Pi0 Few-shot（少样本微调：用很少 demo 训练）实验设计与推荐参数（5/10/20 demos）

本文档面向 RoboTwin 的 Pi0 微调流程（`pi0_base_aloha_robotwin_lora`），目标是在 **5/10/20 条 demo（示范轨迹：一次任务的专家演示记录）**下，用更少训练预算（steps（更新步：参数更新一次）缩短）尽量逼近 50 demo 的成功率（clean 与 randomized 都评测）。

你现有的 baseline 流程不变；本文档只新增"可选开关"的 few-shot 训练配方（recipe（配方：一套默认参数 + 协议）），核心依赖 Plan-STaR：Sampling Plan（采样计划文件：提前把"训练看哪些样本、每个样本权重多少"写死，训练只读 plan 单阶段运行）。

---

0) 为什么 few-shot 会掉点（通俗解释）

少样本下最常见的三类问题：

1. **重复暴露太高 → 过拟合/遗忘**
   - 过拟合（overfitting：把训练 demo 背下来，换初始状态就不行）
   - 灾难性遗忘（catastrophic forgetting：把 base 模型原本能力训没了）
2. **样本相关性太强 → 同样信息反复出现**
   - 相邻帧很像，短训预算下等于"把步数浪费在重复画面上"
3. **短训 schedule 不匹配 → 不收敛或震荡**
   - schedule（学习率计划：学习率随训练步数如何变化）如果仍按 30000 步配置，用到 6000/3000 步容易不合适
   - warmup（学习率热身：开头从小学习率逐渐升高）与 decay（学习率衰减：后面逐渐变小）都需要按总步数缩放

Plan-STaR 的核心就是把"训练看到的数据分布"做成 **可审计、可复现、可消融** 的静态协议；它不改变评测任务定义（评测仍按真实环境 rollout，不做窗口化重采样）。

---

1) 你需要记录的关键指标（用于诊断）

1.1 在线评测指标（eval 输出）

每次评测都记录：

- `success_rate`：成功率（clean / randomized）
- `failure_counts.timeout`：timeout（超时：达到评测步数上限仍未成功）的次数
- `avg_steps_success`：成功 episode 平均步数（越小通常表示更利落）
- `eval_time_sec`：评测耗时（避免"成功率提升但推理变慢"的误判）

这些在评测输出 `_metrics.json` 里：
`eval_result/<task>/<policy>/<task_config>/<exp>/<time>/_metrics.json`。

1.2 训练"有效样本暴露量"（exposure）指标（train 输出）

训练时会在 `run_summary.json` 里记录：

- `plan_num_anchors`：anchors（锚点：窗口起点索引）的数量 N
- `exposure_n = steps * batch_size`：总抽样次数 n
- `exposure_expected_unique`：期望看到的不同 anchors 数
- `exposure_avg_repeat = n / expected_unique`：平均重复次数（越大越容易过拟合/遗忘）
- `exposure_ESS`：ESS（有效样本量：权重分布等效的“均匀样本数”）

实用经验：**当 `exposure_avg_repeat` 比 50-demo baseline 高一个数量级时，clean 也可能掉穿**（尤其表现为 timeout 激增）。

---

2) 推荐的 few-shot 总体策略（不引入外部大离线库）

2.1 两条“公平预算口径”（论文友好）

1. compute-limited（算力受限：固定训练 steps）
   - 固定 `steps ∈ {3000, 6000}`（对应 30000 的 1/10 与 1/5）
2. repeat-matched（重复匹配：固定平均重复次数）
   - 先生成 plan 得到 `N = plan_num_anchors`
   - 设目标平均重复次数 `R_target`（建议用 50-demo@6000 的量级，约 30~40）
   - 计算：`steps_repeat = round(R_target * N / batch_size)`
   - 直觉：让 5/10/20-demo 不会因为“样本太少但训太久”而被不公平地过拟合

两条口径都做，审稿人更难挑公平性毛病。

2.2 先保证“覆盖”，再谈“关键事件加权”

你当前 10-demo 计划文件里 anchors 只有几百个，6000 steps 会导致每个样本重复几百次。

因此推荐优先顺序：

1. **增大 anchors 数（覆盖更广）**：`stride_S=1` + `valid_start_tail_keep=H-1`
2. **抑制短训过拟合**：启用 `plan-aware-lr-scale`（plan 感知学习率缩放）
3. **强调关键瞬间**：启用 `gripper_event_enabled`（夹爪事件加权：让“夹爪开/合附近窗口”出现得更频繁）

---

3) 推荐参数配置（可直接用）

下面给出 3 套 plan 配方（A/B/C） + 1 套训练配方（T），你可以在 5/10/20 demos 上复用。

3.1 Plan-A（覆盖优先：先把 anchors 做大）

适用：你现在 clean 掉点很大时，先用它把 training exposure 拉回合理范围。

```
cd policy/pi0
uv run scripts/build_sampling_plan.py \
  --repo-id beat_block_hammer-demo_clean-50 \
  --train-config-name pi0_base_aloha_robotwin_lora \
  --num-episodes-keep 10 \
  --episode-selection-seed 0 \
```

```
  --plan-seed 0 \
  --sampler-seed 0 \
  --replacement true \
  --stride-S 1 \
  --valid-start-tail-keep 49 \
  --stage-strategy time_quantile \
  --num-stages-K 4 \
  --stage-balance uniform_over_stage \
  --clip-ratio 5.0 \
  --epsilon-mix 0.01 \
  --out-dir sampling_plans/beat_block_hammer/demo10_s0_planA
```

为什么这么选（通俗）：

- `stride-S=1`（步幅：隔几帧取一次样本）让 anchors 尽可能多，减少“同一帧反复出现”的浪费。
- `valid-start-tail-keep=49`：允许更多靠近 episode 尾部的起点进入训练，进一步增大 anchors（降低重复暴露）。
- `time_quantile`（时间分位数：按时间把轨迹均分成 K 段）最稳、跨任务不容易出错，先当安全基线。

3.2 Plan-B（关键阶段更语义：change-point stage）

适用：Plan-A 已经不掉穿，但 randomized 仍然明显掉点/方差大时。

```
cd policy/pi0
uv run scripts/build_sampling_plan.py \
  --repo-id beat_block_hammer-demo_clean-50 \
  --train-config-name pi0_base_aloha_robotwin_lora \
  --num-episodes-keep 10 \
  --episode-selection-seed 0 \
  --plan-seed 0 \
  --sampler-seed 0 \
  --replacement true \
  --stride-S 1 \
  --valid-start-tail-keep 49 \
  --stage-strategy change_point_state \
  --num-stages-K 4 \
  --change-point-delta 3 \
  --change-point-min-gap 10 \
  --stage-balance uniform_over_stage \
  --clip-ratio 5.0 \
  --epsilon-mix 0.01 \
  --out-dir sampling_plans/beat_block_hammer/demo10_s0_planB
```

为什么这么选（通俗）：

- `change_point_state`（变化点：看 state 变化突然变大处）更像“关键动作发生的转折点”，比时间均分更语义化，但也更容易受噪声影响，所以放在 Plan-A 之后。

3.3 Plan-C（夹爪事件加权：强调开合附近窗口）

适用：你认为任务成败强依赖夹爪开/合（例如抓取/释放），且评测失败主要是 timeout（做到一半卡住）。

```
cd policy/pi0
uv run scripts/build_sampling_plan.py \
  --repo-id beat_block_hammer-demo_clean-50 \
  --train-config-name pi0_base_aloha_robotwin_lora \
  --num-episodes-keep 10 \
  --episode-selection-seed 0 \
  --plan-seed 0 \
  --sampler-seed 0 \
  --replacement true \
  --stride-S 1 \
  --valid-start-tail-keep 49 \
  --stage-strategy time_quantile \
  --num-stages-K 4 \
  --stage-balance uniform_over_stage \
  --gripper-event-enabled \
  --gripper-dims 6,13 \
  --gripper-smooth-window 5 \
  --gripper-delta1 1 \
  --gripper-delta2 5 \
  --gripper-quantile-q 0.97 \
  --gripper-min-gap 10 \
  --gripper-event-mode soft_exp \
  --gripper-sigma 10.0 \
  --gripper-gamma 2.0 \
  --clip-ratio 5.0 \
  --epsilon-mix 0.01 \
  --out-dir sampling_plans/beat_block_hammer/demo10_s0_planC_gripper
```

为什么这么选（通俗）：

- 夹爪事件加权不是“看到夹爪有一点点变化就加权”，而是：
  - 先做中值滤波（median filter：用邻域中位数去噪，抑制细微抖动）
  - 再用高分位阈值 `q=0.97` 只保留“变化幅度最大的那一小撮时刻”（避免正常运动的细微变化误报）
  - 再用 NMS（非极大值抑制：把挤在一起的多个峰合并）保证事件不密集重复
- 结果是：训练更常见到“抓取/释放”附近的窗口，减少“做到关键一步卡住 → timeout”。

> 备注：`gripper-dims 6,13` 是 Aloha 双臂夹爪维度（见 `policy/pi0/src/openpi/policies/aloha_policy.py`），如果你的 state 维度定义不同，需要先确认。

3.4 训练配方 T（单阶段训练：短训稳定优先）

推荐默认（先不要 `freeze-mode strong`）：

```
cd policy/pi0
XLA_PYTHON_CLIENT_MEM_FRACTION=0.45 uv run scripts/train.py
```

```
pi0_base_aloha_robotwin_lora \
  --exp-name beat_block_hammer-demo_clean-10_planC_steps3000_s42 \
  --overwrite \
  --seed 42 \
  --num-train-steps 3000 \
  --budget-aware-schedule \
  --freeze-mode default \
  --plan-aware-lr-scale \
  --plan-target-effective-repeat 50 \
  --plan-lr-min-scale 0.05 \
  --sampling-plan-path
sampling_plans/beat_block_hammer/demo10_s0_planC_gripper
```

为什么这么选（通俗）：

- `budget-aware-schedule`（预算感知学习率计划：把 warmup/decay 按总步数缩放）避免“把 30000 的学习率曲线硬套到 3000/6000 导致不收敛”。
- `plan-aware-lr-scale`（plan 感知学习率缩放：样本重复太高时自动把学习率降下来）抑制过拟合/遗忘。
- `freeze-mode default`：先对齐 baseline 的可训练容量；`strong` 更像“额外正则”，建议作为消融而非默认。

---

4) norm stats（归一化统计：把输入标准化）的对照：reuse vs recompute

norm stats 在少样本下可能是“稳定器”也可能是“噪声源”，因此建议纳入主矩阵的控制变量：

4.1 reuse（复用：用 50-demo 的统计，不重算）

训练时不额外传 `--data.assets.asset-id`，默认会用 `repo_id` 对应的资产目录。

4.2 recompute（重算：只用 plan 选中的 episodes_valid 统计）

```
cd policy/pi0
uv run scripts/compute_norm_stats.py pi0_base_aloha_robotwin_lora \
  --plan-path sampling_plans/beat_block_hammer/demo10_s0_planC_gripper \
  --plan-subset-mode episodes_valid \
  --output-asset-id beat_block_hammer-demo_clean-10_planC_stats
```

训练时追加：

```
... uv run scripts/train.py pi0_base_aloha_robotwin_lora \
  ... \
  --data.assets.asset-id beat_block_hammer-demo_clean-10_planC_stats \
  --sampling-plan-path
sampling_plans/beat_block_hammer/demo10_s0_planC_gripper
```

为什么要做这个对照（通俗）：

- reuse：统计更稳定（样本多），但可能和你选出来的 5/10/20 demo 分布不完全一致
- recompute：更贴合当前子集，但样本少可能导致统计更噪（尤其是 std/quantile）

---

5) 实验路线图（建议按时间顺序）

下面给出一个“最小但论文友好”的实验路线（先 1 个任务跑通，再扩到多任务）。

5.1 E0（地基实验：先解释掉点来自哪里）

组名	demos	steps	plan	freeze	norm	目的
E0-50-3000	50	3000	off	default	reuse	50-demo 短训基线（对齐你已有 0.47）
E0-50-6000	50	6000	off	default	reuse	50-demo 更长短训基线
E0-10-3000-A	10	3000	Plan-A	default	reuse	验证“先增 anchors 是否能避免掉穿”
E0-10-3000-C	10	3000	Plan-C	default	reuse	验证“夹爪事件加权是否减少 timeout”
E0-10-3000-C+stats	10	3000	Plan-C	default	recompute	归一化统计是否是关键变量

评测：每组都跑 `demo_clean` 和 `demo_randomized`。

E0 的 baseline（plan=off）训练命令示例：

```
cd policy/pi0
XLA_PYTHON_CLIENT_MEM_FRACTION=0.45 uv run scripts/train.py
pi0_base_aloha_robotwin_lora \
  --exp-name beat_block_hammer-demo_clean-50_baseline_steps3000_s42 \
  --overwrite \
  --seed 42 \
  --num-train-steps 3000 \
  --budget-aware-schedule \
  --freeze-mode default
```

5.2 主矩阵（5/10/20/50 demos）

对每个任务（建议先 `beat_block_hammer`）做：

compute-limited（固定 steps）

- demos ∈ {5,10,20,50}
- steps ∈ {3000,6000}
- plan：demos<50 用 Plan-C（或 Plan-A 作为更稳的备份）；50 用 off（baseline）

- seeds：至少 3 个 `--seed`（例如 42/43/44）；episode 子集也至少 3 个 `--episode-selection-seed`（例如 0/1/2）

repeat-matched（固定 avg_repeat）

- 先读 plan_meta 的 `summary.num_anchors`
- 取 `R_target=35`（建议与 50-demo@6000 的重复量同量级）
- 用 `steps_repeat = round(R_target * N / batch_size)` 计算 steps（并记录到 runlist）

5.3 严格消融（固定 10-shot，控制成本）

固定：`demos=10, steps=3000, seed=42, episode_selection_seed=0`，每次只改一个因素：

组名	变化点	你期望看到什么（写论文解释用）
A0	Plan-A（覆盖）	若 A0 比无 plan 更好：说明“去相关/增 anchors”是必要条件
A1	A0 + plan-aware-lr-scale=off	若掉点：说明“少样本高重复需要降 LR 防遗忘”
A2	A0 + valid_start_tail_keep=0	若掉点/timeout 增：说明尾部过滤过严导致 anchors 太少
A3	Plan-B（change-point）	若 randomized 提升：说明“阶段更语义化”有助鲁棒
A4	Plan-C（gripper on）	若 timeout 明显下降：说明“夹爪事件”确实是关键瞬间
A5	freeze-mode strong	若 clean 掉点但 randomized 更稳：说明强冻结是正则（但可能欠拟合）
A6	norm recompute	若稳定性↑或↓：把 norm stats 作为必要控制变量写进论文

6) 具体命令行模板（把占位符替换即可）

6.1 生成 plan（以 5/10/20 为例）

```
cd policy/pi0
for D in 5 10 20; do
  for ES in 0 1 2; do
    uv run scripts/build_sampling_plan.py \
      --repo-id beat_block_hammer-demo_clean-50 \
      --train-config-name pi0_base_aloha_robotwin_lora \
      --num-episodes-keep ${D} \
      --episode-selection-seed ${ES} \
      --plan-seed 0 \
      --sampler-seed 0 \
      --replacement true \
      --stride-S 1 \
      --valid-start-tail-keep 49 \
      --stage-strategy time_quantile \
      --num-stages-K 4 \
```

```
      --stage-balance uniform_over_stage \
      --gripper-event-enabled \
      --gripper-dims 6,13 \
      --gripper-quantile-q 0.97 \
      --gripper-gamma 2.0 \
      --out-dir sampling_plans/beat_block_hammer/demo${D}_es${ES}_planC
  done
done
```

6.1.1 repeat-matched：从 plan_meta 读 N 并计算 steps

plan 的 anchors 数 `N` 在 `plan_meta.json` 里：

```
cd policy/pi0
python -c "import
json;print(json.load(open('sampling_plans/beat_block_hammer/demo10_es0_plan
C/plan_meta.json'))['summary']['num_anchors'])"
```

给定 `R_target=35`、`batch_size=32`，repeat-matched 步数：

```
steps_repeat = round(R_target * N / batch_size)
```

6.2 训练（compute-limited：steps=3000/6000）

```
cd policy/pi0
for D in 5 10 20; do
  for ES in 0 1 2; do
    for S in 42 43 44; do
      for STEPS in 3000 6000; do
        XLA_PYTHON_CLIENT_MEM_FRACTION=0.45 uv run scripts/train.py
pi0_base_aloha_robotwin_lora \
          --exp-name beat_block_hammer-
demo_clean-${D}_planC_es${ES}_s${S}_steps${STEPS} \
          --overwrite \
          --seed ${S} \
          --num-train-steps ${STEPS} \
          --budget-aware-schedule \
          --freeze-mode default \
          --plan-aware-lr-scale \
          --sampling-plan-path
sampling_plans/beat_block_hammer/demo${D}_es${ES}_planC
      done
    done
  done
done
```

6.3 评测（clean + randomized）

```
cd policy/pi0
# clean
bash eval.sh beat_block_hammer demo_clean pi0_base_aloha_robotwin_lora \
  beat_block_hammer-demo_clean-10_planC_es0_s42_steps3000 0 0 -1

# randomized
bash eval.sh beat_block_hammer demo_randomized pi0_base_aloha_robotwin_lora
\
  beat_block_hammer-demo_clean-10_planC_es0_s42_steps3000 0 0 -1
```

> `-1`（latest：最新）会自动选当前实验目录下最大的 checkpoint id。

建议你额外评测多个 checkpoint（相当于“事后早停”（early stopping：性能不再提升就停）），避免 5/10-shot 训到最后反而过拟合：

```
cd policy/pi0
for CKPT in 1000 2000 3000 6000; do
  bash eval.sh beat_block_hammer demo_clean pi0_base_aloha_robotwin_lora \
    beat_block_hammer-demo_clean-10_planC_es0_s42_steps6000 0 0 ${CKPT}
done
```

---

7) 快速排查：few-shot 成功率异常低时先看什么

1. `_metrics.json` 里是不是 `exception` 很多?
   - 若是：先修环境/依赖/ckpt 路径；不要把崩溃当成“模型不行”
2. `run_summary.json` 的 `exposure_avg_repeat` 是否极高（比如 >200）？
   - 是：优先增大 anchors（stride=1 + tail_keep=H-1）或改用 repeat-matched steps
3. 是否用了 `freeze-mode strong`?
   - 是：先换回 `default` 对齐 baseline；再把 strong 作为消融项
4. `gripper_event.enabled` 是否为 true?
   - 若你明确认为夹爪是关键：建议打开并把 `q` 调高（0.95→0.97/0.99）抑制小抖动误报

---

8) 扩展到其他任务（place_container_plate / stack_bowls_three 等）

把本文档中的：

- `task_name=beat_block_hammer`（评测第 1 个参数）
- `repo-id=beat_block_hammer-demo_clean-50`（plan 第 1 个参数）
- `out-dir` 和 `exp-name` 的路径前缀

替换成你的目标任务即可，例如：

- `place_container_plate-demo_clean-50`

- stack_bowls_three-demo_clean-50