LOOPED VIRTUAL CELL

Anonymous authors
Paper under double-blind review

ABSTRACT

Virtual cell perturbation models often collapse a dynamic cellular response into a static endpoint expression profile. We introduce **Looped Virtual Cell**, instantiated as **Looped X-Cell**, which turns endpoint prediction into a learned recurrent rollout over genes. Rather than predict all genes in one pass, Looped X-Cell repeatedly ranks unresolved genes, commits selected expression values, carries the partial endpoint state forward, and uses lookahead supervision so stages preview genes that will be committed shortly afterward. A scorer-latent outer loop can then revisit the staged rollout through recurrent latent state without directly feeding predicted expression values back into the input. Across four Replogle-Nadig cellular contexts, Looped X-Cell improves endpoint prediction over a fine-tuned X-Cell model, scGPT, and a linear baseline under matched held-out perturbation splits. The learned ranks also expose biological structure that is never used as supervision: they recover Reactome pathways across Th17, Replogle-Nadig, and Marson, recover CORUM protein-complex relationships in rank-feature perturbation clusters, and reveal GO-enriched RNA/protein rank gaps in a multimodal Th17 model. Component ablations identify staged global ranking and lookahead supervision as the main drivers of HepG2 endpoint gains, while compute sweeps show that inner-loop resolution improves prediction more consistently than outer-loop recurrence. Together, these results show that learned gene rollout is a structured compute axis for virtual-cell endpoint models and an auditable internal trace for perturbation biology.

1 INTRODUCTION

Virtual cell perturbation models aim to predict how a cell responds to a genetic or chemical perturbation (Bunne et al., 2024). When a perturbation is introduced, its effects propagate through regulatory programs and signaling cascades that result in downstream expression changes (Dixit et al., 2016; Replogle et al., 2022). By the time an endpoint profile is measured, much of that intermediate structure is no longer observed directly.

Although the underlying response is dynamic, most current models are trained and evaluated as static endpoint predictors: given a control cell state and a perturbation, they predict the final perturbed expression profile (Lotfollahi et al., 2023; Hetzel et al., 2022; Roohani et al., 2024; Cui et al., 2024; Adduri et al., 2026; Dong et al., 2026; Wang et al., 2026). This endpoint formulation supports in silico prediction and experimental prioritization, where a generalizing model can predict and rank unmeasured perturbations in new cellular contexts before they are tested experimentally (Roohani et al., 2025; Rubbi et al., 2026; Bereket & Leskovec, 2026). However, it collapses a dynamic process into a single prediction.

This motivates **Looped Virtual Cell**, a framework for turning endpoint prediction into an ordered recurrent computation. We instantiate this idea on top of **X-Cell** (Wang et al., 2026), an endpoint perturbation model that predicts gene-expression responses from a control cell state and perturbation priors, yielding **Looped X-Cell**. Instead of predicting the endpoint profile in one shot, the model learns an order in which to resolve genes. Looped X-Cell scores unresolved genes, commits a subset of their predicted expression values, and carries those commitments forward before scoring the remaining genes again. Repeating this rank-and-commit process across inner-loop stages induces a learned rollout trajectory through the endpoint prediction. An outer loop then revisits the completed rollout through recurrent scorer state to refine the ordering over additional passes. The resulting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ranks and stages expose an inspectable internal trace that can be tested against external pathway and protein-complex annotations. Our main contributions are:

- **A looped modeling framework that exposes virtual cell computation through gene rollout.** Looped X-Cell explicitly models endpoint prediction as a recurrent rank-and-commit rollout over genes, exposing internal ranks and stages while improving endpoint prediction consistently across four Replogle-Nadig contexts (Replogle et al., 2022; Nadig et al., 2025).
- **Inspectable traces that recover biological structure without explicit supervision.** The learned rollout recovers context-specific pathways (Ragueneau et al., 2026) and protein-complex structure (Steinkamp et al., 2025) in Replogle-Nadig, Th17 UPR, and genome-wide Marson cross-donor predictions (Zhu et al., 2025), and can be used to probe RNA–protein relationships in a multimodal Th17 model.
- **New looped compute axes for perturbation prediction.** By varying inner-loop rollout resolution and outer-loop recurrent depth, we probe how the amount and granularity of internal computation affect endpoint prediction and learned biological structure.

2 METHOD

2.1 ENDPOINT PREDICTION AND THE X-CELL PRE-TRAINED MODEL

Given a control-cell gene expression profile $x_c \in \mathbb{R}^G$ and a perturbation p, the goal is to predict the perturbed endpoint expression profile $x^\star \in \mathbb{R}^G$. Standard endpoint models learn a direct map $\hat{x} = f_\theta(x_c, p)$. X-Cell (Wang et al., 2026) is one such endpoint perturbation model: it represents each gene with gene-identity and continuous expression-value encoders, injects perturbation information through cross-attention to perturbation-prior tokens, and decodes endpoint expression for all genes. At inference time, X-Cell can iteratively refine endpoint predictions by randomly remasking subsets of genes, but the remasking schedule is not learned as a perturbation-conditioned order over genes.

Looped X-Cell keeps the X-Cell tokenization, perturbation-prior pathway, endpoint decoder, and fine-tuning losses, but replaces one-shot endpoint decoding with a sequence of partial expression states. At each stage, the model scores unresolved genes, selects a subset to resolve, predicts and commits their expression values, and conditions the next stage on the accumulated commitments. The final committed state is supervised by the measured endpoint expression, while a lookahead loss trains each stage to preview near-future genes before they are committed. Thus, Looped X-Cell learns a rollout order from endpoint and lookahead supervision without observed response trajectories or ground-truth order labels.

2.2 LOOPED X-CELL OVERVIEW

As illustrated in Figure 1, Looped X-Cell introduces two looped compute axes. The **inner loop** constructs an endpoint prediction over K rank-and-commit stages. At each stage, a global scorer ranks unresolved genes and selects the current commit set together with a near-future lookahead bucket. A local scorer orders these genes for a Mamba executor (Lahoti et al., 2026) placed around the X-Cell cross-attention path to perturbation priors. The model then commits only the current-stage predictions into the evolving expression state, while executor state and committed expression values are carried forward to condition later stages.

This separation between lookahead and commitment lets each stage preview genes that will be committed shortly afterward without writing those preview predictions into the current state. The **outer loop**, indexed by T, repeats the staged rollout through recurrent scorer state: each pass can revisit the learned gene ordering through latent recurrence, but endpoint expression is read out only from the final pass.

The same rollout exposes three internal views for analysis: a **stage view**, a **global-rank view**, and a **local-rank view**. Section 2.7 defines these views and uses them to test whether the learned computation recovers biological structure.

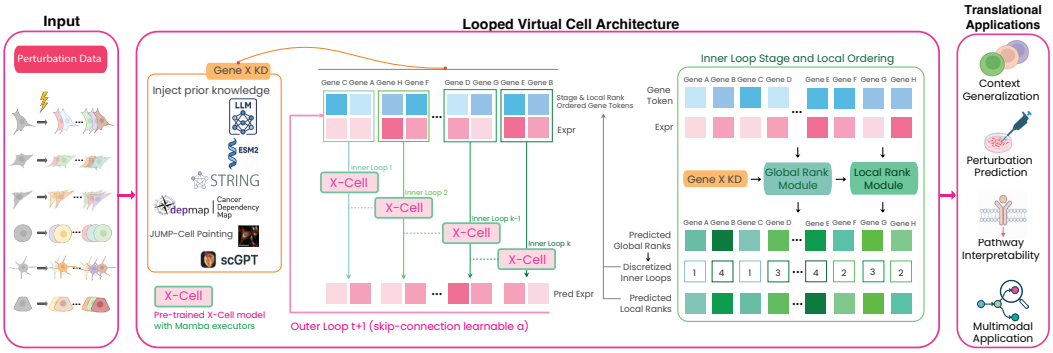

Figure 1: **Looped Virtual Cell overview.** Looped X-Cell predicts perturbation responses with two looped compute axes. The inner-loop staged rollout learns a global gene order, selects commit sets, and builds the endpoint stage by stage. Within each stage, a local scorer and Mamba executor process the selected genes sequentially and carry state across stages. The outer-loop recurrent pass repeats the staged rollout through scorer-latent recurrence and reads out the endpoint on the final pass. The resulting stage, global-rank, and local-rank views provide analysis features for probing the learned rollout.

We use G for the number of genes, K for the number of **inner-loop stages**, and T for the number of **outer-loop passes**. We index genes by $i \in \{1, \ldots, G\}$, inner-loop stages by $k \in \{1, \ldots, K\}$, and outer-loop passes by $t \in \{1, \ldots, T\}$. During outer-loop pass t and inner-loop stage k, the model maintains a partial expression state $x_{t,k}$ and a set of genes committed so far, $A_{t,k}$. After the final stage of the final pass, the endpoint prediction is $\hat{x} = x_{T,K}$ and is supervised against $x^\star$. Endpoint supervision and lookahead supervision on near-future genes train the model to decompose endpoint prediction into staged commitments without requiring ground-truth order labels. Recurrent executor and scorer states carry information across commit boundaries.

2.3 Inner-Loop Ranking and Local Execution

At stage k of outer-loop pass t, Looped X-Cell represents gene i using its identity, current expression value, and commit/perturbation flags:

$$h_{t,k,i} = E_g(i) + E_v(x_{t,k-1,i}) + E_m(m_{t,k,i}), \tag{1}$$

where E_g is the gene embedding, E_v is the continuous value encoder inherited from X-Cell, and E_m encodes stage/perturbation flags when enabled. For cell-set inputs, the corresponding gene features are averaged across cells for scoring; $H_{t,k}$ denotes the resulting gene representations. The global scorer cross-attends these features to perturbation-prior tokens $P(p)$, which may include learned perturbation identity embeddings and external biological priors:

$$s_{t,k} = \text{GlobalScorer}_\theta(H_{t,k}, P(p)) = \text{ScoreHead}_\theta\left(\text{CrossAttn}_\theta(H_{t,k}, P(p))\right) \in \mathbb{R}^G. \tag{2}$$

This **global scorer** provides the stage-level plan at the pseudobulk level: it averages gene features across the input cells and ranks unresolved genes according to the current partial expression state and perturbation context. The resulting ranks are therefore perturbation-conditioned internal features, reflecting the current expression state together with the perturbation priors available to the scorer.

The global plan supplies a local bucket $L_{t,k}$ containing genes selected for current-stage commitment and additional genes considered for lookahead. The formal selection of the commit set is given in Section 2.4. Within the bucket, a **local scorer** uses a summary $\bar{h}_{t,k,L}$ to score each gene for each single cell:

$$\ell_{t,k,i} = \text{LocalScorer}_\phi(h_{t,k,i}, \bar{h}_{t,k,L}) = \left(W_\ell\, \sigma(W_q h_{t,k,i})\right)^\top \bar{h}_{t,k,L}, \qquad i \in L_{t,k}. \tag{3}$$

Here $L_{t,k}$ denotes the bucket and $\ell_{t,k,i}$ the local score of gene i. A softsort operator converts these scores into an ordered sequence for the Mamba executor. In the grouped-sort setting, current-stage genes precede lookahead genes, preserving a current-stage prefix in the executor sequence.

Mamba executor blocks process the ordered sequence around the X-Cell cross-attention operations to perturbation-prior tokens. In the selected Replogle-Nadig model, these executor blocks are placed at backbone indices 2, 5, 8, and 11. Mamba state is carried forward across inner-loop stages within the same outer-loop pass. When lookahead genes are present, only the CLS token and current-stage committed prefix update the carried Mamba cache. Consequently, later stages can use executor state produced for earlier commitments without carrying lookahead-token state forward. Unresolved genes may still contribute their available input features to global scoring and lookahead processing; the restriction applies to committed predictions and the state carried between stages.

2.4 Staged Commitment and Lookahead Learning

The module stack above defines a perturbation-conditioned compute path; staged commitment and lookahead supervision make that path trainable. Given the global scores $s_{t,k}$, a straight-through top-q_k operator selects the current-stage commit set,

$$C_{t,k} = \mathrm{ST}_{\mathrm{Top}K}\left(s_{t,k}, q_k; \{1, \ldots, G\} \setminus A_{t,k-1}\right), \tag{4}$$

using hard top-k selection in the forward pass and a soft relaxation for gradients. This creates discrete inner-loop stages while keeping the stage assignment trainable.

For a selected gene $i \in C_{t,k}$, the stage decoder predicts $\hat{x}_{t,k,i}$. The commit operator updates only selected genes,

$$x_{t,k,i} = \begin{cases} \hat{x}_{t,k,i}, & i \in C_{t,k}, \\ x_{t,k-1,i}, & i \notin C_{t,k}. \end{cases} \tag{5}$$

Thus the inner loop constructs the endpoint progressively rather than predicting all genes simultaneously.

The hard stage assignment alone does not guarantee a meaningful order. Looped X-Cell therefore trains each stage to anticipate genes that will be committed shortly after it. For non-final stage k, the model predicts values for a lookahead bucket $L_{t,k}$ that is larger than the commit set $C_{t,k}$. The lookahead loss is applied only to genes in this bucket that are committed in the next stage,

$$\mathcal{L}_{\mathrm{lookahead}} = \frac{1}{Z} \sum_{t,k} \sum_{i \in L_{t,k} \cap C_{t,k+1}} w_{t,k,i} \left| \hat{x}^{\mathrm{look}}_{t,k,i} - x^{\star}_{i} \right|, \tag{6}$$

where $w_{t,k,i}$ is the straight-through lookahead weight and Z normalizes over active terms. This loss is the main intermediate supervision on the learned ordering. In the selected Looped X-Cell runs, the base X-Cell endpoint and auxiliary losses are applied to the final readout, so the intermediate stages are organized primarily through routing, local ordering, Mamba state carry-forward, and lookahead pressure.

2.5 Outer-Loop Recurrence

The outer loop repeats the full inner-loop rollout. When multiple passes are used, Looped X-Cell carries scorer-latent state between passes rather than feeding a predicted endpoint back as the next expression input. Let $Z_{\mathrm{ctrl}} = E_g + E_v(x_c)$ be the control-anchored scorer latent state. After outer-loop pass t, the inner loop returns an updated scorer latent $\tilde{Z}_t$ over committed genes. The next outer-loop pass scores genes using a control-residual latent skip,

$$Z_{t+1} = Z_{\mathrm{ctrl}} + \alpha\, \tilde{Z}_t, \tag{7}$$

where α is a learnable skip coefficient. This allows the model to revisit its gene ordering while anchoring each pass to the control input. The endpoint prediction and inherited backbone losses are taken from the final pass.

This recurrence choice also separates repeated latent computation from repeated expression-value prediction. In the recurrence comparisons in Appendix C, feeding predicted values forward can accumulate error under additional test-time passes, whereas the control-anchored scorer-latent variant remains stable. We therefore use the outer loop to study whether another pass can reorganize the rollout without requiring expression-value carry-forward.

Table 1: **Main Replogle-Nadig benchmark.** Held-out perturbation prediction across four Replogle-Nadig contexts Replogle et al. (2022); Nadig et al. (2025); Adduri et al. (2026) comparing Linear, scGPT (Cui et al., 2024), X-Cell (Wang et al., 2026), and Looped X-Cell. Looped X-Cell uses four inner-loop stages and one outer-loop pass. Metrics are reported as mean ± s.d. over five training seeds. Lower is better for MAE; higher is better for all other metrics.

Context	Model	MAE ↓	Pearson Δ ↑	Overlap@N ↑	DE Dir. ↑	DE LFC ↑	Centroid Acc. ↑
HepG2	Linear	0.0843 ± 0.0001	0.2292 ± 0.0013	0.1678 ± 0.0005	0.6144 ± 0.0021	0.3545 ± 0.0011	0.7054 ± 0.0026
	scGPT	0.0807 ± 0.0002	0.1872 ± 0.0026	0.2159 ± 0.0098	0.6834 ± 0.0070	0.4469 ± 0.0229	0.6732 ± 0.0033
	X-Cell	0.0669 ± 0.0003	0.6604 ± 0.0031	0.3618 ± 0.0028	0.9433 ± 0.0029	0.8404 ± 0.0071	0.8462 ± 0.0033
	Looped X-Cell	**0.0618** ± 0.0005	**0.6838** ± 0.0033	**0.3739** ± 0.0044	**0.9639** ± 0.0014	**0.8941** ± 0.0034	**0.8512** ± 0.0150
Jurkat	Linear	0.0990 ± 0.0001	0.1877 ± 0.0014	0.1127 ± 0.0013	0.6314 ± 0.0020	0.2841 ± 0.0031	0.7149 ± 0.0023
	scGPT	0.1008 ± 0.0001	0.0492 ± 0.0029	0.1068 ± 0.0015	0.5544 ± 0.0018	0.2365 ± 0.0079	0.5875 ± 0.0025
	X-Cell	0.0862 ± 0.0006	0.6320 ± 0.0020	0.2313 ± 0.0039	0.9406 ± 0.0017	0.7819 ± 0.0049	0.7912 ± 0.0040
	Looped X-Cell	**0.0772** ± 0.0019	**0.6553** ± 0.0047	**0.2440** ± 0.0052	**0.9578** ± 0.0037	**0.8053** ± 0.0078	**0.8021** ± 0.0187
K562	Linear	0.0834 ± 0.0000	0.2365 ± 0.0006	0.1567 ± 0.0005	0.6377 ± 0.0010	0.4008 ± 0.0028	0.8264 ± 0.0027
	scGPT	0.0796 ± 0.0001	0.1291 ± 0.0044	0.1470 ± 0.0054	0.7160 ± 0.0067	0.4521 ± 0.0315	0.5639 ± 0.0030
	X-Cell	0.0696 ± 0.0008	0.6498 ± 0.0031	**0.3160** ± 0.0043	0.9427 ± 0.0067	0.8079 ± 0.0223	0.8164 ± 0.0068
	Looped X-Cell	**0.0631** ± 0.0032	**0.6771** ± 0.0069	0.3145 ± 0.0085	**0.9705** ± 0.0045	**0.9008** ± 0.0019	**0.8347** ± 0.0333
RPE1	Linear	0.1065 ± 0.0001	0.3051 ± 0.0017	0.1985 ± 0.0008	0.6522 ± 0.0017	0.3944 ± 0.0023	0.5555 ± 0.0011
	scGPT	0.0936 ± 0.0002	0.4796 ± 0.0058	0.3211 ± 0.0067	0.8006 ± 0.0020	0.6571 ± 0.0090	0.6619 ± 0.0038
	X-Cell	0.0782 ± 0.0006	0.7435 ± 0.0019	0.4404 ± 0.0018	0.9442 ± 0.0011	0.8527 ± 0.0013	0.8061 ± 0.0021
	Looped X-Cell	**0.0706** ± 0.0022	**0.7656** ± 0.0064	**0.4506** ± 0.0098	**0.9654** ± 0.0034	**0.8844** ± 0.0033	**0.8291** ± 0.0176

2.6 Training Objective

Looped X-Cell inherits the endpoint perturbation losses used by X-Cell fine-tuning and adds rollout-specific supervision. We write the training objective as

$$\mathcal{L} = \mathcal{L}_{\mathrm{XCell}}(\hat{x}, x^{\star}) + \lambda_{\mathrm{lookahead}}\mathcal{L}_{\mathrm{lookahead}} + \lambda_{\mathrm{reg}}\mathcal{L}_{\mathrm{rollout\ reg}}, \tag{8}$$

where $\mathcal{L}_{\mathrm{XCell}}$ denotes the endpoint and backbone auxiliary losses used in X-Cell fine-tuning. Details of model initialization, inherited auxiliary losses, and the rollout-specific sparsity regularizer are provided in Appendix A.4. The main architectural contribution is the learned rollout; the inherited auxiliary terms are held fixed across the primary comparisons unless explicitly ablated.

2.7 Interpreting Learned Rollouts

The rollout exposes three internal views. The **stage view** assigns each gene to the inner-loop stage k in which it is committed. The **global-rank view** orders genes by concatenating stages and their local within-stage orders, producing a full learned gene rollout. The **local-rank view** orders genes only within a stage or bucket. These views are model-derived features, not measured biological time. We use them to test whether the learned computation recovers pathway structure, protein-complex structure, and RNA/protein modality organization.

3 Experiments

We organize the experiments around endpoint prediction, component ablations, biological recovery, and looped computation. We first evaluate held-out perturbation prediction across four Replogle-Nadig contexts (Replogle et al., 2022; Nadig et al., 2025), comparing Looped X-Cell against a linear baseline, scGPT (Cui et al., 2024), and the fine-tuned X-Cell model (Wang et al., 2026). We then ablate model components on the HepG2 benchmark to assess the contributions of staged global ranking, lookahead supervision, local sequence execution, recurrent state carryover, and sparsity. Next, we test whether the learned ranks and stages recover known biological structure, including Reactome pathways (Ragueneau et al., 2026), CORUM protein-complex structure (Steinkamp et al., 2025), and RNA–protein relationships that are not used as supervision. Finally, we vary the number of inner-loop stages K and outer-loop passes T to ask how rollout resolution and recurrent depth affect endpoint accuracy and learned biological organization.

3.1 Held-Out Endpoint Prediction on Replogle-Nadig

The primary endpoint benchmark uses the four Replogle-Nadig cellular contexts (Replogle et al., 2022; Nadig et al., 2025) evaluated by `cell-eval` (Adduri et al., 2026): HepG2, Jurkat, K562,

Table 2: **Inner-loop component ablations on Replogle-Nadig HepG2.** Rows add inner-loop components from the finetuned X-Cell baseline toward the selected Looped X-Cell setting. All Looped X-Cell variants use four inner-loop stages. X-Cell and Looped X-Cell are reported over five training seeds; intermediate component ablations are reported over three training seeds. Metrics are reported as mean ± s.d. Lower is better for MAE; higher is better for all other metrics.

Inner-loop Setting	MAE ↓	Pearson Δ ↑	Overlap@N ↑	DE Dir. ↑	DE LFC ↑	Centroid Acc. ↑
X-Cell	0.0669 ± 0.0003	0.6604 ± 0.0031	0.3618 ± 0.0028	0.9433 ± 0.0029	0.8404 ± 0.0071	0.8462 ± 0.0033
+ Global rank + staged rollout	0.0632 ± 0.0010	0.6852 ± 0.0019	0.3642 ± 0.0066	0.9645 ± 0.0011	0.8948 ± 0.0009	0.8446 ± 0.0078
+ Lookahead supervision	0.0611 ± 0.0004	0.6865 ± 0.0009	0.3817 ± 0.0008	0.9645 ± 0.0007	0.8948 ± 0.0012	0.8519 ± 0.0018
+ Local rank + Mamba executor	0.0619 ± 0.0003	0.6834 ± 0.0023	0.3683 ± 0.0067	0.9637 ± 0.0004	0.8912 ± 0.0034	0.8550 ± 0.0103
+ Recurrent Mamba state carryover	0.0613 ± 0.0005	0.6844 ± 0.0034	0.3782 ± 0.0033	0.9640 ± 0.0011	0.8950 ± 0.0019	0.8479 ± 0.0057
+ Sparsity regularizer	0.0615 ± 0.0010	0.6852 ± 0.0005	0.3820 ± 0.0024	0.9641 ± 0.0008	0.8923 ± 0.0016	0.8513 ± 0.0020
Looped X-Cell	0.0618 ± 0.0005	0.6838 ± 0.0033	0.3739 ± 0.0044	0.9639 ± 0.0014	0.8941 ± 0.0034	0.8512 ± 0.0150

and RPE1. For each test context, models are fine-tuned on training perturbations from the other three contexts and a subset of perturbations from the test context, then evaluated on a matched set of up to 380 held-out perturbation identities from that same test context. Table 1 reports MAE, Pearson Δ, Overlap@N, DE-direction agreement, DE LFC, and centroid accuracy from `cell-eval` package (Adduri et al., 2026). Metrics are averaged over held-out perturbations and reported as mean ± s.d. over five training seeds (Appendix A.6). Under matched splits and inference conventions, Looped X-Cell improves MAE, Pearson Δ, DE-direction agreement, and DE LFC over fine-tuned X-Cell in all four contexts, and matches or improves Overlap@N.

3.2 Ablating Staged Rank-and-Commit Prediction

We next investigate which looped components drive the HepG2 endpoint gains. In the cumulative ablation in Table 2, introducing global gene ranking and staged commitment reduces MAE from 0.0673 to 0.0632, accounting for most of the gain over the X-Cell fine-tuned model. Adding lookahead supervision gives the largest additional lift, reducing MAE further from 0.0632 to 0.0611, and also reorganizes the learned rank structure (Appendix Figure A3). The remaining components do not improve every aggregate endpoint metric, but they support the full recurrent rollout: local Mamba execution defines a within-stage gene order, recurrent Mamba state carries information across commit boundaries, and sparsity regularization encourages concentrated commit decisions.

3.3 Learned Ranks Recover Pathway and Protein-Complex Structure

Endpoint accuracy alone does not show whether the learned rollout carries biological structure. We therefore test the stage, global-rank, and local-rank views against annotations that are never used as labels. Reactome enrichment asks whether genes committed together recover known pathways (Ragueneau et al., 2026). CORUM recovery asks whether perturbations grouped by learned rank features share protein-complex membership (Steinkamp et al., 2025). As a control, we permute the learned gene ranks before rebuilding nearest-neighbor graphs, Leiden clusters, and CORUM scores. Analysis details are provided in Appendix B.2.

Figure 2A shows the Reactome stage probe in Th17 UPR cross-donor prediction. Genes assigned high rank within early stages recover immune-sensing and cytokine/chemokine pathways; intermediate stages recover extracellular and Rho-family signaling; and late stages recover cell-cycle, DNA-repair, and genome-maintenance programs. The same stage-level analysis on Marson D2-to-D3 8-hour cross-donor prediction recovers immune, antigen-presentation, cell-cycle, and RNA-processing programs across 6,997 held-out perturbations (Figure A1). In Replogle-Nadig, learned stages also recover context-specific pathway terms that are not recovered by control-mean or control-variance rank bins computed from the input cells alone (Table A4). Thus, pathway recovery is not explained simply by highly expressed or highly variable genes appearing early in the rollout.

The CORUM probe provides a perturbation-level check on the same learned views (Figure 2B). Across the four Replogle-Nadig contexts, each with at most 380 held-out perturbations, and again in the larger Marson transfer, clustering stage, global-rank, or local-rank features recovers more CORUM complexes and same-complex target pairs than matched feature-permutation controls. Thus, the learned ordering is not only pathway-enriched over genes; when used as a perturbation representation, it also preserves protein-complex relationships among targeted genes.

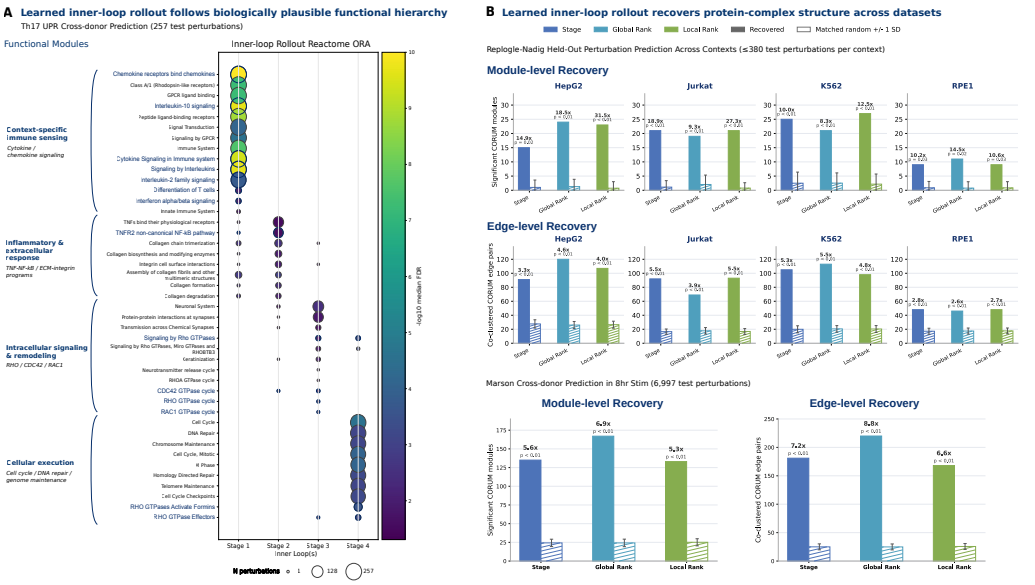

Figure 2: **Learned inner-loop rollout recovers pathway and protein-complex structure across perturbation settings.** (A) Reactome over-representation analysis of Th17 UPR cross-donor predictions across inner-loop stages. Bubble size indicates the number of held-out perturbations assigned to each enriched term, and color indicates $-\log_{10}$ median FDR. Terms are grouped into functional modules that progress from immune sensing and inflammatory response to intracellular remodeling and cellular execution programs. (B) CORUM recovery from learned internal representations. For Replogle-Nadig held-out perturbation prediction across four cell contexts and Marson cross-donor 8-hour stimulation prediction, we cluster perturbations using stage, global-rank, or local-rank features and test whether clusters recover known CORUM protein-complex modules or co-clustered CORUM edge pairs. Solid bars show recovered structure from learned representations; hatched bars show matched random controls with ± 1 s.d.

3.4 RNA+Protein Ranks Expose Cross-Modal Structure

The learned rollout can also be used as a cross-modal feature. The Th17 multiome dataset contains single proteomic measurements captured by CITE-seq sequencing of bound antibody-derived tags (ADTs; Appendix B.1.1), which lets us use the learned rollout as a multimodal rank probe. We train on a joint pool of RNA and protein features and inspect whether matched RNA/protein pairs occupy similar positions in the learned global rank. Matched protein tokens are initialized from the corresponding RNA gene embeddings when available and then allowed to update independently during fine-tuning. This conservative initialization makes the rank gap between a protein and its matched RNA a useful diagnostic: persistent separation suggests that the model is learning modality-specific ordering rather than simply duplicating gene identity.

We analyze matched RNA/protein pairs using the global rollout rank. Let $\Delta r_i = r_i^{\text{protein}} - r_i^{\text{RNA}}$, so positive values indicate that the protein feature is resolved later than its matched RNA feature. In Figure 3A, matched proteins are usually ranked after their RNA counterparts: 75.6% of matched perturbation-feature observations have $\Delta r_i > 0$ (one-sided permutation $p \leq 10^{-4}$, from 10,000 shuffles of matched RNA ranks within each perturbation-level ordering), with median $\Delta r = 613$.

The main caveat is value scale, as RNA and ADT features have different normalization and sparsity. ADT counts are typically less sparse and skew toward higher values than RNA counts, and protein rank is strongly associated with protein abundance in control cells (Spearman $\rho = 0.82$, $p = 9.7 \times 10^{-29}$; Appendix Figure A2A). However, the association of protein abundance in control cells with the RNA/protein rank gap itself is much weaker and not significant at 0.05 (Spearman $\rho = 0.17$, $p = 0.066$; Appendix Figure A2B): the rank gap is not simply tracking which proteins are abundant.

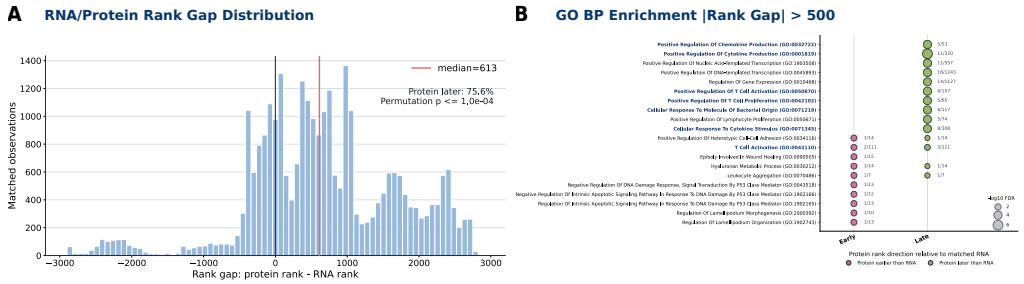

Figure 3: **Matched proteins ranked later than RNA are enriched for immune-response programs.** (A) Distribution of matched RNA/protein global rank gaps across perturbation-level inner-loop orderings, defined as protein rank minus matched RNA rank. Positive values indicate that the protein is ranked later than its RNA counterpart. The red line marks the median rank gap; the permutation p-value is computed from 10,000 shuffles of matched RNA ranks within each perturbation-level ordering. (B) GO Biological Process enrichment for proteins with large rank separation, $|\text{Protein Rank} - \text{RNA Rank}| > 500$. The union of the top 10 enriched terms from the protein-earlier and protein-later sets is shown. Pink denotes proteins ranked earlier than matched RNA, and green denotes proteins ranked later. Bubble size indicates $-\log_{10}$ FDR; adjacent text reports selected-set overlap with each GO term.

Together, these results indicate that the model separates matched RNA and ADT features in its internal rollout rather than treating them as redundant. We next ask whether the separated features correspond to interpretable programs. Figure 3B compares matched pairs with large median rank gaps, $|\text{median}(\Delta r)| > 500$. The ADT-later set is larger and more coherent, recovering cytokine, chemokine, inflammatory, oxidative-stress, and transcriptional-response GO Biological Process terms (Appendix B.3). This pattern is consistent with the measured protein features carrying downstream effector or remodeling information rather than simply duplicating matched RNA signal.

3.5 Looped Compute Axes: Inner-Loop Resolution and Outer-Loop Recurrence

We vary looped computation along the two axes exposed by Looped X-Cell: inner-loop rollout resolution and outer-loop recurrent depth. These axes spend computation in different parts of the looped model. Increasing K decomposes a single endpoint prediction into more rank-and-commit stages, so the model resolves smaller gene subsets at each stage and learns a finer ordering over genes. Increasing T repeats the staged rollout itself, allowing the model to revisit the gene ordering through recurrent scorer state before the final endpoint readout.

Figure 4A compares these two axes on the Replogle-Nadig benchmark, using the seed-42 instance of the 4-inner-loop, 1-outer-loop implementation from Table 1 as the base. In this train-time compute probe, inner-loop resolution gives the clearest endpoint effect where the four-context mean MAE improves in HepG2 as the number of inner-loop stages increases from 2 to 4 to 8, with additional `cell-eval` metrics reported in Appendix Table A1. In contrast, outer-loop recurrence, which is more analogous to looping in sequence models (Dehghani et al., 2018; Giannou et al., 2023), behaves differently. We hypothesize that this reflects the continuous expression space in which virtual-cell predictors operate: directly carrying predicted expression values forward can compound value errors across passes. Consistent with this view, stability is sensitive to the recurrence implementation at training time (Appendix Table A2), and value carry-forward drifts under additional test-time passes (Appendix Table A3). We therefore use a scorer-latent variant that follows a "reason multiple times, read out once" design: each pass revisits the learned gene ordering through a control-anchored latent skip, while expression values are read out only from the final pass. In the HepG2 test-time stability sweep, scorer-latent recurrence remains stable through $T = 16$, whereas value carry-forward drifts under additional test-time passes.

The two-pass scorer-latent model gives only a modest endpoint gain, but the recurrent pass still changes the learned rank trace. In a Th17 two-loop pathway analysis, the second pass is not a duplicate of the first: Reactome terms shift from broad chemokine, GPCR, and DNA-repair programs

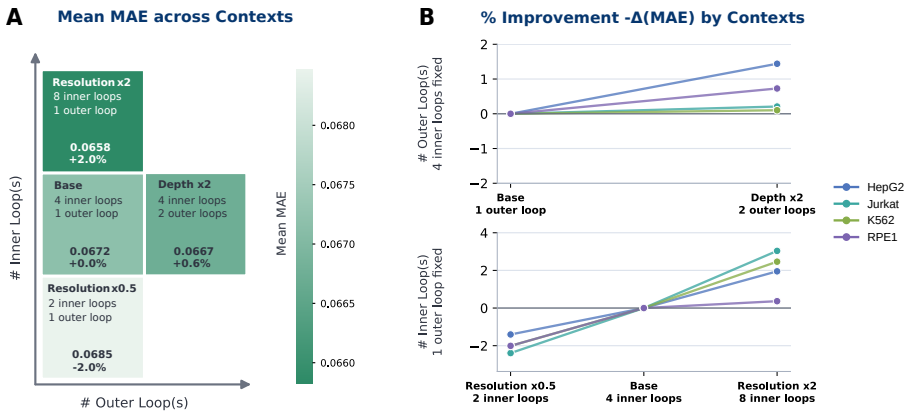

Figure 4: **Inner- and outer-loop compute axes for Looped X-Cell.** (A) Mean MAE across four Replogle-Nadig cellular contexts as a function of the number of inner loops and outer loops. The center setting uses 4 inner loops and 1 outer loop, the seed-42 instance of the architecture reported in Table 1. Moving upward increases the number of inner-loop rollout steps while holding the outer loop fixed; moving right increases the number of recurrent outer-loop passes while holding the inner-loop resolution fixed. Each cell reports the four-context mean MAE and percent improvement relative to the 4-inner-loop, 1-outer-loop base. (B) Context-wise MAE improvement for the two one-dimensional compute sweeps. The outer-loop sweep compares the base model to a 2-outer-loop scorer-latent model with 4 inner loops fixed. The inner-loop sweep compares 2, 4, and 8 inner loops with 1 outer loop fixed. Positive values indicate lower MAE than the matched base setting.

toward more specific interferon and mitotic programs (Appendix Figure A4). Figure 4B shows the endpoint side of this distinction by context: increasing the number of inner-loop stages improves all four Replogle-Nadig settings, whereas adding a second scorer-latent outer loop gives smaller, context-dependent gains. We therefore treat inner-loop resolution as the stronger endpoint-scaling axis in the current experiments, and outer-loop recurrence as a stable recurrent revision axis that can reorganize the internal trace.

4 Discussion and Limitations

Looped X-Cell turns endpoint perturbation prediction into an ordered partial-commitment problem. On the Replogle-Nadig benchmark, this formulation improves endpoint prediction over the fine-tuned X-Cell model across four cellular contexts. The gains are modest, but the component ablations show that they arise primarily from staged global ranking and lookahead supervision rather than from simply adding all rollout modules at once.

The main benefit of this staged computation is that it exposes internal variables that can be tested after training. Reactome enrichment, CORUM recovery, and RNA+protein rank-gap analyses probe complementary views of the same learned ordering and compare them with control-expression or feature-permutation baselines where possible. These analyses show that endpoint-trained ranks are not arbitrary routing artifacts, but they do not establish a causal response path or recover true biological time. The rollout should therefore be interpreted as an auditable internal computation whose pathway, protein-complex, and cross-modal structure can be measured directly.

Outer-loop recurrence was most useful as a stability and trace-reorganization probe. Feeding predicted expression values back into the model can drift under additional test-time passes, while scorer-latent recurrence with a control-anchored skip remains stable. This suggests that repeated computation in virtual-cell regression benefits from a protected latent state rather than direct iteration through the continuous endpoint space. Current Looped X-Cell still uses a deterministic fixed-budget rollout, with the number of stages and outer-loop passes set in advance. Future work could make the rollout adaptive, model uncertainty over plausible gene orders, and ground the learned trace against time-course data, response intermediates, or physical interaction constraints.

5 Reproducibility Statement

The method is specified in Section 2.2–2.7, with model initialization and inherited X-Cell losses described in Appendix A.4. The Replogle-Nadig benchmark uses five training seeds $\{0, 1, 2, 3, 42\}$; HepG2 component ablations use three seeds per intermediate variant. Dataset splits, `cell-eval` aggregation, Reactome/CORUM scoring, RNA+protein matching, and recurrence sweeps are detailed in Appendices A–C.

6 Ethics Statement

This work analyzes previously generated perturbation-response datasets and does not introduce new human-subject recruitment or intervention. The primary risks are scientific rather than clinical: learned rollouts can be useful for interpretation, but they should not be treated as measured biological time or used for individual-level decision making. We report matched controls and limitations to reduce overinterpretation of these internal model traces.

AI Assistance Statement

In this work, we used generative AI tools to provide feedback on research methodology and experiments, implement methods, and interpret results. We did not use generative AI tools to generate synthetic data, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist with proofs, propose or refine hypotheses, clean or reformat datasets, or support qualitative or thematic analysis. We additionally used generative AI tools to assist with software implementation and review, literature discovery, citation formatting, consistency checks, and drafting and editing the manuscript. The authors executed the experiments, checked AI-assisted code and analyses against experiment logs and source artifacts, reviewed all AI-assisted text and claims, and take responsibility for the complete submission and final content of this work.

References

Abhinav K. Adduri, Dhruv Gautam, Beatrice Bevilacqua, Mohsen Naghipourfar, Alishba Imran, Rohan Shah, Noam Teyssier, Rishi Verma, et al. Predicting cellular responses to perturbation across diverse contexts with State. *Cell*, 189(19):5914–5931.e20, 2026. doi: 10.1016/j.cell.2026.07.052.

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

Michael Bereket and Jure Leskovec. Are current ai virtual cell models useful for scientific discovery? *bioRxiv*, pp. 2026–04, 2026.

Charlotte Bunne, Yusuf Roohani, Yanay Rosen, Ankit Gupta, Xikun Zhang, Marcel Roed, Theo Alexandrov, Mohammed AlQuraishi, Patricia Brennan, Daniel B Burkhardt, et al. How to build the virtual cell with artificial intelligence: Priorities and opportunities. *Cell*, 187(25):7045–7063, 2024.

Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. *Nature Methods*, 21(8):1470–1480, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Atray Dixit, Oren Parnas, Biyu Li, Jenny Chen, Charles P Fulco, Livnat Jerby-Arnon, Nemanja D Marjanovic, Danielle Dionne, Tyler Burks, Raktima Raychowdhury, et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *cell*, 167(7):1853–1866, 2016.

Mingze Dong, Abhinav Adduri, Dhruv Gautam, Christopher Carpenter, Rohan Shah, Chiara Ricci-Tam, Yuval Kluger, Dave P Burke, and Yusuf H Roohani. Stack: In-context learning of single-cell biology. *bioRxiv*, 2026.

Zhuoqing Fang, Xinyuan Liu, and Gary Peltz. GSEApy: a comprehensive package for performing gene set enrichment analysis in Python. *Bioinformatics*, 39(1):btac757, 2023. doi: 10.1093/bioinformatics/btac757.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Leon Hetzel, Simon Boehm, Niki Kilbertus, Stephan Günnemann, Fabian Theis, et al. Predicting cellular responses to novel drug perturbations at a single-cell resolution. *Advances in Neural Information Processing Systems*, 35:26711–26722, 2022.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, volume 2026, pp. 86173–86202, 2026.

Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Leon Hetzel, Yuge Ji, Ignacio L Ibarra, Sanjay R Srivatsan, Mohsen Naghipourfar, Riza M Daza, Beth Martin, et al. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular systems biology*, 19(6):MSB202211517, 2023.

Ajay Nadig, Joseph M Replogle, Angela N Pogson, Mukundh Murthy, Steven A McCarroll, Jonathan S Weissman, Elise B Robinson, and Luke J O'Connor. Transcriptome-wide analysis of differential expression in perturbation atlases. *Nature Genetics*, 57(5):1228–1237, 2025.

E. Ragueneau, C. Gong, P. Sinquin, C. Sevilla, D. Beavers, A. Grentner, others, and P. D'Eustachio. The Reactome knowledgebase 2026. *Nucleic Acids Research*, 54(D1):D673–D681, 2026.

Joseph M. Replogle, Reuben A. Saunders, Angela N. Pogson, Jeffrey A. Hussmann, Alexander Lenail, Alina Guna, et al. Mapping information-rich genotype-phenotype landscapes with genome-scale Perturb-seq. *Cell*, 185(14):2559–2575, 2022.

Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with gears. *Nature biotechnology*, 42(6):927–935, 2024.

Yusuf Roohani, Andrew Lee, Qian Huang, Jian Vora, Zachary Steinhart, Kexin Huang, Alexander Marson, Percy Liang, and Jure Leskovec. Biodiscoveryagent: An ai agent for designing genetic perturbation experiments. In *International Conference on Learning Representations*, volume 2025, pp. 26417–26466, 2025.

Andrea Rubbi, Arpit Merchant, Samuel Ogden, Amir Akbarnejad, Pietro Liò, Sattar Vakili, and Mo Lotfollahi. Many needles in a haystack: Active hit discovery for perturbation experiments. *arXiv preprint arXiv:2605.10196*, 2026.

Ralph Steinkamp, George Tsitsiridis, Barbara Brauner, Corinna Montrone, Gisela Fobo, Goar Frishman, others, and Andreas Ruepp. CORUM in 2024: protein complexes as drug targets. *Nucleic Acids Research*, 53(D1):D651–D657, 2025.

Chloe Wang, Mehran Karimzadeh, Neal G. Ravindra, Lexi R. Bounds, Nader Alerasool, Ann C. Huang, Shihao Ma, Daniel R. Gulbranson, et al. X-Cell: Scaling causal perturbation prediction across diverse cellular contexts via diffusion language models. *bioRxiv*, 2026. doi: 10.64898/2026.03.18.712807.

R. Zhu, E. Dann, J. Yan, J. R. Retana, R. Goto, R. C. Guitche, et al. Genome-scale Perturb-seq in primary human $CD4^+$ T cells maps context-specific regulators of T cell programs and human immune traits. *Cell*, 2025.

A Dataset and Evaluation Details

A.1 Replogle–Nadig CRISPR Perturbation Benchmark

We use the Replogle–Nadig Perturb-seq benchmark as the primary endpoint-prediction evaluation (Replogle et al., 2022; Nadig et al., 2025; Adduri et al., 2026). Following the X-Cell benchmark setup (Wang et al., 2026), this dataset provides genetic perturbations across essential-gene knockdown screens and a 2,000-gene highly variable feature space. The prior X-Cell manuscript emphasized held-out HepG2 perturbations; here, we use the same benchmark setup but report all four contexts: HepG2, Jurkat, K562, and RPE1.

For each context, models receive sampled non-targeting control cells and a perturbation identity, then predict perturbed single-cell expression in the target feature space. The train/test split follows the held-out perturbation protocol used for the X-Cell benchmark: perturbation identities in the test set are excluded from fine-tuning, so the task is a few-shot cellular-context adaptation and generalization setting over perturbation identities within each cellular context. Endpoint metrics are computed per held-out perturbation before aggregating over perturbations, contexts, and training seeds.

After filtering, the seed-replicate benchmark evaluates 380 held-out HepG2 target genes, 318 Jurkat target genes, 285 K562 target genes, and 305 RPE1 target genes. For inference, we use a fixed sample of 32 non-targeting control cells per perturbation across datasets. The model-aligned result tensors contain 1,898 evaluated gene features after strict feature intersection with the inference feature set.

A.2 Task Setup

The endpoint benchmark follows the X-Cell fine-tuning protocol: each model receives sampled non-targeting control cells and a perturbation identity, then predicts the single-cell expression distribution for held-out perturbations. The biological-structure analyses use internal Looped X-Cell outputs rather than endpoint values alone. Reactome analyses use stage or rank assignments over genes; CORUM analyses cluster perturbations using learned stage, global-rank, or local-rank features; and RNA+protein analyses compare matched RNA and protein feature ranks within the joint feature space.

A.3 Model Initialization and Fine-Tuning

Looped X-Cell is implemented as a structured fine-tuning extension of X-Cell rather than a model trained from scratch. The pretrained X-Cell backbone follows the X-Cell preprint setup: it is initialized from scGPT encoder weights and continually pretrained on the X-Atlas/Pisces perturbation corpus spanning HCT116, HEK293T, HepG2, and iPSC screens.

For the Replogle-Nadig benchmark, X-Cell and Looped X-Cell are fine-tuned from the same pretrained backbone weights. Looped X-Cell adds the staged scorer, local scorer, Mamba executor, commit mechanism, and recurrence modules during fine-tuning. The reported Replogle-Nadig Looped X-Cell runs use 20 fine-tuning epochs, learning rate 3×10^{-4}, four inner-loop stages, and lookahead multiplier 2.0.

For the Th17 and Marson biological-structure analyses, Looped X-Cell is initialized from the X-Cell checkpoint that was continually pretrained on genome-wide Jurkat resting perturbation data, matching the corresponding X-Cell transfer setup. The Th17 pathway analyses use 20-epoch checkpoints with a linear schedule on the sparsity regularization term. The Marson D2-to-D3 Stim8hr transfer uses the epoch-10 checkpoint from a genome-scale fine-tuning run; this shorter schedule reflects the larger data volume of the Marson screen.

A.4 X-Cell Backbone and Loss Terms

Section 2.2 describes how Looped X-Cell reuses the X-Cell backbone, tokenization, cross-attention pathway, and endpoint decoder; this appendix instead details the specific loss terms inherited from X-Cell and added by the rollout.

The inherited term $\mathcal{L}_{\text{XCell}}$ includes endpoint expression reconstruction and the auxiliary losses used in X-Cell fine-tuning, including masked-value/token losses, MMD-style distribution matching, fold-change consistency, and delta/differential-expression penalties. These inherited terms are held fixed between X-Cell and Looped X-Cell in the main benchmark. The rollout-specific objective adds lookahead supervision and, when enabled, an L1 sparsity penalty on differentiable stage-selection weights. Let $a_{t,k,i} \in [0, 1]$ denote the soft routing weight for gene i at outer-loop pass t and inner-loop stage k before hard commitment. We use

$$\mathcal{L}_{\text{sparse}} = \frac{1}{Z_a} \sum_{t,k,i} |a_{t,k,i}|, \tag{9}$$

where Z_a normalizes over active routing weights. This term discourages diffuse stage assignments. The selected Replogle-Nadig setting uses sparsity weight 0.05; the Th17 pathway runs use a linear schedule for this term.

A.5 Seeded Inference and Test-Split Convention

The seeded Replogle-Nadig benchmark uses manifest-driven inference runs for scGPT, X-Cell, and Looped X-Cell. We restrict replicated comparisons to models that could be run under the same splits, inference conventions, and `cell-eval` version; STATE/STACK are therefore not included because licensing and model-access constraints prevented a matched replicated benchmark. For each context, target perturbations are filtered to the annotated test split, and evaluated observations are restricted to those target genes plus non-targeting controls. Non-targeting controls are pooled from the available cell-line control pool for test evaluation, matching the original Replogle inference behavior and avoiding empty split-specific control pools. No training or parameter updates are performed during inference; checkpoints are loaded from the corresponding seed-specific fine-tuning runs.

The learned-model benchmark uses five training seeds, $\{0, 1, 2, 3, 42\}$. X-Cell seed runs use the non-looped inference script with epoch 20 checkpoints and cumulative inference mode. Looped X-Cell seed runs use the CAS inference script with epoch 20 checkpoints, four inner-loop stages, and lookahead multiplier 2.0. scGPT seed runs use the scGPT inference script with epoch 10 checkpoints and single-pass inference mode. HepG2 ablation runs use the same CAS inference convention but are reported over three training seeds for each intermediate component variant.

A.6 Evaluation Metrics

Endpoint prediction metrics are computed with `cell-eval`, the standardized perturbation-prediction evaluation suite introduced with STATE (Adduri et al., 2026). For each held-out perturbation, predicted and measured perturbed expression distributions are compared relative to matched non-targeting controls. Metrics are averaged over held-out perturbations within each run; benchmark tables report the mean and standard deviation of these run-level aggregates across training seeds. Higher is better for all metrics except MAE.

MAE. measures mean absolute error between predicted and measured expression values.

Pearson Δ. Mean absolute error between predicted and measured expression values.

Overlap@N. Fractional overlap between the predicted and measured top-N differentially expressed genes, with N set by the number of measured differentially expressed genes for that perturbation.

DE direction agreement. Agreement between predicted and measured up- or down-regulation direction over measured differentially expressed genes.

DE LFC. Spearman correlation between predicted and measured log-fold changes over measured differentially expressed genes.

Centroid accuracy. L2 discrimination score from `cell-eval`, which asks whether each predicted profile is closest to its matched measured perturbation centroid.

B Additional Biological Structure Analyses

The main text uses Reactome and CORUM as unsupervised probes of whether learned ranks recover biological structure. We use Replogle-Nadig ranks from the primary endpoint benchmark and two T-cell transfer settings as external biological-probe datasets: Th17 donor transfer and Marson primary T-cell donor transfer. For Replogle-Nadig, we additionally compare context-relevant Reactome terms recovered by learned stages against simple control-expression baselines.

B.1 Biological Validation Datasets

B.1.1 Th17 multiome donor transfer

Primary human Th17 cells from two donors were isolated from peripheral blood mononuclear cells by magnetic separation, activated, and transduced with lentivirus to deliver dCas9-KRAB and the sgRNA library. After seven days of culture, cells from both donors were split into resting and stimulated conditions, the stimulated condition was re-dosed with activation reagent, and cells were harvested 48 hours after reactivation for CITE-seq profiling with 10x Genomics Flex v2.

Gene-expression, guide-capture, and antibody-derived tag (ADT) libraries were processed with Cell Ranger `multi` v10.0.0. Per-sample RNA matrices with guide assignments and cell-level QC flags were loaded with Scanpy; ADT counts were loaded with Muon, matched to RNA by barcode, and concatenated across the two donors and stimulation conditions. The resulting MuData object contains 539,172 cells, 18,135 genes, and 348 ADT features. RNA counts were normalized to counts per 10,000 and transformed as $\log_2(x + 1)$; ADT counts were CLR-transformed within each cell.

RNA-only experiments use stimulated Donor07 and Donor33 samples in a cross-donor transfer setup. The RNA modeling subset unions donor-specific highly variable genes with 917 pre-selected Crohn's disease-relevant genes, yielding 2,639 RNA entries after cleaning and deduplication. The Crohn's gene enrichment broadens the Th17 feature space but is not used as pathway or endpoint supervision.

RNA+protein experiments use the same donor-transfer split but train on a joint RNA+ADT feature pool. The joint feature list appends all 348 ADT features to the 2,639-gene RNA list, giving 2,987 model features. When a protein feature has a matched RNA gene, its token embedding is initialized from the corresponding RNA gene embedding; RNA and protein tokens are then allowed to update independently during fine-tuning. Losses are applied over the joint feature space, so this setting is best interpreted as a multimodal enriched-feature rollout rather than a modality-factorized objective.

The RNA-only and RNA+protein splits use the same cell-level filters. A cell is eligible if it passes guide-assignment and count QC and has a nonempty `gene_target`; non-targeting controls are retained as controls, and targeting perturbations are retained only if at least 25 eligible cells remain for that donor.

B.1.2 Marson D2-to-D3 Stim8hr transfer

We use the Marson primary T-cell Perturb-seq screen as a larger external transfer setting for Reactome and CORUM analysis (Zhu et al., 2025). Perturbations with detectable transcriptomic effects were identified with the binary classifier used by X-Cell (Wang et al., 2026). The analysis highlighted here trains on donor D2 under the 8-hour stimulation condition and evaluates on donor D3 under the same stimulation condition; the current feature space contains 18,130 gene-expression features.

D2 Stim8hr contains 3.03M cells before split filtering and 1.82M cells in the train split. D3 Stim8hr contains 2.48M cells before split filtering and 1.45M cells in the train/evaluation split. Cells are retained when they pass count-based and guide-assignment QC, and targeting perturbations are retained for model fitting and evaluation only when at least 25 eligible cells pass these filters. The reported D3 Stim8hr external-transfer analyses use 6,997 held-out perturbations after QC and minimum-cell filtering, with exported inference and rank artifacts sampling 32 cells per perturbation.

B.2 Reactome and CORUM Analysis Details

Reactome over-representation analysis. For each held-out perturbation and learned inner-loop stage, we take the top-ranked genes assigned to that stage and run a hypergeometric over-representation test against Reactome terms (Ragueneau et al., 2026). The reported manuscript bubble plots use top-500 stage genes unless otherwise indicated by the output directory name. The background universe is taken from the control-cell feature space when available, otherwise from the genes appearing in the rollout. Ensembl IDs are mapped to symbols before enrichment when symbol columns are not already present. We retain Reactome terms with gene-set size between 10 and 500 and at least three overlapping genes; p-values are Benjamini–Hochberg corrected within each perturbation-stage query. Reported bubble plots and tables use FDR ≤ 0.05. Bubble size is the number of held-out perturbations for which a term is recovered in a stage, and color is $-\log_{10}$ median FDR over those perturbations.

CORUM clustering. CORUM analyses use human protein-complex annotations from CORUM 5.3 (Steinkamp et al., 2025). Perturbations are represented by one of three internal views: the stage view encodes the committed stage for each gene, the global-rank view encodes normalized rollout rank, and the local-rank view uses reranked genes within fixed global stage buckets using local-scorer order. For each view, perturbation feature matrices are clustered with a Scanpy nearest-neighbor graph using 10 neighbors, cosine distance, no PCA, and Leiden clustering with random seed 0. Replogle-Nadig four-context analyses use Leiden resolution 1.2. The Marson D3 Stim8hr analysis uses resolution 5.0, because the genome-scale perturbation set is much larger and requires a more granular partition to avoid coarse clusters approaching random recovery ceilings. With these settings, the Replogle-Nadig analyses produce 9–11 clusters per context/view, with mean cluster size 26–38 perturbations. The Marson analysis produces 47–51 clusters across the three internal views, with mean cluster size 137–149 perturbations.

Module recovery. For each perturbation cluster, we test overlap between clustered perturbation targets and each CORUM complex using a hypergeometric test over the perturbation-target universe. Complexes require at least two recovered members, and cluster-complex enrichments are filtered at Benjamini–Hochberg FDR ≤ 0.05. Module-level recovery counts significant co-clustered CORUM member sets, collapsing rows that recover the same overlapping target set even if that set is annotated by multiple CORUM complexes.

Edge recovery and random baselines. For edge-level recovery, we enumerate all unordered pairs of perturbation targets that belong to the same CORUM complex and count how many such pairs fall in the same perturbation cluster. Matched random baselines are generated separately for each internal view by shuffling feature values across gene columns within each perturbation row, rerunning nearest-neighbor graph construction, Leiden clustering, and CORUM scoring. The paper figures use 100 feature permutations with seed 0. Baseline bars show the permutation mean with ± 1 s.d.; empirical p-values use $(1 + \#\{\text{random} \geq \text{observed}\})/(1 + n_{\text{perm}})$, and fold-change scores are computed as observed recovery divided by the matched random mean.

B.3 RNA+Protein Rank-Gap GO Analysis

For the RNA+protein analysis, we first match protein features to RNA genes using the Ensembl gene identifier embedded in the protein feature name. Across the reported Th17 RNA+protein rollout, this gives 112 matched protein features linked to 106 unique RNA gene identifiers and 28,784 perturbation-feature observations across 257 held-out perturbations. For each perturbation and matched pair, we compute the global rank gap

$$\Delta r = r^{\text{protein}} - r^{\text{RNA}},$$

where positive values indicate that the protein feature is ranked later than the matched RNA feature. Figure 3A shows the observation-level distribution of this quantity.

For GO enrichment, we aggregate across perturbations before forming gene sets so that each protein feature contributes at most once. Specifically, for each matched protein feature, we compute the median signed rank gap across held-out perturbations and define directional sets using $|\text{median}(\Delta r)| > 500$. The protein-earlier set contains 7 protein features, which collapse to 6 unique

gene symbols after label normalization. The protein-later set contains 60 protein features, which collapse to 56 unique gene symbols. Protein labels are converted to gene symbols by stripping antibody suffixes and applying a small alias map for panel labels such as `CD62L` → `SELL` and `T-bet` → `TBX21`.

GO Biological Process enrichment is computed with `gseapy.enrichr` (gseapy v1.1.4) using the `GO_Biological_Process_2023` Enrichr library and the Enrichr default background (Ashburner et al., 2000; Fang et al., 2023). We use the Enrichr adjusted p-value as the reported FDR. For the $|\mathrm{median}(\Delta r)| > 500$ sets, 37 protein-earlier terms and 175 protein-later terms pass FDR ≤ 0.05. Figure 3B plots the union of the top 10 GO terms from the protein-earlier and protein-later sets ranked by FDR; bubble size denotes $-\log_{10}$ FDR and adjacent text reports the selected-set overlap with each GO term. Because this analysis uses the Enrichr default background rather than a custom antibody-panel background, we interpret it as an exploratory summary of directional RNA/protein rank separation rather than a calibrated panel-specific over-representation test.

C Technical Ablations and Recurrence Comparisons

This appendix separates the looped compute-axis analyses into inner-loop resolution, outer-loop implementation choices, and test-time recurrence stability. These comparisons use the HepG2 setting to isolate how the staged rollout behaves when compute is added within a pass versus through repeated outer-loop passes. We distinguish *train-time implementation probing* from *test-time scaling*. In train-time implementation probing, each outer-loop variant is trained and evaluated with the same number of outer-loop passes, so the comparison asks which recurrence mechanism is learnable under matched training and evaluation depth. In test-time scaling, a checkpoint trained at $T = 2$ is evaluated with larger inference-time T without further training, so the comparison asks whether additional recurrent passes remain stable or accumulate drift.

Table A1 reports the inner-loop resolution comparison, which asks whether the number of learned gene-commit stages matters when the model is otherwise a single outer-loop predictor. Increasing K improves most endpoint metrics through $K = 8$, suggesting that the benefit is not only recurrence but also a finer decomposition of the endpoint prediction.

Table A1: **HepG2 inner-loop resolution comparison.** All models use one outer loop ($T = 1$) and vary the number of inner-loop stages (K), matching the inner-loop axis in Figure 4. Metrics are computed on the same HepG2 held-out perturbation split. Lower is better for MAE; higher is better for all other metrics.

Variant	K	MAE ↓	Pearson Δ ↑	Overlap@N ↑	DE Dir. ↑	DE LFC ↑	Centroid Acc. ↑
1-outer-loop baseline	2	0.0626	0.6783	0.3831	0.9616	0.8914	0.8505
	4	0.0618	0.6840	0.3754	0.9638	0.8915	**0.8617**
	8	**0.0605**	**0.6860**	**0.3872**	**0.9642**	**0.8945**	0.8428

Table A2 reports the outer-loop implementation comparison, which asks what state should be carried from one full rollout pass to the next. The variants differ only in the object passed from outer-loop pass 1 into pass 2. *Transformer latent* carries the backbone hidden representation produced by the first pass into the second pass. *Scorer latent* carries the latent state used by the global stage scorer, so recurrence changes the next pass's gene-ranking computation without directly overwriting expression values. *Value carry-forward* instead uses the predicted expression state from the first pass as the input expression state for the second pass, closest to iterated endpoint refinement. The skip variants mix the carried state with the fresh control-anchored state using a learnable coefficient α, so recurrence is residual rather than a full replacement of the next-pass state. Value carry-forward gives the lowest MAE among non-skip variants but degrades several differential-expression metrics. Scorer-latent recurrence better preserves Pearson delta, Overlap@N, DE direction, and DE LFC, motivating the scorer-latent + skip setting used for the main compute-axis analysis.

Table A3 reports the test-time recurrence comparison, which fixes trained two-outer-loop checkpoints and repeatedly applies additional outer-loop passes at inference time. This separates endpoint performance at the trained depth from stability under extra recurrence. Value carry-forward accumulates error as T increases, while scorer-latent recurrence remains nearly invariant through $T = 16$.

Table A2: **Outer-loop implementation comparison on HepG2.** All variants use 4 inner-loop stages ($K = 4$). The single-pass row is the no-recurrence baseline; all other variants use 2 outer loops. The underlined T indicates the number of outer loops used for both training and evaluation of each model. Rows compare which state is passed from the first outer loop to the second, including transformer latent, scorer latent, or predicted expression values (value carry-forward). The final block adds a learnable skip coefficient α. We use the scorer-latent + skip (α) setting for the compute-axis analysis in Figure 4.

Outer-loop Variant	T	MAE ↓	Pearson Δ ↑	Overlap@N ↑	DE Dir. ↑	DE LFC ↑	Centroid Acc. ↑
1-outer-loop baseline	1	0.0618	0.6840	0.3754	0.9638	0.8915	0.8617
Transformer latent	2	0.0640	**0.6760**	0.3589	0.9604	0.8888	0.8539
Scorer latent	2	0.0615	0.6730	**0.3637**	**0.9604**	**0.8926**	0.8455
Value carry-forward	2	**0.0600**	0.6665	0.3607	0.9558	0.8869	**0.8604**
Value carry-forward + skip (α)	2	**0.0601**	0.6644	0.3589	0.9552	0.8875	**0.8598**
Scorer latent + skip (α)	2	0.0609	**0.6847**	**0.3732**	**0.9636**	**0.8920**	0.8551

Table A3: **HepG2 test-time outer-loop stability.** Models are trained with 2 outer loops (underlined) and evaluated with $T = 2, 4, 8, 16$ outer loops at inference time. The value carry-forward and scorer-latent variants correspond to settings in Table A2. Lower is better for MAE; higher is better for all other metrics.

Outer-loop Variant	T	MAE ↓	Pearson Δ ↑	Overlap@N ↑	DE Dir. ↑	DE LFC ↑	Centroid Acc. ↑
Value carry-forward + skip (α)	2	0.0601	0.6644	0.3589	0.9552	0.8875	0.8598
	4	0.0636	0.6450	0.3538	0.9530	0.8786	0.8668
	8	0.0703	0.5925	0.3416	0.9381	0.8638	0.8615
	16	0.0901	0.4675	0.3178	0.8996	0.8273	0.8279
Scorer latent + skip (α)	2	0.0609	0.6847	0.3732	0.9636	0.8920	0.8551
	4	0.0609	0.6840	0.3728	0.9623	0.8908	0.8547
	8	0.0609	0.6840	0.3733	0.9624	0.8910	0.8550
	16	0.0609	0.6841	0.3727	0.9622	0.8905	0.8550

Table A4: **Context-relevant Reactome terms recovered by learned Replogle-Nadig stages and control-expression baselines.** Rows are restricted to curated context-relevant Reactome terms recovered at FDR ≤ 0.05 by at least one analysis. Control baselines rank genes by control-cell pseudobulk mean expression or control-cell variance, split each ranking into four equal-size control-rank bins, and run the same Reactome over-representation analysis. Entries indicate the learned inner-loop stage ($S1$–$S4$) or matched control-rank bin ($C1$–$C4$) in which the term is recovered; – indicates that the term is not recovered by that source.

Context	Reactome term	Learned rollout	Control mean	Control variance
HepG2	Metabolism of amino acids and derivatives	S4	C1	–
	Metabolism of proteins	S4	–	–
	Cholesterol biosynthesis	S2	–	C1
	Metabolism of steroids	S2	–	C1
	Activation of gene expression by SREBF (SREBP)	–	–	C1
	Regulation of cholesterol biosynthesis by SREBP (SREBF)	–	–	C1
Jurkat	Innate Immune System	S1	–	–
	Immune System	S1	–	–
	Cytokine Signaling in Immune system	S1	–	–
	MHC class II antigen presentation	S4	–	–
	Adaptive Immune System	S4	–	–
	Signaling by Interleukins	S4	–	–
	Interleukin-12 signaling	S4	–	–
	Class I MHC mediated antigen processing & presentation	S4	–	–
K562	Cell Cycle	S4	–	C1
RPE1	Cell Cycle, Mitotic	S1	–	C1
	Cell Cycle	S1	–	C1
	M Phase	S1	–	C1
	DNA Repair	S1	–	–

The Replogle context table is intended as a specificity check rather than a comprehensive pathway catalog. Jurkat immune terms are recovered by the learned rollout but not by the two control-expression baselines shown here, while several HepG2 cholesterol and sterol terms are also recovered by control variance, indicating that some context terms are partly explained by baseline expression structure.

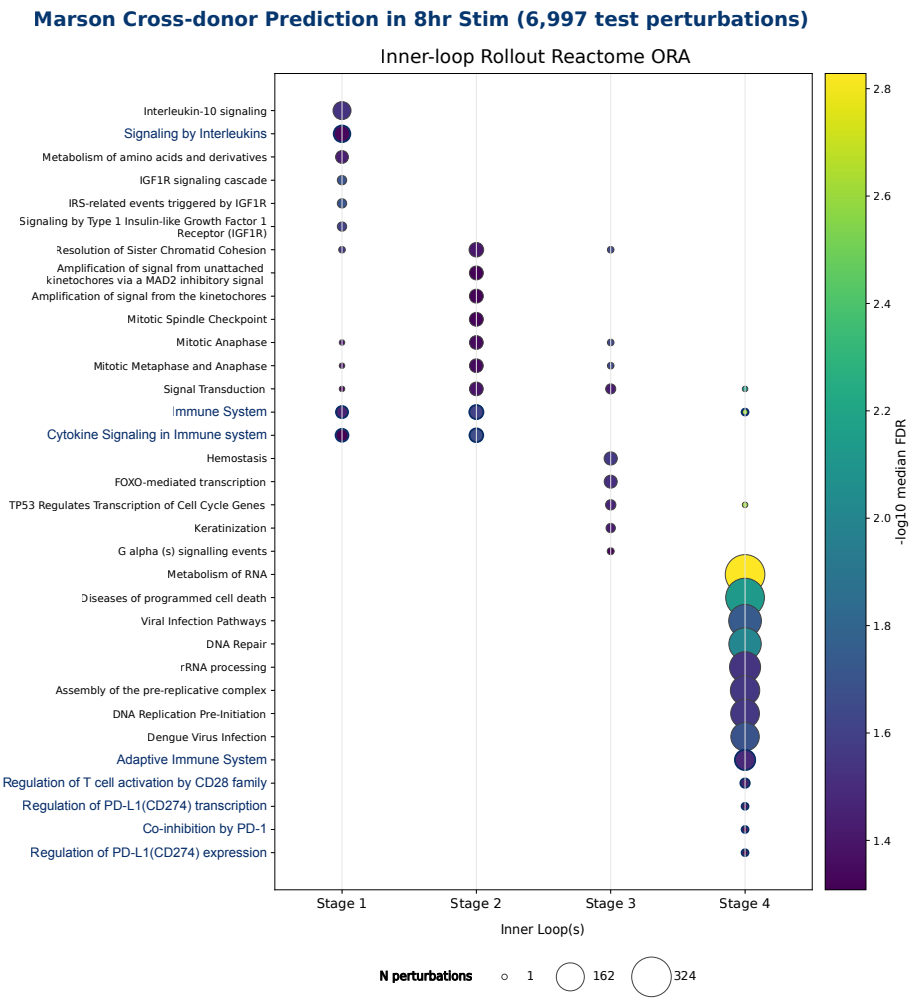

Figure A1: **Reactome enrichment across learned inner-loop stages in Marson genome-scale prediction.** As an external-dataset check, we repeat the Reactome stage-enrichment analysis on Marson D2-to-D3 8-hour stimulation transfer with 6,997 held-out perturbations. Bubbles show Reactome terms recovered at FDR ≤ 0.05 from genes within each learned inner-loop stage; bubble size denotes the number of perturbations supporting the term, and color denotes $-\log_{10}$ median FDR. The rollout recovers immune signaling, antigen-presentation, cell-cycle, RNA-processing, and adaptive immune programs across stages.

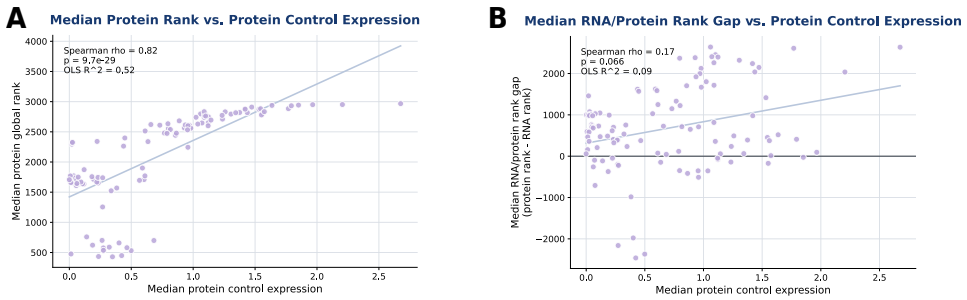

Figure A2: **Protein abundance is associated with protein rank but weakly associated with RNA/protein rank gap.** For each matched RNA/protein pair, we compare median protein control expression against learned rank summaries across perturbation-level inner-loop orderings. (A) Median protein control expression is strongly associated with median protein global rank, indicating that learned protein ranks capture abundance-related structure within the protein modality. (B) In contrast, median RNA/protein rank gap, defined as protein rank minus RNA rank, shows only a weak association with protein control expression. This diagnostic indicates that cross-modal rank separation is not fully explained by protein abundance.

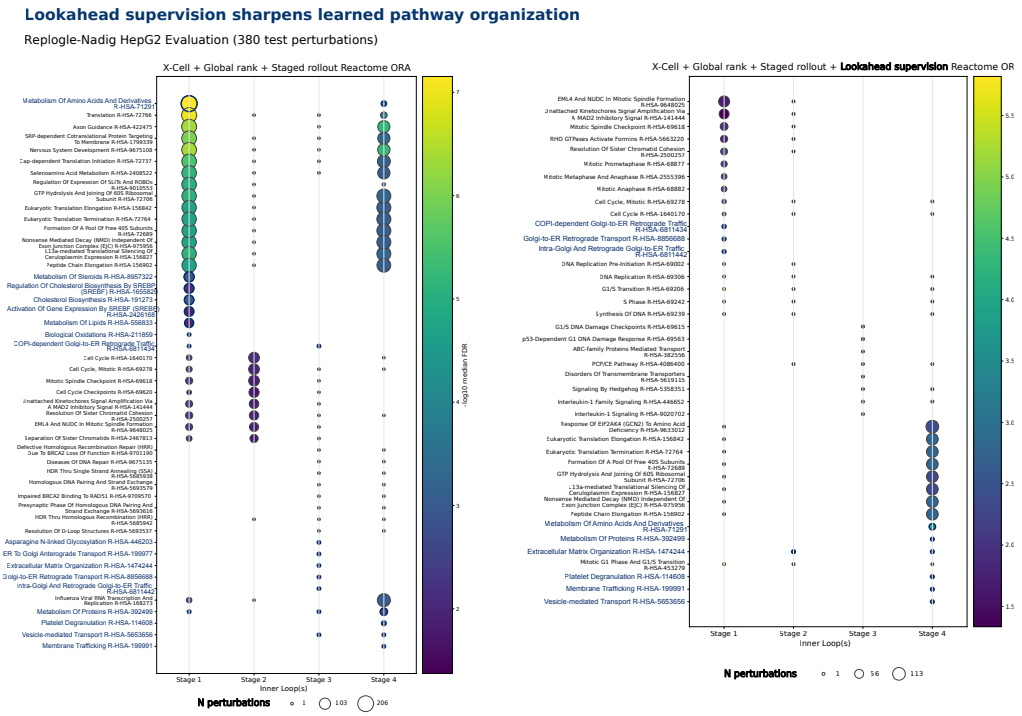

Figure A3: **Lookahead supervision sharpens learned pathway organization.** Reactome over-representation analysis of top-ranked genes within learned inner-loop stages for Replogle-Nadig HepG2 held-out perturbations ($n = 380$). The panels compare the global-rank + staged-rollout ablation with and without lookahead supervision, corresponding to rows in Table 2. Bubble size denotes the number of supporting perturbations, and color denotes $-\log_{10}$ median FDR. Blue labels and outlines mark curated HepG2-relevant terms.

The two-pass Th17 pathway probe uses the RNA-only Th17 UPR cross-donor setting, not the joint RNA+protein model. It evaluates an epoch-10 warm-started checkpoint with two outer loops, scorer-latent control-residual recurrence, default skip coefficient initialization, and no outer-loop normalization. Reactome enrichment is computed with the same top-ranked-gene and FDR convention described in Appendix B.

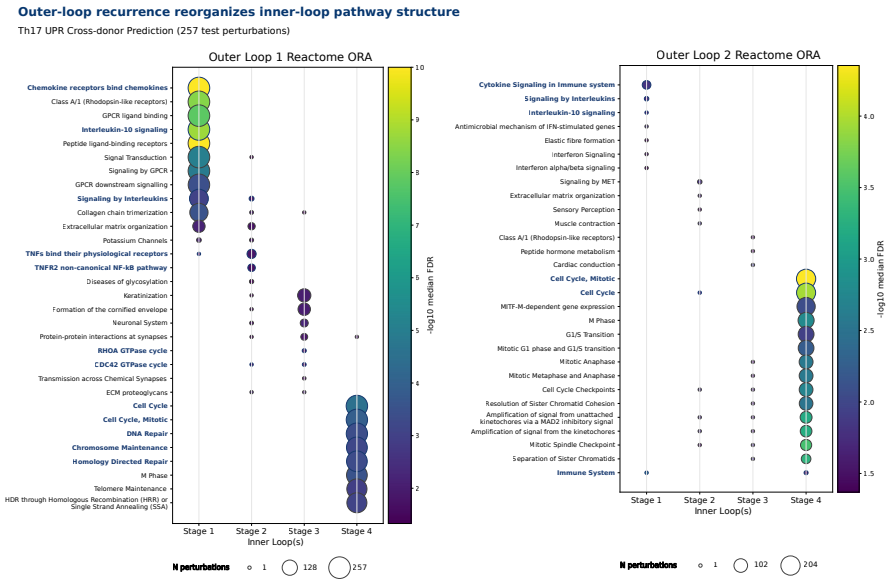

Figure A4: **Outer-loop recurrence reorganizes inner-loop pathway structure.** Reactome enrichment of top-ranked genes within each learned inner-loop stage for a two-outer-loop Th17 UPR cross-donor model. Panels show the first and second outer-loop passes. Bubble size denotes the number of held-out perturbations supporting each term, and color denotes $-\log_{10}$ median FDR. Although endpoint prediction does not substantially improve in this setting, the recurrent pass is not simply a duplicate of the first ordering. The first pass recovers broad pathway blocks, with chemokine/GPCR/interleukin programs enriched early, remodeling terms in intermediate stages, and broad cell-cycle/DNA-repair programs late. The second pass preserves this early-to-late organization while resolving more specific programs, including interferon-related terms early and mitotic G1/S transition, anaphase, spindle-checkpoint, and sister-chromatid programs late.