Federated Reinforcement Learning

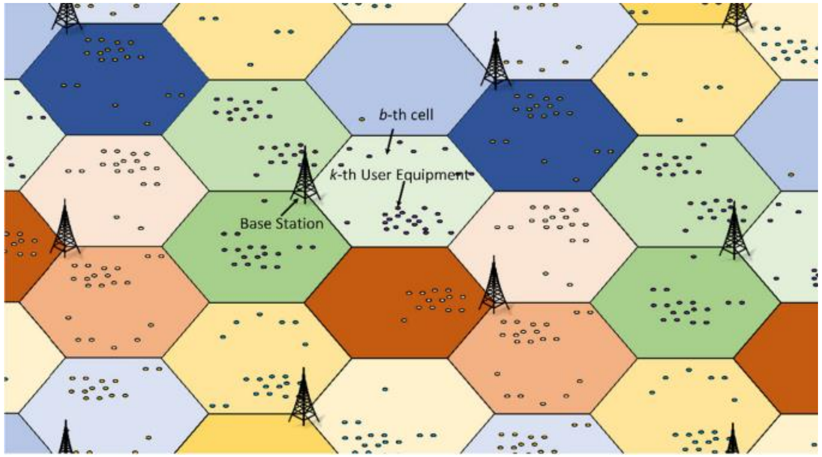
Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, Yufeng Lin

Sun Yat-Sen University & **Webank**

Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, Yufeng Lin. Federated Reinforcement Learning. https://arxiv.org/abs/1901.08277. 2019

Multi-agent reinforcement learning

- Global state or sharing state
- Global reward or sharing reward



Configuration of base stations

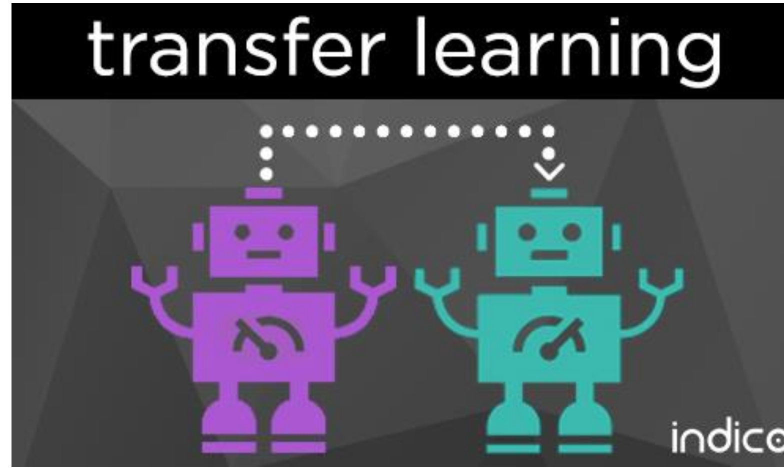
Game!

Logistics

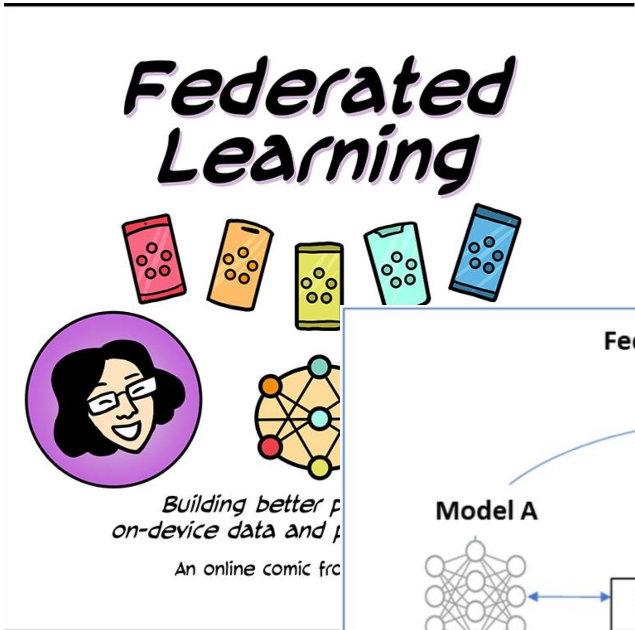
Transfer data instance?

Transfer feature space?

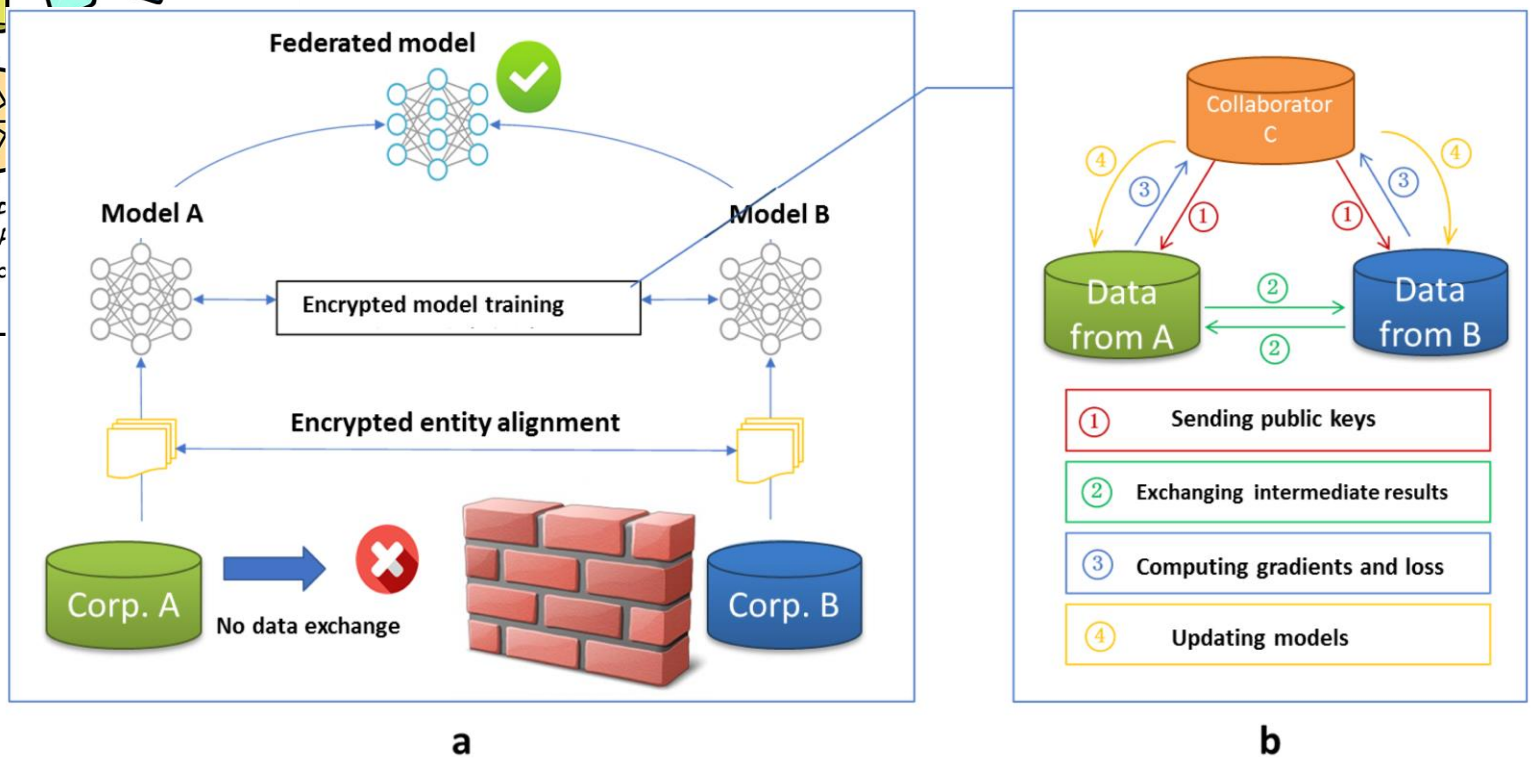
Transfer models?



- Instance is private!
- Feature space is private!
- Model is private!



Google AI

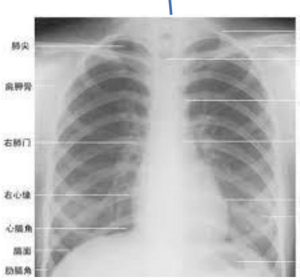


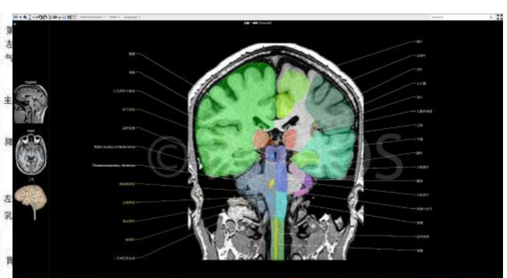
Federated Machine Learning: Concept and Applications. Q Yang, Y Liu, T Chen, Y Tong. ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2), 12, 2019

Why does RL need to be federated?

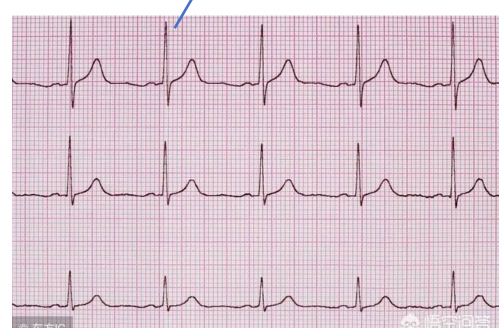
State, action, reward

State, action





Surgery



Pills

In the manufacturing industry, producing products may involve various factories which produce different components of the products. Factories’ decision policies are private and will not be shared with each other. On the other hand, building individual decision policies of high-quality on their own is often difficult due to their limited businesses and lack of rewards

Problem setting

MDP:

$\langle S, A, T, r \rangle$

Output of MDP:

policies π

	Agent α	Agent β
Input:	$\{\langle s_\alpha, a_\alpha, s'_\alpha, r_\alpha \rangle\}$	$\{\langle s_\beta, a_\beta \rangle\}$
Output:	policies π_α^*	Policies π_β^*

A1: The feature spaces of states s_α and s_β are *different* between agents α and β.

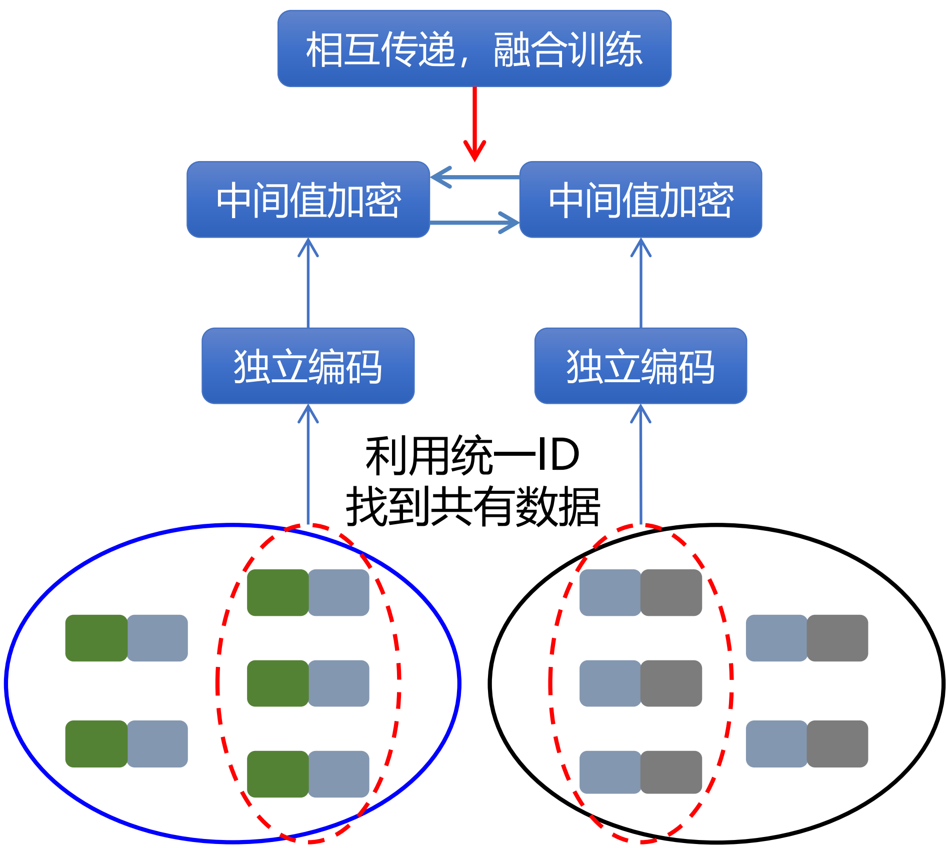
A2: D_α and D_β cannot be shared directly between α and β

A3: The output of functions Q_α and Q_β *can* be shared with each other

FRL算法模型

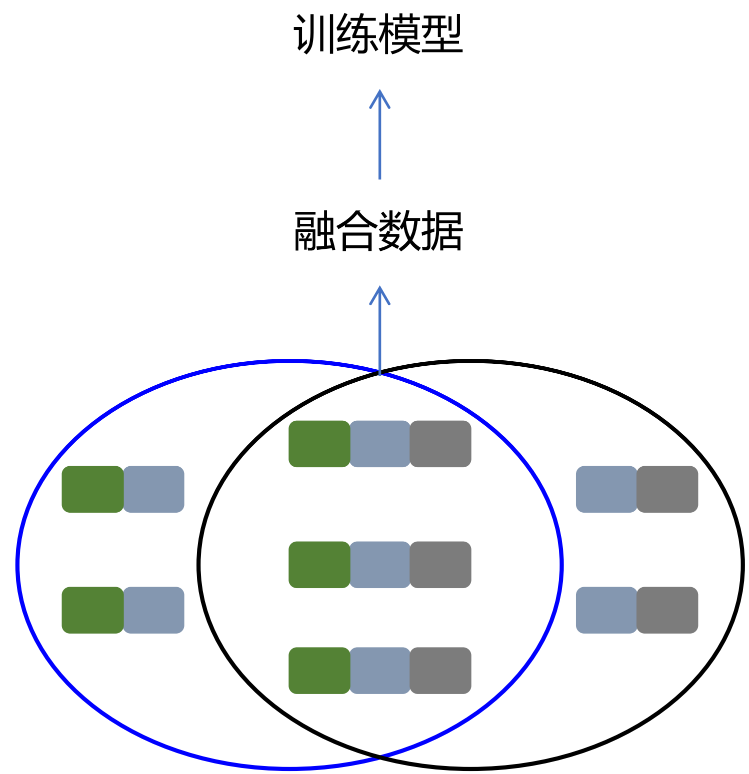
如何利用分散的数据和特征来训练模型？

FRL方法



传统方法

缺点：机密数据无法融合，方法失效



The Q learning model:

两个网络之间传递
各自计算的加密Q值

基础Q网络

$$Q_f^{\alpha}(\cdot, C_\beta; \theta_\alpha, \theta_f) = MLP([\hat{Q}_\alpha(\cdot; \theta_\alpha); \boxed{C_\beta}]; \theta_f)$$

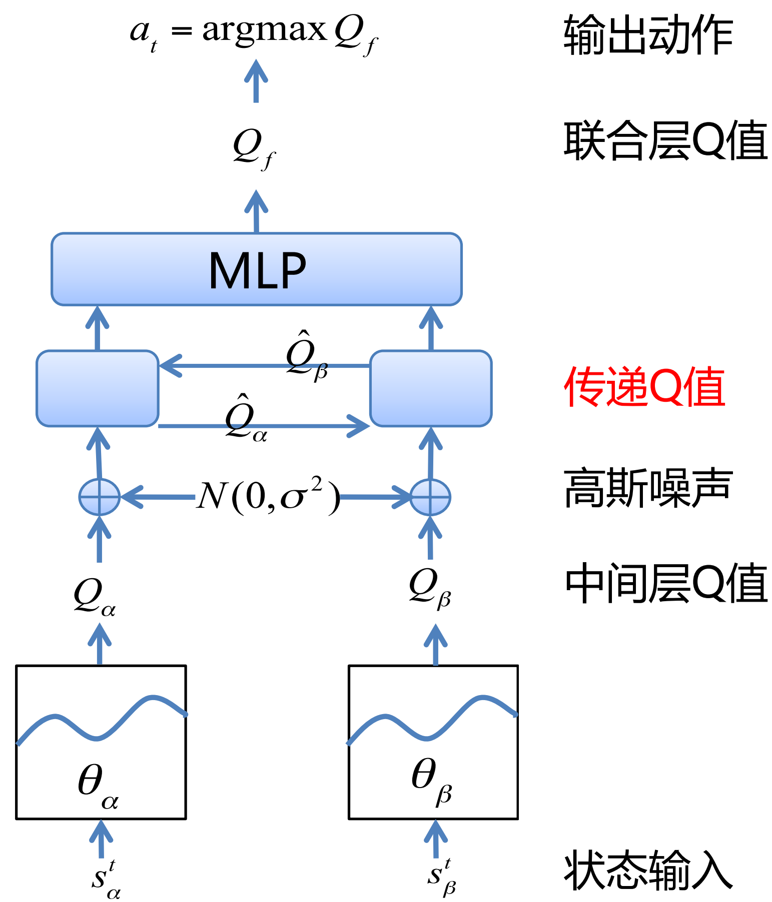
$$Q_f^{\beta}(\cdot, C_\alpha; \theta_\beta, \theta_f) = MLP([\hat{Q}_\beta(\cdot; \theta_\beta); \boxed{C_\alpha}]; \theta_f)$$

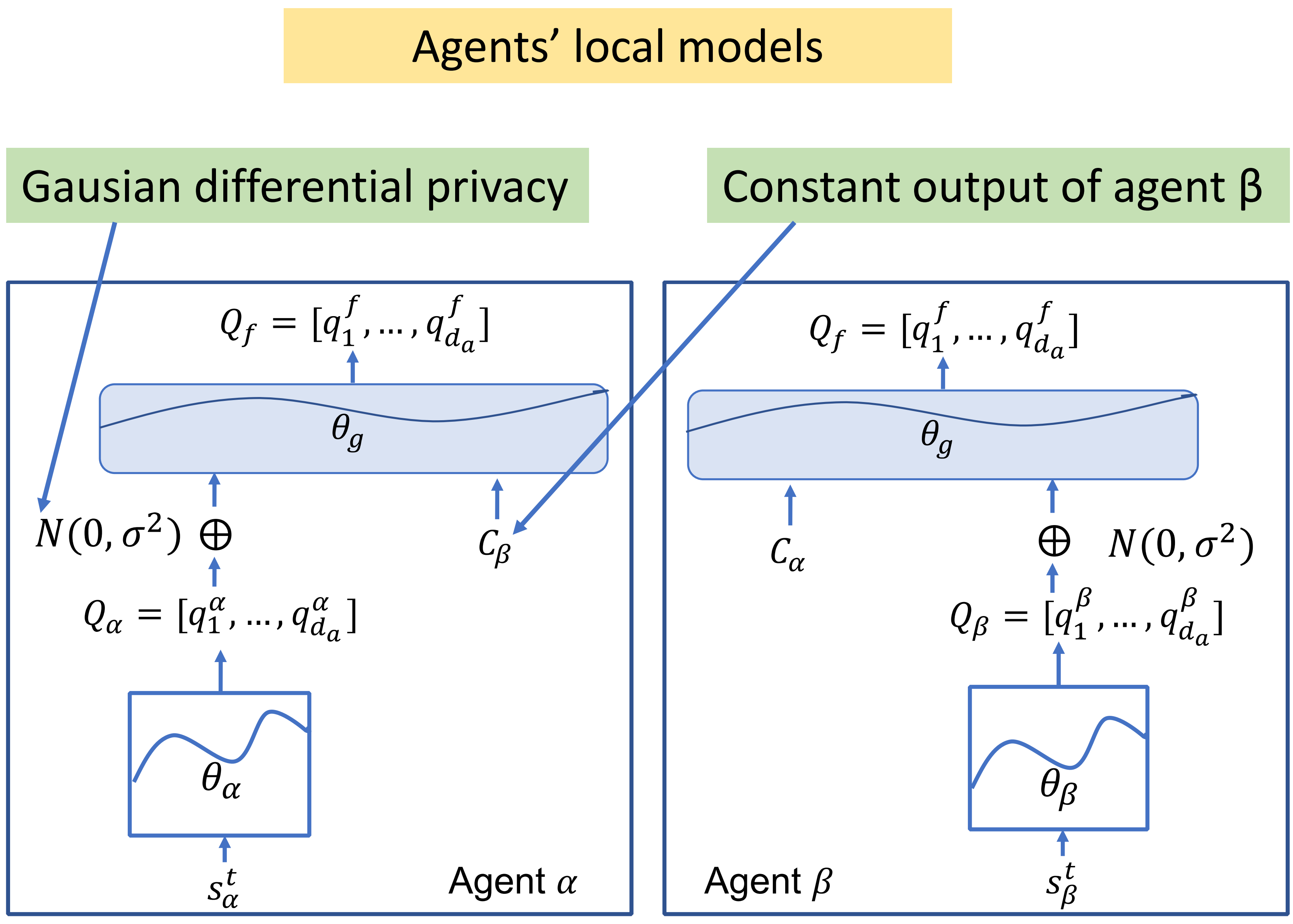
损失函数

$$L_\alpha(\theta_\alpha, \theta_f) = \mathbb{E}\{(y^i - Q_f^{\alpha}(s_\alpha^i, a^i, C_\beta; \theta_\alpha, \theta_f))^2\}$$

$$L_\beta(\theta_\beta, \theta_f) = \mathbb{E}\{(y^i - Q_f^{\beta}(s_\beta^i, a^i, C_\alpha; \theta_\beta, \theta_f))^2\}$$

$$y^i = r^i + \gamma \max_a Q_f^{\alpha}(s_\alpha^i, a, C_\beta; \theta_\alpha, \theta_f)$$



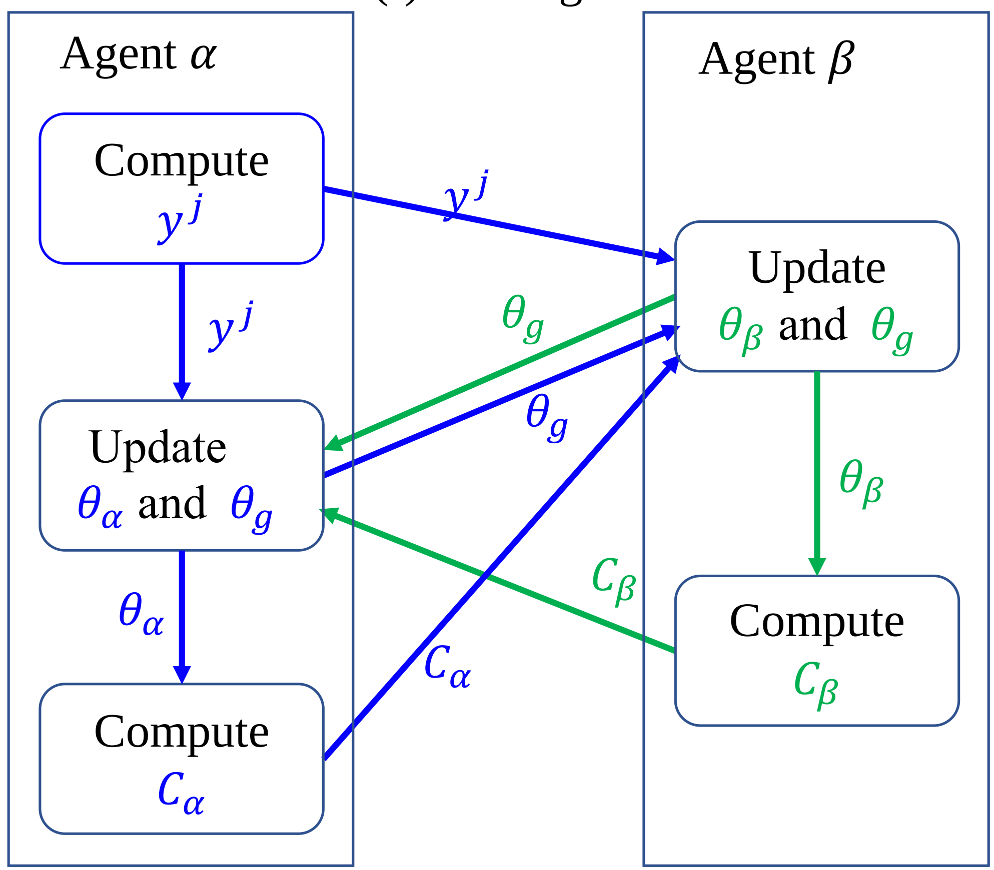


Agents' local models
Gausian differential privacy
Constant output of agent β
$Q_f = [q_1^f, \ldots, q_{d_a}^f]$
θ_g
$N(0, \sigma^2)$ $\oplus$
C_β
$Q_\alpha = [q_1^\alpha, \ldots, q_{d_a}^\alpha]$
θ_α
s_α^t
Agent α
$Q_f = [q_1^f, \ldots, q_{d_a}^f]$
θ_g
C_α
$\oplus$ $N(0, \sigma^2)$
$Q_\beta = [q_1^\beta, \ldots, q_{d_a}^\beta]$
θ_β
s_β^t
Agent β

Training procedure

Interaction between two agents **sequentially** or **in parallel**

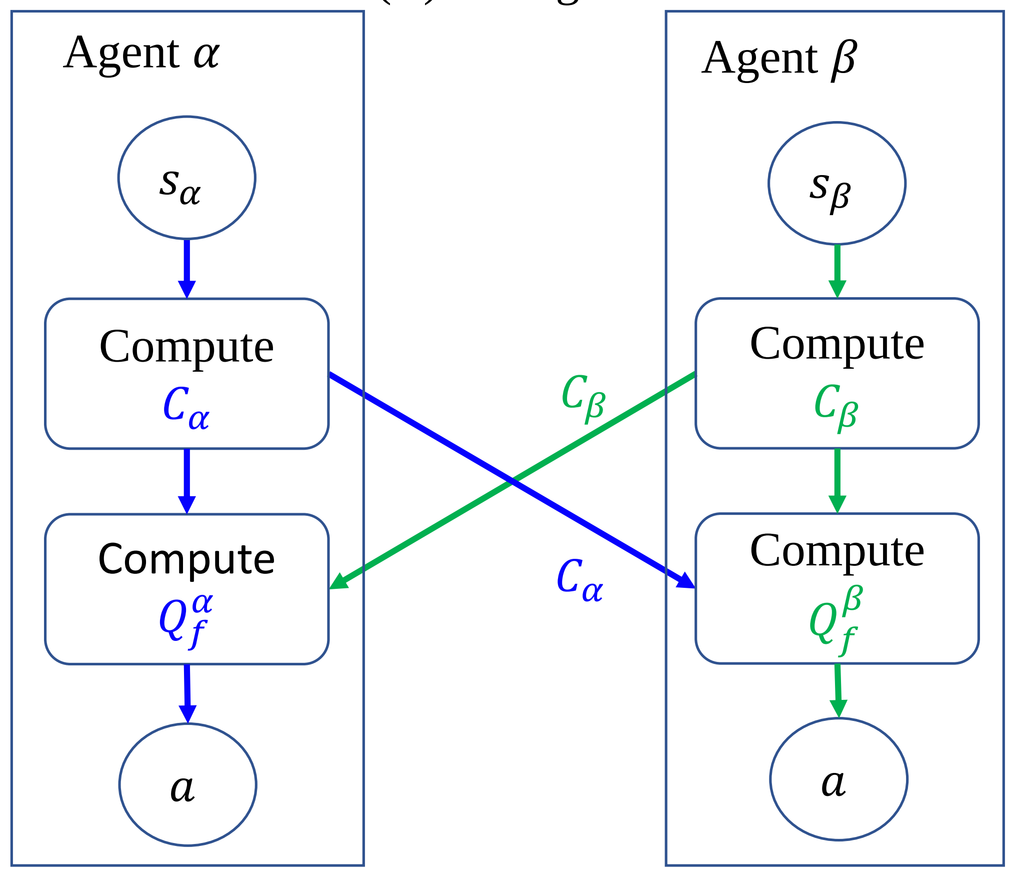
(I) training



Testing

- Need other agent's local output to make decisions
- Not necessarily making the same decisions

(II) testing



Algorithm 1: FRL-ALPHA

Input: state space S_α, action space A_α, rewards r
Output: θ_α,θ_g

Initialize Q_α, Q_f with random values for θ_α, θ_g
Initialize replay memory D_α
Call FRL-BETA.$Init()$
for episode = 1: M **do**
  repeat
    Observe s_α^t
    Call C_β=FRL-BETA.$ComputeQBeta()$
    Select action a^t with probability ϵ
    Otherwise
    $a^t = \arg\max_a Q_f^\alpha(s_\alpha^t, a, C_\beta; \theta_\alpha, \theta_g)$
    Execute action a^t, obtain reward r^t and state s^{t+1}
    Observe s_α^{t+1}, store $(s_\alpha^t, a^t, r^t, s_\alpha^{t+1})$ in D_α
    Sample $(s_\alpha^j, a^j, r^j, s_\alpha^{j+1})$ from D_α
    Call C_β=FRL-BETA.$ComputeQBeta(j)$
    $Y^j = r^j + \gamma \max_a Q_f^\alpha(s_\alpha^j, a, C_\beta; \theta_\alpha, \theta_g)$
    Update θ_α, θ_g according to Eq. (4), (6)
    $C_\alpha = \hat{Q}_\alpha(s_\alpha^j, a; \theta_\alpha)$
    Call
    θ_g=FRL-BETA.$UpdateQ(Y^j, j, C_\alpha, \theta_g)$
  until terminal t
end for

Algorithm 2: FRL-BETA

Input: state space S_β, action space A_β
Output: θ_β, θ_g

function $Init()$
  Initialize Q_β with random values for θ_β
  Initialize replay memory D_β
end function
function $ComputeQBeta()$
  Observe s_β
  Select $a_\beta \in A_\beta$ with probability ϵ
  Otherwise $a_\beta = \arg\max_{a_\beta} Q_\beta(s_\beta, a_\beta; \theta_\beta)$
  Store (s_β, a_β) in D_β
  Let $C_\beta = \hat{Q}_\beta(s_\beta, a; \theta_\beta)$
  return C_β
end function
function $ComputeQBeta(j)$
  Select (s_β, a_β) from D_β based on index j
  Let $C_\beta = \hat{Q}_\beta(s_\beta, a_\beta; \theta_\beta)$
  return C_β
end function
function $UpdateQ(Y^j, j, C_\alpha, \theta_g)$
  Select (o_β^j, a_β^j) from D_β based on index j
  Update θ_β, θ_g based on Eq. (5), (7)
  return θ_g
end function

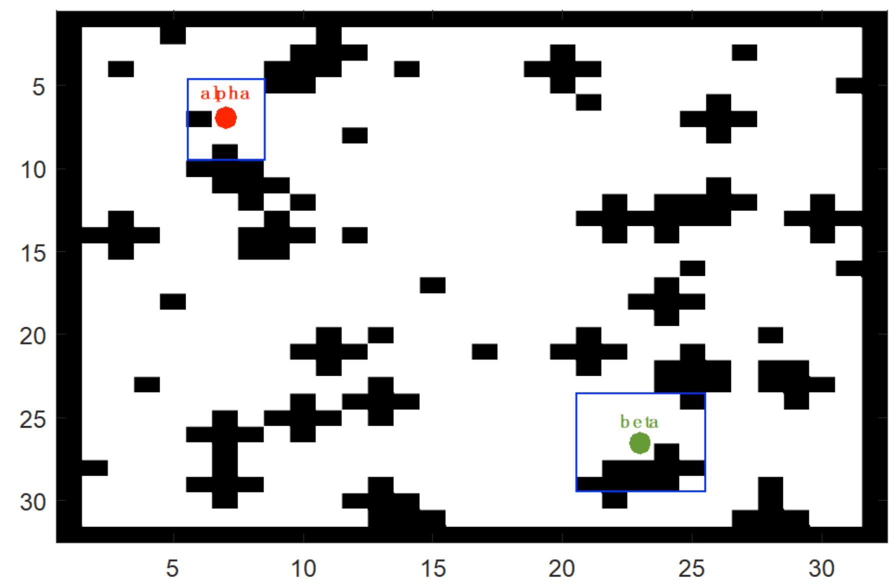
baselines

DQN-alpha	A deep Q-network trained with agent α's data only. It takes observations sα as input and outputs actions corresponding to s_α.
DQN-full	A deep Q-network trained by directly putting data together from both agent α and β, i.e., neglecting data privacies between agents α and β.
CNN-alpha	A convolutional neural network trained with agent α's data only, similar to DQN-alpha.
CNN-full	A convolutional neural network trained with all data of agent α and β put together directly, similar to DQN-full.

Domain: Grid-World

States: The domain is represented by a Ng × Ng

Actions: There are 4 actions for each agent, i.e. going towards 4 directions, denoted by {east, south, west, north}.



Rewards: The reward is composed of two parts, i.e., local reward r_l and global reward r_g

Dataset: We generated 8000 different maps (or matrices) for each size of 8 × 8, 16 × 16 and 32 × 32

Domain 2: Text2Action

数据集

- **CT**: 做菜教程 （Cooking Tutorials）
- **WHS**: 电脑指令 （Branavan et al. ACL 2009）
- **WHG**: 家庭和花园打理指南 （Malmaud et al., ACL 2014）

评价指标

- 累积奖励
- F1分数

基准算

任务

算法	
DepR	(Lindsay et al., I
LSTM	Hovy, ACL 2016)
EncD	H et al., AAAI 201
CMLP	
DQN-	的EASDRL模型
DQN-	的EASDRL模型

指令

输入的一个指令性文本

How to Make Egg Fried Rice?（如何做蛋炒饭？）

Cook the rice the day before, or use leftover rice in the refrigerator. The important thing to remember is not to heat up the rice, but keep it cold. In a bowl, add 1 tablespoon of oil to rice. Use a spoon or your hands to work the oil into the rice, evenly coating the rice. Transfer the rice to a colander and drain. Combine eggs and salt in a small bowl and gently whisk until blended. Heat 1 tablespoon oil in a wok. Add whisked eggs and cumin seeds to wok. Stir frequently, working the eggs to a scramble. Heat the remaining oil in the wok. If desired, you can recycle some of the oil that drained from the rice. Add the garlic and onion to the wok. Stir-fry together over high heat for about 5 minutes or until the onion looks transparent, but is not soft. Add the rice, eggs, soy sauce, chili sauce, vinegar, and celery. Mix together, continuing to stir-fry over high heat for 1-2 minutes while stirring frequently. Spoon onto a plate and serve.

动作
- 动作名称
- 动作参数

所有的动作序列输出

煮（饭）

- Cook (rice) → Keep (rice, cold)→ Add (oil) → Use (spoon) → Work (oil, rice) → ··· → Work (eggs) → Heat (oil) → ···
- Use (leftover rice) → Keep (rice, cold) → Add (oil) → Use (spoon) → Work (oil, rice) → ··· → Work (eggs) → Heat (oil) → ··· → Serve ()
- Use (leftover rice) → Keep (rice, cold) → Add (oil) → Use (hands) → Work (oil, rice) → ··· → Work (eggs) → Heat (oil) → ··· → Serve ()
- Use (leftover rice) → Keep (rice, cold) → Add (oil) → Use (hands) → Work (oil, rice) → ··· → Work (eggs) → Recycle (oil) → Heat (oil) → ··· → Serve ()
- ···

Domain 2: Text2Action

States: $s_\alpha \in R^{Nw \times K1}$ is real-valued matrix that describes the part-of-speech of words, and $s_\beta \in R^{Nw \times K2}$ is real-valued matrix that describes the embedding of words.

Actions: There are two actions for each agent, i.e., {select, neglect}

Rewards: The instant reward include a basic reward and an additional reward

Criteria:

$$precision = \frac{\#TotalRight}{\#TotalSelected}$$

$$recall = \frac{\#TotalRight}{\#TotalTruth}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

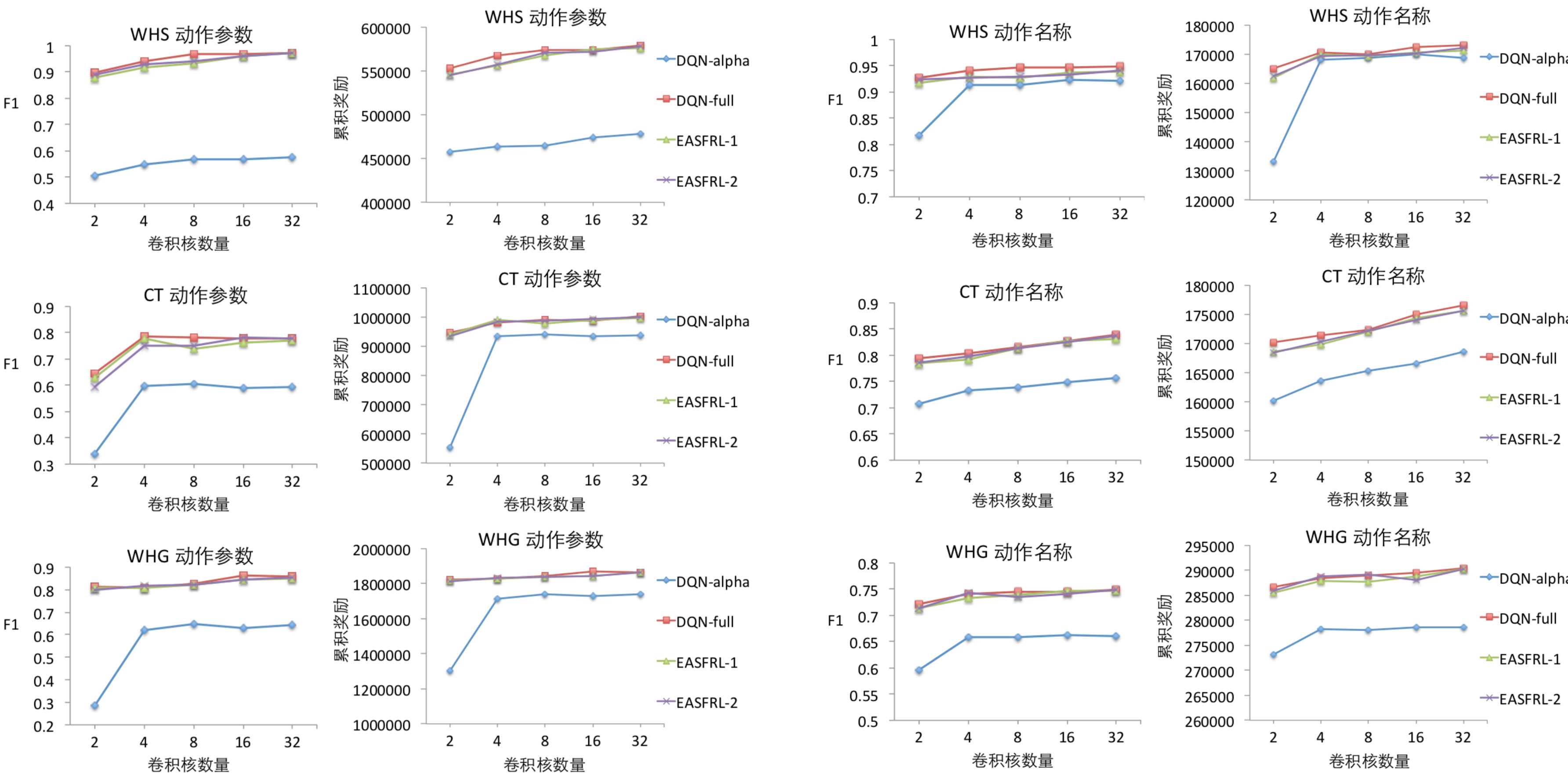
实验结果分析

评价标准	算法	动作名称			动作参数		
		WHS	CT	WHG	WHS	CT	WHG
F1 分数 (%)	DQN-alpha	92.11	75.64	66.37	54.13	59.46	61.09
	EASFRL-1	93.76	83.05	74.64	97.18	76.97	84.95
	EASFRL-2	**94.11**	**83.72**	**74.85**	**97.27**	**77.75**	**85.44**
	DQN-full	94.75	83.87	74.87	97.35	77.58	85.66
累积奖励 ($\times 10^4$)	DQN-alpha	16.87	16.86	27.86	47.78	93.82	174.01
	EASFRL-1	17.13	**17.56**	29.01	57.55	99.69	**186.13**
	EASFRL-2	**17.22**	**17.56**	**29.03**	**57.82**	**100.00**	186.12
	DQN-full	17.31	17.65	29.04	57.88	100.19	186.42

结论：

- FRL模型在所有实验中表现都明显高于只有部分数据的模型，说明将数据联合起来训练能明显提高模型的性能
- FRL模型性能非常接近于直接融合全部数据来训练的模型，说明FRL算法能够保护数据隐私的同时，保证模型的性能几乎不降低

探究性实验结果分析



结论：

- 模型简单时FRL相对部分数据训练的模型的优势更大，说明联邦学习能够取得较好的效果。
- 模型复杂度变化的时，FRL模型的性能都能很好的逼近融合全部数据训练的模型的性能。

Thank You