实验报告

总结

这个星期，主要研究模型治疗相关的问题：

引入了由知识蒸馏 + 正交约束再训练组合的模型治疗策略，如下是具体效果（由于刚跑通，实验只做了几次，后续会扩大）：

遗忘目标客户端是0号（其主要数据集是4号，都已加粗）：

阶段	测试集精度 (test_acc)	客户端0自有数据精度
Training	0.8083	93.01%
Retrain	0.8026	60.00%
FAIR-VUE	0.7277	41.80%
HEAL	0.7726	50.89%

类别ID	Training	Retrain	FAIR-VUE	HEAL
0	0.9776	0.9959	0.9724	0.9776
1	0.9912	0.9982	0.9903	0.9930
2	0.9680	0.9729	0.9603	0.9671
3	0.6931	0.9139	0.5129	0.8317
4	**0.9053**	**0.3136**	**0.0000**	**0.1354**
5	0.7444	0.4854	0.9664	0.8812
6	0.9729	0.9666	0.9687	0.9760
7	0.9844	0.9650	0.9154	0.9650
8	0.0021	0.3871	0.0000	0.0421

类别ID	Training	Retrain	FAIR-VUE	HEAL
9	0.7948	0.9455	0.9514	0.9078

客户端ID	Training	Retrain	FAIR-VUE	HEAL
0	**93.01**	**60.00**	**41.80**	**50.89**
1	27.82	56.21	26.42	30.88
2	76.26	93.49	83.38	87.94
3	96.38	90.21	84.43	89.18
4	97.46	91.55	87.72	89.78
5	63.43	86.36	47.75	75.46
6	76.85	61.17	96.16	89.66
7	98.39	99.62	97.89	97.97
8	75.06	86.49	72.99	78.44
9	96.95	98.17	96.08	96.51

能明显看到，引入治疗后准确率有一定程度的回升

现在最大的问题是mnist手写数据集太容易学习，导致重训练后即使抛弃了目标客户端的数据，依旧能学得较好，后续会替换为cifar数据集进行下一步实验

下面是技术细节

healing 模块（也就是 `selective_kd_heal` 与 `heal_model`）本质上是一种 **“知识蒸馏 + 正交约束再训练（Knowledge Distillation with Orthogonal Regularization）”** 的治疗机制，用来在 FAIR-VUE 遗忘之后修复模型的整体性能。
它的**核心思想**可以分为三步来理解：

一、背景：为什么需要“模型治疗”

在联邦遗忘中，我们会对模型权重沿着被遗忘客户端的“影响子空间”做投影擦除（即 FAIR-VUE 的 `V_spec` 空间）。
这一步虽然可以让模型“忘掉”目标客户端的知识，但副作用是：

- 全局模型的结构被破坏；
- 其他客户端的性能（尤其是共享类）显著下降。

> 🔹 所以 healing 的目标：在不重新训练被遗忘数据的前提下，**让模型在 retained clients 的分布上恢复性能，同时避免重新学回被忘知识**。

二、原理核心：知识蒸馏 + 正交惩罚

（1）知识蒸馏（KD）

healing 阶段会使用一个 **teacher 模型**（即遗忘前的模型或未投影版本）来指导 **student 模型**（投影后的遗忘模型）。

两者在同样的数据上计算预测分布，然后通过 **KL 散度损失** 让 student 向 teacher 的知识分布靠近：

$$\mathcal{L}_{KD} = T^2 \cdot KL\big(\text{softmax}(z_s/T); ||; \text{softmax}(z_t/T)\big)$$

- (z_s, z_t)：student/teacher 的 logits；
- (T)：温度参数（一般 2～4），用于平滑预测分布；
- KD loss 让 student 重新吸收非遗忘客户端的通用知识。

（2）正交惩罚（Orthogonal Regularization）

这一部分就是 healing 的“关键创新”：

之前遗忘时擦除了一个特征子空间 (V_{spec})，表示被遗忘客户端的影响方向。
为了防止 student 在治疗过程中**重新学回被遗忘知识**，需要约束模型梯度不再朝 (V_{spec}) 方向更新。

具体做法是：

$$\mathcal{L} * ortho = \lambda * ortho \cdot |Q^\top \theta| * 2^2$$

其中 (Q) 是对 (V*{spec}) 的正交基，(\theta) 是当前模型参数向量。
这相当于：

- 投影当前模型参数到被遗忘方向；
- 惩罚这些分量的大小；
- 让模型保持在被遗忘空间的正交补内。

所以你最终优化的总损失为：

$$\mathcal{L} * total = \mathcal{L} * task + \lambda_{KD}\mathcal{L} * KD + \lambda * ortho\mathcal{L}_{ortho}$$

这一步相当于 **“在不触碰遗忘方向的情况下重新蒸馏模型”**。

三、训练过程

整体流程可以简化为下图的逻辑：

```
遗忘后模型 (student_init)
        ↓
teacher (未遗忘模型)
        ↓
healing dataloader (仅保留客户端数据)
        ↓
Loss = CE + KD + λ·Orthogonal
        ↓
优化几百步（num_steps=200）
        ↓
student_final（恢复准确率但仍遗忘目标客户端）
```

四、关键点总结

模块	目的	对应参数
知识蒸馏 (KD)	恢复 retained clients 的知识	λ_kd, T

模块	目的	对应参数
正交惩罚 (Orthogonal)	防止重新学回被遗忘方向	λ_ortho, v_spec
短步数微调 (Selective Heal)	控制治疗强度	num_steps
Teacher 冻结	保持指导一致性	teacher.eval()