Future-Seed：一种基于跨层状态回灌的原地约束修复机制（以 RWKV7-CUDA 上的 4×4 数独填空为例）

yanghu819

2026 年 2 月 14 日

摘要

固定记忆的递归语言模型(RNN/SSM 系)在长上下文与全局一致性任务上常出现脆弱性:早期位置的决策无法有效利用序列右侧信息,从而在需要全局约束传播的任务中产生冲突解。本文基于 `future-seed` 的核心思想：将上一层扫描完整段序列后的终止状态作为下一层的初始状态（跨层回灌），从而在深度方向形成“再读一遍序列”的信息通路。我们在单卡 RTX 4090 上，使用 RWKV7 `cuda_wind` 核心实现了一个可复现的 **in-place 数独填空**基准（4×4，2×2 宫），对比 **FUTURE_SEED=0** 与 **FUTURE_SEED=1** 的解题率随挖空数（holes）变化的相变曲线，并探索一个极简的软约束正则项。结果表明：在相同算力与规模下，Future-Seed 能显著推迟相变点，在 holes=12 时将 solve-rate 从 0 提升至约 0.55；加入简单的行/列/宫计数一致性正则可进一步提升 holes=12 的 solve-rate（约 0.48 → 0.59）。这些结果支持了 Future-Seed 作为“全局约束记账/传播通道”的机制解释，并为后续扩展到更真实的原地编辑与结构修复任务提供了明确的实验信号。

1 引言

我们关心的一类任务并非“从头生成文本”，而是**原地（in-place）修复**：在不重写整体结构的前提下，局部填补缺失并满足全局一致性约束。典型例子包括：置换一致性（不重不漏）、括号/JSON 结构合法性、实体/变量命名一致性、以及约束满足问题（如数独）。

对固定记忆的递归模型而言，困难来自强因果扫描：序列左侧的预测很难利用右侧信息，导致在并行填空或一次性解码时出现结构性冲突。Future-Seed 通过跨层状态回灌，在深度方向提供额外的信息通路，使早期位置能间接依赖“未来”上下文。本工作将该直觉落地为一个极小但硬的基准：4×4 数独 in-place 填空，并用 solve-rate/相变曲线给出可复现的对照证据。

2 方法：Future-Seed

设模型由 L 层堆叠而成。对第 l 层（$l \geq 1$），Future-Seed 将上一层在读完整段长度 T 后的终止状态 $s_T^{(l-1)}$，作为本层的初始状态 $s_0^{(l)}$（或其门控版本），从而形成跨层回灌：

$$s_0^{(l)} \leftarrow s_T^{(l-1)}.$$

直观上，这等价于在深度方向对同一序列进行“多次线性扫描”，使得后层在处理早期 token 时，能携带更多来自整段序列的信息摘要。本文实验使用 RWKV7 的 `cuda_wind` 实现以保证吞吐与可复现性。

3 任务：4×4 数独 in-place 填空

3.1 数据生成

我们合成生成合法 4×4 数独解盘（数字 1–4，2×2 宫），并随机挖空 h 个位置作为输入。输入与输出在同一固定长度序列中编码（字符级词表），空位用 mask token 表示，训练损失仅在 mask 位置计算。

3.2 指标

我们报告两类指标：

- **solve-rate**：模型填出的 16 个格子满足行/列/宫约束（每组恰好包含 {1,2,3,4}）。
- **exact**：预测解与合成的隐藏解完全一致的比例（更苛刻，且在存在多解时可能偏低）。

4 实验设置

4.1 实现与硬件

实验在单卡 NVIDIA RTX 4090 上运行，采用 RWKV7 `cuda_wind` 内核。我们使用同一训练脚本，通过环境变量切换任务与超参。

4.2 模型与训练

主要对比为：

- **FS=0**：不启用 Future-Seed。
- **FS=1**：启用 Future-Seed（跨层回灌）。

我们使用 holes 课程学习（训练时 $h \sim U[4, 12]$），并在固定 holes 下进行大样本评估（每点评估 trials=2000）。

4.3 极简软约束正则

为缓解冲突解，我们加入一个简单的软约束：对每行/列/宫，要求“每个数字的概率计数之和接近 1”。实现上对 mask 位置使用模型 softmax 概率，对非 mask（clue）位置使用真值 one-hot，并对 12 个约束组做平方误差平均：

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{cons}.$$

5 结果

5.1 相变曲线：Future-Seed 显著推迟失效点

表 1 给出相同基准下 solve-rate 随 holes 增长的变化。Future-Seed 在 holes=10 与 holes=12 仍保持较高 solve-rate，而 FS=0 很快塌陷至接近 0。

表 1: 4×4 数独 in-place 填空：solve-rate 相变曲线（trials=2000）。FS=0 使用 L4/e128；FS=1 使用 L12/e128。

holes	FS=0 solve	FS=1 solve
4	0.3530	1.0000
6	0.1665	1.0000
8	0.0520	1.0000
10	0.0095	0.9510
12	0.0000	0.5510
14	0.0000	0.0000

5.2 软约束正则：提升 holes=12，但 holes=14 仍困难

表 2 显示：在 holes $\in [4, 14]$ 的课程训练下，加入 $\lambda = 0.1$ 的一致性正则能显著提升 holes=12 的 solve-rate（约 0.48→0.59），但 holes=14 仍然难以解出（solve-rate 约为 0）。

表 2: 一致性正则的影响（FS=1，L12/e128，trials=2000）。

setting	holes=10	holes=12	holes=14
$\lambda = 0.0$	0.9385	0.4795	0.0000
$\lambda = 0.1$	0.9515	0.5855	0.0000

5.3 负结果

我们观察到：

- 仅训练极少参数（`FS_MASK_ONLY=1`）无法学习该任务（solve-rate ≈ 0）。
- 将 mask 固定为 prefix（更强"用未来"压力）在当前设定下训练不稳定，效果很差（solve-rate ≈ 0）。这提示仅靠单次前向的 argmax 填充可能不足，可能需要迭代式推理协议。

6 讨论与后续工作

本工作给出了一个干净、硬指标的证据：Future-Seed 能在 in-place 约束修复任务上显著提升 solve-rate，并推迟相变点。我们认为其核心贡献是提供了一个跨层的"全局记账/约束传播通道"。后续可沿两条路径推进：

- **推理协议**：引入迭代填充（mask-predict / refinement）以进一步提升 holes=14 的成功率。
- **任务扩展**：扩展到 9×9 数独、以及更真实的结构修复（JSON/括号/一致性编辑）与 in-place 置换修复（key-value 排序 + mask）。

复现说明

代码与脚本基于 `git@github.com:yanghu819/future-seed.git`。主要入口为 `rwkv_diff_future_seed.py`，通过环境变量启用 `SUDOKU_TASK=1` 并设置 `FUTURE_SEED`、holes 课程与评估 trials。