Supplemental Methods Results Generation

1 Geneformer

Geneformer was implemented using the original repository[1] as a BERT-based transformer (BertForMaskedLM). The architecture was deliberately reduced in size compared to standard implementations to enable systematic scaling law evaluation across thousands of training runs. The model contains approximately 13.5 million trainable parameters (13,480,151 for typical datasets with 20,000–30,000 genes), with parameter count scaling linearly with vocabulary size due to the embedding layer. Complete hyperparameters are provided in Table 1.

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. This scaling strategy ranges from 1 epoch (10M cells) to 100,000 epochs (100 cells), 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 to learn gene-gene relationships.

Data preprocessing involves ranking genes by median expression to assign token IDs. Tokenization was performed using up to 64 parallel processes, partitioning data into 10–15 million cell chunks. Cell embeddings are extracted via mean pooling over all gene tokens from the final transformer layer, producing 256-dimensional vectors.

Our Geneformer implementation uses a smaller architecture than other single-cell transformer models (Table 2) to enable systematic scaling law evaluation across thousands of training runs. Compared to Geneformer-V1-10M[1] (6 layers, 2048 tokens, ∼30M parameters, pretrained on ∼30M cells), our model uses half the layers and a quarter of the sequence length, resulting in 2.2× fewer parameters. Larger variants including Geneformer-V2-104M[2] (12–18 layers, 4096 tokens, pretrained on ∼95M cells), Tahoe-X1[3] (12 layers, 1024 tokens, ∼70M parameters, pretrained on ∼271M cells), and scGPT[4] (12 layers, 5000 tokens, ∼100M parameters, pretrained on ∼33M cells) are computationally prohibitive for this evaluation, which requires training thousands of models across diverse dataset sizes, qualities, and random seeds.

1.1 Training Loss

Figure 1 shows the training and validation loss curves for Geneformer trained on the developmental task using the full 10 million cells dataset at quality level 1. The model demonstrates stable convergence with both training and validation losses decreasing monotonically. 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: 41M for ∼20,000 genes, 51M for ∼25,000 genes, and 61M for ∼30,000 genes. Complete hyperparameters are provided in Table 3.

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
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
Save steps	1,000
Max epochs	Dynamic (depends on dataset size)
Group by length	True
Early Stopping	
Early stopping	Based on validation loss
Patience	5 steps
Metric	eval_loss

Table 2: Comparison with other single-cell transformer models

Parameter	**Value**
Our Geneformer	
Number of layers	3
Hidden dimension	256
Attention heads	4
Intermediate size	512
Max sequence length	512 tokens
Total parameters	∼13.5M
Geneformer-V1-10M[1]	
Number of layers	6
Hidden dimension	256
Attention heads	4
Intermediate size	512
Max sequence length	2048 tokens
Total parameters	∼30M
Geneformer-V2-104M[2]	
Number of layers	12
Hidden dimension	768
Attention heads	12
Intermediate size	3072
Max sequence length	4096 tokens
Total parameters	∼104M
Tahoe-X1 (70M)[3]	
Number of layers	12
Hidden dimension	512
Attention heads	8
Intermediate size	2048
Max sequence length	1024 tokens
Total parameters	∼70M
scGPT (pretrained)[4]	
Number of layers	12
Hidden dimension	512
Attention heads	8
Intermediate size	512
Max sequence length	5000 tokens
Total parameters	∼100M

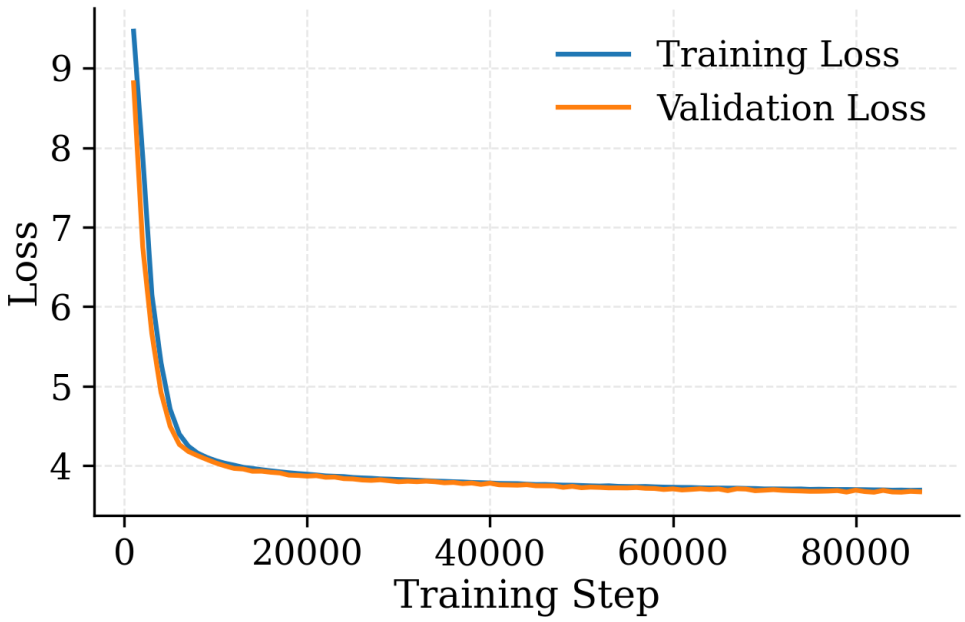

Figure 1: Training and validation loss curves for Geneformer trained on the developmental task with 10 million cells at quality level 1. The model shows stable convergence with both losses decreasing consistently throughout training.

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 as $\max(1, \lfloor 1 \times (10,000,000/\text{dataset\_size}) \rfloor)$ 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, leveraging its probabilistic framework to handle technical noise, and produces 16-dimensional embeddings as the posterior mean of the latent distribution.

3 PCA

PCA was implemented using `sklearn.decomposition.TruncatedSVD`[7] with 16 components and random state 0x5EED. 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 has no trainable parameters and 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 projection matrix is sampled from $\mathcal{N}(0, 1/16)$ and has no trainable parameters. Fitting uses only the first 5 cells to establish matrix dimensions. The method operates on raw counts without normalization and provides 16-dimensional embeddings.

5 Mutual Information Estimation

Mutual information between cell embeddings and biological signals was estimated using Latent Mutual Information (LMI), a neural network-based method that learns a low-dimensional representation optimized

Table 3: 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}
Batch size	512
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

for mutual information estimation. LMI was computed using the `latentmi` library with the following hyperparameters: validation split of 0.3, batch size of 512, and maximum epochs of 300. The method trains a neural network to predict the signal from embeddings, with pointwise mutual information computed for each cell. The final mutual information value is reported as the mean of pointwise estimates across all cells. This approach enables accurate estimation of mutual information for high-dimensional embeddings and continuous or discrete signals, making it suitable for evaluating how well different algorithms preserve biological information across varying data qualities and sizes.

6 Efficient H5AD File Reading

To enable efficient processing of datasets containing up to 10 million cells, we developed a custom chunked H5AD reader that allows memory-efficient access to large single-cell datasets stored in the HDF5-based AnnData format. Standard AnnData loading methods require loading entire datasets into memory, which becomes prohibitive for datasets exceeding available RAM. Our `H5adReader` implementation provides an iterator interface that reads data in configurable chunks (default 1,000 cells), enabling processing of datasets of arbitrary size while maintaining low memory footprint. The reader efficiently handles sparse CSR matrices by directly accessing HDF5 indptr, indices, and data arrays, reconstructing sparse matrices only for requested row ranges. This engineering contribution was essential for processing the largest datasets in our study, particularly the 10 million cell datasets that would otherwise require over 100GB of RAM to load entirely. The chunked reader also supports random sampling and selective row access, enabling efficient data preprocessing and quality-based subsampling without full dataset loading.

7 Computational Scale

This study required the execution of approximately 4,000 training runs across 4 datasets (Shendure, PBMC, Larry, MERFISH), 10 dataset sizes (100 to 10 million cells), 10 data quality levels, 4 algorithms (PCA, SCVI, Geneformer, RandomProjection), and typically 3 random seeds (42, 1404, 2701; Shendure used 1 seed). To manage this scale, we developed a custom scheduling system that dynamically allocates GPUs based on real-time memory availability, maintains up to 100 concurrent worker processes, and executes each training run as an isolated subprocess. The system monitors GPU memory usage and assigns jobs only when memory falls below algorithm-specific thresholds (240GB for Geneformer, 33GB for other methods). Epoch counts are dynamically computed based on dataset size for parametric methods, while non-parametric methods require no training configuration. All models and embeddings were saved to disk with comprehensive WandB logging.

8 References

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., et al. (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.