

A Hyper-parameters

We first define different hyper-parameters.

1. Dropout: The dropout rate is the probability of making the neural unit inactive or closed. If a unit is closed, it will not participate in the training process but will take part in the inference process. The design of dropout is to avoid or reduce overfitting. The scope is $[0, 0.1, 0.2, 0.3, 0.5]$.

2. Loss weight: Changing the weights of different loss function components is a well-known approach to increasing the performance of neural networks. Here the loss weight means the weight of the task-specific loss component. For example, for the Batch Effect Correction task, such weight represents the loss weight for the loss of the gradient reverse network. The scope is $[1, 5, 10, 50, 100]$.

3. Bins: To reduce the batch effect during the pre-training or fine-tuning process in the pre-processing step, the gene expression is divided into different intervals and the endpoint values of the intervals are used to replace the original gene expression. The intervals here are the bins. Formally, the binned expression value $x_j^{(i)}$ for cell i is:

$$x_j^{(i)} = \begin{cases} k, & \text{if } x_j^{(i)} > 0 \text{ and } x_j^{(i)} \in [b_k, b_{k+1}], \\ 0, & \text{if } x_j^{(i)} = 0, \end{cases}$$

where $k \in \{1, 2, \dots, B\}$ and B represents the number of bins. The scope is $[51, 101, 151, 201, 501, 1001]$.

4. Mask ratio: In the pre-training and fine-tuning process, we choose to mask a fixed ratio of gene expression levels and try to use our decoder to reconstruct the expression levels of the masked genes. Such a ratio is defined as a mask ratio. The scope is $[0.1, 0.3, 0.4, 0.5, 0.7, 0.9]$.

5. Epoch: Epoch represents the number of steps we use to train our model based on the whole training dataset. The scope is $[1, 5, 10]$.

6. Learning rate: Learning rate is designed for the optimizer, it represents the step size of gradient descent. The scope is $[5e-5, 1e-4, 1e-3, 1e-2]$.

7. ECS threshold: The threshold β in the ECS Loss represents the upper bound for the similarity between paired cells. The scope is $[0, 0.2, 0.4, 0.6, 0.8, 1.0]$.

B Initial Settings

In this section, we delineate the initial settings for Batch Effect Correction, Multi-omics Data Integration, Cell-type Annotation and Perturbation Prediction tasks.

1. Zero-shot learning: This refers to using our pre-trained model without any further training for unseen datasets on specific tasks. This setting aids in validating the significance of pre-training.

2. No pre-training: This setting entails starting the fine-tuning process without any pre-trained weights, allowing us to train a *de novo* single-cell LLM. This configuration helps ascertain the importance of pre-training.

3. Cross-entropy (default setting): Here, we utilize the cross-entropy loss function for the reconstruction of gene expression levels, which is the default setting in scGPT.

4. Mean Squared Error (MSE) loss: In this setting, we employ the MSE loss function for the reconstruction of gene expression levels.

5. Freeze except encoder: This setting involves freezing the weights of the encoder component during the fine-tuning process.

6. Freeze except decoder: Here, we freeze the weights of the decoder section during the fine-tuning process.

7. Training by reward network: The successful application of Human-feedback Reinforcement Learning (HFRL) to the training process for LLMs has been established [98], and we borrow ideas from [65, 99]. We consider cell types as the human label for different cells, akin to the labels of sentences in the NLP area. A reward network is employed to predict the cell-types of the provided cells and minimize the loss in this process. This procedure can also be integrated into the pre-training process or prompt-based learning framework. This approach can also be regarded as joint fine-tuning for Batch Effect Correction and Cell-type Annotation.

The results from this analysis provide valuable insights into how to configure scGPT most effectively for this specific task.

C Metrics Information

Here we describe details about the metrics we used in the evaluation process for different tasks.

C.1 Batch Effect Correction, Multi-omics Data Integration and Simulation

1. Normalized Mutual Information (NMI): NMI is a score to evaluate the performance of biological information preservation. We calculate this score based on computing the mutual information between the optimal Leiden clusters and the known cell type labels and then take the normalization. Therefore, $NMI \in (0, 1)$ and higher NMI means better performance.

2. Adjusted Rand Index (ARI): ARI is a score to evaluate the performance of biological information preservation. ARI can measure the agreement between optimal Louvain clusters and cell type labels. $ARI \in (0, 1)$ and higher ARI means better performance.

3. Average Silhouette Width (ASW): Here we have cell type ASW (ASW_{cell}) and batch ASW (ASW_{batch}). For one cell point, ASW calculates the ratio between the inner cluster distance and the intra cluster distance for this cell. Therefore, higher ASW_{cell} means better biological information preservation and lower ASW_{batch} means better batch effect correction. To make them consistent, for ASW_{cell} , we take the normalization, that is:

$$ASW_{cell} = \frac{ASW_{cell}^{raw} + 1}{2}.$$

Similarly, for ASW_{batch} , we take the inverse value of the normalization, that is:

$$ASW_{batch} = 1 - \frac{ASW_{batch}^{raw} + 1}{2}.$$

Both of metrics are in (0,1), and a higher score means better model performance. For multi-omics data integration, all the metrics are the same.

4. Graph Connectivity (GC): GC means the connectivity of cells in different cell types. If the batch effect is substantially removed, the connectivity of cells of the same cell type from different batches will have a higher connectivity score based on the k-NN neighbor graph. Therefore, we can calculate the GC score for each cell type and take the average. GC score is in (0,1) and higher means better batch effect correction.

5. Principal Component Regression (PCR): PCR is a metric to evaluate the performance of batch effect correction. We calculate the R^2 for a linear regression of the covariate of interest onto each principal component. The variance contribution of the batch effect for all the PCs is based on the sum of the product between the variance of each PC and the R^2 of each PC across all the PCs. Therefore, the score can be represented as:

$$PCR = \sum_{i=1}^G \text{Var}(C \mid PC_i) \times R^2(PC_i \mid B).$$

where G denotes the number of PCs and B denotes the batch information. PCR is in (0,1) and a higher score means better performance.

These metrics are also used for Multi-omics Data Integration. For simulation analysis, we only evaluated the biological information preservation for the simulation output with batch effect setting.

C.2 Cell-type Annotation and Gene Function Prediction

1. Accuracy: We calculate the accuracy based on the ratio between the total number of cells (or genes) being classified correctly and the total number of cells (or genes).

2. Precision, Recall, and F1: For classification results, we calculate true positives (tp), false positives (fp), true negative (tn), and false negatives (fn). Precision, Recall, and F1 score are calculated as follows:

$$\begin{aligned} Precision &= \frac{tp}{tp + fp}, \\ Recall &= \frac{tp}{tp + fn}, \\ F1 &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \end{aligned}$$

Note that precision, recall, and F1 are calculated based on the weighted average, where the weight of each cell type is determined by the cell number.

For the Gene Function Prediction task, all of the metrics are the same but for genes rather than cells.

C.3 Perturbation Prediction

We use the Mean Pearson Correlation (MPC) in the validation dataset to evaluate the performance of scGPT across different datasets.

The Pearson correlation for the predicted gene expression g'_i of cell i and the ground truth gene expression g_i of cell i is:

$$S_{PCi} = \frac{g'_i \cdot g_i}{||g'_i|| \cdot ||g_i||},$$

$$S_{MPC} = \frac{\sum_i^n S_{PCi}}{n}.$$

Higher S_{MPC} means better performance.

C.4 Gene Network Analysis

Here we extract the genes whose names start with CD from the Human Immunology pathway. Moreover, we extract genes that have significant correlations from our gene embedding dataset. We then measure the Jaccard similarity for these two gene sets G_{path} and G_{net} as follows,

$$Jaccard = \frac{|G_{path} \cap G_{net}|}{|G_{path} \cup G_{net}|},$$

where G_{path} represents the genes contained in the prior known pathway, and G_{net} represents the genes contained in the GCN generated by scGPT. The Jaccard similarity measure the similarity between two gene sets to evaluate the quality of GCN inference proposed by scGPT.

C.5 Imputation

1. Average bio: The average bio score is calculated based on cell-type NMI and ARI scores using the latent space.
2. Correlation: The correlation score is calculated based on the average correlation between the raw data and imputed data for known common genes. The justification of this metric is based on the assumption that the target of spatial imputation is to predict the expression of missing or unmeasured genes while keeping the distribution of known gene expression levels. We use Pearson correlation here.
3. Significance proportion: The significance proportion corresponds to the proportion of genes with their p-value of correlation smaller than a specific threshold, which is 0.005 in our evaluation. The proportion can be used as a metric to evaluate the similarity between two paired data.

D Evaluation Methods

Here we describe the details of the implementation of the methods discussed in this manuscript.

Single-cell LLMs include:

- tGPT: tGPT is a single-cell LLM based on the GPT-2 [100] structure. It utilized large-scale scRNA-seq datasets for pre-training and set the pre-training task as

predicting gene expression rankings. The downstream applications of tGPT follow the zero-shot learning framework and include clustering, batch effect correction, and bulk RNA-seq analysis.

- **scBERT**: scBERT is a pre-training-based single-cell LLM focusing on cell-type prediction. It is based on Performer [101] with gene embeddings initialized by Gene2vec [102]. It has six self-attention blocks. The default fine-tuning process of scBERT on downstream datasets is freezing the penultimate layer. scBERT is considered for the Cell-type Annotation task.
- **Geneformer**: Geneformer is a single-cell LLM using transfer learning to predict cell types and gene functions. The Geneformer tokenization step is done based on ranking gene expression values in single cells following scaling across the whole training dataset. Cells are represented as token strings with genes rankings as tokens. Geneformer is used for the Cell-type Annotation task and the Gene Function Prediction task. Both tasks were performed by fine-tuning on the basis of the published pre-trained Geneformer network. The default hyper-parameters were used for Geneformer fine-tuning. Prior to tokenization, gene names in all datasets were converted to ENSEMBL IDs using python packages *mygene* [103] and *pyensembl* [104], and most of unmatched genes are non-coding RNAs. For consistency with other benchmarked LLMs, Geneformer’s workflow in Gene Function Prediction and non-cross Cell-type Annotation was altered to split all labeled gene into the training and the testing datasets using the same manners mentioned in our Methods section, instead of using the cross-validation evaluation method provided by the authors.
- **CellLM**: CellLM is a single-cell LLM using three different pre-training strategies. The pre-training loss function includes: 1. masked gene expression level reconstruction; 2. cell condition discrimination; and 3. self-supervised contrastive learning. Moreover, it incorporated protein-protein interaction networks as prior information during the pre-training process. The downstream tasks of CellLM are all related to Cell-type Annotation. Here we modified CellLM so that it can perform cross-dataset cell-type annotation, by separating the training and the testing datasets.
- **scFoundation**: scFoundation employs a pre-training methodology similar to BERT and introduces Bayesian down-sampling as a data pre-processing step. The input data of scFoundation also contain target total counts and input total counts as extra information. The downstream tasks of scFoundation include clustering (a function of cell embeddings across all models), drug response prediction (belong to Cell-type Annotation) and Perturbation Prediction. We did not evaluate the performance of scFoundation because it was closed-source.
- **SCimilarity**: SCimilarity declares that it serves as a foundation model for new data querying or searching based on the cell embeddings generated from known large-scale scRNA-seq datasets. The downstream tasks of SCimilarity include Batch Effect Correction and Cell-type Annotation. We did not evaluate the performance of SCimilarity because it was closed-source.

Task-specific methods are SOTA models in their areas based on their description. They include:

- ResPAN: ResPAN is a batch effect correction tool based on Generative Adversarial Network (GAN) [105, 106]. The high-level idea of ResPAN is based on the distribution alignment or domain adaption across data from different batches. Such requirement can be treated as optimal transport, which can be accomplished by training a GAN. ResPAN is used for the batch effect correction task and the Multi-omics Data Integration task.
- scVI: scVI is a batch effect correction tool based on variational inference and variational auto-encoder [107]. scVI encodes the gene expression data with batch information using a neural network and set the output of the network as parameters for a distribution. Based on such distribution of the latent space, scVI can correct the batch effect in the latent space as well as the original space, as long as we consider the output of the decoder model.
- *Vanilla* NNs: This neural network contains three MLP layers with batch normalization and uses Mish [108] as the activation function. *Vanilla* NNs are used for the Cell-type Annotation task and the Gene Function Prediction task.
- TOSICA: TOSICA is a deep learning-based method for one-stop cell type annotation. TOSICA is designed with the self-attention multi-heads transformer without pre-training. It also provides interpretation for researchers about the attention embeddings and uses attention embeddings to perform biological analysis.
- GEARS: GEARS is a tool for single and multi-gene perturbation prediction based on single-cell RNA sequencing datasets. It combines gene-gene interaction network as prior information and uses a cross-gene neural network with a graph neural network to predict gene expression after perturbation.
- Tangram: Tangram is a toolbox for spatial transcriptomic data analysis based on neural networks. The key idea behind Tangram is using neural networks to find a good mapping function from single-cell data space to spatial data space. After the mapping process, by integrating the information from the single-cell level and spatial level, it can perform several downstream tasks, including data imputation, cell-type deconvolution and others.
- scDesign3: scDesign3 is a model based on Copula distribution [109] to generate different single-cell datasets. Such datasets can be multimodal. Moreover, based on the input parameters and requirements of scDesign3, it can also generate datasets with specific conditions, including batch effect, cell conditions, and the stages of cell differentiation. The data generation of scDesign3 is based on real datasets.

E Model Architecture

The model architecture of scGPT, scBERT, CellLM, Geneformer and tGPT are provided in the Supplementary file 4.

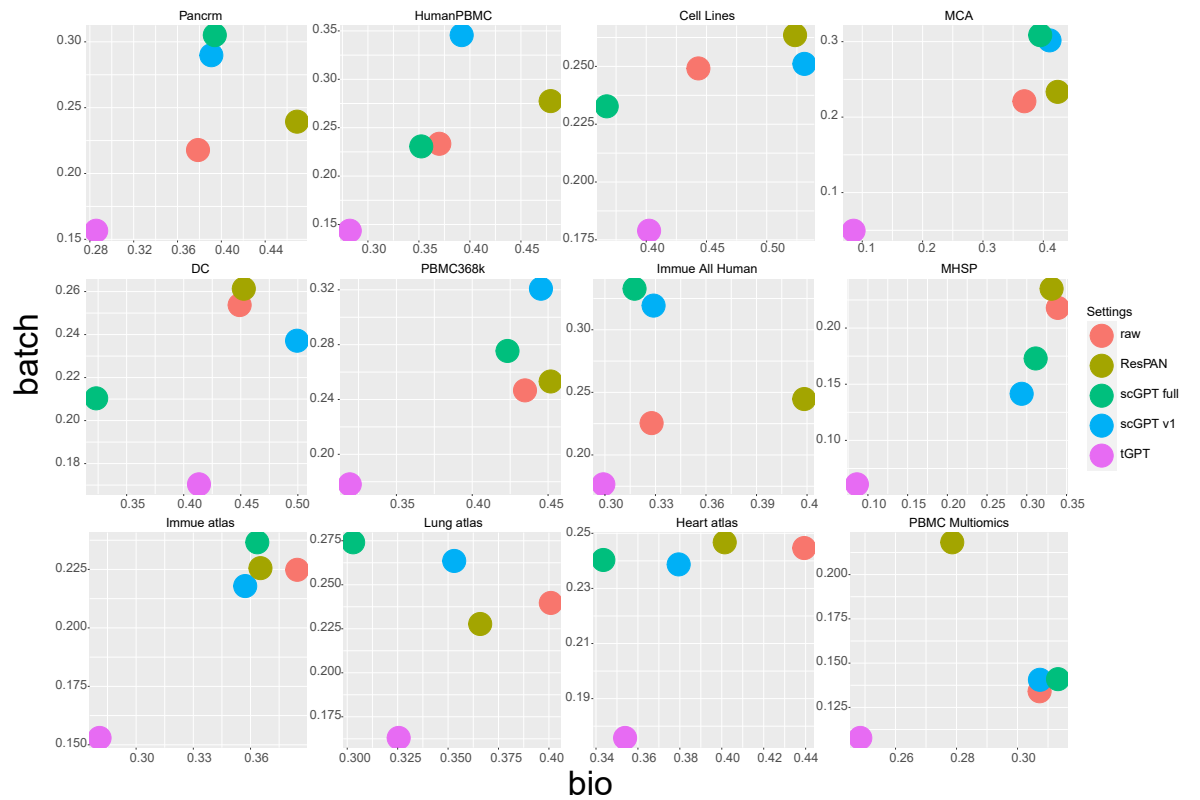
F Data availability

No new sequencing data were generated for this current study. Supplementary file 5 provides the sources and download links for the datasets used in each task. These datasets come from the following papers:

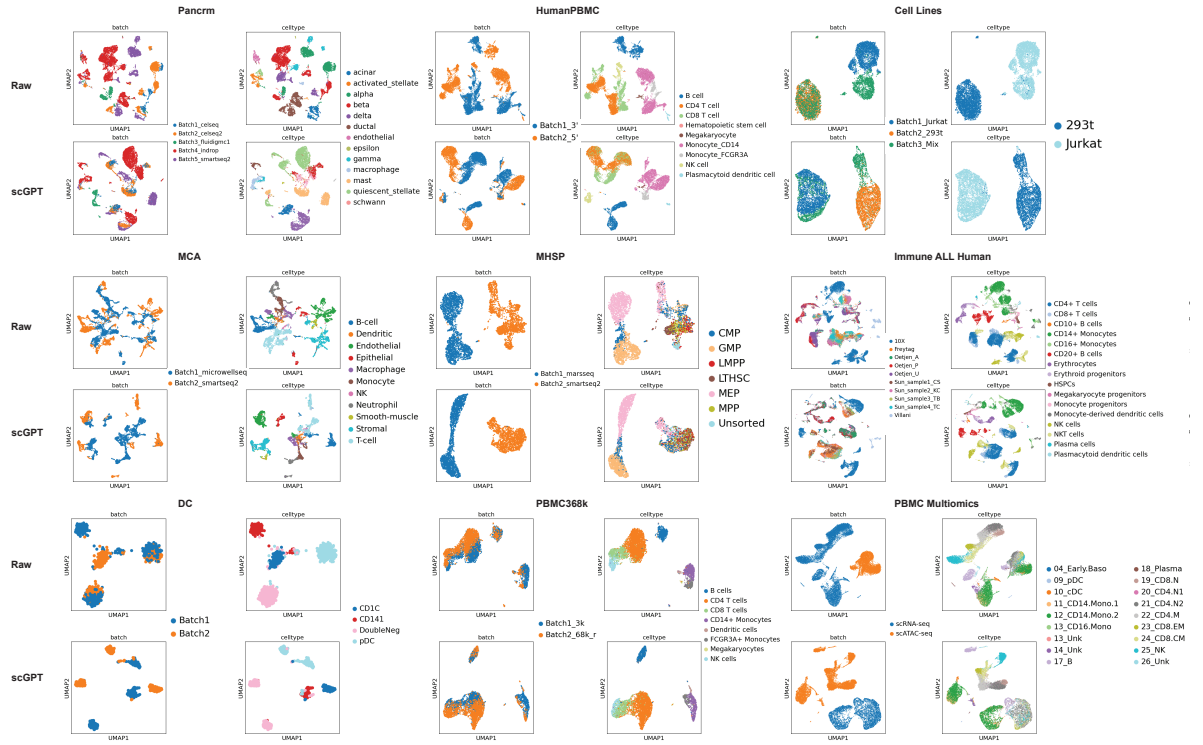
Batch Effect Correction: [35, 46, 110–113].

Cell-type Annotation: [23, 35, 46, 110, 114, 115].
Gene Function Prediction: [116].
Multi-omics Data Integration: [117].
Perturbation Prediction: [42].
Imputation: [55].
Gene Network Analysis: [46, 110].
Simulation Analysis: [110].
Emergent Ability: [110, 114, 118].

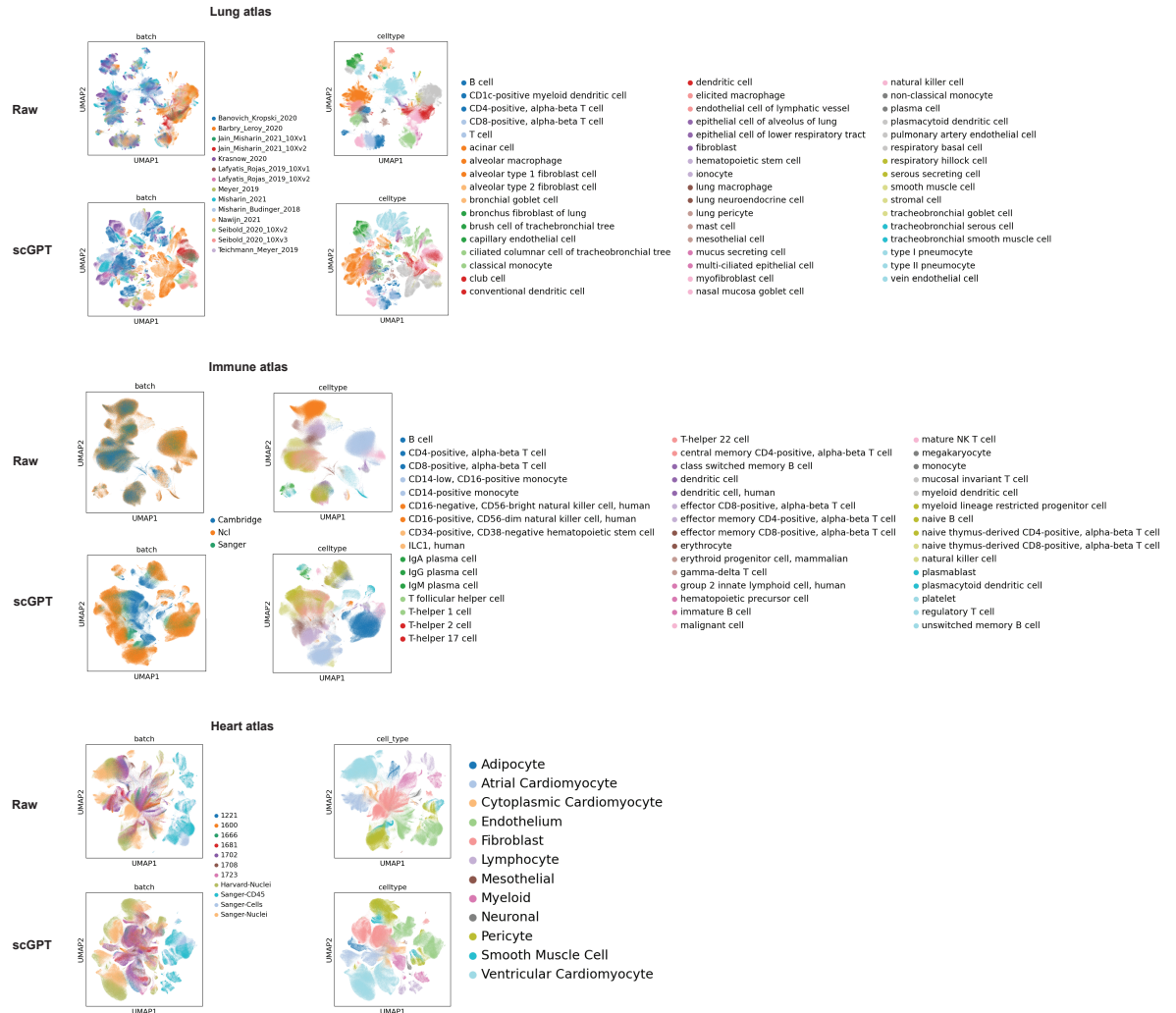
G Supplementary Figures



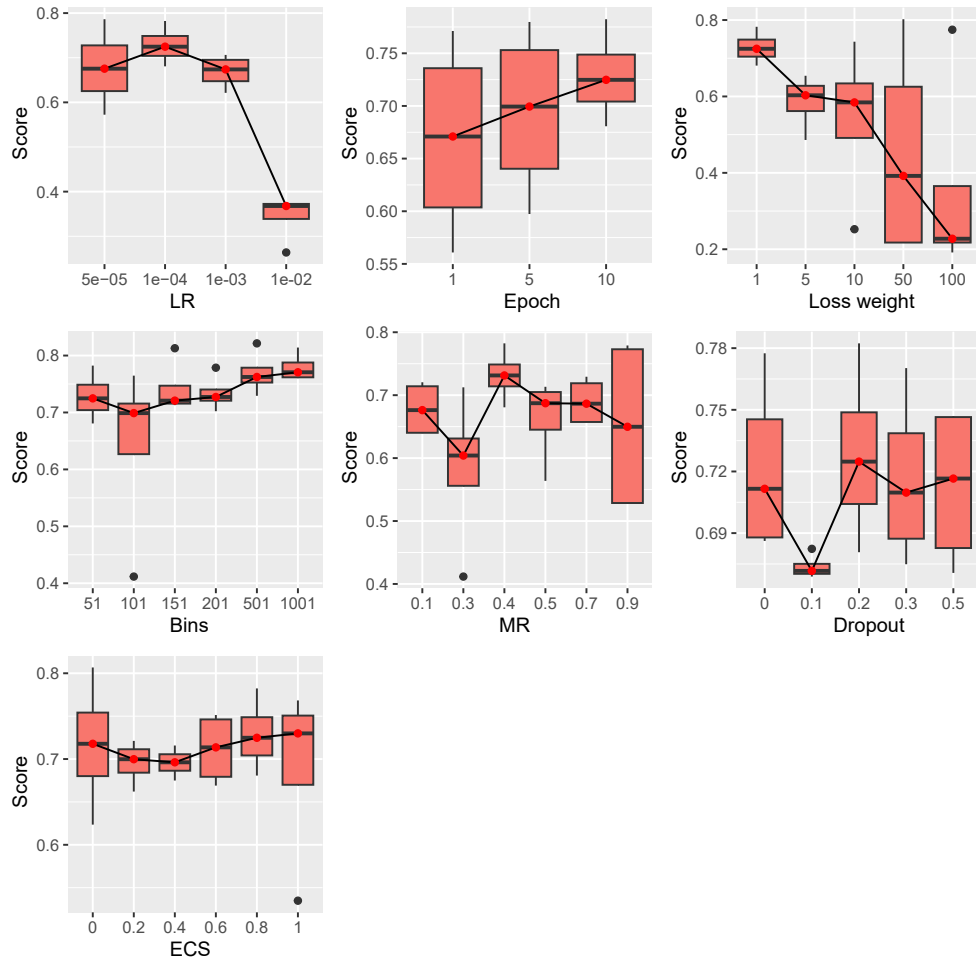
Extended Data Fig. 1: Visualization of batch score and biology conservation score for each dataset.



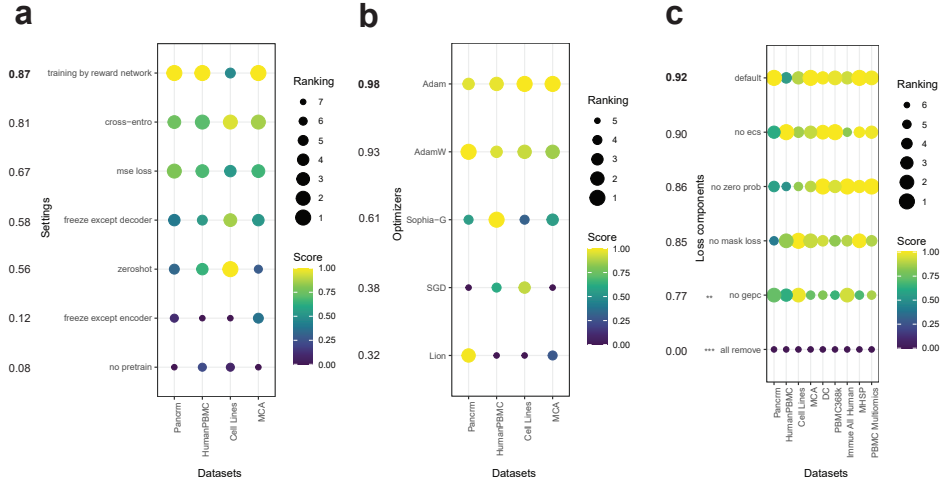
Extended Data Fig. 2: UMAPs for raw data and embeddings of scGPT after batch effect correction.



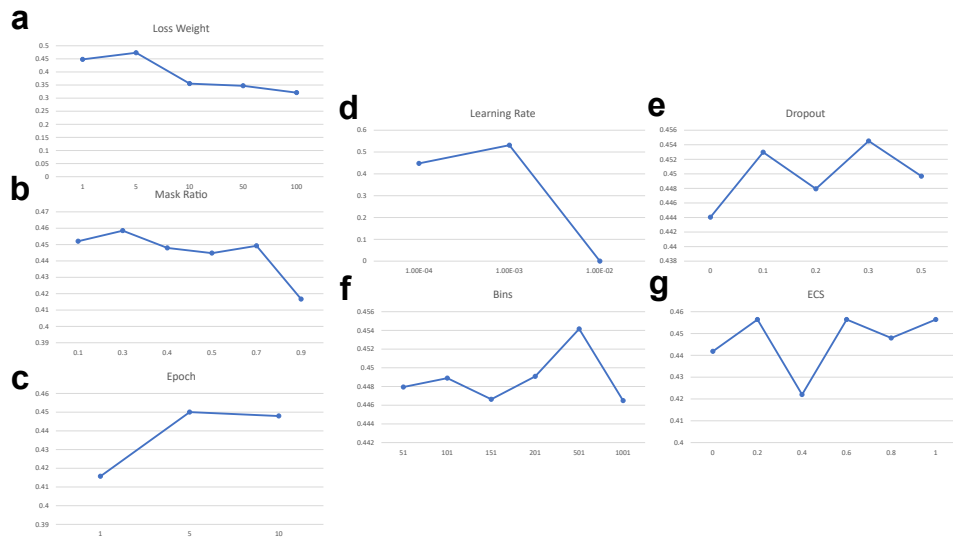
Extended Data Fig. 3: UMAPs for raw data and embeddings of scGPT after batch effect correction.



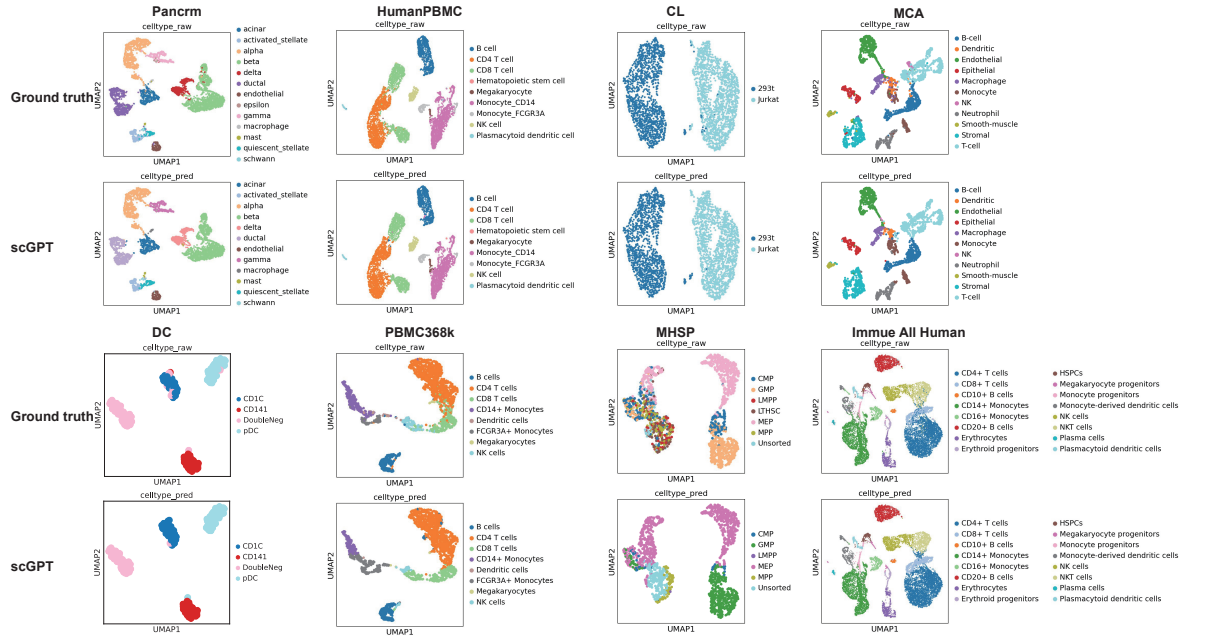
Extended Data Fig. 4: Benchmarking results for different hyper-parameters for Batch Effect Correction.



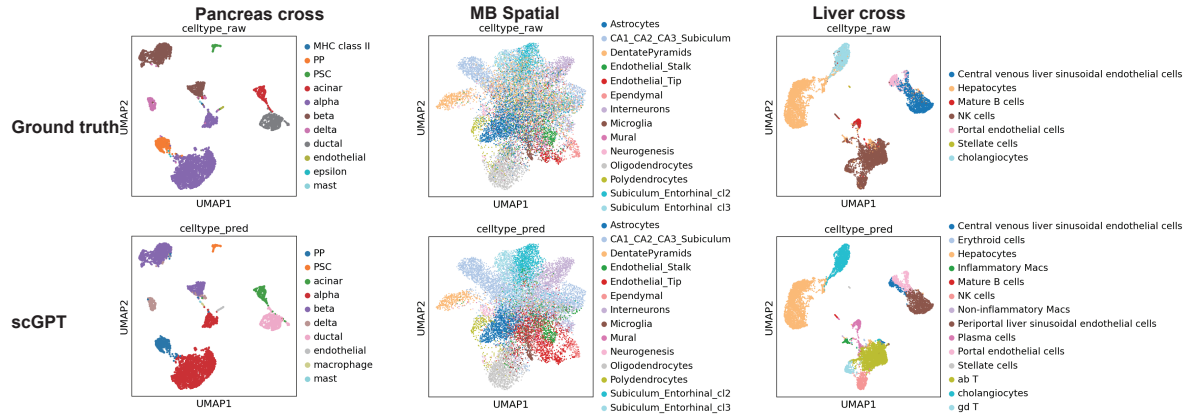
Extended Data Fig. 5: Overall evaluation of different components of scGPT. (a): The final score of different settings across different datasets. (b): The final score of different optimizers across different datasets. (c): The final score of including different loss function terms across different datasets. The number of stars represents the significance level ($*** : p\text{-value} < 0.005$, $** : p\text{-value} < 0.05$, $* : p\text{-value} < 0.1$). The numbers on the left side of each sub-figure represent the average score across different datasets for one condition.



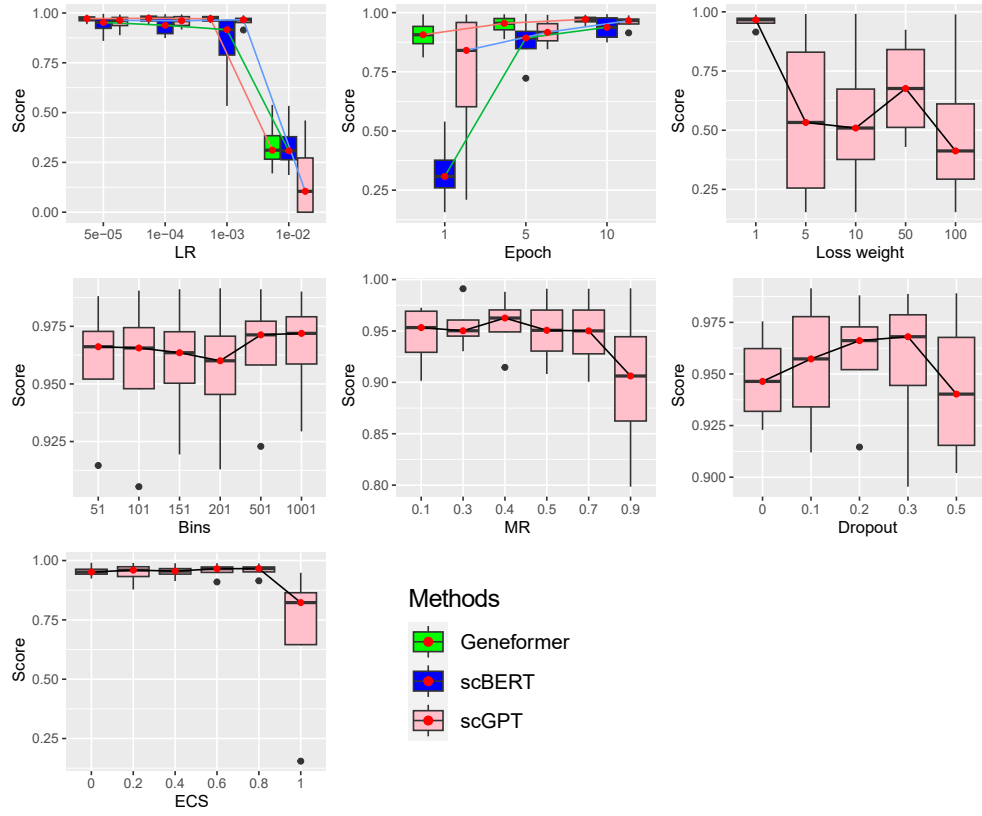
Extended Data Fig. 6: Tuning parameters for multi-omics data integration. (a)-(g) represent the score of scGPT under different hyper-parameters after training.



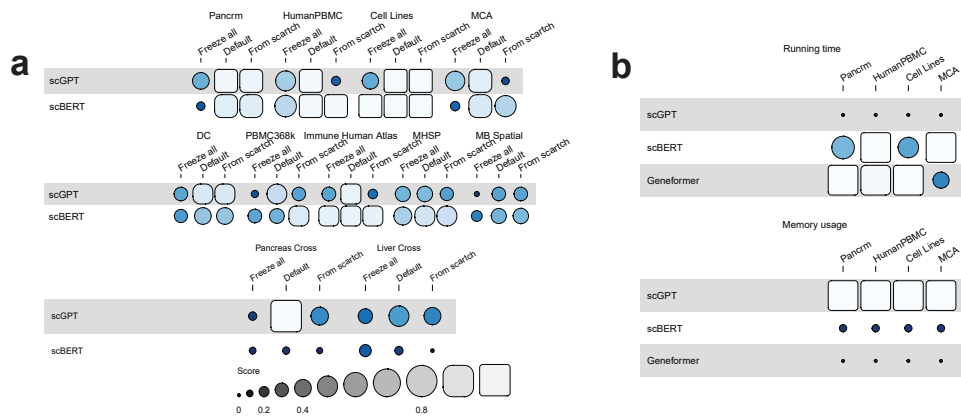
Extended Data Fig. 7: UMAPs for ground truth cell types and prediction results based on scGPT.



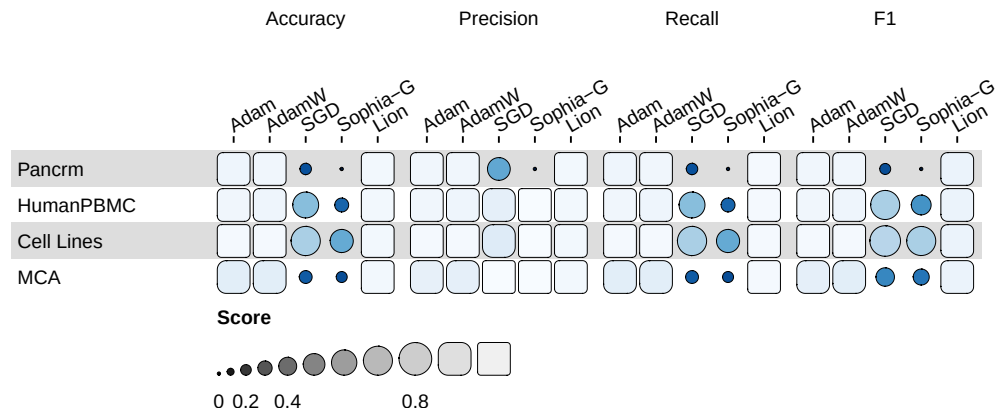
Extended Data Fig. 8: UMAPs for ground truth cell types and prediction results based on scGPT.



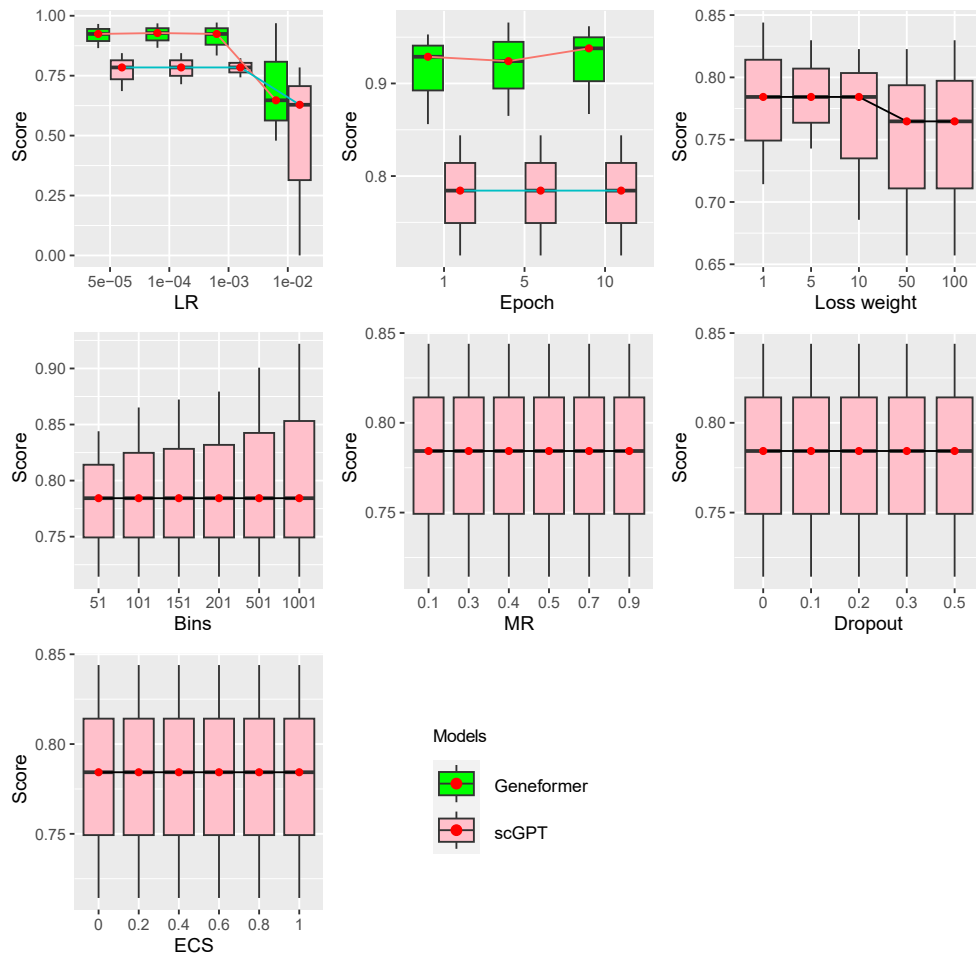
Extended Data Fig. 9: Tuning parameters for cell-type annotation. Sub-figures represent the score of scGPT under different hyper-parameters after training.



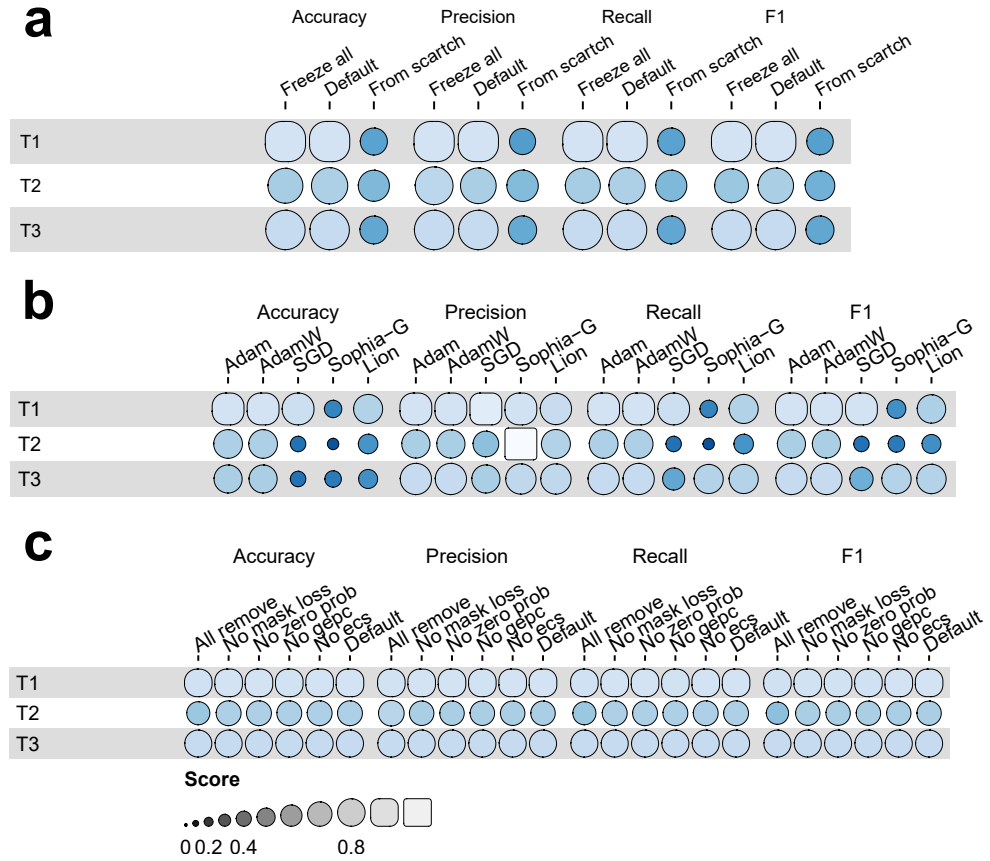
Extended Data Fig. 10: Results of different settings, running time, and memory usage for cell-type annotation task. a: Accuracy of scGPT and scBERT for the Cell-type Annotation task across different datasets. b: Scaled running time (up) and scaled memory usage (down) statistics for all three single-cell LLMs.



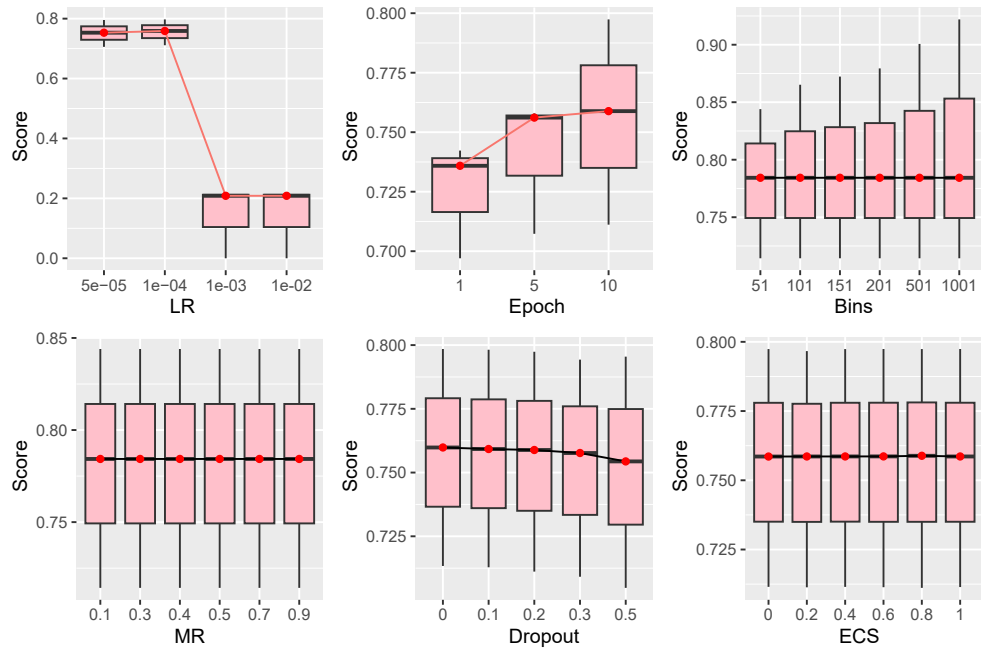
Extended Data Fig. 11: Benchmarking results for different optimizers for cell-type annotation.



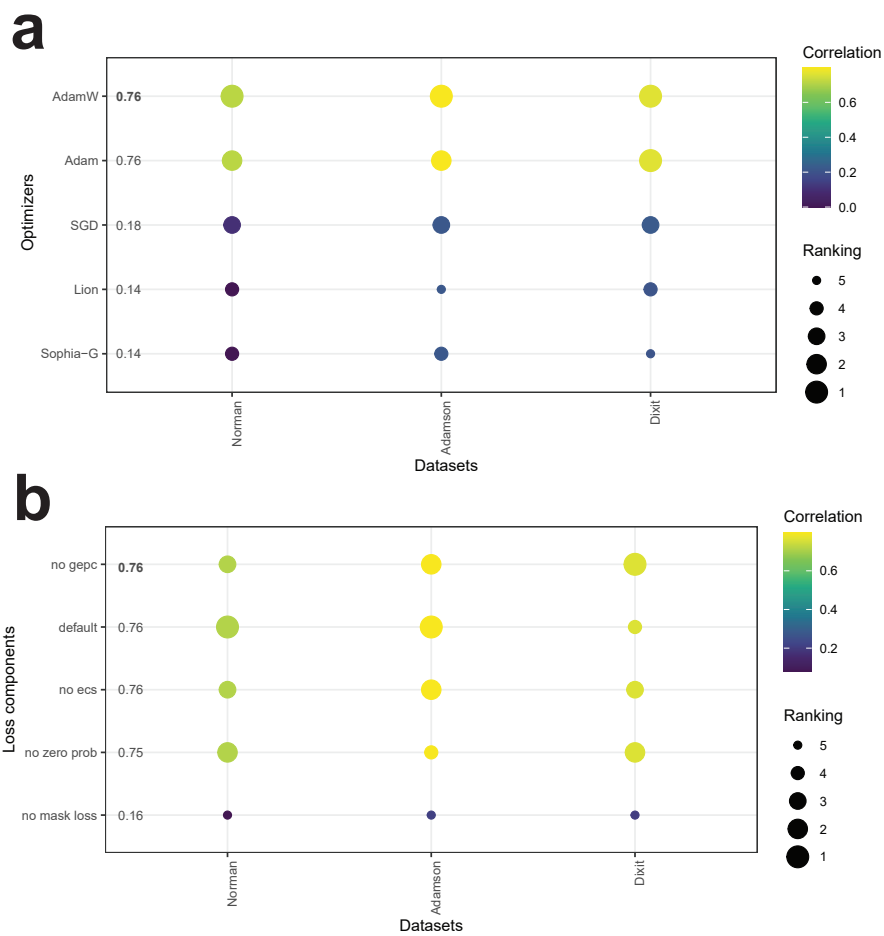
Extended Data Fig. 12: Tuning hyper-parameters for gene function prediction.



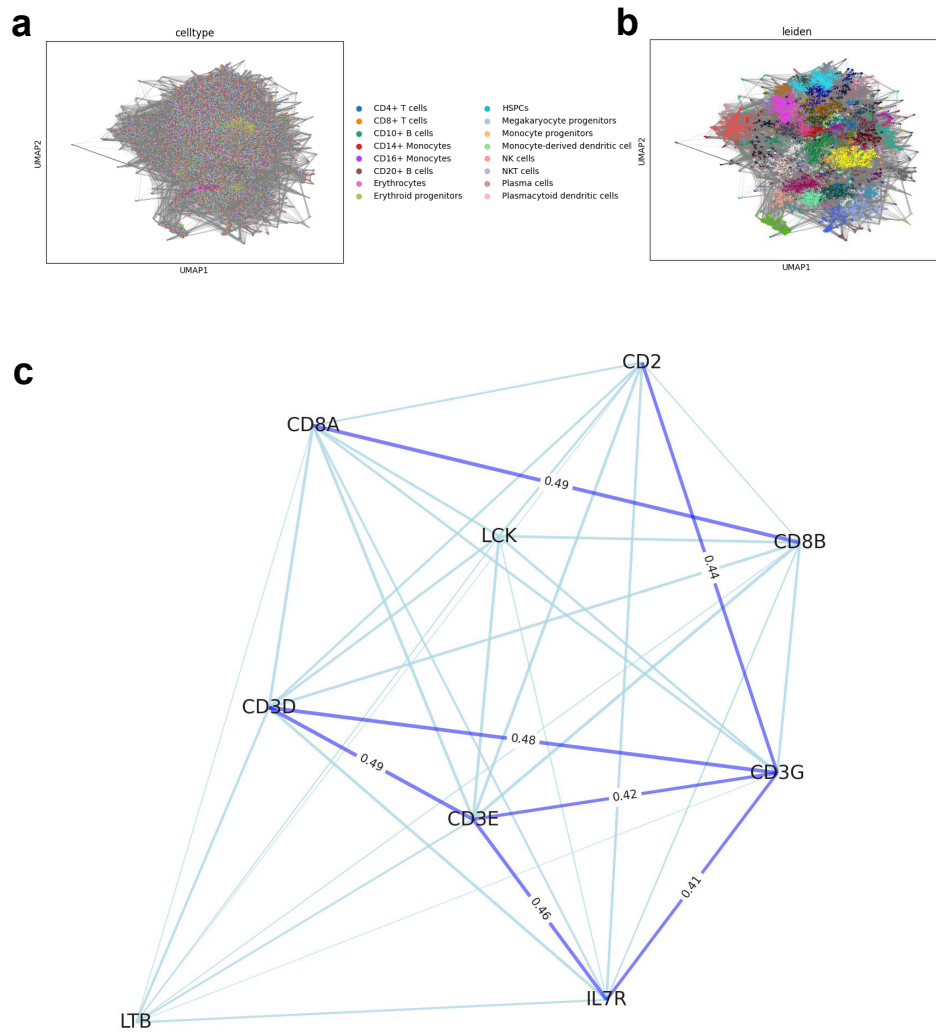
Extended Data Fig. 13: Benchmarking results of different initial settings, optimizers and loss components for gene function prediction. T1-T3 represent different gene prediction cases. (a): Results of adjusting initial settings. (b): Results of adjusting optimizers. (c): Ablation tests based on different loss components.



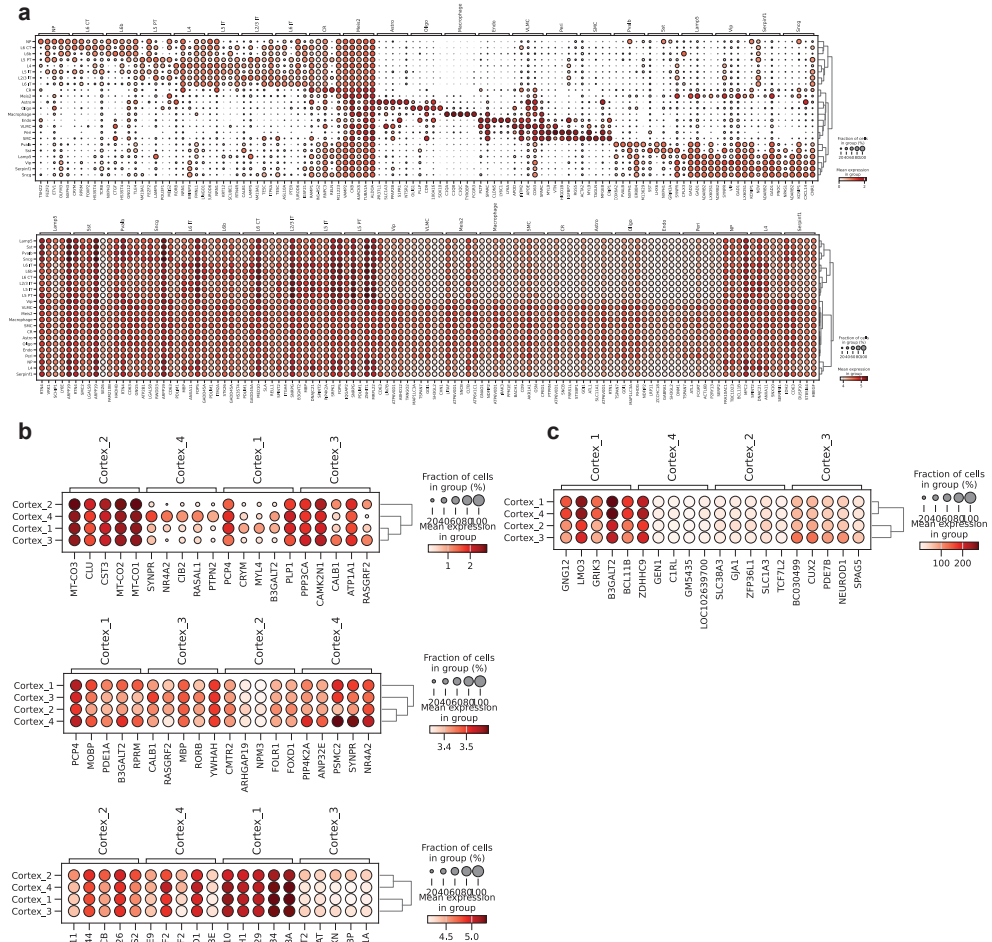
Extended Data Fig. 14: Tuning hyper-parameters for perturbation prediction.



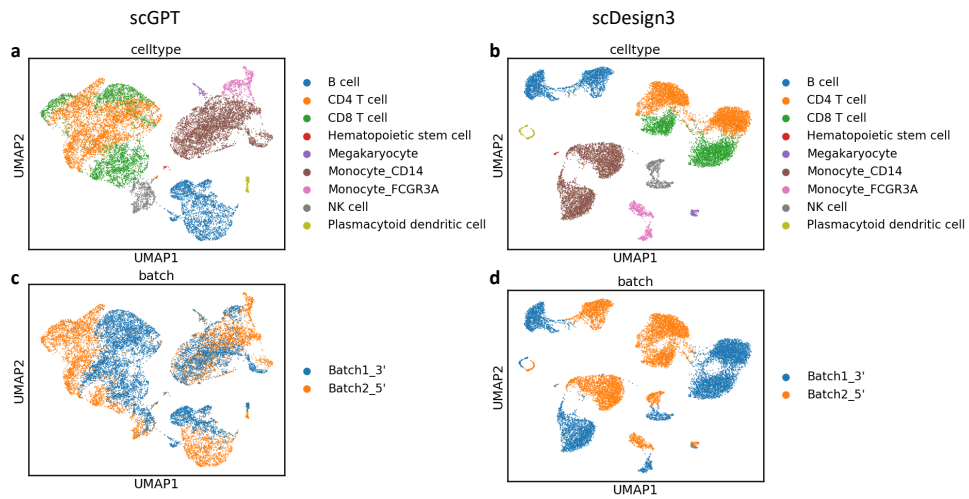
Extended Data Fig. 15: Benchmarking results of different optimizers and loss components for perturbation prediction. (a): Results of adjusting optimizers. (b): Ablation tests based on different loss components.



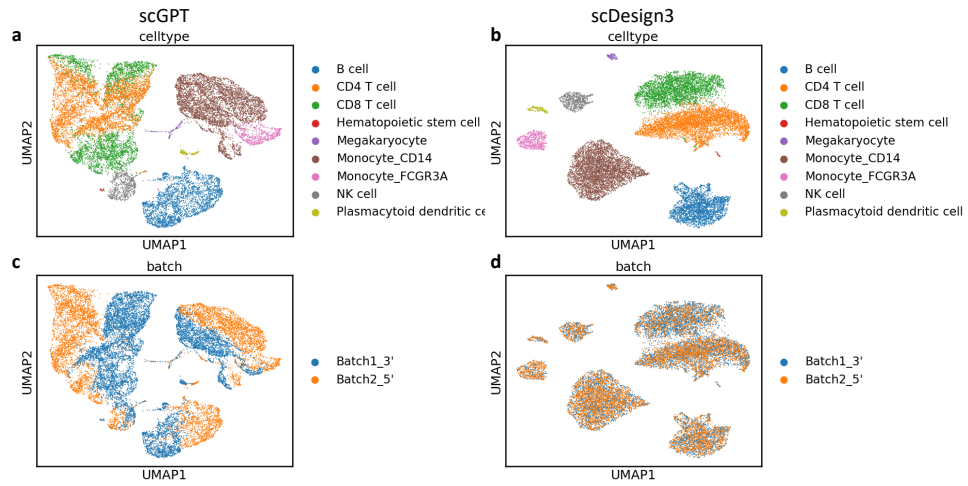
Extended Data Fig. 16: Examples of GCN for the Immune Human Atlas dataset. (a): An example of GCN for human immunology dataset. It is a network with CD-genes as major nodes. (b): Cell-type-level gene embeddings colored by the cell types. (c): Leiden cluster results based on the cell-type-level gene embeddings.



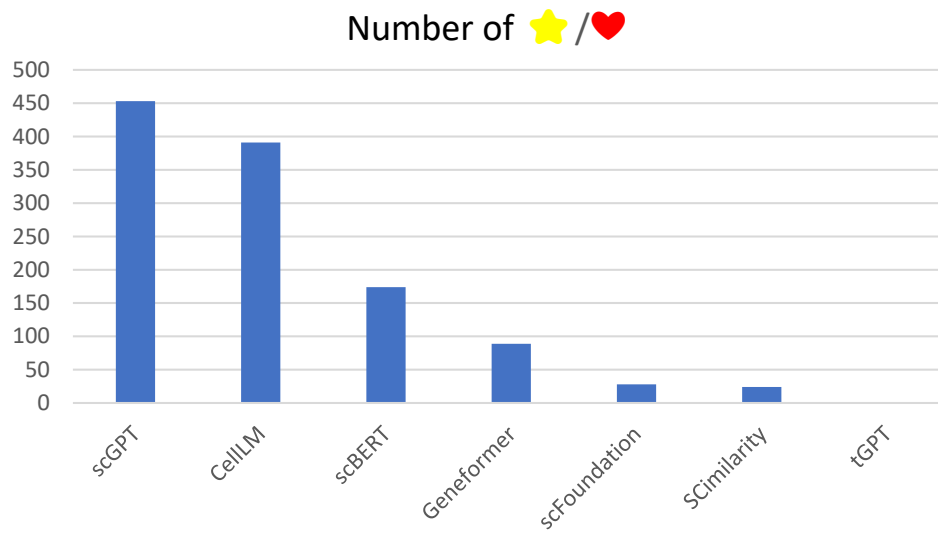
Extended Data Fig. 17: Differentially expressed genes discovery based on results before imputation and after imputation. We used the Mouse scRNA-seq and the Mouse spatial transcriptomic datasets as examples. (a): Differentially expressed genes by cell types for scRNA-seq data based on pre-imputation data (top) and post-imputation data (bottom). (b): Differentially expressed genes by cluster types for spatial transcriptomic data based on pre-imputation data (top), post-imputation data based on zero-shot learning (middle) and post-imputation data based on fine-tuning (bottom). (c): Differentially expressed genes by cluster types for spatial transcriptomic data based on Tangram.



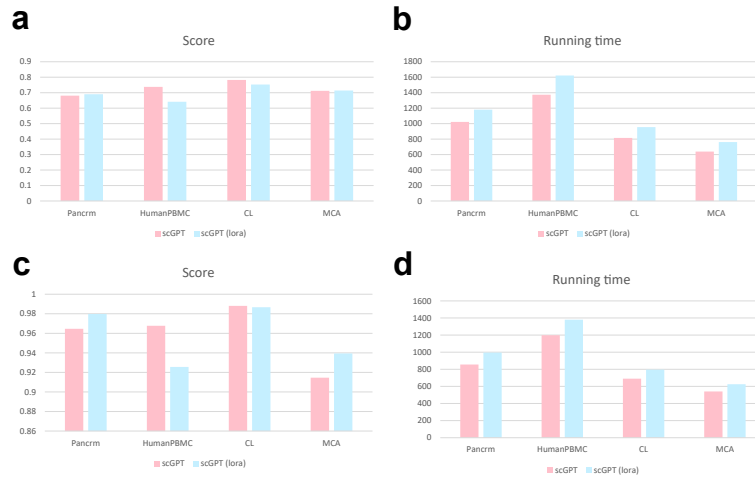
Extended Data Fig. 18: UMAPs for the simulation results with batch effect, using HumanPBMC dataset as an example.



Extended Data Fig. 19: UMAPs for the simulation results without batch effect, using HumanPBMC dataset as an example.



Extended Data Fig. 20: The number of stars in Github or likes in Huggingface for different single-cell LLMs.



Extended Data Fig. 21: Preliminary results of parameter-efficient fine-tuning based on LoRA for Batch Effect Correction (the upper two figures) and Cell-type Annotation (the bottom two figures). (a): The comparison of scores for scGPT with/without LoRA. (b): The running time for scGPT with/without LoRA. (c): The comparison of scores for scGPT with/without LoRA. (d): The running time for scGPT with/without LoRA.