Supplemental Methods Results Generation

1 Geneformer

Geneformer was implemented using the original repository[1] as a BERT-based transformer (BertForMaskedLM). Training was performed using the HuggingFace Trainer API with length-grouped batching to improve efficiency by grouping cells with similar numbers of expressed genes. To ensure consistent training across diverse dataset sizes, epochs scale dynamically as $\max(1, \lfloor 10 \times (10,000,000/\text{dataset\_size}) \rfloor)$, ensuring smaller datasets receive proportionally more training while larger datasets train for fewer epochs, thus maintaining a constant effective training budget. Early stopping prevents overfitting by monitoring validation loss, and the model uses masked language modeling with 15% token masking.

Parameter count scales linearly with vocabulary size (number of genes) due to the embedding layer and language model head. Measured parameter counts across datasets range from 1.9M to 13.5M depending on the number of genes in each dataset (Table 3). Complete hyperparameters are provided in Table 1.

We use a reduced-parameter Geneformer architecture compared to published single-cell transformer models due to smaller training datasets and the large number of studies required. Larger models such as Geneformer-V1[1] (pretrained on ∼30M cells), Geneformer-V2-104M[2] (pretrained on ∼95M cells), Tahoe-X1[3] (∼70M parameters, pretrained on ∼271M cells), scGPT[4] (∼100M parameters, pretrained on ∼33M cells), TranscriptFormer[5] (368–542M parameters, pretrained on 57–112M cells), and STATE[8] (∼600M parameters) were trained with massive compute budgets and extensive pretraining data, making such large-scale pretraining infeasible for systematic benchmarking.

Figure 1 shows the training and validation loss curves for the Geneformer model trained on the developmental task using the full dataset of 10 million cells at full quality level. The model demonstrates stable convergence with both the training and validation losses decreasing monotonically, and the validation loss closely tracks the training loss, indicating good generalization without overfitting. Early stopping was triggered based on validation loss monitoring, preventing overfitting while allowing sufficient training.

2 SCVI

SCVI[6] was implemented using `scvi.model.SCVI` with a shallow architecture to balance expressiveness and computational efficiency. The model uses gene-specific dispersion and zero-inflated negative binomial (ZINB) likelihood to capture the over-dispersed and sparse nature of single-cell count data. Parameter count scales with input genes due to gene-specific parameters in the decoder. Measured parameter counts across datasets range from 1.3M to 116.8M depending on the number of genes in each dataset (Table 3). Complete hyperparameters are provided in Table 2.

Training employs a conservative learning rate schedule with minimal weight decay to allow stable optimization of the generative model. Similar to Geneformer, epochs scale dynamically to maintain consistent training across dataset sizes, though with a lower scaling factor reflecting the faster convergence of variational autoencoders. KL divergence warmup is applied to gradually introduce the regularization term, preventing the model from collapsing to the prior early in training. SCVI operates on raw count data without preprocessing. The encoder outputs both mean and variance parameters of the approximate posterior distribution over latent variables, but embeddings are extracted as the posterior mean to provide deterministic 16-dimensional representations, following standard practice in the field.

Table 1: Hyperparameters and Implementation Details for Geneformer

Parameter	**Value**
Model Architecture	
Number of layers	3
Hidden dimension	256
Attention heads	4
Intermediate size	512
Max sequence length	512 tokens
Activation function	ReLU
Attention dropout	0.02
Hidden dropout	0.02
Initializer range	0.02
Layer norm epsilon	1×10^{-12}
Training Hyperparameters	
Learning rate	1×10^{-3}
Optimizer	AdamW
Weight decay	0.001
Train batch size (per device)	64
Eval batch size (per device)	100
Learning Rate Schedule	
LR scheduler type	Linear
Warmup steps	5,000
Training Configuration	
Evaluation strategy	Steps
Evaluation steps	1,000
Max epochs	Dynamic (depends on dataset size)
Early Stopping	
Early stopping	Based on validation loss
Patience	5 steps

Table 2: Hyperparameters and Implementation Details for SCVI

Parameter	**Value**
Model Architecture	
Hidden size	512
Latent dimension	16
Number of hidden layers	1
Dropout rate	0.1
Dispersion	Gene-specific
Gene likelihood	Zero-inflated negative binomial (ZINB)
Latent distribution	Normal
Training Hyperparameters	
Learning rate	1×10^{-3}
Optimizer	Adam
Weight decay	1×10^{-6}
Adam epsilon	0.01
Batch size	512
Learning Rate Schedule	
LR scheduler	Reduce on plateau
LR scheduler metric	Validation ELBO
Minimum LR	1×10^{-6}
KL Annealing	
KL warmup epochs	1
Max KL weight	1.0
Min KL weight	0.0
Training Configuration	
Train/Val split	80% / 20%
Shuffle split	True
Max epochs	Dynamic (depends on dataset size)
Early Stopping	
Early stopping	Based on validation ELBO
Patience	5 epochs
Min delta	0.01

Table 3: Parameter counts for SCVI and Geneformer models across different datasets

Dataset	**SCVI (M)**	**Geneformer (M)**
Larry	64.9	8.3
MERFISH	1.3	1.9
PBMC	53.2	7.1
Shendure	116.8	13.5

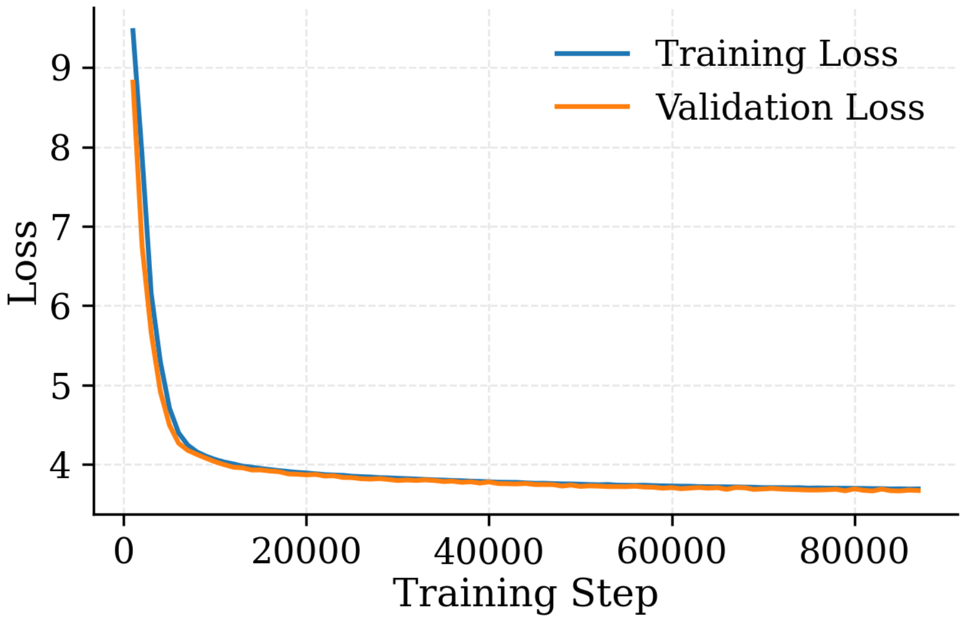

Figure 1: Training and validation loss curves for Geneformer trained on the developmental task with 10 million cells at quality level 1. The model shows stable convergence with both losses decreasing consistently throughout training.

3 PCA

PCA was implemented using `sklearn.decomposition.TruncatedSVD`[7] with 16 components. Preprocessing includes total count normalization to 10,000 counts per cell, $\log(x+1)$ transformation, and selection of the top 750 highly variable genes. The method computes a linear transformation matrix of size 750×16.

4 Random Projection

Random Projection was implemented using `sklearn.random_projection.GaussianRandomProjection`[7] with 16 components and random state 42. The method operates on raw counts without normalization and provides 16-dimensional embeddings.

5 Computational Scale

This study required the execution of approximately 4,000 training runs across tasks, dataset sizes, data qualities, algorithms, and different random seeds. Most models required GPU access for training, embedding extraction, or computation of mutual information. To manage this scale, we developed a custom scheduling system that dynamically allocates GPUs based on real-time memory availability, maintains hundreds of concurrent worker processes, and executes each training run as an isolated subprocess. The system monitors GPU memory usage and assigns jobs. All models and embeddings were saved for reproducibility. To enable processing of datasets containing up to 10 million cells, we developed a custom H5AD reader that handles sparse CSR matrices by directly accessing HDF5 indptr, indices, and data arrays, reconstructing sparse matrices only for requested row ranges. This enables efficient chunked reading with a low memory footprint.

References

[1] Theodoris, C. V., Xiao, L., Chopra, A., Chaffin, M. D., Al Sayed, Z. R., Hill, M. C., Mantineo, H., Brydon, E., Zeng, Z., Liu, X. S., & Ellinor, P. T. (2023). Transfer learning enables predictions in network biology. *Nature*, 618(7965), 616–624.

[2] Chen, H., Venkatesh, M. S., Gomez Ortega, J., Mahesh, S. V., Nandi, T., Madduri, R., Pelka, K., & Theodoris, C. V. (2024). Quantized multi-task learning for context-specific representations of gene network dynamics. *bioRxiv*. `https://www.biorxiv.org/content/10.1101/2024.08.16.608180v1`

[3] Gandhi, S., Javadi, F., Svensson, V., Khan, U., Jones, M. G., Yu, J., Merico, D., Goodarzi, H., & Alidoust, N. (2025). Tahoe-x1: Scaling Perturbation-Trained Single-Cell Foundation Models to 3 Billion Parameters. *bioRxiv*. `https://www.biorxiv.org/content/10.1101/2025.10.23.683759v1`

[4] Cui, H., Wang, C., Maan, H., Pang, K., Luo, F., & Wang, B. (2024). scGPT: toward building a foundation model for single-cell multi-omics using generative AI. *Nature Methods*, 21, 1470–1480.

[5] Pearce, J. D., Simmonds, S. E., Mahmoudabadi, G., Krishnan, L., Palla, G., Istrate, A.-M., Tarashansky, A., Nelson, B., Valenzuela, O., Li, D., Quake, S. R., & Karaletsos, T. (2025). A Cross-Species Generative Cell Atlas Across 1.5 Billion Years of Evolution: The TranscriptFormer Single-cell Model. *bioRxiv*. `https://www.biorxiv.org/content/10.1101/2025.04.25.650731v2`

[6] Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., & Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12), 1053–1058.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[8] Adduri, A. K., Gautam, D., Bevilacqua, B., Imran, A., Shah, R., Naghipourfar, M., Teyssier, N., Ilango, R., Nagaraj, S., Dong, M., Ricci-Tam, C., Carpenter, C., Subramanyam, V., Winters, A., Tirukkovular, S., Sullivan, J., Plosky, B. S., Eraslan, B., Youngblut, N. D., Leskovec, J., Gilbert, L. A., Konermann, S., Hsu, P. D., Dobin, A., Burke, D. P., Goodarzi, H., & Roohani, Y. H. (2025). Predicting cellular responses to perturbation across diverse contexts with State. *bioRxiv*. `https://www.biorxiv.org/content/10.1101/2025.06.26.661135v2`