ADVANCED SCIENCE Open Access
www.advancedscience.com

Deep Batch Integration and Denoise of Single-Cell RNA-Seq Data

Lu Qin, Guangya Zhang, Shaoqiang Zhang, and Yong Chen**

Numerous single-cell transcriptomic datasets from identical tissues or cell lines are generated from different laboratories or single-cell RNA sequencing (scRNA-seq) protocols. The denoising of these datasets to eliminate batch effects is crucial for data integration, ensuring accurate interpretation and comprehensive analysis of biological questions. Although many scRNA-seq data integration methods exist, most are inefficient and/or not conducive to downstream analysis. Here, DeepBID, a novel deep learning-based method for batch effect correction, non-linear dimensionality reduction, embedding, and cell clustering concurrently, is introduced. DeepBID utilizes a negative binomial-based autoencoder with dual Kullback–Leibler divergence loss functions, aligning cell points from different batches within a consistent low-dimensional latent space and progressively mitigating batch effects through iterative clustering. Extensive validation on multiple-batch scRNA-seq datasets demonstrates that DeepBID surpasses existing tools in removing batch effects and achieving superior clustering accuracy. When integrating multiple scRNA-seq datasets from patients with Alzheimer's disease, DeepBID significantly improves cell clustering, effectively annotating unidentified cells, and detecting cell-specific differentially expressed genes.

1. Introduction

Single-cell sequencing has enabled the study of cell heterogeneity and molecular mechanisms in individual cells. In recent years, due to the widespread application of single-cell RNA sequencing (scRNA-seq), a vast amount of scRNA-seq datasets of the same cell lines or tissues have been generated under different experimental conditions or laboratories, especially in several important cell atlas projects.[1–4] However, comprehensive analysis of these scRNA-seq datasets is challenging and limits the discovery of biological insights due to the inevitable nonlinear batch effects.[5] Batch effects in scRNA-seq datasets refer to systematic differences in gene expression measurements that arise from technical variations introduced during sample processing, library preparation, and sequencing. These variations can occur between different experimental batches, such as samples processed on different days or using different protocols, and can confound the biological signal in the data. Batch effects can significantly impact downstream analysis, leading to spurious associations, false positives, and decreased reproducibility. Consequently, developing efficient methods to denoise data, eliminate batch effects, and integrate multiple datasets has emerged as a priority in the comprehensive analysis of scRNA-seq data.[6]

Many methods for removing batch effects have been developed, and their performance has been compared using common benchmarking data.[5,7] Integration methods generally operate under the assumption that shared cell types exist across different batches, or at least, some cells of the same type are present across batches. Three technical strategies are commonly used for batch effect removal: *k*-nearest neighbors (KNN), mutual nearest neighbors (MNN),[8] and canonical correlation analysis (CCA).[9] For example, Seurat4[10] employs diagonalized CCA with L_2 normalization to jointly reduce the dimensionality of datasets from two different batches, and then it searches for MNNs in the shared low-dimensional subspace across datasets. BBKNN[11] applies principal component analysis (PCA) for dimensionality reduction and identifies KNNs for each cell in each batch independently. Subsequently, it utilizes the UMAP[12] algorithm to transform the neighbor information into the connectivity of the neighbor sets across batches. DeepMNN[13] searches for MNN pairs across different batches in a PCA subspace and a batch correction network with a stack of two residual blocks is constructed based on these MNN pairs. Scanorama[14] applies randomized singular value decomposition (SVD) to cell-by-gene expression matrices to learn a low-dimensional embedding of each cell. Then, it merges cells from different batches with similar embeddings using an approximate MNN strategy paired with a panorama stitching strategy. scDML[15] first employs a graph-based clustering algorithm to

L. Qin, G. Zhang, S. Zhang
College of Computer and Information Engineering
Tianjin Normal University
Tianjin 300387, China
E-mail: zhangshaoqiang@tjnu.edu.cn

Y. Chen
Department of Biological and Biomedical Sciences
Rowan University
NJ 08028, USA
E-mail: chenyong@rowan.edu

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/advs.202308934

DOI: 10.1002/advs.202308934

cluster cells within each batch and then uses KNN and MNN information both within and between batches to evaluate the similarity between cell clusters hierarchically. RPCI (Reference Principal Component Integration)[16] independently decomposes the cell-by-gene expression matrix of each batch dataset using SVD based on a global reference gene eigenvector and projects all cells into a global reference RPCI space. Harmony[17] utilizes PCA to achieve a low-dimensional embedding of all batches and then employs soft *k*-means clustering to group cells of each batch into clusters. After clustering, it repeatedly learns cluster-specific linear correction factors and adjusts cell assignments until they are stabilized. LIGER[18] first employs integrative non-negative matrix factorization (iNMF) to decompose the data into shared factors and dataset-specific factors, through which it identifies cell types that are common across datasets and those unique to specific conditions.

In addition to using dimensionality reduction strategies such as CCA, PCA, and SVD, several deep-learning-based methods, including BERMUDA,[19] scVI,[20] iMAP,[21] and DESC,[22] perform data integration by mapping the cell profiles from each batch to a low-dimensional latent space within a neural network. BERMUDA[19] first clusters cells from each batch and then removes batch effects using an autoencoder with a transfer loss that is determined by estimating the difference in distributions between pairs of cell clusters. scVI[20] operates as a deep generative model, utilizing deep neural networks and stochastic optimization to obtain a probabilistic representation of scRNA-seq data with minimized batch effects. iMAP[21] builds a deep model with an encoder designed to extract low-dimensional, batch-agnostic representations of each cell profile and two generators for profile reconstruction. Subsequently, a generative adversarial network (GAN) structure is trained on extended MNN pairs from two batches to further eliminate batch effects. DESC[22] uses a stacked autoencoder to learn a low-dimensional representation of cell profiles. The resulting encoder is then added to a neural network to cluster cells iteratively with a Kullback–Leibler (KL) divergence loss.

While these computational methods have shown potential in addressing batch effects in scRNA-seq data, there remain several technical limitations hindering their performance. First, MNN-based techniques are primarily devised to integrate only two batches simultaneously. As a result, these methods lack a consistent global reference space for seamlessly integrating multiple datasets. Second, existing deep learning-based methods often overlook the underlying data distribution. Recent statistical analyses of scRNA-seq data, especially from mainstream commercial platforms like 10X genomics, indicate that the gene expression profiles among cells typically follow a negative binomial (NB) distribution.[23–26] Implementing an autoencoder that fits the NB distribution has been shown to enhance data imputation and the accuracy of cell clustering.[27] For example, BERMUDA[19] and SAUCIE[28] employ NB distributions, whereas DESC[22] and scDeepCluster[29] utilize zero-inflated negative binomial (ZINB) distributions to model gene expression counts in clustering analysis of scRNA-seq data. However, many other methods do not explicitly incorporate the distributions of gene expression counts. Third, only a few algorithms, such as DESC,[22] offer simultaneous data integration and cell clustering, which streamlines subsequent data analysis. In contrast, methods like Harmony,[17] BERMUDA,[19] LIGER,[18] and Seurat4[10] treat data integration and clustering as distinct steps in their pipelines. For example, BERMUDA[19] first clusters cells within each batch before merging clusters from different batches. Meanwhile, most of the current methods optimize between either dimensionality reduction and integration (e.g., RPCI, scVI, and iMAP), integration and clustering (e.g., Harmony), or dimensionality reduction and clustering (e.g., DESC). Integrating the removal of batch effects with downstream analysis such as cell clustering can be beneficial in improving performance, facilitating benchmarking, increasing robustness, and offering benefits in computational scalability and efficiency. Hence, there is a significant need for a method that can coordinate optimization across dimensionality reduction, integration, and clustering.[30] To address this, we introduce DeepBID (Deep Batch Integration and Denoise), a method that facilitates cell clustering in a low-dimensional latent space while progressively mitigating batch effects during cell profile reconstruction. Nonlinear dimension reduction, batch effect removal, and clustering are optimized by employing fuzzy *k*-means on an NB-based autoencoder, which incorporates two KL divergence losses into the adaptive loss function of fuzzy *k*-means. Extensive validation tests on multi-batch datasets and real applications to multiple datasets from patient samples with Alzheimer's Disease (AD) demonstrate that DeepBID is highly effective in denoising batch effects, and cell clustering, and consistently outperforms other well-known tools. When applied to multiple scRNA-seq datasets from AD patients, DeepBID significantly improved clustering results, effectively annotating unidentified cells and detecting cell-specific differentially expressed genes (DEGs).

2. Results

2.1. Overview of DeepBID

DeepBID is designed to efficiently denoise batch effects and integrate multiple scRNA-seq datasets by leveraging advanced deep learning strategies (**Figure 1**). Its primary optimization goal is to achieve a close overlap of cells from different batches in the integrated space while maintaining a clear separation between different cell types. It employs an autoencoder to map all cell profiles X from different batches to a common, non-linear, low-dimensional latent feature space H. The encoder component of the autoencoder is responsible for reducing the dimensionality of input data from multiple batches, while the decoder reconstructs the data. To denoise the input data, an NB distribution is integrated by appending two independent fully connected layers to the last reconstruction layer, aiming to estimate its mean and dispersion parameters. To cluster closely related cells effectively, the objective function of fuzzy *k*-means with weighted adaptive loss is embedded within the latent feature space H of the autoencoder.[31] The solution of fuzzy clustering is a soft assignment matrix of cells $P = (p_{jk})_{N\times K}$, where p_{jk} is the probability that cell j belongs to the k-th cluster. To preserve and amplify the association between similar cells, a specific KL divergence between soft assignment and an auxiliary distribution for each cell is used as a loss function, a technique employed by some clustering algorithms such as scziDesk.[32] Moreover, to ensure that cells in the same cluster originate from different batches, a second KL divergence loss is introduced to measure the disorder of cells from

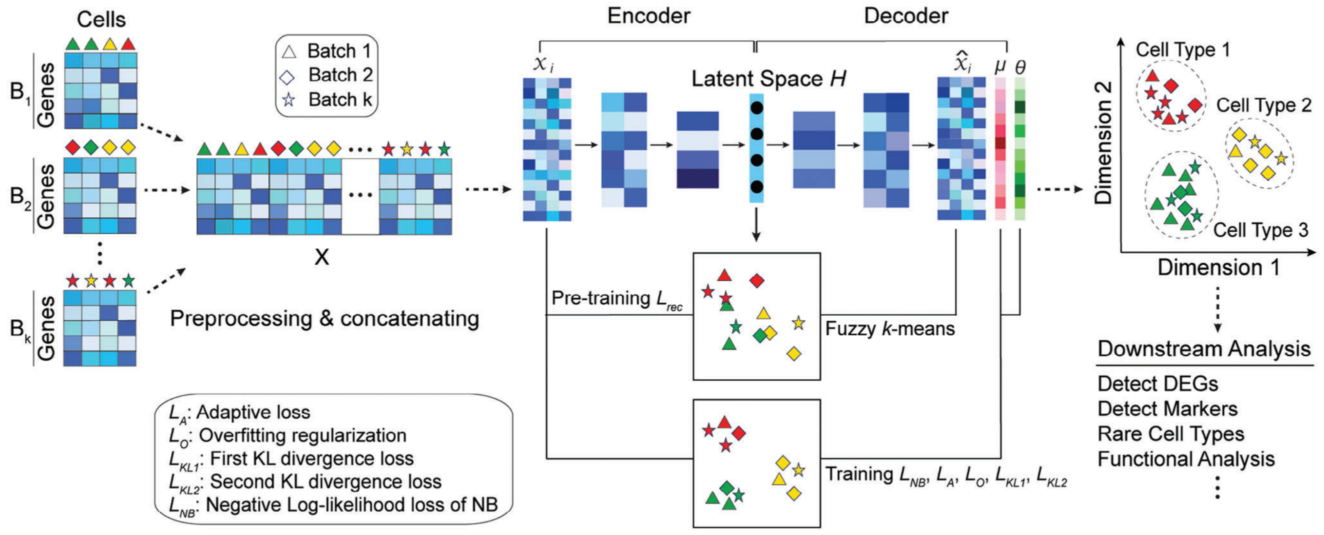

Figure 1. The workflow of the DeepBID method. DeepBID processes scRNA-seq datasets from multiple batches as inputs. It employs an NB-based autoencoder for pretraining to optimize the latent space, enabling iterative integration and clustering. The loss function is composed of multiple sub-functions that cater to nonlinear dimension reduction, batch effect removal, and clustering, while also preventing overfitting. The outputs of DeepBID include an integrated gene expression matrix and clustered cell types, ready for various downstream analyses. The various shapes (such as triangles, diamonds, and stars) represent different batches, while the distinct colors denote different cell types.

various batches within each cluster. Finally, to counteract overfitting in DeepBID, a regularization loss is incorporated into the total loss function.

The DeepBID software is designed for user-friendly operation within a PyTorch environment and offers flexibility by accommodating a variety of user-defined parameters. Its input can consist of multiple scRNA-seq datasets from different batches, whether from the same experiment, the same tissue samples, or even different labs. The software outputs an integrated gene expression matrix and cell type annotation, which can be differently used for downstream analysis (right side, Figure 1). Furthermore, DeepBID features an option to use CPU+GPU hybrid computing for processing extensive datasets. The inputs and outputs of DeepBID can be directly integrated with popular scRNA-seq analysis pipelines like SCANPY.[33]

2.2. Hyperparameter Selections of DeepBID for Diverse Datasets

DeepBID possesses four hyperparameters in its final objective function: λ_1, λ_2, κ_1, and κ_2 (refer to Equation 11 in Experimental Section). We initially assessed the performance of DeepBID with various values for λ_1 and λ_2 while keeping $\kappa_1 = 0.01$ and $\kappa_2 = 100$ fixed. Adjusted Rand index (ARI)[34] and normalized mutual information (NMI)[35] scores were used to evaluate the quality of clustering results. ARI measures the similarity between two sets of cluster assignments while NMI measures the mutual information between the true labels and the clustering results, normalizing to account for differences in cluster sizes. High ARI and NMI scores indicate strong agreement between the true labels and the clustering results. Here, NMI and ARI scores were computed for five datasets: “DC”, “Cell_lines”, “SC_mixology”, “Pancreas”, and “PBMCs”, which were sequenced either by a single sequencing platform or generated through multiple platforms (see Table S1, Supporting Information). All five datasets possess labels for batches and cell types and have been utilized as benchmarks in various integration or clustering tools.[14,17,21,22] As indicated in **Figure 2**, $\lambda_1 = 5$ and $\lambda_2 = 0.01$ yield excellent outcomes across these datasets. Notably, for the two large-scale datasets “Pancreas” and “PBMCs” (with total cell numbers exceeding 10,000), the optimal settings are $\lambda_1 = 1$ and $\lambda_2 = 0.0001$. Consequently, we recommend setting λ_2 to be 0.01 for small-scale datasets and 0.0001 for large-scale datasets to prevent model overfitting.

After determining the values of λ_1 and λ_2, we proceeded to select suitable values for the hyperparameters κ_1 and κ_2, which represent the two KL divergence losses. Fixing λ_1 and λ_2 values ($\lambda_1 = 5$ and $\lambda_2 = 0.01$ for the datasets “DC”, “Cell_lines” and “Sc_mixology”, and $\lambda_1 = 1$ and $\lambda_2 = 0.0001$ for “Pancreas” and “PBMCs”), we executed DeepBID using seven distinct κ_1 values (i.e., 0.001, 0.01, 0.1, 0, 1, 10, or 100). When the 20 epochs were used, the ARI and NMI scores peaked at $\kappa_1 = 0.01$ for four of the five datasets (Figure S1, Supporting Information). However, the “PBMCs” dataset displayed more consistent results under $\kappa_1 = 0.01$ compared to $\kappa_1 = 1$. Therefore, we uniformly adopted $\kappa_1 = 0.01$ for all datasets. Notably, clustering results with $\kappa_1 = 0$ significantly underperformed those with $\kappa_1 = 0.01$, underscoring the importance of proper parameter configurations for the primary KL divergence loss in enhancing clustering accuracy.

The purpose of the secondary KL divergence is to amplify the integration effect. We experimented with a range of κ_2 values from 0.001 to 1,000 while maintaining λ_1, λ_2 and $\kappa_1 = 0.01$. After 20 epochs, the clustering outcomes of the integrated data at $\kappa_2 = 100$ were more resilient compared to other settings for most of the datasets (Figure S2, Supporting Information). Although the “Cell_lines” dataset remained largely unchanged across different κ_2 configurations, the results at $\kappa_2 = 100$ outpaced those at $\kappa_2 = 0$. This indicated that the secondary KL divergence loss played a

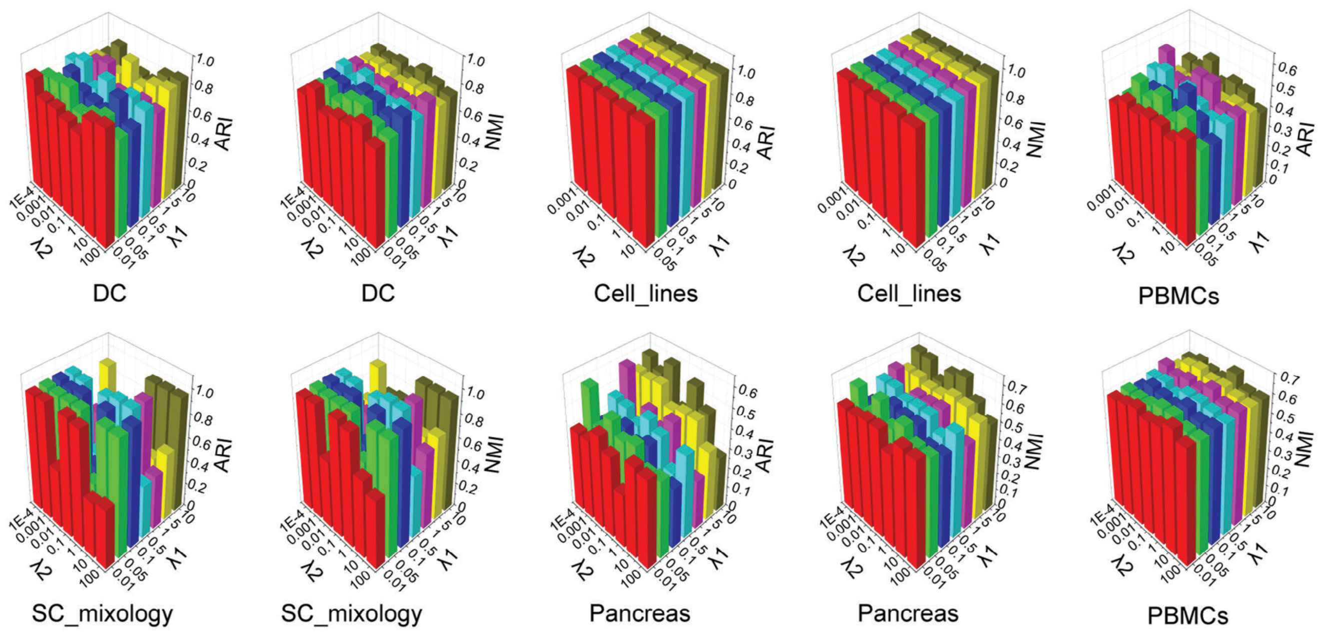

Figure 2. Clustering results of ARI and NMI scores on the five datasets for different λ_1 and λ_2 with fixed $\kappa_1 = 0.01$ and $\kappa_2 = 100$.

crucial role in achieving precise integration and clustering. This was especially evident in large-scale datasets like "Pancreas" and "PBMCs", where the benefits of $\kappa_2 = 100$ were particularly pronounced. Therefore, we set hyperparameters $\kappa_1 = 0.01$, $\kappa_2 = 100$ in the practical applications of DeepBID. Furthermore, we observed that DeepBID could achieve stable NMI scores after 20 epochs (Figures S1 and S2, Supporting Information). Thus, DeepBID sets the default number of epochs to 20 to balance performance and runtimes.

2.3. DeepBID Performs Better Than the Other Five Methods in Batch Integration

We computed LISI (Local Inverse Simpson Index)[17] to evaluate the effect of data integration. LISI has two specific metrics, iLISI (i.e., integration LISI) and cLISI (i.e., cell-type LISI), which measure the number of batches and the number of cell types within a local neighborhood, respectively. We applied six different methods to each dataset to integrate various batches. We first constructed box plots and determined the median iLISI score for every method on each dataset (**Figure** 3a; Table S2, Supporting Information). It is important to note that a higher iLISI score indicates a greater mixing degree of different batches. Results showed that DeepBID boasted the highest median iLISI scores for four datasets: "Cell_lines", "DC", "Pancreas", and "PBMCs". The only exception was the "Sc_mixology" dataset, where DeepBID's median iLISI score was surpassed by LIGER's. However, the clustering performance of DeepBID was markedly superior to that of LIGER. Specifically, we visualized the UMAP plots for both raw and integrated cells on the "Sc_mixology" dataset. The visual representation revealed that DeepBID accurately segregated cells into three clusters consistent with cell types (**Figure** 4a). In contrast, LIGER partitioned them into nine clusters, even though it recorded the highest median iLISI scores. Moreover, the remaining four methods segmented the "Sc_mixology" cells into more than six clusters. Taken together, these results suggested that DeepBID integration aligned more closely with cell types than LIGER.

We further calculated the cLISI score, which indicates the proximity of cells of the same type, with lower scores indicating closer similarity. As shown in Figure 3b and Table S3 (Supporting Information), DeepBID achieved the lowest median cLISI scores on three datasets: "DC", "Sc_mixology", and "Pancreas". Although the median cLISI score of DeepBID was higher than those of Harmony, Seurat, and DESC for the "Cell_lines" and "PBMCs" datasets, its clustering results were superior to the other six methods for these datasets. This could be explained by differences in cluster compaction across methods. For illustration, the UMAP plots for the "Cell_lines" dataset, as produced by the six methods, are showcased in Figure 4b. The cell clusters produced by DeepBID appear to be more scattered relative to Harmony, Seurat, and DESC. DeepBID distinctly partitioned the dataset into two groups, closely reflecting the two cell types: "293T" and "Jurkat". Contrastingly, DESC, Harmony, and Seurat identified more than two domains, thereby introducing false positives. Therefore, solely relying exclusively on LISI scores, without precise clustering, compromises the accurate assessment of batch integration efficacy. DeepBID provided the most effective integration among the six methods for the "Sc_mixology" and "Cell_lines" datasets, even if its LISI scores were suboptimal.

2.4. DeepBID Performs Better Than the Other Five Methods in Cell Clustering

To investigate whether data integration could enhance cell clustering, we applied DeepBID and five other methods to five

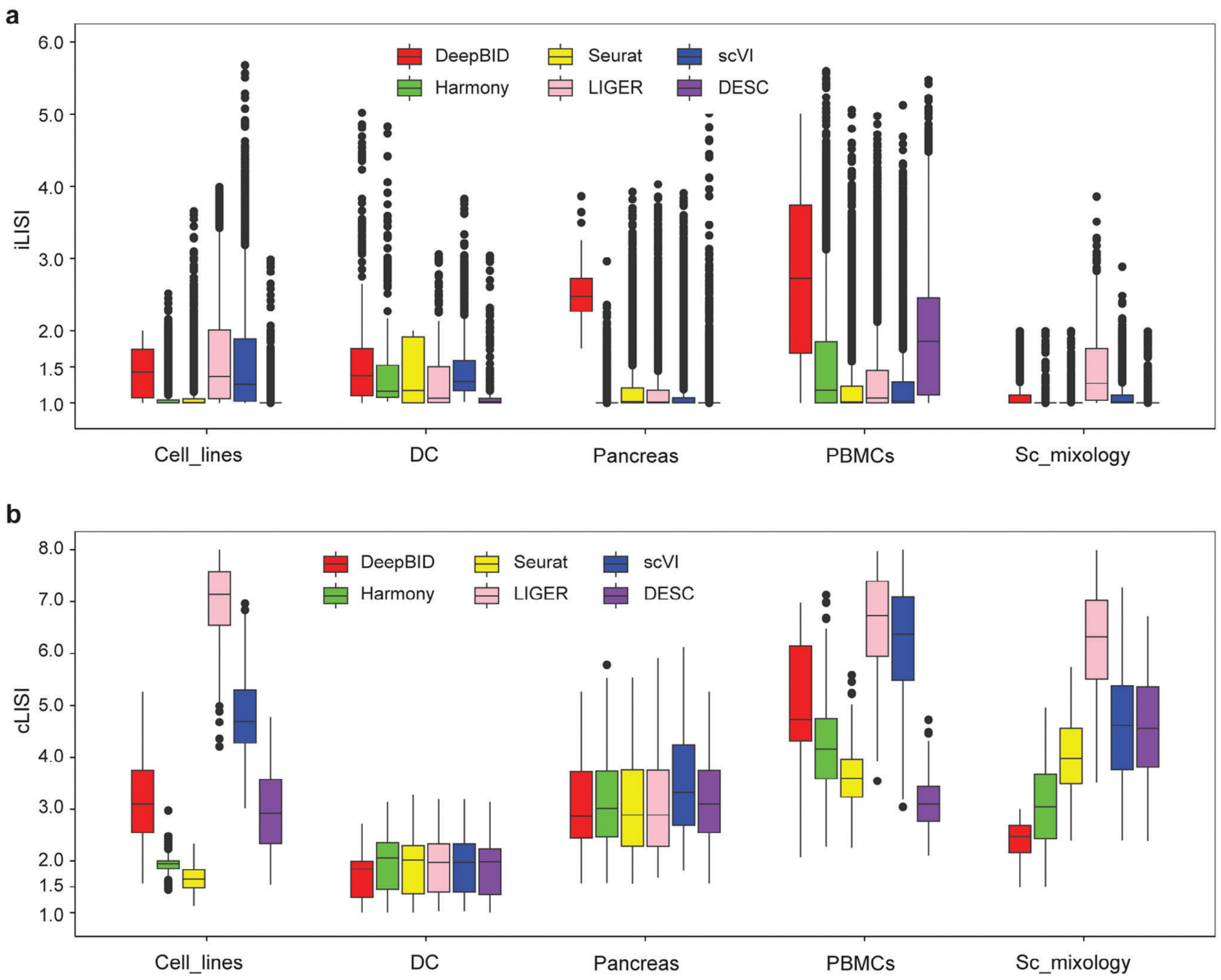

Figure 3. Validation results of data integration of six methods across five datasets. a) Boxplots of the iLISI scores for six methods on five datasets. The outline dots represent the extreme values in each dataset, which are 1.5 times of the interquartile range. b) Boxplots of the cLISI scores for six methods on five datasets.

benchmark datasets (detailed in Experimental Section). We calculated the ARI and NMI scores for all clustering results and discovered that DeepBID consistently outperformed the other methods (**Figure** 5; Table S4, Supporting Information). Specifically, DeepBID demonstrated better clustering performance on the "Sc_mixology" and "Cell_lines" datasets. The ARI score of DeepBID on "Sc_mixology" was 0.99146, which was 1.6 times higher than the second-best score (0.61212) achieved by the Seurat method (Figure 5a). On the "Cell_lines" dataset, the ARI score of DeepBID is 0.99821, which was more than double the second-best score (0.44557) generated by Harmony. Similar patterns were observed with the NMI scores (Figure 5b). On the "Sc_mixology" dataset, DeepBID achieved an NMI score of 0.98498, which was 1.3 times the second-best score (0.74793) generated by DESC. On the "Cell_lines" dataset, the NMI score of DeepBID is 0.97854, 1.6 times the second-best score (0.58549) by the Harmony method. The high performance of DeepBID was also observed on other datasets. For the "DC" dataset, both DeepBID and DESC performed better than the rest, while on the "PBMCs" dataset, DeepBID, scVI, and Harmony emerged as the top three for clustering efficiency. On the "Pancreas" dataset, DeepBID and LIGER achieved the highest ARI scores compared to the other four methods.

As deep learning-based methods typically demand more runtime for model training and prediction compared to conventional statistical approaches, assessing runtimes across datasets of varying sizes is imperative. This becomes especially crucial as scRNA-seq datasets may expand to millions of cells in large research projects.[36,37] To address this, we recorded the runtime performance for six methods on the five datasets with increasing cell numbers: DC (569 cells), Sc_mixology (1,401 cells), Cell_lines (9,531 cells), Pancreas (14,375 cells), and PBMCs (21,526 cells). The runtime measurement captured the core processing of each tool but excluded preprocessing time. The results, summarized in Table S5 (Supporting Information) and plotted in Figure 5c, highlighted DeepBID's relatively low

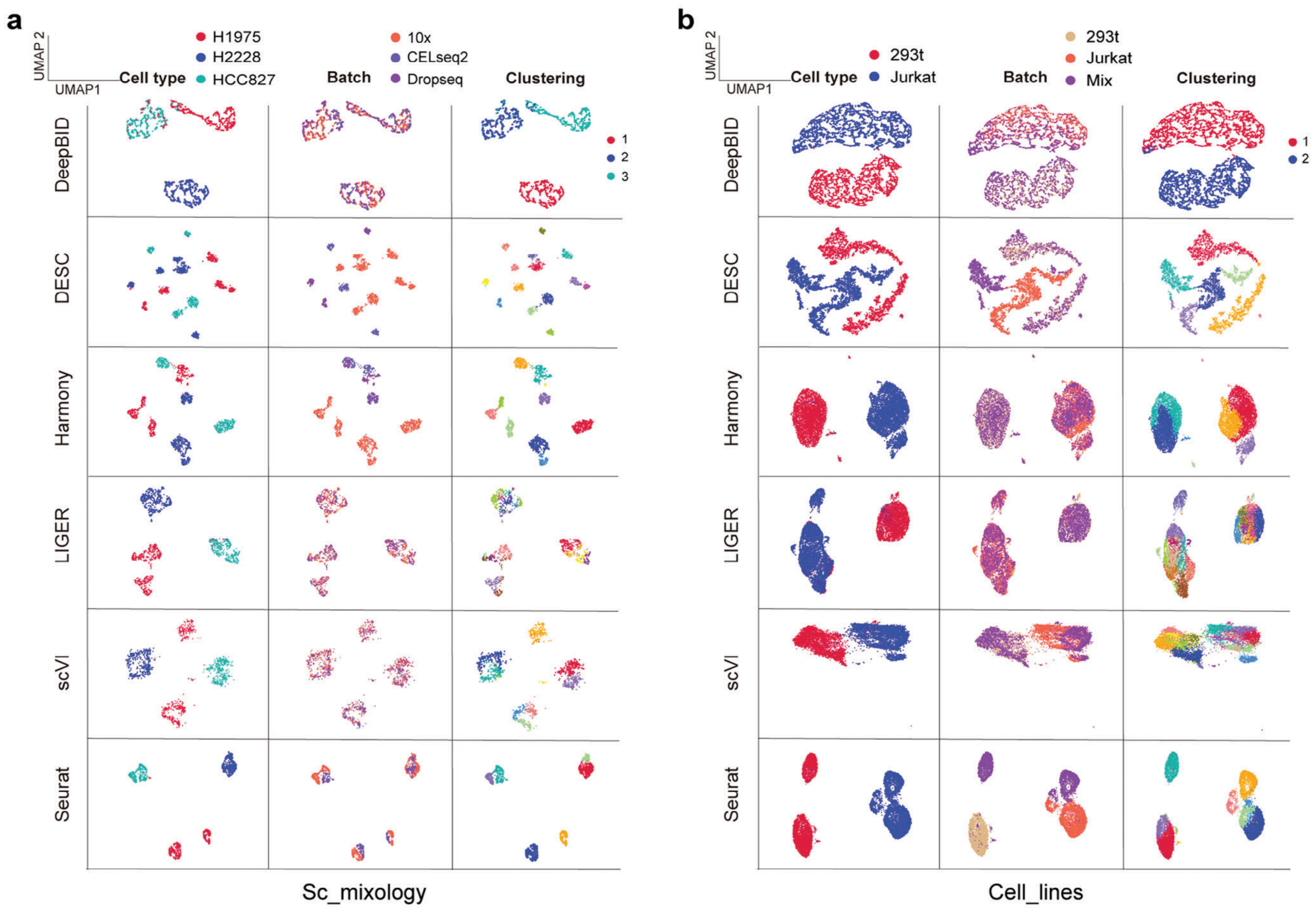

Figure 4. UMAP plots of six methods on the "Sc_mixology" and "Cell_lines" datasets. a) UMAP plots on the "Sc_mixology" dataset that has three cell types. The first column illustrates the original cell type labels by method, the second column the batch labels, and the third column the clustering outcomes after batch effect removal. b) UMAP plots on the "Cell_lines" dataset with two cell types. The figures in the first column show the labels of two cell types for each method. Columns are organized similarly to (a), showing original cell type labels, batch labels, and clustering outcomes.

runtimes across all datasets. For instance, it processed the DC dataset of 569 cells in only 4.45 seconds (s) and the PBMC dataset of 21,526 cells in 416.5 s. We observed a nearly linear increase in runtime with increasing cell numbers. Although DeepBID is slower than two non-deep learning-based methods, Harmony and Seurat, it outperforms the other three deep learning-based methods in terms of speed.

Beyond these compared criteria, we further plotted and manually examined the cell clusters for the five datasets. First, we observed that although the cells identified by DeepBID seem more dispersed compared to other methods, DeepBID effectively preserved the true cell type clusters (see "Sc_mixology" and "Cell_lines" in Figure 4). Other methods tended to produce artificial clusters, suggesting potential artifact introduction during batch effect removal. Thus, these methods may introduce pseudo-cell types into real data analysis. Second, for datasets with many cell types, such as "DC" (Figure S3, Supporting Information), "Pancreas" (Figure S4, Supporting Information), and "PBMCs" (Figure S5, Supporting Information), methods like LIGER and scVI often lead to merged cell clusters, indicating a failure to adequately separate cell types after batch effect adjustment. Overall, these results demonstrate that DeepBID consistently outperformed existing methods in integration and deep clustering across these benchmark datasets.

2.5. DeepBID Enhances Clustering Performance and Downstream DEG Analysis in Integrating Multiple AD scRNA-seq Datasets

To provide a comprehensive demonstration of DeepBID's efficacy and its downstream analytical advantages, we applied DeepBID to three scRNA-seq datasets obtained from three AD patients. These datasets, denoted as AD1-2, AD3-4, and AD5-6, were sourced from post-mortem entorhinal cortex tissue and characterized by the presence of 8 annotated cell types.[38] As a demonstration, we sequentially conducted cell clustering and DEG detection on these datasets, both with and without the application of DeepBID batch correction. Subsequently, we compared their results to assess their improvements.

We first evaluated whether DeepBID could enhance clustering performance. The UMAP plot of DeepBID's latent space illustrates that cells from different datasets aggregate more closely than those plotted from the original datasets (**Figure** 6a,b). For

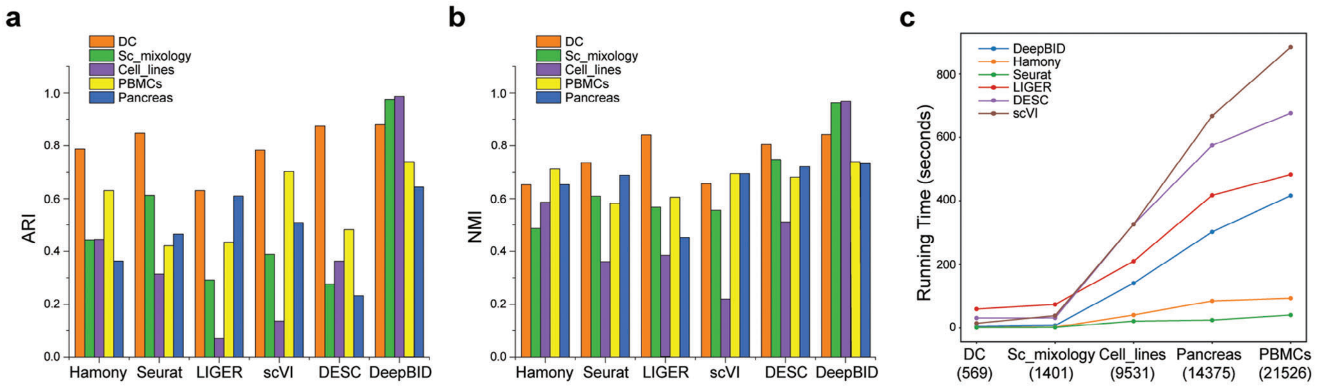

Figure 5. Performances comparison of six methods across five datasets. Cell clustering by six methods after batch removal across five datasets. a) ARI scores. b) NMI scores. c) Running time. The cell numbers for each dataset are provided in parentheses, and detailed runtimes are available in Table S5 (Supporting Information).

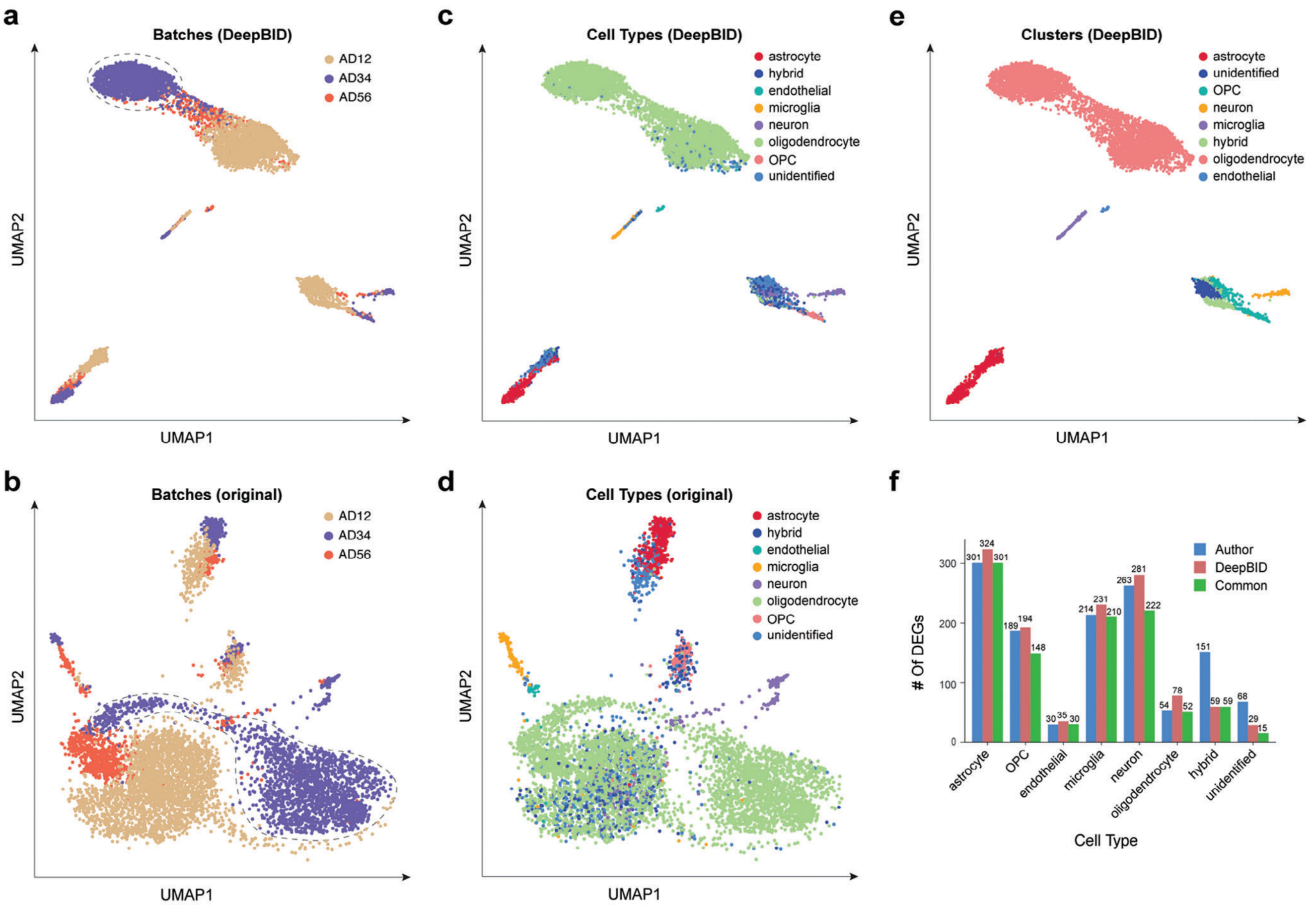

Figure 6. DeepBID enhances clustering performance and downstream analysis for multiple scRNA-seq datasets from AD patients. a) UMAP plot of DeepBID's latent space marked for the three datasets. The dotted line highlights the largest cell group in the AD3-4 dataset. b) UMAP plot of the merged original three datasets without batch correction. c) UMAP plot of DeepBID's latent space marked for eight cell types with original annotations. OPC (Oligodendrocyte progenitor cell). d) UMAP plot of the merged original three datasets without batch correction. Eight cell types are marked with original annotations. e) New clustering based on DeepBID's latent space. Eight cell types were manually annotated in this research. f) Comparison of the DEGs of eight cell types. The DEGs were calculated by using the authors' clustering result and DeepBID analysis respectively. Commonly detected DEGs are shown in green.

instance, the largest cell group (annotated as oligodendrocytes) in the AD3-4 dataset disperses widely based on original data (Figure 6b) but converges into a densely packed group after DeepBID batch correction (marked in a dotted circle in Figure 6a). Furthermore, these oligodendrocytes exhibit close aggregations with those in the AD1-2 and AD5-6 datasets (Figure 6a,c). In contrast, oligodendrocytes primarily overlap with neurons and unidentified cells when using original data (Figure 6d). Integrating these three datasets, DeepBID effectively clustered the eight-cell types (Figure 6e), achieving a high ARI score of 0.8239, significantly higher than the score of 0.6912 calculated without batch correction, and the score of 0.7458 obtained with the second-ranked method, LIGER.

We then investigated whether the enhanced clustering results by DeepBID could improve the detection of cell-specific DEGs. For each of the eight cell types, we predicted the DEGs based on the original authors' labels and the clustering results obtained using DeepBID. Interestingly, the results revealed that in six cell types, the number of detected DEGs increased compared to the original analysis (Figure 6f; Table S7, Supporting Information). These six cell types (microglia, astrocytes, neurons, oligodendrocyte progenitor cells, oligodendrocytes, and endothelial cells) are all well-annotated cell types in previous studies.[38] Additionally, we observed a decrease in DEGs for two cell types (i.e., hybrid and unidentified), which are not well-defined cell types. Furthermore, the changes in DEGs are associated with the methodological differences in clustering results. For example, DeepBID increased the number of astrocytes to 548, compared to the 472 astrocytes originally annotated in the author's analysis. Those astrocytes are closely co-localized with some unidentified cells in the original annotation (red dots and light blue dots in Figure 6c), but our clustering results suggest that all the cells in the cluster would be astrocytes (red dots in Figure 6e). Meanwhile, the number of cells clustered as unidentified decreased to 278 (blue dots in Figure 6e), compared to the original clustered number of 614 (light blue dots in Figure 6c). Taken together, the increased number of DEGs in well-annotated cell types and the decreased number of DEGs in unidentified cell types indicate that DeepBID can indeed improve the cell clusters, consequently enhancing DEG analysis.

3. Discussion

Adjusting batch effects in multiple scRNA-seq datasets is challenging but essential for downstream computational analysis. Deep learning methods have gained popularity for their potential to model complex nonlinear relationships within data, making them particularly suitable for correcting batch effects in scRNA-seq data. In this direction, our method, DeepBID, effectively facilitates cell clustering in a low-dimensional latent space and progressively mitigates batch effects during cell profile reconstruction. When evaluated across five datasets, featuring varying numbers of batches, DeepBID demonstrates superior performance over five other well-established methods. This enhanced accuracy is primarily attributed to two sophisticated techniques implemented in DeepBID: the NB-based autoencoder for data imputation according to the NB distribution, and the simultaneous optimization of batch integration and clustering of data.

Integrating multiple scRNA-seq datasets is not purely a data-driven, but rather a problem-driven approach. We would like to highlight that efforts to remove batch effects, combined with downstream analysis such as clustering, can indeed be beneficial for several reasons. First, incorporating batch correction into analysis workflows preserves signals relevant to the biological questions at hand. Compared with traditional batch correction methods, which may remove or alter biological signals while correcting for batch effects, an integrated approach could minimize this risk by directly optimizing for the preservation of relevant biological information during batch correction. Second, batch correction can be made more robust by directly observing its effects on downstream tasks, thus facilitating result interpretation. Finally, merging batch correction with downstream analysis can be more computationally efficient, as it avoids the need to store and manage separate intermediate representations of the data. Thus, this integrative strategy has the great potential to enhance the accuracy and biological relevance of diverse single-cell data analysis. It is noteworthy that different downstream tasks might be differently affected by batch effects, depending on the specific biological context. For example, batch effects can introduce systematic biases in gene expression profiles across experiments, affecting the estimation of pseudotime and the accuracy of trajectory inference.[39] Meanwhile, cells from different batches may be incorrectly positioned along the trajectory, leading to distorted or inaccurate representations of developmental trajectories. Thus, it is reasonable to carefully analyze these potential factors and design proper loss functions accordingly.

While DeepBID performs well on the datasets used for validation, there are several potential directions for further development. First, it is important to continue testing on a wide range of datasets to optimize hyper-parameters to various types of batch effects across different tissues, cell types, and sequencing platforms. Second, with the emergence of multi-single-cell omics, a possible direction could be adapting DeepBID to integrate scRNA-seq data with other types of single-cell data, such as scATAC-seq,[40] scHi-C,[41–43] and methylation data.[44] Third, DeepBID can be complemented by other strategies to effectively detect rare cell populations. For instance, a recent deep learning-based method called DeepScena is specifically tailored for hierarchical detection of rare cell populations and has demonstrated superior performance compared to other methods.[27] Therefore, an updated approach to detect rare cell types could involve utilizing DeepBID to integrate multiple batches and subsequently applying DeepScena for hierarchical detection. These iterative refinements and enhanced robustness will further bolster the applicability of DeepBID in addressing evolving scRNA-seq datasets and analysis requirements.

4. Experimental Section

DeepBID projected the data of all batches onto a non-linear low-dimensional latent space H and simultaneously learned a soft assignment matrix P of all cells in the latent space. It minimized a total loss function consisting of the NB-based loss, an adaptive loss, a regularization loss, and two KL divergence losses (Figure 1).

Data Preprocessing: Before deploying DeepBID, the Python package SCANPY[33] (version 1.9.2) was employed to preprocess the raw expression count matrices of all batches of a scRNA-seq dataset to select highly variable genes (HVGs) and simply merge batch data. Supposedly, a scRNA-seq dataset contained t batches $B_1, B_2, \cdots, B_t$ with corresponding count matrices $M_1, M_2, \ldots, M_t$. For each batch matrix M_b, the '*scanpy.pp.filter_genes*' function was applied to filter out genes with nonzero counts in fewer than three cells. Next, the '*scanpy.pp.normalize_total*' function was used to normalize the counts per cell by the total counts across all genes, with a size factor of 10^4. This was followed by a log transformation of the normalized matrix by using the '*scanpy.pp.log1p*' function. It then employed the '*scanpy.pp.highly_variable_genes*' function to select the top 1000 HVGs. Lastly, the '*scanpy.AnnData.concatenate*' function was used to concatenate the normalized matrices to retain only the union of the selected HVGs from all batches. This process, comprising the normalization and HVG selection procedures in SCANPY, accomplished preliminary data integration. Subsequently, DeepBID integrated the preprocessed concatenated matrix X of all batches through soft clustering based on an NB-based autoencoder with two KL divergence losses and additional loss functions.

NB-Based Denoising Autoencoder: As the NB distribution was widely used in characterizing gene expression in scRNA-seq data,[23–25] an NB-based denoising autoencoder was applied. This autoencoder mapped the input cell profiles to an embedded latent space H to perform both batch integration and cell clustering.

For the preprocessed matrix $X = (X_1, X_2, \ldots, X_N) \in \mathbb{R}^{f\times N}$, where N represents the total number of cells and $X_j = (X_{1j}, X_{2j}, \cdots, X_{fj})^T$ is an f-dimensional vector representing the profiles of f selected feature genes of the j-th cell, it was assumed that each element X_{ij} of X conformed to an NB distribution parameterized with mean μ_{ij} and dispersion θ_{ij}. That is,

$$P_{NB}\left(\left[X_{ij}\right]|\mu,\theta\right) = \frac{\Gamma\left(\left[X_{ij}\right]+\theta_{ij}\right)}{\left[X_{ij}\right]!\Gamma\left(\theta_{ij}\right)}\left(\frac{\theta_{ij}}{\theta_{ij}+\mu_{ij}}\right)^{\theta_{ij}}\left(\frac{\mu_{ij}}{\theta_{ij}+\mu_{ij}}\right)^{\left[X_{ij}\right]} \quad (1)$$

where $[X_{ij}]$ is the integer obtained by rounding the preprocessed expression value X_{ij}. To predict the putative true transcriptional profiles, in addition to the reconstruction output layer $\hat{X}$, two output layers are appended to estimate the two parameter sets of the NB distribution, specifically the mean set μ and the dispersion set θ, as shown in Figure 1.

The encoder function was defined as $H = f_w(X)$ and the decoder function $\hat{X} = g_{w'}(H)$, where W and W' are the learned weights of the encoder and decoder functions, respectively. To introduce non-linearity and learn complex relationships in the data, both the encoder and decoder functions consisted of fully connected neural networks with "tanh" activation. The "tanh" function was characterized by a smooth, S-shaped curve that mapped input values to output values in the range from −1 to 1. If D represents the last hidden layer of the decoder, two independent fully connected output layers were added to D to estimate the mean set $\mu = \exp(W_\mu D)$ and the dispersion set $\theta = \exp(W_\theta D)$, respectively, where W_μ and W_θ are the learned weights from the last hidden layer of the decoder to the two output layers, respectively. The exponential function was used as the activation function for the mean and dispersion parameters due to their non-negativity. The loss function of the NB-based autoencoder was formulated as the negative log-likelihood of NB distribution:

$$L_{NB}(\mu,\theta) = -\log\left(P_{NB}(X|\mu,\theta)\right) = -\sum_{j=1}^{N}\sum_{i=1}^{f}\log\left(P_{NB}\left(X_{ij}|\mu_{ij},\theta_{ij}\right)\right) \quad (2)$$

Deep Fuzzy Clustering with Adaptive Loss Function and Regularization: Supposedly, all input cell profiles $(X_1, X_2, \ldots, X_N) \in \mathbb{R}^{f\times N}$ were projected into a d-dimensional latent space, represented as $H = (h_1, h_2, \ldots, h_N) \in \mathbb{R}^{d\times N}$. Here, $h_j \in \mathbb{R}^d$ is a low-dimensional representation of $X_j \in \mathbb{R}^f$. A fuzzy k-means clustering with an adaptive loss L_A in the latent space and a regularization term L_O to prevent overfitting in each layer were performed.[31] Let there be K clusters with centers $C = (c_1, \ldots, c_k, \ldots, c_K)$ in the latent space. To force each cell point h_j to move closer to its nearest cluster center c_k, L_A is defined as

$$L_A = \sum_{j=1}^{N}\sum_{k=1}^{K}\left(d_{jk}p_{jk}\|h_j - c_k\|_2^2\right) \quad (3)$$

where $d_{jk} = \frac{\|h_j - c_k\|_2 + 2}{(\|h_j - c_k\|_2 + 1)^2}$ and $\|\cdot\|_2$ represents ℓ_2-norm. L_O is defined as

$$L_O = \sum_{m=1}^{M}\left(\|W^{(m)}\|_F^2 + \|b^{(m)}\|_2^2\right) \quad (4)$$

where $W^{(m)}$ and $b^{(m)}$ are the weight matrix and bias of the m-th layer of the autoencoder, M is the total number of layers in the autoencoder, and $\|\cdot\|_F^2$ represents the squared Frobenius norm of a matrix. Thus, the objective function of the fuzzy k-means with the NB-based autoencoder can be defined as

$$\min_{W,b,P,C} L_{NB} + \lambda_1 L_A + \lambda_2 L_O \quad (5)$$

$$s.t. \sum_{k=0}^{K} p_{jk} = 1,\ 0 \le p_{jk} \le 1,\ j = 1, 2, \ldots, N \quad (6)$$

Integration by Deep Clustering with Two KL Divergence Losses: The fuzzy k-means clustering previously described does not account for the pairwise distances and movements of similar cells. Ideally, clustering should group similar cells from the same batch into the same cluster. To facilitate this, a KL divergence loss function, as the one used in scziDesk,[32] was introduced to reinforce the correlation between similar cells in each batch. Similar to the t-SNE method,[34] the Student t-distribution kernel function with one degree of freedom was used to describe the pairwise similarity among cell points of a batch in the latent space H. Supposedly, cells i and j were from the same batch B, the similarity of cell point h_j to cell point h_i is the conditional probability $q_{j|i}$, which indicates the likelihood of h_i choosing h_j as its neighbor based on their proximity in batch B.

$$q_{j|i} = \begin{cases} \frac{\left(1+\|h_i - h_j\|^2\right)^{-1}}{\sum_{k\in B, k\neq i}\left(1+\|h_i - h_k\|^2\right)^{-1}} & i \neq j \\ 0 & i = j \end{cases} \quad (7)$$

Further, the pairwise similarity is refined with an auxiliary target distribution p, defined as:

$$p_{j|i} = \frac{q_{j|i}^2 / \sum_{l\in B, l\neq j} q_{j|l}}{\sum_{k\in B, k\neq i}\left(q_{k|i}^2 / \sum_{l\in B, l\neq k} q_{k|l}\right)} \quad (8)$$

Using Equations (7) and (8), the first KL divergence loss function is defined:

$$L_{KL1} = KL(p\|q) = \sum_B \sum_{i\in B} \sum_{j\in B} p_{j|i} \log\frac{p_{j|i}}{q_{j|i}} \quad (9)$$

As the target distribution p is derived from q, this approach adopts a self-training strategy, which is instrumental in learning a latent space more amenable to clustering.

To enhance batch integration, a second KL divergence loss function, L_{KL2}, pertaining to batch information to increase the intermixing of cells from different batches within each cluster is introduced. The proportion of cells from batch B in the k-th soft cluster is denoted as $u_{Bk} = \sum_{j\in B} p_{jk} / \sum_{j=1}^{N} p_{jk}$, where p_{jk} is the probability that cell j belongs to the k-th cluster. Suppose that v_B is the ratio of cells in batch B to the total number of cells across all batches, the second KL divergence loss is then defined as

$$L_{KL2} = KL \ (u||v) = \sum_k \sum_B u_{Bk} \log \frac{u_{Bk}}{v_B} \tag{10}$$

Consequently, the final objective function is defined as

$$\min_{W,b,P,C} \mathcal{F} = L_{NB} + \lambda_1 L_A + \lambda_2 L_O + \kappa_1 L_{KL1} + \kappa_2 L_{KL2} \tag{11}$$

$$\text{s.t.} \sum_{k=0}^{K} p_{jk} = 1,\ 0 \le p_{jk} \le 1,\ j = 1, 2, \ldots, N \tag{12}$$

As illustrated in Figure 1, $L_{rec} = \sum_{j=1}^{N} \|x_j - \hat{x}_j\|_2^2$ is initially used as the initial loss functions to pre-train the model, establishing a latent space H, along with weights $W^{(m)}$ and biases $b^{(m)}$. An initial soft cell assignment matrix P_0 is derived from fuzzy k-means. Based on P_0 and H, the centers C of K clusters can be calculated by using Equation (13).

$$c_k = \frac{\sum_{j=1}^{N} p_{jk} h_j}{\sum_{j=1}^{N} p_{jk}} \tag{13}$$

Subsequently, P is updated by Equation (14) with fixed H and C.

$$p_{jk} = \frac{\exp\left(-||h_j - c_k||_2\right)}{\sum_{k=1}^{K} \exp\left(-||h_j - c_k||_2\right)} \tag{14}$$

The optimization of the final objective function $\mathcal{F}$ in Equation (11) utilizes back-propagation and stochastic gradient descent (SGD) algorithms to update W and b, while maintaining P and C. Here, W and b are the weight matrix and bias of the autoencoder layers, respectively, while P is the targeted distribution and C is the centers of the fuzzy k-means clustering. The input of the m-th layer is updated as $x_j^{(m)} = W^{(m)} h_j^{(m-1)} + b^{(m)}$. Successive updates to C and P are performed with the new W, b, and H, iterating until P and C stabilize. The two KL divergence losses within the low-dimensional latent space are anticipated to gradually eliminate batch effects while enhancing clustering accuracy.

Implementation: DeepBID is implemented in Python3 using the PyTorch framework. The NB-based autoencoder is first pretrained for 20 epochs using the Adam optimizer, with a learning rate of 10^{-3} for small datasets ($<10^4$ cells) and 10^{-5} for large datasets ($\geq 10^4$ cells). The clustering procedure is trained for another 20 epochs using the SGD optimizer at the same learning rates. DeepBID utilizes batch training techniques to train the neural network on minibatch sizes of data rather than the entire dataset at once. Here, the minibatch size for both pretraining and training is set to 128 cells similarly to scziDesk.[45] To improve performance and reduce the computational cost, DeepBID selects the top 1,000 HVGs as suggested in the ZINBMM method.[46] For the four hyperparameters in the final objective function, DeepBID sets $\kappa_1 = 0.01$ and $\kappa_2 = 100$ for all datasets, while $\lambda_1 = 5$ and $\lambda_2 = 0.01$ for small datasets, $\lambda_1 = 1$ and $\lambda_2 = 0.0001$ for large datasets.

The encoder features two hidden fully connected layers with sizes set to 500 and 300, respectively, while the decoder mirrors the encoder structure, resulting in an autoencoder of '$f - 500 - 300 - 500 - f$', where f is the dimension of the input data. To speed up the training process and ensure the stability of results across different initializations, a pre-training strategy is employed. Specifically, in the pre-training stage, a '$f - 500 - f$' subnetwork, followed by a '$500 - 300 - 500$' subnetwork was trained first. The pre-trained weights W and biases b are then used as the initial parameters for the autoencoder. All experiments were conducted on a Linux workstation equipped with an Nvidia GTX 1080Ti GPU.

Evaluation Metrics and Datasets: The clustering performance is evaluated by ARI[34] and NMI,[35] which are commonly used to assess the clustering of scRNA-seq data. Given a predicted label set V from clustering and a true cell type label set U for total n cells, the ARI is defined as

$$\mathrm{ARI} = \frac{\sum_{i\in V, j\in U}\binom{n_{ij}}{2} - \left(\sum_{i\in V}\binom{a_i}{2}\sum_{j\in U}\binom{b_j}{2}\right)/\binom{n}{2}}{\frac{1}{2}\left(\sum_{i\in V}\binom{a_i}{2} + \sum_{j\in U}\binom{b_j}{2}\right) - \left(\sum_{i\in V}\binom{a_i}{2}\sum_{j\in U}\binom{b_j}{2}\right)/\binom{n}{2}} \tag{15}$$

where n_{ij} is the number of cells that are present in both cluster $i \in V$ and cell type $j \in U$, a_i is the number of cells in cluster i, and b_j is the number of cells in cell type j. NMI is defined as the mutual information between U and V normalized by the maximum entropy of U and V.

$$\mathrm{NMI} = \frac{\sum_{i\in U, j\in V} n_{ij} \log \frac{n n_{ij}}{a_i b_j}}{\max\left(-\sum_{i\in U} a_i \log \frac{a_i}{n}, -\sum_{j\in U} b_j \log \frac{b_j}{n}\right)} \tag{16}$$

To directly evaluate the effect of data integration, the Local Inverse Simpson Index (LISI) was employed,[17] which featured two specific indicators: iLISI (i.e., integration LISI) and cLISI (i.e., cell-type LISI). These indicators are used respectively to calculate the number of batches and the number of cell types within a local neighborhood. The cLISI and iLISI values are calculated for each cell, and then the distributions of the cLISI and iLISI values in all cells are obtained. The lower bound values of both iLISI and cLISI are 1. An ideal cLISI value of 1 indicates that cells of the same cell type are neighbors, whereas an iLISI of 1 indicates poor integration, suggesting that cells from the same batch are neighbors. The LISI code was downloaded from https://github.com/immunogenomics/LISI.

DeepBID's performance on five scRNA-seq datasets that were sequenced either by a single sequencing platform or generated by multiple platforms (see Table S1, Supporting Information) was evaluated. These datasets, labeled with batch and cell type information, have been used as benchmarks for various integration or clustering tools.[14,17,21,22] Specifically, the "DC" dataset contains four types of human dendritic cells (DCs) across two batches, both sequenced by SMART-seq2 and available from the NCBI GEO database under accession number 'GSE94820'.[47] The "cell_lines" dataset consists of three batches from two cell lines: 1) pure "Jurkat", 2) pure "293T", and 3) a 50/50 mix of "Jurkat" and "293T".[48] The "Sc_mixology" dataset, under GEO accession number GSE118767, includes three human lung adenocarcinoma cell lines HCC827, H1975, and H2228, mixed equally and sequenced using three different platforms (10X Chromium, CEL-seq2 and Drop-seq).[49] The "PBMCs" dataset includes three batches of human peripheral blood mononuclear cells (PBMCs), each processed using the 10X Chromium platform but with different protocols: 3′ end v1 (3pV1), 3′ end v2 (3pV2) and 5′ end (5p) chemistries. Lastly, the "Pancreas" dataset contains five batches of human pancreatic islet cells that were collected from five independent studies using different sequencing platforms.[50–54]

As a real application, DeepBID were applied in the integrative analysis of three scRNA-seq datasets from patients with Alzheimer's disease. The data were obtained from the NCBI GEO database under accession number GSE138852. The three scRNA-seq datasets were derived from entorhinal cortex tissue post-mortem and were labeled as AD1-2 (GSM4120429, 3,028 cells), AD3-4 (GSM4120424, 2,005 cells), and AD5-6 (GSM4120423, 1,040 cells). Each dataset has been annotated for 8 cell types in previous studies.[38] For each cell cluster, the SCANPY package with the command '*scanpy.tl.rank_genes_groups*(adata, method = 'wilcoxon')' was used to find DEGs (one vs others). The cell types outputted by DeepBID were manually annotated by mapping them to the authors' clustering results.

Method Selection and Settings for Comparison: Recent benchmarks of batch effect correction methods have shown that Harmony,[17] LIGER,[18] Seurat4,[10] and scVI[20] outperform others in scRNA-seq data integration.[5,7] Additionally, DESC[22] is a newly designed method for deep clustering-based integration. Therefore, these five methods were selected for comparison with DeepBID. Default parameters were used unless otherwise specified. For methods with alternative parameter settings recommended in their respective publications for specific datasets, those recommendations were adhered to. For example, although the default value of the τ hyperparameter in Harmony is 0, it was set to 5 for pancreas analysis. The number of iNMF factors for LIGER, the number of integration anchors for Seurat4, and the number of principal components (PCs) for Harmony were all fixed at 30. For data integration methods without inherent clustering algorithms, the Leiden algorithm[55] from SCANPY (using the "*scanpy.tl.leiden*" function) for scVI and the clustering methods of Seurat4 (using the "*FindNeighbors*" and "*FindClusters*" functions) for Seurat4, Harmony, and LIGER to cluster the integrated data using default parameter settings were employed. For DESC, its built-in clustering strategy was utilized with default settings. The code used for the five methods is provided in Table S6 (Supporting Information).

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors would like to thank Alicia Petrany for critically reading the manuscript and assistance in AD data analysis. This work was supported by the NSF CAREER Award DBI-2239350 for Y.C.; a key project of the Natural Science Foundation of Tianjin City (19JCZDJC35100) and the National Science Foundation of China (61572358) to S.Z.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

S.Z. and Y.C. performed conceptualization; S.Z. performed methodology; S.Z. and Q.L. acquired software; S.Z., Q.L., G.Z., and Y.C. performed the formal analysis; Y.C. and S.Z. performed writing-original draft preparation; Y.C. and S.Z. wrote-review and editing; All authors have read and agreed to the published manuscript.

Data Availability Statement

Data used in this study are available in Gene Expression Omnibus (GEO) with the accession numbers GSE94820, GSE81076, GSE85241, GSE86469, GSE84133, GES138852 and on the 10x genomics official websites at https://www.10xgenomics.com/resources/datasets (access on June 2023). The code of DeepBID is available at https://github.com/shaoqiangzhang/DeepBID.

Keywords

batch effect, cell typing, data integration, deep learning, scRNA-seq

Received: November 20, 2023
Revised: March 14, 2024
Published online: May 22, 2024

[1] The Tabula Muris Consortium, *Nature* **2020**, *583*, 590.
[2] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, H. Clevers, B. Deplancke, I. Dunham, J. Eberwine, R. Eils, W. Enard, A. Farmer, L. Fugger, B. Göttgens, N. Hacohen, M. Haniffa, M. Hemberg, S. Kim, P. Klenerman, A. Kriegstein, E. Lein, S. Linnarsson, E. Lundberg, J. Lundeberg, P. Majumder, et al., *Elife* **2017**, *6*, e27041.
[3] E. Mereu, A. Lafzi, C. Moutinho, C. Ziegenhain, D. J. McCarthy, A. Álvarez-Varela, E. Batlle, n. Sagar, D. Grün, J. K. Lau, S. C. Boutet, C. Sanada, A. Ooi, R. C. Jones, K. Kaihara, C. Brampton, Y. Talaga, Y. Sasagawa, K. Tanaka, T. Hayashi, C. Braeuning, C. Fischer, S. Sauer, T. Trefzer, C. Conrad, X. Adiconis, L. T. Nguyen, A. Regev, J. Z. Levin, S. Parekh, et al., *Nat. Biotechnol.* **2020**, *38*, 747.
[4] M. P. Emont, C. Jacobs, A. L. Essene, D. Pant, D. Tenen, G. Colleluori, A. Di Vincenzo, A. M. Jørgensen, H. Dashti, A. Stefek, E. McGonagle, S. Strobel, S. Laber, S. Agrawal, G. P. Westcott, A. Kar, M. L. Veregge, A. Gulko, H. Srinivasan, Z. Kramer, E. De Filippis, E. Merkel, J. Ducie, C. G. Boyd, W. Gourash, A. Courcoulas, S. J. Lin, B. T. Lee, D. Morris, A. Tobias, et al., *Nature* **2022**, *603*, 926.
[5] M. D. Luecken, M. Büttner, K. Chaichoompu, A. Danese, M. Interlandi, M. F. Mueller, D. C. Strobl, L. Zappia, M. Dugas, M. Colomé-Tatché, F. J. Theis, *Nat. Methods* **2022**, *19*, 41.
[6] D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, N. Beerenwinkel, A. Mahfouz, L. Pinello, P. Skums, A. Stamatakis, C. S.-O. Attolini, S. Aparicio, J. Baaijens, M. Balvert, B. D. Barbanson, A. Cappuccio, G. Corleone, B. E. Dutilh, M. Florescu, V. Guryev, R. Holmer, K. Jahn, T. J. Lobo, E. M. Keizer, I. Khatri, S. M. Kielbasa, J. O. Korbel, et al., *Genome Biol.* **2020**, *21*, 31.
[7] H. T. N. Tran, K. S. Ang, M. Chevrier, X. Zhang, N. Y. S. Lee, M. Goh, J. Chen, *Genome Biol.* **2020**, *21*, 12.
[8] L. Haghverdi, A. T. L. Lun, M. D. Morgan, J. C. Marioni, *Nat. Biotechnol.* **2018**, *36*, 421.
[9] A. Butler, P. Hoffman, P. Smibert, E. Papalexi, R. Satija, *Nat. Biotechnol.* **2018**, *36*, 411.
[10] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, Y. Hao, M. Stoeckius, P. Smibert, R. Satija, *Cell* **2019**, *177*, 1888.
[11] K. Polanski, M. D. Young, Z. Miao, K. B. Meyer, S. A. Teichmann, J.-E. Park, *Bioinformatics* **2020**, *36*, 964.
[12] L. McInnes, J. Healy, J. Melville, arXiv, **2020**.
[13] B. Zou, T. Zhang, R. Zhou, X. Jiang, H. Yang, X. Jin, Y. Bai, *Front Genet* **2021**, *12*, 708981.
[14] B. Hie, B. Bryson, B. Berger, *Nat. Biotechnol.* **2019**, *37*, 685.
[15] X. Yu, X. Xu, J. Zhang, X. Li, *Nat. Commun.* **2023**, *14*, 960.
[16] Y. Liu, T. Wang, B. Zhou, D. Zheng, *Nat. Biotechnol.* **2021**, *39*, 877.
[17] I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-R. Loh, S. Raychaudhuri, *Nat. Methods* **2019**, *16*, 1289.
[18] J. D. Welch, V. Kozareva, A. Ferreira, C. Vanderburg, C. Martin, E. Z. Macosko, *Cell* **2019**, *177*, 1873.
[19] T. Wang, T. S. Johnson, W. Shao, Z. Lu, B. R. Helm, J. Zhang, K. Huang, *Genome Biol.* **2019**, *20*, 165.
[20] R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, N. Yosef, *Nat. Methods* **2018**, *15*, 1053.
[21] D. Wang, S. Hou, L. Zhang, X. Wang, B. Liu, Z. Zhang, *Genome Biol.* **2021**, *22*, 63.
[22] X. Li, K. Wang, Y. Lyu, H. Pan, J. Zhang, D. Stambolian, K. Susztak, M. P. Reilly, G. Hu, M. Li, *Nat. Commun.* **2020**, *11*, 2338.
[23] V. Svensson, *Nat. Biotechnol.* **2020**, *38*, 147.
[24] W. Chen, Y. Li, J. Easton, D. Finkelstein, G. Wu, X. Chen, *Genome Biol.* **2018**, *19*, 70.
[25] T. S. Andrews, M. Hemberg, *Bioinformatics* **2019**, *35*, 2865.

[26] W. Tang, F. Bertaux, P. Thomas, C. Stefanelli, M. Saint, S. Marguerat, V. Shahrezaei, *Bioinformatics* **2020**, *36*, 1174.
[27] T. Lei, R. Chen, S. Zhang, Y. Chen, *Brief Bioinform* **2023**, *24*, bbad335.
[28] M. Amodio, D. van Dijk, K. Srinivasan, W. S. Chen, H. Mohsen, K. R. Moon, A. Campbell, Y. Zhao, X. Wang, M. Venkataswamy, A. Desai, V. Ravi, P. Kumar, R. Montgomery, G. Wolf, S. Krishnaswamy, *Nat. Methods* **2019**, *16*, 1139.
[29] T. Tian, J. Wan, Q. Song, Z. Wei, *Nat Mach Intell* **2019**, *1*, 191.
[30] Y. Ryu, G. H. Han, E. Jung, D. Hwang, *Mol. Cells* **2023**, *46*, 106.
[31] R. Zhang, X. Li, H. Zhang, F. Nie, *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 2814.
[32] L. Chen, W. Wang, Y. Zhai, M. Deng, *NAR Genom. Bioinformat.* **2020**, *2*, lqaa039.
[33] F. A. Wolf, P. Angerer, F. J. Theis, *Genome Biol.* **2018**, *19*, 15.
[34] M. Lvd, G. Hinton, *J. Mach. Learning Res.* **2008**, *9*, 2579.
[35] A. Strehl, J. Ghosh, *J Mach Learn Res* **2003**, *3*, 583.
[36] Z. Yao, C. T. J. van Velthoven, M. Kunst, M. Zhang, D. McMillen, C. Lee, W. Jung, J. Goldy, A. Abdelhak, M. Aitken, K. Baker, P. Baker, E. Barkan, D. Bertagnolli, A. Bhandiwad, C. Bielstein, P. Bishwakarma, J. Campos, D. Carey, T. Casper, A. B. Chakka, R. Chakrabarty, S. Chavan, M. Chen, M. Clark, J. Close, K. Crichton, S. Daniel, P. DiValentin, T. Dolbeare, et al., *Nature* **2023**, *624*, 317.
[37] C. Qiu, B. K. Martin, I. C. Welsh, R. M. Daza, T.-M. Le, X. Huang, E. K. Nichols, M. L. Taylor, O. Fulton, D. R. O'Day, A. R. Gomes, S. Ilcisin, S. Srivatsan, X. Deng, C. M. Disteche, W. S. Noble, N. Hamazaki, C. B. Moens, D. Kimelman, J. Cao, A. F. Schier, M. Spielmann, S. A. Murray, C. Trapnell, J. Shendure, *Nature* **2024**, *626*, 1084.
[38] A. Grubman, G. Chew, J. F. Ouyang, G. Sun, X. Y. Choo, C. McLean, R. K. Simmons, S. Buckberry, D. B. Vargas-Landin, D. Poppe, J. Pflueger, R. Lister, O. J. L. Rackham, E. Petretto, J. M. Polo, *Nat. Neurosci.* **2019**, *22*, 2087.
[39] W. Saelens, R. Cannoodt, H. Todorov, Y. Saeys, *Nat. Biotechnol.* **2019**, *37*, 547.
[40] K. Zhang, J. D. Hocker, M. Miller, X. Hou, J. Chiou, O. B. Poirion, Y. Qiu, Y. E. Li, K. J. Gaulton, A. Wang, S. Preissl, B. Ren, *Cell* **2021**, *184*, 5985.
[41] T. Nagano, Y. Lubling, T. J. Stevens, S. Schoenfelder, E. Yaffe, W. Dean, E. D. Laue, A. Tanay, P. Fraser, *Nature* **2013**, *502*, 59.
[42] S. Collombet, N. Ranisavljevic, T. Nagano, C. Varnai, T. Shisode, W. Leung, T. Piolot, R. Galupa, M. Borensztein, N. Servant, P. Fraser, K. Ancelin, E. Heard, *Nature* **2020**, *580*, 142.
[43] I. M. Flyamer, J. Gassler, M. Imakaev, H. B. Brandão, S. V. Ulianov, N. Abdennur, S. V. Razin, L. A. Mirny, K. Tachibana-Konwalski, *Nature* **2017**, *544*, 110.
[44] C. H. Ludwig, L. Bintu, *Development* **2019**, *146*, dev170217.
[45] L. Chen, W. Wang, Y. Zhai, M. Deng, *NAR Genom. Bioinformat.* **2020**, *2*, lqaa039.
[46] Y. Li, M. Wu, S. Ma, M. Wu, *Genome Biol.* **2023**, *24*, 208.
[47] A.-C. Villani, R. Satija, G. Reynolds, S. Sarkizova, K. Shekhar, J. Fletcher, M. Griesbeck, A. Butler, S. Zheng, S. Lazo, L. Jardine, D. Dixon, E. Stephenson, E. Nilsson, I. Grundberg, D. McDonald, A. Filby, W. Li, P. L. De Jager, O. Rozenblatt-Rosen, A. A. Lane, M. Haniffa, A. Regev, N. Hacohen, *Science* **2017**, *356*, eaah4573.
[48] G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, et al., *Nat. Commun.* **2017**, *8*, 14049.
[49] L. Tian, X. Dong, S. Freytag, K.-A. Lê Cao, S. Su, A. JalalAbadi, D. Amann-Zalcenstein, T. S. Weber, A. Seidi, J. S. Jabbari, S. H. Naik, M. E. Ritchie, *Nat. Methods* **2019**, *16*, 479.
[50] Å. Segerstolpe, A. Palasantza, P. Eliasson, E.-M. Andersson, A.-C. Andréasson, X. Sun, S. Picelli, A. Sabirsh, M. Clausen, M. K. Bjursell, D. M. Smith, M. Kasper, C. Ämmälä, R. Sandberg, *Cell Metab.* **2016**, *24*, 593.
[51] M. Baron, A. Veres, S. L. Wolock, A. L. Faust, R. Gaujoux, A. Vetere, J. H. Ryu, B. K. Wagner, S. S. Shen-Orr, A. M. Klein, D. A. Melton, I. Yanai, *Cell Syst* **2016**, *3*, 346.
[52] N. Lawlor, J. George, M. Bolisetty, R. Kursawe, L. Sun, V. Sivakamasundari, I. Kycia, P. Robson, M. L. Stitzel, *Genome Res.* **2017**, *27*, 208.
[53] D. Grün, M. J. Muraro, J.-C. Boisset, K. Wiebrands, A. Lyubimova, G. Dharmadhikari, M. van den Born, J. van Es, E. Jansen, H. Clevers, E. J. P. de Koning, A. van Oudenaarden, *Cell Stem Cell* **2016**, *19*, 266.
[54] M. J. Muraro, G. Dharmadhikari, D. Grün, N. Groen, T. Dielen, E. Jansen, L. van Gurp, M. A. Engelse, F. Carlotti, E. J. P. de Koning, A. van Oudenaarden, *Cell Syst* **2016**, *3*, 385.
[55] V. A. Traag, L. Waltman, N. J. van Eck, *Sci. Rep.* **2019**, *9*, 5233.