On-Policy Distillation

Thinking Machines

几种SFT方式

1. 监督式 FT (Supervised FT)
    a) 学习对象： “标准答案” (Ground-truth) 。
    b) 学习方式： 学生模型被训练去模仿一个固定的、真实的输出数据集
    c) 训练信号 (Signal)： **“硬”标签 (Hard Label)**
2. 序列级 KD (Sequence-Level KD)
    a) 学习对象： “老师的答案”
    b) 学习方式： 首先，我们让老师模型（Teacher）自己先做一遍题，生成一套它的答案（高概率序列）。然后，学生模型再去模仿“老师的这套答案”
    c) 训练信号 (Signal)： **教师的“硬”标签 (Hard Label)**
3. 监督式 KD (Supervised KD)
    a) 学习对象： “老师的完整思路”
    b) 学习方式： 老师在写每一个词的时候，它脑子里是怎么想的
    c) 训练信号 (Signal)： “软”标签 (Soft Label)。在每一步，学生模型被训练去模仿老师的“完整token概率分布” 。（可以理解为是**logic分布**）

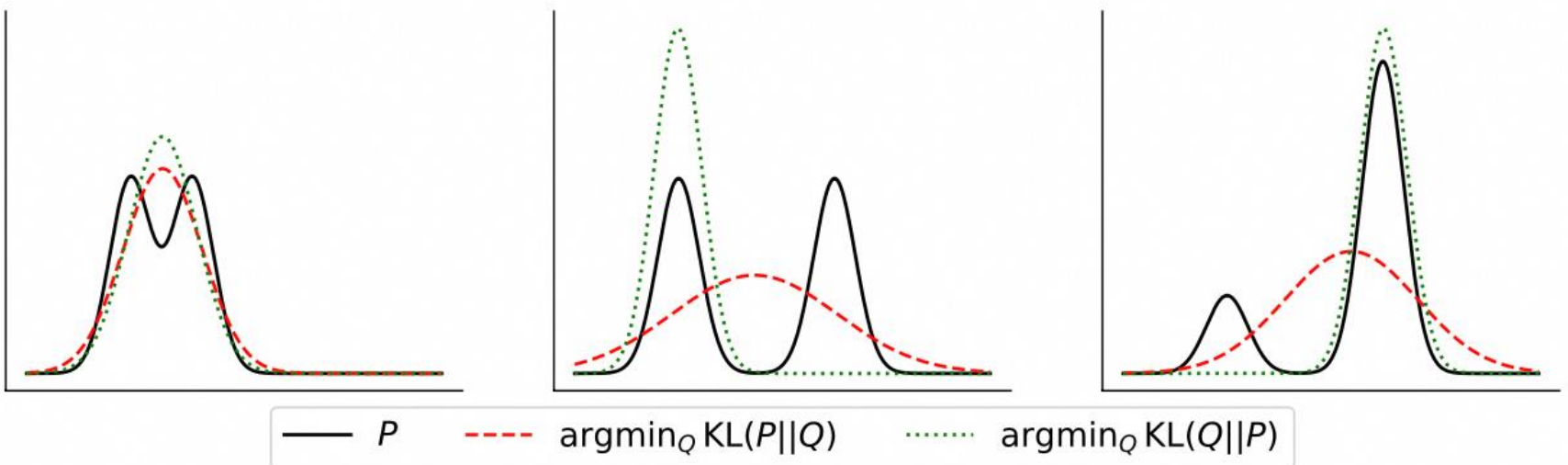
前置知识：KL and JS

1. Forward KL Divergence（Mode-Covering）：前向 KL 是指从“真实”分布 P（老师）到“近似”分布 Q（学生）的散度，学生 Q 试图覆盖老师 P 的所有可能性
2. Reverse KL Divergence（Mode-Seeking）：反向 KL 是指从“近似”分布 Q（学生）到“真实”分布 P（老师）的散度，学生 Q 试图只说老师 P 最常说的话
3. Generalized JS Divergence：JSD 是一种试图在前向 KL 和反向 KL 之间进行插值（Interpolate）或权衡的散度

$$D_{KL}(P||Q) = \sum_{c \in C} P(c) \log \frac{P(c)}{Q(c)}$$

$$D_{KL}(Q||P) = \sum_{c \in C} Q(c) \log \frac{Q(c)}{P(c)}$$

$$D_{JSD(\beta)}(P||Q) = \beta D_{KL}(P||\beta P + (1-\beta)Q) + (1-\beta) D_{KL}(Q||\beta P + (1-\beta)Q)$$



跟SFT对比：Generalized Knowledge Distillation – GDM,ICLR 2024

Distribution Mismatch: 对于传统蒸馏/SFT，**存在**“训练 (Training)” **和** “**推理** (Inference)” **有**问题：

1.训练 (Training) 时：模型学习时，通常是基于一个“标准答案”数据集。在生成每个词元时，它都会被喂入正确的上文。永远只**会见到正确的轨迹** – 之前OAI提到的幻觉成因

2.推理 (Inference) 时：模型在实际生成文本时，没有“标准答案”可供参考。它必须基于自己上一步生成的词元来预测下一个词元。如果它在第一步犯了个小错，下一步它就必须基于这个错误继续往下生成

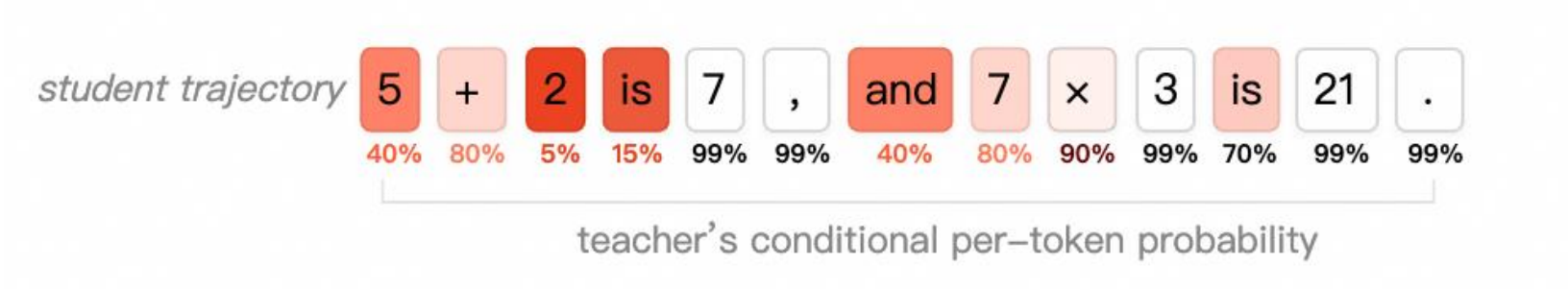
训练的模型只在**“完美路径”上训练过**，却被要求在“真实路径”（包含自己的错误）上工作

跟RL的对比

1. 同策略强化学习（On-policy RL） 类似于在没有指导的情况下自己下棋。赢或输的反馈与你自己的棋局直接相关，但每局只在最后获得一次，并且不会告诉你哪些棋步对结果有贡献。
2. 异策略蒸馏（Off-policy distillation） 类似于观看一位特级大师下棋——你观察到非常高超的棋步，但这些棋步是在新手玩家几乎永远不会遇到的棋盘状态下走出的

怎么解决？

Method	Sampling	Reward signal
Supervised finetuning	off-policy	dense
Reinforcement learning	on-policy	sparse
On-policy distillation	on-policy	dense

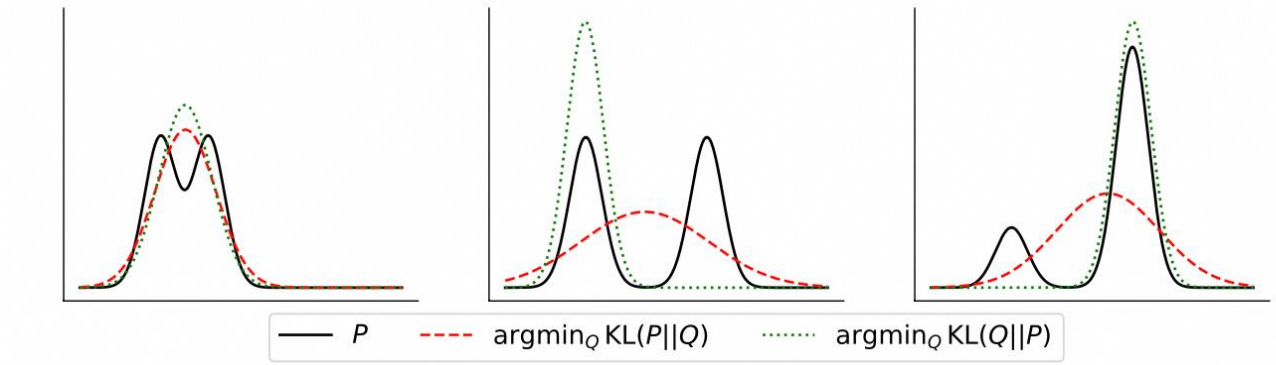


他们的做法Reverse KL

$$Reward = -\mathrm{KL}\left(\pi_\theta \| \pi_{\text{teacher}}\right) = -\mathbb{E}_{x\sim\pi_\theta}\left[\log \pi_\theta\left(x_{t+1} \mid x_{1..t}\right) - \log \pi_{\text{teacher}}\left(x_{t+1} \mid x_{1..t}\right)\right]$$
$$= \log \pi_{\mathrm{d}}\left(x_{t+1} \mid x_{1..t}\right) - \log \pi_\theta\left(x_{t+1} \mid x_{1..t}\right)$$

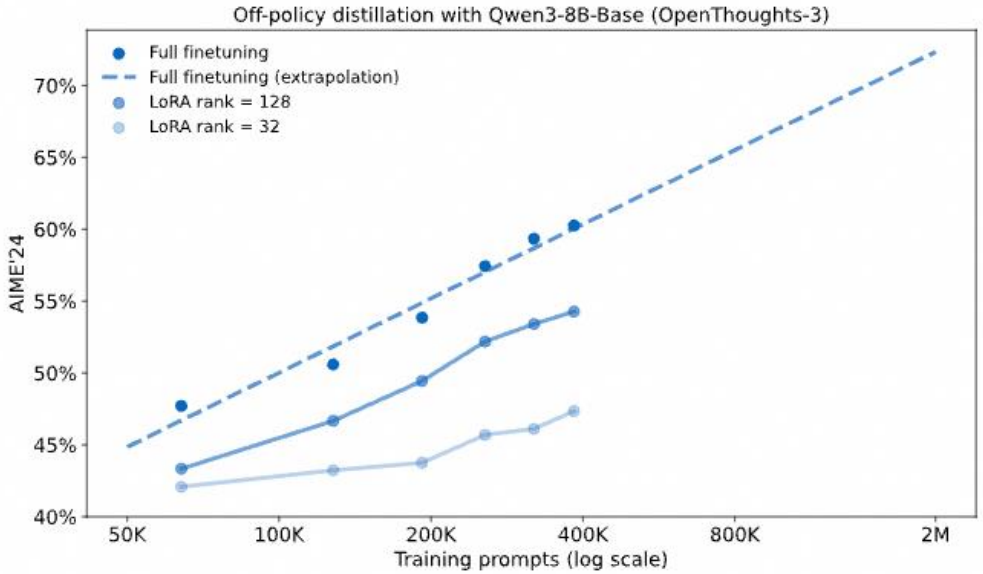
从三个角度理解:

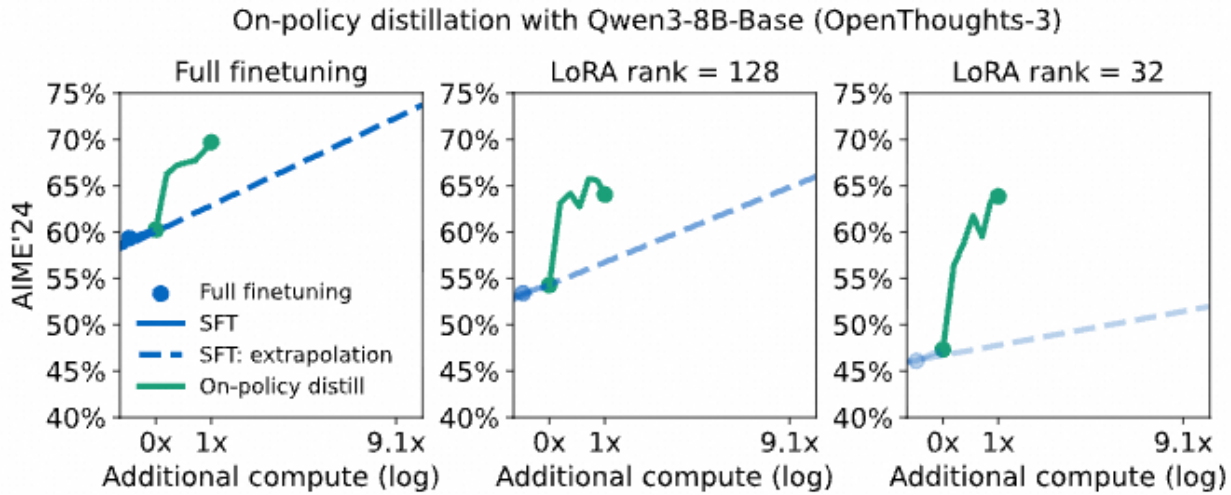
1. 它是“寻求模式的”（mode seeking） —— 它学习一种特定行为（教师的行为），而不是将其分布分散在几个次优选项上；并且它减少了“暴露偏差”（exposure bias）。
2. 从llm本身的logic是最优reward思路一致（yuyang的那篇inverse RL）
3. 我上次推导的那个rl里面的最优reward方向一致（他这个只是我上次推出来的下位替代）



组合	策略 (Policy)	散度 (Divergence)	含义 (学生 Q vs 老师 P)
1. 监督 KD	**Off-Policy** (老师的轨迹)	**Forward KL** (模式覆盖)	要求学生模型 (Q) 在*老师的轨迹*上，**覆盖**老师模型 (P) 的*整个*概率分布。
2. GKD (变体)	**On-Policy** (学生的轨迹)	**Forward KL** (模式覆盖)	要求学生模型 (Q) 在*自己的轨迹*上，**覆盖**老师模型 (P) 的*整个*概率分布。
3. On-Policy 蒸馏	**On-Policy** (学生的轨迹)	**Reverse KL** (模式寻求)	要求学生模型 (Q) 在*自己的轨迹*上，其输出分布**集中于**老师模型 (P) 的*高概率模式*。
4. (罕见组合)	**Off-Policy** (老师的轨迹)	**Reverse KL** (模式寻求)	要求学生模型 (Q) 在*老师的轨迹*上，其输出分布**集中于**老师模型 (P) 的*高概率模式*。

对于reasoning，on policy distill效果如何？





GPU 小时计算的成本降低接近 18 倍，比SFT和RL比

Method	AIME'24	Teacher FLOPs	Student FLOPs	CE vs SFT-2M
Initialization: SFT-400K	60%	8.5×10^{20}	3.8×10^{20}	–
SFT-2M (extrapolated)	~70% (extrapolated)	3.4×10^{21}	1.5×10^{21}	1×
Reinforcement learning	68%	-	-	≈1×
On-policy distillation	70%	8.4×10^{19}	8.2×10^{19}	**9-30×**

不同训练代表什么?

1. 预训练（pre-training）/**SFT**：通过随机梯度下降（stochastic gradient descent）在高维参数空间中进行探索，在参数空间进行搜索（让「给定输入 → 预测下一个 token」尽量拟合数据分布，用哪些参数组合？）
2. RL：在语义策略空间（semantic strategy space）中进行探索，每一条traj用哪些token进行组合？**RL 并不会将大量算力花在梯度更新本身上。我们应当将 RL 理解为主要把计算资源花在**搜索（search）上——即 rollout 一个策略并分配奖励（assigning credit），而「反向传播更新参数」只占其中一小部分算力
3. 蒸馏：一旦找到了一种好的策略，蒸馏就能作为学习该策略的捷径：**on-policy distillation 不需要建模 RL 训练过程中出现的所有中间策略，而只需学习最终得到的策略。**

如果用来蒸馏强化后的模型会发生什么？

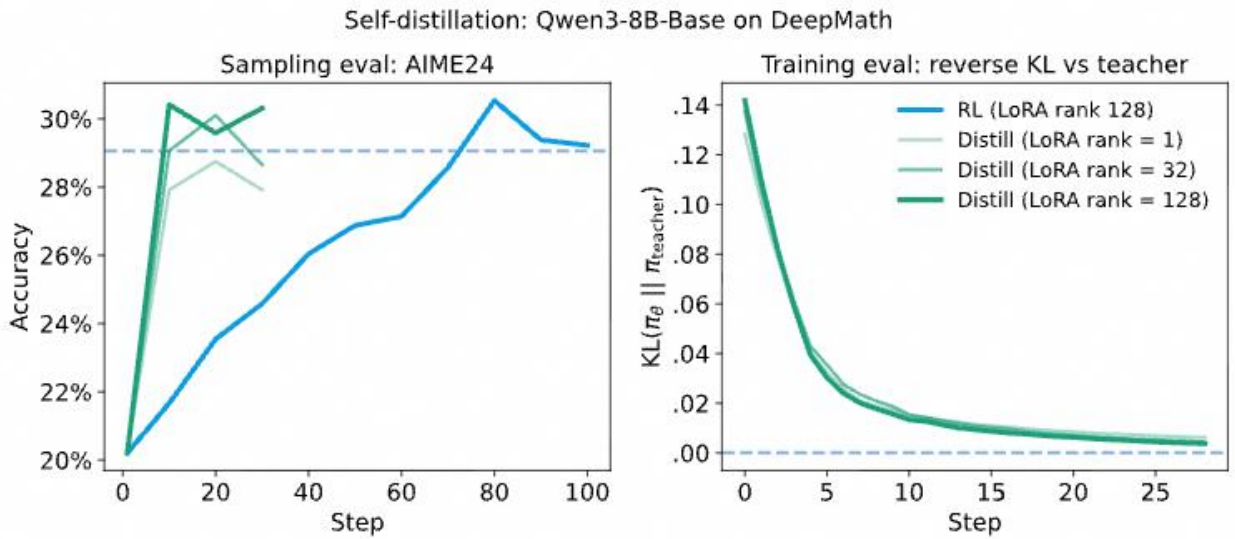


Figure 12: Starting from the same initialization, on-policy distillation can learns the RL-trained policy in approximately 7-10x fewer gradient steps, which corresponds to a compute efficiency of 50-100x.

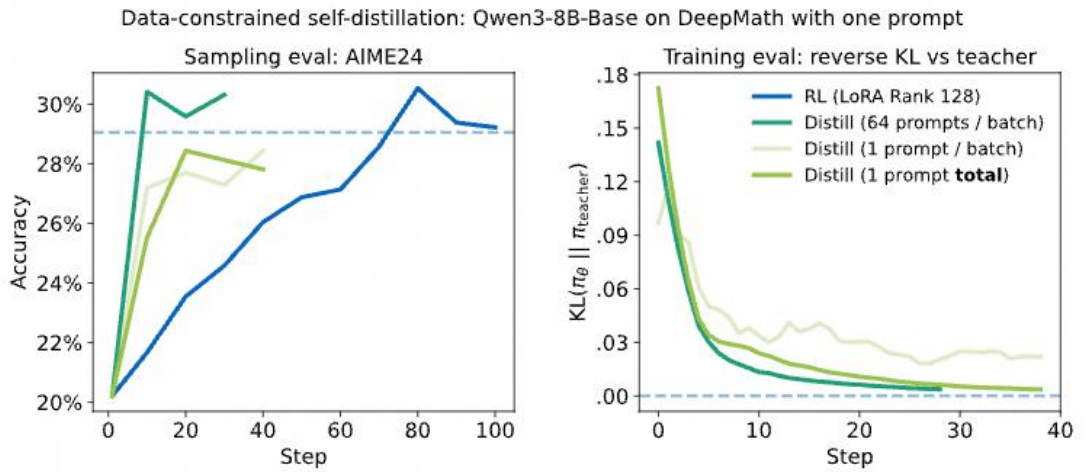


Figure 13: In this example, multi-epoch training on top of one training example is sufficient to distill the teacher's AIME'24 performance. Our default configuration (also used in the personalization experiments) runs on-policy distillation with 64 prompts / batch, and 4 samples / prompt. All methods shown are trained with 256 samples / batch. Note that the right chart is showing training KL, hence it is natural for 1 prompt total to outperform 1 prompt / batch.

RL学的内容，是非常Sparse的，只需要改几个token就成立，而且是on policy的，非常容易蒸回来

Catastrophic forgetting

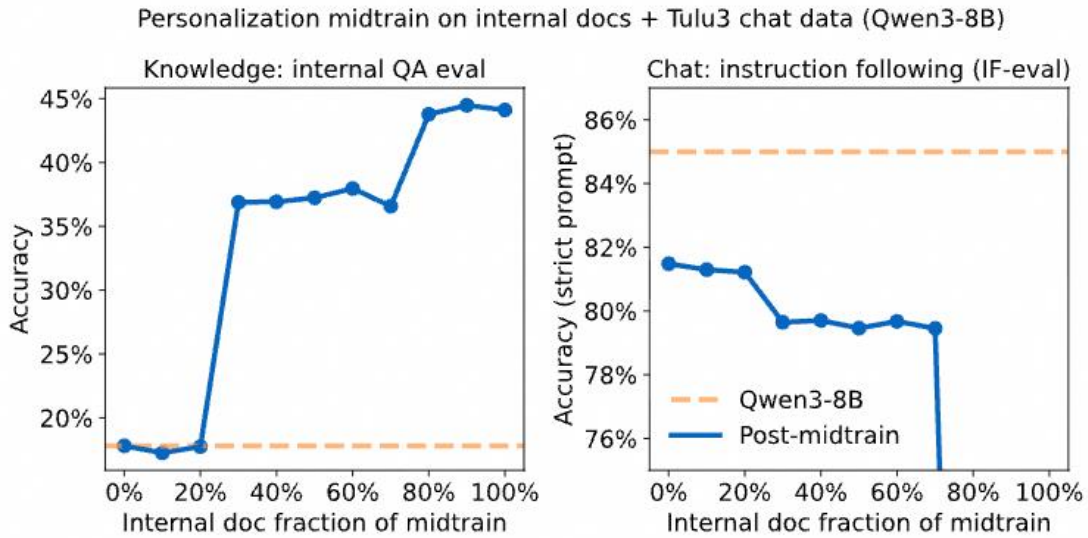
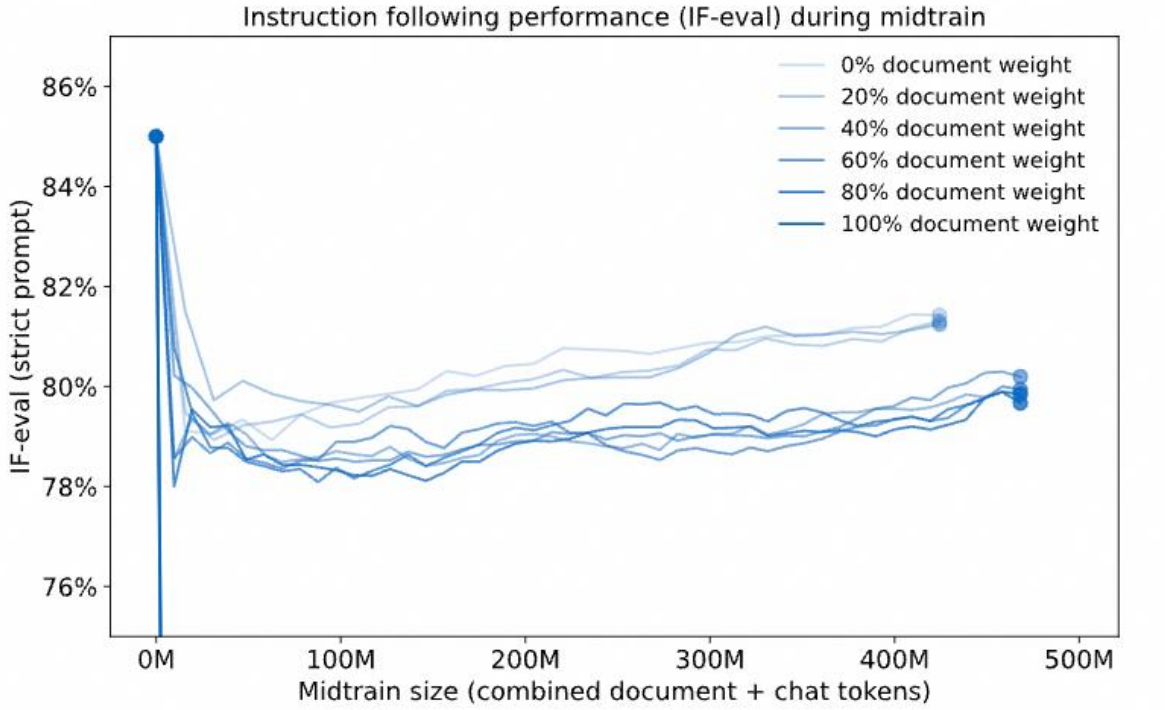
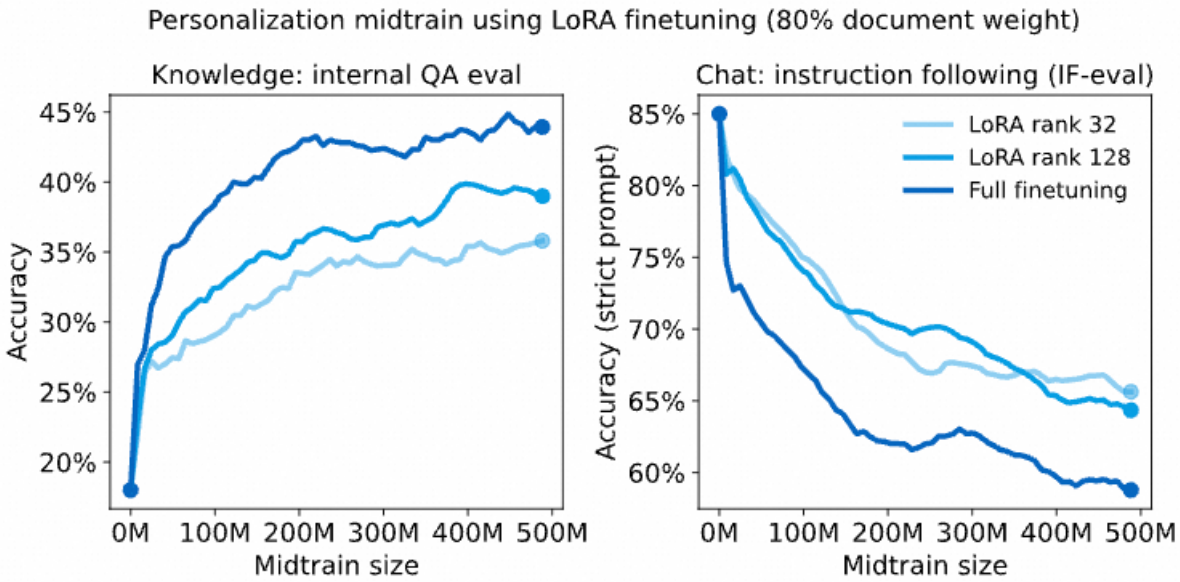


Figure 9: Sweeping over the ratio of internal documents: background chat data during mid-training. Although mixing in a little chat data helps to prevent a catastrophic regression, no weight maintains the original IF-eval performance.





on-policy 学习（RL）的遗忘程度比 off-policy 学习要小

在该模型自身样本的数据集上运行 SFT，会发生什么？

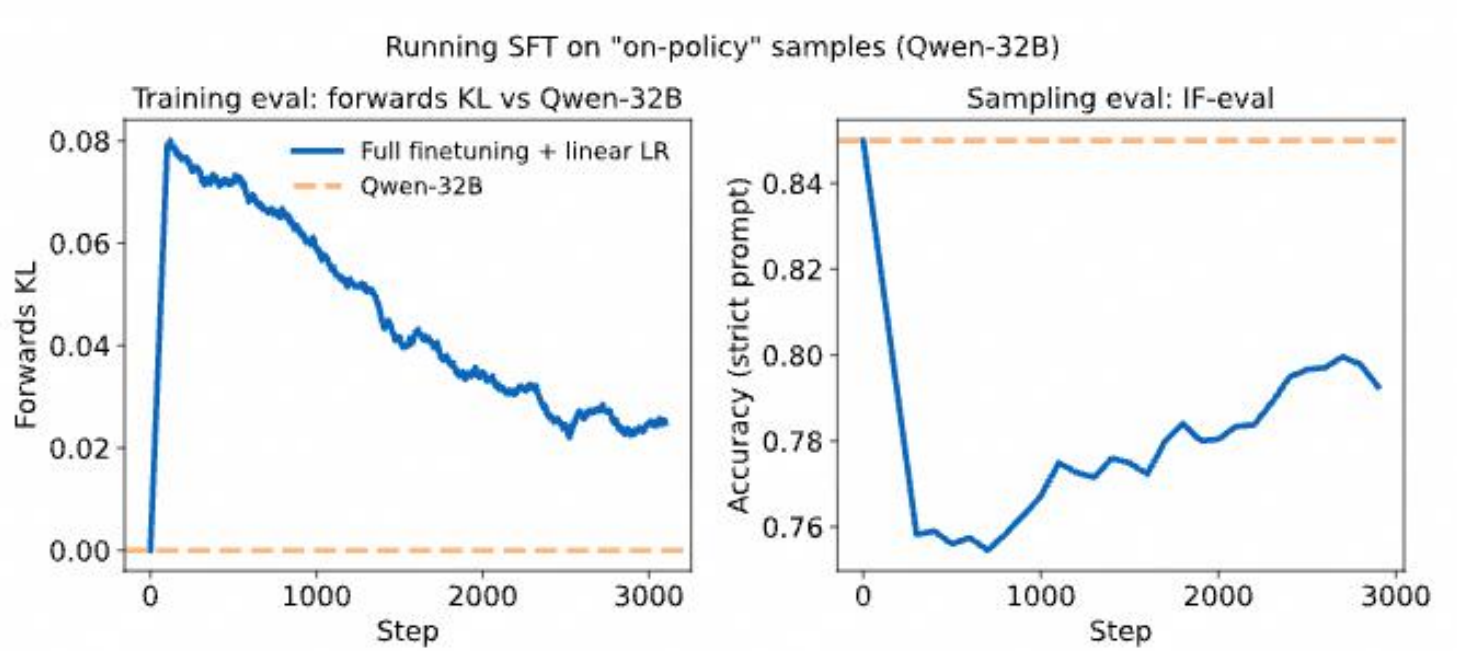


Figure 14: Running SFT on top of Qwen3-32B's own samples degrades performance. We use the same learning rate as from the personalization section, which was swept for practical performance considerations. A linear learning rate can prevent forwards KL / IF-eval from regressing indefinitely, but does not recover performance before the LR decays to zero.

虽然 KL 散度在期望意义上为 0，但**每个有限批次**在实际中都会表现出略有不同的分布。
在这些有限批次上训练会引起**非零梯度更新**，从而使更新后的模型策略偏离原始状态。
这种过程会将“在自身样本上训练”逐渐转化为 off-policy 训练，导致长序列上与 off-policy 训练相同的误差积累和发散问题

用on policy distill可以完美解决

Model	Internal QA Eval (Knowledge)	IF-eval (Chat)
Qwen3-8B	18%	**85%**
+ midtrain (100%)	**43%**	45%
+ midtrain (70%)	36%	79%
+ midtrain (70%) + distill	**41%**	**83%**

Domain-specific (Internal QA eval) and chat (IF-eval) performance after mid-training. Although mid-training forgets the post-trained behaviors of Qwen3-8B, they are cheaply restored via on-policy distillation, alongside the additional knowledge learned via the mid-train.

且由于教师模型保持固定，学生模型会逐步收敛到教师期望的行为，而不会像 SFT 那样在自蒸馏设置中出现性能回退

Future work & Limitation

1. 持续学习，绝对吊打之前的方法（首先之前的bench就有问题，都太短了）
2. 无缝迁移到MLLMs里面去，只需要再多考虑个perception的东西就好了
    1. 非常好做
3. AI4S，从之前的任务看，两个模型互相蒸，虽然故事不好讲，但是绝对可以训一个很牛逼的模型
4. 自回归图片生成，这个可太好做了
5. MoE? 是否可以做的更sparse?

1. 这个不是理论最优，理论最优是上次组会分析的版本，这个只是取log近似
2. 不能无脑蒸，如果真的想做work，得配合sft和on policy distill一起
3. 比较适用于，sparse的更新，例如RL。差异过大，效果不可以