SkyML: A MLaaS Federation Design for Multicloud-based Multimedia Analytics

Shuzhao Xie, Yuan Xue, Yifei Zhu, *Member, IEEE*, Zhi Wang†, *Senior Member, IEEE*

***Abstract*—The advent of deep learning has precipitated a surge in public machine learning as a service (MLaaS) for multimedia analysis. However, reliance on a single MLaaS can result in product dependency and a loss of better performance offered by multiple MLaaSes. Consequently, many enterprises opt for an intercloud broker capable of managing jobs across various clouds. Though existing works explore the efficient utilization of intercloud computational resources and the enhancement of intercloud data transfer throughput, they disregard improving the overall accuracy of multiple MLaaSes. In response, we conduct a measurement study on object detection services, which are designed to identify and locate various objects within an image. We discover that combining predictions from multiple MLaaSes can improve analytical performance. However, more MLaaSes do not necessarily equate to better performance. Therefore, we propose SkyML, a user-side MLaaS federation broker that selects a subset of MLaaSes based on the characteristics of the request to achieve optimal multimedia analytical performance. Initially, we design a combinatorial reinforcement learning approach to select the sound MLaaS combination, thereby maximizing user experience. We also present an ingenious, automated taxonomy unification algorithm to minimize human efforts in merging MLaaS-specific labels into a user-preferred label space. Moreover, we devise an optimized ensemble strategy to aggregate predictions from the selected MLaaSes. Evaluations indicate that our similarity-based taxonomy unification approach can reduce annotation costs by 90%. Moreover, real-world trace-driven evaluations further prove that our MLaaS selection method can achieve similar levels of accuracy with a 67% reduction in inference fees.**

***Index Terms*—Multimedia analysis, sky computing, machine learning services, multimedia clouds, taxonomy unification**

I. Introduction

DEEP learning has recently gained substantial attention across diverse sectors. It has not only attracted industries (e.g., Robotics [1]) closely related to computer science but also penetrated other sectors (e.g., E-commerce [2]) as companies incorporate deep learning into their product pipelines. Nonetheless, companies in vertical industries often struggle with a shortage of experts adept at maintaining deep learning services. This typical predicament leads them to lease deep learning services offered by major service providers. For instance, ZOZO [2] leverages Amazon Rekognition [3] to expedite the reviewing process of user-uploaded images by up to 40% [4]. Despite this, the drawback of relying solely on one cloud service is its high instability, leading many companies to utilize multiple cloud services to enhance stability and gains.

To make full use of multiple clouds, Weinman [5] proposed a standard to IEEE for portability, interoperability, and federation among cloud providers. However, this standard is less practical as it necessitates standardization of interfaces by every cloud provider. Later, sky computing [6] proposed to use a compatibility layer to handle different interfaces. Inspired by this design proposition, SkyPilot [7] is proposed for easily and cost-effectively running workloads on any cloud VMs. SkyPlane [8] improves intercloud data transfer speed using indirect paths. Despite their effectiveness, these solutions have not covered a popular cloud service – MLaaS (i.e., ChatGPT [9] operates as a machine learning service). The use of multiple MLaaS providers is also known as MLaaS federation. Such a design enables users to enjoy higher-quality cloud service inference results at a more affordable price. For single-label classification services, FrugalML [10] studies to select the best MLaaS from multiple ones. Additionally, FrugalMCT [11] analyzed whether to request an additional MLaaS to improve the accuracy based on the prediction of a base MLaaS. Above all, these works are based on two questionable assertions: 1) Adding more MLaaSes can lead to at least the same accuracy; 2) Different MLaaSes use the same vocabulary to represent predictions.

To verify these assertions, we first investigate MLaaS-related measurement studies. However, most of these studies focus on irrelevant MLaaSes such as white-box services [12], [13], ML training platforms [14], or out-of-date MLaaSes [15]. The only related work [16] focuses on collecting inference results to explore whether and when an MLaaS was updated. Hence, we have to conduct an independent measurement study. Considering the task-specific nature of MLaaSes, we primarily focus on evaluating the performance of object detection services. Object detection [17] is a versatile vision task that aims to detect the locations and categories of all objects in an image, with applications across diverse multimedia domains [18]–[20], such as video analytics [21], [22] and autonomous driving [23], [24]. The detection result consists of a list of

† Zhi Wang is the corresponding author.

Shuzhao Xie is with Shenzhen International Graduate School, Tsinghua University, China (e-mail: xsz24@mails.tsinghua.edu.cn). Yuan Xue is with Zhongguancun Laboratory, China (e-mail: yxue2021@gmail.com). Yifei Zhu is with University of Michigan-Shanghai Jiao Tong University Joint Institute in Shanghai Jiao Tong University, China (e-mail: yifei.zhu@sjtu.edu.cn). Zhi Wang is with Shenzhen International Graduate School & Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, China (e-mail: wangzhi@sz.tsinghua.edu.cn).

This work is supported in part by National Key Research and Development Project of China (Grant No. 2023YFF0905502), National Natural Science Foundation of China (Grant No. 62472249), and Shenzhen Science and Technology Program (Grant No. JCYJ20220818101014030). The work of Yifei Zhu is supported by the National Natural Science Foundation of China (Grant No. 62302292). We thank Chen Tang, Jiahui Ye, Shiji Zhou, and Wenwu Zhu for their help.

multimodal tuples. Each tuple contains information about a detected object, including its class, represented by a predicted category name, its location defined by a quaternion bounding box, and a confidence score indicating the accuracy of the prediction. Based on the measurement study, we confirm that each MLaaS possesses different sweet-spot categories. For example, AWS's "monitor" usually represents the same object as Azure's "television". Moreover, we observe that naively assembling all MLaaSes can introduce extra false-positive results. For example, the accuracy of ensemble predictions of AWS and Azure outperforms the ensemble predictions of AWS, Azure, and Google for some input samples. To fully leverage the potential of these MLaaSes, it is advantageous to harness the expertise of multiple service providers and carefully select the appropriate combination of MLaaSes.

Based on the analysis above, we summarize three pain points in realizing MLaaS federation. First, different MLaaS providers use varying taxonomies. Unifying thousands of MLaaS-specific labels is error-prone and time-consuming, so we need a manual-free algorithm to unify the description languages used in various MLaaSes. Second, the computational complexity associated with selecting the suitable provider set is heightened by the combinatorial nature of the problem, especially when dealing with resource-constrained edge devices. The extensive list of providers and the exponentially increasing number of choices render the brute-force approach impractical and non-scalable for real-world scenarios. Third, after receiving analytic results from various service providers, further design is required to merge these results to offer optimal aggregated results efficiently.

To tackle these challenges, we introduce SkyML, a federated MLaaS broker that attains optimal performance in multimedia analytics. Our framework comprises MLaaS combination selection, taxonomy unification, and prediction ensemble. Specifically, we propose a combinatorial reinforcement learning (RL) approach to address the provider selection problem. To overcome the computational challenge of selecting the right MLaaS subset, we employ a nearest-neighbor algorithm to map continuous action spaces to discrete binary action spaces within extensive combinatorial domains. The taxonomy unification part maps MLaaS-specific labels onto user-defined labels based on similarities calculated from ground truth and MLaaS predictions. In the ensemble part, we optimize the total analytic results by employing the affirmative voting strategy and weighted box fusion to reduce the impact of outliers. We evaluate the performance of both the taxonomy unification method and the RL-based provider selection algorithm by conducting extensive experiments with real trace-driven data.

In summary, our contributions are:

- Our measurement studies on prominent MLaaS providers uncover significant discrepancies between them and highlight the immense potential of MLaaS federation to enhance overall analytic performance.
- We formulate the MLaaS federation problem as a combinatorial provider selection problem and establish its NP-hardness. Subsequently, a combinatorial reinforcement learning-based approach is proposed to solve the problem and achieve optimal accuracy.

TABLE I: Average precisions (AP) of different MLaaSes.

Provider	mAP	AP_{50}	AP_{75}	Number of Labels
AWS	18.81	28.88	20.84	75
Azure	15.10	24.38	16.14	163
GCP	16.23	23.03	18.12	120
COCO	-	-	-	80

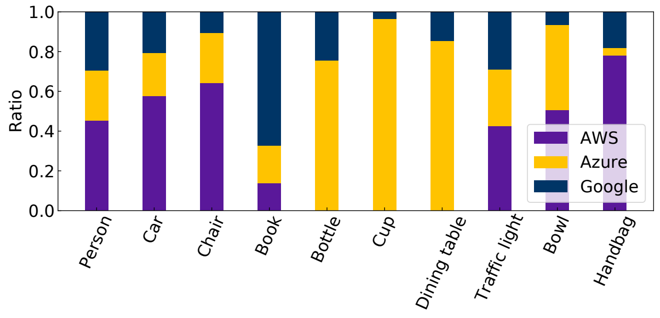


Fig. 1: Comparison of AP across multiple MLaaS providers for the top 10 most frequent categories.

- Similarity-based taxonomy alignment and efficient voting strategies are proposed to achieve vocabulary unification among providers and aggregate the final results.
- Simulations based on real-world traces show that our framework can achieve a 67% reduction in inference costs while maintaining comparable accuracy to other benchmark approaches. Additionally, our similarity-based taxonomy alignment reduces the labor cost by 90%.

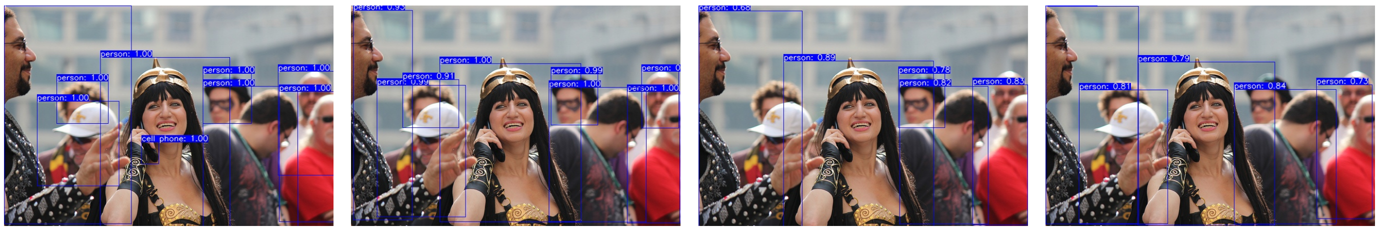
This paper extends our previous work [25] with several improvements. First, we have generalized the framework by introducing the MLaaS Federation within the context of sky computing. Second, in Sec. III, we update the problem formulation and add complexity analysis. Third, we have devised an automated approach to address the taxonomy alignment requirement arising from using different taxonomies by multiple cloud providers. Finally, we have enhanced the evaluation to provide deeper insights and discussed the limitations.

The rest of this paper is structured as follows. Sec. II proposes the measurement study. Sec. III formulates the provider selection problem and explains the soundness of the combinatorial RL approach. Sec. IV introduces the three parts of SkyML. We evaluate SkyML in Sec. V. Sec. VI and Sec. VII present the related work and the limitations, respectively. Finally, Sec. VIII provides the conclusion.

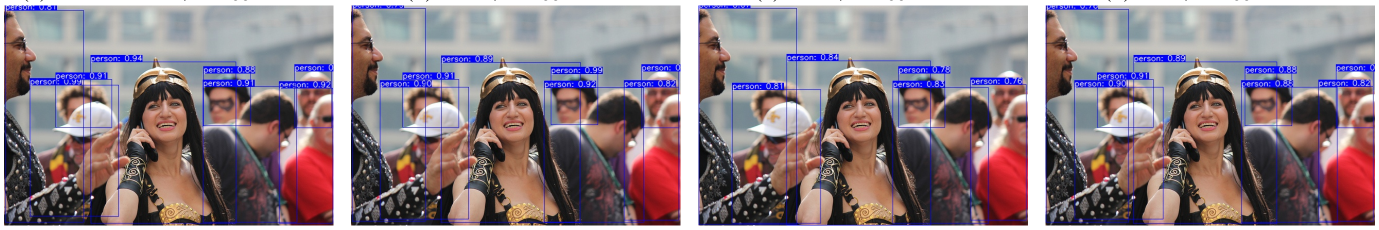
II. Measurement & Motivation

This section provides an analysis of the latency and accuracy of three leading MLaaSes: AWS Rekognition [3], Azure Computer Vision [26], and Google Cloud Vision AI [27], and demonstrates the significant benefits of multi-MLaaS federation along with the feasibility of achieving this. We use AWS, AZU, and GCP to represent these three MLaaSes.

We conducted the measurements in 2021. Virtual machines (VMs) located in Singapore and the USA were rented as clients to request these MLaaSes. These VMs have the same configuration. We requested these MLaaSes via Python SDK and captured the TCP packet by `tcpdump`. We focus on



(a) Ground truth, AP_{50} : 1.00 (b) AWS, AP_{50} : 0.64 (c) AZU, AP_{50} : 0.56 (d) GCP, AP_{50} : 0.56



(e) AWS+AZU, AP_{50} : 0.71 (f) AWS+GCP, AP_{50} : 0.69 (g) AZU+GCP, AP_{50} : 0.67 (h) All MLaaSes, AP_{50} : 0.68

Fig. 2: Visualization of the predictions of object detection services. We display the ensemble predictions and the corresponding metrics under different MLaaS combinations.

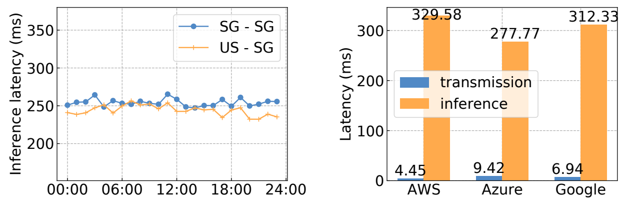


Fig. 3: *Right*: The inference latency in both routes (SG-SG and US-SG) is similar within a 24-hour period, indicating that our division of the total latency is correct. *Left*: The transmission latency is significantly lower than inference latency.

object detection as it is versatile and has applications across diverse multimedia domains. The accuracy metrics used by object detection are mean average precision (mAP), mAP with intersection over union (IoU) threshold 50% (AP_{50}), and mAP with IoU threshold 75% (AP_{75}) [28]. We chose the COCO Val 2017 [29] dataset to test the accuracy. In the case of AWS and AZU, we chose Singapore as the cloud service region as the closest region always leads to the shortest latency. In contrast, GCP automatically selects the MLaaS region for users, preventing users from selecting their preferred region. In this context, the term "region" refers to the location of the MLaaS, while the "location" is the location of the user device.

A. Why We Need SkyML

Tab. I compares the AP of AWS, AZU, and GCP on the COCO dataset. Though AZU demonstrates the lowest average performance, it does not imply poor performance by AZU in all categories within the dataset. We select the top 10 most frequent categories within the COCO dataset and compute their AP_{50} scores of AWS, AZU, and GCP, respectively. Fig. 1 shows that AWS performs the best in categories such as "person", "chair", "car", and "handbag". AZU performs best in categories such as "cup", "bottle", and "dining table" whereas AWS failed to identify any objects in these categories. These observations demonstrate that *the optimal MLaaS provider depends on the characteristics of the input.*

Now, we reveal the benefits of the MLaaS federation. Following the ensemble strategies that will be introduced in Sec. IV-D, we calculate the AP_{50} of AWS, AZU, GCP, and their combinations[1], as demonstrated in Fig. 2. We observe that the ensemble predictions from three MLaaSes yield higher AP_{50} values than predictions from a single provider. However, upon comparing Fig. 2e and Fig. 2h, we find that the ensemble predictions of AWS and AZU ($AP_{50} = 0.71$) are superior to the ensemble predictions of the three cloud providers ($AP_{50} = 0.68$). *These findings suggest that including additional MLaaSes can lead to higher accuracy. However, the addition of more MLaaSes does not imply a guarantee of higher accuracy.*

B. Why SkyML is Possible

Typically, requesting a cloud service consists of transmission and inference latency. The transmission latency relies on the input data size, bandwidth, and the round-trip time (RTT) between the client and the MLaaS. The inference latency is determined by the cloud service, independent of network conditions. The time to send the minuscule response data over the network is negligible. As shown in Fig. 3, our measurement validates that the MLaaS has similar latency properties.

Considering the scenario of requesting multiple MLaaSes, n inputs are injected into the network. The initial route for these packets is the path between the user device and the router, such as 5G WiFi. With a peak upload speed of 10Gbps [31], the 5G network can transmit at least 10 images in 1 ms. Hence, the initial route is not the bottleneck of transmission latency. Subsequently, these packets simultaneously follow different paths to reach their respective MLaaS providers. Therefore, the overall transmission latency can be determined by selecting

[1]The data in our paper reflected the SOTA performance for object detection by the time we conducted the measurement. Deformable DETR [30] was used with an AP_{50} of 65.2% on MSCOCO.

TABLE II: Notations used in SkyML's description.

Notation	Description
$\mathcal{I}$	Set of input.
i	An input ($i \in \mathcal{I}$).
$\mathcal{M}$	Set of MLaaS providers.
m	An MLaaS provider ($m \in \mathcal{M}$).
ϕ	A binary vector ($\phi \in \{0,1\}^{\|\mathcal{M}\|}$ and $\| \phi \| \neq 0$). $\phi_m = 1$ iff the MLaaS provider m is selected.
c_m	Cost to request MLaaS provider m for input i.
P_m	Prediction from MLaaS provider m for input i.
$\mathcal{P}_{all}$	Set of predictions from all avaialble MLaaSes.
$\mathcal{P}$	Set of predictions from MLaaS providers selected by ϕ for input i.
$\mathcal{C}$	Budget for processing the input i.
$E(\cdot)$	Function to ensemble predictions from MLaaSes.
$A(\cdot)$	Function to score the accuracy of the final prediction.

the maximum transmission lag among the n inputs. During the inference phase, the n MLaaSes simultaneously predict the results, resulting in an inference latency equivalent to the maximum inference time among the n MLaaSes. Based on the above analysis, we conclude that the time required to request multiple MLaaSes is almost the same as requesting one. *In summary, requesting multiple MLaaSes is practically feasible.*

III. Problem Formulation

MLaaS refers to a function that receives multimedia input data, such as an image, a sequence of text, or a slice of speech, and returns the predicted output. Examples of MLaaS include categorizing the image, translating text, or generating text based on the input speech. The specific input and output formats vary depending on the type of MLaaS. Here, our formulation is applicable to various tasks.

Let $\mathcal{I} = \{I_1, I_2, ..., I_T\}$ be a set of inputs to be processed by a set of $\mathcal{M}$ MLaaS providers. The goal is to determine the optimal selection of MLaaS providers that maximize accuracy for all input data while ensuring the cost is less than the budget. This selection is based on the notations outlined in Tab. II. The formal formulation of the MLaaS federation problem (Ω) for each input is:

$$\begin{aligned} \max \quad & A(E(\phi, \mathcal{P}_{all})), \\ s.t. \quad & \sum_{m \in \mathcal{M}} c_m \phi_m \leq \mathcal{C}, \\ & \phi_m \in \{0,1\}. \end{aligned} \tag{1}$$

Here $\mathcal{P}_{all}$ represents the predictions of all available MLaaS providers, $E(\cdot)$ is the method to ensemble returned predictions, $A(\cdot)$ is the function to calculate the accuracy, $\mathcal{C}$ is the budget to request the MLaaSes for current input, c_m is the cost to invoke the m-th MLaaS, and ϕ_m indicates whether the m-th MLaaS selected.

The formulation of Ω encompasses two hidden issues: 1) P_{all} is not known in advance; 2) even if P_{all} were known in advance, what method should be used to calculate ϕ? Here, we first assume P_{all} is known and determine the method for calculating ϕ through complexity analysis. Then, we analyze how to provide the model with prior knowledge about P_{all}.

Complexity Analysis. For a given input data, assuming $\mathcal{P}_{all} = \{P_1, P_2, ..., P_{|\mathcal{M}|}\}$ are known in advance, P_m can be represented by a matrix with size of (row_m, d), where row_m represents the number of predictions from the m-th cloud service for this input data. d represents the dimension of each inference result, which is related to the task of the cloud inference service, such as $d = 6$ for object detection service, including category, confidence, and the coordinates of the bounding box.

Then, the inference results of selected MLaaSes can be concatenated as a matrix with a size of $(\sum_{m=1}^{|\mathcal{M}|} \phi_m row_m, d)$, denoted as $\mathcal{P} = [\phi_1 P_1^\top, \phi_2 P_2^\top, ..., \phi_{|\mathcal{M}|} P_{|\mathcal{M}|}^\top]^\top$. The vectorized form of $\mathcal{P}$ is $\phi^\top P_{all}$. According to Sec. IV, the ensemble function $E(\cdot)$ includes two steps, i.e., label mapping and result pruning. Label mapping refers to replacing the cloud service labels with user space labels, and those that cannot be replaced are discarded; result pruning refers to deleting duplicate inference results. Hence, both label mapping and result pruning are equivalent to a filter, which can be represented by multiplying $\mathcal{P}$ with a 0-1 filter matrix. Here, $R_{E1}(\mathcal{P})$ and $R_{E2}(\mathcal{P})$ represent the filter matrices of the two steps above, respectively. Therefore, the ensemble function $E(\cdot)$ can be expressed as:

$$E(\phi) = \phi^\top P_{all} R_{E1}(\phi^\top P_{all}) R_{E2}(\phi^\top P_{all}). \tag{2}$$

Hence, $E(\phi)$ is at least a polynomial of degree 3 in ϕ.

Next, we analyze the complexity of the accuracy function $\mathcal{A}(\cdot)$, which is task-specific. Here, we take the object detection task as an example, and thus, the $\mathcal{A}(\cdot)$ becomes the function to calculate mAP. Let $\mathcal{P}_E = E(\phi)$. We simplify $A(\cdot)$ to the process of calculating AP_{50}. We first sort $\mathcal{P}_E$ by confidence. After that, based on $\mathcal{P}_E$, the ground truth annotations G, and the IoU threshold, we split detection results into three types, including true positive (TP), false positive (FP), and false negative (FN). As FN refers to the ground truth object that was not detected, there are virtually only two types of inference results, i.e., TP and FP. In short, the process of obtaining TP and FP is equivalent to performing element-wise discrimination on $\mathcal{P}_E$, which can be represented as $R_{TP}(\cdot) : \mathbb{R}^{d \times 6} \rightarrow \{0,1\}^d$ and $R_{FP}(\cdot) : \mathbb{R}^{d \times 6} \rightarrow \{0,1\}^d$. Applying $R_{TP}(\cdot)$ and $R_{FP}(\cdot)$ to each inference output yields two binary vectors, V_{TP} and V_{FP}. Before calculating the precision vector and recall vector, we obtain the CV_{TP} and CV_{FP} by accumulating V_{TP} and V_{FP}, respectively. This accumulation process can be expressed as $CV = V^\top U$, where U is a square matrix with upper triangle elements equal to 1. We calculate the precision vector by:

$$Pre = \frac{CV_{TP}}{CV_{TP} + CV_{FP}}, \tag{3}$$

and calculate the recall vector Rec by $CV_{TP}/|G|$, where $|G|$ denotes the number of ground truths. Finally, the AP_{50} value can be calculated with:

$$AP_{50} = Inter(Pre)^\top (R_{rec}(Rec) - Rec), \tag{4}$$

where process of R_{rec} is described in Algo. (1). And $Inter(Pre)$ is the maximum precision corresponding to the recall value greater than the current recall value. In simple terms, it is the maximum precision value to the right. The

Algorithm 1 Workflow of R_{rec}.

```
Require: Recall Vector $Rec$.
1: $n \leftarrow$ length of $Pre$;
2: for $i = n-2$ to 0 do
3:     $Rec_i \leftarrow Rec_{i+1}$;
4: end for
5: $Rec_{n-1} \leftarrow 1$;
6: return $Rec$.
```

value of AP_{50} is the integration of the interpolated precision-recall curve in $[0,1]$, which is at least a linear transformation. We eliminate the $Inter(\cdot)$ step for convenience. Finally, substituting $\mathcal{P}_E$, $R_{TP}(\cdot)$, and $R_{FP}(\cdot)$ into Eq. (4) yields the relationship between $A(\cdot)$ and $\mathcal{P}_E$, as shown in Eq. (5):

$$A(\mathcal{P}_E) = \left[\frac{R_{TP}(\mathcal{P}_E)}{(R_{TP}(\mathcal{P}_E) + R_{FP}(\mathcal{P}_E))}\right] \times \left[R_{rec}\left(\frac{R_{TP}^{\top}(\mathcal{P}_E)U}{|G|}\right) - \frac{R_{TP}^{\top}(\mathcal{P}_E)U}{|G|}\right], \tag{5}$$

where $A(\mathcal{P}_E)$ is at least a polynomial of degree 1 in $\mathcal{P}_E$.

In summary, the object function $A(E(\cdot))$ is at least a cubic term of ϕ. Meanwhile, assuming all the predictions from $\mathcal{M}$ providers for input i are known, we can reduce the above problem to a multidimensional quadratic knapsack problem, an NP-hard problem [32], [33]. Specifically, this optimization problem aims to generate binary options (ϕ) to maximize overall accuracy while satisfying the cost constraint. In practice, however, getting all the predictions ($\mathcal{P}_{all}$) before the request is explicitly *infeasible*.

Prior Knowledge of $\mathcal{P}_{all}$. To inject the prior knowledge of $\mathcal{P}_{all}$ into the SkyML agent, we first profile all available MLaaSes with a pre-prepared request dataset. Then, we try building a precise profile dataset, with each sample as <I, ϕ>, and employing the supervised learning algorithm to inject the prior knowledge into the agent. However, it is computationally expensive to obtain the precise profile dataset since it involves evaluating the accuracy and cost of all possible combinations of $\mathcal{M}$ MLaaSes, which grows exponentially with the number of MLaaSes. Furthermore, the MLaaS model and the corresponding training dataset undergo updates as deep learning algorithms evolve, posing challenges for achieving global optimization, particularly in long-term considerations.

Fortunately, RL provides an alternative perspective for this problem. The key is that RL can update the model parameters to maximize the expected rewards with restricted trials, far less than $2^{|\mathcal{M}|}-1$. Hence, the training complexity of RL is far less than that of supervised learning methods. Additionally, RL agents are typically small and can be trained on edge devices, making them suitable for addressing the challenges posed by inconsistent versions of MLaaSes.

However, if we leverage RL, a difficult issue of representing a huge discrete action space emerges. The reason is that if the selection of MLaaSes (ϕ) becomes the action, then the size of this discrete action space becomes exponential to the number of available MLaaSes. Fortunately, current works [34]–[36] on solving combinatorial RL algorithms provide feasible solutions for dealing with a large discrete action space. Considering these unique constraints, we present a combinatorial RL-based approach to resolve the provider selection problem.

IV. SkyML Framework

In this section, we introduce SkyML, a broker designed to adaptively select the optimal combination of MLaaS providers for various inputs. It also automatically aggregates predictions from the chosen MLaaS providers to maximize analytical performance while minimizing costs. Our discussion begins with an introduction to the workflow of this broker. Subsequently, we delve into the provider selection process, taxonomy unification, and ensemble techniques.

A. Framework Overview

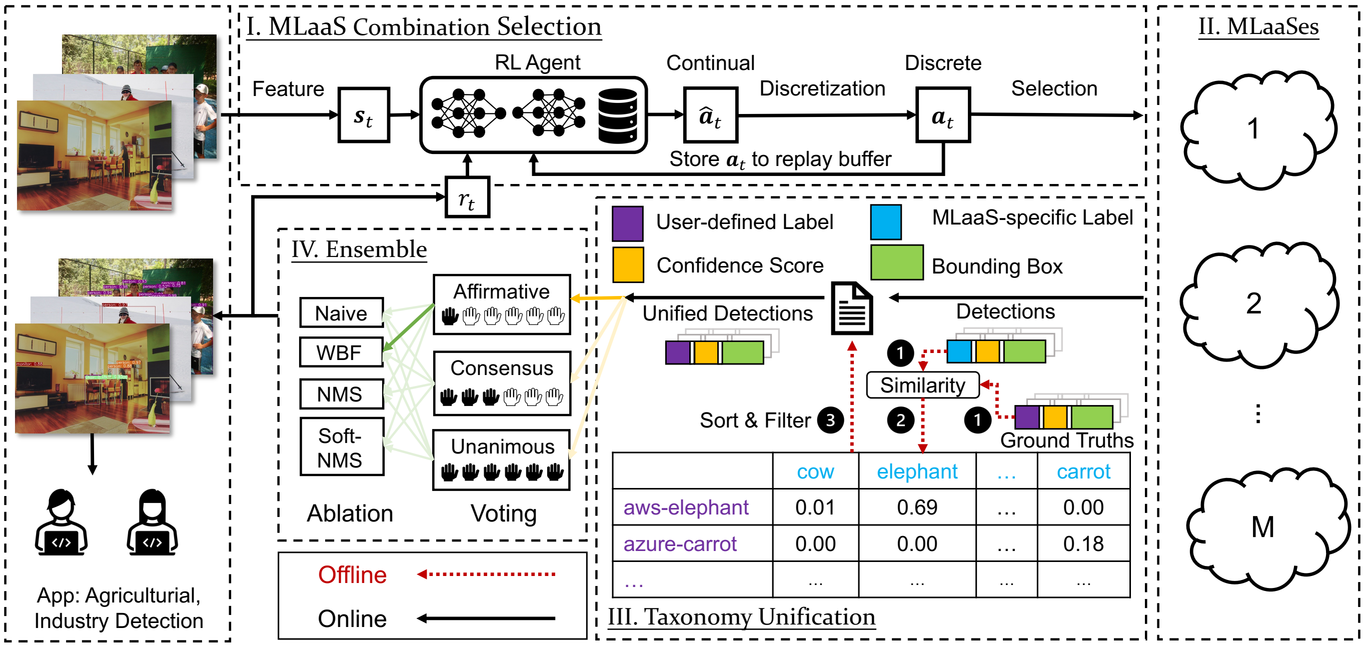
Before deploying SkyML, we will send a pre-prepared request dataset to all MLaaSes and record the returned predictions to construct a profile dataset. Subsequently, we use the profile dataset to train the RL agent, giving it some prior knowledge about the available MLaaSes.

After deploying SkyML, its workflow is illustrated in Fig. 4, a typical scenario of multimedia content analytics. In the provider selection part, the broker first extracts image features (referred to as state $\mathbf{s_t}$) at the edge-side client. Then, it generates a proto action $\hat{\mathbf{a}}_\mathbf{t}$ using the image features and the actor-network trained by the soft actor-critic (SAC) algorithm [37]. Subsequently, the broker maps this proto action into a discrete binary vector $\mathbf{a_t}$. The edge client sends requests to the providers selected by the action $\mathbf{a_t}$ and waits to receive all the predictions from the selected providers. The cloud providers, labeled as Cloud 1, 2, ..., and n, are the available MLaaS providers. Next is the taxonomy unification part. Since different MLaaS providers use different terms to represent the same category, the broker must identify and unify labels into a standardized form to meet users' requirements and ensure smooth operation in the subsequent ensemble part. The ensemble phase aims to eliminate duplicate predictions while retaining the correct ones. There are 12 pathways to choose from, and we ultimately select the Affirmative-WBF path based on our measurements in Sec. II. In this part, the broker also generates rewards r_t, which are stored in the replay buffer along with the binary action $\mathbf{a_t}$, the state $\mathbf{s_t}$, and the next state $\mathbf{s_{t+1}}$. After passing through the modules above, the final prediction is made on the image. The remainder of this section provides further details about the SkyML framework.

B. Provider Selection: A Combinatorial RL Approach

We first describe the state, action, and reward design for our provider selection method. Here, we want to select the most appropriate MLaaS subset to obtain the best accuracy at the minimum cost. Our primary goal is to achieve optimal analytic performance.

State. As shown in the top left of Fig. 4, we extract input features by a pre-trained MobileNet model, which is a commonly used model for image classification. This feature is the state $\mathbf{s}_t$ acquired from the environment at timestamp t, corresponding to the t-th input sample.



	cow	elephant	…	carrot
aws-elephant	0.01	0.69	…	0.00
azure-carrot	0.00	0.00	…	0.18
…	…	…	…	…

Fig. 4: Workflow of SkyML. The blue labels are user-defined labels. The purple ones are MLaaS-specific labels.

Action. In Sec. III, we use a binary vector ϕ to represent the possible combination of $||\mathcal{M}||$ MLaaSes. Here, the action a_t is defined similarly to ϕ. Then the size of action set $\mathcal{A}$ is $2^{||\mathcal{M}||}-1$. Thus, the action space of our provider selection problem is an exponential multiple of $||\mathcal{M}||$. Due to the difficulty of representing such a large discrete action space, we map the predicted action $\hat{\mathbf{a}}$ from continuous action spaces to an element in the discrete binary vector set $\mathcal{A}$:

$$\tau(\hat{\mathbf{a}}) = \arg\min_{\mathbf{a}\in\mathcal{A}} |\mathbf{a}-\hat{\mathbf{a}}|^2, \tag{6}$$

where τ is the nearest-neighbor mapping from the continuous space $\mathbb{R}^{||\mathcal{M}||}$ to the discrete binary vector set $\mathcal{A}$. It returns the action $\mathbf{a}$ that is the closest to the predicted action $\hat{\mathbf{a}}$ according to the l_2 distance. The action $\mathbf{a}$ will be stored in the replay buffer along with other data elements.

Reward. The reward function considers both the mAP and the inference cost. The mAP calculation requires using ground truth, as described in [28]. Nevertheless, not all inputs have ground truth, especially in the case of online inference. Therefore, we consider the ensemble prediction of all the MLaaS providers as the ground truth for the data collected during inference. Despite the suboptimal mAP of the ensemble prediction with all MLaaSes, it is commendable for achieving similar performance to all MLaaSes while incurring fewer inference fees. In summary, the reward is defined as:

$$r_t = \tanh(v_t + \beta c_t), \tag{7}$$

where v_t represents the AP_{50} of the prediction, c_t denotes the cost required to request the subset of MLaaS providers selected by action $\mathbf{a}_t$. β is a non-positive hyperparameter to ensure the selection with a lower cost. We set the v_t as -1 if none of the MLaaSes are selected.

We employ the Soft Actor-Critic (SAC) algorithm to train the agent. SAC is an off-policy actor-critic algorithm that operates within the maximum entropy. Algo. 2 outlines the training details for the RL agent. To begin with, SkyML initializes the necessary components. We use a fully connected network that contains two hidden layers to represent the networks mentioned above. We utilize the multivariate normal distribution to represent the probability density function (PDF) of policy π. Thus, we use two networks to fit the mean and variance of policy π. We use two soft Q-functions to address the positive bias in the policy improvement step, which is well-known for negatively impacting the performance of value-based methods [38]. Second, for each step, SkyML extracts the feature and selects the action $\hat{\mathbf{a}}$ with policy $\pi_\theta(\cdot|\mathbf{s})$, and then maps $\hat{\mathbf{a}}$ to a binary action $\mathbf{a}$. After that, SkyML sends the binary action into the environment to observe the done signal d, the reward r, and the next state $\mathbf{s}'$. It next stores $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', d)$ to replay buffer $\mathcal{B}$. During training, a batch of transitions is sampled from replay buffer $\mathcal{B}$ to update the agent. The target of Q-network is given by:

$$y(r,\mathbf{s}',d)=r+\gamma(1-d)\left(\min_{j=1,2} Q_{\phi_{targ,j}}(\mathbf{s}',\tilde{\mathbf{a}}')-\alpha\log\pi_\theta(\tilde{\mathbf{a}}'|\mathbf{s}')\right), \tag{8}$$

where $\tilde{\mathbf{a}}'$ is sampled from $\pi_\theta(\cdot|\mathbf{s}')$. Then we can update two Q-networks $Q_{\phi_i}, i=1,2$ by:

$$\nabla_{\phi_i}\frac{1}{|B|}\sum_{(\mathbf{s},\mathbf{a},r,\mathbf{s}',d)\in B}(Q_{\phi_i}(\mathbf{s},\mathbf{a})-y(r,\mathbf{s}',d))^2, \tag{9}$$

and update the policy by:

$$\nabla_\theta\frac{1}{|B|}\sum_{\mathbf{s}\in B}\left(\min_{i=1,2} Q_{\phi_i}(\mathbf{s},\tilde{\mathbf{a}}_\theta(\mathbf{s}))-\alpha\log\pi_\theta(\tilde{\mathbf{a}}_\theta(\mathbf{s})|\mathbf{s})\right), \tag{10}$$

where $\tilde{\mathbf{a}}_\theta(\mathbf{s})$ is a sample from $\pi_\theta(\cdot|\mathbf{s}')$. At last we update the target networks $Q_{targ,i}, i=1,2$ with:

$$\phi_{targ,i} \leftarrow \rho\phi_{targ,i} + (1-\rho)\phi_i. \tag{11}$$

Algorithm 2 Training the RL agent with SAC.

```
1: Initialize policy parameters $\theta$, Q-function parameters $\phi_1, \phi_2$, replay buffer $\mathcal{B}$, hyperparameter $\beta$;
2: Set target Q-function parameters equal to main parameters $\phi_{targ,1} \leftarrow \phi_1, \phi_{targ,2} \leftarrow \phi_2$;
3: for time=1,...,$T$ do
4:     Observe an input, extract state $\mathbf{s}$ and select action $\hat{\mathbf{a}} \sim \pi_\phi(\cdot|\mathbf{s})$;
5:     Get the nearest binary vector $\mathbf{a}$ of $\hat{\mathbf{a}}$ in $l_2$ distance;
6:     Request the MLaaS providers selected by $\mathbf{a}$;
7:     Store $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', d)$ to replay buffer $\mathcal{B}$;
8:     if it's time to update then
9:         for j in range(update times) do
10:            Randomly sample a batch of transitions $B = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', d)\}$ from $\mathcal{B}$;
11:            Compute targets for the $Q_{\phi_1}, Q_{\phi_2}$ using Eq. (8);
12:            Update $Q_{\phi_1}, Q_{\phi_2}$ using Eq. (9);
13:            Update policy using Eq. (10);
14:            Update target Q-networks using Eq. (11);
15:        end for
16:    end if
17: end for
```

This part is in the top center of Fig. 4. SkyML will request the MLaaSes selected by the action $\mathbf{a}_t$ next.

C. Taxonomy Unification: A Manual-Free Apporach

The core task of a cloud broker is to satisfy users' requirements. After receiving predictions from selected MLaaS providers, SkyML needs to standardize presentations of the returned predictions to prevent ambiguity. This ambiguity mostly comes from different taxonomies of different MLaaSes. Specifically, in the object detection service, this problem is caused by MLaaS-specific categories. For an input image, this service returns a list of detections $D = [d_1, d_2, ..., d_{l_d}]$, where d_i is given by a triple $[l_i, f_i, b_i]$ that consists of the corresponding category l_i, the corresponding confidence score f_i, and the bounding box b_i. l_d represents the number of objects detected in this image. However, services usually return different category names for the same object, i.e., "motorbike" vs. "motorcycle". It is necessary to merge these service-specific labels into a user-defined format.

However, unifying taxonomies from different MLaaSes by hand is time-consuming and error-prone [39], [40]. UniDet [41] presented a formulation to construct a unified taxonomy across different datasets automatically. However, they ignored label hierarchies and tree-liked mappings. For example, suppose the user wants the broker to return two "dog" labels but selected MLaaSes return "chihuahua" and "redbone". In that case, we should transform these two labels into "dog" even though the returned categories are more elaborate.

Given categories of multiple MLaaS providers, the task of this part is to merge these categories into a user-defined taxonomy. However, there are multitudinous possible mapping relations between MLaaS-specific and user-defined categories. Assume there are $|\mathcal{M}|$ MLaaSes and MLaaS i has n_i labels. The size of user-defined label space L_u is n_u. Then the number of possible mappings $\mathcal{R}$ is the same as putting $\Sigma_{i=0}^{|\mathcal{M}|} n_i$ different balls into n_u different boxes:

$$|\mathcal{R}| = n_u! \times \binom{\sum_{i=0}^{|\mathcal{M}|} n_i}{n_u - 1}. \quad (12)$$

For three MLaaS-specific label spaces and the user-defined one (COCO) that is presented in Tab. I, the number of overall possible mappings is about 4.1×10^{199}. Hence, we must define a rule to eliminate impossible mappings and acquire user-satisfied mappings.

As shown in the central bottom part of Fig. 4, We achieve this pruning requirement with the similarity derived by applying accuracy metrics. For MLaaS i, we represent the mapping between its label space L_i to user-defined label space L_u by $\mathcal{T}_i \in \{0,1\}^{|L_i| \times |L_u|}$. To obtain $\mathcal{T}_i$, we first collect predictions $\mathcal{P}$ from all available MLaaSes for user-given datasets. We group these predictions by category, i.e., $\mathcal{P}_c^i$ contains all the predictions in category c that from MLaaS i. Second, we compute similarities between user-defined labels and MLaaS-specific labels. Given predictions $\mathcal{P}_{c_m}^i$ of an MLaaS-specific label c_m and ground truth $\mathcal{G}_{c_u}$ of user-defined label c_u, we calculate the similarity $\mathcal{S}_{c_m,c_u}$ with:

$$\mathcal{S}_{c_m,c_u} = \mathcal{A}(\mathcal{G}_{c_u}, \mathcal{P}_{c_m}^i, \tau), \quad (13)$$

where $\mathcal{A}(\cdot)$ is the function to measure the accuracy performance of the corresponding task or service, and τ is the IoU threshold, which is used to check whether two boxes are matched. Note that we swap roles of the prediction and the ground truth. For example, the form to calculate the accuracy should be $\mathcal{A}(\mathcal{P}, \mathcal{G})$, but here we exchange places of two input parameters to calculate the similarity, i.e., $\mathcal{A}(\mathcal{G}, \mathcal{P})$. As the number of MLaaS predictions is always less than the size of ground truths, it could naturally magnify recall values.

The computational complexity to caclulate the similarity matrix $\mathcal{S}$ is $O(\xi \cdot |\mathcal{U}| \cdot n_u \cdot \sum_{i=0}^{|\mathcal{M}|} n_i)$, where ξ is the computational complexity of the similarity function and $|\mathcal{U}|$ is the size of the user-given dataset. Finally, for each MLaaS-specific label c_m, we sort and record the user-defined category c_m^u that is most similar to it. Then the $\mathcal{T}_{c_m,c_m^u}$ is assigned with 1, and the rest part of the vector $\mathcal{T}_{c_m}$ is turned into 0.

This method is universal and can be easily transferred to other tasks by substituting the similarity function with a task-dominated accuracy measurement function or a task-specific loss function. For example, we can take mAP as the similarity function for the object detection service and KL divergence as the similarity function for the classification service.

D. Ensemble Part

Fig. 2 illustrates that using numerous MLaaS can lead to enhanced precision. However, suppose we naively aggregate the predictions of selected MLaaSes. In that case, the final predictions usually perform poorly compared to those of a single MLaaS due to the false positive and repetitive predictions. Therefore, we propose an efficient strategy to ensemble the predictions, which consists of two steps: voting and ablation.

Voting Step. Voting methods are important in this context, and common approaches include *Affirmative*, *Consensus*, and *Unanimous* [42]. We group the unified detections into $G = [g_1, g_2, ..., g_r]$, where g_i, $i \in \{1, 2, ..., r\}$ is a subgroup of detections, and r represents the number of objects recognized by selected MLaaSes. The detections d_p and d_q in g_i must confirm that $IoU(b_p, b_q) > 0.5$ and $n_p = n_q$, in which $IoU(a, b)$ refers intersection over union. Let N be the number of selected MLaaSes. After this, we adopt the three voting methods: 1) *Affirmative*. This method considers all groups in G to be valid. Therefore, a prediction is deemed valid if any MLaaSes claims that a region contains an object. 2) *Consensus*. This method retains groups where at least half of the MLaaSes ($N/2$) agree that a region contains an object or more. In other words, a region is considered to contain an object when most clouds agree. 3) *Unanimous*. This method considers only the groups with a size equal to N, indicating that all MLaaSes must agree that the region contains an object.

We choose *affirmative* as the primary voting method. Because the MLaaSes are more of a complementary relationship, their predictions do not overlap much, as analyzed in Sec. II. We select the *affirmative* voting method as our primary approach. This decision is based on the complementary relationship between the three MLaaSes and the minimal overlap in their predictions, as analyzed in Sec. II. By using *consensus* or *unanimous* strategies, some true-positive results may be excluded. Additionally, the evaluation results from [42] demonstrate the superiority of the affirmative method over other approaches.

Ablation Step. Bounding boxes are often duplicated within a group, which results in an increased number of false-positive predictions and a decrease in mAP. To mitigate false-positive predictions, researchers have introduced three techniques: Non-Maximum Suppression (NMS) [43], Soft-NMS [44], and Weighted Boxes Fusion (WBF) [45]. NMS selects and keeps only the bounding box with the highest confidence score while discarding all other bounding boxes. However, NMS inevitably eliminates overlapping object detections, which leads to the introduction of Soft-NMS. Soft-NMS reduces the confidence score of overlapping detections instead of completely discarding them. Nevertheless, both NMS and Soft-NMS discard redundant boxes, preventing them from generating accurate localization predictions for objects from different models. WBF calculates the weighted average of the bounding box coordinates within a group to determine the retained box, utilizing the confidence score of the boxes as the weight. Additionally, the confidence score of the retained box is calculated as the average of the confidence scores from the boxes within the group.

Since the WBF method demonstrates superior performance compared to NMS and Soft-NMS [45], we choose WBF as the primary method for the second ensemble step. As depicted in Fig. 4, during the ensemble phase, we initially apply the voting method to each cluster of similar boxes. Subsequently, we employ the ablation method to eliminate duplicate boxes.

V. Performance Evaluation

This section presents a thorough evaluation of SkyML's performance through comprehensive experiments. Initially, we evaluate the performance of the provider selection approach. Subsequently, we analyze the impact of our approach, based on similarity, for integrating MLaaS-specific labels into the user-defined label space. Specifically, through real trace-driven evaluations, we provide clear evidence of SkyML's superiority over alternative benchmark approaches.

A. Setup and Methodology

Evaluation Setup. We collect predictions of COCO 2017 from AWS, AZU, and GCP to build the profile dataset to train the RL agent. We open-sourced the code and data[2]. AWS and GCP incur an inference cost of 0.001 USD per request, whereas AZU's cost varies by approximately 10% across regions. Since our measurements were conducted in Singapore with AZU's cost being 0.001 USD per image in this region, the inference cost for all MLaaSes mentioned above is set at 0.001 USD per image. We implemented the RL algorithm using the SpinningUp framework [46], with added support for GPU training on a server equipped with an NVIDIA 1080 Ti GPU, an Intel(R) Xeon(R) CPU E5-2650 v4@2.20GHz, and 64 GB of memory. The RL environment is implemented in Python for compatibility purposes. The learning rate η for both the actor-network and the Q-networks is set to 0.0001. Additionally, we assign values of 0.9 and 0.2 to γ and α respectively. We set β to 0 in order to maximize the mAP. The batch size is 1000, the training epoch is 100, and there are 2000 steps per epoch. For taxonomy unification, we employ AP_{50} as the similarity function and 80 labels of the COCO dataset as the user-defined label space.

Baseline Methods. We compare our approach with the following baselines. 1) **RAND-1**: This method randomly selects an MLaaS for each input. 2) **RAND-N**: This method randomly selects an MLaaS subset for each input. 3) **ENS**: This method selects all MLaaSes for each input. 4) **SkyML-OF**: This is the original SkyML, which uses ground truths of COCO to generate the reward. 5) **SkyML-OL**: This method relies on ensemble predictions from all MLaaSes as the ground truth for generating the reward. The hyperparameter β is set to -0.1 to select the action with the lower inference fee cost. The remaining hyperparameters remain unchanged from SkyML. 6) **SkyML-P**: In this approach, we train the RL agent using proximal policy optimization (PPO) [47], a classical on-policy RL algorithm that is worth comparing. 7) **SkyML-T**: This method involves training the RL agent using the twin delayed deep deterministic policy gradient (TD3) algorithm [38]. TD3 is a classical algorithm for training deterministic policies, and comparing it to our approach can highlight the advantages of the maximum entropy property of the SAC algorithm. 8) **UPB**: To increase the mAP while minimizing the cost, we utilize the measurement data to identify and record the MLaaS combination with the highest mAP and the minimal fee for each input. The voting method and ablation strategy are *affirmative* and *WBF*. The overall result from this baseline serves as the upper bound for SkyML-OF.

Evaluation Metrics. We employ the following metrics to evaluate the inference quality: 1) cost c_e: the average inference fee in a test episode, measured in units of 10^{-3} USD. 2) AP_{50}: the average precision of predictions achieved at a

[2]`https://github.com/ShuzhaoXie/Armol`

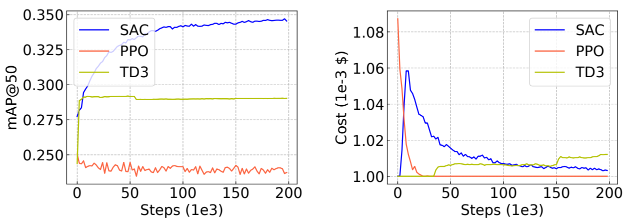


Fig. 5: Training process of SAC, PPO, and TD3. left Y: AP_{50} of a test episode. right Y: average cost of a test episode.

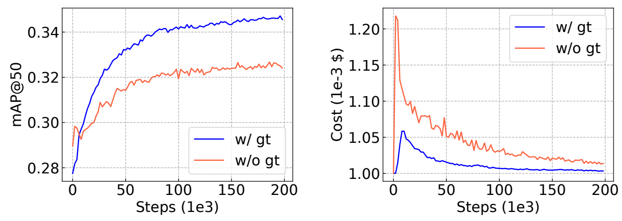


Fig. 6: Training process of SkyML-ON and SkyML-OF. "w/ gt": SkyML-OF; "w/o gt": SkyML-ON.

50% IoU threshold. We opt for AP_{50} over mAP to reduce computational demands and expedite training processes. Unlike mAP, which computes the average precision across IoU thresholds ranging from 50% to 95% in increments of 5%, AP_{50} specifically measures average precision at a 50% IoU threshold, representing only 10% of the computation involved in calculating mAP. Additionally, AP_{50} is a widely accepted standard metric in object detection tasks. We use the percent of reduced annotation cost (κ) to evaluate the quality of taxonomies unification, which is given by:

$$\kappa = 1 - \frac{\sum_{i=0}^{|\mathcal{M}|} n_i + n_u \cdot (n_{UM} + n_{MM})}{n_u \cdot \sum_{i=0}^{|\mathcal{M}|} n_i}. \tag{14}$$

n_{UM} and n_{MM} are the number of unmatched and mismatched MLaaS-specific labels, respectively.

B. Performance of Provider Selection Method

To understand the performance, we have conducted experiments to illustrate the effectiveness of our method, the feasibility of online training, the scalability of our provider selection approach, and the detection quality of SkyML.

Effectiveness of SkyML. As shown in Tab. III, SkyML-OF achieves an equal mAP and a 67% reduction in inference fee compared to the ENS baseline. Moreover, SkyML-OF outperforms RAND-N, resulting in a 3.09% higher mAP and a 41.75% reduction in inference fee. As for the training algorithm, we observe that the mAP of SkyML on SAC (SkyML-OF) surpasses that of other baselines while exhibiting a lower average cost than TD3. In Fig. 5, SAC demonstrates superior and quicker convergence than other algorithms.

Feasibility of Online Training. During online training, we recommend utilizing the ensemble prediction of all MlaaSes as the ground truth. Tab. III shows that the method without ground truth reduces cost by 66% compared to all federated predictions, with only 4.3% lower mAP. This result indicates

TABLE III: Performance of different baselines. "AWS" denotes the quantity of images selected from AWS in an episode, and "AZU" and "GCP" carry analogous interpretations.

Methods	mAP	AP_{50}	Cost	AWS	AZU	GCP
RAND-1	15.75	24.49	1.000	1690	1605	1657
RAND-N	18.66	28.89	1.722	2858	2863	2809
SkyML-P	14.99	25.05	1.087	1300	2541	1543
SkyML-T	18.90	29.20	1.006	4843	114	26
ENS	21.75	34.69	3.000	4952	4952	4952
UPB	23.83	37.70	1.202	3881	1126	944
SkyML-ON	20.81	32.68	1.016	3426	683	924
SkyML-OF	21.75	34.80	1.003	2863	950	1156

TABLE IV: Performance of different simulated MLaaS.

MLaaS	AP_{50}	Cost	MLaaS	AP_{50}	Cost
0	28.88	1.000	5	53.43	1.000
1	24.38	1.000	6	20.76	1.000
2	24.38	1.000	7	51.33	1.000
3	34.69	1.000	8	25.13	1.000
4	50.19	1.000	9	34.81	1.000
All	49.29	10.000	SkyML	53.44	1.002

that taking ENS as the ground truth can achieve the goal of minimizing inference fees while approaching sub-optimal accuracy. Furthermore, it suggests that ENS and ground truth share similarities. The performance of the provider selection algorithm heavily relies on reward supervision, and choosing ENS as the supervision when there is no ground truth is reasonable. Fig. 6 demonstrates that SkyML-ON converges stably, although its AP_{50} and cost are not as good as SkyML-OF during the training process.

Scalability on the Number of MLaaSes. To evaluate the scalability of our approach with a larger number of MLaaSes, we incorporate predictions from Aliyun Object Detection [48] and simulate six additional MLaaSes. The details of simulated MLaaSes can be found in our GitHub repository. We assign index values MLaaS 0-9 to AWS, AZU, GCP, Aliyun, and the simulated MLaaSes. In Tab. IV, we observe that the ensemble predictions of the 10 MLaaSes are lower than that of MLaaS 5. We suggest this is because MLaaS 5 has an AP_{50} value that is 20%-30% higher than the other MLaaSes, which indicates that MLaaS 5 cannot generate more true positive results and only adds to the number of false positive results in the ensemble predictions. However, in the same table, we can see that SkyML's AP_{50} is slightly better than that of MLaaS 5, with almost the same cost, which suggests that our algorithm can select better combinations of MLaaSes despite the significant variation in the AP_{50} values among the MLaaSes. In Fig. 7, we demonstrate that our approach consistently converges when using ten MLaaSes, with 1023 possible actions, achieving a balance between AP_{50} and cost.

Scalability on Other Scenarios. To evaluate the scalability of different scenarios, we apply SkyML to the spoken command recognition services. Compared to the object detection service, the spoken command recognition service is a classification task, with the added complexity of the pricing varying among different cloud services. Specifically, we evaluate SkyML on the AudioMNIST dataset [49], which consists of 30000 audio samples of spoken digits (0-9) of 60

TABLE V: Results of the spoken command recognition task.

Metrics	Google	IBM	MS	RND-1	RND-N	ENS	UPB	SkyML
Acc. (%)	88.48	98.28	98.62	94.53	91.13	98.81	99.73	98.83
Cost (10^{-3})	6.00	2.50	4.10	4.20	6.33	12.59	2.53	2.86

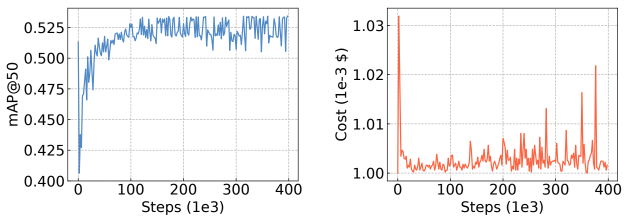


Fig. 7: With 10 available MLaaS providers (1023 actions), the $mAP@50$ and the cost still converge stably.

different speakers. To obtain the corresponding cloud service predictions of AudioMNIST, we use the measurement results from [16], including the recognition results from Google [27], IBM [50], and Microsoft (MS) [26]. The price and accuracy of these providers are listed in Tab. V, where we use the data collected on March 29, 2020.

To apply SkyML to spoken command recognition services, we have made the following modifications for each part. In the provider selection part, we use the STT-En-Fast-Conformer-CTC-Large checkpoint from NeMo [51] to extract the feature vector of each audio and employ the function specified in Eq. 7 to evaluate the reward. In the taxonomy unification part, since these cloud services only return a single word to represent 0 and 9, we only need to specially annotate the recognition results that do not belong to the range of 0-9 to facilitate reward calculation. As for the ensemble part, we choose the category with the maximum votes; in the event of a tie, we choose the category with the highest accumulated confidence. We list the accuracy and the average cost of multiple baselines in Tab. V. The evaluation outcomes evince that SkyML attains a comparable level of accuracy to ENS while concurrently effecting a cost reduction of 77%.

User Study for Detection Quality. To further understand the detection quality of SkyML, we perform a user study involving 25 AI experts from multiple areas, including multimedia, computer vision, and reinforcement learning. We ask these AI experts to compare the detection quality of our methods against the upper bound of multi-MLaaSes. Specifically, we randomly choose 500 detection results in UPB and SkyML baselines. Then we visualized them onto the images and combined images as multiple <`SkyML`, `UPB`> pairs. We also randomly shuffled the order of pairs between SkyML and UPB to avoid influencing user choices. After that, we separate these pairs into 10 forms, each containing 50 choice questions. A question contains an image pair and has three options: "Better", "Similar", and "Worse". Users were asked to vote "Better" if they felt the first detection results were better than the second one, and so on. As every expert fills 2 forms, we finally get 2500 votes, and one question gets 5 votes.

To get the final results, we have to ensemble the 5 votes. We consider the option with the highest count as the final result. Suppose the distribution of these 5 votes presents a split of 2:2:1, meaning two options are tied with two votes each. In that case, we utilize the following rules to determine the final vote: 1) If there are two "Better" votes and two "Similar" votes, then the final result is "Better"; conversely, if there are two "Worse" votes and two "Similar" votes, then the final result is "Worse". 2) If each of the two images receives two "Better" votes, the final option would be "Similar". Final results reveal that 83.2% of the results of our method are recognized as better or similar to the upper bound baseline. Moreover, we find that 44 of 500 detection results are recognized to be better than the upper bound baseline. Consequently, relying solely on the accuracy metric to select the upper bound may not yield the best results for users.

TABLE VI: Labels that hard be assigned to user label space.

Provider	Unmatched Labels
AWS	tablet computer, desk, bulldozer
AZU	bathroom, alpaca/llama, amphibious vehicle, orchard apple tree, microphone, antelope, cart
GCP	traffic sign, flower

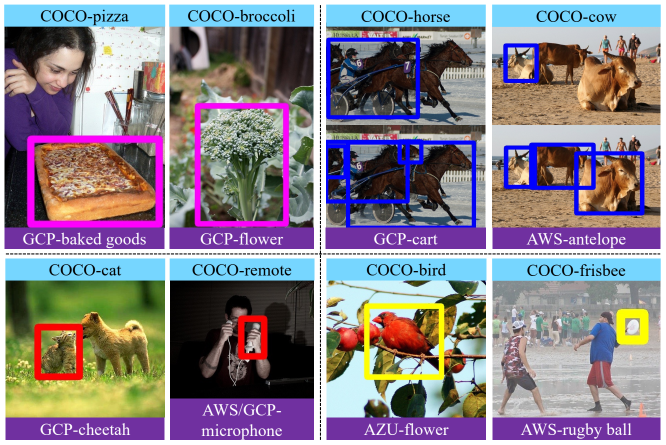


Fig. 8: Examples of mismatched MLaaS-specific labels.

C. Performance of Taxonomy Unification

Among 358 MLaaS-specific labels mentioned in Tab. I, 12 of which cannot be classified into user-defined label space. By carefully skimming auto-generated mappings between user labels and MLaaS labels. we observe that 16 labels are misassigned. Based on the Eq. 14, our approach reduces this cost by 90.9%. The 12 unmatched labels are listed in Tab. VI. This result is due to predictions for these categories having no intersection with ground truths. From our perspective, manually merging these 12 categories into user label space would be better.

Dive into Misassigned Labels. To provide more insights, we divide these mismatched label pairs into 4 categories: 1) **Inclusion.** This correspondence is a subordinate relationship, e.g., A belongs to B. As shown in the left top of Fig. 8, the GCP-baked goods are assigned to the COCO-pizza, and the GCP-flower is correspondence to COCO-broccoli. Both labels are justified, and there is no one right or wrong. We

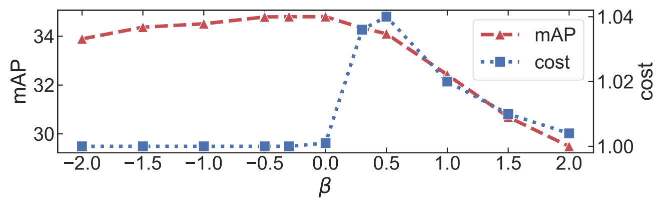


Fig. 9: Ablation analysis of β. We collect the best mAP of the test episode and its corresponding cost for each β value.

can observe that GCP and AZU tend to provide a more universal label. 2) **Attention.** The reason for this error is that the annotators pay different attention. In the right top of Fig. 8, the annotators of the GCP model focus on labeling the car in a cart, while COCO focuses on labeling the horse in a cart. In addition, AWS tries to label the back of the animal with horns as "Antelope", while COCO will label it as "Cow". 3) **Error-prone.** As shown in the left bottom of Fig. 8, a professional labeler cannot tell which category the boxed object belongs to. 4) **Wrong.** The right bottom of Fig. 8 persents this type.

The AP that represents the detection accuracy of one category is the area under the precision-recall curve. We could get larger recall values if we exchange roles of MLaaS-returned predictions and user-defined ground truth, leading to a larger AP value. This trick might be helpful if the developer or user wants to import a threshold to enable further restrictions on mappings between user-defined and MLaaS-specific categories.

D. Ablation Study

Choice of Different Baselines. On the right side of Tab. III, we collect the choice of test images under different provider selection approaches. We find that the Choice of ARM-T is so unbalanced that AWS got most of the selections, which indicates that the TD3 algorithm is too easy to converge and thus cannot help the model learn the advantages of AZU and GCP. Moreover, compared to UPB, we observe that the sum of Choice of SkyML-OF and SkyML-ON is unexpectedly less, demonstrating that we should decrease the cost weight in the reward. Therefore, to understand the effect of cost weight (β) for the whole model, we evaluate its sensitivity in the following subsection.

Sensitivity of β. As shown in Fig. 9, choosing the value between -1.0 and 0.5 for beta is better. Moreover, we reveal the constraint ability of β. On the one hand, when β goes down, the mAP does not reduce too much, but the cost approaches the lower bound of 1.0. As the algorithm tends to select the most accurate one when the budget is low, the above phenomenon shows that the prediction of the right MLaaS is almost the same as the most precise result that multiple MLaaSes can achieve. On the other hand, when β goes up, the mAP shrinks suddenly even though more budgets are provided. The reason is that the weight of the cost in the reward function overrides the accuracy, resulting in higher rewards for some input models by simply picking more cloud services.

Sensitivity of τ. In Fig. 10, we count the number of unmatched and mismatched labels under different τ values. As

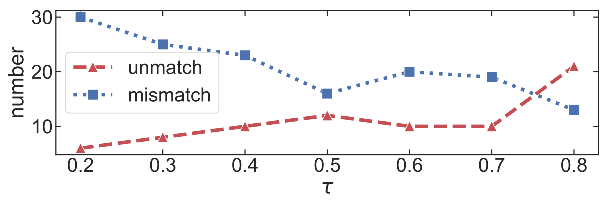


Fig. 10: Ablation analysis of τ. We count the unmatched and mismatched labels under different τ values.

TABLE VII: κ value under different τ values.

τ	0.2	0.3	0.4	0.5	0.6	0.7	0.8
κ (%)	88.69	89.53	89.53	90.92	90.37	90.64	89.25

the value of τ increases, the number of unmatched labels increases while the number of mismatched labels decreases. This is intuitive because τ reflects the strictness of the bounding box matching. The larger the τ value, the stricter the box matching, and the lower the likelihood of finding a user label to match the unmatched MLaaS-specific label. Since the number of unmatched and mismatched labels affects the annotation cost, we further calculate the corresponding κ values. In Tab. VII, the κ are generally stable, which indicates that the predictions of multiple MLaaSes are accurate, and the overlapping ratio (IoU) of the matched bounding boxes is high, ensuring a stable result across different τ values. When τ is between 0.5-0.7, the κ value reaches a small peak. Hence, by adjusting τ, we can balance the error rate and unmatched rate, i.e., minimizing the sum of these two rates.

VI. Related Work

A. Measurements Study for MLaaS

Previous research has focused on measuring the inference accuracy and latency of ML models [12], [13]. However, these studies primarily examined user-known models rather than confidential ML services. Furthermore, Yao et al. [14] aimed to measure ML training platforms instead of inference services. Liu et al. [15] conducted a measurement study on older ML services, which were limited to decision trees, SVMs, and multi-layer fully connected neural networks. These findings differ significantly from the next-generation ML services currently being promoted, which prioritize models that are transparent to users and are focused on deep learning. Although HAPI [16] established a response dataset of ML inference services, it does not measure the latency of ML inference services or the object detection services. Our measurement study has addressed the aforementioned gaps.

B. Machine Learning Inference Services

Prior work on edge-cloud collaborated inference services [52] spans on robustness [53] and pricing mechanisms [54]. Recently, FrugalML [10] studies ML API calling strategies for single-label classification. Further, FrugalMCT [11] adaptively selects the APIs to use for different data in an online fashion while respecting the user's budget. The core weakness of FrugalMCT is that it requires the prediction of a base service to decide whether to request a new service, but the base service may not belong to the best MLaaS combination. However, SkyML does not need to request any underlying services in advance and thus can guarantee the choice of the best MLaaS combination. Besides, SkyML could handle more complicated outputs and employ RL to solve the provider selection problem. One API designed for multi-label image classification might generate the following output: *(person, 0.8), (car, 0.7)*. This indicates that the image contains a person with a confidence score of 0.8 and a car with a confidence score of 0.7. The object detection API returns more complicated predictions that contain boxes to describe the locations of objects, i.e., {*(person, 0.8, [34, 101, 10, 60]), (car, 0.7, [78, 195, 74, 72])*}. The complex prediction of the object detection service makes FrugalMCT inappropriate for it. Further, FrugalMCT has not considered different vocabulary from different providers, but our framework has noticed and solved this problem. Finally, FrugalMCT selects at most two services for each input as its algorithm aims to select only one extra service to enhance the accuracy of the result of the base service.

C. Cloud Federation

Previous studies on cloud federation involve combining cloud services from various providers into a unified pool, allowing for the migration of features-resources, resource redundancy, and complementary resources [55]. These efforts can potentially decrease costs by partially outsourcing to regions with higher cost-efficiency [56]. Nonetheless, prior cloud federation studies concentrated on the integration of explicit resources like storage and compute resources, to optimize expense and stability. In contrast, our approach involves integrating implicit resources, specifically the training data and models underlying commercial ML inference services, to attain optimal analytic performance. Recently, sky computing [6], which is an alternative term for cloud federation, has also gained significant traction. Skyplane [8] enhances inter-region data transfer speeds by utilizing indirect paths at the application layer for routing data. SkyPilot [7] enables the easy and cost-effective execution of ML workloads on any cloud platform. Our work complements SkyPilot since it takes into account the joint utilization of multiple ML inference services, which is not addressed by SkyPilot.

D. User Experience in Multicloud Inference

User experience is a topic that never fades in multimedia [57]. Here, we focus on the user experience of multi-model inference. The primary objective of training on multiple datasets is to achieve dataset unification, which entails merging different semantic concepts. This task is analogous to the unification of MLaaS-specific labels. MSeg [39] manually harmonizes the taxonomies of 7 semantic segmentation datasets and employs Amazon Mechanical Turk to resolve inconsistent annotations between datasets. Wang et al. [58] train a partitioned model on multiple datasets, leveraging diverse sources of supervision to attain robustness. Universal-RCNN [59] models the class relations with an inter-dataset attention module, but it may produce duplicated outputs for objects present in more than one dataset. UniDet [41] enhances the aforementioned works through the unification of visual concepts within a single label space, yielding a singular, consistent model that does not necessitate knowledge of the test domain. In contrast, Zhao et al. [40] manually merge taxonomies and subsequently train with cross-dataset pseudo-labels generated by dataset-specific models, which serves as a complementary approach to UniDet. ScaleDet [60] leverage knowledge from pre-trained models, such as CLIP, to train a visual-language object detector. These works aim to merge dataset-specific taxonomies into a bigger and more accurate label space, while we want to merge MLaaS-specific taxonomies into user-satisfied label space.

E. Combinatorial Reinforcement Learning

Discrete, high-dimensional action spaces are commonly found in various applications [61], such as natural language processing [62], text-based applications [63], and vehicle routing [36]. However, they present a challenge for standard RL algorithms [64] due to the impracticability of enumerating the action space for selecting the next action from a given state. Recent approaches to address this challenge involve choosing the optimal action from a random sample [62], approximating the discrete action space with a continuous one [34], [35], training an additional ML model to eliminate sub-optimal actions [63], and formulating the action selection problem as a mixed-integer program for each state [36]. Zhong et al. [65] apply the Wolpertinger policy to the edge cache problem, but it merely serves as an application and does not contribute to the policy itself. In our provider selection approach, we embed the continuous action into the binary action space by associating it with the nearest neighbor. To enhance exploration, we utilize the SAC algorithm for training.

VII. Discussion

This section examines the security and scalability constraints within our research and the broader significance of our results, offering insights into areas for future investigation and practical applications.

Security and Privacy. Data security and privacy are crucial considerations in cloud services. There is potential for the misuse of user data. Viable solutions encompass anonymization techniques that do not compromise the final identification results, such as locally identifying and masking sensitive information. Alternatively, model service providers can partition their models, deploying a portion within user-end applications and retaining another portion in the cloud. During each inference process, intermediate results are computed using the user-end model and transmitted to the cloud for prediction.

Potential Limitations. Our approach has potential limitations: 1) Model bias: In the provider selection process, we use a pre-trained lightweight model to extract input features. The biases embedded in this lightweight model can influence the whole selection pipeline. 2) Task-oriented: Whenever transitioning to a new task, we must research the methods corresponding to each module again. For instance, in the ensemble part, different tasks yield different outputs, necessitating research into suitable ensemble algorithms. 3) Metrics: We utilize the established mAP metric to quantify the accuracy of object detection services. However, some MLaaSes, such as LLM services [9], lack a mature quantification metric. Precisely quantifying whether a service aligns with a user's preferences is crucial for achieving MLaaS federation.

Scalability across Various Scenarios. The scalability of our approach may fail in the following scenarios: 1) Cloud services capable of handling various tasks. Although we have conducted experiments on the spoken command classification task, applying SkyML to large language model (LLM) services still presents challenges. LLM services excel at diverse tasks, making it challenging to devise an unbiased metric for evaluating their performance and pricing. Further assessment of whether LLM services adequately meet user demands necessitates additional resources. Moreover, aggregating generated texts from multiple LLMs into a precise answer poses an extra challenge. 2) Fluctuating numbers of cloud services. While we have validated the provider selection method's performance across ten cloud services, our approach cannot accommodate dynamic changes in the number of cloud services, as the model's capacity to handle cloud service numbers is fixed. If additional cloud services are introduced, this would entail adding an additional output dimension to the action function in the reinforcement learning model. Our approach cannot dynamically adjust the reinforcement learning model based on changes in the number of cloud services.

Broader Implications. Recently, many search engines have incorporated LLM into their search processes. For instance, Bing [66] has integrated Copilot into its search results, while Perplexity [67] employs LLM to consolidate search outcomes. Similar to vision models from cloud providers with various capabilities, we envision that other emerging models, like LLM in MLaaSes, could also demonstrate such diversity. Our SkyML design can further exploit this to help users derive a more credible answer through federation.

VIII. Conclusion

In this paper, we propose SkyML, a novel broker for utilizing multiple MLaaSes. We begin by conducting a measurement study to identify two key challenges for SkyML: 1) How can the right MLaaS combination be selected to maximize accuracy while minimizing cost? 2) How can the various MLaaS-specific taxonomies be automatically unified and the results are efficiently ensembled? To tackle the first problem, we propose a combinatorial RL approach and represent the huge discrete space with continuous action. For the second problem, we propose a taxonomy unification method based on similarity learned from the data. Our approach yields significant cost savings, reducing inference fees by 67% and annotation costs by 90%, all without compromising analytic accuracy.

References

[1] S. Liu, G. Tian, Y. Zhang, and P. Duan, "Scene recognition mechanism for service robot adapting various families: A cnn-based approach using multi-type cameras," *TMM*, 2021.

[2] ZOZO. [Online]. Available: https://corp.zozo.com/en/

[3] Amazon rekognition. [Online]. Available: https://aws.amazon.com/rekognition/

[4] Aws rekognition customers. [Online]. Available: https://aws.amazon.com/rekognition/customers

[5] J. Weinman, "Intercloudonomics: Quantifying the value of the intercloud," *Cloud Computing*, 2015.

[6] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *HotOS*, 2021.

[7] Z. Yang, Z. Wu, M. Luo *et al.*, "An intercloud broker for sky computing," in *USENIX NSDI*, 2023.

[8] P. Jain, S. Kumar, S. Wooders *et al.*, "Skyplane: Optimizing transfer cost and throughput using cloud-aware overlays," *arXiv:2210.07259*, 2022.

[9] OpenAI Chat GPT. [Online]. Available: https://chat.openai.com/chat

[10] L. Chen, M. Zaharia, and J. Y. Zou, "Frugalml: How to use ml prediction apis more accurately and cheaply," *NeurIPS*, 2020.

[11] L. Chen, M. Zaharia, and J. Zou, "Efficient online ML API selection for multi-label classification tasks," in *ICML*, 2022.

[12] V. J. Reddi, C. Cheng, D. Kanter *et al.*, "Mlperf inference benchmark," in *ISCA*, 2020.

[13] H. Zhang, Y. Huang, Y. Wen *et al.*, "Inferbench: Understanding deep learning inference serving with an automatic benchmarking system," *arXiv:2011.02327*, 2020.

[14] Y. Yao, Z. Xiao, B. Wang *et al.*, "Complexity vs. performance: empirical analysis of machine learning as a service," in *IMC*, 2017.

[15] Y. Liu, H. Zhang, L. Zeng *et al.*, "Mlbench: benchmarking machine learning services against human experts," *VLDB*, 2018.

[16] L. Chen, Z. Jin, S. Eyuboglu *et al.*, "Hapi: A large-scale longitudinal dataset of commercial ml api predictions," in *AAAI*, 2023.

[17] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[18] Q. Zhou, J. Cao, H. Leng, Y. Yin, Y. Kun, and R. Zimmermann, "Sogdet: Semantic-occupancy guided multi-view 3d object detection," *AAAI*, 2024.

[19] D. Liu, C. Zhang, Y. Song, H. Huang, C. Wang, M. Barnett, and W. Cai, "Decompose to adapt: Cross-domain object detection via feature disentanglement," *TMM*, 2023.

[20] Q. Ren, S. Lu, J. Zhang, and R. Hu, "Salient object detection by fusing local and global contexts," *TMM*, 2021.

[21] t. murad, A. Nguyen, and Z. Yan, "Dao: Dynamic adaptive offloading for video analytics," in *MM*. ACM, 2022.

[22] Z. Yang, W. Ji, Q. Guo, and Z. Wang, "Javp: Joint-aware video processing with edge-cloud collaboration for dnn inference," in *MM*. ACM, 2023.

[23] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-oriented autonomous driving," in *CVPR*, 2023.

[24] K. Takumi, K. Watanabe, Q. Ha, A. Tejero-De-Pablos, Y. Ushiku, and T. Harada, "Multispectral object detection for autonomous vehicles," in *MM Thematic Workshops*. ACM, 2017.

[25] S. Xie, Y. Xue, Y. Zhu, and Z. Wang, "Cost effective mlaas federation: A combinatorial reinforcement learning approach," in *INFOCOM*, 2022.

[26] Azure computer vision. [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision

[27] Google vision ai. [Online]. Available: https://cloud.google.com/vision

[28] "COCO - Common Objects in Context," 2021. [Online]. Available: https://cocodataset.org/#detection-eval

[29] T.-Y. Lin, M. Maire, S. Belongie *et al.*, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[30] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," in *ICLR*, 2021.

[31] 5G Speed: How to Understand the Numbers. [Online]. Available: https://www.lifewire.com/5g-speed-4180992

[32] Quadratic knapsack problem. [Online]. Available: https://en.wikipedia.org/wiki/Quadratic_knapsack_problem

[33] M. J. Magazine and M.-S. Chern, "A note on approximation schemes for multidimensional knapsack problems," *Mathematics of Operations Research*, 1984.

[34] G. Dulac-Arnold, R. Evans, H. van Hasselt *et al.*, "Deep reinforcement learning in large discrete action spaces," *arXiv:1512.07679*, 2015.

[35] J. He, J. Chen, X. He *et al.*, "Deep reinforcement learning with a natural language action space," in *ACL*, 2016.

[36] A. Delarue, R. Anderson, and C. Tjandraatmadja, "Reinforcement learning with combinatorial actions: An application to vehicle routing," in *NeurIPS*, 2020.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *ICML*, 2018.
[38] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *ICML*, 2018.
[39] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, “Mseg: A composite dataset for multi-domain semantic segmentation,” in *CVPR*, 2020.
[40] X. Zhao, S. Schulter, G. Sharma, *et al.*, “Object detection with a unified label space from multiple datasets,” in *ECCV*, 2020.
[41] X. Zhou, V. Koltun, and P. Krähenbühl, “Simple multi-dataset detection,” in *CVPR*, 2022.
[42] A. Casado-Garcia and J. Heras, “Ensemble methods for object detection,” in *ECAI*, 2020.
[43] J. Hosang, R. Benenson, and B. Schiele, “Learning non-maximum suppression,” in *CVPR*, 2017.
[44] N. Bodla, B. Singh, R. Chellappa *et al.*, “Soft-nms–improving object detection with one line of code,” in *ICCV*, 2017.
[45] R. Solovyev, W. Wang, and T. Gabruseva, “Weighted boxes fusion: ensembling boxes for object detection models,” *arXiv:1910.13302*, 2019.
[46] J. Achiam, “Spinning Up in Deep Reinforcement Learning,” 2018.
[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, 2017.
[48] Alibaba cloud object detection. [Online]. Available: https://vision.aliyun.com/objectdet
[49] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek, “Interpreting and explaining deep neural networks for classification of audio signals,” *arXiv*, 2018.
[50] Ibm cloud. [Online]. Available: https://www.ibm.com/cloud
[51] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook *et al.*, “Nemo: a toolkit for building ai applications using neural modules,” *arXiv*, 2019.
[52] P. Zhang, F. Huang, D. Wu, B. Yang, Z. Yang, and L. Tan, “Device-edge-cloud collaborative acceleration method towards occluded face recognition in high-traffic areas,” *IEEE Transactions on Multimedia*, 2023.
[53] H. Hosseini, B. Xiao, and R. Poovendran, “Google’s cloud vision API is not robust to noise,” in *ICMLA*, 2017.
[54] L. Chen, P. Koutris, and A. Kumar, “Towards model-based pricing for machine learning in a data marketplace,” in *SIGMOD*, 2019.
[55] T. Kurze, M. Klems, D. Bermbach *et al.*, “Cloud federation,” *Cloud Computing*, 2011.
[56] M. Giacobbe, A. Celesti, M. Fazio *et al.*, “Towards energy management in cloud federation: a survey in the perspective of future sustainable and cost-saving strategies,” *Computer Networks*, 2015.
[57] L. Wang, C. Li, W. Dai, S. Li, J. Zou, and H. Xiong, “Qoe-driven adaptive streaming for point clouds,” *IEEE Transactions on Multimedia*, 2022.
[58] X. Wang, Z. Cai, D. Gao *et al.*, “Towards universal object detection by domain attention,” in *CVPR*, 2019.
[59] H. Xu, L. Fang, X. Liang *et al.*, “Universal-rcnn: Universal object detector via transferable graph r-cnn,” in *AAAI*, 2020.
[60] Y. Chen, M. Wang, A. Mittal *et al.*, “Scaledet: A scalable multi-dataset object detector,” in *CVPR*, 2023.
[61] S. Li, P. Gao, X. Tan, and W. Xiang, “Rlgrid: Reinforcement learning controlled grid deformation for coarse-to-fine point could completion,” *TMM*, 2023.
[62] J. He, M. Ostendorf, X. He *et al.*, “Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads,” in *EMNLP*, 2016.
[63] T. Zahavy, M. Haroush, N. Merlis *et al.*, “Learn what not to learn: Action elimination with deep reinforcement learning,” in *NeurIPS*, 2018.
[64] G. Dulac-Arnold, D. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” *arXiv:1904.12901*, 2019.
[65] C. Zhong, M. C. Gursoy, and S. Velipasalar, “A deep reinforcement learning-based framework for content caching,” in *CISS*, 2018.
[66] Microsoft Bing. [Online]. Available: https://www.bing.com
[67] Perplexity. [Online]. Available: https://www.perplexity.ai

Shuzhao Xie is currently a PhD student at Shenzhen International Graduate School, Tsinghua University. He received his B.S. degree from Beijing Normal University, China, in 2020, and his M.Eng. degree from Tsinghua University, China, in 2023. His research focuses on multimedia and computer vision.

Yuan Xue is currently a research scientist at Zhongguancun Laboratory, China. He received the M.Eng. degree in computer technology with Tsinghua University. His research interests include edge computing, video analytics and domain generalization.

Yifei Zhu is currently an Assistant Professor at the University of Michigan-Shanghai Jiao Tong University Joint Institute in Shanghai Jiao Tong University, China. He received his B.E. degree from Xi’an Jiaotong University, China, in 2012, his M.Phil. degree from The Hong Kong University of Science and Technology, China, in 2015, and his Ph.D. degree in Computer Science from Simon Fraser University, Canada, in 2020. His current research interests include edge computing, multimedia networking, and distributed machine learning systems, where he has published in ACM SIGCOMM, IEEE INFOCOM, ACM Multimedia, and many other venues.

Zhi Wang is currently an associate professor at Shenzhen International Graduate School, Tsinghua University. He received his Ph.D. in 2014 and his B.E. in 2008, both from Department of Computer Science and Technology, Tsinghua University. His research areas include multimedia networks, mobile cloud computing, and large-scale machine learning systems. He is a recipient of the Natural Science Award of the Ministry of Education (First Prize) in 2017, the National Natural Science Award (Second Prize) in 2018, and the Technology Invention Award of the Chinese Institute of Electronics (First Prize) in 2020. In addition, his research won the Best Paper Award of ACM Multimedia, Best Paper Award of IEEE TMM, the Best Student Paper Award of MMM, and the Best Paper Award of ACM Multimedia, HUMA Workshop. He is an Associate Editor of IEEE TMM and Guest Editor of ACM TIST and JCST.