Predicting cellular responses to perturbation across diverse contexts with STATE

Anonymous Author(s)
Affiliation
Address
email

Abstract

Cellular responses to perturbations are a cornerstone for understanding biological mechanisms and selecting drug targets. While machine learning models offer tremendous potential for predicting perturbation effects, they currently struggle to generalize to unobserved cellular contexts. Here, we introduce STATE, a transformer model that predicts perturbation effects while accounting for cellular heterogeneity within and across experiments. STATE predicts perturbation effects across sets of cells and is trained using gene expression data from over 100 million perturbed cells. STATE improved discrimination of effects on large datasets by more than 30% and identified differentially expressed genes across genetic, signaling and chemical perturbations with significantly improved accuracy. Using its cell embedding trained on observational data from 167 million cells, STATE identified strong perturbations in novel cellular contexts where no perturbations were observed during training. Overall, the performance and flexibility of STATE sets the stage for scaling the development of virtual cell models. STATE training code is available: [MASK], and all model weights and datasets are available to download: [MASK].

1. Introduction

Targeted therapeutic discovery relies on accurately predicting the impact of cellular perturbations. By selectively disrupting specific components of cellular systems, scientists can identify causal drivers of phenotypes, an essential step in both target identification and drug development. Recent advances in functional genomics now enable large-scale screening in specific cellular contexts, through approaches like pairing pooled CRISPR perturbations [1–6]. Improving our ability to generalize perturbation response predictions from these data to diverse biological contexts would greatly accelerate causal target discovery, and help deepen our understanding of cellular function and disease.

A range of computational approaches have been developed to tackle this problem [7–13]. However, despite the rapid growth of perturbation datasets in size and scope, proportional gains in predictive capabilities have not been achieved [14–18]. A significant challenge in perturbation prediction is that single-cell RNA sequencing measurements destroy the cell, preventing observation of their pre-perturbation states and accurate inference of each cell's specific perturbation response. Some approaches assume that within-population heterogeneity is negligible compared to perturbation effects and simply map perturbed cells to randomly selected unperturbed cells with shared covariates [10–12]. These approaches often fail to generalize when perturbation effects are more subtle, where heterogeneity in the unperturbed population may even exceed the perturbation signal. Other models treat cell populations as distributions, learning data-generating distributions or explicitly disentangling labeled and unlabeled sources of variation [8, 19–26]. However, in practice, these models often fail to outperform methods that do not explicitly model distributional structure [14]. Optimal transport-based methods that map unperturbed to perturbed populations have also been proposed, but their applicability has been limited by assumptions and poor scalability [9, 27–29].

To overcome these challenges, we introduce STATE, a flexible and expressive architecture for modeling cellular heterogeneity and perturbation effects within and across diverse datasets (**Fig. 1**). STATE is a multi-

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

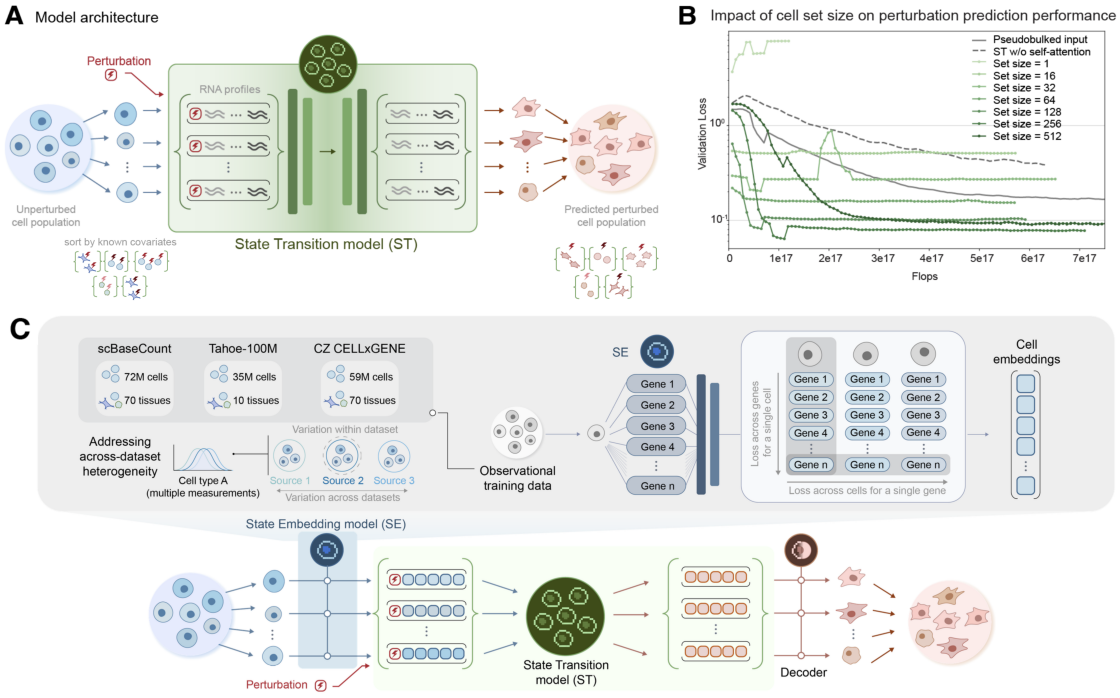

Figure 1: State is a multi-scale machine learning architecture that operates across genes, individual cells, and cell populations. (A) The State Transition model (ST) learns perturbation effects by training on sets of perturbed and unperturbed cell populations grouped by shared covariates (e.g., perturbation type, cell line, or batch). **(B)** Increasing the size of cell sets improves validation loss up to an optimal point, with best performance on the Tahoe-100M dataset achieved when covariate-matched groups are chunked into sets of 256 cells. The full ST model significantly outperforms a pseudobulk model (State w/ mean-pooling instead of self-attention) and a single-cell variant (State with set size = 1). Removing the self-attention mechanism (State w/o self-attention) substantially degrades performance. **(C)** The State Embedding model (SE) is an encoder-decoder model trained on large scale observational single-cell transcriptomics data. The transformer encoder constructs a dense embedding of the cell, and the MLP decoder reconstructs gene expression from the embedding. ST can operate directly on gene expression profiles or on cell representations from SE. When trained with SE, a separate MLP decodes from predicted embeddings to perturbed gene expression.

scale model with two complementary modules: a State Transition model (ST) and a State Embedding model (SE). ST is a transformer that uses self-attention to model perturbation-induced transformations across sets of cells, where each cell is represented either by its raw gene expression profile or a learned embedding (**Fig. 1A**). By leveraging self-attention over sets of cells, ST can flexibly capture biological heterogeneity, and deal with the destructive nature of the assay without relying on explicit distributional assumptions (**Fig. 1B**). SE is pretrained to generate cell embeddings by learning gene expression variation between cells across diverse datasets [30–32], yielding representations that are more robust to technical variation [11, 12, 33–40] and optimized for detecting perturbation effects. We further show that pretraining on perturbations in the SE latent space improves performance in novel contexts with less data, presenting a scalable approach for learning perturbation effects that transfer across datasets and experimental settings (**Fig. 1C**).

The multi-scale architecture of State enables it to leverage both 167 million cells of observational data to train SE and over 100 million cells of perturbation data to train ST. We evaluate State on several large-scale datasets, including drug-based perturbations [30, 41], cytokine signaling perturbations [42], and genome-scale genetic perturbations [4, 6, 43–45]. Across all metrics and data scales spanning multiple orders of magnitude, State consistently outperforms both naive and state-of-the-art models. Together, SE and ST enable State to generalize across many datasets and contexts, improving transferability of perturbation-response modeling.

2. Results

2.1. Building the State Transition model to predict perturbation effects on sets of cells.

STATE leverages (i) at the molecular level, embeddings that represent individual genes across experiments and species; (ii) at the cellular level, embeddings that capture the transcriptomic state of each individual cell, represented either as the cell's log-normalized transcriptome or as embeddings generated by the State Embedding model (SE); and (iii) at the population level, the State Transition model (ST) learns perturbation effects across sets of cells.

The core motivation for ST is to model cellular heterogeneity beyond known covariates, such as cell type and perturbation label, to improve perturbation response prediction. For each covariate-matched perturbed group, ST constructs non-disjoint cell sets of fixed size, which serve as input during training and are paired with unperturbed control cell sets of equal size and matched covariates. Conditioned on the perturbation, ST uses a transformer to perform repeated bidirectional self-attention and feed-forward operations across control cell sets. This enables ST to model the residual, unannotated heterogeneity within the input cell set while predicting downstream transcriptomic responses to perturbation. Empirical results show that increasing cell set size, up to a threshold, achieves much lower validation loss compared to losses on individual cells, and that removing the self-attention degrades performance (**Fig. 1B**).

2.2. STATE improves context generalization of perturbation effects.

We first evaluate State as a set-based transformer operating on cells represented by the expression of 2,000 highly variable genes (ST+HVG). We benchmarked performance against several baselines, including a simple linear model [46], two autoencoder based models CPA [47] and scVI [25], and a single cell foundation model scGPT [11]. We also included two naive baselines that explain a significant portion of observed variance in cell-type generalization: a context mean baseline that predicts the average expression in the test context [48], and the perturbation mean baseline that predicts the average perturbation effect, across training contexts, as a delta to the test basal expression (**Fig. 2A**).

We tested models on chemical perturbations from the *Tahoe-100M* dataset [30], cytokine signaling perturbations from Parse Biosciences [42], and genetic perturbation data from [4] and [43]. Models were evaluated on several metrics in a few-shot context generalization task. Each test cell context contributes 30% of its perturbations to training; the remainder are held out in a test set (**Fig. 2B**). On a variant of the perturbation discrimination score adapted from [14], STATE achieved an absolute improvement of 54% and 29% on the Tahoe-100M and PBMC datasets respectively (**Fig. 2C**), and matches the next best performing baseline on genetic perturbations. We also computed the direct overlap between model predicted DEGs and observed DEGs. Here, STATE is twice as good as the next best baseline on *Tahoe-100M*, 43% better on the Parse-PBMC dataset (**Fig. 2C**), and the second best model on the genetic perturbations. Lastly, we measured accuracy of models in predicting overall perturbation effects by counting the total number of DE genes predicted for each perturbation. STATE accurately ranked perturbations by their relative effect sizes, achieving Spearman correlations 53% higher on Parse-PBMC and 22% higher than baselines on Replogle-Nadig, and 70% higher on *Tahoe-100M* approaching an absolute correlation of 0.8 (**Fig. 2C**).

Notably, STATE confers the biggest benefit as the data scale is increased. On genetic perturbations, where effect sizes are often weaker and less data is available, STATE achieves good performance, often matching or exceeding the next best baseline. On signaling perturbations and drug perturbations, which represent one and two more orders of magnitude of data, respectively, the performance of STATE is significantly better than other methods. This suggests that current perturbations models are in a data-poor regime [49, 50], and shows that the STATE architecture is better able to leverage data scale compared to other models.

2.3. A shared cell embedding (SE) enables transfer across datasets.

Virtual models of cell state should be able to learn cell regulatory information from one dataset and transfer it effectively to other datasets [51]. However, gene expression counts are subject to context-specific variability (e.g., technical noise, sequencing depth, or experimental platform), and thus, do not always generalize well across studies. To address this, we developed the State Embedding model (SE) to unify cell representation across datasets and experiments. Architecturally, SE consists of a transformer encoder that constructs

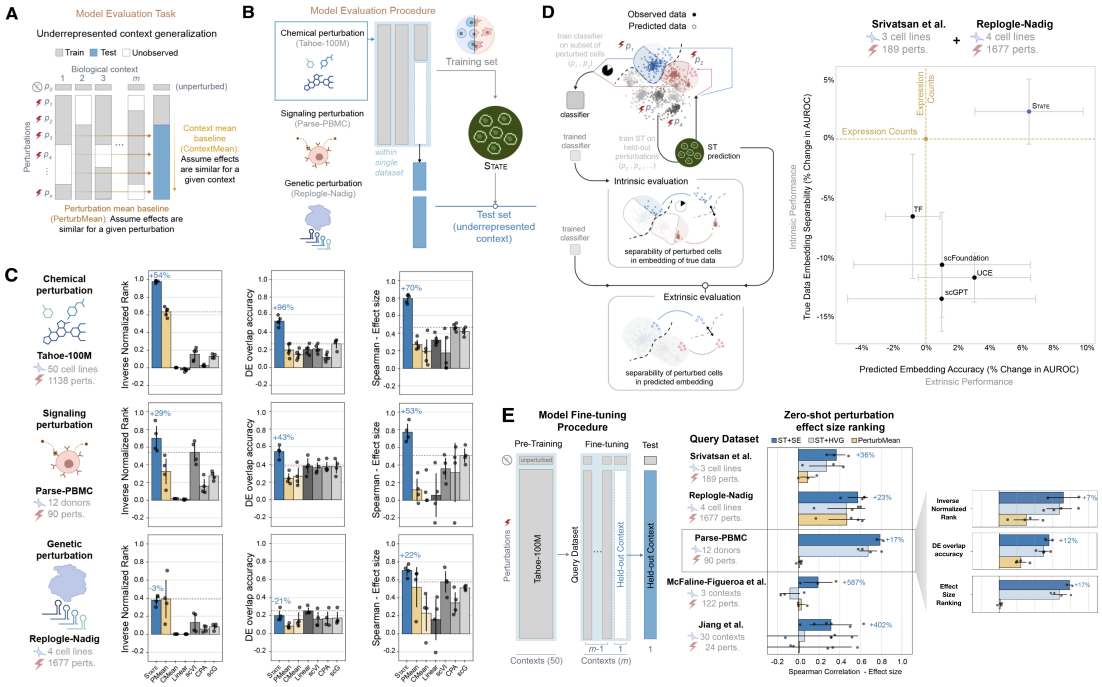

Figure 2: State improves perturbation prediction in context generalization, leverages data scale, and enables cross-dataset transfer learning. (A) Underrepresented context generalization task. Models were trained on perturbation data from one or more cell contexts and evaluated on their ability to predict the effects of the same perturbations in a largely held-out and underrepresented target context. **(B)** Models were trained and evaluated on chemical, signaling, and genetic perturbation datasets [4, 30, 42, 43]. Training and test contexts were drawn from one dataset at a time. **(C)** State performance compared to other models on perturbation discrimination [14], overlap in predicted vs true DEGs, and Spearman correlation of perturbations by effect size (number of predicted DEGs). **(D)** Embedding quality is evaluated using both intrinsic and extrinsic metrics: intrinsic performance reflects classification accuracy of perturbed cells in the embedding generated using observed data; extrinsic performance measures the classification accuracy over perturbed embeddings predicted by ST trained on the cell embeddings. **(E)** Zero-shot performance in ranking perturbations by overall effect size in previously unseen cell contexts over 5 query datasets, after pre-training on *Tahoe-100M*.

a dense representation of a cell from its gene expression values, and an MLP decoder that reconstructs expression from the dense embedding. When used with ST, the embedding enables a smoother landscape over cell states, learned using a vast repository of observational single-cell data.

To probe the information content in SE vs other cell foundation models, we trained shallow MLP classifiers to predict perturbation labels for ground truth and model predicted cells (**Fig. 2D**). Along both axes, State embeddings improves significantly over other models, beating even dataset-specific representations of cells using highly-varying genes [11, 12, 33, 34, 37]. Lastly, to evaluate the transfer learning capability enabled by training perturbation models in SE latent space, we pretrained ST+SE models on *Tahoe-100M* and fine-tuned them on several smaller datasets, spanning various modalities. Across all tested datasets, ST+SE enabled better transfer than ST+HVG and outperforms other baseline methods (**Fig. 2E**).

3. Discussion

State presents a scalable approach for learning a foundation model of cell state and behavior across diverse cellular contexts and experimental conditions. ST uniquely learns perturbation effects across cell populations, thereby capturing residual heterogeneity not explained by known experimental or biological covariates, and SE better enables ST to learn perturbation effects across datasets. These capabilities position State as one step towards realizing the broader vision of a virtual cell capable of exploring the space of possible cell states [51, 52] and guiding autonomous systems, such as AI agents, for experimental design [53–56].

4. References

1. 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.

2. Paul Datlinger, André F Rendeiro, Christian Schmidl, Thomas Krausgruber, Peter Traxler, Johanna Klughammer, Linda C Schuster, Amelie Kuchler, Donat Alpar, and Christoph Bock. Pooled crispr screening with single-cell transcriptome readout. *Nature methods*, 14(3):297–301, 2017.

3. Laralynne Przybyla and Luke A Gilbert. A new era in functional genomics screens. *Nature Reviews Genetics*, 23(2):89–103, 2022.

4. Joseph M Replogle, Reuben A Saunders, Angela N Pogson, Jeffrey A Hussmann, Alexander Lenail, Alina Guna, Lauren Mascibroda, Eric J Wagner, Karen Adelman, Gila Lithwick-Yanai, et al. Mapping information-rich genotype-phenotype landscapes with genome-scale perturb-seq. *Cell*, 185(14):2559–2575, 2022.

5. Thomas M Norman, Max A Horlbeck, Joseph M Replogle, Alex Y Ge, Albert Xu, Marco Jost, Luke A Gilbert, and Jonathan S Weissman. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

6. Claudia Feng, Elin Madli Peets, Yan Zhou, Luca Crepaldi, Sunay Usluer, Alistair Dunham, Jana M Braunger, Jing Su, Magdalena E Strauss, Daniele Muraro, et al. A genome-scale single cell crispri map of trans gene regulation across human pluripotent stem cell lines. *bioRxiv*, pages 2024–11, 2024.

7. Mohammad Lotfollahi, F Alexander Wolf, and Fabian J Theis. scgen predicts single-cell perturbation responses. *Nature methods*, 16(8):715–721, 2019.

8. 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):e11517, 2023.

9. Charlotte Bunne, Stefan G Stark, Gabriele Gut, Jacobo Sarabia Del Castillo, Mitch Levesque, Kjong-Van Lehmann, Lucas Pelkmans, Andreas Krause, and Gunnar Rätsch. Learning single-cell perturbation responses using neural optimal transport. *Nature methods*, 20(11):1759–1768, 2023.

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

11. 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.

12. Minsheng Hao, Jing Gong, Xin Zeng, Chiming Liu, Yucheng Guo, Xingyi Cheng, Taifeng Wang, Jianzhu Ma, Xuegong Zhang, and Le Song. Large-scale foundation model on single-cell transcriptomics. *Nature methods*, 21(8):1481–1491, 2024.

13. Yuge Ji, Mohammad Lotfollahi, F Alexander Wolf, and Fabian J Theis. Machine learning for perturbational single-cell omics. *Cell Systems*, 12(6):522–537, 2021.

14. Yan Wu, Esther Wershof, Sebastian M Schmon, Marcel Nassar, Błażej Osiński, Ridvan Eksi, Kun Zhang, and Thore Graepel. Perturbench: Benchmarking machine learning models for cellular perturbation analysis. *arXiv preprint arXiv:2408.10609*, 2024.

15. Mathieu Chevalley, Yusuf Roohani, Arash Mehrjou, Jure Leskovec, and Patrick Schwab. Causalbench: A large-scale benchmark for network inference from single-cell perturbation data. *arXiv preprint arXiv:2210.17283*, 2022.

16. Lanxiang Li, Yue You, Wenyu Liao, Xueying Fan, Shihong Lu, Ye Cao, Bo Li, Wenle Ren, Yunlin Fu, Jiaming Kong, et al. A systematic comparison of single-cell perturbation response prediction models. *bioRxiv*, pages 2024–12, 2024.

17. Chen Li, Haoxiang Gao, Yuli She, Haiyang Bian, Qing Chen, Kai Liu, Lei Wei, and Xuegong Zhang. Benchmarking ai models for in silico gene perturbation of cells. *bioRxiv*, pages 2024–12, 2024.

18. Aaron Wenteler, Martina Occhetta, Nikhil Branson, Magdalena Huebner, Victor Curean, William Dee, William Connell, Alex Hawkins-Hooker, Pui Chung, Yasha Ektefaie, et al. Perteval-scfm: Benchmarking single-cell foundation models for perturbation effect prediction. *bioRxiv*, pages 2024–10, 2024.

19. Zoe Piran, Niv Cohen, Yedid Hoshen, and Mor Nitzan. Disentanglement of single-cell data with biolord. *Nature Biotechnology*, pages 1–6, 2024.

20. Michael Bereket and Theofanis Karaletsos. Modelling cellular perturbations with the sparse additive mechanism shift variational autoencoder. *Advances in Neural Information Processing Systems*, 36, 2024.

21. Ethan Weinberger, Chris Lin, and Su-In Lee. Isolating salient variations of interest in single-cell data with contrastivevi. *Nature Methods*, 20(9):1336–1345, 2023.

22. Romain Lopez, Natasa Tagasovska, Stephen Ra, Kyunghyun Cho, Jonathan Pritchard, and Aviv Regev. Learning causal representations of single cells via sparse mechanism shift modeling. In *Conference on Causal Learning and Reasoning*, pages 662–691. PMLR, 2023.

23. Efthymia Papalexi, Eleni P Mimitou, Andrew W Butler, Samantha Foster, Bernadette Bracken, William M Mauck III, Hans-Hermann Wessels, Yuhan Hao, Bertrand Z Yeung, Peter Smibert, et al. Characterizing the molecular regulation of inhibitory immune checkpoints with multimodal single-cell screens. *Nature genetics*, 53(3):322–331, 2021.

24. Ethan Weinberger, Ryan Conrad, and Tal Ashuach. Modeling variable guide efficiency in pooled crispr screens with contrastivevi+. *arXiv preprint arXiv:2411.08072*, 2024.

25. Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053–1058, 2018.

26. Bicna Song, Dingyu Liu, Weiwei Dai, Natalie F McMyn, Qingyang Wang, Dapeng Yang, Adam Krejci, Anatoly Vasilyev, Nicole Untermoser, Anke Loregger, et al. Decoding heterogeneous single-cell perturbation responses. *Nature cell biology*, pages 1–12, 2025.

27. Charlotte Bunne, Geoffrey Schiebinger, Andreas Krause, Aviv Regev, and Marco Cuturi. Optimal transport for single-cell and spatial omics. *Nature Reviews Methods Primers*, 4(1):58, 2024.

28. Qun Jiang, Shengquan Chen, Xiaoyang Chen, and Rui Jiang. scpram accurately predicts single-cell gene expression perturbation response based on attention mechanism. *Bioinformatics*, 40(5):btae265, 2024.

29. Jayoung Ryu, Charlotte Bunne, Luca Pinello, Aviv Regev, and Romain Lopez. Cross-modality matching and prediction of perturbation responses with labeled gromov-wasserstein optimal transport. *arXiv preprint arXiv:2405.00838*, 2024.

30. Jesse Zhang, Airol A Ubas, Richard de Borja, Valentine Svensson, Nicole Thomas, Neha Thakar, Ian Lai, Aidan Winters, Umair Khan, Matthew G. Jones, John D. Thompson, Vuong Tran, Joseph Pangallo, Efthymia Papalexi, Ajay Sapre, Hoai Nguyen, Oliver Sanderson, Maria Nigos, Olivia Kaplan, Sarah Schroeder, Bryan Hariadi, Simone Marrujo, Crina Curca Alec Salvino, Guillermo Gallareta Olivares,

Ryan Koehler, Gary Geiss, Alexander Rosenberg, Charles Roco, Daniele Merico, Nima Alidoust, Hani Goodarzi, and Johnny Yu. Tahoe-100m: A giga-scale single-cell perturbation atlas for context-dependent gene function and cellular modeling. *bioRxiv*, pages 2025–02, 2025.

31. CZI Cell Science Program, Shibla Abdulla, Brian Aevermann, Pedro Assis, Seve Badajoz, Sidney M Bell, Emanuele Bezzi, Batuhan Cakir, Jim Chaffer, Signe Chambers, et al. Cz cellxgene discover: a single-cell data platform for scalable exploration, analysis and modeling of aggregated data. *Nucleic Acids Research*, 53(D1):D886–D900, 2025.

32. Nicholas D Youngblut, Christopher Carpenter, Jaanak Prashar, Chiara Ricci-Tam, Rajesh Ilango, Noam Teyssier, Silvana Konermann, Patrick Hsu, Alexander Dobin, David P Burke, et al. scBaseCount: An AI agent-curated, uniformly processed, and continually expanding single cell data repository. *bioRxiv*, pages 2025–02, 2025.

33. Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, et al. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.

34. Yanay Rosen, Yusuf Roohani, Ayush Agarwal, Leon Samotorčan, Tabula Sapiens Consortium, Stephen R Quake, and Jure Leskovec. Universal cell embeddings: A foundation model for cell biology. *bioRxiv*, pages 2023–11, 2023.

35. Nicholas Ho, Caleb N Ellington, Jinyu Hou, Sohan Addagudi, Shentong Mo, Tianhua Tao, Dian Li, Yonghao Zhuang, Hongyi Wang, Xingyi Cheng, et al. Scaling dense representations for single cell with transcriptome-scale context. *bioRxiv*, pages 2024–11, 2024.

36. Yiqun Chen and James Zou. Genept: a simple but effective foundation model for genes and cells built from chatgpt. *bioRxiv*, pages 2023–10, 2024.

37. James D Pearce, Sara E Simmonds, Gita Mahmoudabadi, Lakshmi Krishnan, Giovanni Palla, Ana-Maria Istrate, Alexander Tarashansky, Benjamin Nelson, Omar Valenzuela, Donghui Li, et al. A cross-species generative cell atlas across 1.5 billion years of evolution: The transcriptformer single-cell model. *bioRxiv*, pages 2025–04, 2025.

38. Graham Heimberg, Rajat Bhatnagar, Hana El-Samad, and Matt Thomson. Low dimensionality in gene expression data enables the accurate extraction of transcriptional programs from shallow sequencing. *Cell systems*, 2(4):239–250, 2016.

39. Artur Szałata, Karin Hrovatin, Sören Becker, Alejandro Tejada-Lapuerta, Haotian Cui, Bo Wang, and Fabian J Theis. Transformers in single-cell omics: a review and new perspectives. *Nature methods*, 21 (8):1430–1443, 2024.

40. Jérémie Kalfon, Jules Samaran, Gabriel Peyré, and Laura Cantini. scprint: pre-training on 50 million cells allows robust gene network predictions. *Nature Communications*, 16(1):3607, 2025.

41. Sanjay R Srivatsan, José L McFaline-Figueroa, Vijay Ramani, Lauren Saunders, Junyue Cao, Jonathan Packer, Hannah A Pliner, Dana L Jackson, Riza M Daza, Lena Christiansen, et al. Massively multiplex chemical transcriptomics at single-cell resolution. *Science*, 367(6473):45–51, 2020.

42. Parse Biosciences. 10 million human pbmcs in a single experiment, 2023. URL https://www.parsebiosciences.com/datasets/10-million-human-pbmcs-in-a-single-experiment/.

43. 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*, pages 1–10, 2025.

44. Longda Jiang, Carol Dalgarno, Efthymia Papalexi, Isabella Mascio, Hans-Hermann Wessels, Huiyoung Yun, Nika Iremadze, Gila Lithwick-Yanai, Doron Lipson, and Rahul Satija. Systematic reconstruction of molecular pathway signatures using scalable single-cell perturbation screens. *Nature Cell Biology*, pages 1–13, 2025.

45. José L McFaline-Figueroa, Sanjay Srivatsan, Andrew J Hill, Molly Gasperini, Dana L Jackson, Lauren Saunders, Silvia Domcke, Samuel G Regalado, Paul Lazarchuck, Sarai Alvarez, et al. Multiplex single-cell chemical genomics reveals the kinase dependence of the response to targeted therapy. *Cell Genomics*, 4(2), 2024.

46. Constantin Ahlmann-Eltze, Wolfgang Huber, and Simon Anders. Deep learning-based predictions of gene perturbation effects do not yet outperform simple linear methods. *BioRxiv*, pages 2024–09, 2024.

47. Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Yuge Ji, Ignacio L Ibarra, F Alexander Wolf, Nafissa Yakubova, Fabian J Theis, and David Lopez-Paz. Compositional perturbation autoencoder for single-cell response modeling. *bioRxiv*, 2021.

48. Eric Kernfeld, Yunxiao Yang, Joshua S Weinstock, Alexis Battle, and Patrick Cahan. A systematic comparison of computational methods for expression forecasting. *BioRxiv*, pages 2023–07, 2023.

49. Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

50. Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

51. 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.

52. Emmanuel Noutahi, Jason Hartford, Prudencio Tossou, Shawn Whitfield, Alisandra K Denton, Cas Wognum, Kristina Ulicna, Jonathan Hsu, Michael Cuccarese, Emmanuel Bengio, et al. Virtual cells: Predict, explain, discover. *arXiv preprint arXiv:2505.14613*, 2025.

53. 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. *arXiv preprint arXiv:2405.17631*, 2024.

54. Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, et al. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pages 2025–05, 2025.

55. Syed Asad Rizvi, Daniel Levine, Aakash Patel, Shiyang Zhang, Eric Wang, Sizhuang He, David Zhang, Cerise Tang, Zhuoyang Lyu, Rayyan Darji, et al. Scaling large language models for next-generation single-cell analysis. *bioRxiv*, pages 2025–04, 2025.

56. Menghua Wu, Russell Littman, Jacob Levine, Lin Qiu, Tommaso Biancalani, David Richmond, and Jan-Christian Huetter. Contextualizing biological perturbation experiments through language. *arXiv preprint arXiv:2502.21290*, 2025.

57. Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

58. Jiaqi Zhang, Kristjan Greenewald, Chandler Squires, Akash Srivastava, Karthikeyan Shanmugam, and Caroline Uhler. Identifiability guarantees for causal disentanglement from soft interventions. *Advances in Neural Information Processing Systems*, 36:50254–50292, 2023.

59. Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690, 2019.

60. Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

61. Chunlei Wu, Ian MacLeod, and Andrew I Su. Biogps and mygene. info: organizing online, gene-centric information. *Nucleic acids research*, 41(D1):D561–D565, 2013.

62. Andrew Yates, Kathryn Beal, Stephen Keenan, William McLaren, Miguel Pignatelli, Graham RS Ritchie, Magali Ruffier, Kieron Taylor, Alessandro Vullo, and Paul Flicek. The ensembl rest api: Ensembl data for any language. *Bioinformatics*, 31(1):143–145, 2015.

63. Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

64. Jing Jiang, Cankun Wang, Ren Qi, Hongjun Fu, and Qin Ma. scread: a single-cell rna-seq database for alzheimer's disease. *Iscience*, 23(11), 2020.

65. José L McFaline-Figueroa, Andrew J Hill, Xiaojie Qiu, Dana Jackson, Jay Shendure, and Cole Trapnell. A pooled single-cell genetic screen identifies regulatory checkpoints in the continuum of the epithelial-to-mesenchymal transition. *Nature genetics*, 51(9):1389–1398, 2019.

66. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

67. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

68. Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning, 2024. URL https://arxiv.org/abs/2407.01800.

69. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

70. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

71. Abhay Kumar, Louis Owen, Nilabhra Roy Chowdhury, and Fabian Güra. Zclip: Adaptive spike mitigation for llm pre-training. *arXiv preprint arXiv:2504.02507*, 2025.

72. Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

73. Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

74. Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

75. Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

76. Hadi Daneshmand. Provable optimal transport with transformers: The essence of depth and prompt engineering. *arXiv preprint arXiv:2410.19931*, 2024.

77. Takashi Furuya, Maarten V de Hoop, and Gabriel Peyré. Transformers are universal in-context learners. *arXiv preprint arXiv:2408.01367*, 2024.

78. Shiba Biswal, Karthik Elamvazhuthi, and Rishi Sonthalia. Identification of mean-field dynamics using transformers. *arXiv preprint arXiv:2410.16295*, 2024.

79. Wassily Hoeffding. The strong law of large numbers for u-statistics. *Univ. of North Carolina Institute of statistics. Mimeo Series*, 302, 1961.

80. Victor De la Pena and Evarist Giné. *Decoupling: from dependence to independence*. Springer Science & Business Media, 2012.

81. Maria L Rizzo and Gábor J Székely. Energy distance. *wiley interdisciplinary reviews: Computational statistics*, 8(1):27–38, 2016.

82. Luis A Caffarelli. The regularity of mappings with a convex potential. *Journal of the American Mathematical Society*, 5(1):99–104, 1992.

83. Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

84. Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

85. Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

86. Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

87. Gal Vardi. On the implicit bias in deep-learning algorithms. *Communications of the ACM*, 66(6):86–93, 2023.

5. Appendix

5.1. Data Generation Process

We assume that the observed log-normalized perturbed expression state of each cell, X_{p}, is a random variable generated from an *unobservable* unperturbed state, X_0, which itself is a random variable representing the cell's underlying expression state. X_0 is drawn from a basal cell distribution $\mathcal{D}_{\mathrm{basal}}$ for a given set of covariates (e.g., cell line, batch condition, etc), and the perturbation effect can be modeled as follows:

$$X_p \;=\; X_0 \;+\; T_p(X_0) \;+\; \varepsilon, \qquad X_0 \sim \mathcal{D}_{\mathrm{basal}} \tag{1}$$

where

- $T_p(X_0)$: True effect caused by perturbation p.
- ε: Experiment-specific technical noise, assumed independent of X_0.

As single-cell transcriptomic measurements destroy the cell, X_0 is unobservable, so directly modeling **Eq. 1** is not feasible. Instead, our method operates on the observable $\mathcal{D}_{\mathrm{basal}}$ to predict the perturbed state, denoted as $\hat{X}_p$. This forms the basis of our model:

$$\begin{aligned} \hat{X}_p \;&\sim\; H(\mathcal{D}_{\mathrm{basal}}) \;+\; \hat{T}_p(\mathcal{D}_{\mathrm{basal}}) \;+\; \varepsilon, \qquad \varepsilon \sim P_\varepsilon \\ X_{\mathrm{p}} \;&\overset{d}{\approx}\; \hat{X}_{\mathrm{p}} \end{aligned} \tag{2}$$

In this approximation, the true effect of the perturbation $T_p(X_0)$ is now considered in the context of the entire basal population, denoted as $\hat{T}_p(\mathcal{D}_{\mathrm{basal}})$. Additionally, we explicitly introduce $H(\mathcal{D}_{\mathrm{basal}})$ to represent the biological heterogeneity inherent in the baseline population. This heterogeneity was implicitly removed when sampling $X_0 \sim \mathcal{D}_{\mathrm{basal}}$ in the first equation but is made explicit here to reflect our shift to a distributional view. $\hat{X}_{\mathrm{p}}$ can be seen as a distributional analogue of X_{p}, allowing us to model the perturbed state based on observable population characteristics rather than unobservable individual cell states.

5.2. Task Description

We now describe the core prediction tasks used for evaluation.

5.2.1. Underrepresented Context Generalization Task

To evaluate the generalization capabilities of perturbation models, we first test the generalization of perturbation effects within Perturb-Seq datasets. Given the dataset of interest, let the collection of biological contexts (cell types or cell lines in some dataset) be $\mathcal{C}$ and the perturbation catalogue be $\mathcal{P}$. For every context $c \in \mathcal{C}$, we write $\mathcal{P}_c \subseteq \mathcal{P}$ for the subset of perturbations that were profiled in that context, and we denote by $X_{p,c} \in \mathbb{R}^d$ the gene–expression vector generated for the pair (p, c) according to **Eq. 1**. To test the generalization capabilities of the model in an underrepresented context, we fix a test context $c^* \in \mathcal{C}$. We then choose a proportion $\alpha = 0.30$ and draw a support set

$$\mathcal{P}_{c^*}^{\mathrm{sup}} \subset \mathcal{P}_{c^*}, \qquad |\mathcal{P}_{c^*}^{\mathrm{sup}}| = \big\lfloor \alpha \, |\mathcal{P}_{c^*}| \big\rfloor, \tag{3}$$

uniformly at random. The remaining perturbations form the target perturbation set $\mathcal{P}_{c^*}^{\mathrm{target}} = \mathcal{P}_{c^*} \setminus \mathcal{P}_{c^*}^{\mathrm{sup}}$. The training and test splits are then given as follows:

$$\mathcal{D}_{\mathrm{train}} = \big\{(X_{p,c}, p, c) \,:\, c \in \mathcal{C} \setminus \{c^*\},\, p \in \mathcal{P}_c\big\} \;\cup\; \big\{(X_{p,c^*}, p, c^*) \,:\, p \in \mathcal{P}_{c^*}^{\mathrm{sup}}\big\}, \tag{4}$$

$$\mathcal{D}_{\mathrm{test}} = \big\{(X_{p,c^*}, p, c^*) \,:\, p \in \mathcal{P}_{c^*}^{\mathrm{target}}\big\}. \tag{5}$$

Thus, the model sees all perturbations in $\mathcal{C} \setminus \{c^*\}$ and only 30% of the perturbations in the test context c^*. If $|\mathcal{C}|$ is large, a diverse fixed subset $\mathcal{C}_{\mathrm{test}} \subset \mathcal{C}$ of n contexts is designated as test contexts. We form $\mathcal{D}_{\mathrm{train}}$ from the $\mathcal{C} \setminus \mathcal{C}_{\mathrm{test}}$ contexts, as well as 30% of the perturbations from each $c_j \in \mathcal{C}_{\mathrm{test}}$. We then construct multiple $\mathcal{D}_{\mathrm{test}}$, one from each $c_j \in \mathcal{C}_{\mathrm{test}}$, each containing the remaining unseen perturbations from c_j. If $|\mathcal{C}|$ is small, we *iteratively* leave out a single context $c^* = c_j$, form $\mathcal{D}_{\mathrm{train}}$ with the remaining $\mathcal{C} \setminus \{c_j\}$ contexts as well as 30%

of the perturbations from c_j and form $\mathcal{D}_{\text{test}}$ from the remaining unseen perturbations from c_j. This iteration creates multiple training sets, allowing to report performance over separately trained models. In both cases, we report the mean loss across $\mathcal{D}_{\text{test}}$, yielding a robust estimate of the model's ability to extrapolate to unseen perturbations within an underrepresented context. Unperturbed cells from all contexts including the underrepresented context are available to the model during training.

5.2.2. Zero-Shot Context Generalization Task

To enhance performance on this task, we include a pre-training step. Each model is initially pre-trained on one or more large perturbation datasets (e.g., Tahoe-100M) and subsequently fine-tuned to a different dataset, referred to as the query dataset, via full fine-tuning. The model is then tested on a held-out context within the query dataset.

Specifically, the model is first pre-trained on one or more datasets that include perturbations across multiple contexts. Then, given a query dataset, we hold out one context and use the remaining contexts for fine-tuning. The contexts used for fine-tuning contain *a superset of the perturbations* found in the held-out context. The fine-tuned model is evaluated on its ability to predict the effects of those perturbations within the held-out context of the query dataset. Formally, the fine-tuning process involves partitioning the set of contexts in the query dataset into $\mathcal{C}_{\text{fine-tune}} \subset \mathcal{C}$ and $\mathcal{C}_{\text{test}} \subset \mathcal{C}$ with $\mathcal{C}_{\text{fine-tune}} \cap \mathcal{C}_{\text{test}} = \varnothing$. The data used in the fine-tuning phase

$$\mathcal{D}_{\text{fine-tune}} = \big\{ (X_{p,c}, p, c) \,:\, c \in \mathcal{C}_{\text{fine-tune}}, p \in \mathcal{P} \big\} \tag{6}$$

contain all perturbations in $\mathcal{C}_{\text{fine-tune}}$. After fine-tuning f_θ on $\mathcal{D}_{\text{fine-tune}}$, we evaluate it on the same perturbations $p \in \mathcal{P}$ but in unseen, held-out cell lines $c \in \mathcal{C}_{\text{test}}$. Thus, the test set is

$$\mathcal{D}_{\text{test}} = \big\{ (X_{p,c}, p, c) \,:\, c \in \mathcal{C}_{\text{test}}, p \in \mathcal{P} \big\}. \tag{7}$$

This task directly assesses whether the model has learned generalizable relationships between biological context and perturbation response. Unperturbed cells from all contexts including the held out context are available to the model during training.

5.3. State Transition Model (ST)

ST is a deep learning model that learns the transcriptomic responses to perturbations across populations of cells. The model uses a transformer architecture with self-attention across cells to predict perturbation effects between control and perturbed cell distributions. Let

$$\mathcal{D} \;=\; \big\{ \big(\mathbf{x}^{(i)},\, p_i,\, \ell_i,\, b_i\big) \big\}_{i=1}^{N}, \qquad \mathbf{x}^{(i)} \in \mathbb{R}^G. \tag{8}$$

denote a dataset of single–cell RNA-sequencing measurements over the gene set, with G the number of genes. Here, $\mathbf{x}^{(i)}$ represents the normalized expression vector for cell i, after its raw count values have been processed. Specifically, the raw count value observed for gene j in cell i after perturbation p_i (if any) is initially $\mathbf{x}_j^{(i),\text{raw}} \in \mathbb{N}_0$. These raw counts are depth-normalized and log-transformed using Scanpy(`normalize_total` $\rightarrow$ `log1p`), yielding non-negative, real-valued expression vectors $\mathbf{x}^{(i)}$. Each cell is annotated with perturbation label $p_i \in \{1, \ldots, P\} \cup \{\texttt{ctrl}\}$, a biological context or cell line label $\ell_i \in \{1, \ldots, L\}$, and an optional batch effect label $b_i \in \{1, \ldots, B\}$.

5.3.1. Formation of Cell Sets

We group cells into sets based on their biological context, perturbation, and batch labels:

$$\mathcal{C}_{\ell,p,b} \;=\; \big\{ \mathbf{x}^{(i)} \in \mathcal{D} \;\big|\; \ell_i = \ell,\; p_i = p,\; b_i = b \big\}. \tag{9}$$

Let $N_{\ell,p,b} = \big|\mathcal{C}_{\ell,p,b}\big|$ denote the number of cells in each group. Given a target set size S, we partition $\mathcal{C}_{\ell,p,b}$ into non-overlapping subsets of size S to construct a collection of tensors $\mathcal{S}_{\ell,p,b}^{(k)} \in \mathbb{R}^{S \times G}$, where $k \in \left\{1, \ldots, \left\lfloor \frac{N_{\ell,p,b}}{S} \right\rfloor \right\}$. If $N_{\ell,p,b}$ is not divisible by S, the remaining cells form a final, smaller set, which is padded to size S by sampling additional cells with replacement from itself.

5.3.2. Training on Cell Sets

During training, each set $\mathcal{S}^{(k)}_{\ell,p,b}$, where $p \in \{1,\dots,P\} \cup \{\texttt{ctrl}\}$, is paired with a corresponding control set. The control set is constructed by randomly sampling S control cells from the same cell line ℓ, and optionally same batch b. Formally, for each training example $\mathcal{S}^{(k)}_{\ell,p,b}$, we construct a paired control set via a `map` operation:

$$\texttt{map}(\mathcal{S}^{(k)}_{\ell,p,b}) = \texttt{stack}([\mathbf{x}^{(i)}]_{\mathbf{x}^{(i)} \sim \mathcal{C}_{\ell,\texttt{ctrl},b}}) \in \mathbb{R}^{S \times G} \tag{10}$$

This mapping is applied uniformly, regardless of whether $\mathcal{S}^{(k)}_{\ell,p,b}$ contains perturbed or control cells. The choice of `map` function effectively determines which sources of variation are explicitly controlled for. By conditioning on specific covariates (e.g., cell line ℓ, batch b), the mapping function reduces known sources of heterogeneity that could otherwise confound true perturbation signals. Different mapping strategies introduce different priors on non-perturbational variation. For instance, sampling control cells from the same batch can reduce technical variation but limit the number of unique cells per set, which may limit model performance.

To construct mini-batches, we collect B such set pairs, $\{(\mathcal{S}^{(k_i)}_{\ell_i,p_i,b_i}, \texttt{map}(\mathcal{S}^{(k_i)}_{\ell_i,p_i,b_i}))\}_{i=1,\dots B}$, where different pairs may originate from a different combination of cell line, perturbation, and (optionally) batch. These are then arranged into the following tensors:

$$\begin{aligned}
\mathbf{X}_{\text{target}} &= \texttt{stack}([\mathcal{S}^{(k_1)}_{\ell_1,p_1,b_1}, \dots, \mathcal{S}^{(k_B)}_{\ell_B,p_B,b_B}]) \in \mathbb{R}^{B \times S \times G} \\
\mathbf{X}_{\text{ctrl}} &= \texttt{stack}([\texttt{map}(\mathcal{S}^{(k_1)}_{\ell_1,p_1,b_1}), \dots, \texttt{map}(\mathcal{S}^{(k_B)}_{\ell_B,p_B,b_B})]) \in \mathbb{R}^{B \times S \times G} \\
\mathbf{Z}_{\text{pert}} &\in \mathbb{R}^{B \times S \times D_{\text{pert}}} \quad \text{(perturbation embeddings)} \\
\mathbf{Z}_{\text{batch}} &\in \mathbb{R}^{B \times S \times D_{\text{batch}}} \quad \text{(optional batch covariates)}
\end{aligned} \tag{11}$$

where D_{pert} and D_{batch} denote the dimensionalities of the perturbation and batch embeddings, respectively. For State, we use one-hot encodings of perturbation and batch, so D_{pert} equals the number of unique perturbations across the data, and D_{batch} equals the number of unique batch labels. ST takes $\mathbf{X}_{\text{ctrl}}$ as input along with the perturbation embeddings $\mathbf{Z}_{\text{pert}}$ and learns to predict $\mathbf{X}_{\text{target}}$ as output, learning to transform control cell populations into their corresponding perturbed states.

5.3.3. Neural Network Modules

ST uses specialized encoders that map cellular expression profiles, perturbation labels, and optionally batch labels into a shared hidden dimension d_h, which serves as the input to the transformer.

Control Cell Encoder. Each log-normalized expression vector $\mathbf{x}^{(i)} \in \mathbb{R}^G$ is mapped to an embedding via a 4-layer MLP with GELU activations. The MLP f_{cell} is applied to each cell independently across the entire control tensor:

$$\mathbf{H}_{\text{cell}} \;=\; f_{\text{cell}}(\mathbf{X}_{\text{ctrl}}) \;\in\; \mathbb{R}^{B \times S \times d_h} \tag{12}$$

This transforms the input shape from $(B \times S \times G) \to (B \times S \times d_h)$.

Perturbation Encoder. Perturbation labels are encoded into the same embedding dimension d_h. For one-hot encoded perturbations, the input vector is passed through a 4-layer MLP with GELU activations:

$$\mathbf{H}_{\text{pert}} \;=\; f_{\text{pert}}(\mathbf{Z}_{\text{pert}}) \;\in\; \mathbb{R}^{B \times S \times d_h} \tag{13}$$

This transforms the input shape $(B \times S \times D_{\text{pert}}) \to (B \times S \times d_h)$ (note the perturbation embedding is the same for all cells within the same set of a given batch). Alternatively, when perturbations are represented by continuous features (e.g., molecular descriptors or gene embeddings), the embeddings are directly used in $\mathbf{H}_{\text{pert}}$ and we set $d_h = D_{\text{pert}}$.

Batch Encoder. To account for technical batch effects, batch labels $b_i \in \{1,\dots,B\}$ are encoded into embeddings of dimension d_h:

$$\mathbf{H}_{\text{batch}} = f_{\text{batch}}(\mathbf{Z}_{\text{batch}}) \in \mathbb{R}^{B \times S \times d_h} \tag{14}$$

where f_{batch} is an embedding layer. This transforms the input shape from $(B \times S \times D_{\text{batch}}) \to (B \times S \times d_h)$.

Transformer Inputs and Outputs. The final input to ST is constructed by summing the control cell embeddings with the perturbation and batch embeddings:

$$\mathbf{H} = \mathbf{H}_{\text{cell}} + \mathbf{H}_{\text{pert}} + \mathbf{H}_{\text{batch}} \tag{15}$$

This composite representation is passed to the transformer backbone f_{ST} to model perturbation effects across the cell set. The output is computed as:

$$\mathbf{O} = \mathbf{H} + f_{\text{ST}}(\mathbf{H}) \tag{16}$$

where $\mathbf{O} \in \mathbb{R}^{B \times S \times d_h}$ represents the final output. This formulation encourages the transformer f_{ST} to learn perturbation effects as residuals to the input representation $\mathbf{H}$.

Gene Reconstruction Head. When working with inputs directly in expression space, the gene reconstruction head maps the output of the transformer, $\mathbf{O}$, back to gene expression space. This is done with a linear projection layer, applied independently to the d_h-dimensional hidden representation of each token within each cell set. Specifically, for each cell b in the batch, where $\mathbf{O}^{(b)} \in \mathbb{R}^{S \times d_h}$ is the transformer output for cell b, the reconstructed gene expression $\hat{\mathbf{X}}^{(b)}_{\text{target}} \in \mathbb{R}^{S \times G}$ is given by:

$$\hat{\mathbf{X}}^{(b)}_{\text{target}} = f_{\text{recon}}(\mathbf{O}^{(b)}) = \mathbf{O}^{(b)}\mathbf{W}_{\text{recon}} + \mathbf{b}_{\text{recon}} \tag{17}$$

where $\mathbf{W}_{\text{recon}} \in \mathbb{R}^{d_h \times G}$ and $\mathbf{b}_{\text{recon}} \in \mathbb{R}^{G}$ are learnable parameters. This operation transforms the hidden representations, yielding reconstructed log-transformed gene expression values for each cell in the batch.

5.3.4. Learning Perturbation Effects with Maximum Mean Discrepancy

ST is trained to minimize the discrepancy between predicted ($\hat{\mathbf{X}}_{\text{target}}$) and observed ($\mathbf{X}_{\text{target}}$) transcriptomic responses. This is quantified using the Maximum Mean Discrepancy (MMD) [57], a statistical measure of distance between two probability distributions based on their embeddings in a Reproducing Kernel Hilbert Space (RKHS), via a kernel function of choice. MMD has been applied previously to model single-cell perturbation effects [58].

For each mini-batch element b, we consider the set of S predicted cell expression vectors of cells within the batch (therefore those sharing the same combination of cell line ℓ and perturbation p, as described in **Eq. 11**). With a slight abuse of notation, we denote this set as $\hat{\mathbf{X}}^{(b)}_{\text{target}} = \{\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^G \text{ such that } i \in b\}_{i=1}^{S}$, and the corresponding set of S target cell expression vectors of cells within the batch as $\mathbf{X}^{(b)}_{\text{target}} = \{\mathbf{x}^{(i)} \in \mathbb{R}^G \text{ such that } i \in b\}_{i=1}^{S}$. The MMD is then used to minimize the distance between the empirical probability distributions implicitly defined by these two finite sets of vectors.

The squared MMD between the predicted and observed cell sets is computed as:

$$\text{MMD}^2\big(\hat{\mathbf{X}}^{(b)}_{\text{target}}, \mathbf{X}^{(b)}_{\text{target}}\big) = \frac{1}{S^2}\sum_{i=1}^{S}\sum_{j=1}^{S}\Big[k(\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{x}}^{(j)}) + k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - 2k(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(j)})\Big] \tag{18}$$

where $k(\cdot, \cdot)$ denotes the kernel function. The three terms correspond to: (1) similarity within the predicted set, (2) similarity within the observed set, and (3) cross-similarity between predicted and observed sets.

We use the energy distance kernel:

$$k(\mathbf{u}, \mathbf{v}) = -\|\mathbf{u} - \mathbf{v}\|_2,$$

implemented via the geomloss library [59].

For notational convenience, for a training minibatch of B cell sets, we define the batch-averaged MMD loss as the average of these MMD^2 values:

$$\mathcal{L}_{\text{MMD}}(\hat{\mathbf{X}}_{\text{target}}, \mathbf{X}_{\text{target}}) = \frac{1}{B}\sum_{b=1}^{B} \text{MMD}^2\big(\hat{\mathbf{X}}_{\text{target}}^{(b)}, \mathbf{X}_{\text{target}}^{(b)}\big). \tag{19}$$

Minimizing this loss encourages the model to generate sets of perturbed cell expression vectors whose overall statistical properties, as captured by the MMD, consistent with perturbation labels and match those of the observed cell sets. The total loss is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MMD}}(\hat{\mathbf{X}}_{\text{target}}, \mathbf{X}_{\text{target}}). \tag{20}$$

5.3.5. Training ST in Embedding Spaces

ST offers the flexibility to be trained either directly in gene expression space or in a specified embedding space. When trained directly in gene expression space, ST operates on the top 2,000 highly variable genes (HVGs) as identified in our preprocessing. In this mode, the model is referred to as ST+HVG in our results. To enable training in an embedding space, the architecture is modified to include an additional expression decoder. Formally, let E denote the dimensionality of the embedding space, where typically $E \ll G$. The tensors $\mathbf{X}_{\text{target}}$ and $\mathbf{X}_{\text{ctrl}}$, now denoted as $\mathbf{X}_{\text{target}}^{\text{emb}}$ and $\mathbf{X}_{\text{ctrl}}^{\text{emb}}$, are of dimension $B \times S \times E$. Accordingly, f_{cell} is modified to transform the input shape from $(B \times S \times E) \to (B \times S \times d_h)$, and f_{recon} is modified to transform the output shape from $(B \times S \times d_h) \to (B \times S \times E)$. This output is denoted as $\hat{\mathbf{X}}_{\text{target}}^{\text{emb}}$. The other encoders and the transformer itself remain unchanged.

To recover the original gene expression of the target cells, $\mathbf{X}_{\text{target}}$, we train an additional decoder head f_{decode}. This is a multi-layer MLP with dropout that maps from the embedding space back to the full gene expression space:

$$\hat{\mathbf{X}}_{\text{target}} = f_{\text{decode}}(\hat{\mathbf{X}}_{\text{target}}^{\text{emb}}) \tag{21}$$

This transforms the predicted embeddings from $(B \times S \times E) \to (B \times S \times G)$ to recover gene expression profiles.

The ST loss is hence modified to make use of the two target sets: $\mathbf{X}_{\text{target}}^{\text{emb}} \in \mathbb{R}^{B\times S\times E}$ containing the target embeddings, and $\mathbf{X}_{\text{target}} \in \mathbb{R}^{B\times S\times G}$ containing the target gene expression profiles. Specifically, the model is trained using a weighted combination of MMD losses in the embedding and gene expression spaces:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MMD}}(\hat{\mathbf{X}}_{\text{target}}^{\text{emb}}, \mathbf{X}_{\text{target}}^{\text{emb}}) + 0.1 \cdot \mathcal{L}_{\text{MMD}}(\hat{\mathbf{X}}_{\text{target}}, \mathbf{X}_{\text{target}})$$

The expression loss is down-weighted by a factor of 0.1 to balance the two terms and avoid overwhelming the primary objective in embedding space. This encourages the model to learn perturbation effects primarily in the embedding space while simultaneously decoding to expression space. Our experiments demonstrate that the embedding space improves learning of perturbational effects across datasets, suggesting that the smoother structure of the embedding space facilitates better modeling of perturbation biology.

5.4. State Embedding Model (SE)

ST motivates the development of high-quality cell embeddings that capture relevant biological signal while reducing technical artifacts. For this, we developed the State Embedding model (SE), a self-supervised model trained with a gene expression prediction objective to learn cell representations from single-cell RNA sequencing data. The embeddings produced by SE serve as inputs to ST, enabling more robust transfer across datasets and biological contexts.

5.4.1. Gene Representation via Protein Language Models

Our dataset (**Eq. 8**) consists of cell expression measurements $\mathbf{x}^{(i)}$, where each entry $\mathbf{x}_j^{(i)}$ represents the normalized expression level of gene j in cell i. Beyond these expression values, the SE model incorporates rich information about genes themselves through pretrained protein language model embeddings $\mathbf{g}_j$.

Following recent work [34, 37], we leverage pretrained protein language models to encode gene features. Gene embeddings are computed with ESM-2 (`esm2_t48_15B_UR50D` [60]), first by computing transcript embeddings by averaging per-amino-acid embeddings for each protein-coding transcript in the gene, and then by averaging across all the transcripts in the gene. This featurization captures evolutionary and functional relationships between genes. Genes without Ensembl IDs are mapped using the MyGene API [61] and Ensembl REST API [62], with manual curation for ambiguous cases. Non-protein-coding genes are excluded from SE.

Each gene embedding $\mathbf{g}_j \in \mathbb{R}^{5120}$ is projected into the model's embedding dimension h via a learnable encoder:

$$\tilde{\mathbf{g}}_j = \text{SiLU}(\text{LayerNorm}(\mathbf{g}_j \mathbf{W}_g + \mathbf{b}_g)) \tag{22}$$

where $\mathbf{W}_g \in \mathbb{R}^{5120 \times h}$ and $\mathbf{b}_g \in \mathbb{R}^h$ are learnable parameters, and SiLU denotes the Sigmoid Linear Unit activation function [63].

5.4.2. Cell Representation

We represent each cell i as a sequence of its most highly expressed genes in $\mathbf{x}^{(i)}$. For each cell, we construct an "expression set" by selecting the top $L = 2048$ genes ranked by log fold expression level. Empirically, increasing L beyond 2048 yielded diminishing returns in model performance. The expression set is then augmented with two special tokens:

$$\tilde{\mathbf{c}}^{(i)} = [\mathbf{z}_{\text{cls}}, \tilde{\mathbf{g}}_1^{(i)}, \tilde{\mathbf{g}}_2^{(i)}, \ldots, \tilde{\mathbf{g}}_L^{(i)}, \mathbf{z}_{\text{ds}}] \in \mathbb{R}^{(L+2) \times h} \tag{23}$$

where $\mathbf{z}_{\text{cls}} \in \mathbb{R}^h$ is a learnable classification token used to aggregate cell-level information, and $\mathbf{z}_{\text{ds}} \in \mathbb{R}^h$ is a learnable dataset token that helps disentangle dataset-specific effects. Here, $\tilde{\mathbf{g}}_\ell^{(i)}$ represents the projected embedding of the ℓ-th most highly expressed gene in cell i, with $\ell \in \{1 \ldots, L\}$. The gene selection is cell-specific: different cells may have different sets of highly expressed genes, leading to different gene embeddings in their respective expression sets. If a cell expresses fewer than L genes, the expression set is padded to length L by randomly sampling from the pool of unexpressed genes. This maintains a fixed-length input for transformer processing.

5.4.3. Expression-Aware Embeddings

Although genes in the cell expression set are sorted by expression level, their magnitudes are not explicitly encoded or used by the model. Instead, SE incorporates expression values directly using an expression embedding scheme inspired by soft binning [12]. For the ℓ-th most expressed gene in cell i's expression set (which corresponds to gene ID $j_\ell^{(i)}$) with log-normalized expression value $\mathbf{x}^{(i)}_{j_\ell^{(i)}}$ in cell i, we compute a soft bin assignment:

$$\boldsymbol{\alpha}_\ell^{(i)} = \text{Softmax}(\text{MLP}_{\text{count}}(\mathbf{x}^{(i)}_{j_\ell^{(i)}})) \in \mathbb{R}^{10} \tag{24}$$

$$\mathbf{e}_\ell^{(i)} = \sum_{k=1}^{10} \boldsymbol{\alpha}_{\ell,k}^{(i)} \mathbf{b}_k \tag{25}$$

where $\text{MLP}_{\text{count}} : \mathbb{R} \to \mathbb{R}^{10}$ consists of two linear layers (dimensions $1 \to 512 \to 10$) with LeakyReLU activation, and $\{\mathbf{b}_k\}_{k=1}^{10}$ are learnable bin embeddings of dimension h. The resulting expression encodings $\mathbf{e}_\ell^{(i)}$ are added to the corresponding gene identity embeddings $\tilde{\mathbf{g}}_\ell^{(i)}$:

$$\mathbf{g}_\ell^{(i)} = \tilde{\mathbf{g}}_\ell^{(i)} + \mathbf{e}_\ell^{(i)} \tag{26}$$

5.4.4. Transformer Encoding

The input expression set, composed of expression-aware gene embeddings and special tokens, is passed through the transformer encoder f_{SE}:

$$\mathbf{E}^{(i)} = f_{\text{SE}}([\mathbf{z}_{\text{cls}}, \mathbf{g}_1^{(i)}, \mathbf{g}_2^{(i)}, \ldots, \mathbf{g}_L^{(i)}, \mathbf{z}_{\text{ds}}]) \in \mathbb{R}^{(L+2) \times h} \tag{27}$$

where each $\mathbf{g}_\ell^{(i)}$, for $\ell \in \{1, \ldots, L\}$ is as in **Eq. 26**. Note that the expression encodings are set to zero for the special tokens `[CLS]` and `[DS]`.

The output is a sequence of contextualized embeddings. The cell embedding is then extracted from the `[CLS]` token at position 0 and normalized:

$$\mathbf{e}_{\text{cls}}^{(i)} = \text{LayerNorm}(\mathbf{E}_0^{(i)}) \in \mathbb{R}^h \tag{28}$$

This embedding serves as a summary representation of the cell's transcriptomic state. Similarly, we extract the dataset representation from the `[DS]` token at position $L + 1$:

$$\mathbf{e}_{\text{ds}}^{(i)} = \text{LayerNorm}(\mathbf{E}_{L+1}^{(i)}) \in \mathbb{R}^h \tag{29}$$

This embedding is used to capture and account for dataset-specific effects during training.

The final embedding is the concatenation of these two quantities:

$$\mathbf{z}_{\text{cell}}^{(i)} = [\mathbf{e}_{\text{cls}}^{(i)}, f_{\text{proj}}(\mathbf{e}_{\text{ds}}^{(i)})] \in \mathbb{R}^{h+10} \tag{30}$$

where $f_{\text{proj}}(\mathbf{e}_{\text{ds}}^{(i)}) \in \mathbb{R}^{10}$ is a learned linear projection of the dataset embedding from $h \to 10$. This cell embedding $\mathbf{z}_{\text{cell}}^{(i)}$ serves as the input representation for individual cells in ST, enabling the multi-scale State architecture to leverage rich cellular representations for population-level perturbation modeling.

5.4.5. Pretraining Objectives

SE is trained using a self-supervised learning framework with two complementary objectives: (1) a gene expression prediction task, and (2) an auxiliary dataset classification task that helps disentangle technical batch effects from biological signal.

Gene Expression Prediction. During training, the model receives the complete input cell expression set (as described in **Eq. 23**), and is tasked with predicting expression values for a selected set of 1,280 genes per cell. To ensure coverage across the expression dynamic range, the target genes for prediction are drawn from three categories:

- $\mathcal{P}^{(i)}$, a set of $|\mathcal{P}^{(i)}| = 512$ highly expressed genes (from the top L genes in the cell expression set i),
- $\mathcal{N}^{(i)}$, a set of $|\mathcal{N}^{(i)}| = 512$ unexpressed genes (randomly sampled from genes not in the top L of cell i),
- $\mathcal{R}$ a set of $|\mathcal{N}^{(i)}| = 256$ genes randomly sampled from the full gene set, *shared for all cells in the batch.*

This results in 1,280 genes for which expression values must be predicted per cell. This strategy encourages the model to reconstruct expression values across a wide dynamic range and to learn meaningful representations of both expressed and silent genes, while having access to the complete transcriptomic context, through self attention, during prediction.

Expression Prediction Decoder. To learn from gene expression prediction across different datasets with varying read depth, we use an MLP decoder that combines multiple sources of information:

$$\hat{\mathbf{x}}_j^{(i)} = \text{MLP}_{\text{dec}}([\mathbf{z}_{\text{cell}}^{(i)}; \tilde{\mathbf{g}}_j; r^{(i)}]) \tag{31}$$

where $\mathbf{z}_{\text{cell}}^{(i)} \in \mathbb{R}^{h+10}$ is the learned cell embedding (**Eq. 30**), $\tilde{\mathbf{g}}_j \in \mathbb{R}^h$ is the embedding of the target gene (**Eq. 22**), and $r^{(i)} \in \mathbb{R}$ is a scalar read depth indicator, computed as the mean log expression of expressed genes in the input expression set for each cell i. These are concatenated and passed through MLP_{dec}, which consists of two skip-connected blocks followed by a linear output layer that predicts the log expression for the target gene j in cell i.

For each cell i in a training batch, let $\hat{\mathbf{Y}}^{(i)} = [\hat{\mathbf{x}}_j^{(i)}]_{j \in \mathcal{P}^{(i)} \cup \mathcal{N}^{(i)} \cup \mathcal{R}} \in \mathbb{R}^{1 \times 1280}$ denote the row vector containing the set of predicted expression values for the genes in $\mathcal{P}^{(i)} \cup \mathcal{N}^{(i)} \cup \mathcal{R}$, and let $\mathbf{Y}^{(i)} = [\mathbf{x}_j^{(i)}]_{j \in \mathcal{P}^{(i)} \cup \mathcal{N}^{(i)} \cup \mathcal{R}} \in$

$\mathbb{R}^{1\times 1280}$ denote the row vector with corresponding true expression values. The tensors are stacked across cells in the batch: $\mathbf{Y} = \texttt{stack}([\mathbf{Y}^{(i)}]_{i=1}^{B})$ and $\hat{\mathbf{Y}} = \texttt{stack}([\hat{\mathbf{Y}}^{(i)}]_{i=1}^{B})$, with resulting shape $(B \times 1 \times 1280)$. The gene-level loss is:

$$\mathcal{L}_{\text{gene}} = \frac{1}{B}\sum_{b=1}^{B} \|\hat{\mathbf{Y}}^{(b)} - \mathbf{Y}^{(b)}\|_2 \tag{32}$$

and therefore it effectively measures the similarity between predicted and true gene expression patterns within each cell b (which is then averaged across cells in $\mathcal{L}_{\text{gene}}$).

To capture variation in gene expression across cells in the mini-batch, we also compute a cell-level loss using the shared subset of genes $\mathcal{R}$. Let $\hat{\mathbf{S}}^{(i)} = \texttt{stack}([\hat{\mathbf{x}}_j^{(i)}]_{j\in\mathcal{R}}) \in \mathbb{R}^{256}$ and $\mathbf{S}^{(i)} = \texttt{stack}([\mathbf{x}_j^{(i)}]_{j\in\mathcal{R}}) \in \mathbb{R}^{256}$ denote the predicted and true expression values in cell i for the shared genes $\mathcal{R}$. These tensors are stacked across cells in the batch and transposed: $\mathbf{S}' = \texttt{transpose}(\texttt{stack}([\mathbf{S}^{(i)}]_{i=1}^{B}))$ and $\hat{\mathbf{S}}' = \texttt{transpose}(\texttt{stack}([\hat{\mathbf{S}}^{(i)})]_{i=1}^{B})$, resulting in shape $(|\mathcal{R}| \times 1 \times B)$, with $|\mathcal{R}| = 256$. We then compute the distance between the concatenated predictions and targets across all cells in the batch for each gene. Specifically, we consider the r-th gene (row) of the transposed tensors, denoted as $\hat{\mathbf{S}}'^{(r)}$ and $\mathbf{S}'^{(r)}$ (each being a $1 \times B$ vector):

$$\mathcal{L}_{\text{cell}} = \frac{1}{|\mathcal{R}|}\sum_{r=1}^{|\mathcal{R}|} \|\hat{\mathbf{S}}'^{(r)} - \mathbf{S}'^{(r)}\|_2 \tag{33}$$

and therefore it effectively measures the similarity between predicted and true gene expression across cells in the batch for each gene in $\mathcal{R}$ (which is then averaged across genes in $\mathcal{L}_{\text{cell}}$).

The final training loss for expression prediction combines both axes:

$$\mathcal{L}_{\text{expression}} = \lambda_1 \mathcal{L}_{\text{gene}} + \lambda_2 \mathcal{L}_{\text{cell}} \tag{34}$$

This dual-axis reconstruction loss captures both gene-wise reconstruction fidelity within each cell and the consistency of expression patterns across cells for shared genes.

Dataset Classification Modeling. To disentangle dataset-specific technical effects from biological variation, we introduce an auxiliary dataset prediction task. Using the `[DS]` token embedding, the model predicts the dataset of origin:

$$\hat{d}^{(i)} = \text{MLP}_{\text{dataset}}(\mathbf{e}_{\text{ds}}^{(i)}) \tag{35}$$

with $\mathbf{e}_{\text{ds}}^{(i)}$ as in **Eq. 29**. We employ cross-entropy loss:

$$\mathcal{L}_{\text{dataset}} = \frac{1}{B}\sum_{b}^{B} \text{CrossEntropy}(\hat{d}^{(b)}, d^{(b)}) \tag{36}$$

where $d^{(b)}$ denotes the true dataset label for cell b in the batch, and $\hat{d}^{(b)}$ is the predicted label.

In our implementation, cells within the same AnnData file are treated as originating from the same dataset for this classification task. For our SE model trained on Arc scBaseCount, CZ CELLxGENE, and Tahoe-100M, this task becomes a multi-class classification problem over approximately 14,000 datasets. This auxiliary objective encourages the model to pool the relevant information in this token position, disentangling it from true biological signal. Empirically, since datasets in scBaseCount are organized by Sequence Read Archive experiment identifiers (SRX), which typically correspond to individual cell types or experimental conditions, the dataset token captures cell type and cell line information alongside technical batch effects. Since this token embedding is derived exclusively from the expression set representation without incorporating metadata, the trained model can be used at inference time without requiring explicit specification of technical parameters.

Total Loss. The SE model is trained using a combination of both losses:

$$\mathcal{L} = \mathcal{L}_{\text{expression}} + \mathcal{L}_{\text{dataset}}. \tag{37}$$

Table 1: Summary of datasets used in ST experiments, including total number of cells, perturbations, and biological contexts (e.g., donors, cell types, or conditions).

Dataset	# of Cells	# of Perturbations	# of Contexts
Replogle-Nadig	624,158	1,677	4
Jiang	234,845	24	30
Srivatsan	762,795	189	3
Mcfaline-Figueroa	354,758	122	3
Tahoe-100M	100,648,790	1138	50
Parse-PBMC	9,697,974	90	12/18 (Donors)/(Cell Types)

Table 2: Summary of datasets used in SE experiments, showing total number of cells.

Dataset	# Training Cells	# Validation Cells
Arc scBaseCount [32]	71,676,369	4,137,674
CZ CellXGene [31]	59,233,790	6,500,519
Tahoe-100M [30]	36,157,383	2,780,587

5.5. Datasets

5.5.1. Datasets Used for ST Training

We used several single-cell perturbation datasets in this study: Tahoe-100M dataset [30], the Replogle-Nadig dataset [4, 43], the Parse-PBMC dataset [42], the Jiang dataset [64], the McFaline dataset [65], and the Srivatsan dataset [41] (Table 1). All datasets were filtered to retain measurements for 19,790 human protein-coding Ensembl genes and subsequently normalized to a total UMI depth of 10,000. Raw count data were log-transformed using `scanpy.pp.log1p`. For analyses on highly variable genes (HVGs) throughout this work, for each dataset, the top 2,000 HVGs were identified using `scanpy.pp.highly_variable_genes`. Log-transformed expression values for these HVGs were used as gene-level features. PCA embeddings of cells were computed using `scanpy.pp.pca`.

5.5.2. Additional preprocessing for genetic perturbation datasets

Genetic perturbation datasets were further filtered to only retain perturbations with high knockdown efficacy using the `filter_on_target_knockdown` function from the Cell-Load package. The following three filtering steps were performed:

- **Perturbation-level filtering**: Retain only those perturbations (except controls) whose average knockdown efficiency meets a minimum threshold, residual expression is ≤ 0.30.
- **Cell-level filtering**: Within the selected perturbations, keep only those cells that individually meet a stricter knockdown threshold, residual expression ≤ 0.50.
- **Minimum cell count**: Drop any perturbations that have fewer than a specified number of remaining valid cells (30), while always preserving control cells.

5.5.3. Datasets Used for SE Training

SE was trained on 167 million human cells across the Arc scBaseCount [32], CZ CELL×GENE [31], Tahoe-100M [30] datasets (Table 2). To avoid data leakage in our context generalization benchmarks, we trained on 20 Tahoe-100M cell lines separate from the five held out cell lines from (**Fig. 2**). scBaseCount data was filtered to only retain cells with at least 1,000 non-zero expression measurements and 2,000 UMIs per cell. A subset of AnnData files were left out for computing validation loss.

Table 3: Key model hyperparameters by dataset.

Dataset	cell_set_size	hidden_dim	n_encoder_layers	n_decoder_layers	batch_encoder	transformer_backbone_key	attn_heads	params
Tahoe-100M	256	1488	4	4	false	LLaMA	12	244M
Parse-PBMC	512	1440	4	4	true	LLaMA	12	244M
Replogle-Nadig	32	128	4	4	false	GPT2	8	10M

Table 4: Architectural details for ST components. G denotes the number of genes (**Eq. 12**), E the dimension of cell embeddings (if used) (**Eq. 21**), D_{pert} the dimension of perturbation features (**Eq. 13**), D_{batch} the dimension of batch features (**Eq. 14**), and h the shared hidden dimension. All MLPs use GELU activation and Layer Normalization before activation, unless specified otherwise.

Component	Architecture	Layer Dimensions	Activation	Normalization	Dropout
f_{cell}	4-layer MLP	(G or E) $\to h \to h \to h \to h$	GELU	LayerNorm	None
f_{pert}	4-layer MLP	$D_{\text{pert}} \to h \to h \to h \to h$	GELU	LayerNorm	None
f_{batch}	Embedding Lyr.	$D_{\text{batch}} \to h$	N/A	N/A	None
f_{ST}	LLaMA Transf.	h (input to each of 4 layers)	SwiGLU	RMSNorm	None
f_{recon}	Linear Layer	$h \to G$ or E	N/A	N/A	None
f_{decode}	3-layer MLP	$h \to 1024 \to 512 \to G$	GELU	LayerNorm	0.1
f_{conf}	3-layer MLP	$h \to h/2 \to h/4 \to 1$	GELU	LayerNorm	None

5.6. Training

All models are implemented using PyTorch Lightning with distributed data parallel (DDP) training. We use PyTorch's automatic mixed precision (AMP) to reduce memory usage and accelerate training. During inference, genes not present in the embedding vocabulary are ignored.

5.6.1. ST Hyperparameters

The ST architecture utilizes a shared hidden dimension, denoted as h, across its modules. Each encoders maps its respective inputs to this h-dimensional space, which also serves as the internal dimension for the transformer layers. The core transformer module, f_{ST}, is either based on a LLaMA [66] backbone or a GPT2 [67] backbone, provided in HuggingFace. We use a GPT2 backbone for sparser datasets, such as Replogle-Nadig [4, 43], due to LayerNorm's desirable centering transform operations that revive dead neurons in low-data regimes [68]. All models are modified to use bi-directional attention. Dimensionality and parameterization of the backbone are provided in Table 3. Since cell order within each set is arbitrary, no positional encodings are used. Additionally, dropout is not applied within the transformer. Additionally, as we work directly in vector space, we deactivate the model parameters allocated for word embeddings. Before training, most ST weights are initialized sampling from Kaiming Uniform $w_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{1}{\text{fan\_in}}}, \sqrt{\frac{1}{\text{fan\_in}}}\right)$ [69] with the exception of the transformer backbone which is initialized from $\mathcal{N}(0, 0.02^2)$. Detailed configurations for each neural network module are summarized in Table 4.

5.6.2. ST Training Details

All components are trained end-to-end with using the objectives described previously. These components include the control cell encoder f_{cell}, perturbation encoder f_{pert}, optional batch encoder f_{batch}, transformer backbone f_{ST}, reconstruction and decoding heads f_{recon} and f_{decode}.

For the fine-tuning tasks (**Fig. 2G**), we initialize a new ST model using the pretrained weights while selectively reinitializing specific components that are dataset-specific. In particular, the perturbation encoder f_{pert} is reinitialized to enable transfer across perturbation modalities. If ST is trained using cell embeddings, the gene decoder f_{decode} is also reinitialized to adapt to differences in gene coverage between datasets. Since different experimental platforms and datasets typically measure distinct subsets of genes, the decoder must be retrained to map from the shared embedding space to the target dataset's specific gene expression space. All other components — f_{cell}, f_{batch}, and f_{ST} — retain their pretrained weights and are finetuned on the target dataset.

5.6.3. SE Hyperparameters

SE is a 600M parameter encoder-decoder model designed to learn cell representations by predicting gene expression variability. The encoder consists of 16 transformer layers, each with 16 attention heads and hidden dimension $h = 2048$. Each layer uses pre-normalization with a feed-forward network that expands to dimension $3 \times h$, and uses GELU activation. We apply dropout with probability 0.1 to both attention and feed-forward layers. The decoder is a multi-layer-perceptron (MLP), trained to recover gene expression given learned cell embeddings and target gene embeddings. We used the AdamW optimizer [70] with a maximum learning rate of 10^{-5}, weight decay of 0.01, and gradient clipping with zclip [71]. The learning rate schedule consisted of linear warmup for the first 3% of total steps, followed by cosine annealing to 30% of the maximum learning rate. Before training, all SE weights are initialized sampling from Kaiming Uniform.

5.6.4. SE Training Details

SE was trained on a large-scale corpus of 14,420 AnnData files spanning 167 million human cells across the Arc scBaseCount [32], CZ CELL×GENE [31], and Tahoe-100M [30] datasets for 4 epochs. To avoid data leakage, datasets were split into separate training and validation sets at the dataset level. To enable efficient training at scale, we utilize Flash Attention 2 [72], with mixed precision (bf16) training [73]. Training was distributed across 4 compute nodes, each with 8 NVIDIA H100 GPUs. The model was trained with an effective batch size of 3,072, using per-device batch size of 24 and gradient accumulation over 4 steps.

5.7. Evaluation

5.7.1. Perturbation Evaluation Metrics

A key goal for perturbation models is to distinguish between different perturbation effects. Cell-Eval evaluates this using several complementary metrics:

Perturbation Discrimination Score. Adapted from [14], this metric ranks predicted perturbed pseudobulk by their similarity to ground truth. The score is defined as the normalized rank of the ground truth from the predicted perturbed pseudobulk with respect to all ground truth perturbed pseudobulks. Specifically, let T be the number of distinct perturbations, for any perturbation t, let $\bar{p}_t$ and p_t denote the predicted and observed pseudobulked expressions, respectively. Using a distance $d(\cdot, \cdot)$ (here Manhattan or Euclidean), define

$$r_t \;=\; \sum_{t' \neq t} \mathbf{1}\big\{ d(\bar{p}_t,\, p_{t'}) \;<\; d(\bar{p}_t,\, p_t) \big\}, \qquad \text{(rank)}$$

i.e. the number of other perturbations whose ground-truth pseudobulked expression is closer to $\bar{p}_t$ than the correct one. The per-perturbation score is

$$\mathrm{PDisc}_t \;=\; \frac{r_t}{T}, \qquad \text{(normalized rank)}$$

ranging in $[0, 1)$ with $\mathrm{PDisc}_t = 0$ indicating a perfect match (no closer profiles) and values near 1 signaling poor discrimination. The overall score is the mean

$$\mathrm{PDisc} \;=\; \frac{1}{T} \sum_{t=1}^{T} \mathrm{PDisc}_t. \qquad \text{(overall)}$$

We report the normalized inverse perturbation discrimination score:

$$\mathrm{PDiscNorm} \;=\; 1 - 2\mathrm{PDisc}. \qquad \text{(normalized overall)}$$

so that a random predictor receives a score of 0.0, and a perfect predictor receives a score of 1.0.

5.7.2. Differential Expression

To evaluate biological relevance, CELL-EVAL performs a differential expression analysis using the Wilcoxon rank-sum test and adjusts for multiple hypotheses using the Benjamini-Hochberg procedure, applied to both observed values and model predictions in test cell lines.

Notation. Fix a perturbation $t \in [T] = \{1, \ldots, T\}$ and let G be the set of all genes. Define $G^{(k)}_{t,\text{true}}$ and $G^{(k)}_{t,\text{pred}}$ as the top-k significant DE genes ($p_{\text{adj}} < 0.05$) in ground truth and predictions, ranked by log-fold changes $|\Delta_{tg}|$, $G^{(\text{DE})}_{t,\text{true}}$ and $G^{(\text{DE})}_{t,\text{pred}}$ as the complete significant sets. All rank correlations are Spearman's, denoted by ρ_{rank}.

CELL-EVAL also evaluates model performance using many complementary metrics:

DE Overlap Accuracy For each perturbation, we identify the top k differentially expressed genes ($k = 50, 100, 200, N$), filtered by adjusted p-value and ranked by absolute log fold change. We compute the intersection between the predicted and true DEG sets and report the overlap as a fraction of k. When $k = N$ the k value varies by perturbation where N is the total number of differentially expressed genes in the true DEG set. When k is not specified, we are referring to DE Overlap at N. That is,

$$\text{Overlap}_{t,k} \;=\; \frac{\left|G^{(k)}_{t,\text{true}} \cap G^{(k)}_{t,\text{pred}}\right|}{k}. \tag{38}$$

Effect sizes. To compare the relative effect sizes of perturbations we calculate Spearman correlation coefficients for each perturbation on the number of differentially expressed genes (adjusted p-value < 0.05) between predicted and ground truth. This lets us assess whether models accurately capture relative effect sizes. Formally, for set sizes $n_t = |G^{(\text{DE})}_{t,\text{true}}|$, $\hat{n}_t = |G^{(\text{DE})}_{t,\text{pred}}|$, we compute

$$\text{SizeCorr} = \rho_{\text{rank}}\big((n_t)_{t=1}^{T},\, (\hat{n}_t)_{t=1}^{T}\big). \tag{39}$$

5.7.3. Cell Embedding Evaluation Metrics

We also evaluated cell embeddings using both intrinsic and extrinsic metrics (**Fig. 2D**).

Intrinsic Evaluation. To assess the extent to which embeddings capture perturbation-specific information, we trained a multilayer perceptron (MLP) with one hidden layer to classify perturbation labels from cell embeddings, measuring AUROC and accuracy. This classification task was performed separately for each cell line, with data partitioned such that 20% of cells from each perturbation group were randomly assigned to the test set and the remaining 80% to the training set. The MLP was trained using a cross-entropy loss. High classification performance indicates that embeddings maintain distinct representations of perturbation-induced states.

Extrinsic Evaluation. To evaluate the downstream utility of various cell embeddings, we compared the performance of ST when trained with different embedding spaces. This approach captures the idea that embeddings rich in cellular information may not always be optimal for modeling perturbation effects. The ideal embedding space should balance both: it should preserve perturbation-specific information while supporting accurate prediction of perturbation-induced transitions across cell states.

5.7.4. Other Evaluations

Cell Set Scaling. To evaluate the impact of cell set size in STATE, we ablated the number of cells per set and the batch size, such that the batch size times the cell set size was always 16,384, and measured the validation loss as a function of floating-point operations (FLOPs) (**Fig. 1C**). These models were also compared to a pseudobulk model that replaces the self-attention in STATE with mean pooling, and masked attention version of STATE, where each cell can only attend to itself. As the cell set size increased, the validation loss improved significantly on held out cell lines, up to an optimal value of 256.

5.7.5. Baseline Models

We describe and formalize various baseline models used as comparisons to STATE.

Perturbation Mean Baseline. This baseline predicts a perturbed expression profile as the control mean for that cell context plus a global perturbation offset learned from the training data. For each cell type c and perturbation p we first compute cell-type–specific means

$$\boldsymbol{\mu}_c^{\text{ctrl}} = \frac{1}{|\mathcal{C}_c|}\sum_{i\in\mathcal{C}_c}\mathbf{x}^{(i)}, \qquad \boldsymbol{\mu}_{c,p}^{\text{pert}} = \frac{1}{|\mathcal{P}_{c,p}|}\sum_{i\in\mathcal{P}_{c,p}}\mathbf{x}^{(i)},$$

where $\mathcal{C}_c$ is the set of control cells of type c and $\mathcal{P}_{c,p}$ the perturbed cells of type c receiving perturbation p. Their difference is the cell-type offset $\boldsymbol{\delta}_{c,p} = \boldsymbol{\mu}_{c,p}^{\text{pert}} - \boldsymbol{\mu}_c^{\text{ctrl}}$. Averaging across all cell types that contain p yields a global offset

$$\boldsymbol{\delta}_p \;=\; \frac{1}{|\mathcal{C}_p|}\sum_{c\in\mathcal{C}_p}\boldsymbol{\delta}_{c,p}, \qquad \mathcal{C}_p = \{c \mid |\mathcal{P}_{c,p}| > 0\},$$

Given a test cell type t and a perturbation label p the model outputs

$$\hat{\mathbf{x}} = \boldsymbol{\mu}_t^{\text{ctrl}} + \boldsymbol{\delta}_p, \qquad \boldsymbol{\delta}_{\text{ctrl}} \equiv \mathbf{0}.$$

Thus, controls are reproduced exactly, while every non-control perturbation receives the same global shift.

Context Mean Baseline. This baseline predicts a cell's post-perturbation profile by returning the average perturbed expression of *cells of the same cell type* observed in the training set. For every cell type c we collect all training cells whose perturbation is not the control and form the pseudo-bulk mean

$$\boldsymbol{\mu}_c \;=\; \frac{1}{|\mathcal{T}_c|}\sum_{i\in\mathcal{T}_c}\mathbf{x}^{(i)}, \qquad \mathcal{T}_c \;=\; \big\{\, i \;\big|\; \text{cell\_type}(i) = c,\; p^{(i)} \neq \text{ctrl}\big\}.$$

At inference time, for a test cell i with cell type $c^{(i)}$ and perturbation label $p^{(i)}$ we predict

$$\hat{\mathbf{x}}^{(i)} \;=\; \begin{cases} \mathbf{x}^{(i)} & p^{(i)} = \text{ctrl}, \\ \boldsymbol{\mu}_{c^{(i)}} & p^{(i)} \neq \text{ctrl}. \end{cases}$$

i.e. controls are passed through unchanged, whereas perturbed cells inherit their cell-type mean.

Linear Baseline. This baseline treats a perturbation as a low-rank, gene-wide linear displacement that is added to each cell's own control expression [46]. Let $G \in \mathbb{R}^{G\times d_g}$ be a fixed gene-embedding matrix (e.g. pretrained protein feature vectors, one row per gene) and $P \in \mathbb{R}^{P\times d_p}$ a fixed perturbation-embedding matrix (one row per perturbation, one-hot). From the training set we first build an "expression-change" pseudo-bulk

$$Y_{g,p} \;=\; \frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\big(x_g^{\text{pert},(i)} - x_g^{\text{ctrl},(i)}\big), \qquad \mathcal{P}_p = \big\{\, i \mid p^{(i)} = p\big\}, \tag{40}$$

so $Y \in \mathbb{R}^{G\times P}$ stores, for every gene g and perturbation p, the average change relative to that cell's matched control. The model seeks a low-rank map $K \in \mathbb{R}^{d_g\times d_p}$ and a gene-wise bias $\mathbf{b} \in \mathbb{R}^G$ such that

$$Y \;\approx\; GKP^\top + \mathbf{b}\,\mathbf{1}^\top. \tag{41}$$

We obtain K in a single shot by solving the ridge-regularized least-squares problem

$$\min_K \big\| Y - GKP^\top - \mathbf{b}\,\mathbf{1}^\top\big\|_F^2 \;+\; \lambda\|K\|_F^2, \qquad \mathbf{b} = \frac{1}{P}\,Y\mathbf{1}, \tag{42}$$

whose closed-form solution is

$$K \;=\; \big(G^\top G + \lambda I\big)^{-1} G^\top Y P \,\big(P^\top P + \lambda I\big)^{-1}. \tag{43}$$

No gradient-based optimization is required, as once K is computed the model is fixed. For a test cell i with its own control profile $\mathbf{x}^{\mathrm{ctrl},(i)}$ and perturbation label $p^{(i)}$ the prediction is

$$\hat{\mathbf{x}}^{(i)} \;=\; \begin{cases} \mathbf{x}^{\mathrm{ctrl},(i)} & p^{(i)} = \mathrm{ctrl}, \\ \mathbf{x}^{\mathrm{ctrl},(i)} + GK\, P_{p^{(i)}} + \mathbf{b} & p^{(i)} \neq \mathrm{ctrl}, \end{cases} \tag{44}$$

where $P_{p^{(i)}}$ denotes the row of P corresponding to perturbation $p^{(i)}$. Thus each prediction preserves the cell's basal state exactly and adds a perturbation-specific, low-rank shift learned from the training cohort, with model capacity controlled solely by the embedding dimensions and the ridge parameter λ.

Deep Learning Baselines. We benchmark STATE against several baselines that leverage related deep learning architectures for predicting perturbation effects across diverse cell contexts. These include two autoencoder-based models, scVI [25] and CPA [8], as well as the transformer-based scGPT model [11]. scVI models gene expression distributions while accounting for technical noise and batch effects. CPA learns a compositional latent space that captures the additive effects of perturbation, dosage, and cell type. scGPT leverages generative pretraining on over 33 million cells to support zero-shot generalization across tasks including perturbation prediction.

6. Analysis: State Transition as Optimal Transport

In this section, we analyze the theoretical capacity of State Transition (ST) in learning optimal transport mappings across cellular distributions. First, we prove that the solution family of ST covers the optimal transport map between control and perturbed cell distributions. Second, we derive explicit constraints on ST for learning the unique continuous optimal transport map. Finally, we discuss the implications of the theoretical analysis and possible future directions.

6.1. ST Asymptotic Behavior and Solution Family

Optimal Transport (OT) provides a framework for comparing and aligning probability distributions by finding a mapping or coupling that minimizes a certain fixed cost. Traditional OT methods explicitly define a cost function and solve a linear program or iterative algorithm to find an optimal coupling or map [74, 75]. The advent of deep learning has given rise to Neural Optimal Transport (Neural OT), which leverages neural networks to parameterize and solve OT problems [9, 27].

Neural OT methods [27] typically operate by either: [label=(0)]

Parameterizing the OT map T directly using a neural network designed to enforce OT properties;

Parameterizing the OT coupling matrix ($\mathbf{P}$) or its dual potentials (e.g., α, β) with neural networks;

Explicitly minimizing an OT cost function (or its dual objective) as the primary loss. For instance, CellOT [9] explicitly parameterizes the convex potentials of the dual optimal transport problem using Input Convex Neural Networks (ICNNs) to learn an optimal transport map $T_k = \nabla g_k$ for each perturbation k, with g_k one of the dual potentials.

ST, while not explicitly solving an OT problem in the traditional sense, performs a task related to Neural OT: learning a transformation that aligns unperturbed and perturbed cell distributions. This transformation is learned, not explicitly engineered for a fixed cost. However, motivated by rigorous theoretical work that demonstrates how transformers with fixed parameters can provably solve optimal transport by implementing gradient descent through engineered prompts [76], we demonstrate that, in an asymptotic setting, the solution family of our trained and optimized ST contains the unique continuous optimal transport map between cellular distributions when regularity assumptions on the distributions are met.

For this analysis, we focus on a single training mini-batch, denoted as the b-th batch. We start by describing the mathematical formulation of ST, which aims to learn the effect of one single perturbation from cell population $\mathbf{X}^{(b)}_{\text{ctrl}} \in \mathbb{R}^{S\times G}$ to $\mathbf{X}^{(b)}_{\text{pert}} \in \mathbb{R}^{S\times G}$. Recall that the input of the transformer for this batch is constructed by summing the control cell embeddings with the broadcasted perturbation and batch embeddings, $\mathbf{H}^{(b)} = \mathbf{H}^{(b)}_{\text{cell}} + \mathbf{H}^{(b)}_{\text{pert}} + \mathbf{H}^{(b)}_{\text{batch}} \in \mathbb{R}^{S\times d}$ (**Eq. 15**). The output of the transformer is computed as $\mathbf{O}^{(b)} = \mathbf{H}^{(b)} + f_{\text{ST}}(\mathbf{H}^{(b)}) \in \mathbb{R}^{S\times d}$, where f_{ST} is the transformer encoder (**Eq. 16**). We consider f_{ST} with L layers, where the input to the first layer is $\mathbf{H}^0 = \mathbf{H}^{(b)}$:

$$f_{\text{ST}} = F_L \circ \cdots \circ F_1, \quad \mathbf{H}^i = F_i(\mathbf{H}^{i-1}) = (G + \text{Id})\left(\mathbf{H}^{i-1} + A^i(\mathbf{H}^{i-1}, \mathbf{H}^{i-1})V^i(\mathbf{H}^{i-1})\right). \tag{45}$$

Here A^i denotes the multi-head attention, V^i denotes the attention value encoder, and $G + \text{Id}$ denotes the feedforward layer combined with the residual connection. The layer normalization function is absorbed in each constructed function. Finally, for the b-th batch, a cell-wise linear transform defines the final output $\hat{\mathbf{X}}^{(b)}_{\text{target}}$ for the S cells (**Eq. 17**). We can equivalently rewrite this for each cell s as:

$$\hat{\mathbf{X}}^{(b)}_{\text{pert, s}} = f_{\text{recon}}(\mathbf{O}^{(b)}) = \mathbf{O}^{(b)}_s \mathbf{W}_{\text{recon}} + \mathbf{b}_{\text{recon}} \in \mathbb{R}^G, \tag{46}$$

where $\hat{\mathbf{X}}^{(b)}_{\text{pert, s}}$ denotes the s-th row of $\hat{\mathbf{X}}^{(b)}_{\text{pert}}$, containing the predicted state of cell s, obtained by its embedding $\mathbf{O}^{(b)}_s$ which denotes the s-th row of $\mathbf{O}^{(b)}$.

The constants $\mathbf{H}^{(b)}_{\text{ST}} + \mathbf{H}^{(b)}_{\text{batch}}$, as well as the transforms $f_{\text{cell}}, f_{\text{recon}}$ can be absorbed in the first and last layers of f_{ST}, which we therefore define as $f^{\text{pert,batch}}_{\text{ST}}$. Therefore, we can equivalently rewrite the predicted

state of cell s as:

$$\hat{\mathbf{X}}^{(b)}_{\text{pert},s} = \left[\left(f_{\text{recon}} \circ f_{\text{cell}} + f^{\text{pert,batch}}_{\text{sets}} \right) (\mathbf{X}^{(b)}_{\text{ctrl}}) \right]_s, \tag{47}$$

where $[\cdot]_s$ indicates that we access the s-th row of the obtained output. To simplify our analysis, we focus on the asymptotic setting where the cell set size S tend to infinity. Recall that each cell is sampled from the distribution $\mathcal{D}_{\text{ctrl}}$. Then, as the cell number $S \to \infty$ with certain regularity conditions, the output for a single cell $\hat{\mathbf{X}}^{(b)}_{\text{pert},s}$ depends solely on $\mathbf{X}^{(b)}_{\text{ctrl,s}}$ itself and the overall distribution $\mathcal{D}_{\text{ctrl}}$, because the attention mechanism effectively processes information from the entire distribution. Thus, the limiting operator can be defined per cell as:

$$\hat{\mathbf{X}}^{(b)}_{\text{pert, s}} = \left[\left(f_{\text{recon}} \circ f_{\text{cell}} + f^{\text{pert,batch}}_{\text{ST},\mathcal{D}_{\text{ctrl}}} \right) (\mathbf{X}^{(b)}_{\text{ctrl}}) \right]_s := F_{\text{pert,batch}}(\mathbf{X}^{(b)}_{\text{ctrl},s}). \tag{48}$$

More detailed treatments are provided in recent works considering the limiting mean-field dynamics of transformers [77, 78]. Notably, **Eq. 48** holds for an arbitrary cell s sampled from the distribution $\mathcal{D}_{\text{ctrl}}$. The operator $\mathbf{F}_{\text{pert,batch}}$ defines a mapping from $\mathcal{D}_{\text{ctrl}}$ to $\hat{\mathcal{D}}_{\text{pert}}$. We further assume that the transformer model is expressive enough, such that minimizing the empirical kernel MMD objective is possible: $\widehat{\text{MMD}}(\hat{\mathbf{X}}_{\text{pert}}, \mathbf{X}_{\text{pert}}) = 0$ with energy kernel $k(u, v) = -\|u - v\|_2$. Our following lemma shows that zero empirical MMD indicates distributional matching with probability 1.

Lemma 1 (Distributional matching via Empirical MMD)**.** *Suppose that the supports of $\hat{\mathcal{D}}_{pert}, \mathcal{D}_{pert}$ are bounded. When the cell number $S \to \infty$, if the empirical MMD reaches zero, that is, $\widehat{MMD}(\hat{\mathbf{X}}^{(b)}_{pert}, \mathbf{X}^{(b)}_{pert}) = 0$, then with probability 1, we have*

$$\hat{\mathcal{D}}_{pert} = \mathcal{D}_{pert}. \tag{49}$$

Reversely, if $\hat{\mathcal{D}}_{pert} = \mathcal{D}_{pert}$, then with probability 1, we have $MMD(\hat{\mathbf{X}}^{(b)}_{pert}, \mathbf{X}^{(b)}_{pert}) = 0$.

Proof. The empirical MMD used as the ST objective is defined as

$$\widehat{\text{MMD}}^2(\hat{\mathbf{X}}^{(b)}_{\text{pert}}, \mathbf{X}^{(b)}_{\text{pert}}) = \frac{1}{S^2} \sum_{i=1}^{S} \sum_{j=1}^{S} k(\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{x}}^{(j)}) + k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - 2k(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(j)}). \tag{50}$$

Here we use $\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{x}}^{(j)}, \mathbf{x}^{(i)}, \mathbf{x}^{(j)}$ to denote cells in $\hat{\mathbf{X}}^{(b)}_{\text{pert}}, \mathbf{X}^{(b)}_{\text{pert}}$ respectively. The corresponding theoretical MMD on distributions $\hat{\mathcal{D}}_{\text{pert}}, \mathcal{D}_{\text{pert}}$ is defined as

$$\text{MMD}^2(\hat{\mathcal{D}}_{\text{pert}}, \mathcal{D}_{\text{pert}}) = \mathbb{E}_{x,x' \sim \hat{\mathcal{D}}_{\text{pert}}}[k(x, x')] + \mathbb{E}_{y,y' \sim \mathcal{D}_{\text{pert}}}[k(y, y')] - 2\mathbb{E}_{x \sim \hat{\mathcal{D}}_{\text{pert}}, y \sim \mathcal{D}_{\text{pert}}}[k(x, y)]. \tag{51}$$

The empirical MMD is a V-statistic [57], thus when the support sets are bounded (thus the kernel value k is also bounded), by the result of SLLN for V-statistics [79, 80], we have

$$\widehat{\text{MMD}}^2(\hat{\mathbf{X}}^{(b)}_{\text{pert}}, \mathbf{X}^{(b)}_{\text{pert}}) \overset{a.s.}{\to} \text{MMD}^2(\hat{\mathcal{D}}_{\text{pert}}, \mathcal{D}_{\text{pert}}). \tag{52}$$

With the energy kernel $k(x, y) = -\|x - y\|_2$, the theoretical MMD coincides with the energy distance D^2 between distributions:

$$\text{MMD}^2(\hat{\mathcal{D}}_{\text{pert}}, \mathcal{D}_{\text{pert}}) \equiv D^2(\hat{\mathcal{D}}_{\text{pert}}, \mathcal{D}_{\text{pert}}). \tag{53}$$

Finally, zero energy distance implies equal distributions and vice versa [81]. □

We define the continuous solution families that achieves zero empirical MMD and distributional matching as

$$\begin{aligned} \hat{\mathcal{F}} &= \{F \in \mathbb{C} | \widehat{\text{MMD}}(\hat{\mathbf{X}}^{(b)}_{\text{pert}}, \mathbf{X}^{(b)}_{\text{pert}}) = 0 \text{ and } \hat{\mathbf{X}}^{(b)}_{\text{pert},s} = F(\hat{\mathbf{X}}^{(b)}_{\text{ctrl},s})\}, \\ \mathcal{F} &= \{F \in \mathbb{C} | F(\mathcal{D}_{\text{ctrl}}) = \mathcal{D}_{\text{pert}}\}. \end{aligned} \tag{54}$$

Then, by Lemma 1, we have

$$\hat{\mathcal{F}} = \mathcal{F} \text{ with probability 1.} \tag{55}$$

To simplify our analysis, we assume that our model is expressive enough to learn any element in $\hat{\mathcal{F}}$, which is identical to $\mathcal{F}$ with probability 1. We next demonstrate that, under mild regularity conditions, the unique optimal transport mapping from $\mathcal{D}_{\text{ctrl}}$ to $\mathcal{D}_{\text{pert}}$ is within the solution family $\mathcal{F}, \hat{\mathcal{F}}$.

Theorem 2 (Optimal Transport Mapping within the Solution Family of STATE)**.** *Assume the densities of $\mathcal{D}_{ctrl}, \mathcal{D}_{pert}$ are absolute continuous and bounded, and the support sets are strictly convex and compact with $\mathbb{C}^2$ boundary. Then the continuous optimal transport map T from $\mathcal{D}_{ctrl}$ to $\mathcal{D}_{pert}$ associated with the squared distance cost $c(x, y) = \|x - y\|^2$ satisfies $T \in \mathcal{F}, \hat{\mathcal{F}}$ with probability 1.*

Proof. Under the regularity conditions posed for distributions $\mathcal{D}_{\text{ctrl}}, \mathcal{D}_{\text{pert}}$, applying Caffarelli's theorem [82] shows that there exists an continuous and differentiable optimal transport map $T \in \mathbb{C}^1$ between $\mathcal{D}_{\text{ctrl}}, \mathcal{D}_{\text{pert}}$. Thus by definition, $T \in \mathcal{F}$. Furthermore by Lemma 1, $\mathcal{F} = \hat{\mathcal{F}}$ with probability 1, in which case we also have $T \in \hat{\mathcal{F}}$. □

While the theorem shows the existence of the continuous optimal transport map T and that it is covered by the solution family $\hat{\mathcal{F}}$, $\hat{\mathcal{F}}$ contains infinite additional elements beyond T, thus the optimal transport solution is not guaranteed to be learned by the model. This inherent flexibility is advantageous for STATE, as biological transformations often involve complexities and unmodeled variations that may not align perfectly with a single, fixed optimal transport solution. Nevertheless, in the next theorem, we show that if we impose constraints on the Jacobian in the ST model objective, then the optimal transport mapping is guaranteed to be learned:

Theorem 3 (Constrained ST Model for Unique OT Map)**.** *Under the same assumptions as Theorem 2, consider the constrained solution family:*

$$\begin{aligned}\hat{\mathcal{F}}^* = \{F \in \mathbb{C}^1 | \widehat{MMD}(\hat{\mathbf{X}}^{(b)}_{pert}, \mathbf{X}^{(b)}_{pert}) = 0 \text{ and } \hat{\mathbf{X}}^{(b)}_{pert,s} = F(\hat{\mathbf{X}}^{(b)}_{ctrl,s});\\ J_{\mathbf{F}} = \left\{\frac{\partial \mathbf{F}_i}{x_j}\right\}_{ij} \text{ is symmetric and semi-positive definite}\}.\end{aligned} \tag{56}$$

Then $\hat{\mathcal{F}}^*$ uniquely contains the continuous OT mapping T with probability 1: $\hat{\mathcal{F}}^* = \{T\}$.

Proof. By Breiner's theorem [83], T is the unique continuous optimal transport mapping regarding the quadratic cost, if and only if it is the gradient of a convex function ψ: $T = \nabla\psi$. As we already have $T \in \mathbb{C}^1$ by Caffarelli's theorem, the condition is equivalent to symmetric and semi-positive definite Jacobian in T. Further by Lemma 1, $T \in \hat{\mathcal{F}}^*$ with probability 1, and the uniqueness of T implies $\hat{\mathcal{F}}^* = \{T\}$. □

Our theoretical analysis can be seen as a corollary of Caffarelli and Brenier's theorems on continuous optimal transport maps with quadratic cost. In particular, the final theorem hints that, we can enforce the model to learn an optimal transport mapping from $\mathcal{D}_{\text{ctrl}}$ to $\mathcal{D}_{\text{pert}}$, through imposing adequate regularizations on the Jacobian of the mapping. More precise constraints on the Jacobian may require additional architecture designs that explicitly achieve Jacobian constraints, or directly retrieve gradients of convex neural networks [9, 84].

Despite the large feasible solution set that contains numerous overfitting solutions, neural network models have shown remarkable generalization capacities. An important aspect on neural network generalization capabilities is implicit bias [85–87], which means gradient descent favor certain solutions, for instance, those minimally shifted compared with initial weights [86]. Therefore we hypothesize that the optimization of ST may lead to solutions with smallest "overall shifts" from $\mathcal{D}_{\text{ctrl}}$ to $\mathcal{D}_{\text{pert}}$, thus resembling a optimal transport mapping with some unknown cost function. Theoretical characterization of the implicit bias in the transition model is out of the scope here and remains promising future research.