

FaaS Learner: Resource-Efficient Edge Video Analytics via Correlation-aware Multi-model Continual Learning

Tianen Liu, Borui Li, *Member, IEEE*, Weilong Wang, Yun Cheng, Shuai Wang, *Member, IEEE*, Tian He, *Fellow, IEEE*, and Yunhao Liu, *Fellow, IEEE*

Abstract—Video analytics services utilizing multiple deep neural network models (DNNs) are increasingly being adopted in various edge intelligence applications. To avoid the accuracy reduction caused by data drift, existing works leverage continual learning that directly retrains DNN models in multi-model applications. According to our preliminary experiments, we observe that existing continual learning frameworks neglect the *accuracy correlation* among multiple DNN models and the opportunity to harvest idle resources during model retraining, leading to considerable resource inefficiency. In this paper, we propose FaaSLearner, an algorithm-system co-designed continual learning framework for multi-model video analytics that leverages the agile and fine-grained resource management of serverless computing. Specifically, FaaSLearner proposes the *correlation-aware retraining planning* to analyze the multi-model correlation, and then selectively retrain DNN models with the greatest in accuracy gain. In addition, FaaSLearner proposes the *resource-efficient retraining scheduling* to accurately trigger the continual learning, and then harvest the keep-alive periods for retraining tasks without interfering with normal inference. We evaluate FaaSLearner with four common edge video analytics applications with the Azure public dataset. Extensive experiments show that, FaaSLearner improves the average application accuracy up to 31.9% over representative baselines, and harvests more than $2.51\times$ idle memory resources for retraining.

Index Terms—Video Analytics, Continual Learning, Serverless Computing.

I. INTRODUCTION

Deep neural network (DNN) based video analytics is a key enabler of various modern edge applications such as autonomous driving [1] and embodied intelligence [2]. Moreover, the vast collection of publicly available DNN models via HuggingFace [3] and Kaggle [4] enables developers to easily build *multi-model* edge video analytics applications to meet the diverse requirements of users [5]. These applications structure multiple DNN models in a directed acyclic graph to achieve complex tasks at the edge. As illustrated in Fig. 1, a real-world video surveillance application [6] comprises multiple branches of deep learning components, which are interconnected to locate and classify objects within videos.

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

Yun Cheng is with the Swiss Data Science Center, ETH Zurich, Zurich, Switzerland (Email: chengyu@ethz.ch).

Yunhao Liu is with the Global Innovation Exchange, Tsinghua University, Beijing 100084, China (e-mail: yunhaoliu@gmail.com).

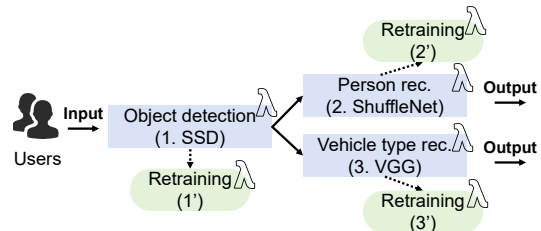


Fig. 1: Illustration of a typical multi-model video analytics application at the edge, including object detection, person recognition, and vehicle type recognition.

Due to the diversity and ever-changing nature of real-world edge environments, edge video analytics application suffers from the impact of *data drift* [7], where the feature distribution of newly submitted data differs from training data. For instance, data drift arises when the ambient lighting conditions [1], weather [8], or even object appearance [9] change in the requests. Pre-trained models in the multi-model application cannot handle the occurrence of data drift, leading to a substantial decrease in analytics accuracy.

To address the data drift issue at the edge, existing works [1], [7], [8] leverage continual learning, which retrains DNN models affected by data drift for overall accuracy improvement. During retraining, the DNNs are updated using newly collected samples from recent inference requests, while the class labels for these retraining data are provided by the cloud-hosted “golden model” [1], [7], [9]. While the labels are provided from the cloud, the process of updating model parameters on edge devices remains resource-intensive. Consequently, existing research has pursued optimizations primarily through algorithm advances and system design. In the realm of algorithm advances, prior works focus on improving the efficiency for single-model continual learning via low-rank adaptation or prudently designed retraining data selection [8]. However, these studies do not fully explore the *data drift relationship among multiple models*, typically opting to retrain all affected models exhibiting such drift. Our preliminary experiments demonstrate there is the existence of a *correlated drift* phenomenon in a multi-model application. Specifically, if two models are correlated, only retraining the preceding model brings an accuracy gain to both models. From the system design perspective, existing continual learning frameworks leverage the prioritized scheduling of retraining tasks [7] or

save multiple versions of the model for model reuse [1] to reduce the resource demand of accurate video analytics. However, these frameworks fail to efficiently manage the timing of retraining triggers and the retraining process itself, leading to suboptimal performance.

Therefore, we argue that an *algorithm-system co-design* is necessary to achieve resource-efficient edge video analytics. By co-design, we mean that to design a holistic framework that fully integrates the correlated drift observation among multiple DNN models in both the algorithm side, e.g., correlation analysis and retraining planning, as well as the system side, e.g., timing of retraining triggers and resource allocation.

To this end, we propose FaaS Learner, a framework that natively supports resource-efficient multi-model continual learning at the edge by leveraging the fine-grained resource management of serverless computing. Serverless computing [10], [11], also known as Function-as-a-Service (FaaS), is a promising paradigm for algorithm-system co-design because it automatically manages the provision and termination of *serverless function* according to ingress workloads. FaaS Learner follows this paradigm as a function-level resource management abstraction from a platform-provider perspective, enabling the platform-side runtime to control function creation, instance reuse, keep-alive duration, and retraining preemption. With the serverless paradigm, as shown in Fig. 1, developers deploy model inference or retraining tasks in separate serverless functions λ to enable fine-grained resource management. This paradigm also aligns with the data drift observation by facilitating the selective and on-demand retraining triggers.

Based on the serverless paradigm, FaaS Learner holistically co-designs multi-model continual learning from both algorithmic and systemic perspectives. This co-design encompasses: (1) at the algorithmic level, identifying models affected by data drift and planning their updates by exploiting the correlated drift phenomenon; and (2) at the system level, determining when to trigger retraining and how to schedule resources for its efficient execution. Specifically, our framework implements this co-design through four key components. On the algorithm side, we propose a *correlated drift analyzer* to select the minimal set of models necessary for retraining. Based on this, we then devise a *correlation-aware retraining planner*, which prioritizes the selected models based on their importance for accuracy recovery. On the system side, we introduce a *fluctuation-agnostic function trigger* to accurately identify genuine data drift amidst natural accuracy fluctuations, which precisely controls the triggering of retraining. Furthermore, we design a *harvest-centric function scheduler* that leverages dynamic, even intermittent, resources harvested from the inference function to execute the retraining inference.

Our main contributions are summarized as follows:

- We present FaaS Learner, the first serverless continual learning framework for multi-model video analytics delivering both high accuracy and resource efficiency.
- Based on the observation of correlated drift, we propose an algorithm-system co-design methodology. On the algorithm side, we employ correlation analysis to select the most critical models for retraining. On the system side,

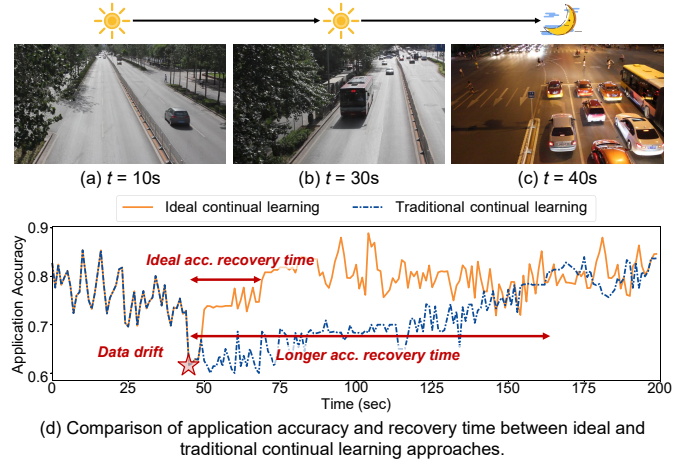


Fig. 2: Example of a scene change when changing in ambient lighting conditions and the comparison of ideal and traditional continual learning.

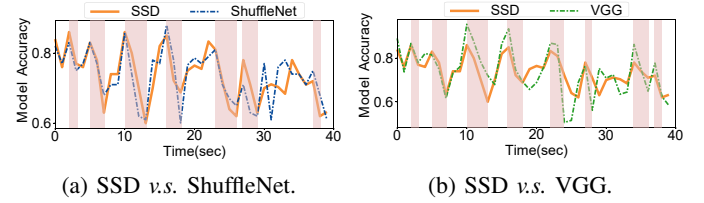


Fig. 3: The accuracy variations of the SSD, ShuffleNet, and VGG models during the 0-40s in Fig. 2.

we achieve precise triggering of the retraining function and efficient harvesting of idle resources.

- We evaluate FaaS Learner on four multi-model applications with the real-world Azure dataset. FaaS Learner improves the average application accuracy up to 31.9% over representative baselines. It harvests over $2.51 \times$ more idle memory for retraining and achieves $4.45 \times$ accuracy recovery per 100 MB-s of retraining memory on average.

II. BACKGROUND AND MOTIVATION

In this section, we introduce the background of continual learning against data shift and opportunities via preliminary experiments, for achieving resource-efficient video analytics with serverless computing. We deploy the multi-model video surveillance application from Fig. 1 and a real-world vehicle detection dataset [12] on an edge server to demonstrate the impact of data drift. The detailed experimental setup can be found in Sec VI.

A. Data Drift in Multi-model Applications

Data drift is a prevalent issue in video analytics that can significantly affect the performance of models over time. As shown in Fig. 2 (a)→(b)→(c), the ambient lighting conditions undergo noticeable changes in the external environment. Fig. 2 (d) further shows that these variations cause a significant drop in application accuracy between 30 and 40 seconds. When the application's accuracy drops below the predefined threshold by

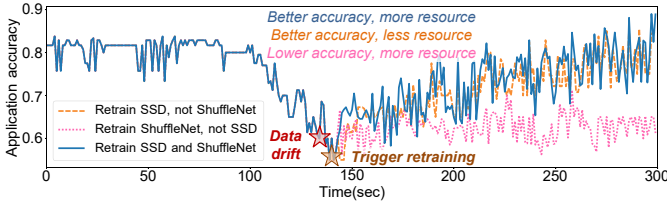


Fig. 4: Illustration of different continual learning strategies on accuracy as trigger retraining starts from 140s.

TABLE I: The comparison of average GPU memory usage (MB-s) and application accuracy for different retraining strategies. ✓ indicates that the DNN model needs to be retrained.

Retraining Strategy	Sub-model			Metric	
	SSD	ShuffleNet	VGG	Memory	Accuracy
Strategy A	✓			385	0.82
Strategy B	✓	✓		632	0.82
Strategy C	✓		✓	657	0.82
Strategy D	✓	✓	✓	1254	0.83

the system, it further triggers a data drift event, as shown at the 45-second mark in Fig. 2 (d). Traditional continual learning strategies typically utilize accuracy degradation as the criterion for detecting data drift and retrain all models exhibiting such degradation, which consumes substantial computational resources. However, due to the resource constraints of edge devices, this continual learning strategy incurs resource contention, consequently leading to a longer accuracy recovery time, as shown in Fig. 2 (d). Therefore, an ideal continual learning strategy featuring a shorter recovery time is critically needed in resource-constrained scenarios to meet service-level objectives (SLOs).

In addition, existing video analytics systems often focus on addressing data drift in single-model applications, where accuracy degradation is the primary concern. However, in more complex directed acyclic graph applications that comprise multiple models [5], [7], the data drift behavior of each model can vary significantly. To explore the data drift characteristics of individual sub-models within directed acyclic graph applications, our analysis focuses on the accuracy variations of the SSD, ShuffleNet, and VGG models during the 0-40 second interval of Fig. 2 (d). The results reveal that when the accuracy of a preceding model (e.g., SSD) declines, subsequent models (e.g., ShuffleNet or VGG) also experience a corresponding degradation in accuracy, as highlighted by the red regions in Fig. 3. We refer to this phenomenon as *correlated drift*. Notably, correlated drift is a recurring behavior in multi-model video analysis applications rather than an artifact confined to the illustrative example in this section.

B. Retraining Strategies for Multi-model Applications

Traditional continual learning approaches typically involve retraining all models that exhibit a decrease in accuracy beyond a predefined threshold. Unfortunately, due to the correlated drift among multiple models, as discussed in Sec. II-A, traditional continual learning methods cause ineffective retraining when applied to multi-model applications. To demon-

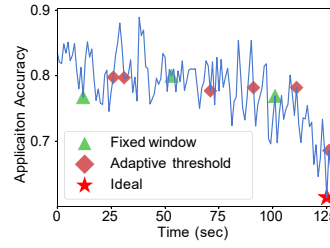


Fig. 5: False trigger.

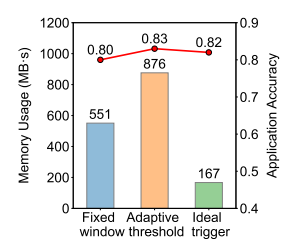


Fig. 6: Resource wastage.

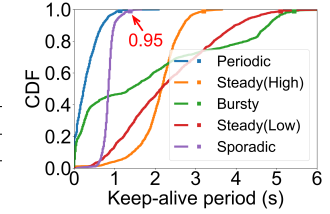


Fig. 7: Keep-alive period.

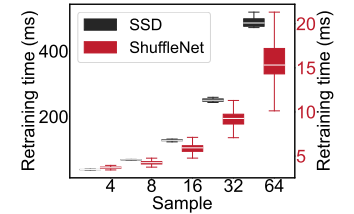


Fig. 8: Retraining time.

strate this issue, we conduct a preliminary experiment where Fig. 4 illustrates the fluctuations in application accuracy over time. The data drift detection method is the same as that described in Sec. II-A. As depicted, data drift is detected 130 seconds, with retraining commencing at 140 seconds. The delay is due to the need to collect sufficient data for retraining. Notably, the three retraining strategies are triggered at 140 seconds, namely only retrain SSD or ShuffleNet, retrain both models. We observe that retraining the SSD model only results in a similar accuracy improvement as retraining both models. This is because the ShuffleNet model for recognition depends on the object detection results of the SSD model.

Further, we compare different retraining strategies for video surveillance applications when encountering correlated drift, focusing on average GPU memory usage (MB-s) and application accuracy. As shown in Tab. I, strategy A achieves comparable accuracy with the least memory. Compared with strategy D, which involves retraining all models, retraining only SSD saves $3.25\times$ memory. This is attributed to the fact that retraining the preceding SSD model also enhances the accuracy of subsequent ShuffleNet models. In other words, improvements in the accuracy of subsequent models can benefit from retraining only the preceding model. Consequently, we take advantage of this correlation to strategically select the models to be retrained, which achieve comparable accuracy against the full retraining with less resource usage.

C. Retraining Trigger against Accuracy Fluctuation

Due to the dynamic nature of user-submitted requests, application accuracy may exhibit fluctuations even in the absence of data drift [8]. As demonstrated by the variations observed in Fig. 2 and Fig. 3, the application accuracy fluctuates significantly over time, with variations exceeding 20% in certain instances.

Current detection methods identify data drift through various approaches, such as fixed-length detection windows [7] and adaptive detection thresholds [13], initiating retraining when accuracy falls below the set thresholds. The former

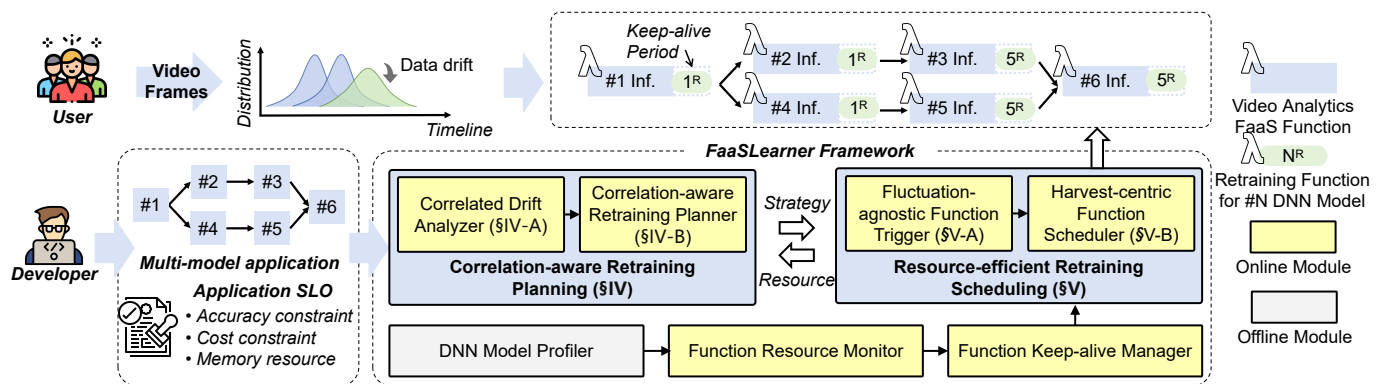


Fig. 9: An overview of FaaSLearner framework.

method employs a fixed-length detection window (e.g., 50 seconds) to monitor whether the accuracy decline exceeds a predefined threshold, as indicated by the triangle in Fig.5. The latter method adaptively determines whether the similarity in input data distribution between consecutive time intervals surpasses a predefined threshold, as shown by the diamond in Fig.5. However, these methods are susceptible to false positives, potentially mistaking normal accuracy fluctuations for data drift. The optimal trigger point occurs near 125 seconds, as indicated by the five-pointed star in Fig.5. Triggering retraining at other times would result in unnecessary retraining.

This frequent retraining not only increases computational overhead but also leads to significant resource wastage. Specifically, we compare the average memory usage (MB-s) and application accuracy of three trigger strategies within 120 seconds. As illustrated in Fig. 6, the fixed-length detection window and dynamic-threshold methods consumed $3.3\times$ and $5.2\times$ more memory resources, respectively, than the ideal trigger method, despite their detection accuracies being relatively comparable. This highlights the limitations of current detection methods, which waste substantial computing resources due to false triggers.

D. Resource Harvesting for Retraining

In the existing serverless computing paradigm, service providers leverage a *keep-alive period* after a function completes its serving to prevent the cold start [10]. During keep-alive, functions are kept idle to await potential new requests. If we harvest these idle resources for model retraining, the overall resource efficiency is improved without additional cost.

To show the possibility of utilizing keep-alive for retraining, we conduct an experiment on the keep-alive period and the retraining time. Fig. 7 illustrates the keep-alive period under 5 workloads of the Azure serverless dataset [10]. We observe that over 95% of keep-alive periods are second-level. In addition, Fig. 8 shows that the retaining time of both SSD and ShuffleNet is in the millisecond range for one iteration when using retraining samples of varying sizes. The iteration-level retraining time includes the forward pass, backward pass, and optimizer update, and excludes data loading time. This is because the retraining samples already reside in GPU memory from continuous inference and are directly reused within the

same function instance. Hence, we observe an opportunity that the idle keep-alive resources can be utilized to retrain for several iterations.

III. FAASLEARNER OVERVIEW

The preliminary experiments presented in Sec. II highlight promising opportunities for achieving resource-efficient multi-model continual learning at the edge. However, these findings also reveal two key challenges that must be addressed to realize such a framework. First, from the algorithm design perspective, it remains difficult to determine whether the accuracy degradation of a model stems from its own data drift or from correlated preceding models within a multi-model application. This ambiguity, coupled with the dynamic and time-varying inter-model correlation, makes it non-trivial to accurately distinguish correlated drift and formulate effective retraining strategies. Second, from the system design perspective, existing drift detection methods are susceptible to false retraining triggers caused by short-lived accuracy fluctuations, leading to severe resource wastage. Meanwhile, although serverless environments provide opportunities to utilize idle resources, their highly dynamic workloads make retraining intermittent and difficult to schedule.

To overcome the above challenges and provide accurate yet resource-efficient video analytics, we build a continual learning friendly serverless framework named FaaSLearner. FaaSLearner targets provider-controlled serverless edge platforms or private FaaS deployments, where the platform runtime can manage function invocation, instance reuse, keep-alive duration, and retraining preemption.

Fig. 9 illustrates the overview of FaaSLearner framework. Developers submit their multi-model video analytics application composed of directed acyclic graphs to FaaSLearner and set the SLO of their applications. A *DNN Model Profiler* analyzes the directed acyclic graph offline to obtain several performance indicators such as retraining time per iteration and the resource usage during inference for retraining planning and scheduling. In addition, cloud-hosted golden models generate pseudo labels for edge-side retraining, not for online inference, because repeated cloud offloading under long or frequent drift violates low-latency SLOs.

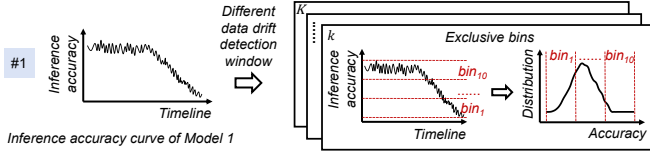


Fig. 10: The generation process of a model's accuracy distribution across different detection windows.

Specifically, users submit video frames to a specified video analytics application, but the data or label distribution of the inputs can be different from the pre-trained models. In FaaS Learner, the *Correlated Drift Analyzer* (§IV-A) identifies the truly necessary models for retraining by analyzing the correlations among multiple models. Then, the *Correlation-aware Retraining Planner* (§IV-B) formulates retraining strategies, including the selection of models, samples, and iterations. Next, we design the *Fluctuation-agnostic Function Trigger* (§V-A) to accurately trigger the retraining task and avoid the waste of computing resources caused by false triggering of accuracy fluctuations. Finally, the *Harvest-centric Function Scheduler* (§V-B) exploits idle resources during the keep-alive period to efficiently manage retraining tasks.

During the execution of each serverless function, the *Function Resource Monitor* watches and adjusts the resource allocation of each function akin to Freyr [11] with the help of offline-generated model profiles. Furthermore, a *Function Keep-alive Manager* handles the keep-alive period of each function according to the request ingress rate. FaaS Learner adopts the widely-used long-short term histogram policy [10]. It is worth noting that the resource scheduling of FaaS Learner is not bound to any keep-alive policies and the policy can be substituted if necessary.

IV. CORRELATION-AWARE RETRAINING PLANNING

On the algorithm side, to address the ever-changing correlated drift across different times and models, we propose *Correlated Drift Analyzer* (§IV-A) and the *Correlation-aware Retraining Planner* (§IV-B) to avoid ineffective retraining and dynamically update retraining strategies.

A. Correlated Drift Analyzer

Preliminary experiments show that, within a given detection window, retraining solely the preceding model can achieve accuracy improvements comparable to those obtained through joint retraining of correlated models. Exploiting this observation, we reduce unnecessary retraining by analyzing the correlations between multiple models in the window. Specifically, we model the inference accuracy curve. Then we quantify the pairwise correlation of these curves to identify which models are actually affected by the correlated drift.

Firstly, we employ a sequence of data drift detection windows $W = \bigcup_{k=1}^K w_k$ to partition video frames. The inference accuracy of each video frame of window w_k is represented as $A_k = \{a_1, a_2, \dots, a_\nu, \dots, a_{|w_k|}\}$, where $|w_k|$ represents the size of k th window, ν represents the index of video

frames. Next, we discretize the inference accuracy into m non-overlapping bins to obtain a probability distribution F , as depicted in Eq. 1, where $\mathbf{1}\{\cdot\}$ denotes the indicator function. For example, when $m = 10$, the accuracy interval $[0, 1]$ is discretized into 10 mutually exclusive bins, each of width 0.1 (i.e., $\text{bin}_1 = [0, 0.1)$, $\text{bin}_2 = [0.1, 0.2)$, ..., $\text{bin}_{10} = [0.9, 1.0]$). Consequently, each model is associated with a accuracy distribution $F = [F(1), F(2), \dots, F(m)]$ within the window w_k . Fig. 10 shows an example of a single model across different detection windows to illustrate the generation process of its accuracy distribution vector.

$$F(\eta) = \frac{1}{|w_k|} \sum_{\nu=1}^{|w_k|} \mathbf{1}\{a_\nu \in \text{bin}_\eta\}, \quad \eta = 1, 2, \dots, m, \quad (1)$$

Secondly, we employ the Kullback-Leibler (KL) divergence to quantify the correlation between the accuracy distributions of different models. The KL divergence is commonly employed to evaluate correlation in the distribution of input content when determining data drift [14]. Meanwhile, high-dimensional visual data must undergo computationally intensive dimensionality reduction [7] prior to evaluation, which can negatively impact service latency.

Fortunately, the inference accuracy distribution of each model can be captured online in real time. Specifically, for a model v_i , its inference accuracy distribution is constructed from its per-frame inference accuracy values observed over time within a detection window. Based on these distributions, we compute the KL divergence between multiple models to characterize the correlation of their inference accuracy distribution curves. This approach avoids the computational burden associated with high-dimensional visual data processing and achieves efficient correlation analytics, as depicted in Eq. 2.

$$IC_{v_i, v_j}^{w_k} = D_{KL}(P_{v_i}^k || Q_{v_j}^k) = \sum_{t \in T} P_{v_i}^k(t) \log \frac{P_{v_i}^k(t)}{Q_{v_j}^k(t)}, \quad (2)$$

where $IC_{v_i, v_j}^{w_k}$ represents the correlation of the inference accuracy distribution curve between models v_i and v_j within window w_k . We use $P_{v_i}^k(t)$ and $Q_{v_j}^k(t)$ to represent the inference accuracy distribution of v_i and v_j with timestamp t in the k th window, respectively. The correlation matrix between multiple models is obtained via Eq. 2. A smaller value indicates higher correlation between models, suggesting a greater likelihood of correlated drift phenomena.

Subsequently, we establish a correlation threshold λ and identify multiple model pairs below this threshold. We posit that these model pairs exhibiting data drift conform to correlated drift characteristics, where comparable retraining outcomes can be achieved solely through retraining preceding models. The correlation analysis in this stage is used to provide an initial retraining-priority signal rather than a final or causal decision rule. Its role is to identify likely correlated model pairs based on statistical association and reduce unnecessary retraining in the current window.

B. Correlation-aware Retraining Planner

Although we mitigate unnecessary retraining through correlation analysis, edge devices with constrained resources com-

Algorithm 1: Priority-driven Retraining Algorithm

Input: Correlation matrix between multiple models, cost constraint $\mathcal{B}$, video frame set $\mathcal{F}$.

Output: Retraining strategy $\pi_{k=1,2,\dots,K}$.

```

1 for  $k$ th window in  $K$  do
2    $v^{ret} = \arg \max_{v_i} \sum_{v_j} IC_{v_i, v_j}^{w_k}$ ;
3    $sample_k \leftarrow \maxDistance(\mathcal{F}_k, \mathcal{F}_{k-1})$ ;
4   for  $iteration = 1, 2, 3, \dots$  do
5      $\pi_k \leftarrow v^{ret}, sample_k, iteration_k$  ;
6      $acc_{\pi_k}, budget_{\pi_k} \leftarrow$ 
        $Ret_{\pi_k}(v^{ret}, sample_k, iteration_k) \triangleright$  Retrain
       model and output accuracy and cost;
7     if  $budget_{\pi_k} \geq \mathcal{B}$  then
8       return  $\pi_k$ ;
9     else if  $\frac{acc_{\pi_k}}{acc_{\pi_{k-1}}} \geq 1$  then
10      continue;
11 return retraining strategy  $\pi_{k=1,2,\dots,K}$ .
```

pel us to meticulously formulate retraining strategies within each detection window. To solve the problem, we propose the *correlation-aware retraining planner* by analyzing the accuracy gain of retraining to generate retraining strategies.

Firstly, we formulate an optimization problem maximizing accuracy gain from retraining to determine optimal strategies. We select retraining models with the strategy π_k under cost constraint $\mathcal{B}$ and accuracy constraint a_{req} set by developers to maximize retraining accuracy gain. As shown in Eq. 3, $Ret_{\pi_k}(S_k^*, sample_k, iteration_k)$ express that in the k th window, the optimal retraining model set S_k^* is selected, and retraining is performed by determining the sampling $sample_k$ and retraining times $iteration_k$.

$$\begin{aligned}
 \max \quad & \sum_{k=1}^K Ret_{\pi_k}(S_k^*, sample_k, iteration_k) \\
 \text{s.t.} \quad & \sum_{k=1}^K \frac{(a_1 + \dots + a_{|w_k|})/|w_k|}{K} > a_{req} \quad (3) \\
 & r \cdot m \cdot T_G \leq \mathcal{B} \quad (3-2).
 \end{aligned}$$

Constraint (3-1) indicates that the average accuracy of windows meets the accuracy constraint. Constraint (3-2) ensures that the total retraining cost does not exceed the specified monetary budget. In constraint (3-2), r is the price per GB-second based on AWS pricing policies [15], m represents the resources usage consumed by applications, and T_G is the execution time after rounding to the nearest billing unit.

Secondly, for solving this constrained optimization problem, we design a *Priority-driven Retraining Algorithm* to determine strategy π_k by selecting parameters S_k^* , $sample_k$, $iteration_k$ in detection window w_k , as shown in Algorithm. 1. This algorithm accelerates model retraining by strategically prioritizing retraining models that demonstrate the most significant accuracy gain during iterative retraining.

Specifically, within each detection window w_k , we select the model v^{ret} that has the highest correlation with subse-

quent models (Line 2) by Eq. 4. This model is assigned the top retraining priority, while other priorities are determined accordingly. Since v^{ret} represents the most influential predecessor, retraining it first not only improves its own accuracy but also propagates accuracy gains to its correlated successors, thereby reducing the need for redundant retraining.

$$v^{ret} = \arg \max_{v_i} \sum_{v_j} IC_{v_i, v_j}^{w_k}. \quad (4)$$

In addition, we select the sampling strategy $sample_k$ by selecting the samples with the greatest difference within adjacent windows following AdaInf [7] (Line 3). Meanwhile, the retraining process is dynamically regulated by simultaneously monitoring accuracy and cost constraint to determine retraining iterations (Lines 4-10). To balance accuracy and cost, we determine $iteration_k$ by establishing a stopping criterion for increasing iterations if the gradient of the accuracy gain between two adjacent windows becomes gradually smaller. The logic behind this is that deep learning model training curves are generally concave (i.e., accuracy increases rapidly and then slows down) [1]. While this approach may result in suboptimal model performance, it enables achieving a satisfactory accuracy gain within a reasonable cost. Finally, we continue to cycle through the above process in the window until the constraints are not satisfied.

Importantly, the priority given by correlation analysis is not fixed across windows. Under confounded-drift scenarios, upstream and downstream models may be simultaneously affected by a common external factor, such as lighting, weather, or camera-view changes. In such cases, retraining the initially selected upstream model may not fully recover downstream accuracy. If this upstream adaptation yields limited accuracy gain, the planner stops further retraining of that model according to the stopping criterion and re-evaluates the retraining target in subsequent windows.

V. RESOURCE-EFFICIENT RETRAINING SCHEDULING

On the system side, to address the false retraining trigger caused by short-lived accuracy fluctuations and intermittent retraining resource, we design the *Fluctuation-agnostic Function Trigger* (§V-A) and the *Harvest-centric Function Scheduler* (§V-B) to precisely trigger retraining function and harvest idle resources in serverless inference function.

A. Fluctuation-agnostic Function Trigger

In a serverless computing paradigm, the detection of data drift directly triggers the serverless function for model retraining. Consequently, the precision of the drift detector is critical: a single false positive incurs the cost of provisioning computational resources for an unnecessary retraining, while a missed detection delays accuracy recovery and degrades service quality. Traditional detection methods [7], [1], being susceptible to transient accuracy fluctuations inherent in dynamic video content, often lead to such resource inefficiency. To address this, we design *fluctuation-agnostic function trigger* that accurately distinguishes genuine data drift from benign accuracy noise, thereby ensuring that the serverless function

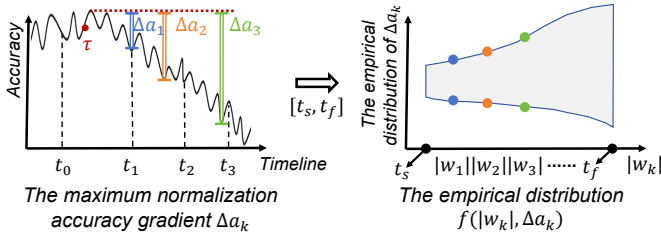


Fig. 11: Illustration of the maximum normalization accuracy gradient and distribution function computation.

for retraining is launched only when truly warranted. Our detector enhances robustness from two perspectives: an adaptive detection window that dynamically adjusts its sensitivity, and a fluctuation-agnostic distribution-aware detector that filters out spurious fluctuations.

Adaptive drift detection window. Conventional data drift detection methods lead to missing the optimal detection location of data drift. For example, a large detection window risks delaying retraining, while a small detection window may trigger false positives. Therefore, we design the adaptive drift detection window to locate the optimal detection point.

Specifically, the window size shrinks when the inference accuracy is close to the accuracy constraint, ensuring that drift is detected in smaller timeframes. Conversely, if the inference accuracy is further from the accuracy constraint, the window size increases to accommodate larger fluctuations and avoid false positives. To implement this adaptive mechanism, the window size $|w_k|$ is regulated by Eq. 5 within the range $(0, |w_{max}|)$, where $|w_{max}|$ represents the maximum detection window size. The experimental setup employs the maximum detection window size of 50 seconds, consistent with the parameter configuration used in existing continual learning systems such as AdaInf [7]. Furthermore, by employing a sliding (rather than segmented) window, our approach can seamlessly capture data drift events that persist beyond a single 50-second interval.

$$\begin{aligned} |w_k| &= |w_{max}| \times \text{Sigmoid}(x) \\ &= |w_{max}| \times \frac{1}{1 + e^{-(\alpha x + \beta)}}. \end{aligned} \quad (5)$$

$x = \text{ave}(A_{k-1})$ represents the average accuracy in the $k-1$ window. Given the properties of the Sigmoid function, when x approaches high accuracy (close to 1), we constrain the Sigmoid output to converge towards 1, thus edging closer to the maximum window $|w_{max}|$. Similarly, when x gets closer to the accuracy constraint a_{req} set by developers, we constrain the Sigmoid value to approach 0 to minimize the size of the detection window to avoid missing the optimal retraining. The hyperparameters α and β dictate the speed at which the window size diminishes.

Fluctuation-agnostic distribution-aware detector. Once the adaptive window defines the scope for detection, this detector determines whether a data drift has occurred within it, thus deciding if a retraining instance should be triggered. It prevents false triggers from transient accuracy drops (e.g., temporary object occlusions) by modeling the empirical distribution of

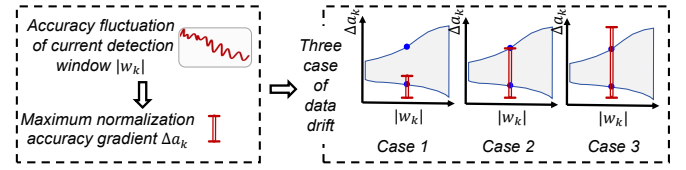


Fig. 12: Illustration of data drift in the current detection window $|w_k|$. Data drift is declared when the gradient deviates beyond the distribution boundaries.

normal accuracy fluctuations. A significant deviation of the current inference accuracy from this distribution indicates genuine data drift and triggers retraining instances.

Specifically, we establish a baseline empirical distribution for accuracy under normal fluctuations, as shown in Fig. 11. Firstly, we identify the maximum normalization accuracy gradient within detection windows at a specific time t_0 :

$$\Delta a_k(|w_k| = t_i - t_0) = \max_{t_0 \leq \tau \leq t_i, i=1,2,\dots} \frac{|a_k(\tau) - a_k(t_0)|}{|a_k(t_0)|}, \quad (6)$$

where Δa_k represents the maximum variation of inference accuracy in window w_k , and τ represents a dynamically advancing microtime marker. Next, we offline construct the function $f(|w_k|, \Delta a_k)$ as depicted in Eq. 7. This function characterizes, for each window size $|w_k|$, the empirical distribution of the maximum normalized accuracy gradient Δa_k observed over the interval $[t_s, t_f]$. In other words, it captures how the maximum accuracy variation evolves with different window sizes across time. Here, t_s and t_f denote the starting and ending points of the baseline distribution, respectively.

$$\begin{aligned} f(|w_k|, \Delta a_k) &= \\ \bigcup_{k,j} \left\{ \Delta a_k(|w_k|_j = (t_j + |w_k|) - t_j) \mid t_s \leq t_j \leq t_f \right\}. \end{aligned} \quad (7)$$

Subsequently, we determine whether the accuracy fluctuation in the current detection window conforms to the baseline empirical distribution. As shown in Fig. 12, a data drift is inferred within the window if its maximum normalized accuracy gradient falls outside this empirical distribution. Otherwise, the fluctuation is regarded as normal.

B. Harvest-centric Function Scheduler

To improve resource efficiency in terms of serverless frameworks, pioneer approaches [11], [16] harvest the over-configured resources and re-allocate them to under-provisioned services. However, these existing works overlook the opportunity to harvest idle resources during the *function keep-alive period* which is widely adopted in state-of-the-art research and commercial serverless platforms. Furthermore, as we illustrated in §II-D, the keep-alive period is suitable for retraining tasks. Hence, to improve resource efficiency in terms of serverless frameworks, FaaS Learner proposes the *harvest-centric function scheduler* to perform continual learning with not only the harvested resources of over-provisioned functions akin to Freyr [11] but more importantly the keep-alive period of each function. In particular, FaaS Learner leverages the aggregated

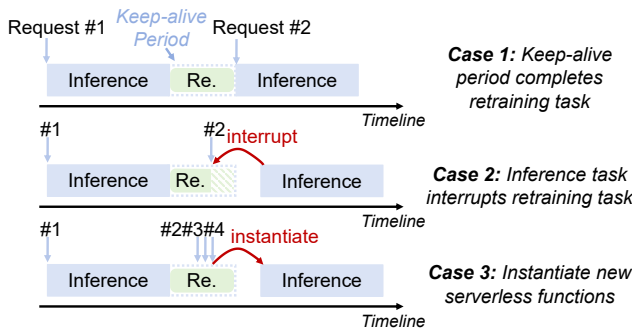


Fig. 13: Three cases of retraining task execution: opportunistic completion during keep-alive period, preemption by an inference task, and execution in a newly instantiated function.

keep-alive resources across multiple function instances, rather than requiring a complete retraining iteration to fit within the keep-alive period of a single instance.

In addition, retraining scheduling presents two key challenges. First, retraining candidates and priorities are dynamic because the correlation-aware retraining planner may defer certain models. To fully exploit keep-alive periods in multi-model applications, FaaS Learner distributes retraining tasks across multiple function instances. For example, in Fig. 9, the highest-priority retraining tasks for Models #1 and #5 are assigned to all instances in the directed acyclic graph. As shown in Case 1 of Fig. 13, the available idle resources during keep-alive are sufficient for retraining. Second, keep-alive periods are unpredictable. New inference requests may arrive at any time, requiring retraining to be preempted for inference serving (Case 2 in Fig. 13). Accordingly, FaaS Learner prioritizes inference and performs retraining opportunistically during inference idle time. Because retraining is executed in millisecond-scale iterations, interrupted execution resumes from the nearest checkpoint, bounding preemption overhead at at most one iteration.

In terms of scheduling strategy, FaaS Learner makes dynamic and intermittent resources continuous and available by utilizing known conditions. These conditions include the dynamically determined keep-alive period obtained by analyzing historical workloads, the retraining time of one iteration per model as obtained by the DNN Model Profiler, and the retraining policy. Furthermore, if requests continuously arrive, FaaS Learner instantiates new serverless function instances for retraining, as shown in Case 3 of Fig. 13. This fallback is also triggered when keep-alive periods are shorter than the per-iteration retraining time, or when commercial FaaS platforms freeze execution environments after response return. In these cases, FaaS Learner retains correlation-aware planning and on-demand triggering, while the additional resource-reuse benefit of keep-alive harvesting becomes unavailable.

To facilitate the shared keep-alive retraining across different functions, a naïve approach is to create a shared file for the model weights and mount it to each function. This direct mounting based on file storage is inefficient and increases the risks of misoperation and malicious abuse. Hence, FaaS Learner creates a virtual in-memory file system with `tmpfs` [10], [17] and mounts it to each retraining

task. Therefore, checkpoint save and restore operations are performed at memory speed rather than disk speed, which substantially reduces the overhead compared with disk-based storage. In addition, because `tmpfs` avoids disk I/O, it also avoids disk-level resource contention with inference requests when retraining is interrupted and later resumed.

VI. EVALUATION

A. Experimental Setup

Testbed. We utilize 4×RTX3090 GPUs as a cloud server for the “golden model”, leveraging their robust computational capabilities to generate retraining labels. For executing the serverless function invocations, we deploy a Docker container on the custom private edge server without the freeze semantics of commercial FaaS platforms such as AWS Lambda. This testbed implements the provider-controlled serverless edge runtime assumed by FaaS Learner, where warm function instances remain executable during keep-alive periods and thus enable continual retraining. The edge server is equipped with 1×RTX 3090 GPU and a 20-core Intel i7-12700 CPU, with CUDA version 12.6. The two servers are connected via 1 Gbps Ethernet. In addition, we employ the Microsoft’s Azure serverless dataset [10] to simulate five typical types of trigger workload, namely Periodic, Steady (High), Bursty, Steady (Low), and Sporadic. We use a widely-adopted weighted hybrid histogram policy [10] to calculate the keep-alive period. The average keep-alive of the five aforementioned workloads is 0.53s, 0.66s, 0.71s, 1.22s, and 2.01s, respectively.

Applications. We implement four DAG applications from [7], [5]: vehicle monitoring (VM), AMBER alert pipeline (AMBER), pedestrian activity recognition and tracking (PART), and automatic driving panoramic perception (ADPP). Complexity increases from VM to ADPP. We employ three high-accuracy golden models consistent with AdaInf [7] and RECL [1]: ResNeXt 101 (88%), YOLOv3 (89.19%), and U2Net (89.6%) for object classification, detection, and segmentation. A more detailed description is shown in Fig. 14, where the model and pre-training weights for the deep learning components are obtained HuggingFace [3]. In addition, the computational cost of this labeling process is not included in our measurements.

Datasets. Current video analytics datasets, such as UA-DETRAC [12], Sports MOT [18], and Cityscapes [19], usually do not directly contain video clips with data drift. To address the absence, we follow traditional approaches [1], [20], [21] to reconstruct the datasets by altering scenes or urban settings at regular time intervals, thereby simulating data drift. For instance, the UA-DETRAC dataset encompasses scenes from 24 different cities and four types of weather conditions. We simulate data drift by switching cities and weather conditions.

Baselines. We compare FaaS Learner with four continual learning frameworks. (1) *Gecko* [21] retrains data drift models by collecting a fixed number of low-accuracy samples and launches a serverless function for reallocating resource under source contention. (2) *AdaInf-Freyr (AI-F)*. Freyr [11], a serverless computing platform, efficiently utilizes idle resources for functions that are over-configured. We extend Freyr

TABLE II: The comparison of application accuracy under resource constraints.

		Gecko	AI-F	AT-F	Ours	Ideal
VM	Periodic	0.4932	0.5643	0.6474	<u>0.8148</u>	0.8522 _{+359.9}
	Steady(High)	0.4856	0.5681	0.6362	<u>0.8378</u>	0.8561 _{+360.3}
	Bursty	0.4963	0.5726	0.6782	0.8643	<u>0.8542</u> _{+372.6}
	Steady(Low)	0.4837	0.5768	0.6812	0.8876	<u>0.8578</u> _{+362.8}
	Sporadic	0.4985	0.5789	0.6836	0.8975	<u>0.8584</u> _{+361.8}
AMBER	Periodic	0.5253	0.5844	0.6585	<u>0.8245</u>	0.8384 _{+371.5}
	Steady(High)	0.5342	0.5768	0.645	<u>0.8353</u>	0.8432 _{+361.8}
	Bursty	0.5353	0.5823	0.6834	0.8563	<u>0.8468</u> _{+360.7}
	Steady(Low)	0.5242	0.5812	0.6854	0.8882	<u>0.8502</u> _{+358.8}
	Sporadic	0.5224	0.5825	0.6876	0.9102	<u>0.8531</u> _{+369.4}

		Gecko	AI-F	AT-F	Ours	Ideal
PART	Periodic	0.5537	0.6796	0.7534	<u>0.8154</u>	0.8645 _{+480.2}
	Steady(High)	0.5539	0.6848	0.7541	<u>0.8353</u>	0.8662 _{+485.1}
	Bursty	0.5558	0.6663	0.7654	0.8664	<u>0.8653</u> _{+475.7}
	Steady(Low)	0.5673	0.6652	0.7614	0.8785	<u>0.8676</u> _{+466.1}
	Sporadic	0.5753	0.6887	0.7743	0.8952	<u>0.8688</u> _{+478.7}
ADPP	Periodic	0.6257	0.8654	<u>0.8853</u>	0.9043	<u>0.8845</u> _{+1081.1}
	Steady(High)	0.6554	0.8787	<u>0.8886</u>	0.9131	<u>0.8823</u> _{+1117.8}
	Bursty	0.6483	0.8754	0.8878	0.9223	<u>0.8914</u> _{+1088.6}
	Steady(Low)	0.6483	0.8735	0.8845	0.9274	<u>0.8952</u> _{+1085.5}
	Sporadic	0.6576	0.8764	0.8852	0.9362	<u>0.8958</u> _{+1079.7}

$\gamma_{+\delta}$ means that *Ideal* consumes additional δ MB-s of memory resources per request, achieving γ application accuracy. Bold and underlined text indicates the best and second-best application accuracy, respectively.

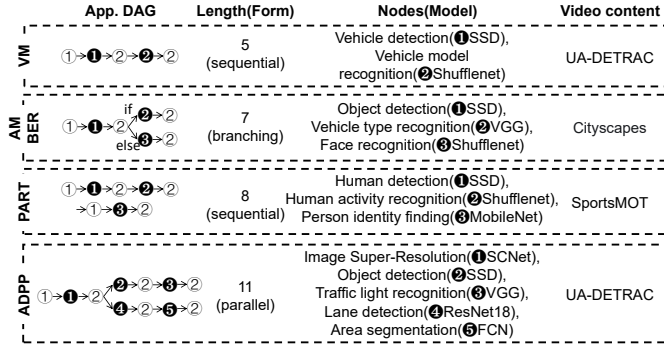


Fig. 14: Detailed description of the applications and datasets. Each application contains (i) deep learning components that can be retrained and (ii) non-deep learning components, such as pre-processing ① (video encoding and decoding, data processing, etc.) and post-processing ② (image cropping, etc.) to improve the data exchange between components.

with continual learning, named AdaInf [7]. AdaInf identifies data drift by utilizing a fixed detection window and quantifies the impact of data drift on the model using an impact degree metric. (3) *AdaTrigger-Freyr* (AT-F). We also extend Freyr with continual learning, named AdaTrigger [13]. AdaTrigger formulates an adaptive threshold based on the decline in sampling frame accuracy to initiate the retraining process. (4) *Ideal*. It serves as an unconstrained baseline to evaluate Gecko's performance across various resource configurations. This configuration removes cost limitations to provide insights into Gecko's accuracy potential under optimal conditions.

Metrics. We employ the application accuracy, memory usage, accuracy gain, and accuracy recovery time to evaluate the performance of different continual learning frameworks for resource-efficient video analytics. (1) *Application Accuracy* refers to the overall accuracy of video analytics applications under developer-specified application SLOs. (2) *Memory Usage* refers to the amount of memory resources consumed during the retraining process. (3) *Accuracy Gain* refers to the percentage increase in model accuracy per unit of resource consumption (e.g., per 100 MB-s of memory resources). This metric reflects the accuracy improvement efficiency through retraining in resource-constrained scenarios. (4) *Accuracy Recovery Time* is the duration from when data drift starts to when the application's accuracy returns to a satisfactory level

of accuracy. This metric evaluates the retraining efficiency of a continual learning framework in addressing data drift. A shorter recovery time indicates a faster response, enhancing service quality.

B. End-to-end Evaluation

We compare FaaS Learner with the state-of-the-art serverless continual learning framework, regarding accuracy and resource efficiency. We validate the efficiency of FaaS Learner by implementing 20 benchmarks that include a mix of four applications and five workloads.

Application Accuracy. As shown in Tab. II, compared with Gecko, AI-F, and AT-F, FaaS Learner improves the average application accuracy by 31.9%, 19.9%, and 12.9%, respectively. Compared with Ideal, FaaS Learner improves the average application accuracy of 1.09%, but Ideal pays an additional 573.96 MB-s of average resource consumption for each retraining. FaaS Learner harvesting idle resources without the extra effort of resource usage. Additionally, the accuracy of FaaS Learner is comparable to that of Ideal, and in some benchmarks, even surpasses Ideal. Despite Ideal not being constrained by retraining resources, the inadequacy of the data drift detection technique results in FaaS Learner outperforming Ideal in terms of accuracy.

In addition, as the directed acyclic graph of applications grows in complexity, the application accuracy of FaaS Learner and baselines steadily increases. For example, under Periodic workload, compared to the simplest-structured VM application, Gecko, AI-F, AT-F, and FaaS Learner achieved improvements of 13.25%, 30.11%, 23.79%, and 8.95% respectively in the most complex-structured ADPP application. The reason for this is that complex applications with directed acyclic graphs include more DNN models, leading to an abundance of idle resources. Consequently, FaaS Learner and baselines, which retrain by utilizing these idle resources, are better equipped to fulfill accuracy and cost constraints in the more complex ADPP application. Furthermore, with the prolonged keep-alive period of workloads, the application accuracy of FaaS Learner steadily improves, whereas the accuracy of the baselines remains relatively stable. The improving accuracy in FaaS Learner can be attributed to retraining by harvesting resources of the keep-alive period, with more resources becoming available as the keep-alive period increases.

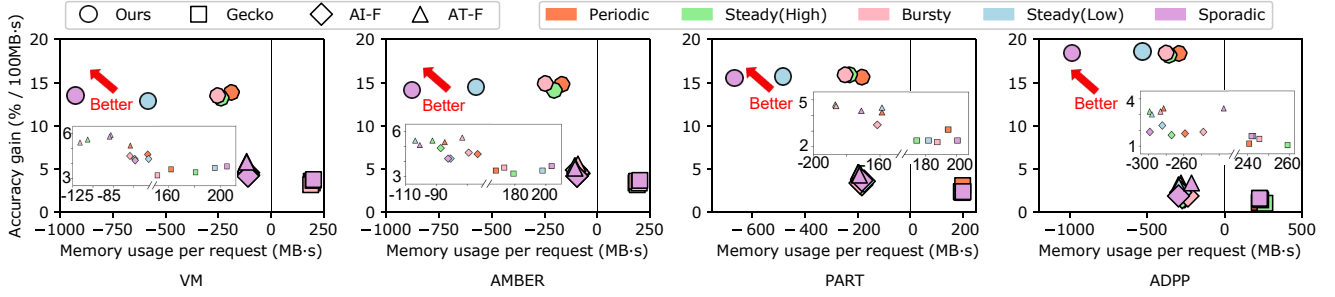


Fig. 15: The comparison of memory usage and accuracy gain in retraining windows. A negative memory usage represents the harvested idle resources from the serverless function of the framework, and a positive value means the additional resource usage for retraining. We draw detailed mini-images to distinguish overlapping baselines.

Accuracy Gain. We compare the average accuracy gain of FaaS Learner and baselines of memory resources consumed, as shown in Fig. 15. The y -axis represents the accuracy gain achieved for every 100 MB-s of consumed memory resources. The x -axis shows the average memory usage per request (MB-s), relative to the default memory configuration. During retraining, the average accuracy gain achieved by FaaS Learner, Gecko, AI-F, AT-F are 15.5%, 2.7%, 3.5%, and 4.6%, respectively. This primarily results from selectively retraining FaaS Learner through correlation analysis between multiple models, thereby preventing the unnecessary retraining of models. Conversely, Gecko, AI-F, and AT-F retrain all models affected by data drift. Under the constraint of the cost budget, retraining unnecessary models competes for memory resources, resulting in slower accuracy gain.

In addition, we analyze the influence of the directed acyclic graph complexity of different applications on the accuracy gain. In the VM application with a simple directed acyclic graph structure, compared with Gecko, AI-F, AT-F, FaaS Learner has an average accuracy gain of $3.7\times$, $3.1\times$, and $2.4\times$, respectively. In the ADPP application with a more complex directed acyclic graph structure, compared with Gecko, AI-F, AT-F, FaaS Learner has an average accuracy gain of $13.2\times$, $9.5\times$, and $5.6\times$, respectively. We notice that FaaS Learner exhibits a greater increase of accuracy gain in more complex applications with a directed acyclic graph due to the increase of triggered retraining frequency.

Memory Usage. As shown in Fig. 15, we compare the memory usage of FaaS Learner and baselines during retraining. FaaS Learner, AI-F, and AT-F harvest an average memory usage of 430.8MB-s, 166.2MB-s, and 171.4MB-s. Compared to AI-F and AT-F, FaaS Learner harvests more than $2.51\times$ memory resources for retraining. Additionally, we harvest most of the memory resources except those consumed by inference, including over-configured memory resources and those during keep-alive. Compared to AI-F, AT-F, and FaaS Learner, Gecko incurs an additional $0.84\times$, $1.82\times$, and $3.12\times$ of average memory resources to support retraining. This is because Gecko needs to launch a new serverless function for each retraining.

In addition, we analyze the influence of different keep-alive periods on memory usage. In comparison to Periodic, the resource consumption in Steady (High) is approximately 27% higher, Bursty is 40% higher, Steady (Low) is 197%

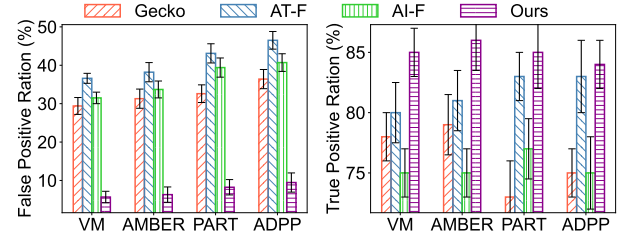


Fig. 16: The comparison of false and true positive ratio of drift detection under the workload *Periodic*.

higher, and Sporadic is 371% higher. We find that the resources harvested by FaaS Learner increased with the growth of the keep-alive periods. Differently, we observe in Fig. 15 that when the keep-alive periods change, the resource usage by AI-F, AT-F, and Gecko are relatively stable. This also shows that the longer the keep-alive, FaaS Learner harvests more idle resources from the serverless function.

C. Effectiveness of Data Drift Detector

We compare the ratio of false positives and true positives and show the resource usage and utilization analysis under a data drift segment. The ground truth of the data drift point is difficult to capture accurately. Therefore, we define the data drift range as the adjacent detection window near the ground truth. If the detection framework can detect data drift within this range, it is considered a true positive; otherwise, it is a false positive.

False Positive and True Positive. As shown in Fig. 16, FaaS Learner achieves the lowest false positive ratio, reducing it to below 10%, which is on average $4.4\times$, $4.9\times$, and $5.5\times$ lower than Gecko, AI-F, and AT-F, respectively. This is because FaaS Learner captures a longer temporal window to predict accuracy, making it more robust to short-term accuracy fluctuations, whereas Gecko, AI-F, and AT-F rely on fixed detection windows or threshold-based comparisons and are therefore more sensitive to such fluctuations. Meanwhile, FaaS Learner also attains the highest true positive rate, with an average of 85%, outperforming Gecko, AI-F, and AT-F by 12%, 4%, and 13%, respectively. Although AT-F achieves a comparable true positive rate due to its adaptive-threshold

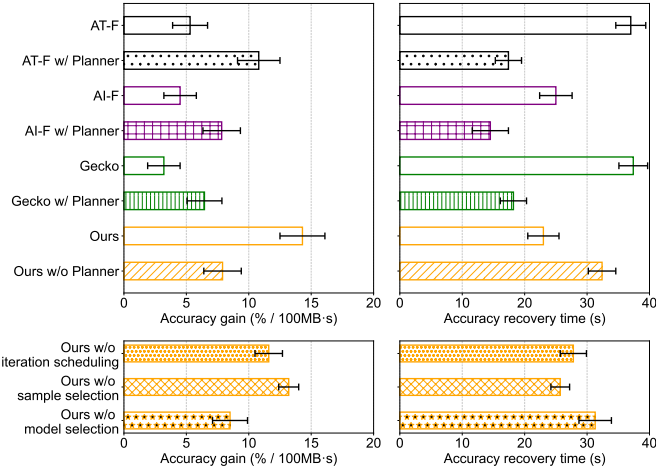


Fig. 17: Comparison of accuracy gain and recovery time under the application *VM* and workload *Periodic*.

mechanism that enables multiple detections within a single window, it also incurs the highest false positive rate among the baselines. As the application becomes more complex, both FaaS Learner and baselines exhibit higher false positive ratios, since accuracy fluctuations in more complex applications propagate across multiple models more significantly.

Comparison with Feature Embedding-Based Drift Detection. We further compare FaaS Learner’s accuracy-based drift trigger with a feature embedding-based alternative from Gecko [21] on ShuffleNet and SSD in terms of detection sensitivity and computational overhead. The accuracy-based method uses scalar accuracy, offering low overhead and direct alignment with SLOs. The feature-based method extracts frame embeddings and computes Hellinger distance to detect drift, explicitly encoding semantic content. To evaluate both methods, we construct 100 real drift windows (50 seconds each) randomly distributed within one hour of video as ground truth. Sensitivity is defined as the proportion of correctly identified drift windows. Overhead is the average time to compute the KL divergence of FaaS Learner or the embedding distance of Gecko. As shown in Tab. III, both methods achieve sensitivity in the range of 0.76–0.81. However, the accuracy-based method introduces substantially lower overhead, achieving a $2.51\times$ speedup over the feature embedding-based approach.

D. Efficiency of Retraining Planner

To demonstrate the efficiency of Correlation-aware Retraining Planner (hereafter referred to as Planner), we initially compare the accuracy recovery times of various continual learning frameworks. Subsequently, we visualize the impact of correlations among multiple models on retraining time within a data drift segment.

Ablation evaluation for Planner. We conduct ablation studies by integrating it into the baseline and removing it from FaaS Learner. As shown in the top row of Fig. 17, the results demonstrate its significant impact: when integrated, the baselines achieve $1.93\times$ and $1.97\times$ average improvements in accuracy gain and recovery time, respectively. Conversely,

TABLE III: Comparison of drift detection algorithms in terms of sensitivity and computational overhead.

Method	Model	Sensitivity	Overhead (ms)
Accuracy-based	ShuffleNet	0.78	40.92
	SSD	0.76	40.30
Embedding-based	ShuffleNet	0.81	99.19
	SSD	0.78	104.76

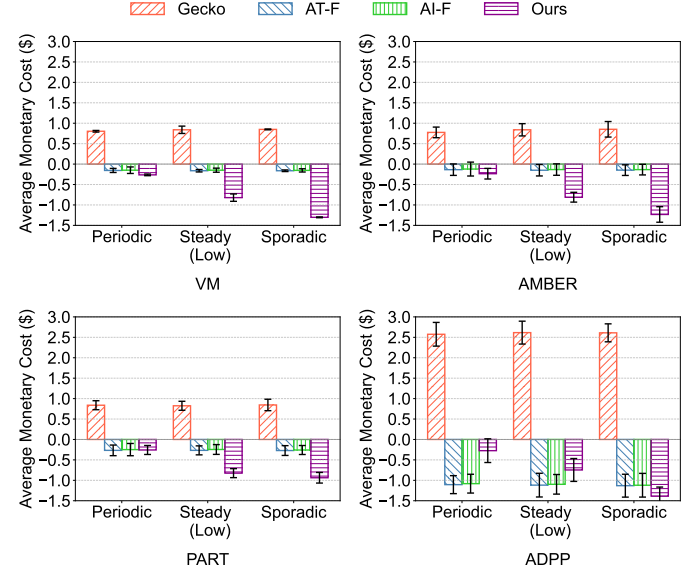


Fig. 18: The comparison of average monetary cost.

removing Planner from FaaS Learner led to a $1.81\times$ decrease in accuracy gain and a 41% increase in recovery time. This improvement can be attributed to the Planner’s strategy of prioritizing models with the highest potential accuracy gain for retraining, thereby enabling efficient continual learning.

We further perform fine-grained ablations on the internal components of the Planner, including model selection, sample selection, and iteration scheduling, as shown in the bottom row of Fig. 17. In particular, removing model selection causes the largest drop in accuracy gain, indicating that prioritizing the most beneficial retraining target is critical for efficient continual learning. Removing sample selection or iteration scheduling also reduces performance, showing that sampling strategy and stopping criterion are both important for balancing recovery quality and retraining cost.

E. Cost Reduction of Resource Harvesting

To validate the effectiveness of the harvest-centric function scheduler, we compare the average monetary cost per retraining under various configurations. Given the similar keep-alive periods of workloads *Steady (High)* and *Bursty*, we select three representative workloads for analysis: Periodic, Steady (Low), and Sporadic, with keep-alive periods of 0.53 seconds, 1.22 seconds, and 2.01 seconds, respectively. We perform simulation experiments using AWS Lambda, a commercial cloud serverless platform. We test the average monetary cost

TABLE IV: Fine-grained edge-cloud labeling latency under different batch sizes. All values are in ms.

Batch	Upload	Inference	Download	End-to-end
4	24.48	11.53	6.76	42.77
16	56.40	18.26	8.72	83.38
64	139.06	85.81	8.48	233.35

within 24 hours and obtain the experimental results in Fig. 18 using Eq. 3, where $r = 1.67 \times 10^{-6}$ USD/GB/s. $y < 0$ indicates the cost saved by harvesting resources, while $y > 0$ represents the additional cost incurred for retraining, relative to the default cost constraints $\mathcal{B}$.

As depicted in Fig. 18, the average monetary cost savings achieved by FaaS Learner increase with the extension of the keep-alive period. This trend can be attributed to FaaS Learner's primary reliance on resources within the keep-alive period for model retraining, thereby allowing it to harvest more resources as the keep-alive period lengthens. Specifically, FaaS Learner achieves cost savings of 33%, 28%, and 143%, respectively, compared to AI-F, AT-F, and Gecko. Furthermore, in the more complex ADPP application, Gecko incurs substantially higher expenditure because more models must be retrained. Under *Periodic* and *Steady (Low)* workloads, AT-F and AI-F achieve greater cost savings than FaaS Learner by harvesting a larger share of resources. This advantage disappears under the *Sporadic* workload.

F. End-to-End Edge-Cloud Overhead

We further evaluate the end-to-end overhead of the edge-cloud pseudo-labeling pipeline. We use ResNeXt-101 as a representative golden model, configured with an input resolution of 224×224 and batch sizes of 4, 16, and 64. Table IV summarizes the average latency of video-frame upload, cloud-side golden-model inference, pseudo-label download, and the corresponding end-to-end latency. The results indicate that cloud-assisted labeling incurs additional but well-bounded overhead, while maintaining end-to-end latency at the millisecond level. Since data drift is typically gradual and FaaS Learner uses second-level detection windows, such latency does not invalidate the retraining signal for subsequent frames within the same local drift period.

G. Hyperparameters analysis

The hyperparameters α and β . As discussed in Sec. V-A, the hyperparameters α and β dictate the speed at which the window size diminishes. We set $a_{req} = 0.8$ by default for the system's drift detection mechanism following [22], [23]. In addition, we set the maximum detection window size $|w_{max}| = 50$ following AdaInf [7]. Based on Eq. 5, we can derive the values of parameters α and β by establishing boundary conditions. Specifically, when the inference accuracy $x = 1$, the desired detection window $|w_k|$ should approach its maximum value of 50, meaning the *Sigmoid* function's output approaches 1. Conversely, when the inference accuracy $x = 0.8$ (i.e., the constraint a_{req}), the desired window $|w_k|$

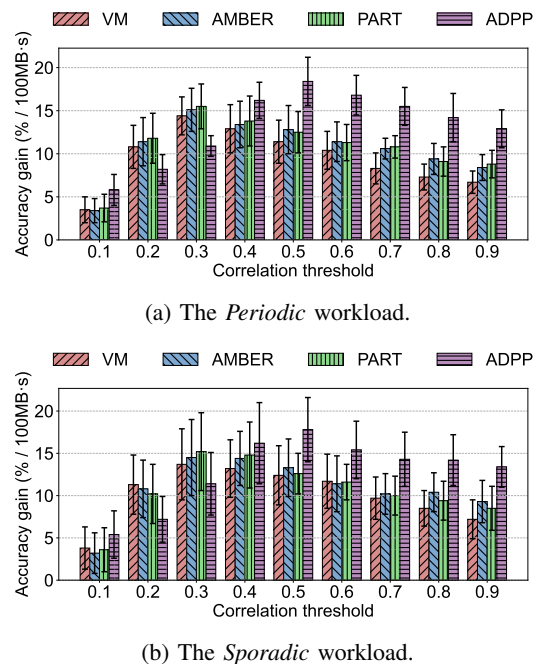


Fig. 19: The analytics of accuracy gain across different applications and workloads under various correlation thresholds.

should shrink rapidly to near 0, meaning the *Sigmoid* output approaches 0. By solving the system of equations formed from these boundary conditions, we obtain the calculated values $\alpha = 45.95$ and $\beta = -41.35$.

The correlation threshold λ . We analyze the impact of the correlation matrix on accuracy gain across different applications and workloads under various correlation thresholds. As shown in Fig. 19, the accuracy gain peaks for applications VM, AMBER, and PART when the threshold is set to 0.3, while ADPP achieves its optimal performance at a threshold of 0.5. As the threshold increases from 0.3 or 0.5 to 0.1, only highly correlated model pairs are selected for retraining, resulting in limited accuracy gain. Conversely, when the threshold further increases from 0.3 or 0.5 to 0.9, more model pairs are considered correlated and included in retraining. This leads to intensified resource competition under constrained conditions, which subsequently reduces accuracy gain. Moreover, the small variation in accuracy gain across different workloads indicates that the threshold selection is robust to workload changes by comparing Fig. 19a and 19b. Therefore, we adopt the threshold that maximizes retraining accuracy gain for filtering the correlation matrix.

The cost constraint $\mathcal{B}$. We analyze the monetary cost constraint on average accuracy gain and accuracy recovery time across different applications and workloads. As illustrated in Fig. 20, the VM, AMBER, PART, and ADPP applications achieve their highest accuracy gain when the cost constraint threshold is set to 1.8, 1.2, 1.2, and 0.6, respectively. While increasing the threshold reduces the accuracy recovery time, excessively high thresholds lead to ineffective retraining due to correlated drift, consequently diminishing accuracy gain. For more complex applications like ADPP, a lower cost constraint

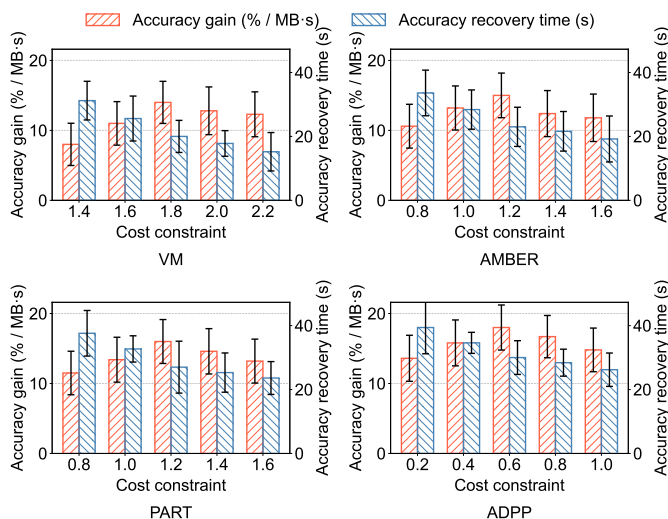


Fig. 20: The analysis of cost constraint threshold.

threshold suffices to achieve efficient retraining, owing to the greater availability of harvestable resources.

VII. RELATED WORK

Continual Learning for Video Analytics. Continual learning [24], [25] enhances the accuracy of video analytics by continuously retraining models with new data samples, allowing them to adapt to the evolving dynamics of video content and environmental changes. LifeLearner [26] proposes a hardware-aware meta continual learning on resource-constrained embedded systems that drastically optimizes system resources while ensuring high accuracy. For detecting data drift, AdaTrigger [13] and AdaEvo [20] design keyframe and adaptive frame sampling strategies, respectively. AdaInf [7] defines the impact degree of data drift for each model and updates the directed acyclic graph within a fixed time window (50s). MOCHA [8] proposes a mobile-cloud collaborative continuous adaptation framework to combat the degraded performance during adaptation and delayed reactions to environment shifts. AVA [27] studies multi-edge collaborative data acquisition for continuous video analytics, using reinforcement learning to select informative drift frames across edge nodes to reduce labeling and retraining costs. Regarding optimizing retraining strategies, RECL [1] improves continual learning efficiency by integrating model reuse and model retraining. Ekya [9] configures resources for retraining and inference through a scalable scheduler.

Many of the methods mentioned above initially detect data drift before retraining affected models. However, these methods do not take into account the correlation impact between inference and retraining across multiple models. In contrast, FaaS Learner first detects data drift, then removes those models that are not sensitive to accuracy improvement through analyzing the correlation between multiple models, and finally performs selective retraining, thereby achieving efficient continual learning.

Serverless Computing for Video Analytics. The serverless computing [28], [29] provides a detailed, on-demand

resource allocation approach for video analytics. It employs adjustable parameters for individual operations to enhance video analytics, such as sampling rate, batch size, and hardware selection. Llama [5] and Charmseeker [30] develop an automated resource configuration system for distributed computing resources. Wang *et al.* [31] manage costs by modifying particular sets of content, models, and resources. Nonetheless, these frameworks do not emphasize enhancing the accuracy of applications, particularly in addressing data drift issues. To tackle the mentioned concerns, Gecko [21] introduces a function instance to facilitate retraining for models encountering data drift. Nevertheless, the retraining tasks in Gecko utilize excessive resources, leading to an imbalance between resource usage and accuracy.

In contrast, FaaS Learner proposes a harvesting-centric retraining scheduling strategy to support retraining by harvesting idle resources in serverless functions, thereby achieving efficient accuracy and resources.

VIII. DISCUSSION

Application Service-Level Objectives. We set a default accuracy constraint of 0.8 for drift detection, following prior work [22], [23]. This level delivers useful results in real-time video analytics while allowing room for efficient retraining, and users can adjust it as needed. For memory, we assign a default allocation of 2GB of memory resources to each serverless function. This allocation ensures the stable operation of most deep learning models during inference.

The Impact of Data Drift Types on Detectors. Due to the high similarity between video frames, data drift is typically characterized as gradual drift [8], [1], [21], [7]. FaaS Learner is primarily designed to address gradual drift. Nonetheless, the architecture of FaaS Learner is sufficiently flexible to adapt to other types of data drift by updating the distribution functions and detection frequency within the data drift detector. For instance, in response to sudden drift [20], FaaS Learner dynamically adjusts the hyperparameters α and β to enhance the sensitivity of detection, thereby enabling a swift response to abrupt changes in data distribution.

Addressing Catastrophic Forgetting in FaaS Learner. Catastrophic forgetting in continual learning has been extensively studied in prior work [32], [33]. These techniques focus on preserving historical knowledge during continual updates. FaaS Learner addresses a different problem: when and which model to retrain under constrained resources. Its objective is to restore accuracy to the required SLO level as quickly as possible under a given budget. Therefore, FaaS Learner is orthogonal to catastrophic-forgetting mitigation methods and remains compatible with them as future extensions.

Cloud-assisted Pseudo Labeling. Cloud-side inference with a golden model handles short-term drift but incurs persistent latency, bandwidth, and data-exposure costs under long or frequent drift, violating edge SLOs. Hence, in FaaS Learner, the golden model provides only pseudo labels for edge-side retraining, not online inference. Additionally, while golden-model labels are generally accurate, imperfect pseudo labels may still appear in long-term deployments and

may accumulate errors during continual updates. Existing confidence-based filtering methods [34] can be seamlessly integrated into FaaS Learner as a retraining-data preparation step between golden-model pseudo labeling and edge-side retraining, mitigating this risk by selecting only pseudo-labeled samples whose golden-model confidence exceeds a predefined threshold. We leave the systematic design and evaluation of confidence-aware pseudo-label selection as future work.

IX. CONCLUSION

In conclusion, FaaS Learner provides a framework that not only maintains high accuracy in the face of data drift but also optimizes resource usage. By selectively retraining models based on their correlation in the multi-model application and leveraging idle serverless resources, FaaS Learner delivers a sustainable solution for video analytics. Furthermore, we conduct comprehensive evaluations of FaaS Learner using real-world datasets to assess its performance in multi-model applications. The experimental results demonstrate that FaaS Learner outperforms state-of-the-art approaches, achieving significant accuracy improvements while enhancing resource efficiency by reusing otherwise idle keep-alive memory resources for retraining.

REFERENCES

- [1] 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)*, 2023, pp. 917–932.
- [2] L. Mi, W. Wang, W. Tu, Q. He, R. Kong, X. Fang, Y. Dong, Y. Zhang, Y. Li, M. Li *et al.*, "Empower vision applications with lora lmm," in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 261–277.
- [3] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [4] Kaggle, "Kaggle: Your home for data science," 2023, accessed: 2024-10-07. [Online]. Available: <https://www.kaggle.com/>
- [5] F. Romero, M. Zhao, N. J. Yadwadkar, and C. Kozirakis, "Llama: A heterogeneous & serverless framework for auto-tuning video analytics pipelines," in *Proceedings of the ACM symposium on cloud computing*, 2021, pp. 1–17.
- [6] D. Li, Z. Zhang, K. Yu, K. Huang, and T. Tan, "Isee: an intelligent scene exploration and evaluation platform for large-scale visual surveillance," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2743–2758, 2019.
- [7] S. S. Shubha and H. Shen, "Adainf: Data drift adaptive scheduling for accurate and slo-guaranteed multiple-model inference serving at edge servers," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 473–485.
- [8] M. Zhao, S. Liu, F. Wu, and G. Chen, "Responsive dnn adaptation for video analytics against environment shift via hierarchical mobile-cloud collaborations," in *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*, 2025, pp. 317–331.
- [9] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karanakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 119–135.
- [10] Y. Yang, L. Zhao, Y. Li, H. Zhang, J. Li, M. Zhao, X. Chen, and K. Li, "Infless: a native serverless system for low-latency, high-throughput inference," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 768–781.
- [11] H. Yu, H. Wang, J. Li, X. Yuan, and S.-J. Park, "Accelerating serverless computing by harvesting idle resources," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1741–1751.
- [12] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, p. 102907, 2020.
- [13] Y. Kong, P. Yang, and Y. Cheng, "Edge-assisted on-device model update for video analytics in adverse environments," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 9051–9060.
- [14] A. Suprem, J. Arulraj, C. Pu, and J. Ferreira, "Odin: automated drift detection and recovery in video analytics," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 2453–2465, Jul. 2020. [Online]. Available: <https://doi.org/10.14778/3407790.3407837>
- [15] AWS Lambda. (2024) Welcome to aws lambda. [Online]. Available: <https://aws.amazon.com/cn/lambda/pricing/>
- [16] Y. Zhang, I. Goiri, G. I. Chaudhry, R. Fonseca, S. Elnikety, C. Delimitrou, and R. Bianchini, "Faster and cheaper serverless computing on harvested resources," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 724–739.
- [17] P. Snyder, "tmpfs: A virtual memory file system," in *Proceedings of the autumn 1990 EUUG Conference*, 1990, pp. 241–248.
- [18] Y. Cui, C. Zeng, X. Zhao, Y. Yang, G. Wu, and L. Wang, "Sportsmot: A large multi-object tracking dataset in multiple sports scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9921–9931.
- [19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
- [20] L. Wang, Z. Yu, H. Yu, S. Liu, Y. Xie, B. Guo, and Y. Liu, "Adaevio: Edge-assisted continuous and timely dnn model evolution for mobile devices," *IEEE Transactions on Mobile Computing*, 2023.
- [21] L. Wang, X. Qu, J. Wang, G. Li, J. Wan, N. Zhang, S. Guo, and J. Xiao, "Gecko: Resource-efficient and accurate queries in real-time video streams at the edge," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 481–490.
- [22] Y. Zhang, X. Zhang, G. Ananthanarayanan, A. Iyer, Y. Shu, V. Bahl, Z. M. Mao, and M. Chowdhury, "Vulcan: Automatic query planning for live {ML} analytics," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 1385–1402.
- [23] N. Agarwal and R. Netravali, "Boggart: Towards {General-Purpose} acceleration of retrospective video analytics," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 933–951.
- [24] S. Choi, W. Kim, S. Kim, Y. In, S. Kim, and C. Park, "Dslr: Diversity enhancement and structure learning for rehearsal-based graph continual learning," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 733–744.
- [25] J. Cai, X. Wang, C. Guan, Y. Tang, J. Xu, B. Zhong, and W. Zhu, "Multimodal continual graph learning with neural architecture search," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1292–1300.
- [26] Y. D. Kwon, J. Chauhan, H. Jia, S. I. Venieris, and C. Mascolo, "Life-learner: Hardware-aware meta continual learning system for embedded computing platforms," in *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, 2023, pp. 138–151.
- [27] L. Zhang, G. Gao, H. Yin, and H. Zhang, "Multi-edge reinforced collaborative data acquisition for continuous video analytics by prioritizing quality over quantity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 1084–1092.
- [28] A. Fuerst and P. Sharma, "Faas-cache: keeping serverless computing alive with greedy-dual caching," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 386–400.
- [29] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: a survey of opportunities, challenges, and applications," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1–32, 2022.
- [30] M. Zhang, Y. Zhu, J. Liu, F. Wang, and F. Wang, "Charmseeker: Automated pipeline configuration for serverless video processing," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2730–2743, 2022.
- [31] Z. Wang, S. Zhang, J. Cheng, Z. Wu, Z. Cao, and Y. Cui, "Edge-assisted adaptive configuration for serverless-based video analytics," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 248–258.
- [32] A. Lewandowski, M. Bortkiewicz, S. Kumar, A. György, D. Schuurmans, M. Ostaszewski, and M. C. Machado, "Learning continually by spectral regularization," in *International*

- Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=Hcb2cgPbMg>
- [33] X. Gao, S. Dong, Y. He, Q. Wang, and Y. Gong, "Beyond prompt learning: Continual adapter for efficient rehearsal-free continual learning," in *European Conference on Computer Vision*, 2024, pp. 89–106.
 - [34] C. Fan, Z. Wang, D. Guo, and M. Wang, "Data-free quantization via pseudo-label filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 5589–5598.