SIMULTANEOUS LEARNING FROM BULK AND SINGLE-CELL EXPRESSION DATA WITH PERCEIVER-BASED MODELS

Rafał Powalski[1], **Albert Roethel**[1], **Joanna Krawczyk**[1],
Przemysław Pietrzak[1], **Łukasz Smoliński**[1], **Maciej Sypetkowski**[1],
Paweł Biernat[1], **Dariusz Plewczyński**[2], **Tomasz Jetka**[1]

[1]Ingenix, Warsaw, Poland
[2]Warsaw University of Technology, Warsaw, Poland

ABSTRACT

In recent years, genomic experiments have generated vast amounts of expression data, including bulk RNA-seq and single-cell RNA-seq datasets, spanning both observational and perturbational studies. However, existing models have not fully leveraged this diverse data landscape, focusing on modeling either bulk or single-cell expression data. We present Funomics T0, the first foundation model that can simultaneously learn from bulk RNA-seq and single-cell RNA-seq datasets from observational and perturbational studies. The proposed Perceiver-based model produces latent representations of the expression data that can be further used for various downstream tasks. We evaluate our model on perturbation prediction and tissue annotation tasks, using a comprehensive benchmark suite and demonstrating strong performance across metrics, with Funomics T0 outperforming the State model on multiple perturbation metrics.

1 INTRODUCTION

Next generation sequencing genomic platforms have resulted in the availability of vast amounts of expression data across both bulk and single-cell RNA sequencing modalities. Bulk RNA-seq captures averaged gene expression profiles across heterogeneous mixtures of cells in a tissue or sample, providing an overview of transcriptional activity at the population level, at a small cost. Single-cell RNA-seq, in contrast, quantifies gene expression at the resolution of individual cells, enabling the study of cellular heterogeneity and rare cell populations. These data can originate from observational studies, where native biological variation is explored, or from perturbational experiments designed to measure responses to genetic perturbations, chemical compounds, or other interventions.

The data can be used for various downstream tasks crucial for medical diagnosis and drug discovery, including cell type classification, survival analysis (Wissel et al., 2025), and prediction of perturbation responses (Adduri et al., 2025; Bunne et al., 2023). While these problems can be approached by using small task-specific models, recent advances in foundation models and language modeling have shown that models that can learn from all available data can achieve better performance on various tasks (Brown et al., 2020).

This motivates the recent development of foundation models for transcriptomics that can learn from the vast amount of expression data. This includes foundation models for single-cell RNA-seq data resembling language models, such as scGPT (Cui et al., 2024), scBERT (Yang et al., 2022); models for regulatory inference and gene prioritization (Theodoris et al., 2023; Kalfon et al., 2025); post-perturbation response prediction for genetic or chemical treatments (Adduri et al., 2025; Yang et al., 2024; Wang et al., 2024; Littman et al., 2025); and representation learning for retrieval, zero-shot annotation, or dataset alignment (Rosen et al., 2024; Heimberg et al., 2025). Foundation models for single-cell transcriptomics are primarily based on the transformer architecture (Cui et al., 2024; Yang et al., 2022; Theodoris et al., 2023; Kalfon et al., 2025; Hao et al., 2024; Zeng et al., 2025; Rosen

et al., 2024; Yang et al., 2024; Tejada-Lapuerta et al., 2025) with several exceptions based on dense vector or graph representation of the gene expression data (Heimberg et al., 2025).

However, while bulk RNA-seq data is often used for survival analysis (Wissel et al., 2025) and single-cell RNA-seq data is used for cell type classification and perturbation prediction (Adduri et al., 2025; Bunne et al., 2023), existing models focus on a single modality. Recent works such as scFoundation (Hao et al., 2024) and CancerFoundation (Theus et al., 2024) have explored the generalizability of single-cell foundation models to bulk RNA data, but no model jointly learns from both modalities across diverse tasks. This is the gap that we aim to fill with Funomics T0.

1.1 CONTRIBUTIONS

We present Funomics T0, a foundation model for transcriptomic modeling that can simultaneously learn from both bulk RNA-seq and single-cell RNA-seq datasets from observational and perturbational studies. Our model learns latent representations of the expression data which can be further used for various downstream tasks. We demonstrate the capabilities of our model on perturbation prediction tasks and tissue annotation tasks.

2 METHODS

In this section, we outline the key architectural choices and algorithms behind the Funomics T0 model, which similarly to prior work (Connell et al., 2022; Litman et al., 2025), is based on the Perceiver architecture (Jaegle et al., 2021b;a) and employs the set transformer (Lee et al., 2019) adaptation of cross-attention models widely used in natural language processing following the T5 architecture (Raffel et al., 2023; Zhang et al., 2025a).

The gene expression profiles of either single-cell or bulk RNA-seq are high-dimensional vectors $X \in \mathbb{R}^G$, for $G \approx 19{,}000$ protein-coding genes. Despite the high dimensionality, the profiles contain substantial biological redundancy due to the functional structure, such as co-regulated pathways, transcriptional programs, and correlated expression patterns (Baek et al., 2025).

If most biologically meaningful variation is concentrated in lower-dimensional expression programs, then a compact representation can faithfully capture underlying structure, while removing the random noise inherent in experimental measurements. To realize this, Funomics T0 introduces a set of $N_{\text{latent}} \ll G$ learnable latent tokens that serve as a compressed, information-aggregating representation.

This design follows the Perceiver architecture (Jaegle et al., 2021b), in which the model backbone (see Figure 1) consists of a condition encoder and a latent decoder with cross-attention. Gene expression inputs are treated as unordered sets, without positional encodings, reflecting the fact that gene order has no biological meaning. Condition information (*e.g.* perturbation identity, dosage, or metadata) is embedded and processed through a bidirectional self-attention encoder, allowing interactions among condition descriptors. The decoder applies self-attention over the latent tokens, followed by cross-attention to both gene embeddings and condition embeddings, producing compressed latent states $L \in \mathbb{R}^{N_{\text{latent}} \times d}$. These latent states are then projected back to gene space via projection cross-attention, producing gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G \times d}$ as visible in Figure 2.

We note that compared to models based on the classical transformer architecture of Vaswani et al. (2017), the complexity of processing the expression representation is reduced from $\mathcal{O}(G^2)$ operations for naive self-attention to $\mathcal{O}(G \cdot N_{\text{latent}})$ for the cross-attention based approach. The latent array thus captures structured biological signals while enabling efficient training and inference on full-transcriptome data.

The gene-level hidden states processed by the backbone serve as a shared representation used by lightweight task-specific heads. Different tasks attach their own projection and decoding layers, enabling Funomics T0 to learn from different tasks simultaneously. The detailed forward pass algorithm is provided in Appendix A.

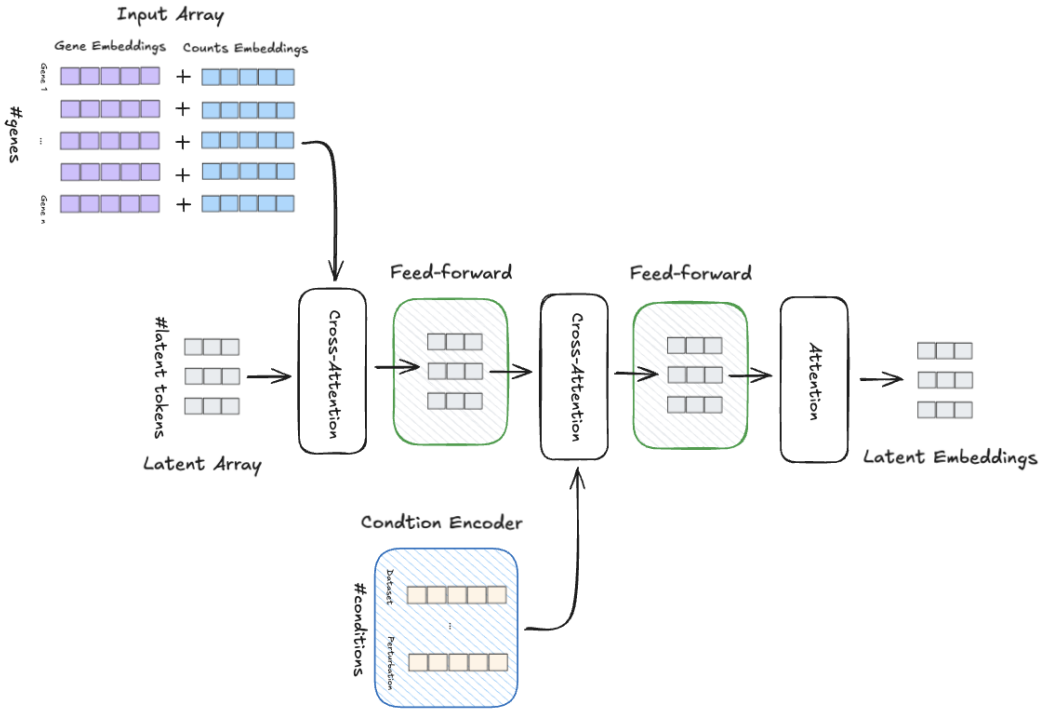

Figure 1: Funomics T0 backbone architecture. The model processes gene expression data through embedding layers, encodes conditions, and uses a decoder with cross-attention to produce compressed latent representations.

2.1 Backbone Architecture

The backbone of the Funomics T0 model is composed of two primary modules: a condition encoder and a decoder utilizing cross-attention iteratively enhancing the representation of the latent tokens to capture the essential information from the high-dimensional gene expression profile through cross-attention. This architecture employs a set transformer approach (Lee et al., 2019). As a result, the model is inherently permutation-invariant, which aligns well with gene expression data where the ordering of genes is arbitrary. A simplified representation of the backbone architecture is shown in Figure 1.

The condition encoder processes condition embeddings $E_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$ using bidirectional self-attention, allowing each condition token to attend to all other condition tokens. This enables the encoder to capture relationships between different condition components (*e.g.*, perturbation target gene, dosage, task type) and produce encoded condition representations $H_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$.

The decoder operates on a fixed set of N_{latent} decoder input embeddings, where N_{latent} is chosen to be much smaller than G ($N_{\text{latent}} \ll G$), achieving significant dimensionality reduction. These decoder inputs are obtained by embedding a fixed vocabulary of N_{latent} tokens through learned embedding lookups. The decoder uses self-attention among the latent tokens to enable information exchange, followed by cross-attention to both gene embeddings and encoded condition embeddings. Specifically, gene embeddings $E_{\text{gene}} \in \mathbb{R}^{G \times d}$ obtained through the gene token embedding module (see H.1) and counts embeddings $E_{\text{counts}} \in \mathbb{R}^{G \times d}$ through the count embedding module (see H.2) are combined as $E_{\text{input}} = E_{\text{gene}} + E_{\text{counts}}$ and serve as input to the decoder's cross-attention mechanism. The decoder also receives encoded condition embeddings H_{cond} through encoder-decoder cross-attention, allowing the model to condition its predictions on the specific experimental context.

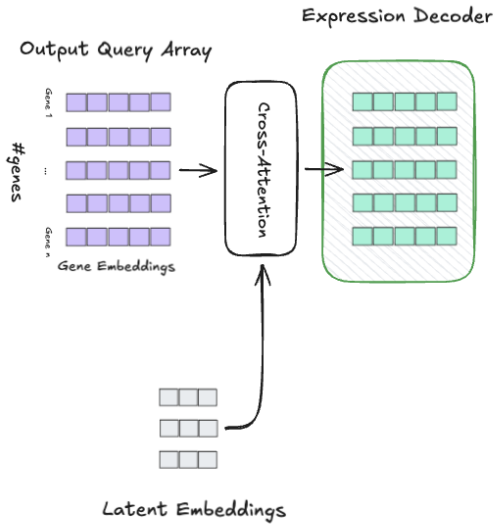

Figure 2: Funomics T0 gene expression decoder. The model uses a linear projection to produce gene-level expression values $\hat{X} \in \mathbb{R}^{B\times G}$ from gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G\times d}$.

The decoder produces compressed latent states $L \in \mathbb{R}^{N_{\text{latent}}\times d}$, which capture the essential information from the high-dimensional gene expression profile while remaining computationally tractable. These latent states are then projected back to gene space via projection cross-attention, producing gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G\times d}$ that serve as input to task-specific heads. This projection step enables task heads to operate on full-gene representations while maintaining the computational efficiency of the compressed latent processing. The dimensionality reduction achieved by the decoder is crucial for handling the large number of genes ($G \approx 19{,}000$) efficiently. Architectural details of the backbone components are provided in Appendix E.

2.1.1 Employed Embeddings

Given expression counts $X \in \mathbb{R}^{B\times G}$ for a batch of B cells with G genes, we construct gene embeddings and counts embeddings independently. Gene embeddings $E_{\text{gene}} \in \mathbb{R}^{G\times d}$ are learnable token embeddings that provide unique representations for each gene, where d is the embedding dimension. counts embeddings $E_{\text{counts}} \in \mathbb{R}^{G\times d}$ encode expression values using either a binning approach or an MLP encoder. These embeddings are combined as $E_{\text{input}} = E_{\text{gene}} + E_{\text{counts}}$ to form the input representation for the decoder.

Experimental conditions are encoded as separate embeddings that condition the model's predictions. Condition embeddings handle two distinct cases: genetic perturbations, where target genes are embedded using the same gene token embedding procedure, and chemical perturbations, where molecular embeddings are combined with dosage and optionally the gene targets of a molecule are concatenated.

Task embeddings encode the task type (*e.g.*, perturbation prediction, masked gene prediction, annotation, drug sensitivity). All condition embeddings are concatenated into a condition sequence $E_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}}\times d}$, where N_{cond} varies based on the perturbation type and task.

Detailed procedures and mathematical formulations for gene token embeddings, counts embeddings, and condition embeddings (including genetic and chemical perturbation embeddings) are provided in Appendix H.

2.2 PREDICTION HEADS

Gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G \times d}$ from the Funomics T0 backbone (obtained via projection cross-attention from compressed latent states) are processed by a collection of modular, task-specific prediction heads, each head adapting the backbone-generated compressed representations to the output requirements of its task.

The model employs specialized heads for different tasks: a gene expression decoder for perturbation prediction (producing $\hat{X} \in \mathbb{R}^{B \times G}$ from H_{genes} via linear projection), classification heads that map aggregated gene states to class probabilities ($\hat{y} \in [0,1]^{B \times C}$), and regression heads for continuous scores ($\hat{s} \in \mathbb{R}^{B}$). Multi-task heads can output several predictions jointly. Further architectural details are provided with each task in Appendix M.

2.3 TASK FORMULATIONS

The Funomics T0 model is designed to handle multiple tasks through a unified architecture. Each task is formulated as a specific prediction problem that extends the shared backbone with task-specific prediction heads (see subsection 2.2). The following subsections describe the formulation for each task category. Losses corresponding to each task (see also Appendix J) are then combined into the total loss function, used to train the model.

2.3.1 PERTURBATION PREDICTIONS

In the genetic perturbation prediction task, the model predicts perturbed cell expression profiles given control cell counts and genetic perturbation conditions. For a batch of B cell groups, each containing N cells, the model receives control counts $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$ and produces predictions $\hat{X} \in \mathbb{R}^{B \times G}$ for the perturbed state, where G is the number of genes. The target perturbed counts are $X_{\text{pert}} \in \mathbb{R}^{B \times G}$. Predictions are optionally normalized and can be computed in residual mode by adding control means or average perturbation effects. Detailed task description, loss function and data details are provided in subsubsection I.1.1.

In the chemical perturbation prediction task, the model predicts perturbed cell expression profiles given control cell counts and chemical perturbation conditions (including compound identity and dosage). The formulation is similar to genetic perturbations, but condition embeddings incorporate molecular representations and dosage information. Detailed task description, loss function and data details are provided in subsubsection I.1.2.

2.3.2 UNMASKING PERTURBATION PREDICTION

In the unmasking perturbation prediction task, a subset of genes is masked during training, and the model predicts expression values for the masked genes. The model receives control counts $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$ with some genes masked, and produces predictions $\hat{X} \in \mathbb{R}^{B \times G}$ for all genes, where B is the batch size and G is the number of genes.

We perform masking directly on the input count matrix $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$: a binary mask $M \in \{0,1\}^{B \times G}$ is defined where $M_{b,g} = 1$ indicates that the count value for gene g in batch b is masked (not observed), and $M_{b,g} = 0$ indicates it is observed. For each masked entry in X_{ctrl}, the corresponding value is replaced during count embedding with a special mask token embedding. The model is then trained to predict the original observed count value at every masked position.

2.3.3 ANNOTATION TASKS

In the annotation tasks, the model predicts discrete labels (such as cell type or tissue of origin) from gene expression profiles. The model receives expression counts $X \in \mathbb{R}^{B \times G}$ and produces predictions $\hat{y} \in \mathbb{R}^{B \times C}$, where C is the number of cell types. Detailed task description, loss function, and data details are provided in subsection I.3.

Table 1: Detailed performance metrics on genetic perturbation prediction.

		HEPG2 (Nadig et al., 2025)				K562 (Replogle et al., 2022)			
metric	preprocessing	State	Funomics T0	mean pert	Bound*	State	Funomics T0	mean pert	Bound*
Cosine Sim. ↑	train pert centered	0.33	0.41	0.00	0.34	0.21	0.28	0.00	0.34
MSE ↓	–	0.02	0.03	0.02	0.03	0.02	0.03	0.01	0.01
WMSE ↓	–	0.02	0.03	0.04	0.02	0.01	0.02	0.02	0.01
Rank.T (Cosine) ↓	train pert centered	0.22	0.06	1.00	0.11	0.25	0.09	1.00	0.11
Rank.T (L2) ↓	–	0.26	0.08	0.50	0.25	0.30	0.16	0.50	0.28
Rank (Cosine) ↓	train pert centered	0.22	0.05	1.00	0.11	0.20	0.05	1.00	0.12
Rank (L2) ↓	–	0.22	0.07	0.87	0.13	0.22	0.16	0.79	0.10
SMD (Cosine) ↓	train pert centered	155.19	10.41	11.48	6.60	211.43	14.06	7.32	5.10
SMD (L2) ↓	–	1562.68	57.09	449.56	55.20	1417.55	44.11	326.49	44.90

*Bound is the better of technical duplicate and interpolated duplicate baselines.

3 Experimental Results

3.1 Perturbation Prediction

Perturbation prediction is central to drug discovery applications, as it enables *in silico* screening of candidate compounds and genetic interventions before costly experimental validation (Baek et al., 2025; Csendes et al., 2025). The task involves predicting cellular gene expression responses following genetic or chemical perturbations, given control cell expression profiles and perturbation conditions. This capability is essential for understanding drug mechanisms of action, identifying off-target effects, and prioritizing therapeutic candidates (Wu et al., 2025).

Formally, given control expression profiles $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$ for a batch of B cells and G genes, along with condition embeddings $\mathcal{C} \in \mathbb{R}^{B \times N_{\text{cond}} \times d_{\text{cond}}}$, the model predicts perturbed expression profiles $\hat{X}_{\text{pert}} \in \mathbb{R}^{B \times G}$. The goal is to match the distribution of predicted responses, $\mathbb{P}(\hat{X}_{\text{pert}} \mid X_{\text{ctrl}}, \mathcal{C})$, to the true distribution of perturbed responses: $\mathbb{P}(X_{\text{pert}} \mid X_{\text{ctrl}}, \mathcal{C})$, capturing both the magnitude and direction of expression changes across all genes.

Metrics To evaluate model performance, we report a compact, complementary metric set that separates magnitude accuracy from directional agreement. We include Cosine similarity (direction), MSE and WMSE (absolute values), rank metrics for Cosine and L2 distances, and similarity matrix distance (SMD) for Cosine and L2. For cosine-based metrics, we center profiles by the training perturbation mean, consistent with prior work Littman et al. (2025). Detailed metric definitions are provided in Appendix K.

Baselines. We compare against three baselines (see subsection L.1): (1) **State model** (Adduri et al., 2025), (2) **post-perturbed mean** (see subsubsection L.1.1), which predicts the mean of perturbed expression in the training dataset for each perturbation following Littman et al. (2025). The post-perturbed mean baseline values were calculated on pseudobulks, without accounting for the number of samples per perturbation which was not present in the pseudobulk data; and (3) **interpolated duplicate baseline** (see subsubsection L.1.3), which addresses limitations of technical duplicates for weak perturbations (Miller et al., 2025).

We present performance on genetic perturbations (HEPG2 (Nadig et al., 2025) and K562 (Replogle et al., 2022)) in Table 1 and chemical perturbations (Tahoe (Zhang et al., 2025b)) in Table 2. We follow the same splitting strategy as described in the State paper (Adduri et al., 2025) and defer additional dataset results to Appendix D.

3.1.1 Genetic Perturbations

Overall, Funomics T0 performs better than State on both genetic perturbation datasets on most metrics, with clear gains on cosine-based similarity, rank metrics, and SMD. These gains are especially notable given that Funomics T0 was trained on a broader, more diverse set of datasets, yet still generalizes strongly to these benchmarks. It is slightly lower on absolute-value metrics (MSE/WMSE), though both models perform well and are close to the mean pert baseline; these magnitude errors are less biologically informative. Both models still improve over the mean pert baseline on WMSE, showing

WMSE better captures meaningful gains on well-calibrated metrics Mejia et al. (2025). On Nadig HEPG2, we observe results better than the interpolated duplicate bound, which may reflect technical duplicate splits that partition a single-cell population and introduce noise for small populations. We attribute this tradeoff to latent compression, which may soften exact value prediction while preserving the direction of changes that is more biologically relevant.

3.1.2 CHEMICAL PERTURBATIONS

On the Tahoe dataset, both models achieve strong performance across metrics, suggesting this benchmark is comparatively easy. State is better on cosine similarity, rank metrics, and MSE/WMSE, indicating more accurate absolute magnitudes and ordering. Funomics T0 is markedly better on SMD, which reflects closer alignment of the predicted perturbation–perturbation similarity structure with the ground truth, even when per-perturbation magnitudes are less calibrated. This suggests Funomics preserves global relational structure among perturbations, a property that can be useful for downstream grouping and interpretation. The limited separation between models on Tahoe may reflect the dataset's scale, where single-dataset training already provides enough coverage and reduces the benefit of multi-dataset pretraining.

Table 2: Chemical perturbation prediction results on Tahoe dataset (Zhang et al., 2025b).

metric	preprocessing	State	Funomics T0
Cosine Sim. ↑	train pert centered	0.78	0.67
MSE ↓	–	0.01	0.02
WMSE ↓	–	0.01	0.02
Rank.T (Cosine) ↓	train pert centered	0.01	0.04
Rank.T (L2) ↓	–	0.00	0.05
Rank (Cosine) ↓	train pert centered	0.00	0.01
Rank (L2) ↓	–	0.01	0.05
SMD (Cosine) ↓	train pert centered	209.68	65.62
SMD (L2) ↓	–	803.86	172.57

3.2 TISSUE ANNOTATION

In the tissue annotation task, the model predicts tissue labels from gene expression profiles. For this purpose, we employed the Gene Tissue Expression (GTEx; Lonsdale et al. (2013)) v10 dataset, composed of 19,788 samples from 946 donors, spanning 54 distinct tissue types, profiled using the bulk RNA-sequencing technology. As the model inputs we used the Transcripts Per Million (TPM) normalized expression data. As the number of samples per tissue is quite unbalanced, with a minimum of 11 observations per tissue, and maximum of around eight hundred, we excluded five tissues with less than 100 observations from the validation splits (Kidney-Medulla, Cervix-Ectocervix and -Endocervix, Fallopian Tube, Bladder).

For the tissue annotation task, we employ two complementary validation splits: `val_purity` (2,426 samples) and `val_tissue` (2,341 samples), with the remaining 14,849 samples used for training. The `val_purity` split is designed to select the hardest samples for model evaluation based on neighborhood purity in PCA space, while the `val_tissue` split provides a tissue-balanced benchmark comparable to prior work. Detailed descriptions of the split algorithms and dataset preprocessing are provided in subsection B.3.

For the tissue annotation task, we tested the performance of the Logistic Regression baseline model on the `val_purity` and `val_tissue` splits. The Logistic Regression model was trained on 5,000 Highly Variable Genes (HVGs) selected using the `scanpy` package, on the

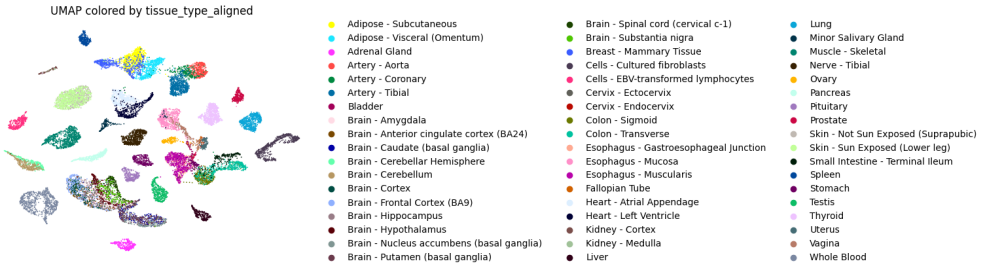

Figure 3: **UMAP visualization of GTEx gene expression profiles.** Each point represents a bulk sample, colored according to its annotated tissue label.

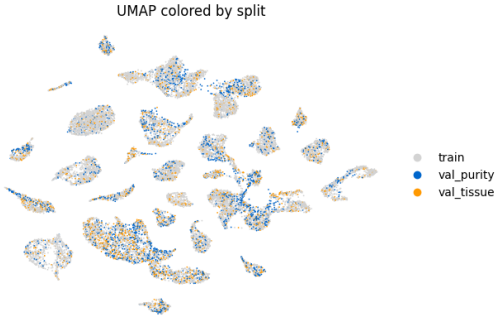

Figure 4: **UMAP visualization of validation splits** GTEx expression data, colored by the splits annotation: `val_purity`, `val_tissue`, `train`.

`log1p(normalize_total(data))` transformed data and standardized. As both splits were balanced in terms of tissue type, we report Accuracy metric.

Yap et al. (2021) report the F1-score of 96.1% on the split similar to our `val_tissue` split, a value that we report as a reference for our model performance together with the Logistic Regression baseline in Table 3. Additionally, we report the performance of our model trained from scratch (T0) and with pretraining (T0 (pretrained)) on multiple tasks and employing single-cell data. Pretraining yields substantial gains, improving accuracy by over 6 percentage points on `val_purity` and achieving near-perfect performance (99.5%) on `val_tissue`, demonstrating the benefit of large-scale multitask pretraining for downstream tissue classification.

Table 3: **Tissue Annotation Performance Comparison.** Accuracy (in %) of different models on the `val_purity` and `val_tissue` splits of GTEx v10.

Model	val_purity (%)	val_tissue (%)
Funomics T0	86.8	95.7
Funomics T0 (pretrained)	**93.2**	**99.5**
Logistic Regression	82.9	98.2

4 Discussion and Conclusion

In this work, we present Funomics T0, a foundation model trained to simultaneously learn a shared expression representation across RNA-seq datasets that are typically treated separately: bulk and single-cell, in vitro and patient-derived, and observational and perturbational studies.

The underlying idea is that the main obstacle to a practically useful omics model is the ability to extract stable biological patterns across heterogeneous study designs, parallel biological samples and technical context.

Across perturbation prediction and tissue annotation, we demonstrate its ability to effectively utilize the full spectrum of available transcriptomic data for solving downstream tasks. The latent representations learned by Funomics T0 transfer to task-specific predictors without additional encoders, suggesting that the model captures robust expression programs that are both detectable in single cells, compatible with bulk signals, and present in controlled perturbations and patient data. We see the clearest gains on the smaller genetic perturbation datasets, where multi-dataset training provides stronger generalization. Our results suggest that this unified architecture combined with massive multitask training across heterogeneous data sources can unlock improved modeling of cellular responses, pointing toward a promising direction for deeper understanding of cell processes.

Limitations and Future Work While Funomics T0 represents an advancement in foundation models for omics, several limitations remain. Currently, our model focuses on expression data from bulk and single-cell RNA-seq experiments. Training on heterogeneous RNA-seq gives path to domain imbalance and technical artifacts, and therefore performance can be skewed toward tissues, protocols, or cohorts that are more represented.

Additionally, while we demonstrate the model's effectiveness on perturbation prediction and tissue annotation tasks, there are numerous other relevant applications to evaluate, including disease classification, drug response prediction, and patient stratification.

Finally, while the latent space is predictive, we aim to interrogate its interpretability and relation to pathways and mechanistic clusters.

Rather than expanding to additional omics modalities, we focus next on expression-based generalization under true biological heterogeneity.

ACKNOWLEDGMENTS

This research has been kindly supported by Ingenix. We thank Paweł Czyż for his help with editing the paper.

REFERENCES

Abhinav K. Adduri, Dhruv Gautam, Beatrice Bevilacqua, Alishba Imran, Rohan Shah, Mohsen Naghipourfar, Noam Teyssier, Rajesh Ilango, Sanjay Nagaraj, Mingze Dong, Chiara Ricci-Tam, Christopher Carpenter, Vishvak Subramanyam, Aidan Winters, Sravya Tirukkovular, Jeremy Sullivan, Brian S. Plosky, Basak Eraslan, Nicholas D. Youngblut, Jure Leskovec, Luke A. Gilbert, Silvana Konermann, Patrick D. Hsu, Alexander Dobin, Dave P. Burke, Hani Goodarzi, and Yusuf H. Roohani. Predicting cellular responses to perturbation across diverse contexts with State. July 2025. doi: 10.1101/2025.06.26.661135. URL https://www.biorxiv.org/content/10.1101/2025.06.26.661135v2. ISSN: 2692-8205 Pages: 2025.06.26.661135 Section: New Results.

Seungbyn Baek, Kyungwoo Song, and Insuk Lee. Single-cell foundation models: bringing artificial intelligence into cell biology. *Experimental & Molecular Medicine*, 57(10):2169–2181, October 2025. ISSN 2092-6413. doi: 10.1038/s12276-025-01547-5. URL https://www.nature.com/articles/s12276-025-01547-5.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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, November 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01969-x. URL https://www.nature.com/articles/s41592-023-01969-x.

William Connell, Umair Khan, and Michael J. Keiser. A single-cell gene expression language model. October 2022. doi: 10.48550/arXiv.2210.14330. URL http://arxiv.org/abs/2210.14330. arXiv:2210.14330 [q-bio].

Gerold Csendes, Gema Sanz, Kristóf Z. Szalay, and Bence Szalai. Benchmarking foundation cell models for post-perturbation RNA-seq prediction. *BMC Genomics*, 26(1):393, April 2025. ISSN 1471-2164. doi: 10.1186/s12864-025-11600-2. URL https://doi.org/10.1186/s12864-025-11600-2.

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, August 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02201-0. URL https://www.nature.com/articles/s41592-024-02201-0.

Claudia Feng, Elin Madli Peets, Yan Zhou, Luca Crepaldi, Sunay Usluer, Alistair Dunham, Jana M. Braunger, Jing Su, Magdalena E. Strauss, Daniele Muraro, Kimberly Ai Xian Cheam, Marc Jan Bonder, Edgar Garriga Nogales, Sarah Cooper, Andrew Bassett, Steven Leonard, Yong Gu, Bo Fussing, David Burke, Leopold Parts, Oliver Stegle, and Britta Velten. A genome-scale single cell CRISPRi map of trans gene regulation across human pluripotent stem cell lines, November 2024. URL https://www.biorxiv.org/content/10.1101/2024.11.28.625833v1. Pages: 2024.11.28.625833 Section: New Results.

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, August 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02305-7. URL https://www.nature.com/articles/s41592-024-02305-7.

Graham Heimberg, Tony Kuo, Daryle J. DePianto, Omar Salem, Tobias Heigl, Nathaniel Diamant, Gabriele Scalia, Tommaso Biancalani, Shannon J. Turley, Jason R. Rock, Héctor Corrada Bravo, Josh Kaminker, Jason A. Vander Heiden, and Aviv Regev. A cell atlas foundation model for scalable search of similar human cells. *Nature*, 638(8052):1085–1094, February 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08411-y. URL https://www.nature.com/articles/s41586-024-08411-y.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A General Architecture for Structured Inputs & Outputs. October 2021a. URL https://openreview.net/forum?id=fILj7WpI-g.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4651–4664. PMLR, July 2021b. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Guillaume Jaume, Paul Doucet, Andrew H Song, Ming Y Lu, Cristina Almagro-Pérez, Sophia J Wagner, Anurag J Vaidya, Richard J Chen, Drew F K Williamson, Ahrong Kim, and Faisal Mahmood. HEST-1k: A Dataset for Spatial Transcriptomics and Histology Image Analysis.

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, January 2024. URL https://zenodo.org/records/10520190.

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, April 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-58699-1. URL https://www.nature.com/articles/s41467-025-58699-1.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753. PMLR, May 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Elon Litman, Tyler Myers, Vinayak Agarwal, Ekansh Mittal, Orion Li, Ashwin Gopinath, and Timothy Kassis. GeneJepa: A Predictive World Model of the Transcriptome. October 2025. doi: 10.1101/2025.10.14.682378. URL https://www.biorxiv.org/content/10.1101/2025.10.14.682378v1. ISSN: 2692-8205 Pages: 2025.10.14.682378 Section: New Results.

Russell Littman, Jacob Levine, Sepideh Maleki, Yongju Lee, Vladimir Ermakov, Lin Qiu, Alexander Wu, Kexin Huang, Romain Lopez, Gabriele Scalia, Tommaso Biancalani, David Richmond, Aviv Regev, and Jan-Christian Hütter. Gene-embedding-based prediction and functional evaluation of perturbation expression responses with PRESAGE. June 2025. doi: 10.1101/2025.06.03.657653. URL https://www.biorxiv.org/content/10.1101/2025.06.03.657653v1. Pages: 2025.06.03.657653 Section: New Results.

Qiyuan Liu, Qirui Zhang, Jinhong Du, Siming Zhao, and Jingshu Wang. Effects of Distance Metrics and Scaling on the Perturbation Discrimination Score, November 2025. URL http://arxiv.org/abs/2511.16954. arXiv:2511.16954 [stat].

John Lonsdale, Jeffrey Thomas, Mike Salvatore, Rebecca Phillips, Edmund Lo, Saboor Shad, Richard Hasz, Gary Walters, Fernando Garcia, Nancy Young, Barbara Foster, Mike Moser, Ellen Karasik, Bryan Gillard, Kimberley Ramsey, Susan Sullivan, Jason Bridge, Harold Magazine, John Syron, Johnelle Fleming, Laura Siminoff, Heather Traino, Maghboeba Mosavel, Laura Barker, Scott Jewell, Dan Rohrer, Dan Maxim, Dana Filkins, Philip Harbach, Eddie Cortadillo, Bree Berghuis, Lisa Turner, Eric Hudson, Kristin Feenstra, Leslie Sobin, James Robb, Phillip Branton, Greg Korzeniewski, Charles Shive, David Tabor, Liqun Qi, Kevin Groch, Sreenath Nampally, Steve Buia, Angela Zimmerman, Anna Smith, Robin Burges, Karna Robinson, Kim Valentino, Deborah Bradbury, Mark Cosentino, Norma Diaz-Mayoral, Mary Kennedy, Theresa Engel, Penelope Williams, Kenyon Erickson, Kristin Ardlie, Wendy Winckler, Gad Getz, David DeLuca, Daniel MacArthur, Manolis Kellis, Alexander Thomson, Taylor Young, Ellen Gelfand, Molly Donovan, Yan Meng, George Grant, Deborah Mash, Yvonne Marcus, Margaret Basile, Jun Liu, Jun Zhu, Zhidong Tu, Nancy J. Cox, Dan L. Nicolae, Eric R. Gamazon, Hae Kyung Im, Anuar Konkashbaev, Jonathan Pritchard, Matthew Stevens, Timothèe Flutre, Xiaoquan Wen, Emmanouil T. Dermitzakis, Tuuli Lappalainen, Roderic Guigo, Jean Monlong, Michael Sammeth, Daphne Koller, Alexis Battle, Sara Mostafavi, Mark McCarthy, Manual Rivas, Julian Maller, Ivan Rusyn, Andrew Nobel, Fred Wright, Andrey Shabalin, Mike Feolo, Nataliya Sharopova, Anne Sturcke, Justin Paschal, James M. Anderson, Elizabeth L. Wilder, Leslie K. Derr, Eric D. Green, Jeffery P. Struewing, Gary Temple, Simona Volpi, Joy T. Boyer, Elizabeth J. Thomson, Mark S. Guyer, Cathy Ng, Assya Abdallah, Deborah Colantuoni, Thomas R. Insel, Susan E. Koester, A. Roger Little, Patrick K. Bender, Thomas Lehner, Yin Yao, Carolyn C. Compton, Jimmie B. Vaught, Sherilyn Sawyer, Nicole C. Lockhart, Joanne Demchok, and Helen F. Moore. The Genotype-Tissue Expression (GTEx) project. *Nature Genetics*, 45(6):580–585, June 2013. ISSN 1546-1718. doi: 10.1038/ng.2653. URL https://www.nature.com/articles/ng.2653.

Gabriel M. Mejia, Henry E. Miller, Francis J. A. Leblanc, Bo Wang, Brendan Swain, and Lucas Paulo de Lima Camillo. Diversity by Design: Addressing Mode Collapse Improves scRNA-seq Perturbation Modeling on Well-Calibrated Metrics, June 2025. URL http://arxiv.org/abs/2506.22641. arXiv:2506.22641 [q-bio].

Henry E. Miller, Gabriel M. Mejia, Francis J. A. Leblanc, Bo Wang, Brendan Swain, and Lucas Paulo de Lima Camillo. Deep Learning-Based Genetic Perturbation Models Do Outperform Uninformative Baselines on Well-Calibrated Metrics. October 2025. doi: 10.1101/2025.10.20.683304. URL https://www.biorxiv.org/content/10.1101/2025.10.20.683304v1. ISSN: 2692-8205 Pages: 2025.10.20.683304 Section: New Results.

Pablo Moreno, Silvie Fexova, Nancy George, Jonathan R Manning, Zhichiao Miao, Suhaib Mohammed, Alfonso Muñoz-Pomer, Anja Fullgrabe, Yalan Bi, Natassja Bush, Haider Iqbal, Upendra Kumbham, Andrey Solovyev, Lingyun Zhao, Ananth Prakash, David García-Seisdedos, Deepti J Kundu, Shengbo Wang, Mathias Walzer, Laura Clarke, David Osumi-Sutherland, Marcela Karey

Tello-Ruiz, Sunita Kumari, Doreen Ware, Jana Eliasova, Mark J Arends, Martijn C Nawijn, Kerstin Meyer, Tony Burdett, John Marioni, Sarah Teichmann, Juan Antonio Vizcaíno, Alvis Brazma, and Irene Papatheodorou. Expression Atlas update: gene and protein expression in multiple species. *Nucleic Acids Research*, 50(D1):D129–D140, November 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab1030. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC8728300/.

Ajay Nadig, Joseph M. Replogle, Angela N. Pogson, Mukundh Murthy, Steven A. McCarroll, Jonathan S. Weissman, Elise B. Robinson, and Luke J. O'Connor. Transcriptome-wide analysis of differential expression in perturbation atlases. *Nature Genetics*, 57(5):1228–1237, May 2025. ISSN 1546-1718. doi: 10.1038/s41588-025-02169-3. URL https://www.nature.com/articles/s41588-025-02169-3.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, September 2023. URL http://arxiv.org/abs/1910.10683. arXiv:1910.10683 [cs].

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, Nika Iremadze, Florian Oberstrass, Doron Lipson, Jessica L. Bonnar, Marco Jost, Thomas M. Norman, and Jonathan S. Weissman. Mapping information-rich genotype-phenotype landscapes with genome-scale Perturb-seq. *Cell*, 185(14):2559–2575.e28, July 2022. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2022.05.013. URL https://www.cell.com/cell/abstract/S0092-8674(22)00597-9.

Yanay Rosen, Yusuf Roohani, Ayush Agrawal, Leon Samotorčan, Tabula Sapiens Consortium, Stephen R. Quake, and Jure Leskovec. Universal Cell Embeddings: A Foundation Model for Cell Biology. October 2024. doi: 10.1101/2023.11.28.568918. URL https://www.biorxiv.org/content/10.1101/2023.11.28.568918v2. Pages: 2023.11.28.568918 Section: New Results.

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, Fan Zhang, Frank Steemers, Jay Shendure, and Cole Trapnell. Massively multiplex chemical transcriptomics at single-cell resolution. *Science*, 367(6473):45–51, January 2020. doi: 10.1126/science.aax6234. URL https://www.science.org/doi/10.1126/science.aax6234.

Alejandro Tejada-Lapuerta, Anna C. Schaar, Robert Gutgesell, Giovanni Palla, Lennard Halle, Mariia Minaeva, Larsen Vornholz, Leander Dony, Francesca Drummer, Till Richter, Mojtaba Bahrami, and Fabian J. Theis. Nicheformer: a foundation model for single-cell and spatial omics. *Nature Methods*, 22(12):2525–2538, December 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02814-z. URL https://www.nature.com/articles/s41592-025-02814-z.

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, and Patrick T. Ellinor. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, June 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06139-9. URL https://www.nature.com/articles/s41586-023-06139-9.

Alexander Theus, Florian Barkmann, David Wissel, and Valentina Boeva. CancerFoundation: A single-cell RNA sequencing foundation model to decipher drug resistance in cancer, November 2024. URL https://www.biorxiv.org/content/10.1101/2024.11.01.621087v1. Pages: 2024.11.01.621087 Section: New Results.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Gefei Wang, Tianyu Liu, Jia Zhao, Youshu Cheng, and Hongyu Zhao. Modeling and predicting single-cell multi-gene perturbation responses with scLAMBDA. December 2024. doi: 10.1101/2024.12.04.626878. URL https://www.biorxiv.org/content/10.1101/2024.12.04.626878v1. Pages: 2024.12.04.626878 Section: New Results.

David Wissel, Nikita Janakarajan, Aayush Grover, Enrico Toniato, Maria Rodríguez Martínez, and Valentina Boeva. SurvBoard: standardized benchmarking for multi-omics cancer survival models. *Briefings in Bioinformatics*, 26(5):bbaf521, September 2025. ISSN 1477-4054. doi: 10.1093/bib/bbaf521. URL https://doi.org/10.1093/bib/bbaf521.

Yan Wu, Esther Wershof, Sebastian M. Schmon, Marcel Nassar, Błażej Osiński, Ridvan Eksi, Zichao Yan, Rory Stark, Kun Zhang, and Thore Graepel. PerturBench: Benchmarking Machine Learning Models for Cellular Perturbation Analysis. October 2025. doi: 10.48550/arXiv.2408.10609. URL http://arxiv.org/abs/2408.10609. arXiv:2408.10609 [cs].

Fan Yang, Wenchuan Wang, Fang Wang, Yuan Fang, Duyu Tang, Junzhou Huang, Hui Lu, and Jianhua Yao. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. *Nature Machine Intelligence*, 4(10):852–866, October 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00534-z. URL https://www.nature.com/articles/s42256-022-00534-z.

Xiaodong Yang, Guole Liu, Guihai Feng, Dechao Bu, Pengfei Wang, Jie Jiang, Shubai Chen, Qinmeng Yang, Hefan Miao, Yiyang Zhang, Zhenpeng Man, Zhongming Liang, Zichen Wang, Yaning Li, Zheng Li, Yana Liu, Yao Tian, Wenhao Liu, Cong Li, Ao Li, Jingxi Dong, Zhilong Hu, Chen Fang, Lina Cui, Zixu Deng, Haiping Jiang, Wentao Cui, Jiahao Zhang, Zhaohui Yang, Handong Li, Xingjian He, Liqun Zhong, Jiaheng Zhou, Zijian Wang, Qingqing Long, Ping Xu, Xin Li, Hongmei Wang, Zhen Meng, Xuezhi Wang, Yangang Wang, Yong Wang, Shihua Zhang, Jingtao Guo, Yi Zhao, Yuanchun Zhou, Fei Li, Jing Liu, Yiqiang Chen, Ge Yang, and Xin Li. GeneCompass: deciphering universal gene regulatory mechanisms with a knowledge-informed cross-species foundation model. *Cell Research*, 34(12):830–845, December 2024. ISSN 1748-7838. doi: 10.1038/s41422-024-01034-y. URL https://www.nature.com/articles/s41422-024-01034-y.

Melvyn Yap, Rebecca L. Johnston, Helena Foley, Samual MacDonald, Olga Kondrashova, Khoa A. Tran, Katia Nones, Lambros T. Koufariotis, Cameron Bean, John V. Pearson, Maciej Trzaskowski, and Nicola Waddell. Verifying explainability of a deep learning tissue classifier trained on RNA-seq data. *Scientific Reports*, 11(1):2641, January 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-81773-9. URL https://www.nature.com/articles/s41598-021-81773-9.

Yuansong Zeng, Jiancong Xie, Ningyuan Shangguan, Zhuoyi Wei, Wenbing Li, Yun Su, Shuangyu Yang, Chengyang Zhang, Jinbo Zhang, Nan Fang, Hongyu Zhang, Yutong Lu, Huiying Zhao, Jue Fan, Weijiang Yu, and Yuedong Yang. CellFM: a large-scale foundation model pre-trained on transcriptomics of 100 million human cells. *Nature Communications*, 16(1):4679, May 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-59926-5. URL https://www.nature.com/articles/s41467-025-59926-5.

Biao Zhang, Fedor Moiseev, Joshua Ainslie, Paul Suganthan, Min Ma, Surya Bhupatiraju, Fede Lebron, Orhan Firat, Armand Joulin, and Zhe Dong. Encoder-Decoder Gemma: Improving the Quality-Efficiency Trade-Off via Adaptation, April 2025a. URL http://arxiv.org/abs/2504.06225. arXiv:2504.06225 [cs].

Jesse Zhang, Airol A. Ubas, Richard de Borja, Valentine Svensson, Nicole Thomas, Neha Thakar, Ian Lai, Aidan Winters, Umair Khan, Matthew G. Jones, 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, February 2025b. URL https://www.biorxiv.org/content/10.1101/2025.02.20.639398v1. Pages: 2025.02.20.639398 Section: New Results.

Yiming Zhang, Ting Zhang, Gaoxia Yang, Zhenzhong Pan, Min Tang, Yue Wen, Ping He, Yuan Wang, and Ran Zhou. PerturbAtlas: a comprehensive atlas of public genetic perturbation bulk RNA-seq datasets. *Nucleic Acids Research*, 53(D1):D1112–D1119, January 2025c. ISSN 1362-4962. doi: 10.1093/nar/gkae851. URL https://doi.org/10.1093/nar/gkae851.

A Full Forward Pass Algorithm

The complete forward pass algorithm (algorithm 1) orchestrates three main components: task-specific input processing (algorithm 2), backbone processing (algorithm 3), and task head application (algorithm 4).

Algorithm 1: Funomics T0 Forward Pass

Function `ForwardPass(`X, $\mathcal{C}$, $\mathcal{T}$`)`**:**
  Input: Gene expression $X \in \mathbb{R}^{B \times G}$, conditions $\mathcal{C}$, task $\mathcal{T}$
  Output: Task outputs and losses
  $X_{\text{proc}}, \mathcal{C}_{\text{proc}} \leftarrow$ `ProcessTaskInputs(`X, $\mathcal{C}$, $\mathcal{T}$`)`
  `Process inputs according to task requirements (see algorithm 2)`
  $L \leftarrow$ `ProcessBackbone(`X_{proc}, $\mathcal{C}_{\text{proc}}$`)`
  `Process through backbone to obtain latent states (see algorithm 3)`
  $\mathcal{O}, \mathcal{L} \leftarrow$ `ApplyTaskHead(`L, $\mathcal{T}$`)`
  `Apply task-specific head to generate outputs (see algorithm 4)`
  return $\mathcal{O}, \mathcal{L}$

Algorithm 2: Task-Specific Input Processing

Function `ProcessTaskInputs(`X, $\mathcal{C}$, $\mathcal{T}$`)`**:**
  Input: Gene expression X, conditions $\mathcal{C}$, task $\mathcal{T}$
  Output: Processed inputs X_{proc}, $\mathcal{C}_{\text{proc}}$
  if $\mathcal{T}$ *is masked counts task* **then**
    Apply masking to X according to task-specific masking strategy
  else
    if $\mathcal{T}$ *is perturbation prediction task* **then**
      Prepare control and perturbed counts
      `Apply augmentation masking to control counts if training`
    else
      Use inputs as provided
  Process conditions $\mathcal{C}$ according to task requirements
  `Extract relevant condition components (dataset, task, perturbation type)`
  return $X_{\text{proc}}, \mathcal{C}_{\text{proc}}$

Algorithm 3: Funomics T0 Backbone Processing

Function `ProcessBackbone`$(X, \mathcal{C})$**:**
  Input: Gene expression $X \in \mathbb{R}^{B \times G}$, conditions $\mathcal{C}$
  Output: Latent states $L \in \mathbb{R}^{B \times N_{\text{latent}} \times d}$
  $E_{\text{gene}}, E_{\text{counts}}, E_{\text{cond}} \leftarrow$ `EmbedGeneExpressionAndConditions`$(X, \mathcal{C})$
  `Embed gene expression and conditions (see Appendix H)`
  $E_{\text{gene}} \in \mathbb{R}^{G \times d}$, $E_{\text{counts}} \in \mathbb{R}^{G \times d}$, $E_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$
  $E_{\text{input}} \leftarrow E_{\text{gene}} + E_{\text{counts}}$
  `Combine gene and counts embeddings:` $E_{\text{input}} \in \mathbb{R}^{G \times d}$
  $H_{\text{cond}} \leftarrow$ `EncodeConditions`(E_{cond})
  `Encode conditions through bidirectional self-attention`
  `Output:` $H_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$
  $L \leftarrow$ `CompressToLatents`$(E_{\text{input}}, H_{\text{cond}})$
  `Compress gene data to latent representation via decoder cross-attention`
  `Decoder uses self-attention on latent tokens, then cross-attention to` E_{input} `and` H_{cond}
  `Output:` $L \in \mathbb{R}^{N_{\text{latent}} \times d}$
  return L

Function `EmbedGeneExpressionAndConditions`$(X, \mathcal{C})$**:**
  $E_{\text{gene}} \leftarrow$ Embed gene tokens
  `Lookup gene token embeddings, optionally with external embeddings`
  $E_{\text{counts}} \leftarrow$ Embed expression counts
  `Embed counts via MLP encoder, apply masking for task-specific inputs`
  $E_{\text{cond}} \leftarrow$ Embed conditions
  `Embed dataset, task, and perturbation information (gene or chemical)`
  return $E_{\text{gene}}, E_{\text{counts}}, E_{\text{cond}}$

Function `EncodeConditions`(E_{cond})**:**
  Apply bidirectional self-attention over condition tokens
  `Multiple layers of self-attention + FFN with RMSNorm`
  return H_{cond}

Function `CompressToLatents`$(E_{\text{input}}, H_{\text{cond}})$**:**
  Initialize N_{latent} learnable latent tokens
  `Latent tokens:` $L_{\text{init}} \in \mathbb{R}^{N_{\text{latent}} \times d}$
  Apply self-attention among latent tokens
  Apply cross-attention: queries from latents, keys/values from E_{input}
  Apply cross-attention: queries from latents, keys/values from H_{cond}
  `Decoder applies multiple layers with FFN and RMSNorm`
  return L

Algorithm 4: Task Head Application

Function `ApplyTaskHead(`L, $\mathcal{T}$`)`**:**
  Input: Latent states $L \in \mathbb{R}^{B \times N_{\text{latent}} \times d}$, task $\mathcal{T}$
  Output: Task outputs $\mathcal{O}$, losses $\mathcal{L}$
  $H_{\text{genes}} \leftarrow$ `ProjectToGenes(`L`)`
  `Project latent states to gene space via cross-attention`
  `Output:` $H_{\text{genes}} \in \mathbb{R}^{G \times d}$
  if $\mathcal{T}$ *is perturbation prediction* **then**
    $\mathcal{O} \leftarrow$ `ApplyPerturbationHead(`H_{genes}`)`
  else
    if $\mathcal{T}$ *is classification* **then**
      $\mathcal{O} \leftarrow$ `ApplyClassificationHead(`H_{genes}`)`
    else
  $\mathcal{L} \leftarrow$ `ComputeTaskLoss(`$\mathcal{O}$, *targets*, $\mathcal{T}$`)`
  `Compute task-specific loss (see Appendix J)`
  return $\mathcal{O}, \mathcal{L}$
Function `ProjectToGenes(`L`)`**:**
  `Apply projection cross-attention`

  `Queries from gene embeddings, keys/values from latent states` L
  return H_{genes}
Function `ApplyPerturbationHead(`H_{genes}`)`**:**
  `Apply cross-attention with gene token embeddings`

  `Project to gene-level expression values`

  `Output:` $\hat{X} \in \mathbb{R}^{B \times G}$
  return $\hat{X}$
Function `ApplyClassificationHead(`H_{genes}`)`**:**
  `Aggregate` H_{genes} `into per-sample vector`

  `Mean pooling or learned summarization:` $h \in \mathbb{R}^d$
  Map h to class logits
  Apply softmax to obtain class probabilities
  `Output:` $\hat{y} \in [0,1]^{B \times C}$
  return $\hat{y}$
Function `ComputeTaskLoss(`$\mathcal{O}$, *targets*, $\mathcal{T}$`)`**:**
  if $\mathcal{T}$ *is perturbation prediction* **then**
    `Compute distribution loss and bulk prediction loss`
  else
    if $\mathcal{T}$ *is masked counts* **then**
      `Compute MSE loss on masked and unmasked positions`
    else
      `Compute task-specific loss`
  return $\mathcal{L}$

The backbone algorithm processes batches of B cells, each with G genes, producing compressed latent representations of dimension $N_{\text{latent}} \times d$, where $N_{\text{latent}} \ll G$ enables efficient processing of high-dimensional gene expression data. The decoder's dual cross-attention mechanism allows it to selectively attend to relevant genes while conditioning on the experimental context. Task-specific processing and head application extend the backbone outputs to produce task-appropriate predictions and losses.

B DATASETS AND VALIDATION SPLITS

This section provides detailed descriptions of the datasets used for training and evaluating the Funomics T0 model, including the specific validation split strategies employed for each task.

B.1 PRETRAINING DATA

Datasets are split by cell line and listed in Table 4. These datasets are used for both the masked count task and the perturbation prediction task.

B.2 VALIDATION STRATEGY

We evaluate genetic perturbation prediction on two diverse single-cell RNA-seq datasets:

- **HEPG2 (Nadig 2025):** This dataset (Nadig et al., 2025) consists of CRISPR-mediated perturbations in the HEPG2 cell line.
- **K562 (Replogle 2022):** A large-scale genome-wide CRISPRi screen in K562 cells (Replogle et al., 2022).

Validation Strategy: For these datasets, we employ two complementary splitting strategies to assess generalization:

1. **Leave-Perturbation-Out (LPO):** A subset of perturbations is held out during training. The model must predict the response to novel genetic interventions not seen in the training set.
2. **Leave-Cell-line-Out (LCO):** Where applicable, models are evaluated on cell lines entirely withheld from the training phase to test cross-context generalization.

Chemical perturbation performance is evaluated using the Tahoe datasets for HOP62 cell line. These datasets capture cellular responses to a variety of small-molecule compounds at multiple dosages.

Validation Strategy: Similar to genetic perturbations, we use a held-out compound strategy. A fixed percentage of chemical compounds (and all their associated dosages) are moved to the validation set, ensuring the model generalizes to unseen molecular structures.

B.3 TISSUE ANNOTATION (GTEX V10)

For tissue annotation, we use the Gene Tissue Expression (GTEx) v10 dataset (Lonsdale et al., 2013), comprising 19,788 bulk RNA-seq samples from 946 donors across 54 tissue types.

Validation Splits: We define two specific validation splits to test different aspects of model robustness:

1. **Validation Purity Split (`val_purity`):** Designed to identify the most challenging samples where tissue-specific signals may be subtle or mixed. The split is constructed as follows:
   - **Preprocessing:** TPM-normalized counts are total-count normalized and log-transformed (`log1p`). The top 5,000 highly variable genes are selected.
   - **Dimensionality Reduction:** Data is standardized and projected onto 50 principal components.
   - **Purity Calculation:** For each sample, we compute the 10 nearest neighbors in PCA space. Purity is defined as the fraction of neighbors sharing the same tissue label.
   - **Assignment:** For each tissue (with $\geq$ 100 samples), we select 50 samples with the lowest purity (or up to 30% of the tissue's total samples) for the validation set.
2. **Validation Tissue Split (`val_tissue`):** A tissue-balanced split that mimics standard evaluation protocols in prior literature (Yap et al., 2021). We select 50 samples per tissue type (up to 30% of available observations), excluding tissues with fewer than 100 samples.

Samples not included in either validation split are used for training (14,849 samples). Tissues with extremely low representation (Kidney-Medulla, Cervix-Ectocervix, Cervix-Endocervix, Fallopian Tube, Bladder) are excluded from validation to ensure stable metric calculation.

C TASK-SPECIFIC DATA DETAILS

This section provides detailed information about data used for each specific task, including datasets, split strategies, and evaluation metrics.

Datasets are split by cell line. The following table lists all datasets used for perturbation prediction.

C.1 ANNOTATION TASK DATA

Tissue Annotation In the tissue annotation task, the model predicts tissue labels from gene expression profiles. For this purpose, we employed the Gene Tissue Expression (GTEx) v10 dataset, composed of 19788 samples from 946 donors, spanning 54 distinct tissue types, profiled using the bulk RNA-sequencing technology. We used the Transcripts Per Million (TPM) normalized expression data instead of raw counts data to account for different gene lengths n As the number of samples per tissue is quite unbalanced, with a minimum of 11 observations per tissue, and maximum of 800, we excluded five tissues with less than 100 observations from the validation splits. The following tissues were excluded: Kidney - Medulla - 11, Cervix - Ectocervix - 24, Fallopian Tube - 29, Cervix - Endocervix - 23, Bladder - 77.

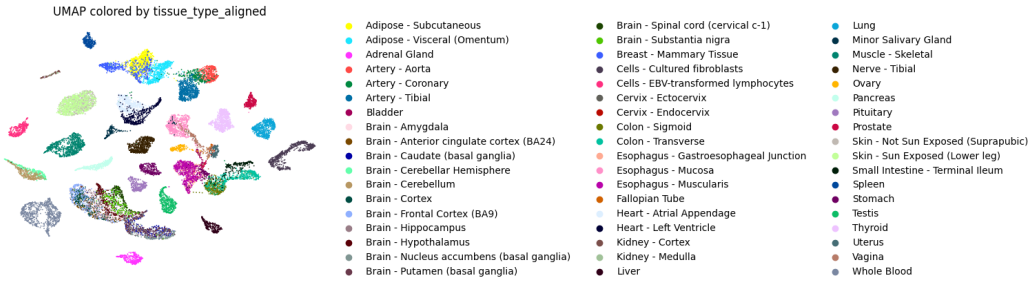

Figure 5: **UMAP visualization of GTEx v10 gene expression profiles colored by tissue type.** Each point represents a sample from the GTEx v10 bulk RNA-seq dataset, colored according to its annotated tissue label. This figure highlights the degree of separability and clustering structure among tissues in high-dimensional expression space.

For the tissue annotation task, we employ two complementary validation splits: `val_purity` (2426 samples) and `val_tissue` (2341 samples), with the remaining 14849 samples used for training.

Details on the GTEx tissue annotation dataset, including split definitions and procedures, are provided in subsection C.1.

D ADDITIONAL RESULTS

Table 5 reports additional L1 metrics (Rank, Rank.T, and SMD) for genetic perturbation prediction.

Table 6 reports additional L1 metrics (Rank, Rank.T, and SMD) for chemical perturbation prediction on the Tahoe dataset.

E BACKBONE ARCHITECTURE DETAILS

The Funomics T0 backbone implementation is based on T5Gemma (Zhang et al., 2025a) components for the encoder and decoder layers. T5Gemma (Zhang et al., 2025a) provides an efficient encoder-decoder architecture that is well-suited for processing structured inputs with cross-attention mechanisms. The architecture follows a set transformer design (Lee et al., 2019), treating inputs as

Table 4: Datasets used for perturbation prediction.

Dataset Name	Modality	Organism	Perturbation Type	Observations	Perturbations	Batches	Reference
Genetic perturbation							
VCC_H1_hESC_training_adata_unified_gene_set	scRNA-seq	*H. sapiens*	Genetic Knock Down	221,273	150	3	(Jaume et al.)
scPerturb_nadig_2025_JURKAT	scRNA-seq	*H. sapiens*	Genetic Knock Down	262,803	2,389	55	(Nadig et al., 2025)
scPerturb_nadig_2025_HEPG2	scRNA-seq	*H. sapiens*	Genetic Knock Down	145,427	2,389	56	(Nadig et al., 2025)
jiang_2025_A549	scRNA-seq	*H. sapiens*	Genetic Knock Down	207,261	218	2	(Jiang et al., 2024)
jiang_2025_BXPC3	scRNA-seq	*H. sapiens*	Genetic Knock Down	314,758	218	2	(Jiang et al., 2024)
jiang_2025_HAP1	scRNA-seq	*H. sapiens*	Genetic Knock Down	277,261	218	2	(Jiang et al., 2024)
feng_2024_TargetedScreen_HPSI0114i-eipl_1	scRNA-seq	*H. sapiens*	Genetic Knock Down	38,699	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0114i-eipl_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	37,180	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0114i-kolf_2	scRNA-seq	*H. sapiens*	Genetic Knock Down	40,563	444	62	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0114i-kolf_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	36,787	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0114i-zapk_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	46,376	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0414i-oikd_2	scRNA-seq	*H. sapiens*	Genetic Knock Down	14,127	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0414i-oikd_5	scRNA-seq	*H. sapiens*	Genetic Knock Down	45,507	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0514i-fiaj_1	scRNA-seq	*H. sapiens*	Genetic Knock Down	41,798	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0514i-fiaj_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	8,254	443	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0614i-paab_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	28,918	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0614i-paab_4	scRNA-seq	*H. sapiens*	Genetic Knock Down	42,507	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0714i-iudw_1	scRNA-seq	*H. sapiens*	Genetic Knock Down	44,469	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0714i-iudw_4	scRNA-seq	*H. sapiens*	Genetic Knock Down	34,702	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0714i-pipw_4	scRNA-seq	*H. sapiens*	Genetic Knock Down	37,178	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0714i-pipw_5	scRNA-seq	*H. sapiens*	Genetic Knock Down	9,325	444	70	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0914i-jejf_2	scRNA-seq	*H. sapiens*	Genetic Knock Down	47,558	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI0914i-jejf_3	scRNA-seq	*H. sapiens*	Genetic Knock Down	34,851	444	54	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI1213i-tolg_4	scRNA-seq	*H. sapiens*	Genetic Knock Down	16,732	444	72	(Feng et al., 2024)
feng_2024_TargetedScreen_HPSI1213i-tolg_6	scRNA-seq	*H. sapiens*	Genetic Knock Down	29,491	444	70	(Feng et al., 2024)
sciplex_gxe2_genetic_A172	scRNA-seq	*H. sapiens*	Genetic Knock Down	47,452	522	32	(Srivatsan et al., 2020)
sciplex_gxe2_genetic_T98G	scRNA-seq	*H. sapiens*	Genetic Knock Down	44,792	522	32	(Srivatsan et al., 2020)
sciplex_gxe2_genetic_U87MG	scRNA-seq	*H. sapiens*	Genetic Knock Down	47,026	522	32	(Srivatsan et al., 2020)
scPerturb_Replogle2022_RPE1essential	scRNA-seq	*H. sapiens*	Genetic Knock Down	247,824	2,389	56	(Replogle et al., 2022)
scPerturb_Replogle2022_K562essential	scRNA-seq	*H. sapiens*	Genetic Knock Down	310,342	2,054	48	(Replogle et al., 2022)
scPerturb_Replogle2022_K562gwps	scRNA-seq	*H. sapiens*	Genetic Knock Down	1,988,657	9,854	267	(Replogle et al., 2022)
jiang_2025_HT29	scRNA-seq	*H. sapiens*	Genetic Knock Down	360,963	218	2	(Jiang et al., 2024)
jiang_2025_K562	scRNA-seq	*H. sapiens*	Genetic Knock Down	207,688	218	2	(Jiang et al., 2024)
jiang_2025_MCF7	scRNA-seq	*H. sapiens*	Genetic Knock Down	260,545	218	2	(Jiang et al., 2024)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	349,927	2,000	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	362,424	1,999	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	344,738	1,998	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	354,504	1,998	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	349,785	2,000	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	50,924	293	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	352,553	2,000	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	354,801	2,000	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	357,281	2,000	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	364,990	1,997	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HCT116	scRNA-seq	*H. sapiens*	Genetic Knock Down	165,777	0	109	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	233,951	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	239,735	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	240,981	997	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	218,838	0	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	234,823	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	232,650	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	249,132	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	232,787	999	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	237,874	998	223	(Moreno et al., 2021)
scPerturb_Replogle2022_K562gwps	scRNA-seq	*H. sapiens*	Genetic Knock Down	876,979	3,286	267	(Replogle et al., 2022)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	219,246	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	231,660	999	223	(Moreno et al., 2021)
scPerturb_Replogle2022_K562gwps	scRNA-seq	*H. sapiens*	Genetic Knock Down	668,691	3,283	267	(Replogle et al., 2022)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	237,919	999	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	234,524	1,000	223	(Moreno et al., 2021)
scPerturb_Replogle2022_K562gwps	scRNA-seq	*H. sapiens*	Genetic Knock Down	442,987	3,288	267	(Replogle et al., 2022)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	231,491	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	72,981	311	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	228,542	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	236,216	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	226,008	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	249,862	1,000	223	(Moreno et al., 2021)
xatlas_filtered_dual_guide_cells_HEK293T	scRNA-seq	*H. sapiens*	Genetic Knock Down	243,051	1,000	223	(Moreno et al., 2021)
PerturbAtlas_hESC_bulk_genetic_KD.h5ad	bulk RNAseq	*H. sapiens*	Genetic Knock Down	18	1	2	(Zhang et al., 2025c)
Chemical perturbation							
sciplex_gxe2_chemical_A172	scRNA-seq	*H. sapiens*	Chemical	236,302	8	32	(Srivatsan et al., 2020)
sciplex_gxe2_chemical_T98G	scRNA-seq	*H. sapiens*	Chemical	212,789	8	32	(Srivatsan et al., 2020)
sciplex_gxe2_chemical_U87MG	scRNA-seq	*H. sapiens*	Chemical	238,788	8	32	(Srivatsan et al., 2020)
sciplex3_MCF7	scRNA-seq	*H. sapiens*	Chemical	344,837	752	16	(Srivatsan et al., 2020)
sciplex3_K562	scRNA-seq	*H. sapiens*	Chemical	173,632	752	16	(Srivatsan et al., 2020)
sciplex3_A549	scRNA-seq	*H. sapiens*	Chemical	244,268	752	20	(Srivatsan et al., 2020)
tahoe_100m_SW1271	scRNA-seq	*H. sapiens*	Chemical	209,283	1,131	42	(Zhang et al., 2025b)
tahoe_100m_A549	scRNA-seq	*H. sapiens*	Chemical	2,567,838	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SW1088	scRNA-seq	*H. sapiens*	Chemical	149,059	1,133	42	(Zhang et al., 2025b)
tahoe_100m_AN3CA	scRNA-seq	*H. sapiens*	Chemical	566,078	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SW1417	scRNA-seq	*H. sapiens*	Chemical	1,921,742	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SKMEL2	scRNA-seq	*H. sapiens*	Chemical	1,548,711	1,133	42	(Zhang et al., 2025b)
tahoe_100m_SW48	scRNA-seq	*H. sapiens*	Chemical	954,199	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SNU1	scRNA-seq	*H. sapiens*	Chemical	1,221,708	1,134	42	(Zhang et al., 2025b)
tahoe_100m_A172	scRNA-seq	*H. sapiens*	Chemical	2,314,134	1,134	42	(Zhang et al., 2025b)
tahoe_100m_ASPC1	scRNA-seq	*H. sapiens*	Chemical	2,259,176	1,134	42	(Zhang et al., 2025b)
tahoe_100m_BT474	scRNA-seq	*H. sapiens*	Chemical	1,271,918	1,133	42	(Zhang et al., 2025b)
tahoe_100m_COLO205	scRNA-seq	*H. sapiens*	Chemical	1,456,805	1,134	42	(Zhang et al., 2025b)
tahoe_100m_HCT15	scRNA-seq	*H. sapiens*	Chemical	1,500,121	1,134	42	(Zhang et al., 2025b)
tahoe_100m_HEC1A	scRNA-seq	*H. sapiens*	Chemical	2,514,625	1,134	42	(Zhang et al., 2025b)
tahoe_100m_HT29	scRNA-seq	*H. sapiens*	Chemical	2,171,036	1,134	42	(Zhang et al., 2025b)
tahoe_100m_HS578T	scRNA-seq	*H. sapiens*	Chemical	1,725,286	1,134	42	(Zhang et al., 2025b)
tahoe_100m_HS766T	scRNA-seq	*H. sapiens*	Chemical	2,535,736	1,133	42	(Zhang et al., 2025b)
tahoe_100m_J82	scRNA-seq	*H. sapiens*	Chemical	2,342,982	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SNU423	scRNA-seq	*H. sapiens*	Chemical	1,501,700	1,134	42	(Zhang et al., 2025b)
tahoe_100m_KATOIII	scRNA-seq	*H. sapiens*	Chemical	2,229,643	1,134	42	(Zhang et al., 2025b)
tahoe_100m_LS180	scRNA-seq	*H. sapiens*	Chemical	1,828,299	1,134	42	(Zhang et al., 2025b)
tahoe_100m_LOVO	scRNA-seq	*H. sapiens*	Chemical	3,013,246	1,134	42	(Zhang et al., 2025b)
tahoe_100m_MIAPACA2	scRNA-seq	*H. sapiens*	Chemical	2,432,647	1,133	42	(Zhang et al., 2025b)
tahoe_100m_NCIH460	scRNA-seq	*H. sapiens*	Chemical	5,736,238	1,134	42	(Zhang et al., 2025b)
tahoe_100m_PANC1	scRNA-seq	*H. sapiens*	Chemical	4,089,586	1,133	42	(Zhang et al., 2025b)
tahoe_100m_RKO	scRNA-seq	*H. sapiens*	Chemical	2,182,314	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SW480	scRNA-seq	*H. sapiens*	Chemical	6,040,371	1,134	42	(Zhang et al., 2025b)
tahoe_100m_A427	scRNA-seq	*H. sapiens*	Chemical	1,430,114	1,134	42	(Zhang et al., 2025b)
tahoe_100m_A498	scRNA-seq	*H. sapiens*	Chemical	2,737,714	1,134	42	(Zhang et al., 2025b)
tahoe_100m_C33A	scRNA-seq	*H. sapiens*	Chemical	1,483,268	1,134	42	(Zhang et al., 2025b)
tahoe_100m_C32	scRNA-seq	*H. sapiens*	Chemical	1,991,252	1,134	42	(Zhang et al., 2025b)
tahoe_100m_C3A	scRNA-seq	*H. sapiens*	Chemical	1,244,192	1,134	42	(Zhang et al., 2025b)
tahoe_100m_CFPAC1	scRNA-seq	*H. sapiens*	Chemical	2,404,632	1,134	42	(Zhang et al., 2025b)
tahoe_100m_CHP212	scRNA-seq	*H. sapiens*	Chemical	537,791	1,134	42	(Zhang et al., 2025b)
tahoe_100m_H4	scRNA-seq	*H. sapiens*	Chemical	1,042,066	1,133	42	(Zhang et al., 2025b)
tahoe_100m_HOP62	scRNA-seq	*H. sapiens*	Chemical	3,307,302	1,134	42	(Zhang et al., 2025b)
tahoe_100m_LOXIMVI	scRNA-seq	*H. sapiens*	Chemical	2,208,183	1,134	42	(Zhang et al., 2025b)
tahoe_100m_NCIH1573	scRNA-seq	*H. sapiens*	Chemical	2,016,991	1,134	42	(Zhang et al., 2025b)
tahoe_100m_NCIH1792	scRNA-seq	*H. sapiens*	Chemical	1,838,530	1,133	42	(Zhang et al., 2025b)
tahoe_100m_NCIH2030	scRNA-seq	*H. sapiens*	Chemical	1,937,218	1,134	42	(Zhang et al., 2025b)
tahoe_100m_NCIH23	scRNA-seq	*H. sapiens*	Chemical	1,774,330	1,133	42	(Zhang et al., 2025b)
tahoe_100m_NCIH2347	scRNA-seq	*H. sapiens*	Chemical	2,227,663	1,133	42	(Zhang et al., 2025b)
tahoe_100m_PANC0327	scRNA-seq	*H. sapiens*	Chemical	1,700,099	1,134	42	(Zhang et al., 2025b)
tahoe_100m_RPMI7951	scRNA-seq	*H. sapiens*	Chemical	1,607,552	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SHP77	scRNA-seq	*H. sapiens*	Chemical	2,282,536	1,134	42	(Zhang et al., 2025b)
tahoe_100m_SW900	scRNA-seq	*H. sapiens*	Chemical	1,708,896	1,134	42	(Zhang et al., 2025b)
tahoe_100m_hTERTHPNE	scRNA-seq	*H. sapiens*	Chemical	1,682,310	1,134	42	(Zhang et al., 2025b)

unordered sets without positional embeddings, making it permutation-invariant and well-suited for gene expression data.

Table 5: Additional L1 metrics on genetic perturbation prediction (not shown in main results).

	HEPG2 (Nadig et al., 2025)				K562 (Replogle et al., 2022)			
metric	State	Funomics T0	mean pert	Bound*	State	Funomics T0	mean pert	Bound*
Rank (L1) ↓	0.22	0.09	1.00	0.02	0.23	0.21	1.00	0.00
Rank.T (L1) ↓	0.27	0.09	0.50	0.26	0.31	0.22	0.50	0.28
SMD (L1) ↓	50876.84	1834.25	14624.03	1675.30	46338.88	1419.17	10507.64	1362.63

*Bound is the better of technical duplicate and interpolated duplicate baselines.

Table 6: Additional L1 metrics on chemical perturbation prediction (Tahoe).

	Tahoe (Zhang et al., 2025b)	
metric	State	Funomics
Rank (L1) ↓	0.00	0.10
Rank.T (L1) ↓	0.00	0.09
SMD (L1) ↓	11129.07	6392.14

E.1 Condition Encoder

The condition encoder processes condition embeddings $E_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$ through multiple layers of bidirectional self-attention, allowing each condition token to attend to all other condition tokens. This enables the encoder to capture relationships between different condition components (*e.g.*, perturbation target gene, dosage, task type) and produce encoded condition representations $H_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$ that integrate information across all condition elements. The encoder uses standard transformer layers with self-attention, feed-forward networks, RMSNorm normalization, and residual connections, following the T5Gemma (Zhang et al., 2025a) architecture.

E.2 Decoder

The decoder operates on a fixed set of N_{latent} decoder input embeddings $L_{\text{init}} \in \mathbb{R}^{N_{\text{latent}} \times d}$, where $N_{\text{latent}} \ll G$ (typically $N_{\text{latent}} \approx 256$ while $G \approx 19{,}000$). These decoder inputs are obtained by embedding a fixed vocabulary of N_{latent} tokens through learned embedding lookups, rather than randomly initialized parameters. The decoder uses a dual cross-attention mechanism to condition on both gene expression data and experimental conditions.

Conditioning on Gene Expression: The decoder applies cross-attention to the combined gene embeddings $E_{\text{input}} = E_{\text{gene}} + E_{\text{counts}}$, where queries come from the latent tokens and keys/values come from the gene embeddings. This allows the latent tokens to selectively attend to relevant genes from the high-dimensional input, enabling the model to focus on genes that are most informative for the prediction task.

Conditioning on Experimental Context: The decoder also applies cross-attention to the encoded condition embeddings H_{cond} from the encoder, enabling the model to condition its predictions on the specific experimental context (e.g., which gene is perturbed, what dosage is used, what task is being performed). This encoder-decoder cross-attention mechanism connects the condition encoder and decoder, allowing condition information to guide the gene expression processing.

Information Flow: Within each decoder layer, self-attention among latent tokens enables information exchange, followed by the two cross-attention mechanisms (to genes and to conditions), and feed-forward processing. The decoder produces compressed latent states $L \in \mathbb{R}^{N_{\text{latent}} \times d}$ that integrate information from both the gene expression input and the experimental conditions. These latent states are then projected back to gene space via projection cross-attention (described in subsection E.3), producing gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G \times d}$ that serve as input to task-specific heads.

E.3 PROJECTION CROSS-ATTENTION

After the decoder produces compressed latent states L, a projection cross-attention mechanism maps these latent representations back to gene space, producing gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G \times d}$. This projection step is essential for task-specific heads that operate on gene-level representations rather than compressed latent tokens.

The projection cross-attention uses gene embeddings E_{gene} as queries and the latent states L as keys and values. Specifically, after applying RMSNorm normalization to the latent states, the projection cross-attention computes:

$$H_{\text{genes}} = \text{CrossAttention}(Q = E_{\text{gene}}, K = L', V = L'), \tag{1}$$

where $L' = \text{RMSNorm}(L)$ are the normalized latent states. This mechanism allows each gene embedding to attend to relevant information from the compressed latent representation, effectively decompressing the latent states back to the full gene space while preserving the information captured during compression.

The projection cross-attention enables task-specific heads to operate directly on gene-level representations H_{genes}, which is particularly important for tasks like perturbation prediction that require gene-wise predictions. This design maintains the computational efficiency of the compressed latent representation during processing while providing full-gene expressivity for downstream task heads.

E.4 NORMALIZATION

Gene token embeddings, counts embeddings, and all intermediate representations are normalized using RMSNorm (Root Mean Square Layer Normalization). For an embedding $E \in \mathbb{R}^d$, RMSNorm is defined as:

$$E' = \text{RMSNorm}(E) = \frac{E}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} E_i^2 + \epsilon}} \odot \gamma, \tag{2}$$

where $\gamma \in \mathbb{R}^d$ is a learnable scale parameter and ϵ is a small constant for numerical stability. RMSNorm is computationally efficient and has been shown to work well in transformer architectures. Detailed descriptions of the embedding modules (gene token embeddings, counts embeddings, and condition embeddings) are provided in Appendix H.

E.5 ARCHITECTURAL CHOICES

Set Transformer Design: The architecture treats inputs as unordered sets without positional embeddings, making it permutation-invariant. This is crucial for gene expression data where gene order is arbitrary and should not affect predictions.

Dimensionality Reduction: By choosing $N_{\text{latent}} \ll G$, the model achieves significant dimensionality reduction, making it computationally tractable while preserving essential information. The compressed latent representation enables efficient processing of high-dimensional gene expression profiles.

T5Gemma Components: The use of T5Gemma components provides proven architectural patterns for encoder-decoder models, including efficient attention mechanisms and normalization strategies that have been validated in large-scale language models.

F TRAINING DETAILS

This section documents the concrete training configuration used for our reported model checkpoints, focusing on details needed for reproducibility. Dataset and task definitions are described elsewhere in the appendix.

F.1 MODEL CONFIGURATION (CHOSEN HYPERPARAMETERS)

We use the following model hyperparameters:

- **Number of latent tokens:** $N_{\text{latent}} = 512$.
- **Hidden dimension of latents in the decoder:** 1024.
- **Attention heads:** 8.
- **Attention dimension:** 128.
- **Decoder depth:** 16 layers.
- **Dropout:** 0.05.
- **Attention dropout:** 0.05.

F.2 Pretraining run configuration

Pretraining was run with the following settings.

- **Objectives:** pretraining jointly mixes count prediction and perturbation prediction tasks.
- **Batch size:** 64.
- **Learning rate:** 1×10^{-4} with a constant schedule and warmup fraction 0.05.
- **Training length:** 1M optimization steps.
- **Compute:** 7 days on 16 NVIDIA H100 GPUs (approximately 3000 GPU hours).
- **Implementation:** PyTorch with FlashAttention-2 enabled.
- **Pointers:** task definitions and losses are in Appendix I and Appendix J; datasets and split strategies are in Appendix B.

G Weight Computation for WMSE

Following the calibrated metrics framework Mejia et al. (2025) weights w_i are used in WMSE to prioritize genes of biological importance. However, we identified a problem with the standard weight computation approach from Mejia et al. (2025) To address this limitation, we propose stratified weights that provide improved gene prioritization.

H Embeddings

This section provides detailed descriptions of all embedding modules used in the Funomics T0 model, including gene token embeddings, counts embeddings, and condition embeddings for experimental contexts.

H.1 Gene Token Embeddings

Each gene is represented by a learnable token embedding $E_{\text{gene}} \in \mathbb{R}^{G \times d}$, where G is the number of genes and d is the embedding dimension. Gene token embeddings are randomly initialized and learned during training. The embedding matrix E_{gene} provides then a representation of each gene in the unified gene set, enabling the model to learn gene-specific features.

Optionally, external gene embeddings can be integrated to provide prior knowledge. When external embeddings $E_{\text{external}} \in \mathbb{R}^{G \times d_{\text{ext}}}$ are available, they are projected through a linear layer to match the model's embedding dimension: $E_{\text{projected}} = E_{\text{external}} W_{\text{proj}} + b_{\text{proj}}$, where $W_{\text{proj}} \in \mathbb{R}^{d_{\text{ext}} \times d}$ and $b_{\text{proj}} \in \mathbb{R}^d$ are learnable parameters. The projected embeddings are then added to the token embeddings: $E_{\text{gene}} = E_{\text{gene}} + E_{\text{projected}}$, similarly to the positional embeddings used in classical transformer architecture (Vaswani et al., 2017).

H.2 Counts Embeddings

Binning Approach: Following Adduri et al. (2025), continuous counts $x_{b,g} \in \mathbb{R}$ are embedded using a learned binning function that maps count values to a weighted combination of bin embeddings. The binning function uses two linear layers with a LeakyReLU activation to produce soft weights over

bins, which are then used to compute a weighted sum of learnable bin embeddings. This approach provides a discrete-like representation that is well-suited for attention mechanisms and can capture non-linear relationships through the learned bin boundaries. Special mask and pad tokens are used for masked and padded positions respectively.

H.3 Condition Embeddings

Condition embeddings encode experimental conditions and task information, allowing the model to condition its predictions on the specific context. All condition embeddings are concatenated into a condition sequence $E_{\text{cond}} \in \mathbb{R}^{N_{\text{cond}} \times d}$, where N_{cond} varies based on the perturbation type and task.

H.3.1 Genetic Perturbation Embeddings

For genetic perturbations, the perturbation target gene is embedded using the same gene token embedding procedure as the input gene embeddings. Specifically, if the perturbation targets gene g_{target}, the condition embedding is $E_{\text{genetic}} = E_{\text{gene}}[g_{\text{target}}] \in \mathbb{R}^d$, where E_{gene} is the same gene token embedding matrix used for input genes. This ensures consistent representation of genes across the model, enabling the model to leverage learned gene relationships when predicting perturbation effects.

H.3.2 Chemical Perturbation Embeddings

For chemical perturbations, multiple components are combined to form the condition embedding. Molecular embeddings $E_{\text{mol}} \in \mathbb{R}^{d_{\text{mol}}}$ are obtained from external sources (e.g., molecular graph encoders or pre-trained chemical embeddings) and projected through an MLP to match the embedding dimension: $E_{\text{mol,proj}} = \text{MLP}_{\text{mol}}(E_{\text{mol}}) \in \mathbb{R}^d$. Dosage information is embedded through a learnable dosage embedding: $E_{\text{dosage}} = E_{\text{dosage\_emb}}[\text{bin}(d)] \in \mathbb{R}^d$, where d is the dosage value and bin maps it to a discrete dosage bin. When available, molecules's target gene embeddings $E_{\text{target}} \in \mathbb{R}^d$ are included using the gene token embedding procedure. The final chemical perturbation embedding is constructed as: $E_{\text{chemical}} = [E_{\text{mol,proj}}; E_{\text{dosage}}; E_{\text{target}}] \in \mathbb{R}^{N_{\text{chem}} \times d}$, where N_{chem} depends on whether gene target information for a drug is available.

H.3.3 Task Embeddings

Task embeddings are learnable embeddings that encode the task type, allowing the model to condition its behavior on the specific prediction task. Given a vocabulary of N_{tasks} task types (*e.g.*, perturbation prediction, unmasking, annotation), each task t is represented by a learnable embedding $E_{\text{task}}[t] \in \mathbb{R}^d$. The task embedding vocabulary size N_{tasks} and embedding dimension d are hyperparameters that determine the model's capacity to distinguish between different task types.

I Task Descriptions

This section provides detailed descriptions of each task implemented in the Funomics T0 model, including task formulations, loss functions, and references to task-specific data details.

I.1 Perturbation Predictions

I.1.1 Genetic Perturbations

In the genetic perturbation prediction task, the model predicts perturbed cell expression profiles given control cell counts and genetic perturbation conditions. For a batch of B cell groups, each containing N cells, the model receives control counts $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$ and produces predictions $\hat{X} \in \mathbb{R}^{B \times G}$ for the perturbed state, where G is the number of genes. The target perturbed counts are $X_{\text{pert}} \in \mathbb{R}^{B \times G}$. Predictions are optionally normalized with scanpy's `log1p(normalize_total)` and can be computed in residual mode by adding control means or average perturbation effects.

Task-specific data details, datasets, and split strategies are provided in subsection B.2.

Loss Function For genetic perturbation prediction, we use a combination of Maximum Mean Discrepancy (MMD) loss and bulk mean squared error (MSE) loss. The MMD term $\mathcal{L}_{\text{MMD}} = \text{MMD}^2(\hat{X}, X_{\text{pert}})$ compares the distributions of predicted and target perturbed expressions, computed using the `geomloss` package with an energy kernel Bunne et al. (2023). Crucially, during training, each batch is constructed to contain all cells from the same cell line and perturbation condition, allowing the MMD loss to be computed over the entire set of predicted and observed cells for that condition. This distributional approach is necessary because there is no cell-level correspondence between control and perturbed populations due to the process of gene expression value measurement which is destructive for the cells. Therefore, rather than using point-wise losses that would require such matching, we measure distributional similarity between the predicted and observed perturbed cell sets. This loss captures the full distributional similarity between predicted and observed cell populations, preserving heterogeneity in perturbation responses. The bulk MSE loss $\mathcal{L}_{\text{bulk}} = \|\bar{\hat{X}} - \bar{X}_{\text{pert}}\|^2$ compares mean expressions across cells, where $\bar{\hat{X}} = \frac{1}{N}\sum_{n=1}^{N} \hat{X}_{:,n}$ and $\bar{X}_{\text{pert}} = \frac{1}{N}\sum_{n=1}^{N} X_{\text{pert},:,n}$, ensuring accurate prediction of average perturbation effects. The total perturbation loss is $\mathcal{L}_{\text{pert}} = 0.2 \cdot \mathcal{L}_{\text{MMD}} + 10 \cdot \mathcal{L}_{\text{bulk}}$. Optionally, gene-specific weights can be applied to emphasize differentially expressed genes, with weight computation details provided in Appendix G. A detailed explanation of the MMD formulation is provided in subsection J.1, and additional loss function details are described in Appendix J.

I.1.2 Chemical Perturbations

In the chemical perturbation prediction task, the model predicts perturbed cell expression profiles given control cell counts and chemical perturbation conditions (including compound identity and dosage). The formulation is similar to genetic perturbations, but condition embeddings incorporate molecular representations and dosage information.

Loss Function For chemical perturbation prediction, we use the same loss function as for genetic perturbations: a combination of Maximum Mean Discrepancy (MMD) loss and bulk mean squared error (MSE) loss. As with genetic perturbations, batches are constructed to contain all cells from the same cell line and perturbation condition, enabling distributional comparison via MMD (see subsubsection I.1.1 for the rationale). The MMD term $\mathcal{L}_{\text{MMD}} = \text{MMD}^2(\hat{X}, X_{\text{pert}})$ compares the distributions of predicted and target perturbed expressions, computed using the `geomloss` package with an energy kernel Bunne et al. (2023). The bulk MSE loss $\mathcal{L}_{\text{bulk}} = \|\bar{\hat{X}} - \bar{X}_{\text{pert}}\|^2$ compares mean expressions across cells. The total perturbation loss is $\mathcal{L}_{\text{pert}} = 0.2 \cdot \mathcal{L}_{\text{MMD}} + 10 \cdot \mathcal{L}_{\text{bulk}}$. Optionally, gene-specific weights can be applied to emphasize differentially expressed genes, with weight computation details provided in Appendix G. A detailed explanation of the MMD formulation is provided in subsection J.1, and additional loss function details are described in Appendix J.

I.2 Unmasking Perturbation Prediction

In the unmasking perturbation prediction task, a subset of genes is masked during training, and the model predicts expression values for the masked genes. The model receives control counts $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$ with a randomly masked subset of genes, and produces predictions $\hat{X} \in \mathbb{R}^{B \times G}$ for all genes, where B is the batch size and G is the number of genes.

We perform masking directly on the input count matrix $X_{\text{ctrl}} \in \mathbb{R}^{B \times G}$: a binary mask $M \in \{0, 1\}^{B \times G}$ is defined where $M_{b,g} = 1$ indicates that the count value for gene g in batch b is masked (not observed), and $M_{b,g} = 0$ indicates it is observed. For each masked entry in X_{ctrl}, the corresponding value is replaced during count embedding with a special mask token embedding. The model is then trained to predict the original observed count value at every masked position.

We employ a random masking strategy. For random masking, each element $M_{b,g}$ is drawn from a Bernoulli distribution with parameter p_{mask}, *i.e.*, $M_{b,g} \sim \text{Bernoulli}(p_{\text{mask}})$ for all b, g. The masking probability is task-specific and applied uniformly across genes.

Loss Function For unmasking tasks, we use MSE loss computed on masked positions during training, where the model is required to predict the original expression values. Given a binary mask $M \in \{0, 1\}^{B \times G}$ where $M_{b,g} = 1$ indicates that gene g in batch b is masked, the unmasking loss

is: $\mathcal{L}_{\text{mask}} = \frac{1}{|\mathcal{M}|}\sum_{(b,g)\in\mathcal{M}}(\hat{X}_{b,g} - X_{\text{ctrl},b,g})^2$, where $\mathcal{M} = \{(b,g) : M_{b,g} = 1\}$ is the set of masked positions. This formulation encourages the model to learn to reconstruct expression values from partial observations, improving its ability to handle missing data and learn robust gene-gene relationships. Additional details are provided in Appendix J.

I.3 Annotation Tasks

I.3.1 Tissue Annotation

In the tissue annotation task, the goal is to predict the tissue of origin of a given sample, based on its expression profile. The formulation is similar to cell type annotation, but with tissue-specific labels.

Task-specific data details, datasets, and split strategies are provided in subsection B.3.

Loss Function For annotation tasks (cell type and tissue classification), we use cross-entropy loss to train the model for multi-class classification. The loss is defined as: $\mathcal{L}_{\text{annot}} = -\frac{1}{B}\sum_{b=1}^{B}\sum_{c=1}^{C} y_{b,c}\log(\hat{y}_{b,c})$, where $y_{b,c} \in \{0,1\}$ is the true one-hot label indicating whether cell b belongs to class c, $\hat{y}_{b,c} \in [0,1]$ is the predicted probability for class c (obtained via softmax), B is the batch size, and C is the number of classes. This loss encourages the model to assign high probability to the correct class while minimizing probability for incorrect classes. Additional details are provided in Appendix J.

J Loss Function Details

J.1 Maximum Mean Discrepancy

Given samples $\{x_i\}_{i=1}^m \sim P$ and $\{y_j\}_{j=1}^n \sim Q$, the squared Maximum Mean Discrepancy (MMD) between the two distributions is estimated empirically as

$$\text{MMD}^2(P,Q) = \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i,x_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(y_i,y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,y_j), \tag{3}$$

where $k(\cdot,\cdot)$ denotes a kernel function that determines the geometry of the comparison.

In our implementation, we adopt the *energy kernel*

$$k(x,y) = -\|x-y\|, \tag{4}$$

which is *conditionally negative definite* (CND). Although this kernel is not positive definite and therefore does not correspond to an inner product in a reproducing kernel Hilbert space (RKHS), it induces a well-defined metric between probability distributions. In particular, the resulting MMD coincides exactly with the *Energy Distance*, a classical statistical metric defined by

$$D_{\text{Energy}}(P,Q) = 2\,\mathbb{E}\|X-Y\| - \mathbb{E}\|X-X'\| - \mathbb{E}\|Y-Y'\|. \tag{5}$$

Using a CND kernel ensures that the empirical estimate above defines a proper probability metric: it is non-negative, equals zero if and only if $P = Q$, and satisfies the triangle inequality. Compared to positive-definite kernels such as the RBF, the energy kernel requires no bandwidth hyperparameters and captures global geometric discrepancies through pairwise Euclidean distances.

J.2 Bulk MSE Loss

The bulk MSE loss compares mean expressions across cells:

$$\mathcal{L}_{\text{bulk}} = \|\bar{\hat{X}} - \bar{X}_{\text{pert}}\|^2, \tag{6}$$

where $\bar{\hat{X}} = \frac{1}{N}\sum_{n=1}^{N}\hat{X}_{:,n}$ and $\bar{X}_{\text{pert}} = \frac{1}{N}\sum_{n=1}^{N} X_{\text{pert},:,n}$.

J.3 WEIGHTED MSE LOSS

The weighted MSE loss corresponds to the Weighted Mean Squared Error (WMSE) metric defined in subsection K.5. When used as a training objective, weighted MSE can be leveraged to drive the model towards focusing more on accurate predictions for genes that are significantly affected by perturbations. By assigning higher weights to genes with substantial perturbation-induced changes, the loss function emphasizes learning to predict these biologically important responses while still maintaining supervision across the full transcriptome.

J.4 UNMASKING TASK LOSS

For unmasking tasks, the model predicts expression values for masked genes. Given a binary mask $M \in \{0,1\}^{B\times G}$ where $M_{b,g} = 1$ indicates that gene g in batch b is masked, the unmasking loss is computed only on masked positions:

$$\mathcal{L}_{\text{mask}} = \frac{1}{|\mathcal{M}|} \sum_{(b,g)\in\mathcal{M}} (\hat{X}_{b,g} - X_{\text{ctrl},b,g})^2, \tag{7}$$

where $\mathcal{M} = \{(b,g) : M_{b,g} = 1\}$ is the set of masked positions, $\hat{X}_{b,g}$ is the predicted expression value, and $X_{\text{ctrl},b,g}$ is the ground truth expression value. This formulation trains the model to reconstruct missing expression values from observed genes, improving its ability to learn gene-gene relationships and handle incomplete data.

J.5 ANNOTATION TASK LOSS

For annotation tasks (cell type and tissue classification), we use cross-entropy loss. Given true one-hot labels $y \in \{0,1\}^{B\times C}$ and predicted probabilities $\hat{y} \in [0,1]^{B\times C}$ (obtained via softmax), the annotation loss is:

$$\mathcal{L}_{\text{annot}} = -\frac{1}{B}\sum_{b=1}^{B}\sum_{c=1}^{C} y_{b,c}\log(\hat{y}_{b,c}), \tag{8}$$

where B is the batch size and C is the number of classes. The predicted probabilities are computed as

$$\hat{y}_{b,c} = \frac{\exp(z_{b,c})}{\sum_{c'=1}^{C}\exp(z_{b,c'})},$$

where $z_{b,c}$ are the logits from the classification head.

J.6 MULTI-TASK LOSS WEIGHTING

When training on multiple tasks simultaneously, the total loss is a weighted combination of task-specific losses. Given tasks $t \in \{1,2,\ldots,T\}$ with corresponding losses $\mathcal{L}_t$ and weights λ_t, the total loss is:

$$\mathcal{L}_{\text{total}} = \sum_{t=1}^{T} \lambda_t \mathcal{L}_t. \tag{9}$$

The weights λ_t can be set manually based on task importance, learned adaptively during training, or determined through hyperparameter search. For perturbation prediction tasks, gene-specific weights can be applied within the loss terms.

K METRICS

K.1 PREPROCESSING OF PREDICTIONS AND TARGETS

Before computing metrics, predictions and targets may be preprocessed using various transformations to improve metric sensitivity and interpretability. We establish the following notation for preprocessing operations:

Base (no preprocessing): Raw expression values are used without transformation, denoted as "$-$" in tables.

Control centering (Δ_{ctrl}): Expression values are centered by subtracting the mean control expression:

$$\mathbf{x}_{\Delta_{\text{ctrl}}} = \mathbf{x} - \bar{\mathbf{x}}_{\text{ctrl}}, \tag{10}$$

where $\bar{\mathbf{x}}_{\text{ctrl}}$ is the mean expression across all control cells. This transformation removes baseline expression levels and emphasizes perturbation-specific changes.

Train perturbation centering (Δ_{train}): Expression values are centered by subtracting the mean training perturbation expression:

$$\mathbf{x}_{\Delta_{\text{train}}} = \mathbf{x} - \bar{\mathbf{x}}_{\text{train}}, \tag{11}$$

where $\bar{\mathbf{x}}_{\text{train}}$ is the mean expression across all training perturbations. This approach increases the dynamic range of metrics, particularly for datasets involving essential genes where perturbations may share effects on basic cellular processes Wu et al. (2025).

K.2 Cosine Similarity

Cosine similarity measures the angular similarity between two vectors, providing a scale-invariant measure of directional agreement. We use two variants of cosine similarity depending on the evaluation context.

K.2.1 Cosine Similarity

Regular cosine similarity is computed between predicted and ground truth perturbation response vectors without any filtering. For a predicted vector $\hat{\mathbf{x}}^i$ and ground truth vector $\mathbf{x}^i$ for perturbation i, cosine similarity is defined as:

$$\text{cosine}(\hat{\mathbf{x}}^i, \mathbf{x}^i) = \frac{\hat{\mathbf{x}}^i \cdot \mathbf{x}^i}{\|\hat{\mathbf{x}}^i\|_2 \|\mathbf{x}^i\|_2}, \tag{12}$$

where $\cdot$ denotes the dot product and $\|\cdot\|_2$ is the L2 norm. This metric ranges from -1 to 1, with values closer to 1 indicating better directional agreement. Higher values indicate better performance.

K.2.2 Cosine Similarity for Response Direction

Cosine similarity for response direction evaluates the directional agreement specifically for perturbations with significant effect sizes. This variant focuses on perturbations that produce meaningful biological changes, filtering out perturbations with negligible effects.

When calculating cosine similarity for response direction, a specific centering step is employed to improve the metric's effectiveness (Littman et al., 2025). The average log-fold change (logFC) perturbation response in the training dataset is subtracted from both the prediction and the ground truth:

$$\text{cosine}_{\text{response}}(\hat{\mathbf{x}}^i, \mathbf{x}^i) = \frac{(\hat{\mathbf{x}}^i - \bar{\mathbf{x}}_{\text{train}}) \cdot (\mathbf{x}^i - \bar{\mathbf{x}}_{\text{train}})}{\|(\hat{\mathbf{x}}^i - \bar{\mathbf{x}}_{\text{train}})\|_2 \|(\mathbf{x}^i - \bar{\mathbf{x}}_{\text{train}})\|_2}, \tag{13}$$

where $\bar{\mathbf{x}}_{\text{train}}$ is the mean perturbation response across the training dataset. This approach of centering at the mean perturbation response is adopted to increase the dynamic range of the metric (Littman et al., 2025). This centering is particularly important for datasets involving essential genes, where perturbations might share effects on basic cellular processes. This centering strategy deviates from conventional approaches used in prior work (Littman et al., 2025).

K.3 Rank Metrics

Rank-based metrics measure perturbation discrimination by evaluating how model predictions compare to observations, following (Wu et al., 2025). This approach avoids the limitations of existing information retrieval metrics such as mean reciprocal rank, which would require a specific desired cell state to create rankings, potentially biasing the evaluation. Rank metrics are computed on a per-perturbation basis and use a generic distance metric $\text{dist}(\cdot, \cdot)$ that can be L1 (Manhattan distance), L2 (Euclidean distance), or cosine similarity. The choice of distance metric can significantly affect the ranking (Liu et al., 2025). All rank metrics yield values between 0 and 1, where 0 is a perfect score and 0.5 is the expected score of a random prediction. Lower values indicate better performance, with a value of 0 indicating perfect discrimination.

K.3.1 $\text{RANK}_{\text{average}}$

The $\text{Rank}_{\text{average}}$ metric measures perturbation discrimination by evaluating, for a given observed perturbation, how close the model prediction is to the observation compared to predictions made for other perturbations. For a set of p perturbations, $\text{Rank}_{\text{average}}$ is defined as:

$$\text{Rank}_{\text{average}} := \frac{1}{p} \sum_{i=1}^{p} \text{rank}(\hat{x}^i), \tag{14}$$

where $\hat{x}^i$ is the predicted response for perturbation i, and the rank of prediction i is computed as:

$$\text{rank}(\hat{x}^i) := \frac{1}{p-1} \sum_{j=1, j\neq i}^{p} \mathbb{I}\big(\text{dist}(\hat{x}^j, x^i) \leq \text{dist}(\hat{x}^i, x^i)\big), \tag{15}$$

where the sum is over all j from 1 to p such that $j \neq i$, x^i is the true (average) expression value for perturbation i, and $\mathbb{I}(\cdot)$ is the indicator function. The rank of prediction i represents the fraction of other predictions that are at least as close to the true response x^i as the prediction $\hat{x}^i$ itself. $\text{Rank}_{\text{average}}$ is robust to scaling of predictions: adding a constant factor to all model predictions will not affect the metric, as it only depends on the relative ordering of predictions with respect to each true response. This makes $\text{Rank}_{\text{average}}$ particularly useful for assessing whether model predictions are ordered correctly relative to the ground truth perturbations. The calibration of this rank metric was investigated by Wu et al. (2025).

K.3.2 $\text{RANK.T}_{\text{average}}$

The $\text{Rank.T}_{\text{average}}$ metric is a transposed variant of $\text{Rank}_{\text{average}}$ that measures perturbation discrimination by evaluating, for a given predicted perturbation response, how close it is to its corresponding observation compared to other observations. For a set of p perturbations, $\text{Rank.T}_{\text{average}}$ is defined as:

$$\text{Rank.T}_{\text{average}} := \frac{1}{p} \sum_{i=1}^{p} \text{rankT}(\hat{x}^i), \tag{16}$$

where $\hat{x}^i$ is the predicted response for perturbation i, and the rankT of prediction i is computed as:

$$\text{rankT}(\hat{x}^i) := \frac{1}{p-1} \sum_{j=1, j\neq i}^{p} \mathbb{I}\big(\text{dist}(\hat{x}^i, x^j) \leq \text{dist}(\hat{x}^i, x^i)\big), \tag{17}$$

where the sum is over all j from 1 to p such that $j \neq i$, x^i is the true (average) expression value for perturbation i, and $\mathbb{I}(\cdot)$ is the indicator function. Unlike $\text{Rank}_{\text{average}}$ K.3.1, which compares different predictions to the same true response, $\text{Rank.T}_{\text{average}}$ compares a single prediction to different true responses. The rankT of prediction i represents the fraction of other true responses that are at least as close to the prediction $\hat{x}^i$ as the true response x^i itself. Unlike $\text{Rank}_{\text{average}}$, $\text{Rank.T}_{\text{average}}$ is sensitive to the scaling of predictions: adding a constant factor to all model predictions will affect the metric, as it directly compares the absolute distances between predictions and true responses. This makes $\text{Rank.T}_{\text{average}}$ useful for users who want a stricter assessment of whether the predictions are similar to the ground truth perturbations, beyond just the ordering of the perturbations.

K.4 SIMILARITY MATRIX DISTANCE

The Similarity Matrix Distance (SMD) metric measures the preservation of relations between perturbations by quantifying the discrepancy between predicted and ground truth similarity matrices, computed on pseudobulked data, following Wu et al. (2025). A similarity matrix $S \in \mathbb{R}^{n\times n}$ captures pairwise distances for n perturbations, where each entry s_{ij} represents the distance between perturbation i and perturbation j, computed from their aggregated expression vectors $x^{(i)}, x^{(j)}$ using a generic distance metric $\text{dist}(\cdot,\cdot)$ that can be L1 (Manhattan distance), L2 (Euclidean distance), or cosine distance. Specifically, for L1 and L2 distances, $s_{ij} = \|x^{(i)} - x^{(j)}\|_p$ where $p = 1$ for L1 and $p = 2$ for L2, and for cosine distance, $s_{ij} = 1 - \text{cosine\_similarity}(x^{(i)}, x^{(j)})$.

We denote by $\hat{S} \in \mathbb{R}^{n\times n}$ the similarity matrix constructed from predicted responses, and by $S \in \mathbb{R}^{n\times n}$ the matrix for ground truth responses, with entries $\hat{s}_{ij}$ and s_{ij}, respectively. The matrices are compared using the Frobenius norm.

K.4.1 BASIC SMD

We define the Similarity Matrix Distance by comparing the similarity matrices using the Frobenius norm:

$$\mathrm{SMD}(\hat{S}, S) = \left\|\hat{S} - S\right\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\hat{s}_{ij} - s_{ij}\right)^2}, \tag{18}$$

where $\hat{s}_{ij}$ and s_{ij} are the similarity values between perturbations i and j in the predicted and ground truth matrices, respectively. Lower values indicate better performance, as they represent smaller differences between predicted and ground truth similarity structures. Expression data from all compared models and baselines were normalized in the same way: total-count normalization to 10k counts per cell followed by $\log(1 + x)$.

K.5 WEIGHTED MEAN SQUARED ERROR (WMSE)

Following the calibrated metrics framework (Mejia et al., 2025), we evaluate perturbation predictions using Weighted Mean Squared Error (WMSE), a weighted variant of the standard MSE metric computed on pseudobulked data. For a single perturbation, WMSE is defined as:

$$\mathrm{WMSE} = \sum_{i=1}^{G} w_i(\mu_p^i - \hat{\mu}_p^i)^2, \tag{19}$$

where G is the number of genes, w_i is the weight for gene i, μ_p^i is the observed mean expression for gene i under perturbation p, and $\hat{\mu}_p^i$ is the predicted mean expression for gene i under perturbation p. Gene-specific weights w_i can be assigned to emphasize particular genes of interest (see Appendix G for weight computation details).

WMSE addresses limitations of unweighted error metrics by incorporating gene-level importance (Mejia et al., 2025). The weighting mechanism enables focus on genes that exhibit substantial changes in response to perturbations, such as differentially expressed genes, rather than treating all genes equally. This selective emphasis is crucial because most genes remain relatively unchanged following perturbations, making it important to prioritize evaluation on genes with meaningful responses. Beyond evaluation, WMSE serves as a viable training objective that can replace standard MSE, providing models with supervision that reflects biological relevance. Lower values indicate better performance, corresponding to reduced weighted prediction errors.

K.6 MEAN SQUARED ERROR (MSE)

MSE is a special case of WMSE K.5 with uniform weights $w_i = 1/G$ for all genes:

$$\mathrm{MSE} = \frac{1}{G}\sum_{i=1}^{G}(\mu_p^i - \hat{\mu}_p^i)^2. \tag{20}$$

Lower values indicate better performance.

K.7 MAXIMUM MEAN DISCREPANCY (MMD)

Maximum Mean Discrepancy (MMD) measures the distributional similarity between predicted and observed perturbed cell populations (Bunne et al., 2023). This metric is used to evaluate how well the model captures the full distribution of cellular responses to perturbations, rather than just point estimates or mean behavior. By comparing the distributions of predicted and observed cells, MMD provides a comprehensive assessment of whether the model preserves the heterogeneity and variability present in real perturbation responses.

The mathematical formulation of MMD is provided in subsection J.1. We follow the approach from (Bunne et al., 2023), using the energy kernel which corresponds to the Energy Distance metric. Compared to positive-definite kernels such as the RBF, the energy kernel requires no bandwidth hyperparameters.

L BASELINES

L.1 PERTURBATION PREDICTION

L.1.1 POST-PERTURBED MEAN

The post-perturbed mean baseline provides a dataset-wide reference by aggregating mean expression profiles across all training perturbations (Miller et al., 2025). For M training perturbations with cell sets $S_1, S_2, \ldots, S_M$ of potentially different sizes, we compute:

$$\mu_{\text{all}} = \frac{1}{M} \sum_{i=1}^{M} \left(\frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_{i,j} \right), \tag{21}$$

where $\mathbf{x}_{i,j}$ denotes the expression vector of cell j from perturbation i. This formulation applies uniform weighting to each perturbation's mean profile, regardless of the number of cells per perturbation. The resulting baseline represents a global average that smooths over perturbation-specific effects and cell-to-cell variability, serving as a simple but informative lower bound for model performance.

L.1.2 TECHNICAL DUPLICATE BASELINE

The technical duplicate baseline establishes an upper performance bound by leveraging the inherent variability within experimental replicates (Miller et al., 2025). For each perturbation p_i with associated cells S_i, we randomly partition the cells into two groups: $S_{i,\text{GT}}$ serves as ground truth, while $S_{i,\text{TD}}$ functions as a technical duplicate prediction. Mean expression profiles $\mu_{i,\text{GT}}$ and $\mu_{i,\text{TD}}$ are computed for each group and compared. This approach isolates the contribution of experimental noise to prediction error, effectively asking how well one experimental replicate predicts another under identical conditions. The baseline's reliability depends on having sufficient cells per perturbation; with few cells, the random split introduces substantial variance. To maintain evaluation integrity, technical duplicate cells are treated as an independent held-out set and excluded from all differential expression analyses used for metric weighting.

L.1.3 INTERPOLATED DUPLICATE BASELINE

The interpolated duplicate baseline addresses the technical duplicate baseline's limitations for perturbations with weak effects by adaptively blending predictions based on gene-level evidence of perturbation impact (Miller et al., 2025). For each perturbation, we construct a per-gene weighted combination:

$$\mu_{i,\text{ID}} = \alpha \odot \mu_{i,\text{TD}} + (1 - \alpha) \odot \mu_{\text{all}}, \tag{22}$$

where α is a gene-specific weight vector and $\odot$ represents element-wise multiplication. The weights are derived from differential expression analysis on the technical duplicate set: $\alpha = 1 - p_{\text{DEGs}}$, where p_{DEGs} contains p-values for each gene. Genes showing strong perturbation effects receive weights near 1, prioritizing the technical duplicate prediction, while genes with no significant changes receive weights near 0, defaulting to the mean baseline. This adaptive strategy provides a more realistic performance estimate by acknowledging that technical duplicates are most informative for genes with clear perturbation signatures.

M ADDITIONAL IMPLEMENTATION DETAILS

This section provides additional implementation details on the task-specific model heads.

Task-specific heads process the compressed latent states $L \in \mathbb{R}^{N_{\text{latent}} \times d}$ from the backbone to produce task-specific outputs. Each head architecture is tailored to the specific prediction task.

Perturbation Prediction Head For perturbation prediction, the head outputs gene expression vectors $\hat{X} \in \mathbb{R}^{B \times G}$. The latent states are projected back to gene space using a cross-attention layer that produces gene-level hidden states $H_{\text{genes}} \in \mathbb{R}^{G \times d}$. These gene-level representations are then mapped to expression predictions via a linear projection: $\hat{X} = H_{\text{genes}} W_{\text{out}} + b_{\text{out}}$, where $W_{\text{out}} \in \mathbb{R}^{d \times G}$ and $b_{\text{out}} \in \mathbb{R}^{G}$ are learnable parameters.

Unmasking Head For unmasking tasks, the head architecture is similar to perturbation prediction, outputting expression values for all genes including masked positions. The head processes latent states to reconstruct the full expression profile: $\hat{X} = \text{Head}_{\text{unmask}}(L) \in \mathbb{R}^{B \times G}$.

Annotation Head (Classification) For annotation tasks (cell type and tissue classification), the head outputs class probabilities. The latent states are aggregated to $h \in \mathbb{R}^d$, then projected through a linear layer to produce logits: $z = hW_{\text{cls}} + b_{\text{cls}} \in \mathbb{R}^C$, where C is the number of classes. The final predictions are obtained via softmax: $\hat{y} = \text{softmax}(z) \in [0, 1]^C$.