Tencent Rhino Bird Key Research Project. This work was partially supported by SJTU Kunpeng & Ascend Center of Excellence. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

References

- [1] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.
- [2] K. Apicharttrisorn, J. Chen, V. Sekar, A. Rowe, and S. V. Krishnamurthy. Breaking edge shackles: Infrastructure-free collaborative mobile augmented reality. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 1–15, New York, NY, USA, 2023. Association for Computing Machinery.
- [3] H. Bao, Z. Zhou, J. Xie, Q. Huang, F. Xu, and X. Chen. COUPLE: Accelerating Video Analytics on Heterogeneous Mobile Processors. Association for Computing Machinery, New York, NY, USA, 2023.
- [4] R. Barjami, A. Miele, and L. Mottola. Intermittent inference: Trading a 1% accuracy loss for a 1.9x throughput speedup. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, Hangzhou, China, November 4–7, 2024, pages 647–660. ACM, 2024.
- [5] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune, and N. Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.
- [6] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica. Ekyu: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 119–135, 2022.
- [7] J. Cao, R. Hadidi, J. Arulraj, and H. Kim. Thia: Accelerating video analytics using early inference and fine-grained query planning. *arXiv preprint arXiv:2102.08481*, 2021.
- [8] Q. Cao, H. Xue, T. Liu, X. Wang, H. Wang, X. Zhang, and L. Su. mmclip: Boosting mmwave-based zero-shot har via signal-text alignment. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, page 184–197, New York, NY, USA, 2024. Association for Computing Machinery.
- [9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.
- [10] Y. Chen, C. Hu, T. Kimura, Q. Li, S. Liu, F. Wu, and G. Chen. Semicmt: Contrastive cross-modal knowledge transfer for iot sensing with semi-paired multi-modal signals. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(4), Nov. 2024.
- [11] T. Contributors. Torchvision: Pytorch's computer vision library, 2016–. Accessed: 2024-06-30.
- [12] M. Dasari, K. Kahatapitiya, S. R. Das, A. Balasubramanian, and D. Samaras. Swift: Adaptive video streaming with layered neural codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 103–118, 2022.
- [13] Y. Deng, S. Yue, T. Wang, G. Wang, J. Ren, and Y. Zhang. Fedinc: An exemplar-free continual federated learning framework with small labeled data. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, SenSys '23, page 56–69, New York, NY, USA, 2024. Association for Computing Machinery.
- [14] J. A. Donenfeld and contributors. Wondershaper. <https://github.com/magnifico/wondershaper>, 2002–2024. A script to limit bandwidth of a network interface.
- [15] C. Fang, S. Liu, Z. Zhou, B. Guo, J. Tang, K. Ma, and Z. Yu. Adashadow: Responsive test-time model adaptation in non-stationary mobile environments. *arXiv preprint arXiv:2410.08256*, 2024.
- [16] M. Gao, X. Tong, J. Chen, Y. Chen, F. Xiao, and J. Han. Eternity in a second: Quick-pass continuous authentication using out-ear microphones. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, Hangzhou, China, November 4–7, 2024, pages 675–688. ACM, 2024.
- [17] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu. Model compression with adversarial robustness: A unified optimization framework. *Advances in Neural Information Processing Systems*, 32, 2019.
- [18] P. Guo, B. Hu, and W. Hu. Mistify: Automating {DNN} model porting for {On-Device} inference at the edge. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 705–719, 2021.
- [19] T. L. Hayes and C. Kanan. Online continual learning for embedded devices. *arXiv preprint arXiv:2203.10681*, 2022.
- [20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [21] Y. He, C. Bian, J. Xia, S. Shi, Z. Yan, Q. Song, and G. Xing. Vi-map: Infrastructure-assisted real-time hd mapping for autonomous driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, New York, NY, USA, 2023. Association for Computing Machinery.
- [22] Y. He, L. Ma, J. Cui, Z. Yan, G. Xing, S. Wang, Q. Hu, and C. Pan. Automatch: Leveraging traffic camera to improve perception and localization of autonomous vehicles. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 16–30, New York, NY, USA, 2023. Association for Computing Machinery.
- [23] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, 2018.
- [24] I. K. Jain, S. M. M., and D. Bharadia. Commrad: Context-aware sensing-driven millimeter-wave networks. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, Hangzhou, China, November 4–7, 2024, pages 633–646. ACM, 2024.
- [25] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling video analytics systems to large camera deployments. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, pages 9–14, 2019.
- [26] H. Ji and P. Zhou. Advancing ppb-based continuous blood pressure monitoring from a generative perspective. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, Hangzhou, China, November 4–7, 2024, pages 661–674. ACM, 2024.
- [27] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger. Mainstream: Dynamic {Stem-Sharing} for {Multi-Tenant} video processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 29–42, 2018.
- [28] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*, pages 253–266, 2018.
- [29] G. Jocher, A. Stoken, J. Borovec, A. Chaurasia, L. Changyu, A. Hogan, J. Hajek, L. Diaconu, Y. Kwon, Y. Defretin, et al. ultralytics/yolov5: v5_0-yolov5-p6_1280 models, aws, supervise, ly and youtube integrations. *Zenodo*, 2021.
- [30] D. Kang, P. Bailis, and M. Zaharia. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046*, 2018.
- [31] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*, 2017.
- [32] D. Kara, T. Kimura, Y. Chen, J. Li, R. Wang, Y. Chen, T. Wang, S. Liu, and T. Abdelzaher. Phymask: An adaptive masking paradigm for efficient self-supervised learning in iot. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, page 97–111, New York, NY, USA, 2024. Association for Computing Machinery.
- [33] D. Kara, T. Kimura, Y. Chen, J. Li, R. Wang, Y. Chen, T. Wang, S. Liu, and T. F. Abdelzaher. Phymask: An adaptive masking paradigm for efficient self-supervised learning in iot. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, Hangzhou, China, November 4–7, 2024, pages 97–111. ACM, 2024.
- [34] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl. {RECL}: Responsive {Resource-Efficient} continuous learning for video analytics. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 917–932, 2023.
- [35] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh. Real-time video inference on edge devices via adaptive model streaming. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4572–4582, 2021.
- [36] T. Kimura, J. Li, T. Wang, D. Kara, Y. Chen, Y. Hu, R. Wang, M. Wigness, S. Liu, M. Srivastava, S. Diggavi, and T. Abdelzaher. On the efficiency and robustness of vibration-based foundation models for iot sensing: A case study. In *2024 IEEE International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things (FMSyS)*, pages 7–12, 2024.
- [37] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd international conference on Machine learning*, pages 449–456, 2005.
- [38] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.
- [39] Y. D. Kwon, J. Chauhan, A. Kumar, P. H. HKUST, and C. Mascolo. Exploring system performance of continual learning for mobile and embedded sensing applications. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 319–332. IEEE, 2021.
- [40] J. Lee, P. Wang, R. Xu, V. Dasari, N. Weston, Y. Li, S. Bagchi, and S. Chaterji. Benchmarking video object detection systems on embedded devices under resource contention. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 19–24, 2021.
- [41] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.
- [42] A. Li, J. Sun, X. Zeng, M. Zhang, H. Li, and Y. Chen. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, page 42–55, New York, NY, USA, 2021. Association for Computing Machinery.
- [43] X. Li, Y. Li, Y. Li, T. Cao, and Y. Liu. Flexnn: Efficient and adaptive DNN inference on memory-constrained edge devices. In W. Shi, D. Ganesan, and N. D. Lane, editors, *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom 2024, Washington D.C., DC, USA, November 18–22, 2024, pages 709–723. ACM, 2024.

- [44] Y. Li, L. Liu, H. Li, W. Liu, Y. Chen, W. Zhao, J. Wu, Q. Wu, J. Liu, and Z. Lai. Stable hierarchical routing for operational LEO networks. In W. Shi, D. Ganesan, and N. D. Lane, editors, *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom 2024, Washington D.C., DC, USA, November 18–22, 2024*, pages 296–311. ACM, 2024.
- [45] Y. Li, J. Lv, H. Lin, Y. Gao, and W. Dong. Combating BLE weak links with adaptive symbol extension and dnn-based demodulation. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems, SenSys 2024, Hangzhou, China, November 4–7, 2024*, pages 619–632. ACM, 2024.
- [46] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. 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*, pages 359–376, 2020.
- [47] N. Ling, X. Huang, Z. Zhao, N. Guan, Z. Yan, and G. Xing. Blastnet: Exploiting duo-blocks for cross-processor real-time dnn inference. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, SenSys '22*, page 91–105, New York, NY, USA, 2023. Association for Computing Machinery.
- [48] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.
- [49] S. Liu, T. Wang, H. Guo, X. Fu, P. David, M. Wigness, A. Misra, and T. Abdelzaher. Multi-view scheduling of onboard live video analytics to minimize frame processing latency. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 503–514, 2022.
- [50] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher. Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading. In *Proceedings of the 30th ACM International Conference on Multimedia, MM '22*, page 3035–3044, New York, NY, USA, 2022. Association for Computing Machinery.
- [51] Y. Lu, D. Ding, H. Pan, Y. Fu, L. Zhang, F. Tan, R. Wang, Y. Chen, G. Xue, and J. Ren. M3cam: Extreme super-resolution via multi-modal optical flow for mobile cameras. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems, SenSys 2024, Hangzhou, China, November 4–7, 2024*, pages 744–756. ACM, 2024.
- [52] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2507–2516, 2019.
- [53] G. Lv, Q. Wu, Y. Liu, Z. Li, Q. Tan, F. Yang, W. Chen, Y. Ma, H. Guo, Y. Chen, and G. Xie. Chorus: Coordinating mobile multipath scheduling and adaptive video streaming. In W. Shi, D. Ganesan, and N. D. Lane, editors, *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom 2024, Washington D.C., DC, USA, November 18–22, 2024*, pages 246–262. ACM, 2024.
- [54] X. Ma, S. Jeong, M. Zhang, D. Wang, J. Choi, and M. Jeon. Cost-effective on-device continual learning over memory hierarchy with miro. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.
- [55] Z. Meng, T. Wang, Y. Shen, B. Wang, M. Xu, R. Han, H. Liu, V. Arun, H. Hu, and X. Wei. Enabling high quality {Real-Time} communications with adaptive {Frame-Rate}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1429–1450, 2023.
- [56] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian. Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 3573–3582, 2019.
- [57] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali. Gemel: Model merging for {Memory-Efficient} {Real-Time} video analytics at the edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 973–994, 2023.
- [58] T. Pan, K. Liu, X. Wei, Y. Qiao, J. Hu, Z. Li, J. Liang, T. Cheng, W. Su, J. Lu, et al. {LuoShen}: A {Hyper-Converged} programmable gateway for {Multi-Tenant} {Multi-Service} edge clouds. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 877–892, 2024.
- [59] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
- [60] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc., 2019.
- [61] O. Pi. Orange pi 5b board. <http://www.orangepi.cn>, 2025. Accessed: 2025-02-15.
- [62] raul.rao. Rknn toolkit. <https://github.com/rockchip-linux/rknn-toolkit>, 2023. Accessed: 2025-02-15.
- [63] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.
- [64] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- [65] H. Shen, S. Han, M. Philipose, and A. Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3646–3654, 2017.
- [66] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang. Vips: real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, MobiCom '22*, page 133–146, New York, NY, USA, 2022. Association for Computing Machinery.
- [67] S. Shi, N. Ling, Z. Jiang, X. Huang, Y. He, X. Zhao, B. Yang, C. Bian, J. Xia, Z. Yan, R. W. Yeung, and G. Xing. Soar: Design and deployment of A smart roadside infrastructure system for autonomous driving. In W. Shi, D. Ganesan, and N. D. Lane, editors, *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom 2024, Washington D.C., DC, USA, November 18–22, 2024*, pages 139–154. ACM, 2024.
- [68] Y. Shi, B. Paige, P. Torr, et al. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *Advances in neural information processing systems*, 32, 2019.
- [69] V. Sivaraman, P. Karimi, V. Venkatapathy, M. Khani, S. Fouladi, M. Alizadeh, F. Durand, and V. Sze. Gemini: Practical and robust neural compression for video conferencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 569–590, Santa Clara, CA, Apr. 2024. USENIX Association.
- [70] T. Srivastava, P. Khanna, S. Pan, P. Nguyen, and S. Jain. Unvoiced: Designing an ILM-assisted unvoiced user interface using earables. In J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, editors, *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems, SenSys 2024, Hangzhou, China, November 4–7, 2024*, pages 784–798. ACM, 2024.
- [71] B. Sudharsan, D. Sheth, S. Arya, F. Rollo, P. Yadav, P. Patel, J. G. Breslin, and M. I. Ali. Elastic: Elastic quantization for communication efficient collaborative learning in iot. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, SenSys '21*, page 382–383, New York, NY, USA, 2021. Association for Computing Machinery.
- [72] J. Sun, A. Li, L. Duan, S. Alam, X. Deng, X. Guo, H. Wang, M. Gorlatova, M. Zhang, H. Li, and Y. Chen. Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, SenSys '22*, page 106–119, New York, NY, USA, 2023. Association for Computing Machinery.
- [73] A. Suprem, J. Arulraj, C. Pu, and J. Ferreira. Odin: Automated drift detection and recovery in video analytics. *Proceedings of the VLDB Endowment*, 13(11).
- [74] T. Wang, J. Li, R. Wang, D. Kara, L. Wertheimer, A. Viros i Martin, R. Ganti, M. Srivatsa, and T. Abdelzaher. Sudokusens: Enhancing deep learning robustness for iot sensing applications using a generative approach. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems, SenSys '23*, page 15–27, New York, NY, USA, 2024. Association for Computing Machinery.
- [75] M. Wong, M. Ramamujam, G. Balakrishnan, and R. Netravali. MadEye: Boosting live video analytics accuracy with adaptive camera configurations. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 549–568, Santa Clara, CA, Apr. 2024. USENIX Association.
- [76] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10734–10742, 2019.
- [77] T. Wu, Y. Dong, Y. Xiao, J. Wei, F. Wan, and C. Song. Vision-based, low-cost, soft robotic tongs for shareable and reproducible tactile learning. In *International Conference on Advanced Robotics and Mechatronics, ICARM 2024, Tokyo, Japan, July 8–10, 2024*, pages 388–393. IEEE, 2024.
- [78] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 129–144, 2018.
- [79] R. Xu, R. Kumar, P. Wang, P. Bai, G. Meghanath, S. Chaterji, S. Mitra, and S. Bagchi. Approxnet: Content and contention-aware video object classification system for embedded clients. *ACM Transactions on Sensor Networks (TOSN)*, 18(1):1–27, 2021.
- [80] Y. Xu, Z. Liu, X. Fu, S. Liu, F. Wu, and G. Chen. Flex: Adaptive task batch scheduling with elastic fusion in multi-modal multi-view machine perception. In *2024 IEEE Real-Time Systems Symposium (RTSS)*, pages 294–307. IEEE, 2024.
- [81] Z. Xue, Y. Song, Z. Mi, L. Chen, Y. Xia, and H. Chen. Powerinfer-2: Fast large language model inference on a smartphone. *arXiv preprint arXiv:2406.06282*, 2024.
- [82] C. Yang, J. Li, R. Wang, S. Yao, H. Shao, D. Liu, S. Liu, T. Wang, and T. F. Abdelzaher. Hierarchical overlapping belief estimation by structured matrix factorization. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 81–88, 2020.
- [83] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

- [84] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 236–252, 2018.
- [85] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and {Delay-Tolerance}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, 2017.
- [86] Z. Zhao, N. Ling, N. Guan, and G. Xing. Miriam: Exploiting elastic kernels for real-time multi-dnn inference on edge gpu. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, SenSys '23, page 97–110, New York, NY, USA, 2024. Association for Computing Machinery.
- [87] J. Zheng, Z. Li, F. Qian, W. Liu, H. Lin, Y. Liu, T. Xu, N. Zhang, J. Wang, and C. Zhang. Rethinking process management for interactive mobile systems. In W. Shi, D. Ganesan, and N. D. Lane, editors, *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom 2024, Washington D.C., DC, USA, November 18–22, 2024*, pages 215–229. ACM, 2024.