论文汇报

汇报人：张振　导师：何昕

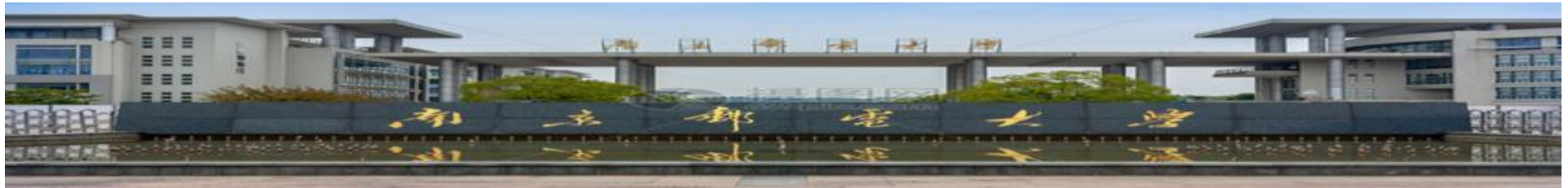
2025.5.6

南京郵電大學 Nanjing University of Posts and Telecommunications

Online Scheduling of Edge Multiple-Model Inference with DAG Structure and Retraining

Yifan Zeng* , Ruiting Zhou* †, Lei Jiao‡, Renli Zhang* ,
* School of Cyber Science and Engineering, Wuhan University, China
†School of Computer Science and Engineering, Southeast University, China
‡Center for Cyber Security and Privacy, University of Oregon, USA

INFOCOM 2025



背景

智能设备的多样化极大地丰富了推理数据的来源。为了及时处理来自各种智能设备的图像、视频和音频等数据流，已经开发了多种多模型应用程序，包括生活日志记录、社交媒体、视频分析等等。这些应用程序使用以有向无环图（DAG）组织的多个模型来分析数据，以实现各种推理目标。

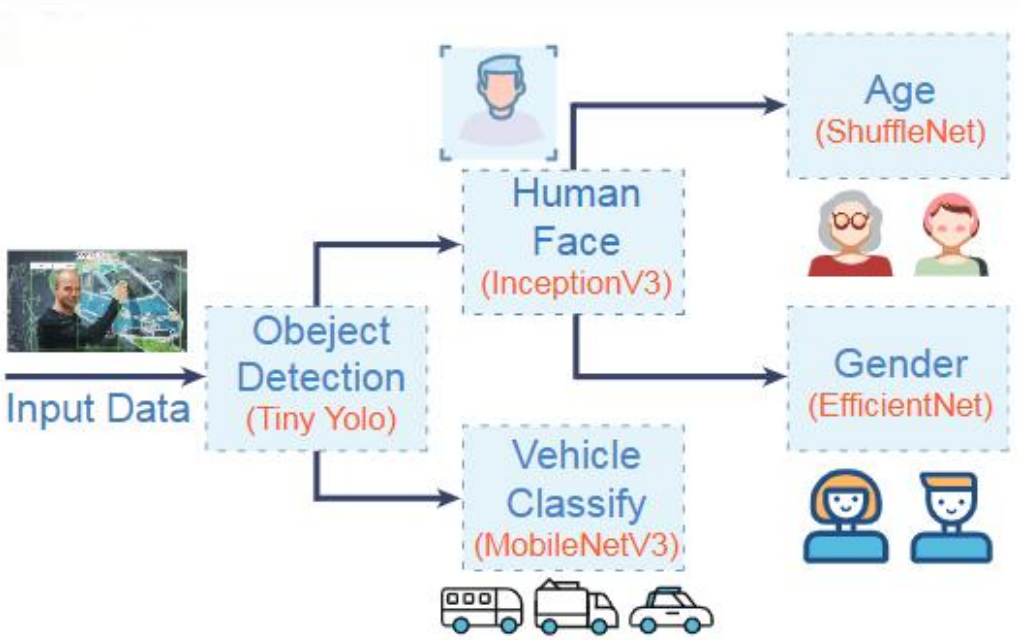


Fig. 1: DAG of the lifelogging application.

数据漂移的影响

由于云计算会引入过多的通信延迟，为了保证多模型应用程序的低响应时间，通常采用边缘服务器来处理这些请求。但是边缘服务器资源有限（如 GPU），通常部署一些结构较浅、参数较少的小型或压缩模型。这些模型无法正确识别推理数据与训练模型所用数据之间存在显著差异，导致模型推理精度下降。

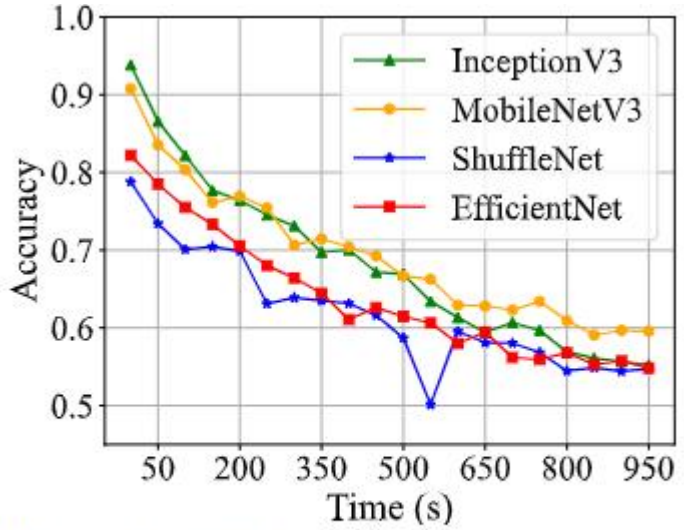


Fig. 2: Inference accuracy of different models (over time).

模型重训练对精度的提升

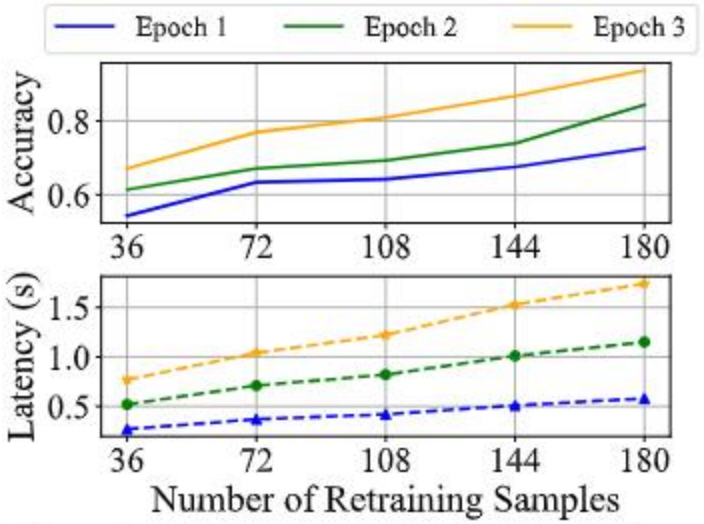


Fig. 3: Impact of the number of retraining samples.

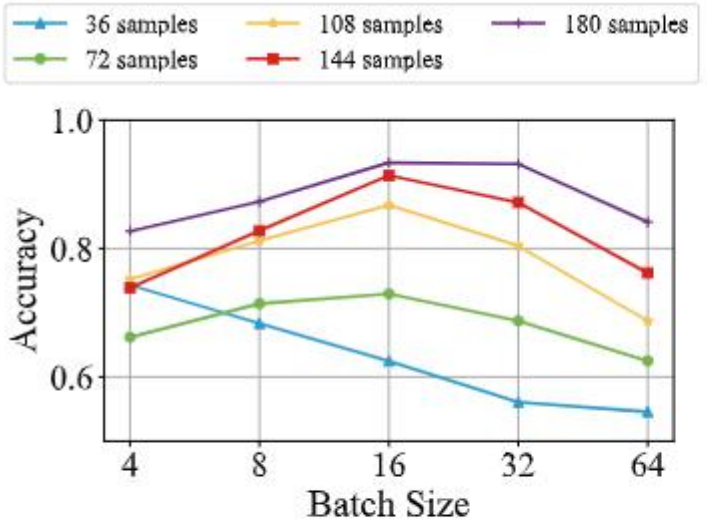


Fig. 4: Retraining accuracy of different batch size.

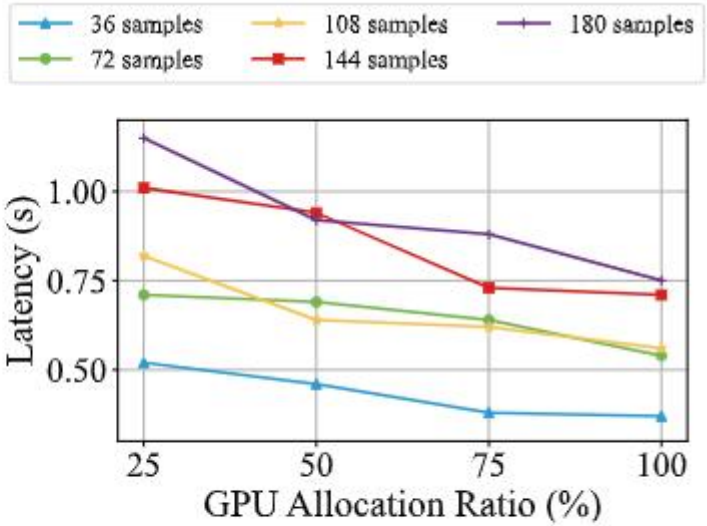


Fig. 5: Retraining latency of different resource.

本文的主要贡献

- 本文证明了通过重训练可以在短时间内提高推理精度。然后，本文在请求 DAG 中添加了模型重训练任务的节点，并将调度问题公式化为最小化在线请求的完成时间和最大化精度。
- 设计 SRS 算法来处理带模型重训练的单请求调度。它通过调度推理任务来估计推理资源的使用情况，然后在资源约束下选择最优的重训练配置，并一起卸载重训练和推理任务。并证明了SRS的理论上界。MRS 作为 SRS 的扩展，用于管理多请求调度。它形成任务列表，选择初始任务创建候选集，并迭代地选择最早完成的任务，直到调度完所有请求。
- 本文在具有真实请求轨迹的边缘系统上进行了广泛的实验。结果表明，与三种最先进的算法相比，本文的算法推理精度提高了 25%，请求完成时间减少了 45%。

持续学习的边缘推理

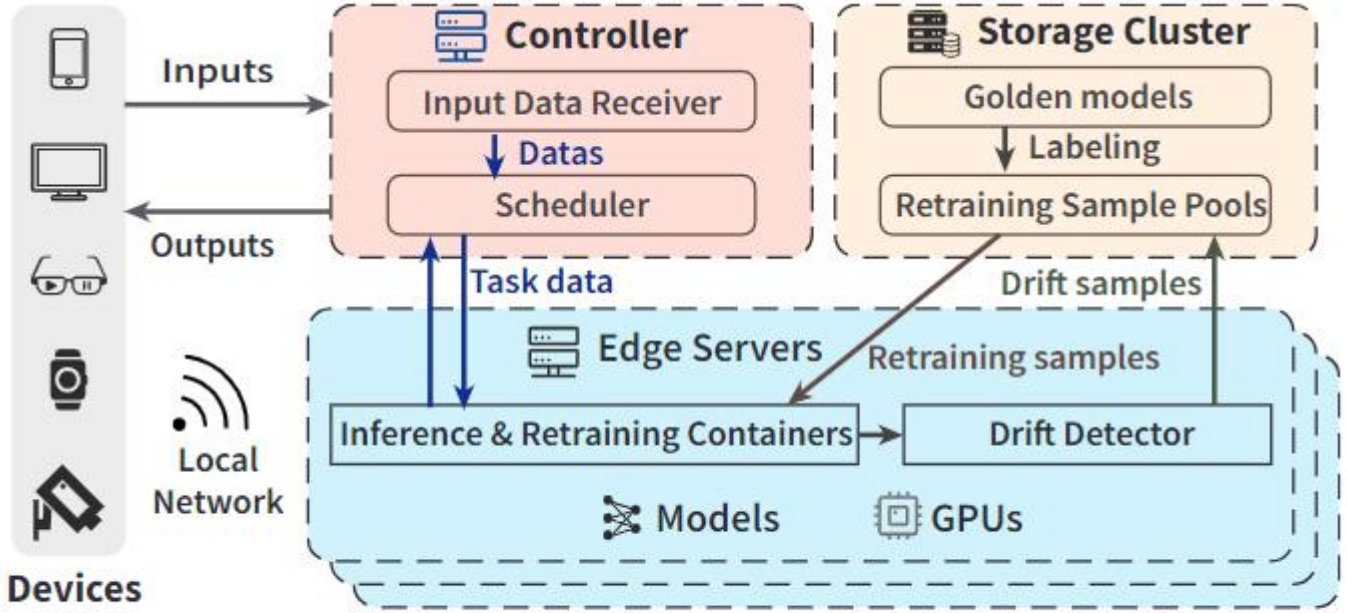


Fig. 6: System Overview.

模型重训练：对于每个推理模型，存储集群维护一个重训练样本池，持续捕获并标记漂移数据样本。在完成一次重训练迭代后，与重训练模型相关的样本池会被清空，并开始积累新的样本。此外，模型重训练过程中获得的更新模型参数随后会应用到每个边缘服务器上。

重训练配置：$u_h = (b_h, n_h, o_h)$依次为训练样本的数据大小，重训练工作量（即 GPU 时长），重训练所需的计算资源（如 GPU）。

决策变量：对于到达系统的请求中的每个任务 v_k^i 做出的决策包括：i) $x_k^{i,j} \in \{0,1\}$表示任务v_k^i是否在服务器e_j上处理 ii) $y_k^{i,h} \in \{0,1\}$表示是否使用重训练配置 u_h 对模型m_k^i进行重训练。

带重训练任务的扩展 DAG

重训练节点：本文用$|V|$表示请求r_k中初始推理任务的数量，那么模型m_k^i 的重训练任务可以表示为 $v_k^{|V|+i}$

对于使用重训练配置 u_h 的重训练任务$v_k^{|V|+i}$其所需的计算资源 $c_k^{|V|+i} = \sum_{u_h \in \mathcal{U}} y_k^{i,h} o_h$,工作量$w_k^{|V|+i} = \sum_{u_h \in \mathcal{U}} y_k^{i,h} n_h$
重训练数据大小 $d_k^{|V|+i} = \sum_{u_h \in \mathcal{U}} y_k^{i,h} b_h$ 。

模型推理精度更新为 $A_k^i = \sum_{u_h \in \mathcal{U}} y_k^{i,h} f(u_h, m_k^i) + (1 - \sum_{u_h \in \mathcal{U}} y_k^{i,h}) a_k^i$， f()模拟了模型重训练配置与训练后精度之间的关系。

DAG 扩展：

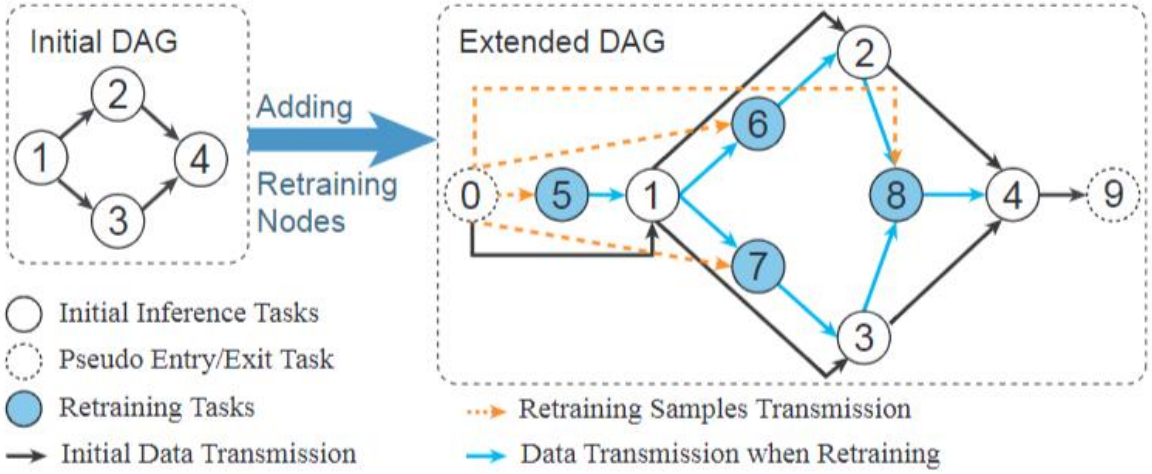


Fig. 7: Adding Retraining Task Nodes.

$$d_k^{i',|V|+i} = \sum_{u_h \in \mathcal{U}} y_k^{i,h} d_k^{i',i}$$

$$d_k^{|V|+i,i} = \sum_{u_h \in \mathcal{U}} y_k^{i,h} (\sum_{i' \in pre_k(i)} d_k^{i',i} + d(m_k^i))$$

$$d_k^{0,|V|+i} = d_k^{|V|+i}$$

请求完成时间

执行时间：$$t_k^{exe}(i) = \sum_{e_j \in \mathcal{E}} \frac{x_k^{i,j} w_k^i}{c_k^i S_j}$$

传输时间：$$t_k^{trans}(i', i) = \sum_{e_{j'} \in \mathcal{E}} \sum_{e_j \in \mathcal{E}} \frac{x_k^{i',j'} x_k^{i,j} d_k^{i',i}}{D_{j',j}}, i' \in pre_k(i) \qquad t_k^{trans}(0, i) = \sum_{e_j \in \mathcal{E}} \frac{x_k^{i,j} d_k^{0,i}}{D_j}$$

完成时间：本文用 $T_k^s(i)$ 和 $T_k^f(i)$ 表示任务 v_k^i 的开始时间和结束时间。

$$T_k^s(0) = T_k^f(0) = T_k^a.$$
$$T_k^f(i) = T_k^s(i) + t_k^{exe}(i).$$

$$T_k^s(i) \geq T_k^{\tilde{f}}(i') + t_k^{trans}(i', i), \forall i' \in pre_k(i).$$

资源约束：$$\sum_{v_k^i \in g_j(t)} c_k^i \leq C_j.$$

问题建模

$$\min \quad P(X,Y) = \sum_{k} \Big(\max_{v_k^i \in \mathcal{V}_k} (T_k^f(i)) + \theta \sum_{v_k^i \in \mathcal{V}_k} (1 - A_k^i) \Big) \tag{9}$$

$$\text{s.t.} \quad \sum_{e_j \in \mathcal{E}} x_k^{i,j} = 1, \quad \forall i, \forall k, \tag{9a}$$

$$\sum_{u_h \in \mathcal{U}} y_k^{i,h} \leq 1, \quad \forall i \leq |V|, \forall k, \tag{9b}$$

$$y_k^{i,h} = 0, \quad \forall i > |V|, \forall k, \forall h, \tag{9c}$$

$$A_k^i \geq A_{\min}, \quad \forall i, \forall k, \tag{9d}$$

$$x_k^{i,j} \in \{0,1\}, \quad \forall i, \forall j, \forall k, \tag{9e}$$

$$y_k^{i,h} \in \{0,1\}, \quad \forall i, \forall h, \forall k, \tag{9f}$$

$$(6), (7), (8)$$

在线调度算法设计

算法 1：列表调度算法

任务 v_k^i 的优先级定义为：$\phi(i) = \max_{i' \in suc(i)} \{\bar{t}^{exe}(i) + t_{\max}^{trans}(i, i') + \phi(i')\}$

Algorithm 1 List Scheduling Algorithm

1: Calculate the priorities of all tasks in $\mathcal{V}$ with Eq.(10);
2: $q \leftarrow$ scheduling list of tasks following decreasing order of priorities;
3: **while** $q \neq \emptyset$ **do**
4: Select the first task v^i in the list q;
5: Calculate $T^f(i)$ on each edge server by Eq.(5)-(7);
6: Find the edge server $e_j \in \mathcal{E}$ that satisfies $\min_X T^f(i)$;
7: Assign task v^i to edge server e_j, $x^{i,j} = 1$;
8: Delete task v^i from list q;
9: **end while**;
10: **return** X;

在线调度算法设计

算法 2 单请求调度算法（SRS）

配置$u_h \in \mathcal{U}$的的变化值ΔP 定义为：

$$\begin{aligned}\Delta P(i,h) &= \frac{n_h}{o_h S_{j*}} + \frac{d^{i',i} + d(m^i)}{D_{j'*,j*}} + \frac{b_h}{D_{j*}} \\ &\quad + \theta(1 - A^i) - \theta(1 - a^i) \\ &= \frac{n_h}{o_h S_{j*}} + \frac{d^{i',i} + d(m^i)}{D_{j'*,j*}} + \frac{b_h}{D_{j*}} \\ &\quad + \theta(a^i - f(u_h, m^i)),\end{aligned}$$

Algorithm 2 Single Request Scheduling Algorithm (*SRS*)

1: Invoke Algorithm 1 to schedule initial inference tasks;
2: Convert the initial DAG into a single chain DAG in topological order;
3: **for** each $v^i \in \mathcal{V}$ **do**
4: Find the retraining configuration $u_h \in \mathcal{U}$ that satisfies $\min \Delta P(i)$ with Eq. (11)
5: Add the retraining node to the DAG with configuration u_h, $y^{i,h} = 1$
6: **end for**
7: Invoke Algorithm 1 to schedule extended DAG to get X;
8: **return** X, Y;

在线调度算法设计

算法 3 多请求调度算法（MRS）

Algorithm 3 Multiple Requests Scheduling Algorithm (*MRS*)

Initialize set $Q \leftarrow \emptyset$;
if new request r_k arrives **then**
    Invoke Algorithm 2 to get Y_k;
    Calculate the priorities of all tasks in $\mathcal{V}_k$ with Eq.(10);
    $q_k \leftarrow$ scheduling list of tasks following decreasing order of priorities;
    $Q \leftarrow Q \cup \{q_k\}$;
end if
while $Q \neq \emptyset$ **do**
    Choose the first task of each list q_k to construct set H;
    Select the earliest task v_k^i that can be processed from set H;
    Calculate $T_k^f(i)$ on each edge server by Eq.(5)-(7);
    Find the edge server $e_j \in \mathcal{E}$ that satisfies $\min_X T_k^f(i)$;
    Assign task v_k^i to edge server e_j, $x_k^{i,j} = 1$;
    Delete task v_k^i from list q_k;
    if $\exists q_k \in Q$ and $q_k = \emptyset$ **then**
        Delete list q_k from set Q;
    end if
end while;
return X, Y;

实验设置

测试平台搭建。本文使用容器构建了一个边缘计算环境，该环境由五台边缘服务器、一个存储集群和一个控制器组成。每台边缘服务器配备两块英伟达 A100 GPU，每块 GPU 有 80GB 内存。边缘服务器的 GPU 计算速度在 [9.7, 19.5] 万亿次浮点运算每秒之间。

本文使用多进程服务（MPS）为请求中的任务划分 GPU 资源。控制器和边缘服务器具有相同的 GPU 配置。存储集群由两台对象存储服务器（OSS）组成，每台服务器有 256GB 的存储空间，并配备一块英伟达 GTX 2060 GPU。

边缘服务器之间、边缘服务器与控制器之间的数据传输速率在 [3, 5] Gbps之间，边缘服务器与存储集群之间的数据传输速率在 [7, 10] Gbps之间。

请求到达。应用程序请求按照阿里巴巴的迹线数据的模式到达，该迹线数据反映了现实世界中的推理请求工作负载情况。本文将迹线数据中相同类型的请求视为来自同一应用程序的请求。然后，根据适当的比例对这些请求的到达时间进行缩放，以便与总运行时间相匹配。推理任务的数据大小在 [0.8, 1.6] MB范围内，而工作负载设置为 0.1 到 0.5 个 GPU 运行时间单位。

应用程序与模型。在实验的默认设置下，本文评估了五种应用程序：生活日志记录、图像处理、视频监控、社交媒体以及 TensorFlow 级联（TF cascade）。对于生活日志记录应用程序，本文使用 TinyYolo [16]、InceptionV3 [17]、MobileNetV3 [18]、ShuffleNet [19] 和 EfficientNet [20] 模型来完成此有向无环图（DAG）应用程序中的任务。数据集是 Adience 图像数据集 [21]，其中包含 26580 张照片和 2284 个拍摄对象。该数据集的 40% 被用于对模型进行预训练。其他四种应用程序的模型和数据集均来自 [22]。在针对不同数量应用程序的实验中，我们从 [8] 获取了额外的应用程序和数据集。在实验过程中，来自同一应用程序的请求具有相同的有向无环图结构。

Baselines

- AdaInf [8]：AdaInf 采用增量式重训练，并考虑请求的服务级别目标（SLO）。它在重训练和推理之间分配 GPU 时间以满足服务级别目标，然后根据每个请求中重训练任务的影响程度，为这些重训练任务分配 GPU 时间。

- Ekya [7]：Ekya 是一种启发式算法，它联合进行推理和重训练的调度。它采用贪心策略来选择能使推理精度最大化的重训练配置以及相应的资源分配方案。

- PASS [23]：PASS 是一种基于优先级的有向无环图（DAG）调度算法，它为每个任务计算优先级，并按照优先级顺序为任务选择边缘服务器。由于 PASS 不涉及模型重训练，我们已将其配置为定期对所有模型进行重训练，以实现最高的精度。

实验结果

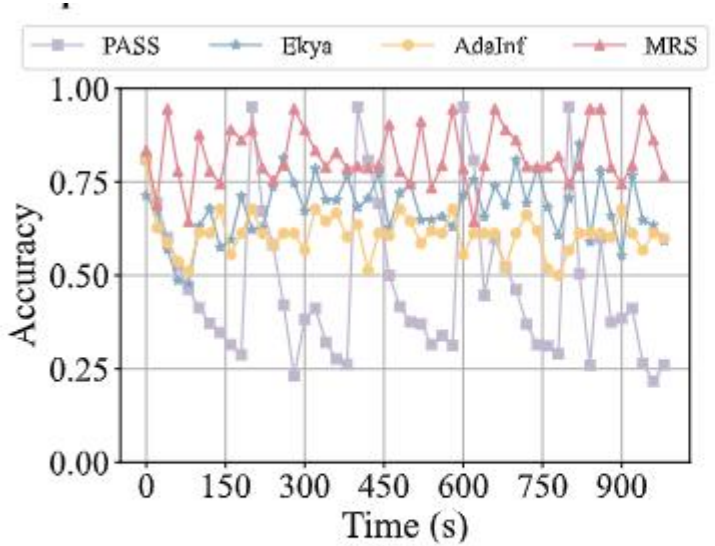


Fig. 8: Average inference accuracy over time.

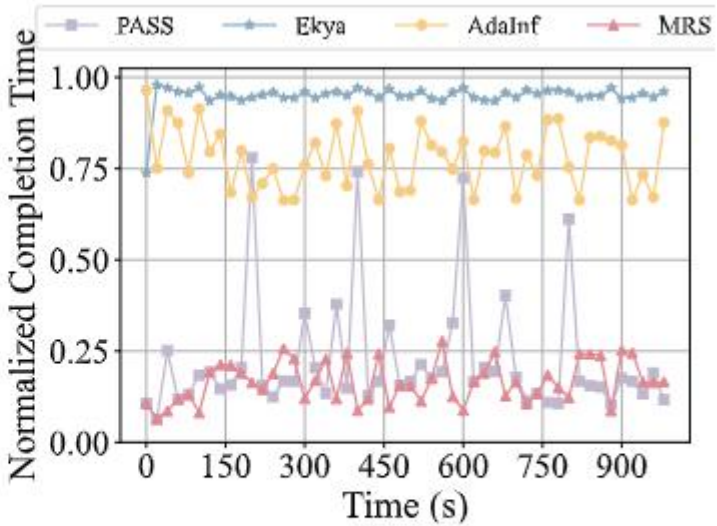


Fig. 9: Normalized completion time over time.

实验结果

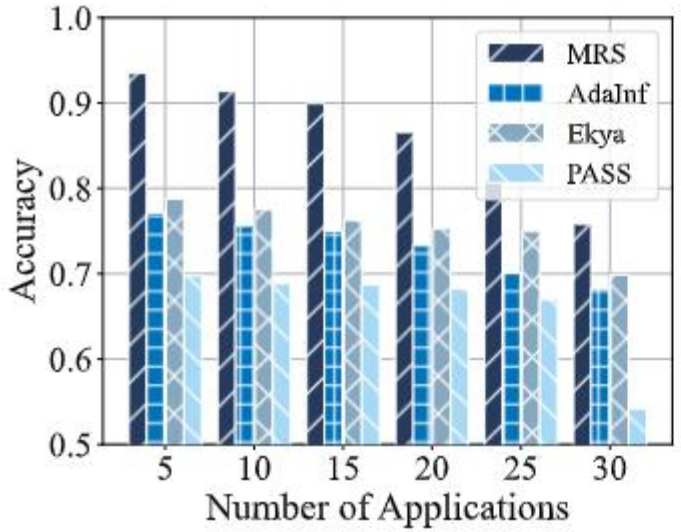


Fig. 10: Average inference accuracy of different number of applications.

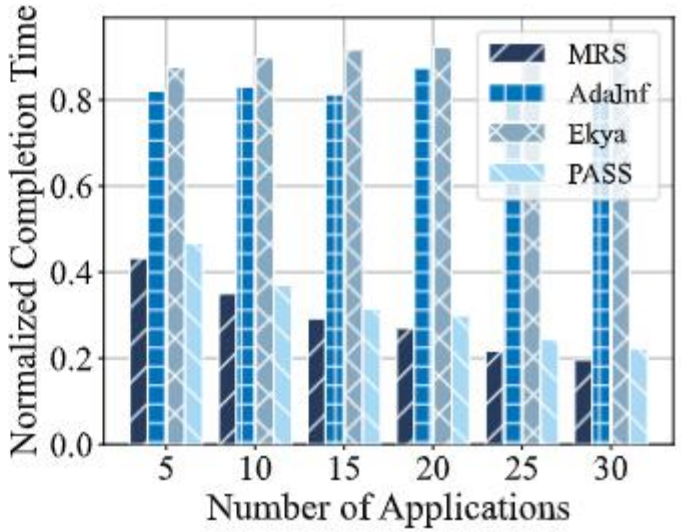


Fig. 11: Normalized completion time of different number of applications.

实验结果

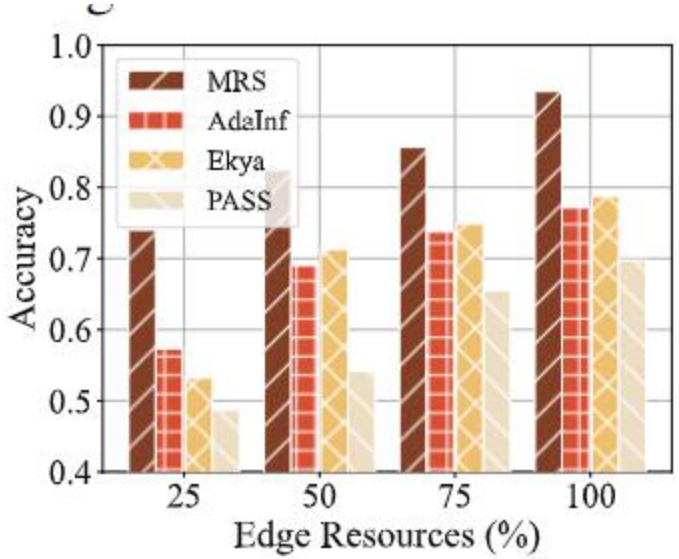


Fig. 12: Average inference accuracy of different edge resources.

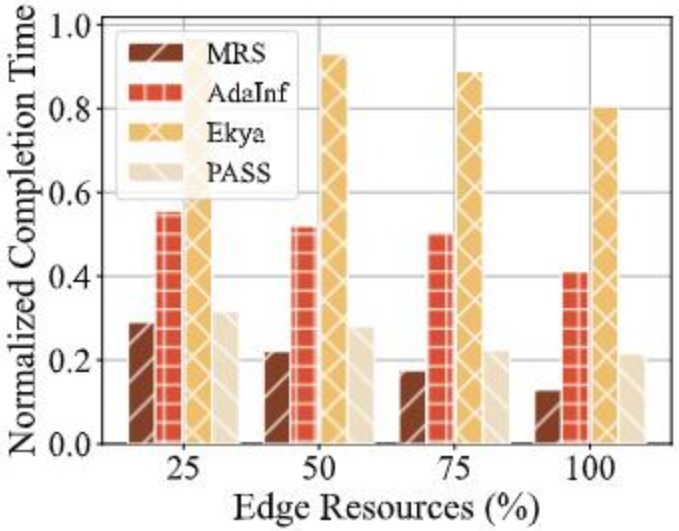


Fig. 13: Normalized completion time of different edge resources.

实验结果

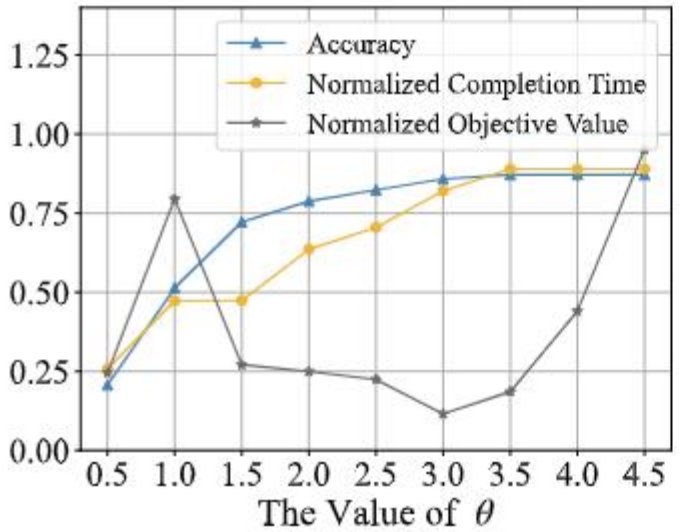


Fig. 14: Accuracy, completion time and objective value of different value of θ.

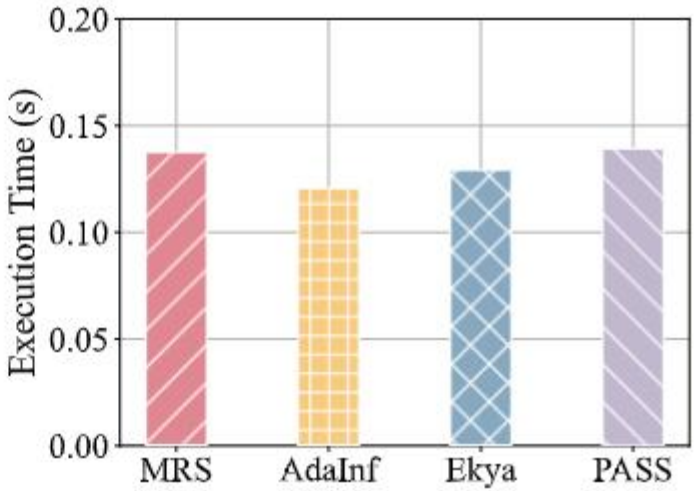


Fig. 15: Execution time of different algorithms.

汇报完毕
恳请指正