*Briefings in Bioinformatics*, 2022, **23(5)**, 1–13

https://doi.org/10.1093/bib/bbac327

**Problem Solving Protocol**

# Scalable batch-correction approach for integrating large-scale single-cell transcriptomes

Xilin Shen[†], Hongru Shen[†], Dan Wu, Mengyao Feng, Jiani Hu, Jilei Liu, Yichen Yang, Meng Yang, Yang Li, Lei Shi, Kexin Chen and Xiangchun Li

Corresponding authors: Xiangchun Li, Tianjin Cancer Institute, Tianjin Medical University Cancer Institute and Hospital, Huanhu Xi Road, Tiyuan Bei, Hexi District, Tianjin 300060, China. Tel. (Fax): (86)22-23372231; E-mail: lixiangchun@tmu.edu.cn; Kexin Chen, Department of Epidemiology and Biostatistics, Tianjin Medical University Cancer Institute and Hospital, Huanhu Xi Road, Tiyuan Bei, Hexi District, Tianjin, 300060, China. Tel: (86) 2237 2231; E-mail: chenkexin@tmu.edu.cn; Lei Shi, Department of Biochemistry and Molecular Biology, School of Basic Medical Sciences, Tianjin Medical University, Qixiangtai Road, Heping District, Tianjin 300070, China. Tel.: (86)22-83336998; E-mail: shilei@tmu.edu.cn

[†]Xilin Shen and Hongru Shen contributed equally.

## Abstract

Integration of accumulative large-scale single-cell transcriptomes requires scalable batch-correction approaches. Here we propose Fugue, a simple and efficient batch-correction method that is scalable for integrating super large-scale single-cell transcriptomes from diverse sources. The core idea of the method is to encode batch information as trainable parameters and add it to single-cell expression profile; subsequently, a contrastive learning approach is used to learn feature representation of the additive expression profile. We demonstrate the scalability of Fugue by integrating all single cells obtained from the Human Cell Atlas. We benchmark Fugue against current state-of-the-art methods and show that Fugue consistently achieves improved performance in terms of data alignment and clustering preservation. Our study will facilitate the integration of single-cell transcriptomes at increasingly large scale.

**Keywords:** single cell, data integration, scalability, deep learning

## Introduction

Single-cell sequencing offers tremendous opportunities for biomedical research to explore the cellular ecosystem and molecular mechanisms [1]. Advances in single-cell technologies have spurred the establishment of several public repositories of single-cell data, including the Human Cell Atlas (HCA), the Single-Cell Expression Atlas and the Mouse Cell Atlas [2, 3]. The HCA project is committed to curate millions to trillions of single cells for constructing a comprehensive reference map of all human cells. As can be foreseen, integration of super large-scale single cells across heterogeneous tissues from diverse sources will be a leading wave for deep exploration of biology [4, 5]. Therefore, scalable computational methods are crucial for integration of single-cell transcriptomes and subsequently their translation into biological significance.

Batch effects are fundamental issues to be addressed for integration of single-cell transcriptomes. Batch effects are inevitable as single-cell data were generated by various groups with diverse experimental protocols and sequencing platforms [6, 7]. Considerable progress has been made on batch correction of single-cell expression. For instance, mutual nearest neighbors (MNN) [8], Scanorama [9] and BBKNN [10] identify MNNs to guide single-cell integration. Seurat [11] utilizes canonical correlation analysis to identify correlations across different datasets to construct MNNs for batch correction. Harmony [12] corrects for batch effects via clustering similar cells from different batches while maximizing the diversity among different batches within each cell cluster. scVI, iMAP and CLEAR [13–15] applies deep learning approaches to learn shared embedding space of gene expression profiles among different

**Xilin Shen** is a PhD student at Tianjin Medical University. Shen is currently working on biological knowledge mining based on large-scale single cell data.
**Hongru Shen** is a PhD student at Tianjin Medical University. She is currently working on the analysis of single-cell transcriptomes.
**Dan Wu** is a PhD student at Tianjin Medical University Cancer Institute and Hospital.
**Mengyao Feng** is a master student at Tianjin Medical University. She is currently working on identifying immunotherapy biomarkers.
**Jiani Hu** is a master student at Tianjin Medical University. She is working on the dissecting tumor genomics.
**Jilei Liu** is a master student at Tianjin Medical University. He is familiar with biostatistics and currently working on genomic analysis.
**Yichen Yang** is a research assistant at Tianjin Medical University Cancer Institute and Hospital. He has done abundant studies on bioinformatics.
**Meng Yang** is a research assistant at Tianjin Medical University Cancer Institute and Hospital. She is an expert on tumor genomics and biostatistics.
**Yang Li** is a scientist at Tianjin Medical University Cancer Institute and Hospital. She has done abundant studies on transcriptome and gut microbiome.
**Lei Shi** is a professor at Tianjin Medical University. He is an expert on molecular biology of cancer.
**Kexin Chen** is a professor at Tianjin Medical University Cancer Institute and Hospital. She is an expert on epidemiology.
**Xiangchun Li** is a professor at Tianjin Medical University Cancer Institute and Hospital. He has extensive experience in deciphering human cancer genomes via bioinformatics and deep learning.

**Received:** April 30, 2022. **Revised:** June 26, 2022. **Accepted:** July 19, 2022

datasets for the elimination of batch effects. However, these methods are not designed for the integration of super large-scale single cells.

In this study, we present self-supervised learning [16] approach termed Fugue for batch correction of super large-scale single-cell transcriptomes. We encode batch information as trainable parameters and added them into expression profiles. The core idea of Fugue is the use of contrastive learning method [17] for narrowing the gap between the original expression profile of each cell and its different augmented views. In concept, the expression profiles of cells from different batches could be seen as superposition of the biological information and different batch information. Fugue incorporates addictive batch information as learnable parameters into gene expression matrix. The batch information can be properly represented after training. By taking batch information as trainable variable, Fugue is scalable and flexibly in integrating atlasing-scale data with fixed memory footprint.

We demonstrated the scalability and efficiency of Fugue on a large-scale dataset curated from HCA repository. We benchmarked its performance on diverse cohorts along with current state-of-the-art methods. We showed that Fugue achieved favorable performance as compared with the other methods. The reference map of HCA dissected by Fugue demonstrated that it can learn smooth embedding for time course trajectory and joint embedding space for immune cells from heterogeneous tissues.

## Results

### Overview of fugue

Fugue integrates single cells through learning batch information by contrastive learning. Specifically, Fugue uses deep neural network as feature encoder to learn feature representation of cell expression profiles. Fugue embeds batch information of the input expression matrix as a learnable matrix (i.e. batch embedding matrix) and adds it to the expression matrix (Figure 1**A** and **B**). We used densely connected neural network of 21 layers [18] as feature encoder to encode feature presentation of the additive expression matrix. The feature encoder was trained iteratively via contrastive learning (Figure 1**C**) [16]. We employed the InfoNCE loss to minimizes the distance between the cell and its noise-added (i.e. data augmentation) view and maximizes the distance between different cells. Subsequently, we used the trained feature encoder to extract feature representations of single-cell expression profiles (Figure 1**D**). The extracted features can be used in downstream analyses such as single-cell cluster delineation and developmental trajectory inference (Figure 1**E**). Details are described in Methods section.

### Benchmark evaluations

We generated a *simulation dataset* that consists of 30 000 cells from three cell types among five batches. Different cell types were aggregated per se by batch (Figure 2**A**) before batch correction. We observed that cells of the same type were well-mixed and cells of different types were dispersed across batches after corrected by Fugue (Figure 2**B**). In addition, we demonstrated that Fugue could maintain batch-specific cell types (Figure 2**C**) by removing a specific cell type from the four batches (named *simulation_rm dataset*; see Methods and Supplementary Figure 1).

We compared Fugue to 10 single-cell integration methods on the *cell line* ($n = 9531$) and peripheral blood mononuclear cell (*PBMC*) ($n = 42\,667$) *datasets* (Figure 2**D** and **G**). We used the harmonic average value of kBET and adjusted rand index (ARI) (i.e. F1 score) as the primary performance metric (see Methods). Fugue removed batch-specific variations and clustered cells into biologically coherent groups (Figure 2**E** and **H**). We found that Fugue had equally good performance as the benchmark methods (Figure 2**F** and **I**). The uniform manifold approximation and projection for dimension reduction (UMAP) plots of the benchmark methods after batch correction were provided in Supplementary Figures 2 and 3. ARI and kBET scores were provided in Supplementary Figure 4. We further showed that the F1 score achieved by Fugue is stable with respect to different degree of data augmentation (Supplementary Figure 5).

### Fugue could accurately remove batch effects

We obtained distinctive cell clusters (Supplementary Figure 6) by applying Fugue to integrate all available data from HCA (Supplementary Table 1). The batch effect removal efficiency of Fugue was evaluated on the *census of immune, lung* and *brain* dataset from HCA.

We observed that there is minimal overlap among common cell types from cord blood ($n = 133\,264$) and bone marrow ($n = 176\,571$) in the *census of immune dataset* before integration (Figure 3**A**). Fugue was able to cluster cells into biologically coherent groups while removing batch-specific variations (Figure 3**A**). Quantitatively, Fugue obtained comparable performance as the benchmark methods (Figure 3**B** and Supplementary Figure 4). The UMAP plots of benchmark methods were provided in Supplementary Figure 7. UMAP plots highlighted by marker gene expression for each cell type were provided in Supplementary Figure 8.

In the *lung dataset*, cells from C30.1 ($n = 75\,387$) and C47 ($n = 2532$) cohorts were mapped into shared areas (Figure 3**D**) with Fugue correction in contrast to minimal overlap without batch correction (Figure 3C). The batch correction result of the benchmark methods was shown in Supplementary Figure 9. We benchmarked the methods based on kBET score, since cell type labels were not available. Quantitatively, Fugue achieved kBET scores comparable to these methods (Figure 3F). The ground truth labels were not able for this dataset; therefore, we were not able to calculate ARI and the F1 score. There are 11 cell types dissected from the *lung* dataset, including monocytes, mast cells and ciliated cells (Figure 3E). Conventional cell markers [19, 20] were expressed uniquely

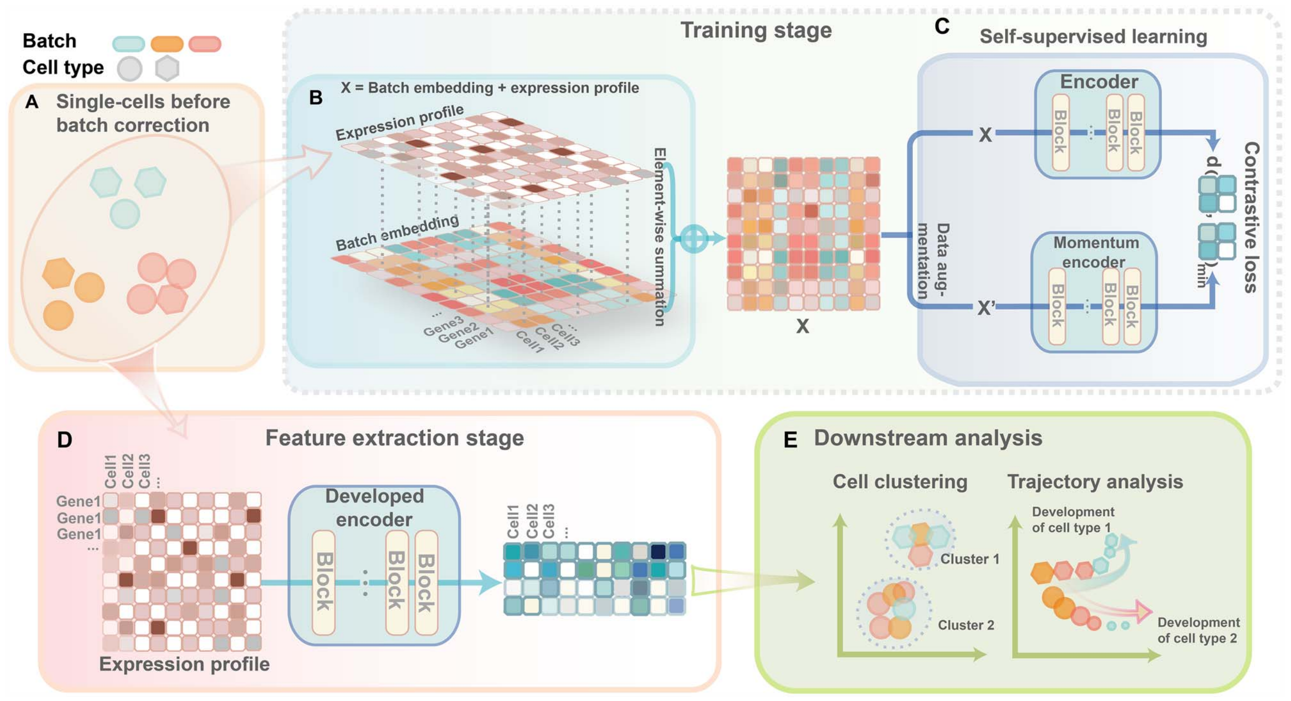

**Figure 1.** Overview of Fugue. (**A**) Given a set of uncorrected single cells, (**B**) Fugue embedded their batch information as a learnable matrix and added them to the expression profile for feature encoder training. (**C**) The feature encoder was trained with contrastive loss. (**D**) At the feature extraction stage, single-cell expression profiles were provided to the feature encoder to extract embedding representation. (**E**) The embedding representation could be utilized for downstream analysis such as cell clustering and trajectory analysis.

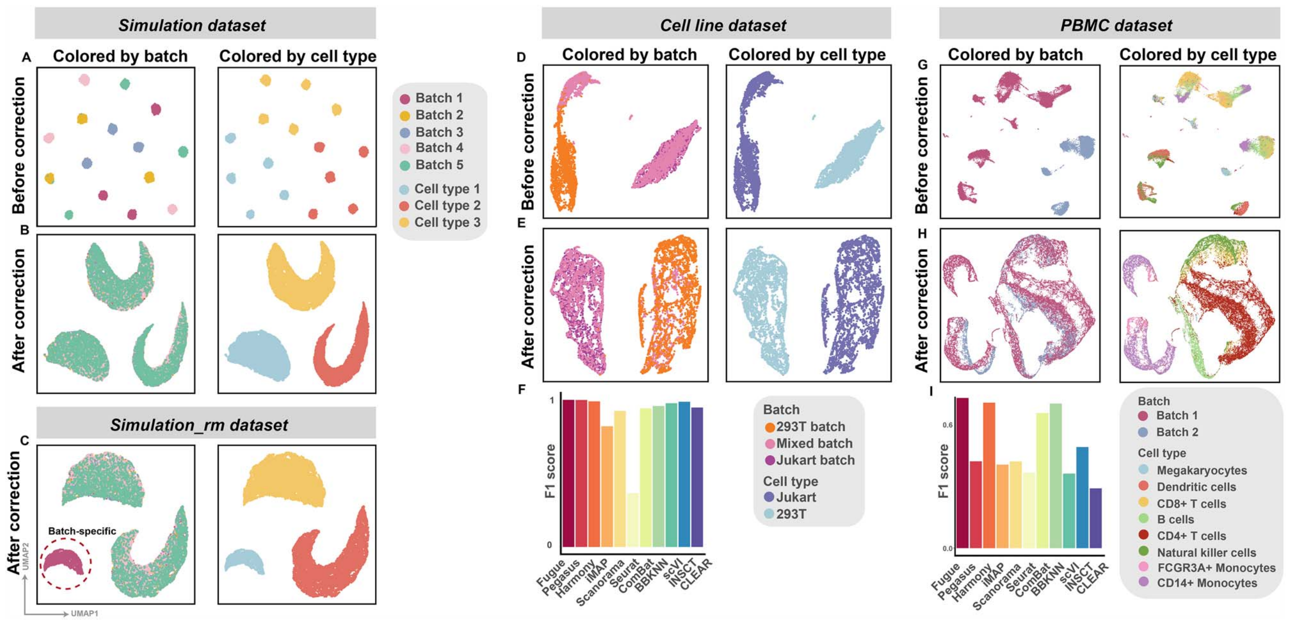

**Figure 2.** Benchmark of batch-correction performance of Fugue across the *simulation*, *simulation_rm*, *cell line* and *PBMC datasets*. (**A**) UMAP plot of cells from the *simulation dataset*, which consists of five different batches and three cell types. (**B–C**) UMAP visualization of Fugue batch effect removing performance on the (**B**) *simulation* and (**C**) *simulation_rm dataset*. (**D**) UMAP plot displays cells from the *cell line dataset*, which consists of three different batches and two cell types. (**E**) UMAP plot of Fugue batch effect removing performance on the *cell line dataset*. (**F**) Quantitative assessments of different batch effect removal methods on the *cell line dataset*. (**G**) UMAP plot displaying cells from the *PBMC dataset*, which consists of two different batches and 8 cell types. (**H**) UMAP plot of Fugue batch effect removing performance on the *PBMC dataset*. (**I**) Quantitative assessments of different batch effect removing methods on the *PBMC dataset*. For (**A–E**, **G–H**), cells are colored by batch (left panel) and cell type (right panel).

within each cell cluster (Figure 3G) and invariant across batches (Figure 3H). Additionally, Fugue achieved comparable performance when using sample barcodes or batch information provided by the dataset as batch labels (Supplementary Figures 10–11).

We obtained similar results on the *brain dataset* (Supplementary Figure 12). These results demonstrated that Fugue can robustly integrate cells from multiple studies. Meanwhile, we observed that Fugue achieved more stable performance versus the other methods

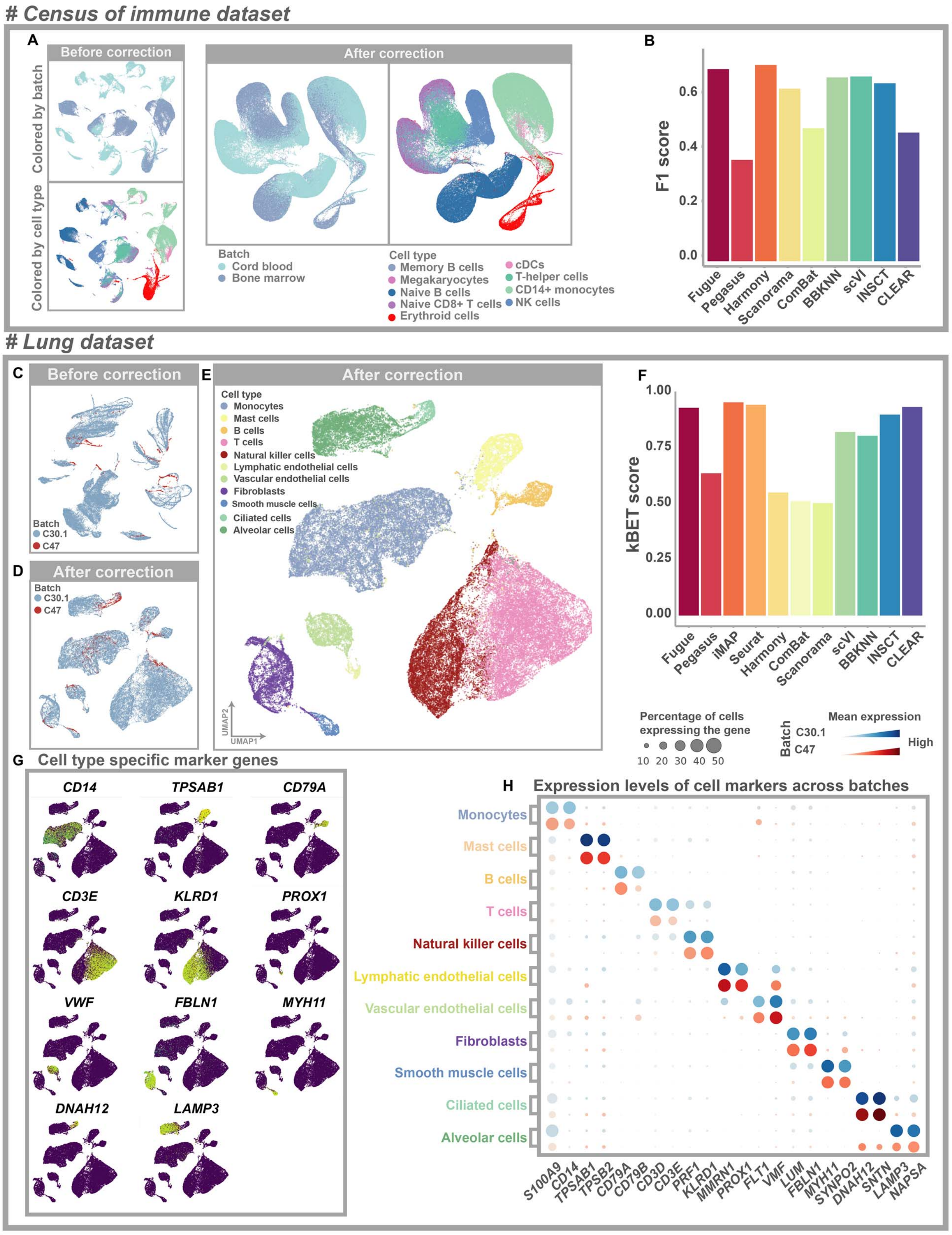

**Figure 3.** Assessment of the batch-correction performance of Fugue. (**A**) UMAP plot of the *census of immune project* before and after batch removal. (**B**) Quantitative assessments of different batch effect removal methods on the *census of immune project*. (**C–E**) UMAP plot depicting cells in the *lung dataset* before (**C**) and after (**D–E**) Fugue integration. Cells are colored by batch in (**C–D**) and cell cluster label in (**E**). (**F**) Bar plot depicting kBET scores of different batch effect removing methods on the *lung dataset*. (**G**) Expression of cell type markers across the feature embedding space. Dark and light colors represent low and high relative expression values, respectively. (**H**) Dot plot representing cell markers across batches. The size of each circle reflects the percentage of cells in a cluster where the gene is detected, and the color intensity reflects the average expression level within each cluster.

among the datasets examined above (Supplementary Figure 13).

## Fugue captures the real batch information

We hypothesize that sequencing samples of the same cohort are subjected to lower batch variation. Therefore, batch embeddings of samples from the same cohort should be more similar than those from different cohorts. We extracted the batch embedding matrix of 373 samples from 53 datasets of HCA. The result showed that samples from the same dataset had higher batch embedding similarity as compared with samples from different datasets (mean cosine similarity, within-dataset = 0.35, between-dataset = 0.04; *t*-test, $P < 1e{-}10$; Supplementary Figure 14A–C). Batch embeddings of the four patients from the C2 cohort are almost identical (mean cosine similarity = 0.97; Supplementary Figure 14D), which is consistent with the previous report that there were no batch effects among these four patients [21]. For the *census of immune dataset*, we observed a higher similarity of batch embeddings within the same batch than between batches (mean cosine similarity, within-batch = 0.89, between-batch = 0.66; *t*-test, $P < 1e{-}10$; Supplementary Figure 14E). For the PBMC and tonsil tissues from C39 subjected to the same sequencing protocol [22], we also observed high similarity among them, especially among samples from the same tissue (mean cosine similarity, within-tissue = 0.87, between-tissue = 0.67; *t*-test, $P < 1e{-}10$; Supplementary Figure 14F). In the C90 dataset consisted of brain tissue samples from mouse and human, we observed higher cosine similarity for samples from the same batch than from different batch (cosine similarity, 0.37 versus 0.05; *t*-test, $P < 1.5e{-}4$; Supplementary Figure 14G). In C12.1 dataset consisted of samples subjected to Smart-seq and 10X sequencing protocols, we observed high similarity among sample from the same protocol and low similarity among samples from different protocols (cosine similarity, 0.83 versus −0.22; *t*-test, $P < 1e{-}8$; Supplementary Figure 14H).

## Fugue aligns precise immune cell subtypes in HCA

We delineated 46 cell clusters in the HCA repository (Supplementary Figures 6, 15A and Supplementary Table 2). There are 36 clusters formed by cells from multiple datasets, with each cluster consisted of cells from seven datasets on average. There are 10 cell-type specific clusters, with >95% of cells from a specific dataset (Supplementary Figure 15B).

We re-clustered the immune cell subtypes in the HCA repository into 17 subtypes (Figure 4A). We observed that T cells and B cells were divided into eight and three subtypes, respectively. Different cell types were distinct from each other (Figure 4A). Canonical markers were expressed in corresponding cell types (Figure 4B). For example, the Pan-B cell markers *CD79A* and *CD79B* were expressed in B cell clusters. B cell precursor specific markers *VPREB1* and *IGLL1* were expressed in relevant cell type. We observed stable expression of marker genes among different batches (Figure 4C, D and Supplementary Figure 17). For example, natural killer (NK) cell markers *PRF1* and *KLRD1* were expressed in all of the NK cells from 27 cohorts (Figure 4D).

## Fugue integrates time course development trajectories

We evaluated whether Fugue could reserve cell developmental trajectories after batch correction. For the mouse cardiac cells of embryonic (E) day 10.5, 13.5 and 16.5 from two datasets (i.e. C18 and C20), previous studies reported that identical cardiac cell types from different developmental stages are analogous [23, 24]. However, cells featured by the same marker tend to form distinct clusters aligned with different sequencing protocols before Fugue correction (Figure 5A and Supplementary Figure 18), suggesting strong batch effect among different cohorts. Fugue correction led to sequentially connected cell clusters among different embryonic periods (Figure 5B). We classified the single cells into five cell types based on cell markers (Figure 5C, D, Supplementary Figures 19 and 20). The force-directed layout embedding (FLE) dimension reduction showed that Fugue captured embryonic developmental trajectories for each cell type (Figure 5E). Cardiac cells from E10.5, E13.5 and E16.5 were orderly arranged according to pseudo-time trajectories (Figure 5E). The pseudo-time scores varied significantly among different embryonic days (*t*-test, adjusted $P < 0.05$, Supplementary Figure 21), with the highest at E16.5, followed by E13.5 and the lowest at E10.5. The expression patterns of canonical cell differentiation markers were consistent with developmental stages (Supplementary Figure 22). For instance, early erythrocyte markers *GYPA* and *TFRC* were highly expressed in E10.5 erythrocytes and negatively correlated with pseudo-time, which was consistent with the previous studies [25, 26].

In the *census of immune dataset*, we observed overlap among different cell types from cord blood and bone marrow datasets along pseudo-time trajectory in the FLE plot (Supplementary Figure 23). Meanwhile, we found that hematopoietic stem cell (HSC) quadrifurcated into B cell, T cell, mono-dendritic and megakaryocyte–erythroid trajectories (Figure 5F). The trajectories were ordered by cell developmental stages and branched by cell differentiation types (Figure 5G–J). The cell development trajectories derived from the other methods were provided in Supplementary Figure 24. Inspired by Luecken *et al.* [6], we calculated the trajectory conservation score of erythrocyte development. Fugue ranks on the top among these benchmarked methods, whereas Harmony, scVI and Scanorama also achieved high trajectory conservation scores (Supplementary Figures 25 and 26). Taken together, Fugue correction reserves cell developmental trajectories.

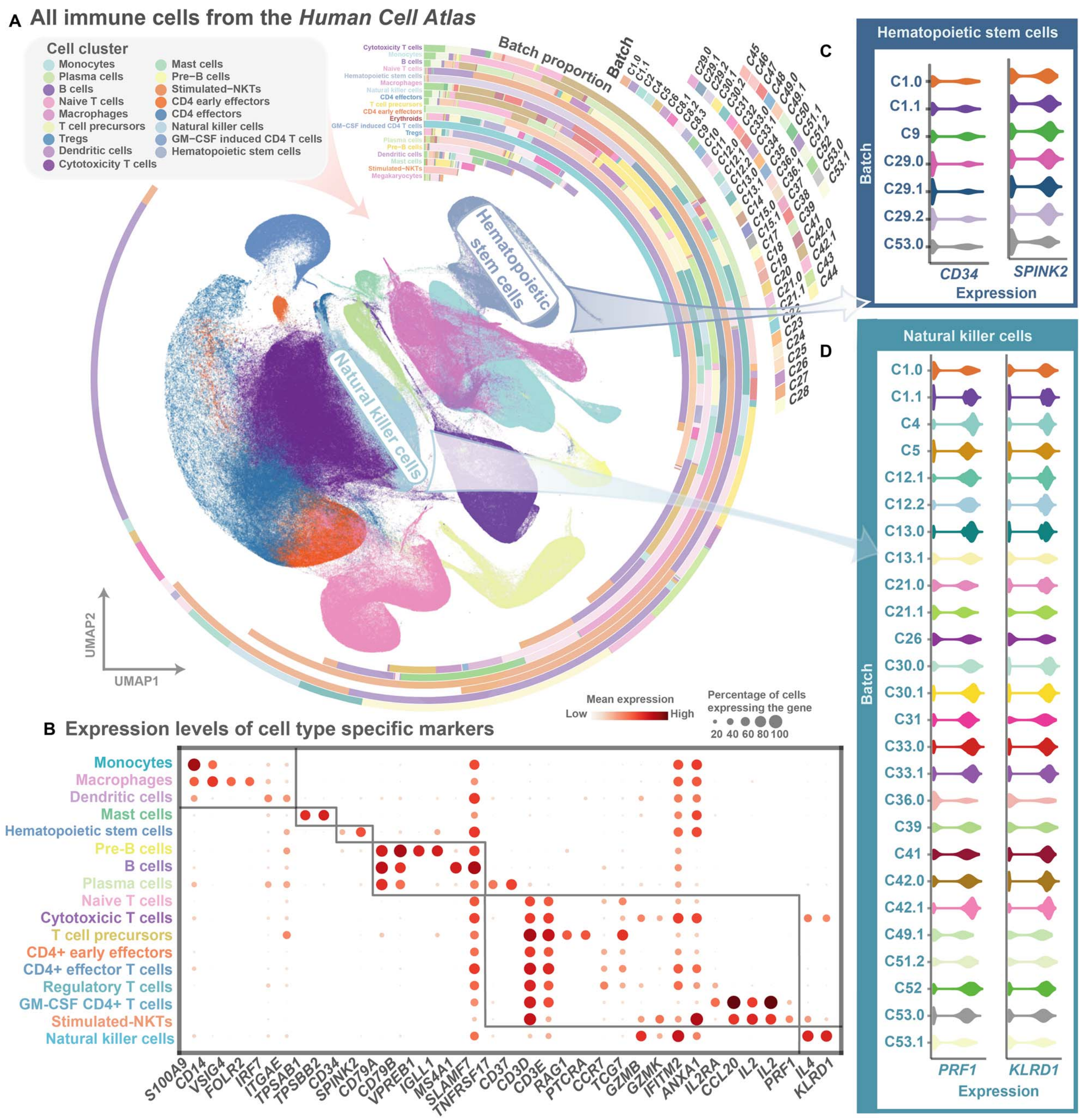

**Figure 4.** Joint analysis of all immune cells across *HCA* with Fugue. (**A**) UMAP plot of the 17 immune cell types inferred from Fugue. Cells are colored by cell type labels. (**B**) Dot plot showing cell type markers across cell clusters. The size of each circle reflects the percentage of cells in a cluster where the gene is detected, and the color intensity reflects the average expression level within each cluster. (**C**–**D**) Violin plot deciphering expression levels of cell type markers for HSCs (**C**) and NK cells (**D**) across HCA cohorts. Cohorts with >1000 cells in each cluster were displayed.

## Fugue scales to large-scale data size

We next quantitatively assessed the scalability and efficiency of Fugue in relation to sample size. The down-sampled *HCA* was applied for the analysis. As shown in Supplementary Table 3, Fugue used fixed memory irrespective of data size as it only loads a mini-batch of sample at each training iteration. For example, the memory consumption was 2.3 Gb for a batch size of 512 for all datasets. The time consumption increased linearly with the number of cells, from about 0.4 hours for 0.1 million cells to 75 hours for 18 million cells.

Furthermore, we compared the peak memory usage and run time. Fugue consumed the least memory among the methods aforementioned (Supplementary Figure 27). However, the gap between Fugue and other methods is more significant on million-scale dataset (Supplementary Figure 27).

## Discussion

In this study, we developed Fugue to address the batch effect for large-scale single-cell transcriptomes with a

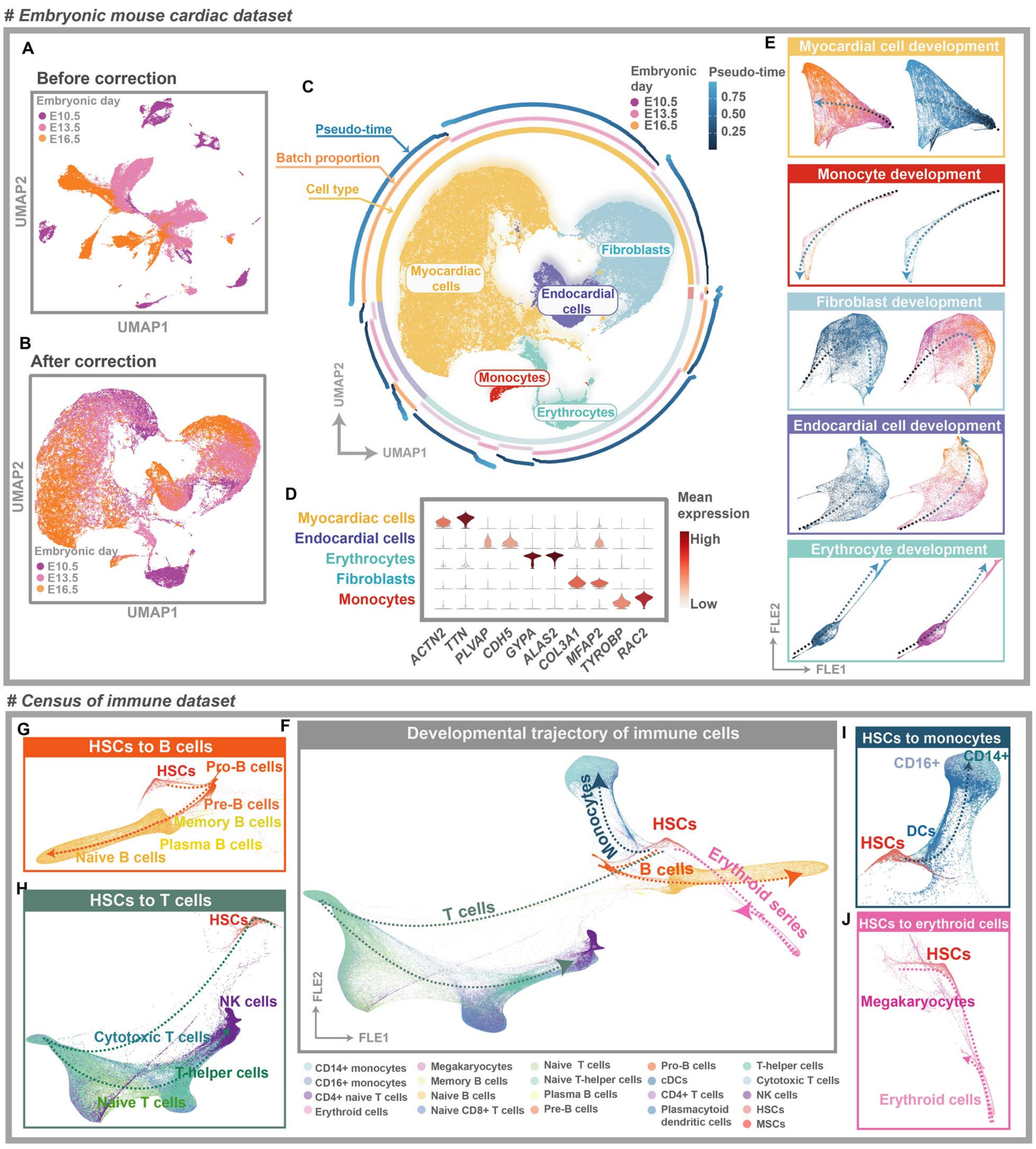

**Figure 5.** Joint analysis of Fugue on batch effect removing and gene expression trajectory recovering during cell development. (**A**–**B**) UMAP plot of cells from the *embryonic mouse cardiac dataset* before (**A**) and after (**B**) Fugue integration. Cells are colored by batch. (**C**) UMAP plot of the *embryonic mouse cardiac dataset* integrated by Fugue, colored by cell clusters. The surrounding circle plot from inner to outer shows the cell types, batch labels and pseudo-time scores of the cells. (**D**) Violin plot deciphering expression levels of cell type markers across cell clusters. The color intensity reflects the average expression level within each cluster. (**E**) FLE plot revealing time course trajectories of cardiac development across different cell types. Arrows indicate inferred cell state transition directions from early to late pseudo-time. (**F**) FLE plot revealing cell state transition directions from HSC to all main blood lineages. (**G**–**J**) FLE plots of the main development trajectory from HSC to B cell, T cell, monocyte and erythrocyte, respectively. B cell series (**G**) were separation from HSCs toward B cell progenitors, precursors of B cells and matured naïve B cells. B cells also differentiate into mature B cells, plasma cells and memory B cells. T cell series trajectory (**H**) was started from HSCs, followed by naïve T cells and finally mature T cells and NK cells. Monocytes series trajectory (**I**) was started from *HCA* and transferred into DCs and CD14+ and CD16+ mature monocytes. Erythrocyte series (**J**) differentiates from HSCs to megakaryocytes and erythroid cells. Pro, progenitor; Pre, precursor; cDCs, classical dendritic cells; MSCs, multipotent progenitor cells.

simple and effective solution. Fugue could be deployed as a scalable method to integrate single-cell transcriptomes of any magnitude with fixed memory. We demonstrated the advantage of Fugue by integrating millions of single-cell transcriptomes.

Fugue was trained in a stochastic manner. Only a mini-batch of samples is required to load into memory during training. Therefore, it scales efficiently on different magnitude of datasets. The memory consumption of Fugue depends on the mini-batch size but not the data size (Supplementary Figure 27). We demonstrated the robustness of Fugue by integrating all available single-cell transcriptomic data from *HCA* repository. Three datasets from the *HCA* repository were utilized to demonstrate the effectiveness of data integration of Fugue. Fugue achieved stable performance for cell typing and batch correction on a diverse collection of datasets (Supplementary Figure 13). Fugue thus provides a good trade-off between scalability and accuracy for single-cell integration. Fugue is helpful for integrating continually accumulating large-scale single-cell datasets.

Although immune cell markers have been studied extensively, knowledge might be limited by their definition via a restricted set of organs or cell types. The integrated analysis of atlasing-scale single-cells enables cross-organ comparisons and provides new perspectives for the understanding of marker genes. Based on the reference map of *HCA* repository delineated by Fugue, we found that many conventional immune cell markers are expressed in non-immune cell types. For example, the conventional monocyte marker *S100A9* was expressed in esophageal squamous epithelium cells (Supplementary Figure 16), which has been confirmed by previous studies [27, 28]. Higher expression level of *SPINK2*, a canonical marker of HSC marker, was observed in epididymal epithelial cells than HSCs (Supplementary Figure 16). The enrichment of *SPINK2* in epididymal tissue was confirmed in a previous report [29]. As it is a super large-scale combinatorial analysis of multiple datasets, Fugue has the potential to illustrate subtle changes among different biological status and may shed new light on biological discovery.

The mainstream batch-correction methods use MNNs to identify paired cells and subsequently use these paired cells as local anchors to help batch correction. However, the performance of these methods highly depends on the qualities of MNN. ComBat and Pegasus correct for batch effects via generalized linear model, which are not necessarily justified in modeling complex biological processes. scVI uses autoencoder for learning latent representation of expression profiles and assumes Gaussian distribution of the latent representations. This assumption might introduce the over-regularization problem and compromise its performance. In contrast, Fugue assumes no prior assumption and learns batch information from data. Theoretically, Fugue is more flexible and appropriate in modeling batch effects, especially in scenario where violation of prior assumption assumed by previous methods is not negligible.

Fugue could be improved in several aspects. First, as an artificial intelligence model, the black-box nature of the approach is a limitation that should be resolved [30, 31]. We explored the batch embedding matrix and found that samples from the same database have higher similarity of batch embeddings. It brings insights into the interpretability of batch information learned by Fugue. Second, as an unsupervised learning model, hyperparameter tuning might influence the performance of Fugue to some extent [16]. In our analysis, we proved the stability of Fugue to data augmentation (Supplementary Figure 5). Besides, Fugue is unable to automatically perform batch detection and correct for batch effect. The batch information must be explicitly embedded into the model by users.

In summary, we present Fugue, a simple and efficient deep learning model for super large-scale single-cell transcriptome integration. We anticipate that Fugue will be helpful in transforming growing scale of single-cell transcriptomes into biological knowledge.

## Methods

### Batch embedding

The key idea of batch-effect removal is to decouple biological signals from nuisance factors of batch effects. The batch embedding matrix is used to capture technical noise of different batches. Instead of using a prespecified and constant matrix, we initialized the *BE* matrix as trainable parameters in that it could be updated iteratively via back-propagation. Each batch is mapped into a row vector of *BE* matrix. The vector representations are identical for cells from the same batch. The input to the deep neural network $X$ was constructed by adding *BE* matrix to expression matrix ($E$), i.e. $X = BE + E$. For the purpose of point-wise addition between *BE* and $E$, the column dimension of matrices *BE* and $E$ must be identical.

### Network architecture and training

We used DenseNet architecture [18] as feature encoder to learn expression embedding of single cells. The DenseNet has 21 layers that are consisted of four dense blocks. The DenseNet architecture is featured by concatenating all the outputs from preceding layers as input for the next layer to make feature transmission more efficient. We replaced convolutional layer of the DenseNet with linear layer to make it able to process gene expression matrix. Self-supervised learning with momentum contrast [16] was adopted to train the feature encoder. We applied multi-layer perceptron as project head, which was demonstrated to be beneficial for contrastive learning [16]. The use of deep neural network as feature encoder makes it possible to learn nonlinear feature representations of the input gene expression profiles.

Here we adapt contrastive learning for feature encoder development. Contrastive learning allows feature encoder to learn representations in a label-free manner [17]. For a given integrated input $I_{cell}$, a feature encoder represents it as $C_q = f_q\ (I_{cell})$, where $f_q$ is *a* query encoder network and $C_q$ is a query sample. A key encoder network $f_k$ encode the noise-adding view of the input $I_{cell+}$ *as* $C_{k+}$ (likewise, $C_{k+} = f_k\ (I_{cell+})$). One cell $C_q$ and its noise-adding view $C_{k+}$ form a positive pair and assemble with a different cell $C_{k-}$ *to* form a negative pair. The InfoNCE loss is optimized through learning the same representation of the positive pairs and dissimilar representation of negative pairs:

$$L_{C_q, C_{k+}, \{C_{k-}\}} = -\log \frac{\exp\left(C_q \times C_{k+}/\tau\right)}{\exp\left(C_q \times C_{k+}/\tau\right) + \sum_{k-} \exp\left(C_q \times C_{k-}/\tau\right)},$$

where $C_{k-}$ denotes a queue of the negative samples. The queue $\{C_{k1-}, C_{k2-}, \ldots, C_{kn-}\}$ represent a sampled subset of all cells that does not contain $I_{cell}$. The queue was updated on-the-fly, in which cells progressively replaced. The current mini-batch of cells en queue and the oldest mini-batch de queue. $\tau$ is a temperature hyper-parameter and was set to 0.2. In this manner, the feature encoder can learn representation of inputs by contrastive loss.

The parameters of query encoder $\Theta_q$ were updated by back-propagation; the parameters of key encoder $\Theta_k$ were updated according to $\Theta_q$:

$$\theta_k \leftarrow m\theta_k + (1 - m)\,\theta_q,$$

where $m$ stands a momentum coefficient, which was set to 0.999. We trained the network at a learning rate of 0.01. The training was ended until loss did not improve over a specified number of epochs (Supplementary Table 4).

The network was trained through mini-batch stochastic gradient descent algorithm [32] with a weight decay of 1e−4. The convergence speed of deep learning model is affected by batch size [33]. For that we integrated thousands to multi-millions single-cells, we set the size of mini-batch from 16 to 256, which was dependent on the volume of training data (Supplementary Table 4).

At the stage of feature extraction, we applied the developed feature encoder $f_q$ as feature extractor. Only expression matrix $E_{cell}$ was provided to $f_q$:

$$F_{\mathbf{cell}} = f_q\left(E_{cell}\right),$$

where $F_{cell}$ is the feature representation of the single-cell transcriptome.

## Data sources

### Simulation dataset

We simulated a total of 30 000 single-cell read counts using Splatter package [34]. The resultant *simulation dataset* contains three cell types; each cell type consists of five batches (Figure 2A). Each batch contains 2000 genes with a differential expression factor of 0.4. To estimate the performance of Fugue on batch-specific cell-type detection, we manually removed cell type 1 from batches 2–5 and maintained them in batch 1 (Supplementary Figure 1). We named the resultant dataset as *simulation_rm dataset*.

### *Cell line dataset*

This dataset consists of the three batches, including 'Jurkat' batch, '293 T' batch and the 50/50 mixture of both cell lines [35]. The dataset is composed of 9531 single cells generated by 10× 3′ protocol. For mixture cell lines, cells were clustered with Louvain algorithm based on Scanpy pipeline. Cell clusters with high expression of *XIST* were annotated as '293 T', whereas others as 'Jurkat'.

## Human PBMC dataset

The data included two sub-datasets of PBMC [36]. This dataset contains 42 667 single cells, which could be grouped into B cells, CD4+ T cells, CD8+ T cells, NK cells, monocytes, megakaryocytes and dendritic cells (DCs). The cell labels were provided by the original publication [36].

### *Human Cell Atlas*

We downloaded single-cell data from *HCA* portal [2] on 16 July 2021. Fifty-three projects (C1–C53) following *HCA* data processing pipeline were collected (Supplementary Table 1). A total number of 373 samples with available cell numbers >1000 were maintained (Supplementary Table 1). We filtered out cells with <500 expressed genes. We used the sample labels as batch information given that technical labels are not always available for every dataset. A total of 3424 607 cells passed quality control were utilized to construct the reference map of *HCA* (Supplementary Figure 6).

The following projects were selected for the assessment of dataset alignment and biological significance preservation performance of the reference map. The first dataset was the *census of immune project* (C1). The *census of immune project* consists of two batches that can be referred to as cord blood (C1.0) and bone marrow (C1.1). The two batches contain immune cells from diverse development statuses. We downloaded cell type labels from *HCA* repository on 28 August 2020. The *lung dataset* consists of C30.1 and C47. Both C30.1 and C47 came from lung tissue and have similar cell types [20, 37]. The *brain dataset* consists of C19, C28 and C32, which contain cells from different subsections of brain tissue with overlapping cell types among each other [19, 38, 39]. The *embryonic mouse cardiac dataset* consists of C18 and C20. C18 contains mouse cardiac cells from embryonic state of E10.5 and E13.5. C20 includes mouse cardiac cells from embryonic state of E16.5. Only healthy embryos were taken into account in this analysis. Since the original author of C18 denotes batch effect exists between cells from E10.5 and E13.5 mouse [23], the embryonic periods were employed as batch labels for the benchmarking methods.

## Data prepossessing

We applied scanpy (version 1.7.0) for data preprocessing. We used 'highly_variable_genes' function with the default parameters to identify highly variable genes. A total of 1959 and 2085 highly variable genes (HVGs) were selected from the *cell line* and *PBMC* datasets, respectively. For *HCA*, 14 550 genes shared among datasets were selected.

For all of the aforementioned datasets, we normalized the count matrix to counts per million normalization (CPM) and took logarithmic transformation (i.e. log2(CPM + 1)). Subsequently, the expression of each gene was scaled by subtracting its average expression then divided by its standard deviation. The scaled expression matrix was applied as inputs for the model.

## Benchmark methods

We benchmarked the performance of Fugue with 10 state-of-the-art batch correction methods, including Seurat V3, BBKNN, Harmony, Scanorama, INSCT, iMAP, scVI, CLEAR, ComBat and Pegasus.

### Seurat V3

Seurat V3 [11] is a batch correction method based on MNN searching. It assumes that confounding variables have uniform effects on all cells of a dataset. Seurat V3 identifies MNNs across batches based on canonical correlation analysis. Based on the detected mutual neighbors, Seurat V3 integrates different batches into a conserved low-dimensional space through nonlinear normalization, giving rise to a low-dimensional matrix as feature representation of the integrated single cells.

### BBKNN

BBKNN [10] is built upon the assumption that at least some cells of the same type exist across batches, and the differences between the same cell type across batches caused by batch effects are less than that within batches. BBKNN finds k-nearest neighbors for each cell per batch, resulting in an independent pool of neighbors for each cell. BBKNN subsequently corrects for batch effect through merging neighbor sets via UMAP algorithm [40]. A weighted neighborhood graph among cells is ultimately returned by BBKNN for downstream analysis.

### Harmony

Harmony [12] assumes to align batches and maximize the batch diversity within each cell cluster. Principal components analysis is leveraged to construct a low-dimensional space for cells. Then two complementary steps are alternately iterated until convergence. First, soft clustering is applied to assign cells to specific cell clusters. It uses maximum diversity clustering as a regularization to penalize the clusters with large differences in batch proportions. After clustering, cluster-specific centroids are applied as linear correction factors to fit linear adjustment functions. Each dataset is corrected through the average the linear factors. Harmony returns a low-dimensional embedding space for the integrated datasets.

### Scanorama

Scanorama [9] integrates cells according to the concept of panoramic stitching, which identifies images with overlapping content and merges them into a larger panorama. Scanorama initializes cells in a low-dimensional embedding space constructed with singular value decomposition. Then, an approximate nearest neighbor searching algorithm is applied to find similar cells among batches. Finally, a low-dimensional cell embedding space free of batch effect is constructed by Scanorama.

### INSCT

INSCT [41] is constructed based on maximizing the distance of dissimilar cells (negative pairs) and minimizing the distances of similar cells (positive pairs). The transcriptome similarity is defined based on MNNs and k-nearest neighbors. INSCT builds batch-aware triplet neural networks for cell representation learning via distance comparisons. A triplet is consisted of an anchor, a positive sample and a negative sample. A positive pair is sampled from two different batches, whereas a negative pair comes from the same batch. The triplet neural network learns to minimize the distance between positive pairs and maximized the distance between negative pairs. The network subsequently learns a data representation that overcomes batch effects.

### iMAP

iMAP [14] is constructed based on deep representation learning and MNNs searching. iMAP firstly constructs an autoencoder to learn low-dimensional representations of cells. Then a generative adversarial network is trained with paired mutual neighboring cells to remove batch effects. The two models are trained conjugately. After model training, the autoencoder can represent cells in a low-dimension space disentangled from batch effect.

### scVI

scVI [13] is a variational autoencoder for representation learning of expression profile accounting for batch effects. The variational autoencoder is constructed as a hierarchical Bayesian model that maps the latent variables to the parameters of the Gaussian distributions. The mapping is conditioned on the batch variable of each cell. The autoencoder can be applied to reconstruct the cells in a latent space conditioned on the batch information. The output of scVI is a low-dimensional representation of cells free of batch effect.

### Clear

CLEAR [15] is a contrastive learning model based on distance measuring between expression profiles of cell pairs. CLEAR eliminates batch effect via technical noise learning of transcriptomes. Batch annotations are not necessary for CLEAR. It integrated single cells through minimize the distance of the noise-adding views of the same cells and maximize the distance of different cells. Low-dimension representations of cells eliminating batch effect are present after model training.

### Combat

Combat [42] is a simple linear regression model for batch correction. It is originally developed for bulk expression transcriptomes and has been applied to single cells in recent years [6]. Combat performs batch correction based on Bayesian linear regression. It fits batch effect based on the mean and variance of expression. ComBat returns a corrected gene expression matrix for single cells.

### Pegasus

Pegasus [43] is also a linear regression model for single-cell batch correction. Pegasus detects batch information based on linear regression. For each gene, the classical location and scale liner regression is applied to fit both additive and multiplicative batch effects. A batch-adjusted gene expression matrix is returned by Pegasus.

All methods were performed with the default parameters (see Supplementary Table 5 for detailed information) throughout the study. Seurat V3 ran out of memory on our server (maximum memory: 256 Gb) for dataset with >150 000 cells, and therefore, it was not evaluated on dataset >150 000 cells. iMAP was not evaluated on dataset >100 000 cells due to excessive graphic processing unit (GPU) memory usage. The benchmark methods failed to integrate all data from HCA repository on our server. For the *census of immune project*, cord blood and bone marrow were utilized as batch information. We employed different cohorts as batch information for the *lung* and *brain datasets* and embryonic development periods as batch information for the *embryonic mouse cardiac dataset*. We provided these methods with explicit batch information because it's the general configuration and suitable for these methods [8–13].

### Evaluation functions

We employed kBET acceptance rate [44] for the assessment of batch effect through *Pegasus* package [43]. The kBET acceptance rate measures whether batches are well mixed in the local neighborhood of each cell. The resulting score ranges from 0 to 1, where a higher score means a better mix. We computed kBET scores based on each cell type and used the average score to evaluate the degree of batch mixing.

The ARI score was applied to evaluate batch correction method in terms of cell-type mixing. The ARI score measures the percentage of matches between two label lists. The resulting score ranges from −1 to 1, where a high score denotes that the data point fits well in the current cluster. We used the Louvain community detection algorithm implemented in 'tl.louvain' of Scanpy package (version 1.7.0) for cell clustering. In our study, Louvain algorithm would generate much more cell clusters than real cell types when the resolution was 1 and far fewer when the resolution was 0.001. Thus, we set the resolution parameter range from 1 to 0.001 with a step of 0.001 and computed ARI score with *sklearn* package for each step. The assessment would be early stopped if ARI score was less than zero. The maximum ARI score was employed as the final evaluation index.

For the combined assessment of batch correction and cell type separation, we defined F1 score, which is the harmonic mean of kBET and ARI:

$$F1 = \frac{kBET \times ARI}{kBET + ARI},$$

a higher F1 score indicates a better trade-off of batch correction and cell type separation. We used F1 score to evaluate the *cell line*, *PBMC* and *census of immune* dataset. Since cell type labels were not provided by the *lung* and *brain* datasets and ARI score cannot be calculated, we used kBET score to evaluate these two datasets. On account of BBKNN cannot give the corrected feature representation, we calculated the evaluation indexes in UMAP embedding space. The embedding was computed with the default parameters based on the same random seed through *umap-learn* package (version 0.4.6).

We used the area of polygon in radar plot of all evaluated datasets to quantify the methods performance. We benchmarked Seurat only on the *cell line, PBMC, lung* and *brain* dataset, since it ran out of memory on the *census of immune* dataset. We benchmarked iMAP on the *cell line*, *PBMC* and *lung* datasets, since it ran out of memory on the *brain* and *census of immune* dataset.

## Evaluation of the effect of data augmentation on cell identity performance

Two data augmentations were applied in Fugue, including gene shuffling and random zero out. The concept of gene shuffling is that we randomly select two sets of gene expression values for each cell and replace one set of values with another. The concept of random dropout is that we randomly replace some gene expression values with zeros.

We evaluated F1 score with different dropout and random shuffle rates on three datasets of different scales, i.e. the *cell line dataset*, $n = 9531$, *smilulation dataset*, $n = 30\ 000$ and *census of immune dataset*, $n = 309\ 835$. We assessed 20 combinations of dropout (0%, 10%, 20%, 30% and 40%) and shuffle rates (20%, 30%, 40% and 50%). To save computational cost, the *census of immune dataset* was randomly downsampled to 10% of the data ($n = 30\ 984$).

### Cell marker inferring and cell type identification

We constructed a new deep neural network (denoted as $F$: $R^n$ -> $[0,1]$) by freezing the parameters of the trained encoder and adding a single linear classifier at the end of it. The classifier was trained for cell cluster prediction. We used the importance score calculated by integrated gradient algorithm [45] as the surrogate metric for the impact of each gene on classification output. In specificity, the integrating gradient algorithm calculates the important score of the $i^{th}$ gene as the gradient of *F*(*x*) along the $i^{th}$ dimension, which is defined as:

$$\mathrm{In}tegratedGrad_i(x) ::= (x_i - x_i{}') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha\left(x - x'\right)\right)}{\partial x_i} d\alpha$$

The *x* and *x*' are the actual and baseline expression levels, respectively. We set *x*' to 0. A higher importance score represents a more significant impact of gene for the specific cell cluster. We manually annotated cell types according to genes with the highest importance scores.

### Benchmark of runtime and memory usage

The peak memory and computing time usage were evaluated on a Linux server with 256 GB memory, 2.2 GHz Intel Xeon E5–2699 v4 processor and two NVIDIA GPUs with GTX 1080Ti graphics cards. We used the downsampled *HCA* repository ($n = 100\ 000$, 500 000, 1000 000, 2000 000, 5000 000, 10 000 000, 15 000 000) to evaluate the computational cost of Fugue on large-scale single cells. The *cell line, PBMC, lung* and *census of immune datasets* were carried out to benchmark Fugue with other methods. Seurat V3 could not scale to the *census of immune dataset*. iMAP ran out of GPU memory on the *brain* and *census of immune* datasets and was therefore not included in the assessment. All datasets were processed as described in the 'Data preprocessing' section. We did not include data preprocessing in runtime benchmarks.

### Statistics

Cosine similarity was applied to represent the similarity of batch embeddings. We performed *t*-test on pseudo-time scores of different embryonic days. *P*-values were adjusted based on false discovery rate. Polynomial regression was used to fit the pseudo-time and expression level of cell differentiation markers.

### External software

Louvain community detection algorithm implemented in *Scanpy* package (version 1.7.0) was applied for cell clustering. We applied UMAP algorithm to visualize cells in a two-dimensional space if unspecified. UMAP failed on 3424 607 cells after 72 hours; thus, *t*-Distributed Stochastic Neighbor Embedding algorithm from *FIt-SNE* package (version 1.2.1) was utilized to construct a global view of *HCA* embedding space. We set the exaggeration coefficient to 4 and the perplexity coefficient to a list of 30, 36, 42 and 48. FLE from Pegasus package was applied for trajectories inferring.

**Key Points**

- Fugue achieves favorable data alignment and cluster preservation performance comparable to state-of-the-art batch correction methods.
- Fugue aligns precise immune cell subtypes from heterogeneous tissues of the *Human Cell Atlas*.
- Fugue reserves time course developmental trajectories across batches.
- Fugue can be deployed as a scalable deep learning model to integrate single-cells of any magnitude with fixed memory.

## Data availability

The data underlying this article are available in the article and in its online supplementary material.

## Author contributions

X.L., K.C. and L.S. designed and supervised the study. X.L. and X.S. wrote the manuscript. X.L., K.C., L.S. and X.S. revised the manuscript. H.S., D.W., M.F., J.H., J.L. collected the data. X.S., Y.Y., M.Y., Y.L. processed the data. X.S., X.L., K.C., L.S., Y.Y., M.Y. and Y.L. interpreted the results. All authors reviewed and approved the submission of this manuscript.

## Acknowledgment

We want to thank all the researchers for their generosity to make their data publicly available.

## Funding

National Natural Science Foundation of China (grant number 31801117 to X.L.); National Key R&D Program of China (no. 2021YFC2500400 to K.C.); Tianjin Municipal Health Commission Foundation (grant number RC20027 to Y.L.).

## References

1. Paik DT, Tian L, Williams IM, *et al.* Single-cell RNA sequencing unveils unique transcriptomic signatures of organ-specific endothelial cells. *Circulation* 2020;**142**:1848–62.
2. Regev A, Teichmann SA, Lander ES, *et al.* The human cell atlas. *Elife* 2017;**6**:e27041.
3. Han X, Wang R, Zhou Y, *et al.* Mapping the mouse cell atlas by microwell-Seq. *Cell* 2018;**172**:1091, e1017–107.
4. Li Y, Ren P, Dawson A, *et al.* Single-cell transcriptome analysis reveals dynamic cell populations and differential gene expression patterns in control and aneurysmal human aortic tissue. *Circulation* 2020;**142**:1374–88.
5. Guo X, Zhang Y, Zheng L, *et al.* Global characterization of T cells in non-small-cell lung cancer by single-cell sequencing. *Nat Med* 2018;**24**:978–85.
6. Luecken MD, Büttner M, Chaichoompu K, *et al.* Benchmarking atlas-level data integration in single-cell genomics. *Nat Methods* 2022;**19**:1548–7105.

7. Leek JT, Scharpf RB, Bravo HC, *et al.* Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat Rev Genet* 2010;**11**:733–9.
8. Haghverdi L, Lun ATL, Morgan MD, *et al.* Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat Biotechnol* 2018;**36**:421–7.
9. Hie B, Bryson B, Berger B. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nat Biotechnol* 2019;**37**:685–91.
10. Polanski K, Young MD, Miao Z, *et al.* BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics* 2020;**36**: 964–5.
11. Butler A, Hoffman P, Smibert P, *et al.* Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat Biotechnol* 2018;**36**:411–20.
12. Korsunsky I, Millard N, Fan J, *et al.* Fast, sensitive and accurate integration of single-cell data with harmony. *Nat Methods* 2019;**16**:1289–96.
13. Lopez R, Regier J, Cole MB, *et al.* Deep generative modeling for single-cell transcriptomics. *Nat Methods* 2018;**15**: 1053–8.
14. Wang D, Hou S, Zhang L, *et al.* iMAP: integration of multiple single-cell datasets by adversarial paired transfer networks. *Genome Biol* 2021;**22**:63.
15. Han W, Cheng Y, Chen J, *et al. Self-supervised contrastive learning for integrative single cell RNA-seq data analysis*. arXiv 2021.
16. Chen X, Fan H, Girshick R, *et al.* Improved baselines with momentum contrastive learning. *arXiv* 2020.
17. He K, Fan H, Wu Y, *et al.* Momentum contrast for unsupervised visual representation learning. *arXiv* 2020.
18. Huang G, Liu Z, Van der Maaten L, *et al.* Densely connected convolutional networks. 2016; arXiv:1608.06993.
19. Welch J, Kozareva V, Ferreira A, *et al.* Integrative inference of brain cell similarities and differences from single-cell genomics. *arXiv* 2018.
20. Habermann AC, Gutierrez AJ, Bui LT, *et al.* Single-cell RNA sequencing reveals profibrotic roles of distinct epithelial and mesenchymal lineages in pulmonary fibrosis. *Sci Adv* 2020;**6**:eaba1972.
21. Kinchen J, Chen HH, Parikh K, *et al.* Structural Remodeling of the human colonic mesenchyme in inflammatory bowel disease. *Cell* 2018;**175**:372, e317–86.
22. Cillo AR, Kurten CHL, Tabib T, *et al.* Immune landscape of viral- and carcinogen-driven head and neck cancer. *Immunity* 2020;**52**:183, e189–99.
23. Hill MC, Kadow ZA, Li L, *et al.* A cellular atlas of Pitx2-dependent cardiac development. *Development* 2019;**146**:1–12.
24. Xiao Y, Hill MC, Zhang M, *et al.* Hippo signaling plays an essential role in cell state transitions during cardiac fibroblast development. *Dev Cell* 2018;**45**:153–169.e156.
25. Andersson LC, Gahmberg CG, Teerenhovi L, *et al.* Glycophorin a as a cell surface marker of early erythroid differentiation in acute leukemia. *Int J Cancer* 1979;**24**:717–20.
26. Levy JE, Jin O, Fujiwara Y, *et al.* Transferrin receptor is necessary for development of erythrocytes and the nervous system. *Nat Genet* 1999;**21**:396–9.
27. Chi ZL, Hayasaka Y, Zhang XY, *et al.* S100A9-positive granulocytes and monocytes in lipopolysaccharide-induced anterior ocular inflammation. *Exp Eye Res* 2007;**84**:254–65.
28. Pawar H, Srikanth SM, Kashyap MK, *et al.* Downregulation of S100 calcium binding protein A9 in Esophageal squamous cell carcinoma. *Sci World J* 2015;**2015**:325721.
29. Bui FQ, Almeida-da-Silva CLC, Huynh B, *et al.* Association between periodontal pathogens and systemic disease. *Biom J* 2019;**42**:27–35.
30. Ghosh A, Kandasamy D. Interpretable artificial intelligence: why and when. *AJR Am J Roentgenol* 2020;**214**:1137–8.
31. Moore JH, Boland MR, Camara PG, *et al.* Preparing next-generation scientists for biomedical big data: artificial intelligence approaches. *Per Med* 2019;**16**:247–57.
32. Li M, Zhang T, Chen Y, *et al.* Efficient mini-batch training for stochastic optimization. *Assoc Comput Mach* 2014;**2014**.
33. Byrd RH, Chin GM, Nocedal J, *et al.* Sample size selection in optimization methods for machine learning. *Math Program* 2012;**134**: 127–55.
34. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol* 2017;**18**:174.
35. Zheng GX, Terry JM, Belgrader P, *et al.* Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 2017;**8**:14049.
36. Kang HM, Subramaniam M, Targ S, *et al.* Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nat Biotechnol* 2018;**36**:89–94.
37. Madissoon E, Wilbrey-Clark A, Miragaia RJ, *et al.* scRNA-seq assessment of the human lung, spleen, and esophagus tissue stability after cold preservation. *Genome Biol* 2019;**21**:1.
38. Agarwal D, Sandor C, Volpato V, *et al.* A single-cell atlas of the human substantia nigra reveals cell-specific pathways associated with neurological disorders. *Nat Commun* 2020;**11**:4183.
39. Jakel S, Agirre E, Mendanha Falcao A, *et al.* Altered human oligodendrocyte heterogeneity in multiple sclerosis. *Nature* 2019;**566**: 543–7.
40. McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction. arXiv 2018.
41. Simon LM, Wang Y-Y, Zhao Z. Integration of millions of transcriptomes using batch-aware triplet neural networks. *Nat Mach Intell* 2021;**3**:705–15.
42. Leek JT, Evan Johnson W, Parker HS, *et al.* The SVA package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics* 2012;**28**.
43. Li B, Gould J, Yang Y, *et al.* Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq. *Nat Methods* 2020;**17**:793–8.
44. Buttner M, Miao Z, Wolf FA, *et al.* A test metric for assessing single-cell RNA-seq batch correction. *Nat Methods* 2019;**16**:43–9.
45. Mukund Sundararajan AT, Yan Q. Axiomatic attribution for deep networks. 2017; arXiv:1703.01365.