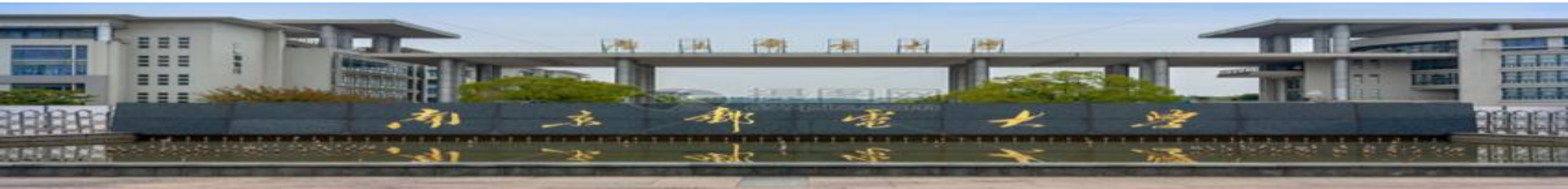
南京郵電大學 Nanjing University of Posts and Telecommunications

在6G云边缘协作计算中使用分布式深度强化学习的主动缓存

姓　　名：朱运涛

日　　期：2025.4.17

南京邮电大学计算机学院



摘要

为了解决多目标联合优化问题，即最大化边缘命中率，同时最小化内容访问等待时间和流量成本。重点研究了基于分布式深度强化学习（DRL）的主动缓存方法，包括内容预测和内容决策。

由于针对当前时间段内用户请求的先验信息难以获取的问题，提出了一种基于时间卷积网络（TCN）和注意力模型的时间卷积序列网络（TCSN）来提高内容预测的准确性。

分布式深度Q网络（DDQN）根据内容预测值，构建收益分布模型，优化内容决策策略。生成式对抗网络（Generative Adversarial Network，GAN）采用分布式方式，强调学习数据分布并生成跨多个节点的有说服力的数据。优先体验回放（PER）有助于从最有效的样本中学习。(多变量融合算法PG-DDQN)

背景

6G云边缘协作计算场景中，多个节点可以灵活、动态地协作，用户数据以去中心化的方式存储在边缘节点上。

缓存在本地 ⟶ 立即从指定的边缘节点下载

缓存在相邻的边缘节点 ⟶ 相关联的边缘节点获取它们并将它们发送到本地边缘节点

内容不在边缘 ⟶ 从云中获取内容。

大量基于强化学习的主动缓存研究表明，交通成本、内容访问延迟和缓存命中率是缓存性能的重要指标。

存在的问题：

1.很少研究同时联合优化这三个指标，要么是优化交通成本、内容访问延迟，要么优化缓存命中率。

2.这些研究做出了不切实际的假设：用户请求或内容流行度是事先已知的。（实际上是未知的）

3.DDQN，优化了现有的指标，但DDQN不能满足高效和稳定的多目标联合优化。

4.多代理缓存框架被用于集中式训练和分布式推理，但是，训练不同的神经网络可能需要有效的样本。

创新点和贡献

探索基于分布式DRL的方法，联合设计智能内容预测、内容访问和内容替换策略，以实现主动缓存的多目标联合优化。

（1）提出了一种基于TCN和注意力模型的新方法：时间卷积序列网络（TCSN），提高了用户请求的预测精度。

（2）构建了一种基于GAN和PER的DDQN方法，称为PG-DDQN，在提高训练效率的同时减少了内容访问延迟和流量开销。

（3）面对复杂的环境，采用多智能体学习结构，以集中式训练和分布式推理的方式有效地学习基于DRL的方法。

（4）同时优化三个指标：最大化边缘命中率，同时最小化内容访问延迟和流量成本。

related work

1.分布式场景中的主动缓存
通过文章分析出可以联合使用多个目标，即内容传递和缓存替换的内容访问时延、流量成本和替换成本来衡量主动缓存的性能。
2.主动缓存解决方案DRL
提到LSTM、TCN、分层的联合内容预测、DQN、DDQN、DDQN-GAN这些用于提高主动缓存的性能的方法。并指出了缺陷：样本数量在每个代理的经验重放缓冲区中增长迅速，并且一个随机抽样的小批量样本导致模型训练的不确定性。（本文提出PER来解决该问题）

网络模型

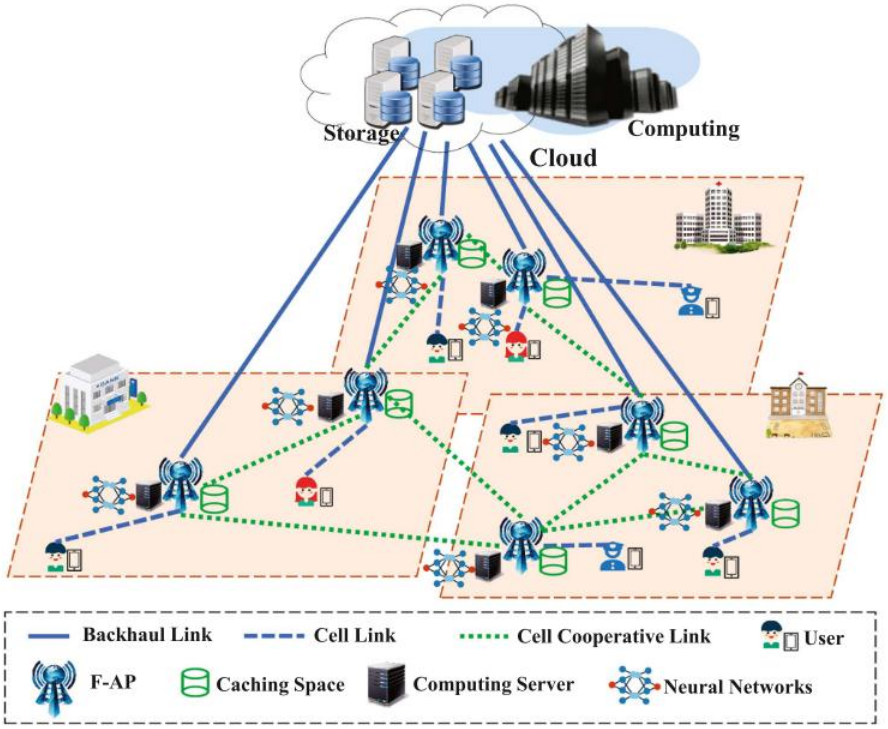


Fig. 1. Proactive Caching in 6G Cloud-Edge Collaboration Scenarios.

云(Y)、边缘节点(F-AP)

F-AP的集合 $\mathcal{L}=\{1,\ 2,\ \ldots,\ l,\ \ldots,\ L\}$

F-AP具有有限的存储空间，F-AP l可以通过每条链路分别从自身、其他F-AP和Y获得所请求的内容。

主动式高速缓存的操作在固定长度的时间内进行，被划分为离散的时段。

$\mathcal{T}=\{1,\ 2,\ \ldots,\ t,\ldots,\ T\}$

每个l中所请求内容的索引集:

$\mathcal{C}_{req}=\{1,\ 2,\ldots,\ c,\ \ldots,\ C\}$

每个l中缓存内容的索引集:

$\mathcal{F}_{cache}=\{1,\ 2,\ \ldots,\ f,\ldots,\ F\}$

请求内容c的大小: e_c

F-AP l的存储容量: W_l

“流量成本”主要包括每个链路上的数据流成本和时间t时每个边缘节点处的替换成本。

内容命中模型

$$\hbar_l(t) = \frac{\mathcal{C}_{req}(t) \cap \mathcal{F}_{cache}(t)}{\mathcal{C}_{req}(t)}.$$

在时间段t,
请求并且被缓存的内容的数量与请求的内容的数量比即FAP l的命中率。

内容访问模型

可以选择三条路径来通过在t的开始处的内容访问来获得所请求的内容

（1）本地通信：如果所请求的内容被缓存在本地，则缓存的内容c可以由F-AP l直接发送给移动的用户。

（2）邻居通信：如果所请求的内容未被缓存在本地，则F-AP l通过蜂窝协作链路从邻居边缘节点获取内容，该蜂窝协作链路具有高带宽和低递送延迟。

（3）云通信：如果所请求的内容既不在本地也不在邻居中，则F-AP l经由回程链路从Y获得内容，这可能导致高接入延迟和业务成本。

每个F-AP的总访问延迟：

$$h_l(t) = \sum_{c \in \mathcal{C}_{req}} \sum_{u \in \{\mathcal{L}', Y\}} \xi_{c,l}(t) \times d_{c,l,u}(t) \times n_{l,u}, \qquad (2)$$

在t的开始处l中的内容c的请求数量

在t的开始处对l中的内容c的请求是否由其他服务器u服务

L和服务器u之间所请求的内容的传输延迟

每个F-AP的总流量成本：

$$q_l(t) = \sum_{c \in \mathcal{C}_{req}} \sum_{u \in \{\mathcal{L}', Y\}} \xi_{c,l}(t) \times d_{c,l,u}(t) \times p_{l,u}, \qquad (3)$$

l和服务器u之间的流量成本

内容替换模型

如果请求缓存的内容与当前缓存的内容完全不同，则每个F-AP需要使用来自服务器u或Y的新内容替换当前内容，产生替换成本。计算每个F-AP在t结束时的替换成本如下：

内容c在t开始处是否被缓存在l中

$$b_l(t) = \sum_{c \in \mathcal{C}_{req}} \sum_{u \in \{\mathcal{L}', Y\}} \max\{y_{c,l}(t) - y_{c,l}(t-1), 0\} \times d_{c,l,u}(t) \times p_{l,u}, \quad (4)$$

优化目标

流量成本

$$\min_{d_{c,l,u}} \frac{1}{T}\frac{1}{L}\sum_{t\in\mathcal{T}}\sum_{l\in\mathcal{L}} -\gamma\hbar_l(t) + \alpha h_l(t) + \beta q_l(t) + \beta b_l(t) \tag{5}$$

$$\text{s.t.} \quad \alpha + \beta + \gamma = 1, \tag{6}$$

$$d_{c,l,u}(t) \in \{0,\ 1\}, \quad \forall c,\ \forall l,\ \forall u \tag{7}$$

$$d_{c,l,u}(t) \le d_{c,u,u}(t), \quad \forall c,\ \forall l,\ \forall u \tag{8}$$

服务器u中缓存了c，l才可能从u中接收到c

$$d_{c,Y,Y}(t) = 1, \quad \forall c \tag{9}$$

云服务器可以为所有内容提供服务

$$\sum_{u\in\{\mathcal{L}',Y\}} d_{c,l,u} = 1, \quad \forall c,\ \forall l \tag{10}$$

一个内容c只能由一个服务器提供到l

$$\sum_{l\in\mathcal{L},c\in\mathcal{C}} d_{c,l,l} \times\ e_c \le W_l, \quad \forall l \tag{11}$$

l中存储的缓存内容不能超过其存储容量

α、β和γ分别是用于调整内容命中率、内容访问延迟和流量成本之间的权重的权重因子

问题陈述

为了最大化内容命中率指标，需要准确地知道下一个时间段的请求内容，从而获得准确的内容决策。此外，在内容决策过程中，适当的路径是重点，以最大限度地减少内容访问延迟和流量成本。此外，在每个边缘节点的环境是潜在的影响，使得它很难优化的主动缓存策略只考虑本地信息。

分布式体系结构和DRL的成功为问题（5）提供了一个可行而有效的解决方案。移动用户丰富的历史请求信息为内容预测和内容决策提供了宝贵的数据资源。

基于分布式学习的机制不仅捕捉本地节点和其他边缘节点的动态，而且还能够提供从用户请求的预测到内容决策的解决方案。

TCSN

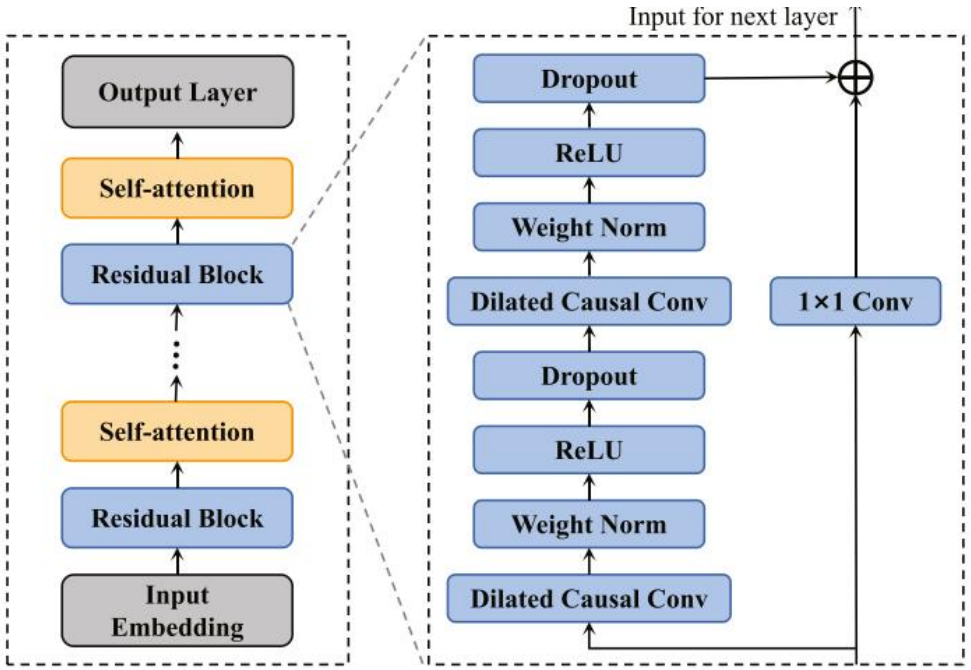


Fig. 2. The TCSN-based Prediction Model.

输入嵌入层，时间卷积层，注意层和输出层

TCSN

输入嵌入层：量化每个内容的信息。神经网络不能直接处理原始分类数据，因此所有输入和输出变量都转换为数值类型，以便模型可读。采用了二进制编码来实现这一操作，相较与独热编码，可以使用更少的特征来表示数据。

时间卷积层：接收到嵌入特征之后，部署时间卷积层以提取高级特征。其包括三个模块，因果卷积、扩张卷积和残差连接。

1）因果卷积：t处的输出仅与来自t的元素或来自前一层的更早时间步进行卷积。

2）扩张卷积：增大感受野，从而捕获长的记忆。

3）残差连接：包括两个分支。第一个是通过一系列堆叠的层来转换输入x，而第二个是输入x的快捷连接。

注意层：考虑到预测输出与各种时间步长之间的相关性，采用了自注意机制。自注意力的好处在于使网络只关注关键输入时间步，访问所有时间步而不需要深层结构。

输出层：这一层使用最终的注意力向量来进行内容预测。

多代理PG-DDQN

为了提高超参数变化的鲁棒性和降低环境噪声，DDQN直接基于收益的分布而不是期望，即：Qπ。

状态空间：每个代理的状态包括缓存情况和t时的预测内容。$s_l(t) = \{\phi_{c,l}(t),\ y_{l,c}(t)\}$

其中前者表示预测的内容，后者表示缓存情况。

行动空间：在t结束时，每个智能体观察来自环境的输入，并考虑其当前策略制定下一步行动。$a_l(t) = \{y_{l,c}(t),\ o\}$ 前者表示预测内容的替换决策，后者表示替换对象。

奖励：在采取自己的行动后，每个智能体都会收到奖励rl（t）。将奖励定义为加权和，即，缓存命中率、内容访问延迟和内容决策的流量成本之和。

算法由两部分组成，即，一个是部署在每个边缘节点上的Actor，用于做出内容替换的本地决策;另一个是部署在云中的Learner，用于响应以更新Actor，近似最优策略。对于每个F-AP，周期性地从学习器复制生成器网络G以生成固定数量的样本。

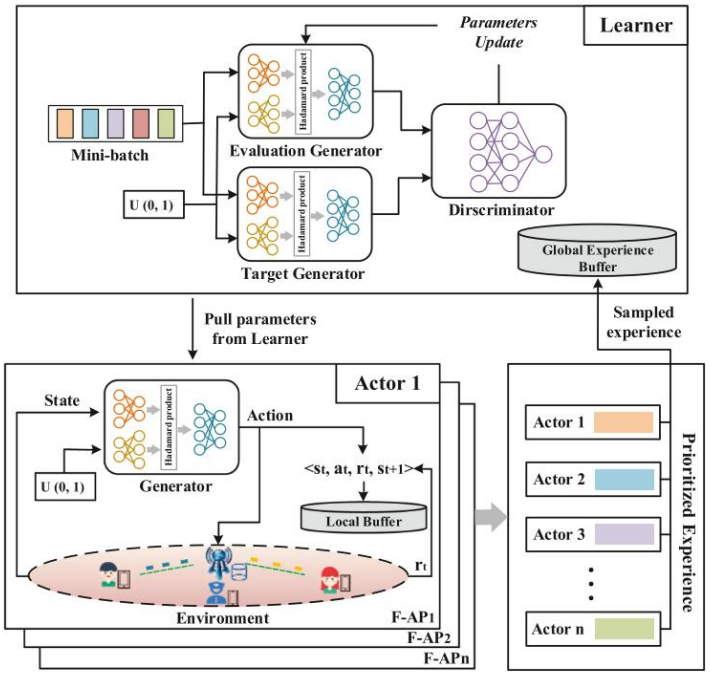


Fig. 3. The PG-DDQN Algorithm.

多代理PG-DDQN

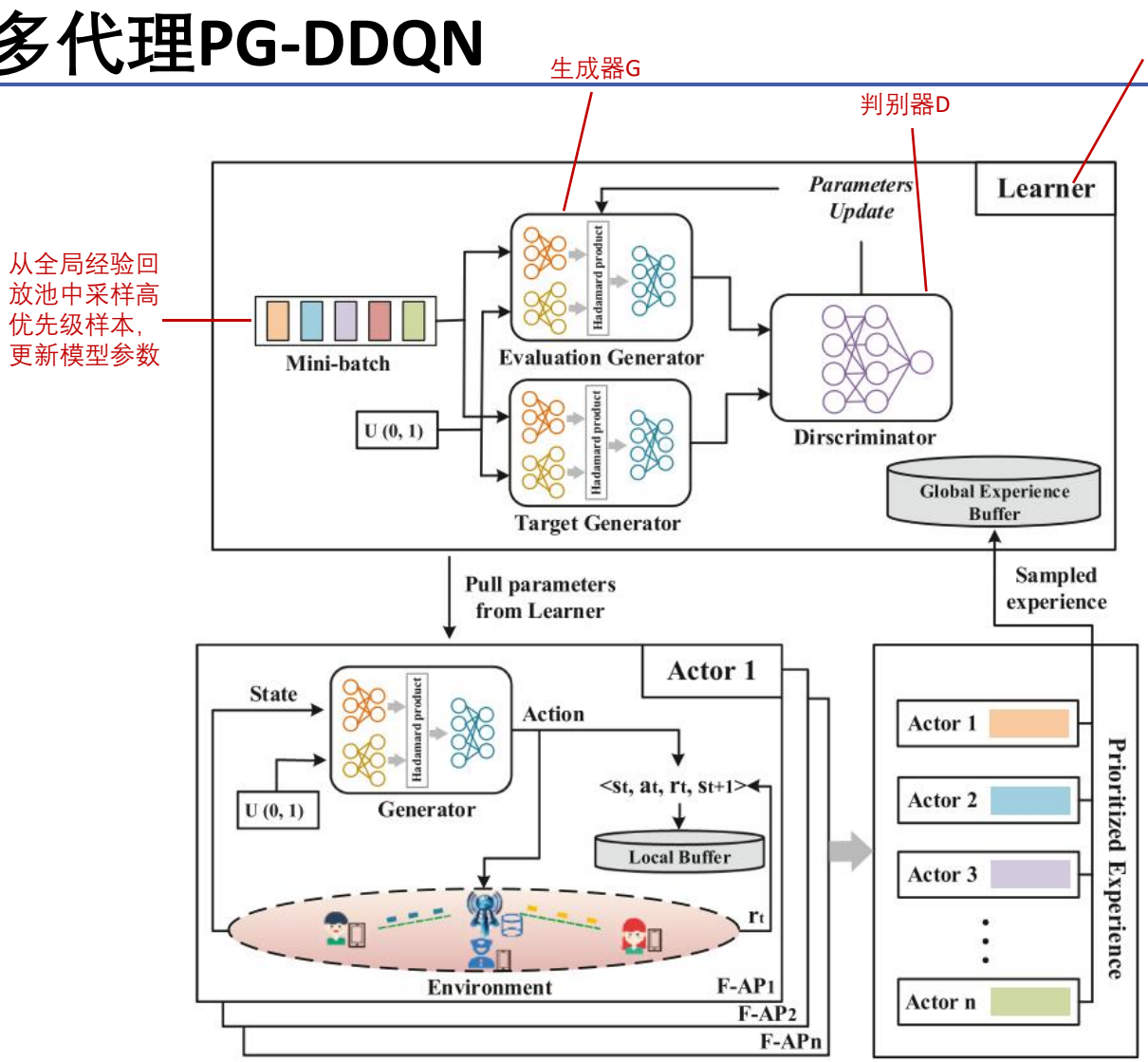


Fig. 3. The PG-DDQN Algorithm.

Learner：
负责全局参数更新，包括生成器和判别器。
从全局经验回放池中采样高优先级样本，更新模型参数。
通过梯度下降优化损失函数。

Actor：
每个边缘节点（F-AP）部署一个Actor，负责与环境交互。
根据当前策略生成动作（如内容替换决策），并将经验（状态、动作、奖励、下一状态）存入本地经验回放池。
定期将本地经验（按优先级排序）上传至全局经验池。

多代理PG-DDQN

期望动作值Qπ可以重写为：$Q^{\pi}(s,\ a)=\mathbb{E}\left[Z^{\pi}(s,\ a)\right].$

Zπ（s，a）：在状态s下采取动作a并遵循策略π时的回报分布（而非单一期望值）。

传统Q学习仅优化期望回报，而分布强化学习通过建模回报的完整分布，捕捉环境的不确定性和多模态特性。

折扣因子，平衡当前与未来回报

$$\mathcal{T}^{*}Z(s,\ a)\overset{D}{=}R(s,\ a)+\gamma Z\left(s',\ \underset{a'\in\mathcal{A}}{\arg\max}\,\mathbb{E}\left[Z\left(s',\ a'\right)\right]\right), \tag{16}$$

及时奖励，确定值

下一状态

选择下一状态s’中期望最大的动作a’

分布贝尔曼最优算子T

将即时奖励与折扣后的未来分布结合，迭代更新动作值分布Z

多代理PG-DDQN

分布强化学习的优化目标，即最小化当前分布Z与通过贝尔曼算子更新后的分布T＊Z之间的最大统计距离。

$$\sup_{s,a} \operatorname{dist}\left(\mathcal{T}^{*} Z(s,\ a),\ Z(s,\ a)\right), \qquad (17)$$

取最大值

随着经验回放缓冲区占用率的增加，随机选择一小批样本可能会在模型训练期间遇到性能下降。为了提高模型的收敛性，PER作为离线DRL的主流技术被集成到多智能体DRL算法中。在每个决策时期，我们的目标是最小化Q与其Bellman目标之间的点距离。因此，我们定义优先级值，该优先级值由当前分布和目标分布之间的偏差来度量。

分布距离度量

$$\delta = dist\left(Z^{\pi}(s,\ a),\ R(s,\ a) - \gamma Z^{\pi}\left(s',\ a'\right)\right). \qquad (18)$$

当前状态s和动作值a的分布（由生成器估计）

Actor算法

Algorithm 1: Actor (in Each F-AP).

Initialize the latest network parameters θ_G and $\theta_{\hat{G}}$ from Learner;

Initialize a local replay buffer $\mathcal{B} \leftarrow \varnothing$;

for $t = 1, 2, \ldots$ do

  Agent observes s;

  Get samples $\tau \sim U(0, 1)$;

  Calculate $Q(s, a) = \frac{1}{N} \sum G^{(a)}(s, \tau), \forall a \in \mathcal{A}$;

  Perform $a^* \leftarrow \arg\max_a Q(s, a), \forall a \in \mathcal{A}$;

  The agent observes s' and receives the reward r;

  Stores transition (s, a^*, r, s') in $\mathcal{B}$;

  if local buffer size > B then

    Get batch of transitions from buffered data;

    Calculate priorities for experience by distribution distance;

    Push prioritized experience to centralized replay buffer;

  end

  Update G and $\hat{G}$ with weight θ_G and $\theta_{\hat{G}}$.

  $s = s'$;

end

初始化生成器参数

Agent（即当前F-AP）从环境中获取当前状态s

从均匀分布中采样分位数τ

对每个动作a,通过生成器 G 生成 N 个分位数值,取平均作为Q(s,a).赖估计动作值分布

选择具有最高平均动作值的动作a*

存满时

使用公式(18)计算每条经验的优先级

将带优先级的经验推送到全局经验回放池E，以供Learner集中训练使用

Learner算法

Algorithm 2: Learner (in Cloud).

Set G and D. The weights are random and are denoted by θ_G and θ_D, respectively. Initialize the gradient coefficient of penalty λ, the discount factor γ, and the batch size m;

Initialize one target generator $\hat{G}$, whose weight is denoted by $\boldsymbol{\theta}_{\hat{G}} \leftarrow \boldsymbol{\theta}_G$; Initialize a centralized replay buffer named E.

for $t = 1,\ 2,\ \ldots$ do

  Sample a batch of prioritized transitions from E and a batch of $\{\tau\}^m \sim U(0,\ 1)$;

  Get the target action-value by $y_i = r_i + \gamma \hat{G}^{(a_i^*)}\left(s_i',\ \tau_i\right)$;

  Set $\hat{x}_i = \varepsilon_i y_i + (1 - \varepsilon_i)\, G^{(a_i)}\left(s_i,\ \tau_i\right)$;

  Update the weight θ_D using gradient descent algorithm to $\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_i$, where $\mathcal{L}_i = D\left(G^{(a_i)}\left(s_i,\ \tau_i\right)\right) - D\left(y_i\right) + \lambda\left(\left\|\nabla_{\hat{x}_i} D\left(\hat{x}_i\right)\right\|_2 - 1\right)^2$;

  Update the weights θ_D by leveraging gradient descent algorithm to $-\frac{1}{m}\sum_{i=1}^{m} D\left(G^{(a_i)}\left(s_i,\ \tau_i\right)\right)$;

  if E is full then

    Remove oldest experience from replay buffer E;

  end

  Clone network G to the target network $\hat{G}$ via resetting $\boldsymbol{\theta}_{\hat{G}} = \boldsymbol{\theta}_G$;

end

初始化生成器G、判别器D、梯度惩罚系数λ、折扣因子γ、批次大小m

从全局池E中按优先级采样一批经验，m个分位数

生成下一状态的目标动作值

将生成器G的输出与目标值yi混合，得到xi,用于计算梯度惩罚

损失函数计算，包含判别器对生成器输出G(si,τi) 和目标值yi的评分差异以及梯度惩罚项。防止判别器过度自信。

更新生成器G的损失函数，通过梯度下降最小化该损失函数。驱动生成器G欺骗判别器D，使其评分最大化

目标生成器网络，输入为下一状态s(t+1)和量化参数τ(t)，输出动作值的分布。

$$\varsigma(t) = r(t) + \gamma \min_{a(t)} \hat{G}(s(t+1),\, \tau(t)). \qquad (19)$$

期望动作值

表示判别器对生成器G生成的动作值样本的评分期望

判别器对真实目标分布样本的评分期望

梯度惩罚项，用于稳定训练

$$L_D = \mathop{\mathbb{E}}_{\tau\sim U(0,1)} \left[D(G^{(a(t))}(s(t),\, \tau(t)) \right] - \mathbb{E}[D(\varsigma(t))] + \psi(\lambda),$$

训练判别器D以区分生成的动作值分布与真实的目标分布。

$$L_G = -\mathop{\mathbb{E}}_{\tau\sim U(0,1)} \left[D(G^{(a(t))}(s(t),\, \tau(t)) \right],$$

训练生成器G以生成更接近真实目标分布的动作值样本。最大化判别器对生成样本的评分，迫使生成器欺骗判别器，使其生成的样本被判别器误认为来自真实分布。

采用分布式强化学习框架以适应动态和复杂的边缘环境。
结合用户行为预测（如使用TCSN或LSTM）来预加载资源，减少延迟。
根据用户历史请求和任务依赖（DAG）动态调整缓存内容。
多目标优化，延迟、能耗和缓存命中率的平衡。

汇报完毕，感谢大家的观看