Cost Effective MLaaS Federation: A Combinatorial Reinforcement Learning Approach

Shuzhao Xie[†], Yuan Xue[†], Yifei Zhu[‡], Zhi Wang[†§*]
[†]Shenzhen International Graduate School, Tsinghua University
[‡]UM-SJTU Joint Institute, Shanghai Jiao Tong University
[§]Peng Cheng Laboratory
{jsz20, xuey21}@mails.tsinghua.edu.cn, yifei.zhu@sjtu.edu.cn, wangzhi@sz.tsinghua.edu.cn

***Abstract*—With the advancement of deep learning techniques, major cloud providers and niche machine learning service providers start to offer their cloud-based machine learning tools, also known as machine learning as a service (MLaaS), to the public. According to our measurement, for the same task, these MLaaSes from different providers have varying performance due to the proprietary datasets, models, etc. Federating different MLaaSes together allows us to improve the analytic performance further. However, naively aggregating results from different MLaaSes not only incurs significant momentary cost but also may lead to sub-optimal performance gain due to the introduction of possible false positive results. In this paper, we propose Armol, a framework to federate the right selection of MLaaS providers to achieve the best possible analytic performance. We first design a word grouping algorithm to unify the output labels across different providers. We then present a deep combinatorial reinforcement learning based-approach to maximize the accuracy while minimizing the cost. The predictions from the selected providers are then aggregated together using carefully chosen ensemble strategies. The real-world trace-driven evaluation further demonstrates that Armol is able to achieve the same accuracy results with 67% less inference cost.**

***Index Terms*—machine learning as a service, cloud federation, combinatorial reinforcement learning, object detection**

I. Introduction

Recent advancements in machine learning techniques and the maturation of cloud services have propelled the introduction of machine learning as a service, in which cloud providers offer machine learning training platforms or machine learning inference services via machine learning APIs to users. Major cloud providers, such as Amazon Web Service[1] (AWS), Microsoft Azure[2], Google Cloud Platform[3] (GCP), etc., and niche machine learning vendors, such as BigML[4], Algorithmia[5], etc., have all offered their own MLaaS. The MLaaS market was valued at 1.60 billion USD in 2020 and is expected to reach 12.10 billion USD by 2026 [1]. The well-defined interfaces and the free maintenance burden for the underlying cloud infrastructures allow more industrial verticals and applications to access the machine learning process from anywhere, at any time.

*Corresponding author.
[1]https://aws.amazon.com
[2]https://azure.microsoft.com
[3]https://cloud.google.com
[4]https//bigml.com
[5]https://algorithmia.com/

From the users' perspective, although the high abstraction of MLaaS brings ease of use, these abstracted services have also made the underlying latency, accuracy unknown to the users. To explore the underlying mechanisms of cloud services, previous works mainly focus on measuring the inference accuracy and latency of user-known models [2], [3]. The performance of cloud-based inference services has not been studied yet. According to our initial measurement by collecting the predictions from object detection services, we find that the overall mAP of the major cloud services significantly differs from each other, and each provider has different sweet-spot categories of tasks that achieve the best analytic performance than other categories. For example, in our measurement, though AWS outperforms Azure by 3.7% in the general object detection task, Azure outperforms AWS by 10.9% in a specific "bottle" category. Therefore, leveraging the service provider with the highest general accuracy loses the opportunities to fully exploit the analytic capability of all providers. On the contrary, aggregating all service providers may also introduce extra false positive results. To gain the most from these MLaaS providers, it is thus beneficial to combine the expertise from different service providers and select the right set of MLaaS providers.

However, realizing MLaaS federation is non-trivial. First, different MLaaS providers may have different vocabulary to describe the same task. With the fast update of each provider's services, we need an efficient algorithm to unify the description languages used in different providers. Second, it is computational challenging to select the right set of providers due to the combinatorial nature of this problem. The large possible provider list and the resulting exponential number of choices make the brute-force approach not scalable and practical in real-world scenarios. Third, after receiving the analytic results from different service providers, how to efficiently merge these results to offer optimal aggregated results also need further design.

Therefore, in this paper, we present Armol, the first work on MLaaS federation for optimal analytic performance. Our framework covers three parts: the provider selection part, the word grouping part, and the ensemble part. Specially, we propose a combinatorial reinforcement learning (RL)-based approach to solve the provider selection problem. We map continuous action spaces to discrete integer action spaces by

finding the nearest neighbor in large discrete combinatorial action spaces of a continuous action so that we can solve the computational challenge of select the right providers. Our word grouping part unifies the categories with the same meaning from different providers based on the synonym dataset extracted from WordNet [4]. In the ensemble part, we use an affirmative voting strategy and weighted box fusion for ablation so that the total analytic results can be further optimized. We conduct extensive real trace-driven experiments to evaluate the performance of our framework.

In summary, our contributions are:

- Our measurement studies on major cloud providers reveal the varying differences among existing MLaaS offerings and the great potential in MLaaS federation to improve analytic performance.
- We formulate the MLaaS federation problem as a combinatorial provider selection problem and propose a combinatorial reinforcement learning-based approach to maximize accuracy.
- Efficient ensemble and grouping strategies are proposed to unify the vocabulary of different providers and aggregate the eventual results.
- Real-world trace-driven simulations demonstrate that our framework can reduce 67% cost of inference fee without sacrificing accuracy compared to other benchmark approaches.

The remainder of this paper is organized as follows. Sec. II explains why MLaaS federation is necessary and possible. Sec. III describes the MLaaS federation problem formulation and explains why we need a combinatorial RL approach. Sec. IV introduces the three parts of Armol. We evaluate Armol in Sec. V. Sec. VI presents the related work, followed by the conclusion in Sec. VII.

II. Measurement & Motivation

In this section, we analyze the latency and accuracy of existing major MLaaS products, AWS Rekognition [5], Azure Computer Vision [6], and Google Cloud Vision AI [7], and demonstrate the great benefit in MLaaS federation and the feasibility for achieving this.

We conduct the measurement from March to July 2021 and rent the virtual machines (VMs) located in Singapore and the USA from AWS and Azure as clients to request these major MLaaS products. The types of AWS VMs are `t1.micro` and `t2.micro`, and the type of Azure VMs is `Standard B2s`. The above VMs have similar CPU, memory, storage, and network bandwidth. We request these services via Python SDK and capture the TCP packets by `tcpdump`. We take the object detection task as an example, and the accuracy metrics selected for this task are mean average precision (mAP), mAP with intersection over union (IoU) threshold 50% (AP_{50}), and mAP with IoU threshold 75% (AP_{75}) [8]. COCO Val 2017 [9] is chosen to test the performance of these services. For AWS Rekognition and Azure Computer Vision, we choose Singapore as the region for cloud service because the users prefer to choose the closest region to reduce the latency. Unlike the above two, Google Cloud Vision AI picks the region for the user, who cannot choose the region themselves.

TABLE I: AP of different MLaaS providers.

Provider	mAP	AP_{50}	AP_{75}
AWS	18.81	28.88	20.84
Azure	15.10	24.38	16.14
GCP	16.23	23.03	18.12

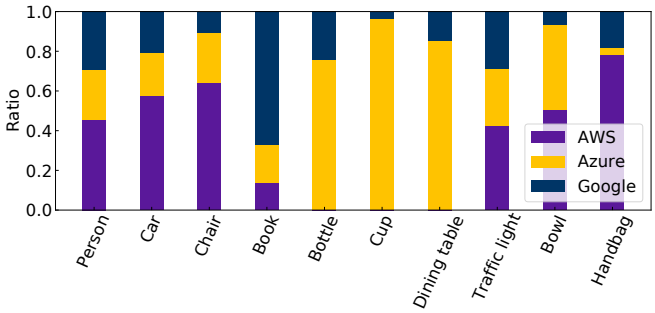


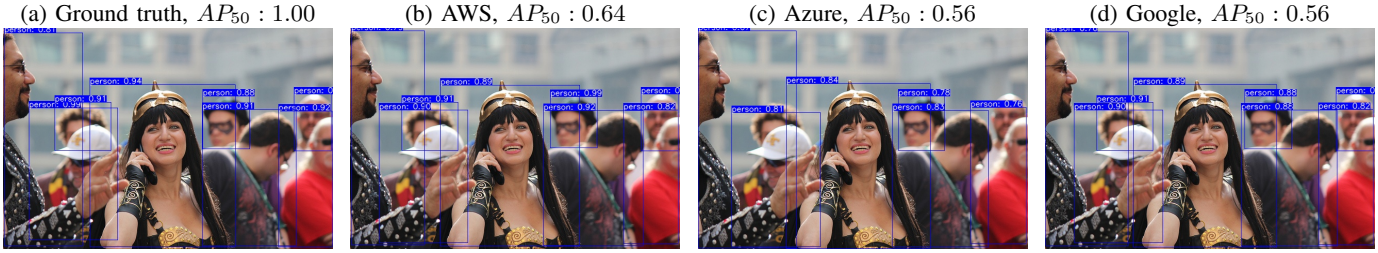
Fig. 1: The AP comparison of AWS Rekognition, Azure Computer Vision, and Google Cloud Vision AI on top-10 frequent categories of COCO Val 2017.

A. Why We Need MLaaS Federation

In Tab. I, we compare the AP on predictions of the COCO Val 2017 from AWS Rekognition, Azure Computer Vision, and Google Cloud Vision AI. We find that Azure has the worst performance on average. However, it does not mean that Azure performs poorly on every category in the dataset. We select top-10 frequent categories in COCO Val 2017 and compare the AP_{50} of the predictions from AWS, Azure, and Google on these categories. In Fig. 1, AWS is the best for categories such as "person", "chair", "car", and "handbag". Azure is the best for categories such as "cup", "bottle", and "dining table" while AWS did not identify any objects on these three categories. Google is the best on category "book". *These phenomena indicate that for input with different features, the most appropriate MLaaS provider differs.*

We next reveal the benefit of MLaaS federation. Following the ensemble strategies which will be introduced in Sec. IV-D, we have the AP_{50} of AWS, Azure and Google are 0.64, 0.56 and 0.56, and the AP_{50} of the ensemble predictions from three MLaaS providers is 0.68, as is demonstrated in Fig. 2. We can see that the ensemble predictions from three MLaaS providers have higher AP_{50} than the prediction from a single provider, verifying that the federation of MLaaS providers can provide more accurate prediction. In addition, by comparing Fig. 2e and Fig. 2h, we find that the ensemble predictions of AWS and Azure ($AP_{50} = 0.71$) is better than the ensemble predictions of three cloud providers ($AP_{50} = 0.68$). *These phenomena suggest that adding multiple MLaaS providers to inference can achieve higher accuracy than a single one. Still, more*

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



(e) AWS+Azure, AP_{50} : 0.71 (f) AWS+Google, AP_{50} : 0.69 (g) Azure+Google, AP_{50} : 0.67 (h) Three providers, AP_{50} : 0.68

Fig. 2: Detections and AP_{50} of different cloud combinations.

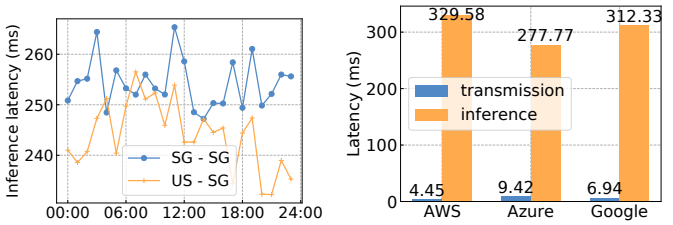
MLaaS providers added do not mean that we can gain higher accuracy.

B. Why MLaaS Federation is Possible

By analyzing the TCP packets, we discover that the latency of a request consists of the transmission latency and the inference latency. The transmission latency is determined by the input data size and the round trip time (RTT) between the location of the client and the region of the MLaaS provider. The inference latency is determined by the MLaaS itself, which is independent of the network conditions. Considering that the size of returned data is very small, the download time is negligible.

We compare the inference latency parsed by our TCP packets in two routes, namely requesting MLaaS in Singapore from Singapore (SG-SG) and requesting MLaaS in Singapore from the US (US-SG). Both routes request MLaaSes within the region of Singapore, so theoretically, the inference latency should be similar. By analyzing the measurement results in Fig. 3a, we find that the inference latency of both SG-SG and US-SG is similar within 24 hours in a day, which proves that the way we divide the total latency is correct.

Since the user must send requests to multiple MLaaS providers, we consider the latency in such a case. The user device transfers input data to various MLaaSes by HTTPS, which indicates that the user device sends n inputs to the MLaaSes sequentially via the same route in the transmission phase. Thus, the transmission latency is equal to the sum of the time to send the n inputs. In the inference phase, n MLaaSes predict the results in parallel, so the inference latency equals the maximum of the inference time among the n MLaaSes. In Fig. 3b, we find that the transmission latency is much smaller than the inference latency. With sufficient network bandwidth, the above phenomenon indicates that although our transmission latency increases linearly with the number of MLaaS providers, the total latency will not increase linearly with MLaaS providers.



(a) (b)

Fig. 3: The inference latency in both routes (SG-SG and US-SG) are similar in 24 hours, which indicates that the way we divide the total latency is correct. The transmission latency is much less than inference latency.

III. System Model and Problem Formulation

MLaaS can be seen as a function that gets input data, such as an image, a text or a speech, and returns the prediction of the input, such as image category, translated text or text generated by the input speech. The specific forms of the input and output depend on the task targeted by MLaaS. Here we generalize it and assume that there are a set of inputs $\mathbf{I} = \{I_1, I_2, ..., I_T\}$ to be processed by N available MLaaS providers. We denote $\mathbf{a}_t = [a_{t,1}, a_{t,2}, ..., a_{t,N}], a_{t,i} \in \{0, 1\}, i \in \{1, ..., N\}$ as the combination of selected MLaaS providers, i.e., $\mathbf{a}_t \in \{0, 1\}^N$. We denote $c_{t,i}$ as the cost to request the i-th MLaaS provider M_i at timestamp t. Then the cost of the combination $\mathbf{a}_t$ can be denoted as $c_t = \sum_{i=1}^{N} c_{t,i} a_{t,i}$. We denote $P_{\mathbf{a}_t}$ as the ensemble prediction from selected MLaaS providers determined by action $\mathbf{a}_t$. Then the final predictions for all input data can be represented as $\mathcal{P} = [P_{\mathbf{a}_1}, P_{\mathbf{a}_2}, ..., P_{\mathbf{a}_T}]$. We denote $v_{\mathbf{a}_t}$ to represent the accuracy of the prediction $P_{\mathbf{a}_t}$.

We strive to identify the appropriate selection of MLaaS providers to maximize the accuracy and minimize the cost for all input data. In summary, the MLaaS federation problem (Ω) can then be formally formulated as:

$$\max \quad \sum_{t \in T} (v_t + \beta c_t), \tag{1}$$

$$\begin{aligned} s.t. \quad & \mathcal{F}(\mathcal{P}) \geq A_o, \\ & \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} c_{t,i} a_{t,i} \leq C_o, \\ & \textstyle\sum_{i=1}^{N} a_{t,i} \neq 0, t \in \{1, 2, ..., T\}, \\ & a_{t,i} \in \{0, 1\}, \\ & \mathcal{P} = [P_{\mathbf{a}_1}, P_{\mathbf{a}_2}, ..., P_{\mathbf{a}_T}], \end{aligned} \tag{2}$$

where β, usually non-positive, is a hyperparameter to adjust the preference between accuracy and cloud cost, $\mathcal{F}$ is a function mapping from the predictions of all input data to accuracy, A_o is the target accuracy, and C_o is the overall budget for processing the whole workload.

The MLaaS federation problem Ω is NP-complete. The goal of Ω is to maximize the accuracy while minimizing the cost. Assume we only consider the computation of the provider selection part, and we have T inputs and N MLaaS providers. Then we have to predict whether the combination of N MLaaS providers is optimal for each input or not. Even if we know the ground truth for each input, we need $O(2^N)$ comparisons to know the optimal combination corresponding to this input. Thus, the overall time complexity is $O(T \cdot 2^N)$.

In a practical application environment, the prior information of input and corresponding optimal MLaaS provider combination is seldom available. Model-based solutions may not sufficiently adapt to the request dynamics and make intelligent provider selection decisions. In addition, the model and the dataset used to train the model of MLaaS are updating with the evolution of deep learning algorithms, making it difficult to achieve global optimally, especially in considering a long-term optimization.

The recent success in combinatorial RL provides an alternative perspective for this problem. Combinatorial RL reduces the complexity of the provider selection part. The rich historical viewer request patterns offer invaluable data resources that could be utilized for a data-driven provider selection problem. Specifically, the learning-based approach can not only well capture the hidden dynamics of input data and the model behind MLaaS but also enable an end-to-end solution from input data to MLaaSes' combination decision. Given these unique advantages, we present a combinatorial RL-based approach to solving this problem in the next section.

IV. MLaaS Federation Framework

In this section, we present Armol, a combinatorial RL-based cost-effective MLaaS federation framework that adaptively makes decisions about which providers to request to maximize the accuracy while minimizing the cost. We start by introducing the workflow of our framework. We then present the details of the provider selection part, the word grouping part, and the ensemble part.

A. Framework Overview

We consider a typical edge scenario for object detection. As shown in Fig. 4, Armol receives an image at the beginning. In the provider selection part, we first extract the image features (i.e., state $\mathbf{s_t}$) at the edge-side client, and then we generate the proto action $\hat{\mathbf{a}}_\mathbf{t}$ based on the image features by using the actor-network trained on the soft actor-critic (SAC) algorithm [10]. However, the proto action is a fractional vector, we have to map it to a binary vector $\mathbf{a_t}$. Then the edge client requests the providers selected by action $\mathbf{a_t}$ and waits until receiving all the predictions from selected providers. Cloud 1, 2 to n are the available MLaaS providers. Next is the word grouping part. Since different cloud services may use different words to represent the same category, we need this part to identify and unify words with the same meaning into one form to ensure that the subsequent ensemble part can be performed correctly. Finally, there is the ensemble part, which aims to ablate duplicate predictions while retaining the correct ones. In total, there are 12 pathways to choose from, and we end up with the Affirmative-WBF path based on our measurements in Sec. II. Armol also generates reward r_t in the ensemble part, which will be store in replay buffer with the binary action vector $\mathbf{a_t}$, the state $\mathbf{s_t}$ and the next state $\mathbf{s_{t+1}}$. Only after going through the above modules, the final prediction can be drawn on the image. The remainder of this section covers the details of Armol.

B. Provider Selection: A Combinatorial RL Approach

We first describe the design of state, action, and reward for our MLaaS federation problem.

State. To facilitate the training of the model, we use a pre-trained MobileNet, a classical model on the image classification task, to extract the feature of input, which represents the state $\mathbf{s}_t$ obtained from the environment at timestamp t. This method is described on the left top of Fig. 4.

Action. A subset of N MLaaS providers can be represented by a vector $\mathbf{a}_t \in \mathcal{A} = \{0, 1\}^N$ and $\mathbf{a}_t \neq \{0\}^N$, where the i-th element 1 means that the i-th provider is in this subset, while 0 means not. i.e, $\mathbf{a}_t = [a_{1,t}, ..., a_{N,t}], a_{i,t} \in \{0, 1\}$. If we have N available MLaaS providers, then the size of action set $\mathcal{A}$ is $2^N - 1$. Thus, the action space of our MLaaS federation problem is an exponential multiple of N. When N is large, it is hard for RL algorithms with discrete action spaces to handle the action spaces with size $2^N - 1$. Thus, we have to solve this problem by mapping the $\hat{\mathbf{a}}$ from continuous action spaces to an element in discrete binary vector set $\mathcal{A}$:

$$\tau : \mathbb{R}^n \to \mathcal{A}, \tag{3}$$

$$\tau(\hat{\mathbf{a}}) = \arg\min_{\mathbf{a} \in \mathcal{A}} |\mathbf{a} - \hat{\mathbf{a}}|^2, \tag{4}$$

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

Reward. There are two modes of the training process, offline and online, and the definition of rewards in the two

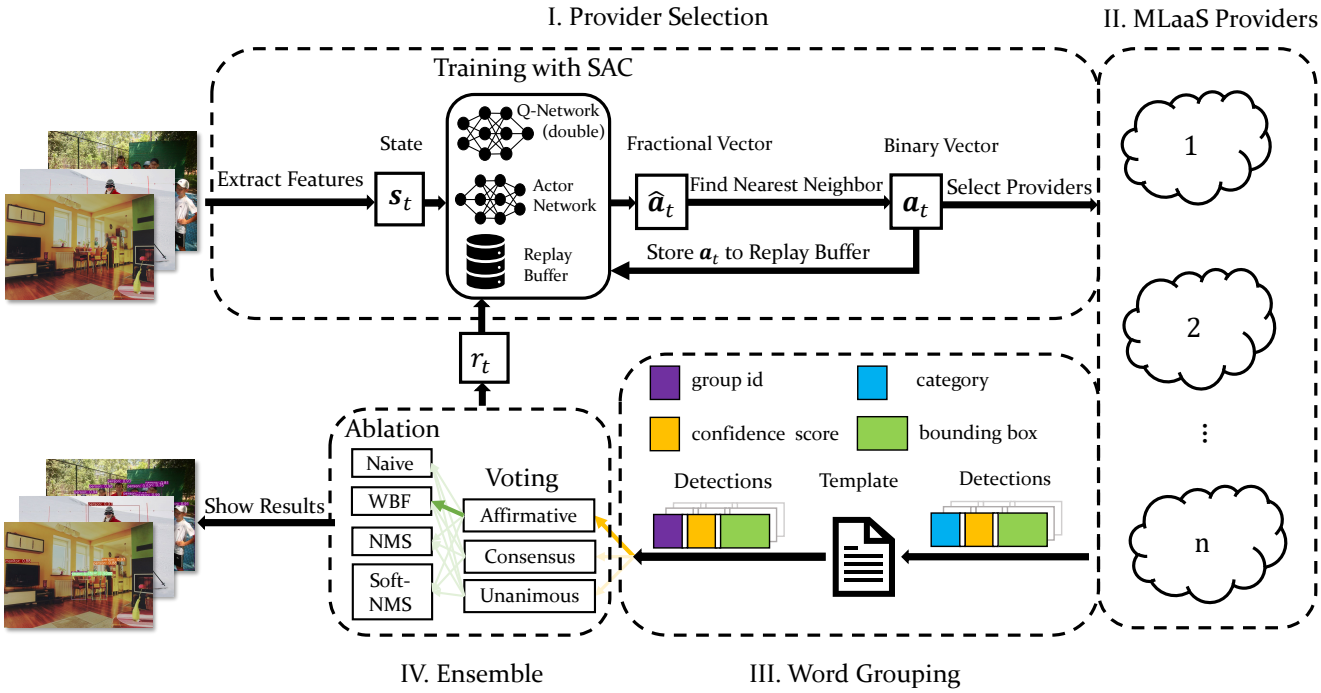


Fig. 4: The overview of Armol. For the object detection task, we need to group the category names with the same meaning into exact representation in the word grouping part; we need to ablate the redundant detections in the ensemble part.

modes are different. As described in [8], we need the ground truth to calculate the mAP. However, in practical applications, not all input images have ground truth. For images without ground truth, we use the ensemble prediction of N MLaaS providers as the ground truth. As shown in Fig. 2, although the mAP of N-providers ensemble prediction is not optimal, it is still better than the prediction of a single provider. Thus, it is feasible to use the ensemble prediction of all available MLaaS providers as ground truth. In addition to increase the mAP, we also want to use as few MLaaS providers as possible to reduce the inference fee. In summary, the reward can be defined as follows:

$$r_t = v_t + \beta c_t, \tag{5}$$

where v_t is the AP_{50} of prediction, c_t is the cost to request the subset of MLaaS providers selected by action $\mathbf{a}_t$, β is a hyperparameter, usually a non-positive number, to ensure that the action with a lower cost is selected. It is possible that providers selected by action $\mathbf{a}$ will not return any prediction, for which case we define the reward as -1.

We leverage SAC to train the RL agent. SAC is an off-policy actor-critic algorithm based on the maximum entropy RL framework. [10] explains the principle of SAC. Thus, we describe the details of training the RL agent next.

The algorithm of training the RL agent is proposed in Algo. 1. First, we initialize the replay buffer $\mathcal{B}$, the hyperparameter β, and the parameters for two Q-networks, two target Q-networks, and an actor-network. We use a fully connected network (FCN) with two hidden layers to represent the above networks, and the difference between the Q-network and actor-network is the input and output layers. Our training algorithm makes use of two soft Q-functions to mitigate positive bias in the policy improvement step that is known to degrade the performance of value-based methods [11].

Second, for each step, we observe the input image and extract the feature as the state. We select the action $\hat{\mathbf{a}}$ by policy $\pi_\theta(\cdot|\mathbf{s})$, then map $\hat{\mathbf{a}}$ to a binary action $\mathbf{a}$ and execute $\mathbf{a}$ in the environment to observe the next state $\mathbf{s}'$, reward r, and done signal d. We next store $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', d)$ to replay buffer $\mathcal{B}$.

Finally, if it is the step to update the networks, we sample a batch of transitions from replay buffer $\mathcal{B}$. The target of Q-network is given by:

$$y(r, \mathbf{s}', d) = \\ r + \gamma(1-d)\left(\min_{j=1,2} Q_{\phi_{targ,j}}(\mathbf{s}', \tilde{\mathbf{a}}') - \alpha \log \pi_\theta(\tilde{\mathbf{a}}'|\mathbf{s}')\right), \tag{6}$$

where $\tilde{\mathbf{a}}'$ is sampled from π_θ:

$$\tilde{\mathbf{a}}' \sim \pi_\theta(\cdot|\mathbf{s}'). \tag{7}$$

Then we can update two Q-networks $Q_{\phi_i}, i = 1, 2$ by one step of gradient descent using:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(\mathbf{s},\mathbf{a},r,\mathbf{s}',d) \in B} (Q_{\phi_i}(\mathbf{s}, \mathbf{a}) - y(r, \mathbf{s}', d))^2, \tag{8}$$

and update the policy by one step of gradient ascent using:

$$\nabla_\theta \frac{1}{|B|} \sum_{\mathbf{s} \in B} \left(\min_{i=1,2} Q_{\phi_i}(\mathbf{s}, \tilde{\mathbf{a}}_\theta(\mathbf{s})) - \alpha \log \pi_\theta(\tilde{\mathbf{a}}_\theta(\mathbf{s}) | \mathbf{s}) \right), \quad (9)$$

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:

$$\phi_{targ,i} \leftarrow \rho \phi_{targ,i} + (1 - \rho)\phi_i. \quad (10)$$

Note that our implementation of SAC omits the extra value function because Q-function and value function can represent each other [10]. This part is in the top center of Fig. 4, Armol will request the MLaaS providers selected by the action $\mathbf{a}_t$ next.

Algorithm 1: Training the RL agent with SAC

```
Initialize policy parameters θ;
Initialize Q-function parameters φ1, φ2;
Initialize replay buffer B;
Initialize hyperparameter β;
Set target Q-function parameters equal to main
  parameters φ_targ,1 ← φ1, φ_targ,2 ← φ2;
for time=1,... do
    Observe an input image, extract state s and select
      action â ~ π_φ(·|s);
    Get the nearest binary vector a of â in l2 distance;
    Request the MLaaS providers selected by a;
    Store (s, a, r, s', d) to replay buffer B;
    if it's time to update then
        for j in range(update times) do
            Randomly sample a batch of transitions
              B = {(s, a, r, s', d)} from B;
            Compute targets for the Q_φ1, Q_φ2 using
              Eq. 6;
            Update Q_φ1, Q_φ2 by one step of gradient
              descent using Eq. 8;
            Update policy by one step of gradient
              ascent using Eq. 9;
            Update target Q-networks using Eq. 10;
        end
    end
end
```

C. Word Grouping Part

After receiving the predictions from the selected providers, we need to standardize the presentation of the returned predictions to prevent ambiguity. This part is task-oriented since different tasks have different outputs. Here we take object detection, a classical computer vision task, as an example.

As we mentioned in Sec. III, an object detection cloud service can be seen as a function that 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 a bounding box b_i. The length of detection list D, namely l_d, represents the number of objects detected on this image. However, the different services may return the different category names for the object in the same category, i.e., "motorbike" vs. "motorcycle". It is clear that these category names have the same meaning, so we propose a feasible algorithm to aggregate the words with the same meaning into one group.

The following are the details. First of all, the user must provide a template T that contains all the category names they need. Here we use the 80 categories of the COCO dataset as T and find the close synonyms of category names in T based on the synonyms dataset extracted from WordNet. Subsequently, based on the measurement results, we collect the category names from all MLaaS providers as set A. However, we find that the synonyms from WordNet are not enough to cover all words in A, so we manually add the missing words within set A to the 80 groups. After that, we discard the remaining words in set A that are irrelevant to the 80 categories in the COCO dataset. For words in the same group, we consider they have the same meaning when used as nouns. Finally, a single detection $\mathbf{d}_i$ can be given by a triple $[n_i, f_i, b_i]$, where n_i is the group index of the corresponding category l_i. Only with this problem solved can we compare the accuracy of different MLaaS providers and ensemble the predictions correctly. This part is on the central bottom of Fig. 4, where the client has received the predictions from the selected MLaaS providers.

D. Ensemble Part

This ensemble part is also task-oriented. Fig. 2 illustrates the mAP of different combinations of three providers. We find that a single MLaaS provider has low mAP while the ensemble of multiple MLaaS providers reaches excellent performance. Therefore, based on the measurement results in Sec. II, we propose a novel strategy to ensemble the predictions from different object detection service providers. We divide this part into two steps: voting and ablation.

Voting methods. Common voting methods include "Affirmative", "Consensus" and "Unanimous" [12]. To conduct the voting methods, we need to standardize the presentation of the predictions from different MLaaS providers first. Then we group the detections of the image into $G = [g_1, g_2, ..., g_r]$, where $g_i, i \in \{1, 2, ..., r\}$ is a list of detections and r represents the total number of objects detected by N cloud service providers. For detections $d_p, d_q \in g_i$, they must confirm that $IoU(b_p, b_q) > 0.5$ and $n_p = n_q$. $IoU(a, b)$ is calculated by dividing the area of intersection between box a and box b by the area of union between a and b. Then we adjust the three voting methods to our work, which are described as follows.

- Affirmative. This method keeps all groups in G, which means that the detection is valid whenever one of the clouds says that a region contains an object.
- Consensus. This method holds the groups with a size greater than $N/2$, meaning most clouds must agree that a region contains an object.
- Unanimous. Only the groups whose size is equal to N are kept in this method, which means that all the object

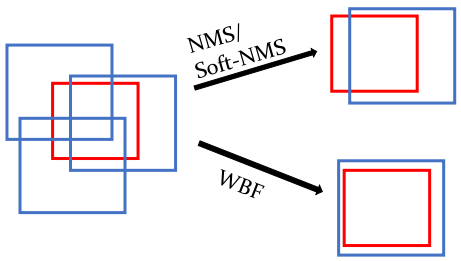


Fig. 5: Red boxes represent the ground truth. The left blue boxes represent the detections from AWS, Azure, and Google. The right top blue box is the box kept by NMS or Soft-NMS; the right bottom blue box is the box generated by WBF [15].

detection cloud services must agree to consider that a region contains an object.

We choose affirmative as the primary voting method. Because the three MLaaS providers are more of a complementary relationship and their predictions do not overlap much, as analyzed in Sec. II. Therefore, consensus or unanimous methods may remove some of the true-positive results. Besides, the evaluation results from [12] on different models also indicate that the affirmative method is superior to other methods.

Ablation Methods. Boxes in a group may repeatedly express an object, increasing the number of false-positive predictions and leading to a lower mAP. To reduce useless boxes (i.e., false-positive predictions), Non-Maximum Suppression (NMS) [13], Soft-NMS [14], and Weighted Boxes Fusion (WBF) [15] are proposed. NMS only saves the box with the most significant confidence score among a group and discards all other boxes. However, NMS inevitably removes the detections of some highly overlapped objects. Thus, Bodla et al. propose Soft-NMS to solve this problem. Instead of completely removing the detections with high IoU, it reduces the confidence score of the detections proportional to the IoU value. However, both NMS and Soft-NMS discard redundant boxes and thus can not effectively produce averaged localization predictions from different models. WBF takes the weighted average of the box coordinates within a group as the retained box, where the weight is the confidence score of the boxes. Moreover, the confidence score of the retained box is the average of the confidence score of boxes within a group.

Our measurements in Sec. II find that the differences between AWS, Azure, and Google are relatively significant. As shown in Fig. 5, for the same object, the boxes of all three cloud services are inaccurate and scattered in three directions. If we use NMS or Soft-NMS methods to ablate the boxes, the box kept is still inaccurate. However, if we use WBF to fuse the boxes in all three directions, we can get a more accurate predicted box. Therefore, we decide to use the WBF method to ablate the duplicated boxes.

As described in Fig. 4, in the ensemble part, we first go through the voting method for each group of similar boxes and then remove the duplicate boxes by the ablation method.

V. Performance Evaluation

In this section, we conduct extensive experiments to evaluate the performance of the MLaaS provider selection part in Armol. Specifically, using real trace-driven evaluations, we demonstrate that the superiority of Armol over other benchmark approaches.

A. Setup and Methodology

Evaluation Setup. We collect the predictions of COCO Val 2017 from AWS Rekognition, Azure Computer Vision, and Google Cloud Platform Vision AI as the environment. We open-source the code and data in `https://github.com/ShuzhaoXie/Armol`. The inference cost for AWS and Google Cloud Vision AI is 0.001 USD per image, while the inference cost for Azure varies by about 10% with the region. Our measurements were done in Singapore, and Azure's price in this region is 0.001 USD. So in the following experiments, we set the inference cost of each request to an MLaaS as 0.001 USD. We implement the algorithm of the RL agent based on the SpinningUp [16] framework and add support for GPU training, which runs on a server with an NVIDIA 1080 Ti GPU card, an Intel(R) Xeon(R) CPU E5-2650 v4@2.20GHz, and 64 GB memory. The RL environment is implemented in Python for compatibility. The learning rate η for the actor-network and the Q-networks is 0.0001, respectively. We set γ as 0.9, and α as 0.2. In order to gain the best mAP, we set β as 0. The batch size is 1000. The training epoch is 100, and the steps per epoch is 2000.

Baseline Methods. We compare our approach with several baseline methods as follows.

- **Random-1**: This method only gives a random selection of MLaaS providers for each image.
- **Random-N**: This method chooses a subset of available MLaaS providers for each image randomly.
- **Ensemble-N**: This method aggregates the predictions of all MLaaS providers.
- **Armol-w/o gt**: This method uses the ensemble predictions of all MLaaS providers as the ground truth to generate the reward. The hyperparameter β is set as -0.1 to select the action with the lower cost of inference fee. The other hyperparameters are the same as Armol.
- **Armol-P**: This method means that we train the RL agent on proximal policy optimization (PPO) [17], which is a classical on-policy RL algorithm that is worth comparing.
- **Armol-T**: This method means that we train the RL agent on twin delayed deep deterministic policy gradient (TD3) [11]. TD3 is a classical deterministic policy training algorithm, the comparison with which can demonstrate the benefits of the maximum entropy property of SAC.
- **Upper Bound**: To reach the goal of gaining more mAP while spending less money, based on the measurements in Sec. II, we use a brute-force search algorithm to select the best combination, which is demonstrated in Algo. 2. The voting method is affirmative, and the boxes ablation method is WBF.

Algorithm 2: Brute Force Search Algorithm

Initialize set $\mathcal{D}$ to store the best detection of all images;
for *image* I_t *in* $\mathbf{I}$ **do**
  Initialize the max mAP $v_{max} = -1$;
  for *action* $\mathbf{a}$ *in* $\{0,1\}^N - \{0\}^N$ **do**
    Get and ensemble the detection $D_{\mathbf{a}}$ by action;
    Calculate the mAP $v_{D_{\mathbf{a}}}$ of detections $D_{\mathbf{a}}$;
    if $v \geq v_{max}$ **then**
      Update v_{max} to $v_{D_{\mathbf{a}}}$;
      Update the best action $\bar{\mathbf{a}}$ to $\mathbf{a}$;
      Update the best detection $\bar{D_{\mathbf{a}}}$ to $D_{\mathbf{a}}$;
    end
  end
  $\mathcal{D} \leftarrow \mathcal{D} \cup \bar{D_{\mathbf{a}}}$;
end
return $\mathcal{D}$.

Evaluation Metrics. We use the following metrics:

- **Cost**: We denote this metric as average cost c_e in a test episode, in unit of 10^{-3} USD:

$$c_e = \frac{\sum_{t=0}^{T-1} c_t}{T} \tag{11}$$

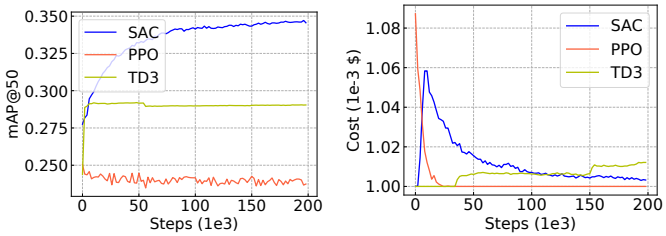
- $\mathbf{AP_{50}}$: This metric means the average precision of predictions with a 50% IoU threshold. We use AP_{50} instead of mAP is because we want to reduce the computation and speed up the training. Because mAP is the average of APs with IoU threshold from 50% to 95% with a 5% increase per step. AP_{50} is the average precision with a 50% IoU threshold, the computation of which is 10% of mAP. Besides, AP_{50} is also a standard metric in object detection tasks.

B. Evaluation Results

To understand the performance of the provider selection part, we conduct experiments to demonstrate: 1) the superiority of training RL agent on SAC; 2) the feasibility to leverage predictions from all MlaaS providers as ground truth; 3) the scalability of our combinatorial RL approach.

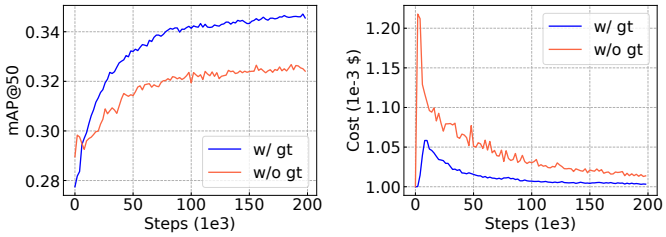
Superiority of training RL agent on SAC. Tab. II shows the metric statistics of Armol on SAC, PPO and TD3. We can see that the mAP of Armol on SAC is greater than PPO and TD3, and the average cost is lower than TD3. In Fig. 6, we can find that the SAC converges better and faster. Compared to Random-N, Armol on SAC have 3.09% higher mAP and 41.75% less cost of inference fee. Compared to Ensemble-N, the Armol with ground truth (Armol-w/ gt) gains equal mAP and reduces 67% cost of inference fee.

Feasibility to leverage predictions from three providers as ground truth. Tab. II shows the without ground-truth method reduces 66% cost compared to all federated predictions with only 4.3% lower mAP. As can be seen in Fig. 7, Armol-w/o gt converges stably, although both AP_{50} and cost are not as good as Armol-w/ gt in the training process.



(a) (b)

Fig. 6: The label of Y-axis on left is $mAP@50$ of the whole episode, which is the same as AP_{50}. The label of Y-axis on right is average cost per test episode. The training algorithm test the RL agent at the end of every epoch. The figure shows the training processs of Armol on SAC, Armol on PPO and Armol on TD3.



(a) (b)

Fig. 7: w/ gt means the "Armol with ground truth"; w/o gt means the "Armol without ground truth".

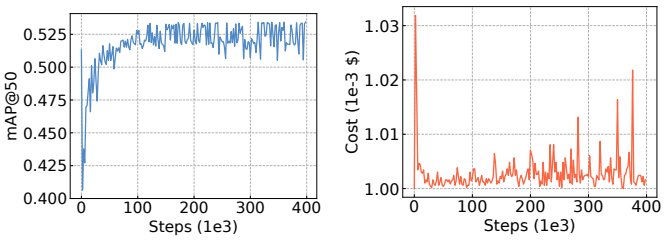
TABLE II: Performance metrics of different baseline methods. "AWS" means that how many images in the test episode choose the AWS, while "Azure" and "Google" have the same meaning. "Armol-w/ gt" has the same meaning with "Armol" and "Armol on SAC". The unit of cost is 10^{-3} USD.

Methods	mAP	$\mathbf{AP_{50}}$	Cost	AWS	Azure	Google
Random-1	15.75	24.49	1.000	1690	1605	1657
Random-N	18.66	28.89	1.722	2858	2863	2809
Ensemble-N	21.75	34.69	3.000	4952	4952	4952
Armol-w/ gt	**21.75**	**34.71**	**1.003**	**2863**	**950**	**1156**
Armol-w/o gt	20.81	32.68	1.016	3426	683	924
Armol-PPO	14.99	25.05	1.087	1300	2541	1543
Armol-TD3	18.90	29.20	1.006	4843	114	26
Upper Bound	23.83	37.70	1.202	3881	1126	944

Scalability of combinatorial RL approach. To test the scalability of our approach towards a more significant number of MLaaS providers, we add the results of Alibaba Cloud Object Detection [18] and synthetic six more MLaaS providers. The details of these simulated MLaaS providers are available in our github reposity. We index AWS, Azure, Google, Alibaba, and six simulated providers as MLaaS 0-9. In Tab. III, we find that the ensemble predictions of 10 MLaaS providers are lower than MLaaS 5. We suggest that the reason for this phenomenon is because the AP_{50} of MLaaS 5 is 20%-30% higher compared to the other MLaaS providers,

TABLE III: Performance metrics of different simulated MLaaS. The unit of cost is 10^{-3} USD.

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	Armol	53.44	1.002



(a) (b)

Fig. 8: Training process with 10 available MLaaS providers (1023 actions). The $mAP@50$ (AP_{50}) and cost converges stably.

which cannot provide more true positive results and only increases the number of false positive results in the ensemble predictions. However, as shown in Tab. III, the AP_{50} of Armol is slightly better than MLaaS 5 with almost the exact cost, which indicates that although the AP_{50} of the MLaaS providers varies greatly, our algorithm still selects the better combinations. In Fig. 8, our combinatorial RL approach still stably converges with ten providers (1023 actions) in both AP_{50} and cost.

VI. Related Work

A. Measurements on Machine Learning Services

There are previous works that measure the inference accuracy and latency of machine learning models [2], [3], but these measurements are mainly on user-known models instead of machine learning services. In addition, [19] aims to measure the machine learning training platforms instead of machine learning inference services. There is also a measurement work [20] on older machine learning services limited to decision trees, SVMs, and multi-layer fully connected neural network services, which is much different from the next generation machine learning services that are now being promoted with models that are transparent to users and in favor of deep learning.

B. Cloud Federation

Cloud federation comprises services from different providers aggregated in a single pool supporting three basic interoperability features-resource migration, resource redundancy, and complementary resources resp. services [21]. In the past, cloud federation represents integrating explicit resources, such as storage and compute resources. In contrast, we integrate the implicit resources, which are the training data and model behind the online public machine learning services. Furthermore, the past concept of cloud federation enables further reduction of cost due to partial outsourcing to more cost-efficient regions [22]. However, we consider the reduction of cost only after reaching the highest mAP.

C. Reinforcement Learning with Combinatorial Action Spaces

Discrete, high-dimensional action spaces are common in applications such as natural language processing [23], text-based applications [24] and vehicle routing [25], but they pose a challenge for standard RL algorithms [26], because enumerating the action space when choosing the next action from a state becomes impossible. Recent remedies for this problem include selecting the best action from a random sample [23], approximating the discrete action space with a continuous one [27], [28], training an additional machine learning model to wean out sub-optimal actions [24], or formulating the action selection problem from each state as a mixed-integer program [25]. [29] adds the wolpertinger policy to the edge cache problem, but it is just an application and has no contribution to the policy. Our provider selection approach embeds the continuous action to the nearest neighbor in binary action space. To increase the probability of exploration, we used SAC for training instead of DDPG.

VII. Conclusion

In this paper, we propose Armol, a novel cost-effective MLaaS federation framework that leverages deep combinatorial RL to boost the average precision of federated object detection services so as to minimize the cost. Through our analysis on the predictions of COCO Val 2017 from AWS Rekognition, Azure Computer Vision, and Google Vision AI, we demonstrate that the mAP of federated MLaaS providers is higher than a single provider, and more MLaaS providers do not mean higher accuracy. Inspired by the recent advances in RL algorithms for combinatorial action spaces, we propose a combinatorial RL-based approach to decide on how to choose the best combination of available MLaaS providers for input. The evaluation further demonstrates the strengths of our approach.

Acknowledgment

Shuzhao Xie thanks Chen Tang, Jiahui Ye, Shiji Zhou, and Wenwu Zhu for their help in making this work possible. This work is supported in part by NSFC (Grant No. 61872215), and Shenzhen Science and Technology Program (Grant No. RCYX20200714114523079). Yifei Zhu's work is funded by the SJTU Explore-X grant. We would like to thank Tencent for sponsoring the research.

REFERENCES

[1] (2021). "Machine learning as a service (mlaas) market - growth, trends, covid-19 impact, and forecasts (2021 - 2026)," [Online]. Available: https://www.reportlinker.com/p06106023/Machine-Learning-as-a-Service-MLaaS-Market-Growth-Trends-COVID-19-Impact-and-Forecasts.html (visited on 07/28/2021).

[2] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, *et al.*, "Mlperf inference benchmark," in *ACM/IEEE ISCA*, 2020, pp. 446–459.

[3] H. Zhang, Y. Huang, Y. Wen, J. Yin, and K. Guan, "Inferbench: Understanding deep learning inference serving with an automatic benchmarking system," *arXiv preprint arXiv:2011.02327*, 2020.

[4] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[5] (2021). "Amazon rekognition," [Online]. Available: https://aws.amazon.com/rekognition/ (visited on 07/18/2021).

[6] (2021). "Azure computer vision," [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/ (visited on 07/18/2021).

[7] (2021). "Google vision ai," [Online]. Available: https://cloud.google.com/vision (visited on 07/18/2021).

[8] (2021). "Average precision," [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/htmldoc/devkit_doc.html (visited on 07/18/2021).

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV 2014*, Springer, pp. 740–755.

[10] 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*, pp. 1861–1870.

[11] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *ICML 2018*, pp. 1587–1596.

[12] A. Casado-Garcia and J. Heras, "Ensemble methods for object detection," in *ECAI 2020*, pp. 2688–2695.

[13] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *IEEE CVPR 2017*, pp. 4507–4515.

[14] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms–improving object detection with one line of code," in *ICCV 2017*, pp. 5561–5569.

[15] R. Solovyev, W. Wang, and T. Gabruseva, "Weighted boxes fusion: Ensembling boxes for object detection models," *arXiv preprint arXiv:1910.13302*, 2019.

[16] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] (2021). "Alibaba cloud object detection," [Online]. Available: https://vision.aliyun.com/objectdet (visited on 07/18/2021).

[19] Y. Yao, Z. Xiao, B. Wang, B. Viswanath, H. Zheng, and B. Y. Zhao, "Complexity vs. performance: Empirical analysis of machine learning as a service," in *Proc. IMC 2017*, pp. 384–397.

[20] Y. Liu, H. Zhang, L. Zeng, W. Wu, and C. Zhang, "Mlbench: Benchmarking machine learning services against human experts," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1220–1232, 2018.

[21] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," *Cloud Computing*, vol. 2011, pp. 32–38, 2011.

[22] M. Giacobbe, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Towards energy management in cloud federation: A survey in the perspective of future sustainable and cost-saving strategies," *Computer Networks*, vol. 91, pp. 438–452, 2015.

[23] J. He, M. Ostendorf, X. He, J. Chen, J. Gao, L. Li, and L. Deng, "Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads," in *Proc. EMNLP 2016*, pp. 1838–1848.

[24] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," in *NeurIPS 2018*, pp. 3566–3577.

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

[26] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[27] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[28] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf, "Deep reinforcement learning with a natural language action space," in *Proc. ACL 2016*, The Association for Computer Linguistics.

[29] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *IEEE CISS 2018*, pp. 1–6.