# AWGAN: A Powerful Batch Correction Model for scRNA-seq Data

Tianyu Liu[1#], Yuge Wang[2,#], Hongyu Zhao[2,*]
[1]University of Illinois at Urbana-Champaign
[2] Yale University
#: These authors contribute equally to this work.
*: To whom the correspondence should be addressed.

**With the advancement of technology, we can access large-scale, high dimensional and diverse genomics data, especially through single-cell RNA sequencing (scRNA-seq). However, integrative downstream analysis from multiple scRNA-seq datasets remains challenging due to batch effects. In this paper, we focus on scRNA-seq data integration and propose a new deep learning framework based on Wasserstein Generative Adversarial Network (WGAN) combined with an attention mechanism to reduce the differences among batches. We also discuss the limitations of the existing methods and demonstrate the advantages of our new model from both theoretical and practical aspects, advocating the use of deep learning in genomics research.**

## 1 Introduction

The advancement of single-cell sequencing technologies and the vast amounts of data generated have contributed significantly to biological research. One major challenge in single cell data analysis is batch effect that is present in essentially all single cell datasets [1–3]. The definition of batch effect refers to the differences in the distributions of single-cell RNA sequences for cells

of the same tissue due to many experimental factors, including different protocols, instruments, operators, and pre-processing methods [4]. The removal of batch effect is not only important to identify true biological signals but also to facilitate multi-omics analyses across studies.

Two main approaches have been proposed to reduce the batch effects [5]. The first one is to assume that data from different batches follow the same distribution so that we can determine the parameters in the distribution and remove the batch effects in the dataset through statistical methods. The second one is to select one batch as the reference batch and others as query batches and get the batch-effect-corrected data by constructing a mapping from the query batches to the reference batch. Representative methods for the first strategy include Limma [6], Combat [7], Liger [8], and scVI [9]. Representative methods for the second strategy include Mutual Nearest Neighbor (MNN) [10], Seurat v4 [11], Batch balanced kNN (BBKNN) [12], iMAP [13], and Harmony [14].

Even with the many methods available, the task of batch correction still presents many difficulties. The most crucial point is to find a way to perfectly mix cells from the same cell type across different batches and to be able to discern batch-specific cell types [13]. The existing methods have different focuses on the above two points, but the different focuses of some methods can lead to poor results sometimes after batch effect removal (see our discussion part). Therefore, how to achieve a reasonable trade-off between the two remains a challenge in practice.

Moreover, the variety of single-cell datasets and the volume of data is increasing, challenging the assumptions made in some of the batch effect removal methods. For example, some statistical methods assume that single-cell data follow a specific distribution and need to estimate parameters associated with the distribution (e.g. mean and variance). However, this assumption may not hold for many single cell datasets. For example, we cannot assume that the scRNA-seq data of liver cells and blood cells follow the same distribution because these two

tissues are very different biologically. In addition, the accumulating data make it more difficult to estimate model parameters, and develop algorithms with manageable time complexity. As a result, traditional statistical models may not be the best candidates for batch effect removal for large and complex single cell data. Therefore, researchers have been adopting machine learning models based on approximation theory, where there is more flexibility on the distribution of input data [15]. However, there are also issues with currently deployed machine learning strategies. Firstly, the existing machine learning-based correction models are poorly generalized, with varying results on different datasets. Secondly, the machine learning models sometimes generate incorrect results due to the acquisition of datasets, processing, and the choice of model structure, which will be discussed later in this paper. Therefore, there is a need to overcome this limitation. Finally, machine learning-based models tend to be computationally more expensive and time consuming to train [16].

To address the limitations of the existing methods, we propose Attention Wasserstein Generative Adversarial Network (AWGAN) for batch effect removals. This model has a lighter structure, a relatively shorter training time (using a combination of hardware and software acceleration), and better stability and generalization. These benefits are demonstrated through our results.

# 2 Results

## 2.1 Overview of AWGAN

The workflow of AWGAN has three key steps: attention-driven data preprocessing, AWGAN training, and model evaluation (Figure 1):

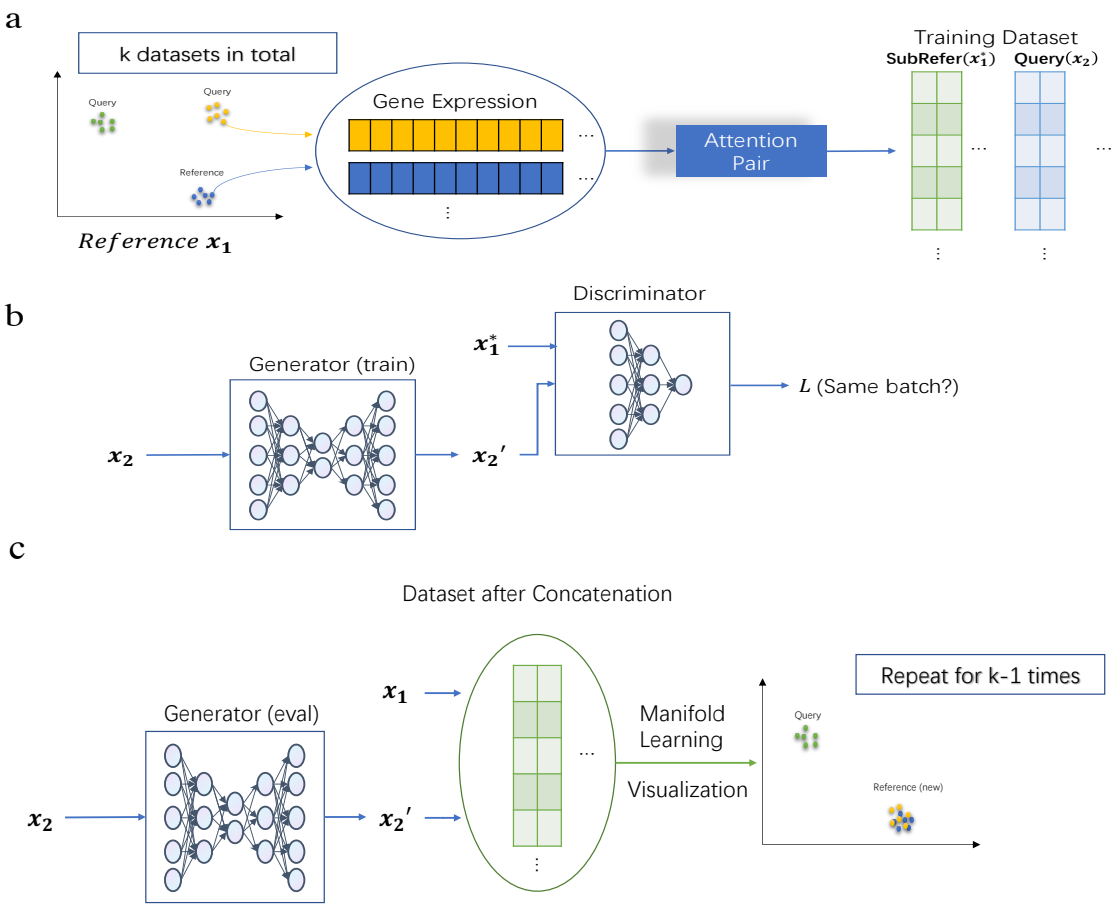


Figure 1: Workflow Chart. (a) This represents how we generate the training set. The attention pair will be transferred into gene expression data pair and fed into the model. (b) Training process for AWGAN. (c) Downstream process for the AWGAN model. The results received from the generator will be utilized for further work shown in Results and Discussion.

In the first step, highly variable genes are selected across batches as common features for data integration. Moreover, the dataset cannot be directly fed into the training model because we need to ensure that our method can preserve the biological information. Therefore, the number of cells in the training datasets needs to be compatible as $\#(SubReferData) = \#(QueryData)$. In this paper, inspired by the attention mechanism in the nature language processing scheme [17], we utilize a similar attention mechanism that can be used to generate a subset of reference data matched to the query data by selecting the nearest neighbor pairs

(see Methods). Since training GAN is very tricky and requires the two sample distributions as similar as possible [18], utilizing matching pairs to generate the training set is important. When there are two batches, we only need to train the model once. When there are $k(k > 2)$ batches in the target dataset, we utilize the incremental matching learning strategy by running the attention algorithm and the model training process $k - 1$ times. After each round, one query dataset will be integrated into the total reference batch, making the reference batch larger and larger. This strategy is adopted because different query batches may have different distributions.

In the second step, after generating the training dataset, we utilize the WGAN [19] with gradient penalty [20] to generate a map that can transfer the query data into distribution similar to the reference data. In addition, to reduce the possibility of model collapse, we also allow the self-adaptive mechanism in finding suitable regularization coefficients for the gradient penalty and making adjustment for the number of training epochs. WGAN is composed of one generator model and one discriminator model. The former model is used to generate data that follow a new distribution from the query data, while the latter model is used to discriminate whether the corrected query data have a similar distribution with the reference data. Moreover, the discriminator can find batch specific cell types and maintain their property. We adopt a strategy of confrontation training to improve the ability of the two models and finally achieve the Nash equilibrium between them [21].

In the last step, we generate an integrated scRNA-seq dataset after removing the batch effect. To visualize the results, we prefer UMAP [22] as a manifold learning choice.Moreover, to quantify the results after removing batch effects, we consider four metrics: ASW [23], kBET [24], LISI [14], and Graph Connectivity [5] for quantitative evaluation of the performance of batch effect removal in this paper.

## 2.2 Small-scale scRNA-seq Datasets

We first evaluated our model against others on datasets with known cell type labels and moderate data size (less than 30,000 cells), which can be used as a standard to compare the effectiveness of different batch effect removal techniques. The datasets we chose in this part are listed as following: Pure Cell Lines (CL) [25], Human Dendritic Cells (DC) [26], Pancreas with removed certain cell types (Pancreas rm) [13] and PBMC3&68K [27] [25], mouse hematopoietic stem and progenitor cells (MHSP) [28] , Mouse Cell Atlas [29] [30], human peripheral blood mononuclear cell (Human PBMC) [25] and Mouse Retina [31] [32]. In order to better demonstrate the superiority of our model, we provided two coordinate representations of two different metrics (ASW [23], and LISI [14]) for each dataset. We define the batch specific cell type discerning task as the cell type optimization task and the mixing of common cells in different batches as the batch optimization task. For the cell type optimization task, higher casw and 1-clisi values indicate better cell type separation. For the batch optimization task, higher 1-basw and blisi and values indicate a better mixing of cells from the same cell type across different batches. The results based on UMAP and ASW/LISI map for the CL dataset, the DC dataset, the Pancreas rm dataset, and the PBMC3&68k dataset are shown in Figures 2-5, and the results of the MHSP dataset, the Mouse Cell Atlas dataset, the Human PBMC dataset and the Mouse Retina data can be found in supplementary file 1 part 1

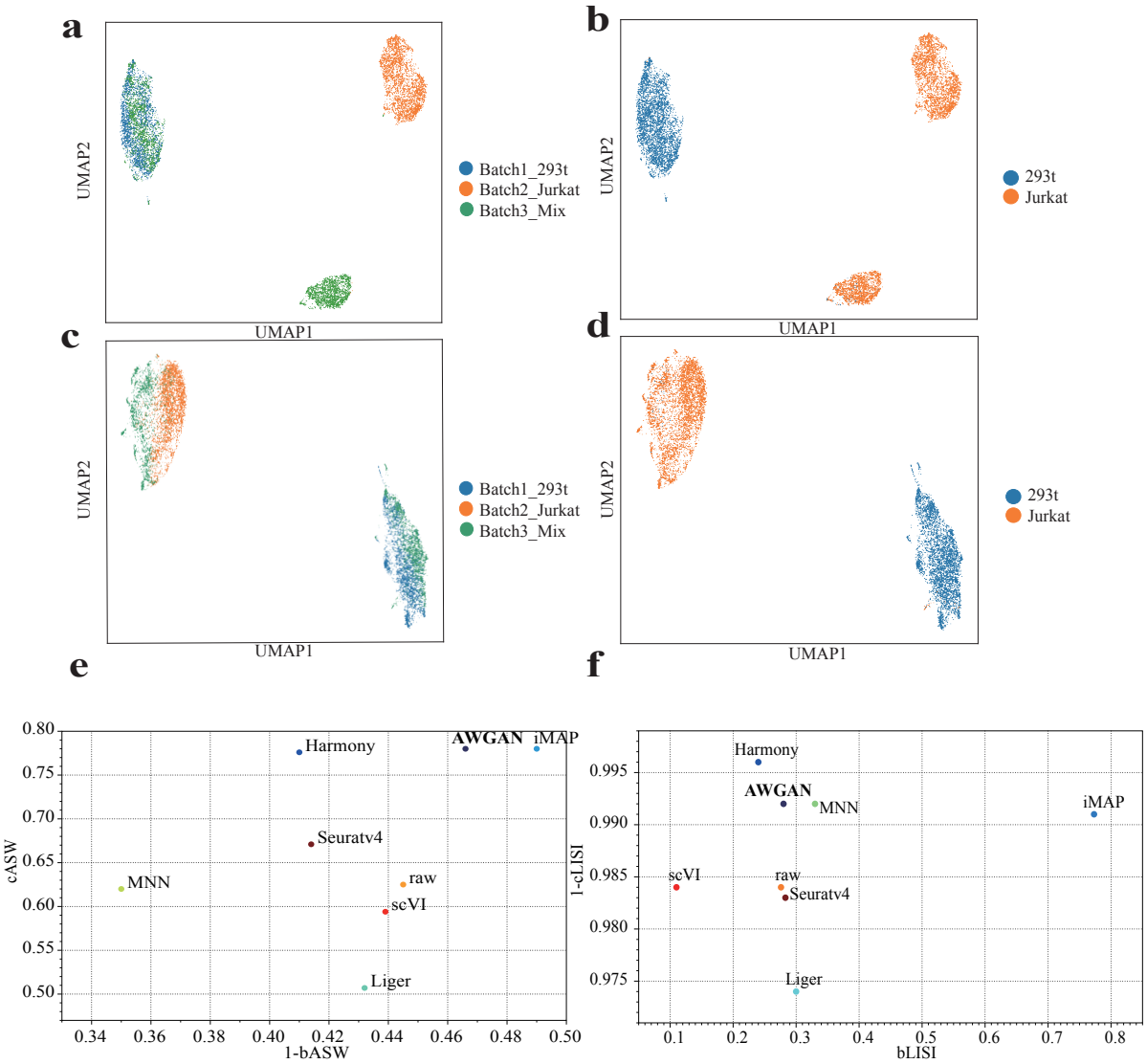


Figure 2: The performance of different batch effect removal methods for the CL dataset. (a) Visualization for the data distribution with batch label before batch effect removal. (b) Visualization for the data distribution with cell type label before batch effect removal. (c) Visualization for the data distribution with batch label after batch effect removal. (d) Visualization for the data distribution with cell type label after batch effect removal. (e) ASW assessments of different batch effect removal methods on the CL dataset. (f) LISI assessments of different batch effect removal methods on the CL dataset.

The CL dataset (Figure 2) is composed of three different batches, including two pure batches "Jurkat" and "293T" containing Jurkat cells and 293T cells only and one batch "Mix" consisting of a 50-50 mixture of Jurkat and 293T cells. We can see from the results that the batch effect is mostly eliminated for the three datasets, and the cells in Mix are mixed into the first two batches. The biological properties are also preserved, as cells in Mix are successfully mapped to their specified cell clusters. We can see that from Figure 2 e that no method performed better than AWGAN in cell type optimization and batch optimization simultaneously except iMAP. AWGAN performed better than Liger, Seurat v4 and iMAP in cell type optimization according to Figure 2 f, suggesting that AWGAN can better discern batch-specific cell types in this dataset. According to Figure 2 e and f, AWGAN only performed worse than iMAP in mixing cells in different batches for both ASW metric and LISI metric, suggesting good performance of AWGAN in this aspect.

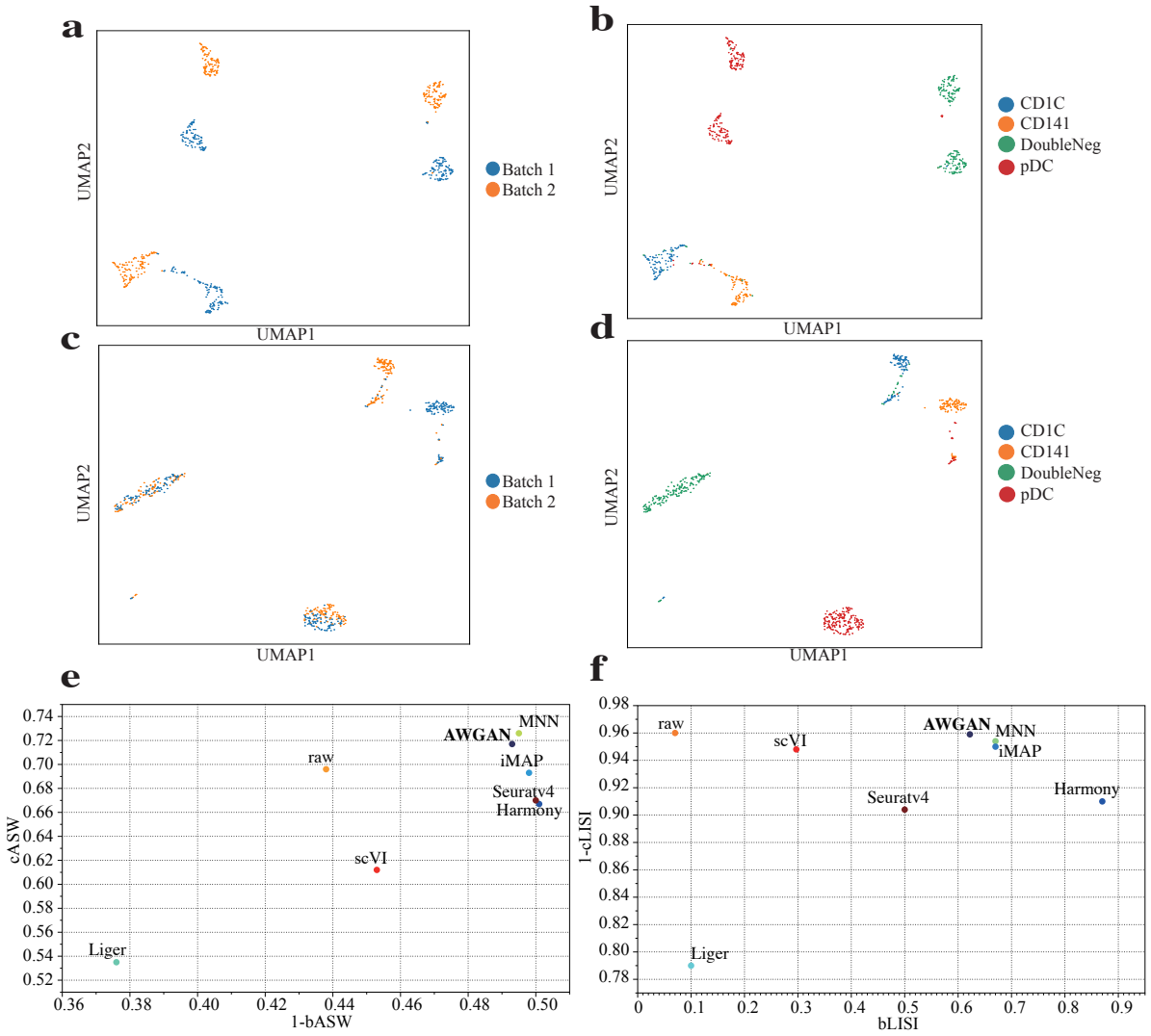


Figure 3: The performance of different batch effect removal methods for the DC dataset. (a) Visualization for the data distribution with batch label before batch effect removal. (b) Visualization for the data distribution with cell type label before batch effect removal. (c) Visualization for the data distribution with batch label after batch effect removal. (d) Visualization for the data distribution with cell type label after batch effect removal. (e) ASW assessments of different batch effect removal methods on the DC dataset. (f) LISI assessments of different batch effect removal methods on the DC dataset.

The DC dataset (Figure 3) contains two batches, including four types of human dendritic cells: CD1C, DC, CD141, and plasmacytoid DC (pDC). In the DC dataset, most of the batch variation was removed by AWGAN in both cell type optimization and batch optimization. Moreover, most of the cells are classified in the correct clusters. However, there still exist a few cells which are mixed incorrectly. This may be caused by the misclassification problem in the original data, because according to Figure 3 b, some cells in the prior batch-correction cluster have cell type labels different from neighbors. According to Figure 3 e and f, AWGAN performs better than most of the benchmark methods. Only MNN performs better than AWGAN on both the batch optimization task and the cell type optimization task with the ASW metric. Under the LISI metric, no method is better than AWGAN for both the batch optimization task and the cell type optimization task. Since this dataset contains batch-specific cell type labels, this experiment demonstrates AWGAN's effectiveness in the cell type optimization task.

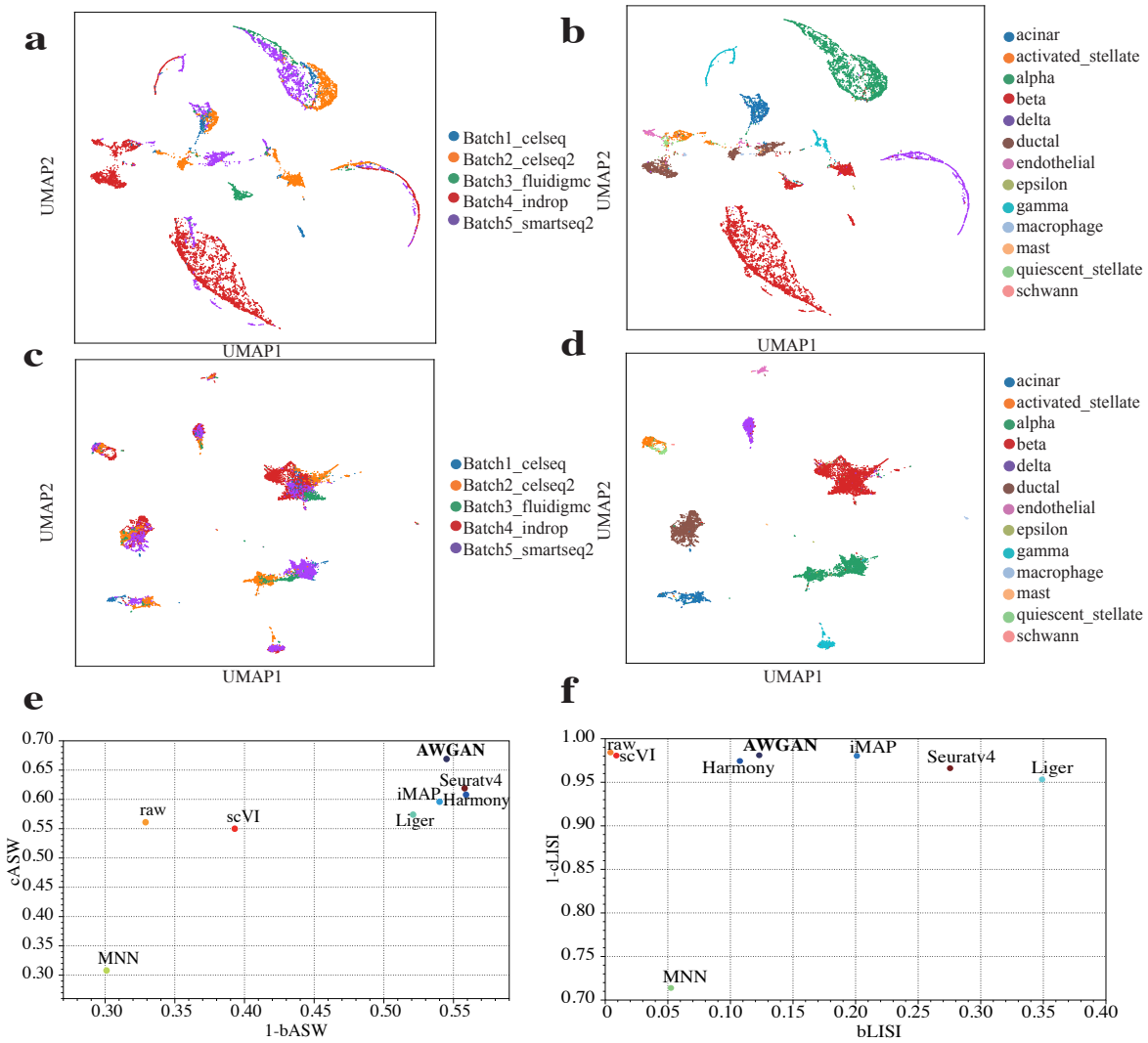


Figure 4: The performance of different batch effect removal methods for the Pancreas rm. (a) Visualization for the data distribution with batch label before batch effect removal. (b) Visualization for the data distribution with cell type label before batch effect removal. (c) Visualization for the data distribution with batch label after batch effect removal. (d) Visualization for the data distribution with cell type label after batch effect removal. (e) ASW assessments of different batch effect removal methods on the Pancreas rm dataset. (f) LISI assessments of different batch effect removal methods on the Pancreas rm dataset.

The Human Pancreas dataset is initially composed of human pancreatic cells from five different sequencing methods. To evaluate the ability of AWGAN on both the mixing of the distribution of common cell types across different batches and discerning the batch-specific cell type, we intentionally removed some cell types from this original dataset. Specifically, we removed acinar and alpha cells from the inDrop batch data, and ductal cells from the CEL-seq batch data. In the end, we renamed the new dataset as Pancreas rm. We can conclude from the visualizations that AWGAN maintains a balance between common cell type mixing and specific cell type discernment across different batches according to Figure 4 c and d. In addition, according to the results in Figure 4 e and f, AWGAN performs well in this dataset based on quantitative results. For example, it achieves the best result in the cell type optimization task evaluated by ASW rate. In the function of preserving cell type information, AWGAN performs better than powerful methods like Harmony, Liger and iMAP.

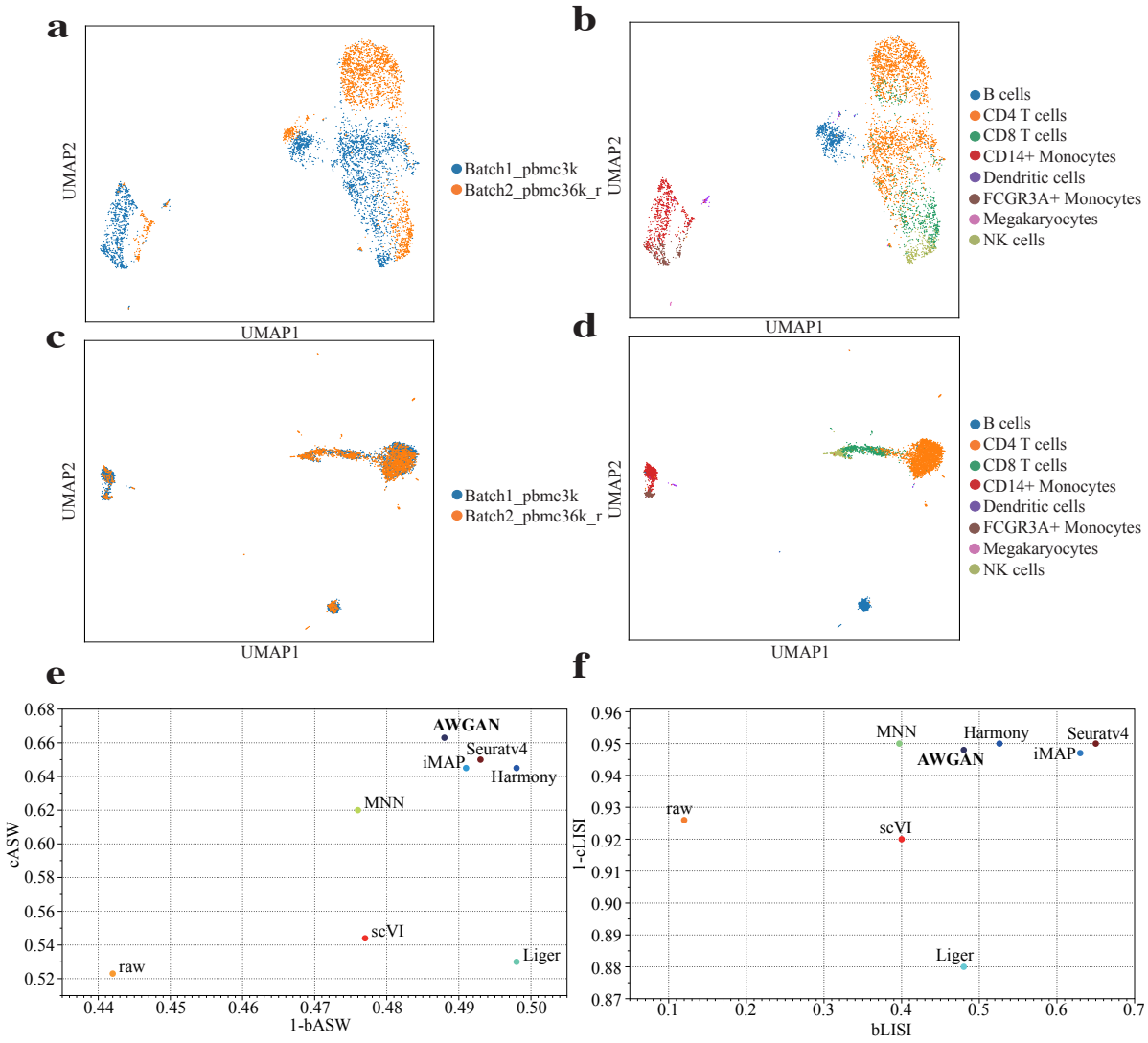


Figure 5: The performance of different batch effect removal methods for the PBMC 3&68K. (a) Visualization for the data distribution with batch label before the batch effect removal. (b) Visualization for the data distribution with cell type label before the batch effect removal. (c) Visualization for the data distribution with batch label after the batch effect removal. (d) Visualization for the data distribution with cell type label after the batch effect removal. (e) ASW assessments of different batch effect removal methods on the PBMC3&68K dataset. (f) LISI assessments of different batch effect removal methods on the PBMC3&68K dataset.

The PBMC 3&68K dataset (Figure 5) two PBMC datasets with different number of cells. For PBMC 3K, we utilized the entire dataset whose number of cells is near 3,000, but for PBMC 68K, we sampled the raw version with 2,000 cells to decrease the workload. The main cell types contained in PBMC are B cells, CD 4/8 T cells, Megakaryocytes and NK cells. According to the steps in the Scanpy document, we first mapped cell type labels from the PBMC 3k dataset to the PBMC 68k reduced dataset by using INGEST [33] and then performed the batch correction. On this dataset, AWGAN still maintains good performance. Because the two batches do not contain batch specific cell types, we reached the batch removal target easier than the previous work. From Figure 5 c and d, we can conclude that AWGAN is able to remove the batch effect in this dataset effectively, and most of the benchmark methods have worse performance based on the results shown in Figure 5 e and f.

We plot running time for different batch correction methods in Figure 6.

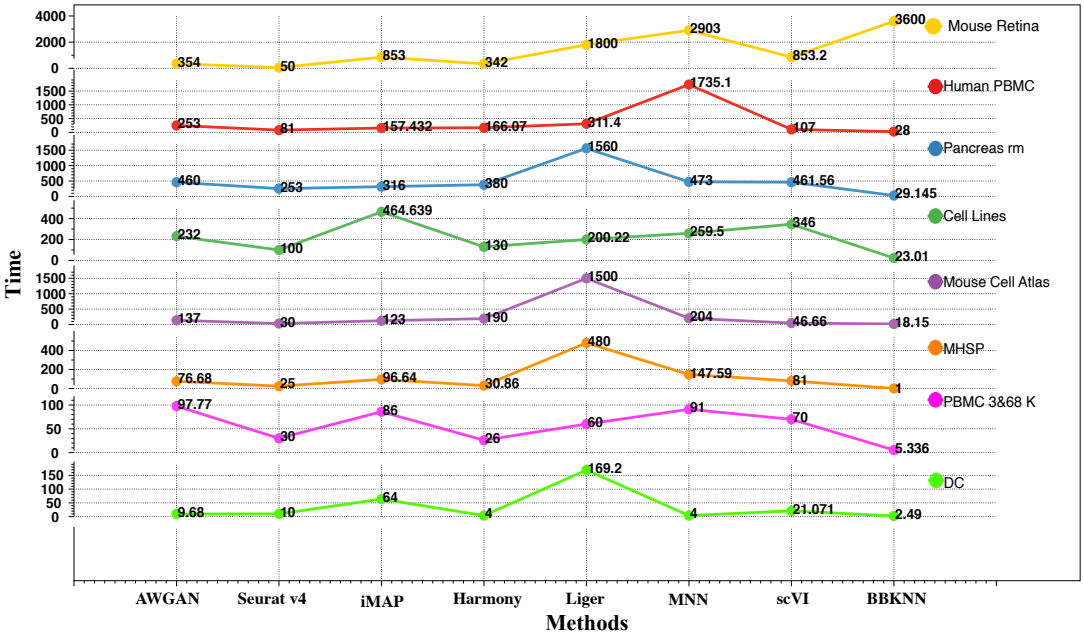


Figure 6: Running time comparisons across different datasets. We ordered the methods by the rank we calculated in Table 2 and ordered the datasets by their size. Methods closer to the left have smaller overall rank in Table 2, and datasets closer to the bottom have smaller number of cells.

It can be seen that MNN and Liger are the two most time-consuming methods in some datasets, while BBKNN and Seurat v4 are the two fastest methods for running on most datasets. This might be because BBKNN and Seurat v4 do not involve large matrix decomposition as Liger does or complex matching learning process as other methods. As for AWGAN, its running time is relatively short in most datasets among all the machine-learning based methods (e.g. iMAP, Harmorny, and scVI).

## 2.3 Large-scale scRNA-seq Datasets

In this part, we consider two scRNA-seq datasets with more than 100,000 cells and investigate the ability of AWAGAN to integrate large-scale datasets. The first dataset is from the Human Cell Atlas [34], which is a large dataset containing more than 700,000 cells. The second dataset is a Mouse Brain dataset [35] [36], which contains over 800,000 cells. The analysis of large-scale dataset can take much space, memory, and time. Our model utilizes a more economical approach to handle large datasets. We show the results for the Human Cell Atlas dataset and the Mouse Brain dataset in this section.

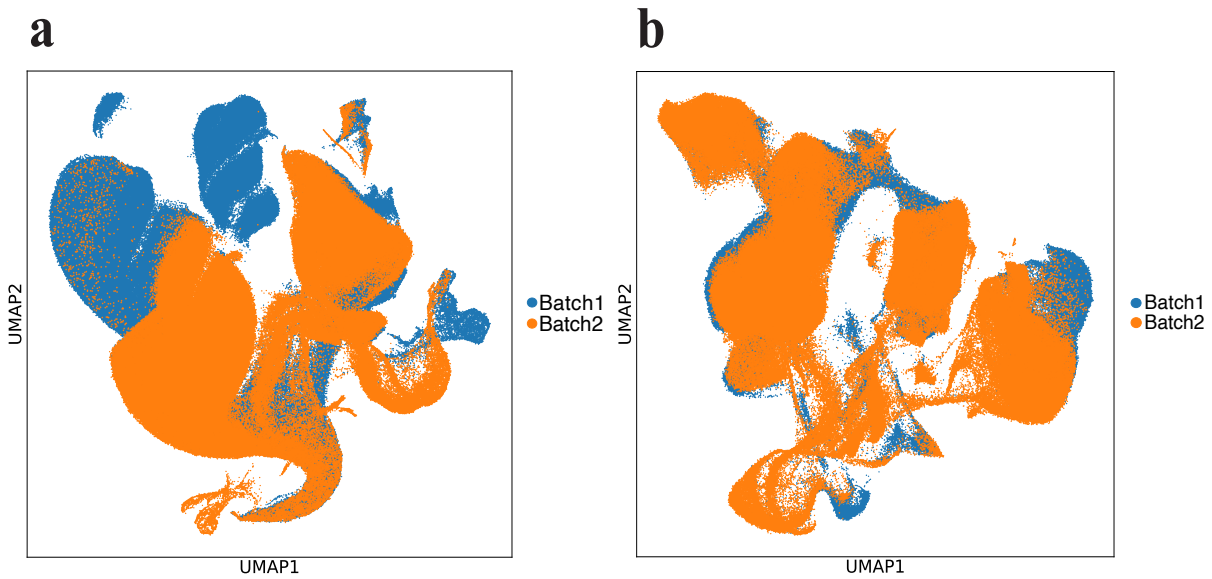


Figure 7: Human Cell Atlas dataset. (a) Visualization for the data distribution with batch label before the batch effect removal. (b) Visualization for the data distribution with batch label after the batch effect removal.

As can be seen from Figures 7 and 8, AWGAN also performs well for large-scale datasets. These two selected single-cell datasets have significant batch effects. Moreover, both of them have over 700,000 cells, making batch effect removal very challenging. To save time, we simplify the attention process for the two datasets. AWGAN is able to remove the batch effect for these two large-scale datasets, demonstrating its scalability to large datasets.

The running time and memory usage of an algorithm are important indicators of its efficiency, especially for models based on deep learning. We compare AWGAN with another GAN-based model, iMAP, from the perspective of resource utilization. The results are listed in Table 1.

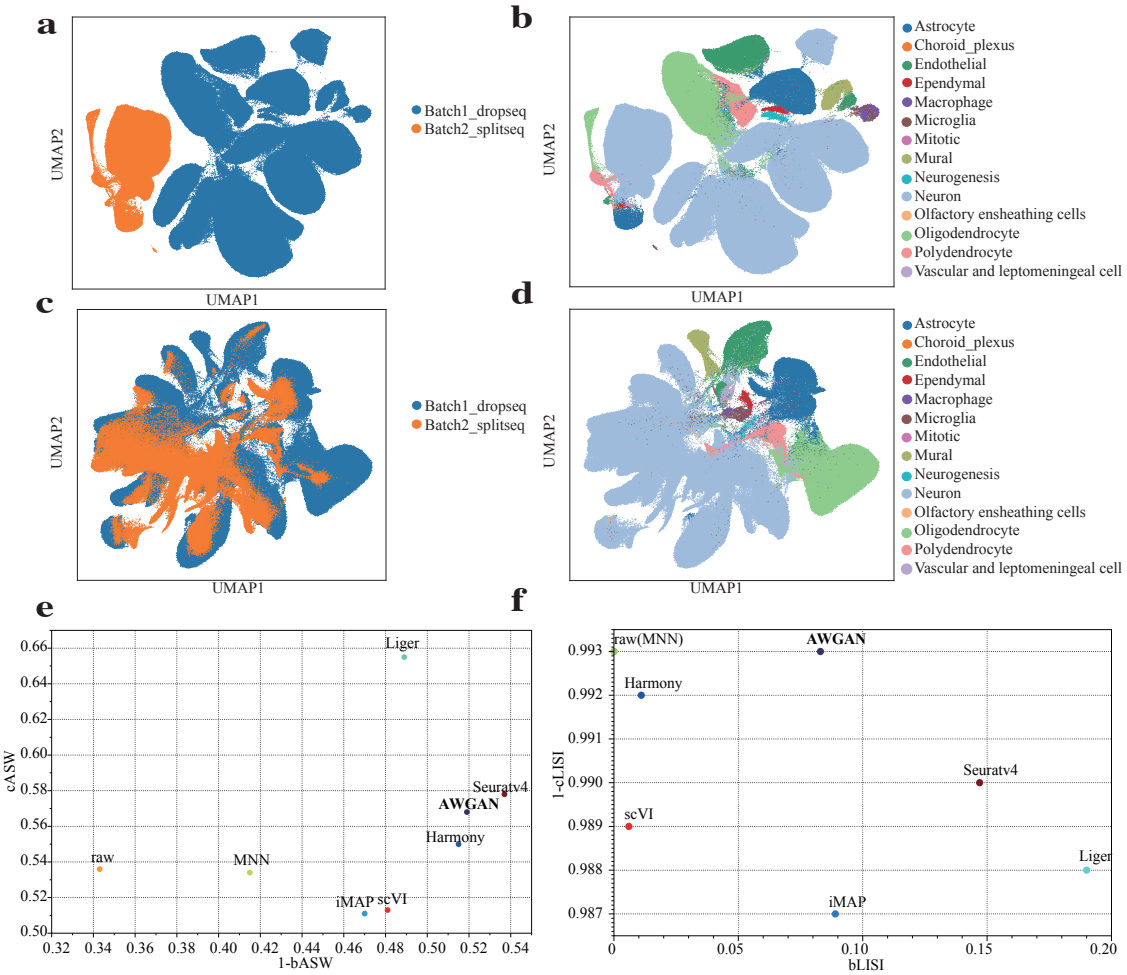


Figure 8: The performance of different batch effect removal methods for the Mouse Brain dataset. (a) Visualization for the data distribution with batch label before batch effect removal. (b) Visualization for the data distribution with cell type label before batch effect removal. (c) Visualization for the data distribution with batch label after batch effect removal. (d) Visualization for the data distribution with cell type label after batch effect removal. (e) ASW assessments of different batch effect removal methods on the Mouse Brain dataset. (f) LISI assessments of different batch effect removal methods on the Mouse Brain dataset.

| Datasets | Model Name | Running Time | Memory Usage |
|---|---|---|---|
| Mouse Brain | iMAP | 7920 s (HPC) | 39 GB |
| | AWGAN | 3500 s | 25 GB |
| Human Cell Atlas | iMAP | 2040 s (HPC) | 30 GB |
| | AWGAN | 1800 s | 20 GB |

Table 1: Comparison on Running Time and Memory Usage between AWGAN and iMAP.

Note that for the Human Cell Atlas, iMAP failed on our cloud platform since in the step of finding the rwMNN pairs, memory usage was larger than available memory (25GB). As a result, running iMAP in a large-scale dataset requires larger memory than AWGAN. However, using AWGAN, our cloud platform can still generate the results because AWGAN only runs the attention step once to use memory more wisely in the computer, compared with iMAP.

## 2.4 Biological Characteristics Discovery based on AWGAN

Colorectal Cancer (CRC) dataset is a scRNA-seq dataset containing tumor-infiltrating immune cells from with 54,285 cells in total [37]. In this part, we took a deeper dive into the functional diversity of AWGAN by looking into the CRC dataset. Results shown in Figure 9 a-d suggest that AWGAN can mostly eliminate the batch effect in the CRC dataset and preserve the biological variation of the dataset. From Figure 9 d, we note that innate lymphoid cells (ILCs) and CD8 T cells are distributed close to each other in the UMAP after removing batch effects. We assumed that the ILCs are transformed from CD8 T cells. To verify this part, in Figure 9 e, based on the heat map clustered by cell type labels, we found the gene expression amount between CD8 T cells and ILCs are very similar. Moreover, from Figure 9 f, a specific UMAP graph for CD4 T cells, CD8 T cells and ILCs, we also discover that some ILC cells are mixed within the cluster of CD8 T cells. Therefore, it is possible that some ILCs may be incorrectly marked. This may be caused by the expression levels of some maker genes being too close or noises in the original data.

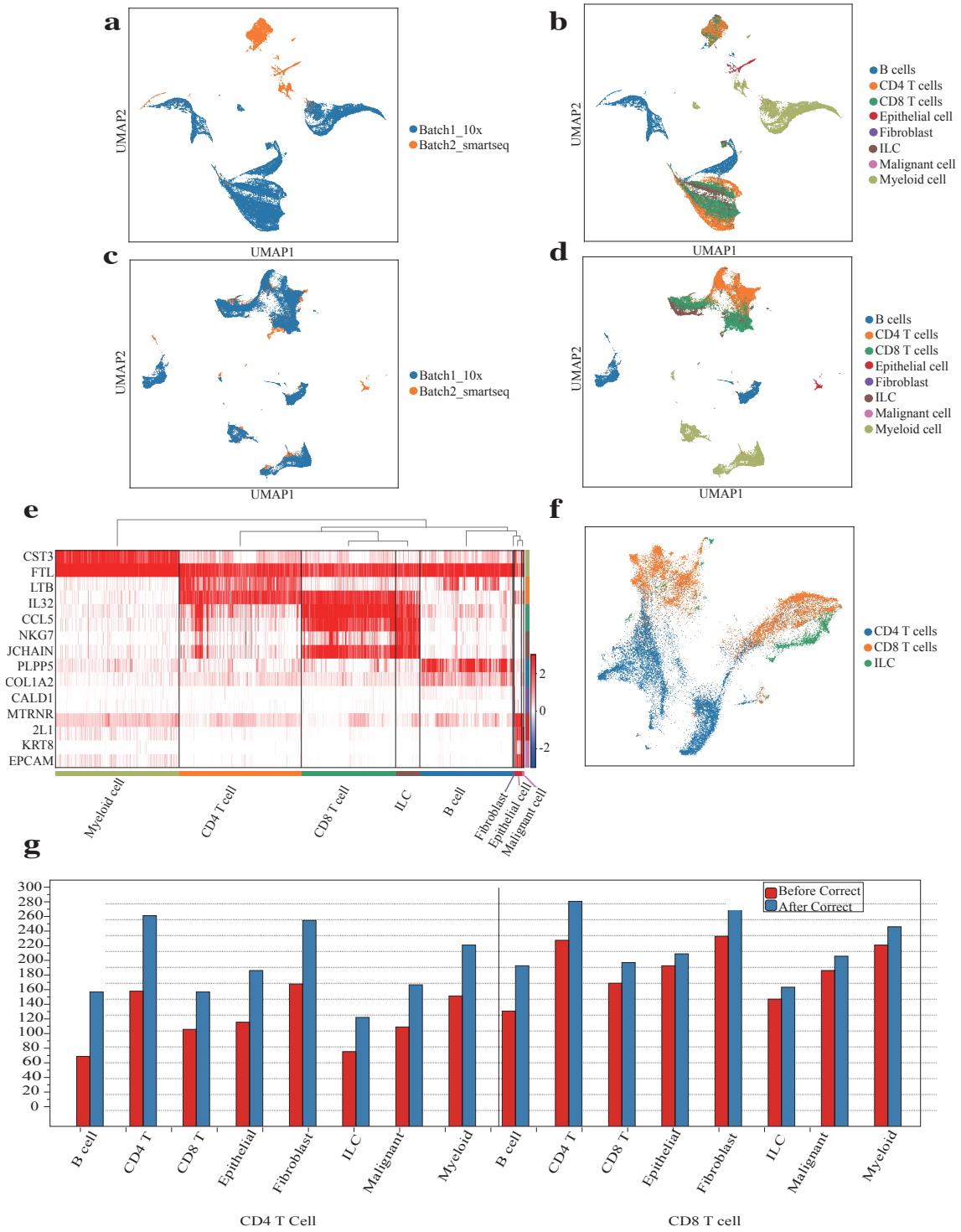


Figure 9: Research on the CRC dataset. (a) Visualization for the data distribution with batch label before batch effect removal. (b) Visualization for the data distribution with cell type label before batch effect removal. (c) Visualization for the data distribution with batch label after batch effect removal. (d) Visualization for the data distribution with cell type label after the batch effect removal. (e) Gene heatmap of the CRC dataset after batch effect removal. (f) Visualization for the misclassification data distribution with cell type label after batch effect removal. (g) Significant ligand-receptor pairs ($p<0.05$) defined between all other cell types and T cells.

We also studied the interactions among different cell types. The two batches of the CRC dataset were sequenced based on different methods, and the two methods had their own characteristics. The 10X data (10x original data) are sparser but have higher throughput in terms of cell numbers, while the Smart-seq2 data measure more accurate gene expression levels [13, 37, 38]. Therefore, relying on the noisy 10X data only may make it difficult to estimate correlation between cells and calculate cell-cell communication scores. AWGAN provides a potential solution to this problem by mapping the 10X data to the Smart-seq2 data. We used CellPhoneDB [39] to calculate cell-cell communication scores and generate visualization results. Based on Figure 5 in supplementary file 1 part 2, 10X data after batch correction (10x corrected data) could find some new ligand-receptor pairs, which could be defined as novel cell-cell interactions. For example, the pair **XCL1-CCR5** between CD4 T cells and B cells could only be uncovered by 10x corrected data but not by 10x original data. Based on Figure 9 g, we found that the number of ligand-receptor pairs of the whole dataset increased after running the AWGAN correction. For more interesting pairs, please refer to the supplementary file 1 part 2.

## 2.5 Benchmark Results Comparison

It is difficult to evaluate and compare the performance of different models from visualization results, so we utilized four metrics to quantitatively evaluate AWGAN and other methods in removing batch effect: ASW, kBET, LISI, and Graph Connectivity. Moreover, since different metrics also have their own limitations, we need to consider results from the four metrics together to judge the model's performance. For example, kBET value cannot accurately measure batch effect removal results in the presence of varying proportions of cell types in different batches [14]. In addition, the value of kBET is also very sensitive to its parameters as shown in supplementary file 1 part 3. Therefore, kBET may not be the only metric to evaluate different models. Instead, LISI, ASW and Graph Connectivity should be considered together with kBET

to provide comprehensive evaluations of the performance of different methods. The details of these four metrics can be found in the Methods section. In order to summarize the overall performance of the $m_{th}$ method for the $d_{th}$ dataset, we consider:

$$TotalRank(d, m) = \sum_{i \in M} Rank(d, i, m) \tag{1}$$

where we sum the rank of this method over different metrics in set $M$ that consists of the four metrics discussed above. Therefore, a smaller $TotalRank$ for a bacth effect removal method implies its better performance. Table 2 shows the results for different methods based on $TotalRank$.

| | **Total Rank Summary by Methods** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Datasets** | **AWGAN** | **Harmony** | **iMAP** | **Liger** | **MNN** | **scVI** | **Seurat v4** |
| **Human PBMC** | **11** | 19 | 16 | **12** | 20 | 24.5 | **9.5** |
| **Pancreas rm** | **11** | **10** | **13** | 18 | 29 | 22 | 14 |
| **Cell Lines** | **15** | **19** | **9** | 20 | 20 | 26 | 20 |
| **Mouse Cell Atlas** | **13** | 15.5 | 18 | **14** | 15.5 | 25 | 14 |
| **MHSP** | **11.5** | 17.5 | **11** | 19 | 18 | 22 | **17** |
| **PBMC 3&68K** | **11.5** | **9.5** | 17 | 23 | 24 | 24 | **8** |
| **DC** | **11** | **7** | 16 | 31 | **13** | 20 | 22 |
| **Mouse Retina** | **14** | 22.5 | **14** | **13** | 21 | **12** | 20 |
| **Mouse Brain** | **14** | 16 | 22 | **11** | 21.5 | 17.5 | **10** |
| **Final Total Rank** | **112** | 136 | **136** | 161 | 182 | 193 | **134.5** |
| **Overall Rank** | **1** | **3.5** | **3** | 5 | 6 | 7 | **2** |

Table 2: Total Rank Results. The total rank of top three performers for each dataset (row) are highlighted in bold font.

When calculating the final total ranks, we did not consider the Human Cell Atlas dataset and the CRC dataset because the former dataset did not have available cell type labels and the latter dataset was mainly used to show the potential of AWGAN on downstream analysis. From table 2 we can see AWGAN performs better than other methods in the overall ranking and it is the only method that is always among the top three performers across all datasets. This result proves that AWGAN has a strong generalization ability.

Moreover, since Seurat v4 and iMAP are the two strong competitors in most datasets, we also conducted pairwise comparison between AWGAN and Seurat v4, and between AWGAN and iMAP on each dataset and with each metric (Figure 10).

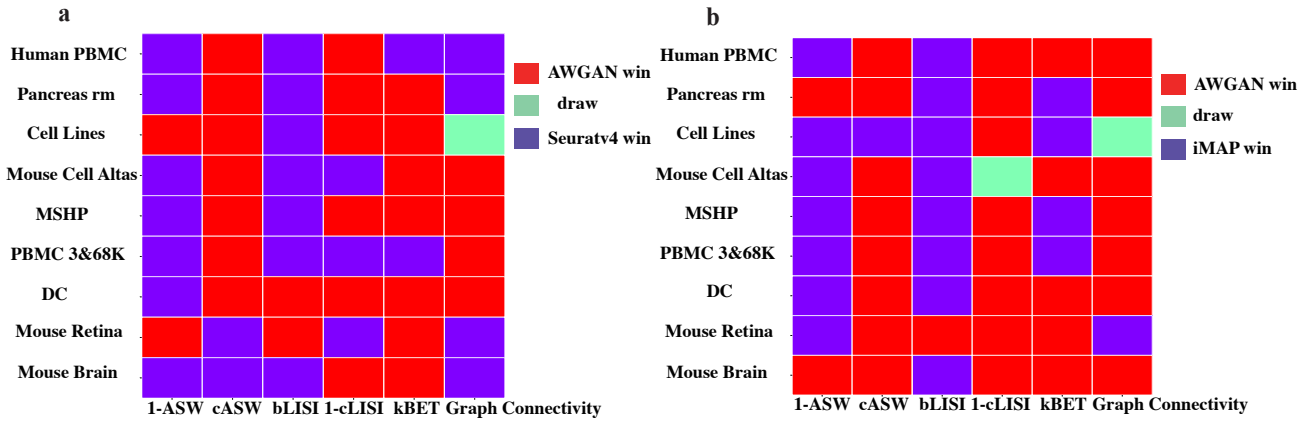


Figure 10: Comparison Heatmap. (a) Results for AWGAN vs. Seurat v4 in each dataset (row) and by each metric (column). (b) Results for AWGAN vs. iMAP in each dataset (row) and by each metric (column).

As we can observe from Figure 10 a, AWGAN performs better than Seurat v4 in terms of three metrics for cell type optimization (casw, 1-clisi, and Graph Connectivty) in the MHSP dataset, and the DC dataset. From Figure 10 b, AWGAN performs better than iMAP in terms of all these three metrics for cell type optimization in the Human PBMC dataset, the Pancreas dataset, the MHSP dataset, the PBMC 3&68K dataset, the DC dataset, and the Mouse Brain dataset. This explains why AWGAN has better overall performance in Table 2 than Seurat v4 and iMAP in datasets that contain a larger proportion of batch-specific cell types, such as the DC dataset, the MHSP dataset, and the Pancreas dataset. Moreover, AWGAN performs better than iMAP in terms of nearly all the metrics in the Mouse Brain dataset, which suggests the advantage of AWGAN when dealing with such a large-scale dataset. Apart from better performance on preserving batch-specific cell types, the benefits of AWGAN in terms of lighter structure and less memory usage, compared with iMAP, are important factors for researchers to

choose more suitable batch correction tools.

The detailed results for each of the four metrics are included in supplementary file 2. Overall, AWGAN performs well for both small-scale and large-scale datasets, especially for datasets with batch-specific cell types, for example, the DC dataset, the MHSP dataset, and the Pancreas dataset.

# 3 Discussion

In this paper, we have proposed and evaluated AWGAN, a novel model for removing batch effects. With a framework based on deep learning theory and generative models, AWGAN utilizes attention strategy to generate nearest neighbor pairs as a training set and has good performance both in terms of effectiveness and faster time compared with other methods.

More specifically, except for iMAP, Seurat v4 and Harmony, AWGAN achieves better results than the traditional batch effect removal algorithms in both ASW score and LISI score in most datasets. We note although both AWGAN and iMAP have advantages over other methods on different datasets, there are disadvantages of iMAP compared to AGWAN. Firstly, iMAP sometimes crashes (in 20% conditions, see example in Figure 6 in supplementary file 1 part 4), which is triggered by the improper neural network initialization. Secondly, according to Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) [20], iMAP chooses the same value for the weight of gradient penalty ($\lambda$), to satisfy 1- Lipschitz condition, which can be interpreted as one condition to ensure the robustness of the generative model. However, WGAN-GP chooses this value based on experiments on image datasets, so it is not suitable for iMAP to choose the same $\lambda$. AWGAN also has a similar design to WGAN-GP, but we find that we need to apply different $\lambda$ for some datasets. Therefore, this affects iMAP performance. Thirdly, since iMAP needs to train five different neural networks in total, its efficiency cannot be guaranteed, and this design extends its running time for training the whole

model especially when dealing with large-scale datasets. Finally, iMAP has poor performance on preserving batch specific cell types as we discussed in the last section of the benchmark results.

Some quantitative assessments of the performance of Seurat v4 are very favorable. However, Seurat v4 suffers from the drawback of not considering the conservation of cell type labels. For example, it ignores biological variation and overemphasizes data integration function. This drawback may negatively impacts biological research. We can see from Figure 7 in supplementary file 1 part 4 that in the DC dataset, CD1C cells are mapped to another type of cells, which is not desirable. In addition, we can also observe that in the Cell Lines dataset, after processing based on Seurat v4, the cells with the label ”Jurkat” still have batch effect from Figure 8 in supplementary file 1 part 4. Harmony also has issues related to batch-specific cell types, shown in Figure 9 in supplementary file 1 part 4. So overall, AWGAN performs better than Seurat v4 and Harmony with better visualization results.

Moreover, for batch correction methods like scVI, BBKNN, and Liger, there are some flaws in the method design principles. For scVI, this method extracts the gene expression statistics as an embedding in another space, so we need to regenerate the count matrix through a decoding model. Therefore, this variational inference method is even more complex based on its work. For BBKNN, this method is similar to the neighbors function in Scanpy [33], which can only make the visualization result better. BBKNN does not manipulate the raw data, so it cannot contribute much to downstream analysis. Liger has similar issues as it only returns integration results in a low dimensional space. Therefore, we cannot directly obtain corrected gene expression profiles in the original feature space from Liger. Nevertheless, methods like AWGAN, MNN, Seurat v4 and iMAP can adjust the original gene expression data and provide a better basis for downstream analysis.

In the problems related to large-scale data, AWGAN can also overcome the memory limi-

tation problems and finish the batch correction task without collapse error. In addition, in the research related to the CRC dataset, AWGAN shows its ability to find out misclassification cells and uncover the hidden cell-cell interactions. Overall, in most cases, the performance of AWGAN on the datasets mentioned in this paper is better than the current approaches.

# 4 Conclusion

We demonstrate that AWGAN is a valuable method for batch effect removal through experiments on ten different real-world datasets in comparison with seven different batch effect removal methods using four metrics. It takes the advantages of deep learning models and uses relatively few resources to achieve better results. Firstly, AWGAN can remove the batch effect in different datasets and preserve the biological variation (for example, gene expression difference and cell type distribution), while other state-of-the-art models, for example, iMAP, Harmony, and scVI, fail to keep it in some conditions. Secondly, our model supports various improvement extension applications, and researchers can adopt many hyper-parameters in our model. The strong adaptability of our model also allows it to be improved to achieve better results. Thirdly, running our model does not require very high technique or advanced device. Researchers can choose to use either cloud platform or CPU core in a personal computer to train our model and get a satisfactory result. We believe that the output of our model can better inform downstream analyses, such as specific cell function analysis, cell heterogeneity analysis, and others.

# 5 Methods

## 5.1 Attention-driven Data Pre-processing

We observed that most studies on batch effect removal chose the gene number of one cell within the range of at least **200** and up to **2000** ($N \in (200, 2000)$) [13,40] based on the highly variable gene (HVG) filtering method designed in Seurat v4 [11]. In this paper, we follow the above rules and process the raw count matrix. After obtaining the filtered count matrix, we proceed to the step of generating the training set. In this paper, inspired by the basic attention mechanism, we construct nearest neighbor cell pairs using cosine distance. For the set of cells with larger cosine similarity, we consider them as the nearest cell pair, and their attention to each other is 1, while the attention of the rest of the cells to them is 0. This type of relationship of attention is not transmissive but satisfies the principle of superposition. Therefore, any cell in the query dataset could find its own nearest neighbor cell in the reference dataset, generating a neighbor cell pair. The definition of nearest neighbor cell pairs can be explained as the Figure 10 in supplementary file 1 part 5.

To find the NNP, we only need to solve the following optimization problem. Assume there are two scRNA-seq datasets $Data_{true}$ (usually reference dataset) and $Data_{false}$ (usually combined with all the extra batch data, which is defined as query dataset), our target is to find:

$$(C_1^*, C_2^*) \in Pair = argmax_{C_i, C_j} CosineSimilarity(C_i, C_j) \tag{2}$$

$$C_i \in Data_{true}, C_j \in Data_{false} \tag{3}$$

We utilize the iteration algorithm with Numba acceleration [41] to solve this problem and collect the ideal solution. Here we declare that our algorithm always prefers to find cell pars with the largest cosine similarity (based on cosine value of two vectors). Based on this rule, sometimes NNP may not choose cells with the same cell type label, but since this situation

is very rare and to save time for running the attention matching algorithm, we choose this mechanism (other methods like MNN [10], iMAP [13] also ignore the effect of cell types). NNP with different cell type labels has little effect on the normal operation of the model, and we verify this in part 5.2. The algorithm for finding the target pairs is shown in Algorithm 1.

We also considered some variations in this algorithm, including selecting more pairs and considering the construction of training sets after averaging multiple attention pairs. However, the above selection effects were not as good as the algorithm in Algorithm 1. Details of the experiment can be found in the supplementary file 1 part 6.

**Algorithm 1** NNP Generating Algorithm. This algorithm is based on iterative approach and optimal solution searching approach. Its target is to generate the NNPs.

**Input:** Input parameters: Reference dataset $Data_{ref}$; Query dataset $Data_{que}$
**Ouput:** Output parameters: Training dataset $Data_{true}$; Training dataset $Data_{false}$.

```
INIT: initialize all parameters.
for item1 in Data_que do
    MaxSim = -inf; Q = None; R=None;
    for item2 in Data_ref do
        sim = CosineSimilarity(item1,item2)
        if sim > MaxSim then
            MaxSim = sim
            Q = item1
            R = item2
    Data_true.append(R)
    Data_false.append(Q)
Return Data_true, Data_false
```

## 5.2 AWGAN Training

Description of notations used in this section can be found in table 3 and the structure of AW-GAN is shown in the table 4. For the generator model, we apply the batch normalization [42] technique to accelerate the training speed, and we also utilize the Dropout method [43] based on momentum to prevent the model from overfitting. Based on the requirement in WGAN-GP [20],

we remove the BatchNorm layer for the discriminator model because we intend to ensure the smooth requirement for Lipschitz condition could not be affected by batch normalization layers.

| Notation | Description |
|---|---|
| $G$ | Generator Model in AWGAN |
| $D$ | Discriminator Model in AWGAN |
| $L$ | Total loss for AWGAN |
| $L_G$ | Loss in Generator Training |
| $L_D$ | Loss in Discriminator Training |
| $Data_{false}$ | Data need to remove the batch effect |
| $Data_{true}$ | Data without the batch effect (reference dataset) |
| $Data_{fixed}$ | Data after batch effect correction |
| $Pair$ | Pair set after running the attention method |
| $\lambda$ | Coefficient for gradient penalty |
| $P_g$ | Distribution of generator data |
| $P_r$ | Distribution of real data |
| $P_x$ | Distribution of interpolation data |
| $N$ | Selected number of genes |

Table 3: Notation List

| Generator | | Discriminator | |
|---|---|---|---|
| Layer Type | # | Layer Type | # |
| Input Layer | $N$ | Input Layer | $N$ |
| Hidden Layer | 1024 | Hidden Layer | 1024 |
| Batch Norm | 1024 | Hidden Layer | 512 |
| Hidden Layer | 512 | Output Layer | 1 |
| Batch Norm | 512 | - | - |
| Hidden Layer | 1024 | - | - |
| Batch Norm | 1024 | - | - |
| Output Layer | $N$ | - | - |

Table 4: AWGAN Structure

The minimization optimization target of our model can be concluded as:

$$L = E_{x^* \sim P_g}[D(x^*)] - E_{x' \sim P_r}[D(x')] + \lambda E_{x \sim P_x}[(\|\nabla_x D(x)\|_2 - K)^2] \tag{4}$$

The first two terms mean the Wasserstein Distance, which is the original expression in WGAN that we intend to minimize. However, if we minimize this value directly, the K-Lipschitz condition may not be satisfied. To satisfy the K-Lipschitz condition, the original WGAN uses a direct parameter range limitation. However, in the training progresses, the parameters will spontaneously concentrate on the boundary, so the gradient will not be well passed back to the model. Therefore, we introduce the third term and the gradient penalty mechanism [20]. The third term, gradient penalty, will first interpolate the sampling between true data and fake data and obtain a new distribution. In this problem, proved by the WGAN-GP paper, we takek $K = 1$ because the L2 norm of the ideal gradient for WGAN is 1 (1-Lipschitz condition) [20]. We then use this new distribution to compute the gradient penalty to match the Lipschitz condition over as large as possible space, thus eliminating the negative effects of parameters on the boundary. Some researchers agree with the idea to set $K = 0$, which is more reasonable [44]. We provide our experiment results with this choice in the supplementary file 1 part 7 and show that there is no significant difference between these two choices. The interpolation function between the $True$ dataset and $Fake$ dataset (including raw Fake data and temporary $G(Fake)$ data) is defined as:

$$\begin{cases} Interpolation(True, Fake) = \epsilon \cdot True + (1 - \epsilon) \cdot Fake \\ \epsilon \sim Uniform(0, 1) \\ x \sim P_x(Interpolation(True, Fake)) \end{cases} \tag{5}$$

For the choice of the activation function, we utilize the state-of-the-art activation function: Mish [45] to construct our model. The expression of Mish function can be defined as:

$$Mish(x) = x \cdot tanh(ln(1 + e^x)) \tag{6}$$

where $f(x) = ln(1 + e^x)$ is also known as SoftPlus function. Mish function has smoother gradient compared with Rectified Linear Unit (ReLU) [46] function, and smoother activation function allows better information integration and information mining performed by the deep neural network [45].

For the output layer of the generator, we adopt ReLU function with ResNet structure [47], which is also known as identity shortcut connection, to generate more stable results. To implement the ResNet structure, we change the expression of output layer into this form:

$$Y = T(F(X, \{W_i\}) + W_s X) \quad T, F \in \text{set(Activation Functions)} \tag{7}$$

$\{W_i\}$ means the set of inner weight matrix. $W_s$ is a linear map which helps to ensure the feature dimensions of X and F(X) are consistent. In this problem, their feature dimensions are the same, so $W_s$ can be treated as identity mapping. To make our model perform better in different dataset situations, we defined a new function for output layer, the transformation function $T$. For datasets which originally utilize log preprocessing method, we set $T = 1$, otherwise, $T = ReLU()$.

From the perspective of the back-propagation algorithm, we always meet non-zero terms when we are trying to find gradients based on ResNet structure, so the problem of gradient vanishing and white noise (very small gradient) can be solved to some extent [48].

For the overall formal design of the generator, inspired by the encoder-decoder model, we also add features such as embedding generation and embedding decoding in our network, which can extract the biological variation in scRNA-seq data.

In most of the time, since cells are highly differentiated, there are significant differences in gene expressions between different types of cells, so theoretically, it is easy for us to distinguish batch specific cell types [49, 50]. However, sometimes there may exist cells with similar gene expression levels but different cell identification. For such problem, we utilize the Discriminator and Dropout method design to help keep the balance between mixing the cells in different batches and discerning batch specific cell types. Firstly, the discriminator is designed to identify the cells in different batches. For batch specific cell types, the discriminator is able to quickly treat them as cells in other batches, and the model will not evaluate this situation. Secondly,

mixing these cells into known cells can be treated as an overfitting problem. The dropout method is designed to solve this kind of problems. Moreover, adjusting the training epochs and batch sizes can also generate a better result, depending on the chosen dataset. The Figure 11 supplementary file 1 part 5 illustrates the function of discriminator.

In total, the training algorithm can be found in Algorithm 2.

**Algorithm 2** Training process with AdamW [51] optimization method for AWGAN. The training epoch is the number of training iterations, and the batch size set for this model is $k$. They are all hyper parameters.

**Input:** Input parameters: Model $D$,$G$; Dataset $Data_{false}$, $Data_{true}$; Batch marker set $Batch$.
**Ouput:** Output parameters: Model $G$; Dataset $Data_{true}$

```
INIT: initialize all parameters.
for Type in Batch do
    Data_batch = Data_false[Type]
    for number of training iterations do
        for k steps do
            Refer = Data_true (sample with batchsize=k)
            Query = Data_batch (sample with batchsize=k)
            L_D = −E[D(Refer)]
            L_G = E[D(G(Query))]
            x = Interpolation(Refer,Query)
            Div = E[(‖∇_x D(x)‖_2 − 1)^2]
            L = L_D + L_G + Div/λ
            AdamW(D, L) # Training D
            L_G = -E[D(G(Query))]
            AdamW(G, L_G) # Training G
            if G is not convergent then
                Adjust λ = λ_new
                goto INIT
    Data_fixed = G(Data_batch)
    Data_true = concat(Data_true,Data_fixed)
Return G, Data_true
```

## 5.3 Model Adjustment and Evaluation

There are five kinds of parameters that can be adjusted in our model: epoch, batch size, gradient penalty, dropout rate, and learning rate. Moreover, for the choice in calculating the gradient penalty, we prefer using raw $Fake$ data for the first attempt. If the results are not satisfactory, we shift to temporary $G(Fake)$ data. The strategy we choose to improve the performance of our model is based on both the visualization result and quantitative assessments (especially ASW rate and LISI rate). We also refer from the stage2 of iMAP's structure for network design [13]. Finally, we chose the combination of different parameters which could optimize the quantitative rates and visualization performance. The choice for all key hyperparameters can be found in supplementary file 1 part 8. Moreover, for all the datasets, we run our model 10 times in each dataset to ensure that our output is convergent and stable.

To evaluate AWGAN and other benchmark methods except for Liger and BBKNN, we used the top 20 PCs [40] based on PCA method [52] with batch information and cell type information of the processed data as the input part for all the four quantitative metrics. For Liger, we chose to evaluate the norm matrix of the data generated by Liger with the original batch and cell type information. For BBKNN, since this method does not generate either gene expression information or latent space information, we did not compute metrics on this method. To evaluate batch optimization task more reliably, we removed the batch specific cell types when we generated the metrics related to this part. The visualization of quantitative comparisons can be found in the 'Results' section and the detailed quantitative results can be found in the supplementary file 2.

## 5.4 Metrics Details

- ASW: Average silhouette width, also known as ASW, is computed based on cluster and average distance. For one data point, we first calculate the difference between the average

distance between this point and the other points in a neighboring cluster to the average distance between this point and other members in the same cluster. After finishing this step, we divide this result by the larger distance to get the silhouette value. We repeat the process for each data point for a certain number of iteration times and take the average across them. More specifically, ASW is defined as:

$$ASW = \frac{(d_{neighbors} - d_{members})}{max(d_{neighbors}, d_{members})} \tag{8}$$

Since we normally have two kinds of clusters: batch and cell type, we use the F1 score of ASW to evaluate our model. The F1 score of ASW is defined as:

$$F1_{ASW} = \frac{2(1 - ASW_{batch})(ASW_{celltype})}{1 - ASW_{batch} + ASW_{celltype}} \tag{9}$$

For the F1 score, since we intend to have different batches integrated in one reference dataset, and keep the cell type distribution to be the same, so the model with low $ASW_{batch}$ rate and high $ASW_{celltype}$ rate will be considered as having good performance. Therefore, a higher F1 score suggests a better performing model.

- kBET: k-Nearest Neighbor (kNN) batch-effect test, as known as kBET, utilizes kNN-based algorithm to measure the results of batch effect removal. The algorithm randomly selects some data points and tests whether the local batch label distribution among them is similar to the global distribution. Ideally, the batch label distribution in the sample and the global data distribution should be very close to each other, so the rejection rate should be 0. In fact, there are some differences between the two, so we get an actual rejection rate between 0 and 1. To be consistent with the rest of the methods, we define the acceptance rate as:

$$\text{Acceptance Rate} = 1 - \text{Rejection Rate} \tag{10}$$

A higher acceptance rate of one model suggests better performance of this model. For

large-scale datasets, we find the rejection rates are often too large to compare across different batch correction methods and the kNN searching process often fails because of memory issues, so we adopt two strategies: neighborhood size adjustment and subsampling to overcome this obstacle.

The parameter that controls neighborhood size is **heuristics**. If it is set to be True, kBET will return the optimal neighborhood size that can generate the maximal rejection rate. However, for large-scale datasets, this value is very large under **heuristics** =True so that the differences between the different methods' rejection rates are not significant. Therefore, we set this value to False so that kBET can automatically choose 25% cells in the dataset to calculate the rejection rate, which magnifies the difference among different methods' rejection rates.

For subsampling, if we use $n$ to represent the sample size and k to represent neighborhood size, and if $nk > 2^{31}$, the kNN algorithm will stop searching in R, which is often the case for large-scale datasets [24]. Therefore, for large-scale datasets, we split each of them into ten subsamples and calculated overall rejection rate based on the average of the 10 rates from the subsamples.

- LISI: Local inverse Simpson's index (LISI) adopts similar principle to kBET, but the rule for choosing the neighbors in kNN is different from kBET. LISI chooses neighbors based on the local distribution of distances with an invariant perplexity. Moreover, the chosen neighbors will be used to calculate inverse Simpson's index, which is the number of accepting cells in neighbors. The definition of Inverse Simpson's rate is:

$$ISR = \frac{1}{\sum_{b=1}^{B} p(b)} \tag{11}$$

where b in the cell index and B is the cluster of one batch. ISR is the batch probabilities in local neighbors. Since LISI assigns a value for every different cell, we evaluate our

model based on the average LISI rate. If it approaches the real batch number, the model with such LISI is better than the others. According to the definition of the LISI rate, we define two metrics: cell type LISI rate (cLISI) and batch LISI rate (bLISI). Our goal is to increase the bLISI rate while decreasing the cLISI rate. Therefore, we utilize the F1 score of LISI to evaluate different methods. The F1 score of LISI is defined as:

$$F1_{LISI} = \frac{2(1 - cLISI)(bLISI)}{1 - cLISI + bLISI} \tag{12}$$

- Graph Connectivity: This method is based on kNN and its target is to evaluate the conservation of cell type distribution after batch correction work [5]. For the kNN graph $G$, of one set of integrated data, we evaluated whether cells with cell type label $c$ are directly connected with each other across batches. Moreover, we could create the subset kNN graph $G(N_c, E_c)$ only for the cluster with cell type label $c$. Let $T$ represent the set containing all cell type labels, we could use this equation to compute the graph connectivity score:

$$gc = \frac{1}{|T|} \sum_{c \in T} \frac{|LCC\left(G\left(N_c, E_c\right)\right)|}{|N_c|} \tag{13}$$

  In this equation, $|LCC()|$ means the number of nodes in the largest connected component (LCC) of the given graph, and $|N_c|$ means the number of nodes with cell type label $c$. Based on the concept of LCC, higher graph connectivity score means better cell type optimization performance of a method.

## 5.5 Device Information

The CPU of our device is Intel(R) Xeon(R), 2.30 GHz. The GPU of our device is NVIDA Tesla P100, the RAM is 25GB, and the hard disk size is 140GB. The version of Pytorch is 1.7.0 [53]. To run iMAP on the human cell atlas dataset, iMAP, MNN, Harmony, Seurat v4 and Liger on mouse retina dataset, and all methods on mouse brain dataset, we used a HPC tool. To make

life easier, the Colab platform under Google can meet most needs of running our code. The versions of software packages we used can be found in supplementary file 1 part 9.

## 5.6 Dataset Information

We considered eleven single-cell datasets to investigate the performance of AWGAN in comparisons with other methods, including seven small-scale datasets and four large-scale datasets. The Cell Lines dataset [25] was generated by 10x Genomics with three different batches. The DC dataset [26] is composed of two different batches based on the Smart-seq2 protocol. The Pancreas rm dataset [13] includes human pancreatic cells measured by 5 different platforms and is loaded from R package: SeuratData. The PBMC 3&68K [27] [25] is also generated by 10x Genomics, and it is worthy to mention that PBMC 68K is the dataset with processed version. The MHSP dataset [28] [54] is acquired by researchers using the Mars-seq protocol and the Smart-seq2 protocol. The Mouse Cell Atlas dataset [29] [30] contains two batches from the Microwell-seq and the Smart-seq2 protocol. The Human PBMC dataset [25] is generated by 10x Genomics protocol with two batches. The Human Cell Atlas dataset [34] is a large dataset with bone marrow cells in batch 1 and cord blood-derived cells in batch 2 based on the 10x Genomics protocol. The Mouse Retina dataset [31] [32] is composed of mouse retina data generated using the Drop-seq technology by two unassociated laboratories. The Mouse Brain dataset [35] [36] is composed of two datasets from two different research groups, which were generated by Drop-seq and SPLiT-seq protocols separately. The CRC dataset [37] is a scRNA-seq dataset provided by 18 CRC patients using both Smart-seq2 and 10x Genomics. Moreover, the cell type composition of each dataset can be found in supplementary file 3.

## 5.7 Code Availability

An open-source implementation of the AWGAN and source codes to reproduce the results can be downloaded from: https://github.com/HelloWorldLTY/AWGAN.

## 5.8 Data Availability

Download approaches for the datasets we used in this paper can be found in the supplementary file 4.

# References


[1] E. Papalexi and R. Satija, "Single-cell rna sequencing to explore immune cell heterogeneity," *Nature Reviews Immunology*, vol. 18, no. 1, p. 35, 2018.

[2] B. Pijuan-Sala, C. Guibentif, and B. Göttgens, "Single-cell transcriptional profiling: a window into embryonic cell-type specification," *Nature Reviews Molecular Cell Biology*, vol. 19, no. 6, pp. 399–412, 2018.

[3] M. L. Suvà and I. Tirosh, "Single-cell rna sequencing in cancer: lessons learned and emerging challenges," *Molecular cell*, vol. 75, no. 1, pp. 7–12, 2019.

[4] J. T. Leek, R. B. Scharpf, H. C. Bravo, D. Simcha, B. Langmead, W. E. Johnson, D. Geman, K. Baggerly, and R. A. Irizarry, "Tackling the widespread and critical impact of batch effects in high-throughput data," *Nature Reviews Genetics*, vol. 11, no. 10, pp. 733–739, 2010.

[5] M. Luecken, M. Büttner, K. Chaichoompu, A. Danese, M. Interlandi, M. Mueller, D. Strobl, L. Zappia, M. Dugas, M. Colomé-Tatché, and F. Theis, "Benchmarking

atlas-level data integration in single-cell genomics," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/05/23/2020.05.22.111161

[6] M. E. Ritchie, B. Phipson, D. Wu, Y. Hu, C. W. Law, W. Shi, and G. K. Smyth, "limma powers differential expression analyses for rna-sequencing and microarray studies," *Nucleic acids research*, vol. 43, no. 7, pp. e47–e47, 2015.

[7] W. E. Johnson, C. Li, and A. Rabinovic, "Adjusting batch effects in microarray expression data using empirical bayes methods," *Biostatistics*, vol. 8, no. 1, pp. 118–127, 2007.

[8] J. D. Welch, V. Kozareva, A. Ferreira, C. Vanderburg, C. Martin, and E. Z. Macosko, "Single-cell multi-omic integration compares and contrasts features of brain cell identity," *Cell*, vol. 177, no. 7, pp. 1873–1887, 2019.

[9] R. Lopez, J. Regier, M. Cole, M. Jordan, and N. Yosef, "Bayesian inference for a generative model of transcriptome profiles from single-cell rna sequencing," *bioRxiv*, 2018. [Online]. Available: https://www.biorxiv.org/content/early/2018/10/14/292037

[10] L. Haghverdi, A. T. Lun, M. D. Morgan, and J. C. Marioni, "Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors," *Nature biotechnology*, vol. 36, no. 5, pp. 421–427, 2018.

[11] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck III, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija, "Comprehensive integration of single-cell data," *Cell*, vol. 177, no. 7, pp. 1888–1902, 2019.

[12] K. Polański, M. D. Young, Z. Miao, K. B. Meyer, S. A. Teichmann, and J.-E. Park, "Bbknn: fast batch alignment of single cell transcriptomes," *Bioinformatics*, vol. 36, no. 3, pp. 964–965, 2020.

[13] D. Wang, S. Hou, L. Zhang, X. Wang, B. Liu, and Z. Zhang, "imap: integration of multiple single-cell datasets by adversarial paired transfer networks," *Genome biology*, vol. 22, no. 1, pp. 1–24, 2021.

[14] I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-r. Loh, and S. Raychaudhuri, "Fast, sensitive and accurate integration of single-cell data with harmony," *Nature methods*, vol. 16, no. 12, pp. 1289–1296, 2019.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[16] X. Li, K. Wang, Y. Lyu, H. Pan, J. Zhang, D. Stambolian, K. Susztak, M. P. Reilly, G. Hu, and M. Li, "Deep learning enables accurate clustering with batch effect removal in single-cell rna-seq analysis," *Nature communications*, vol. 11, no. 1, pp. 1–14, 2020.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[18] K. Kurach, M. Lučić, X. Zhai, M. Michalski, and S. Gelly, "A large-scale study on regularization and normalization in gans," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3581–3590.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *ArXiv*, vol. abs/1701.07875, 2017.

[20] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *arXiv preprint arXiv:1706.08500*, 2017.

[22] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[23] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[24] M. Büttner, Z. Miao, F. A. Wolf, S. A. Teichmann, and F. J. Theis, "A test metric for assessing single-cell rna-seq batch correction," *Nature methods*, vol. 16, no. 1, pp. 43–49, 2019.

[25] G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu *et al.*, "Massively parallel digital transcriptional profiling of single cells," *Nature communications*, vol. 8, no. 1, pp. 1–12, 2017.

[26] A.-C. Villani, R. Satija, G. Reynolds, S. Sarkizova, K. Shekhar, J. Fletcher, M. Griesbeck, A. Butler, S. Zheng, S. Lazo *et al.*, "Single-cell rna-seq reveals new types of human blood dendritic cells, monocytes, and progenitors," *Science*, vol. 356, no. 6335, 2017.

[27] 10X-Genomics, "Data from "3k pbmcs from a healthy donor."," https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k Accessed July 30th, 2021.

[28] S. Nestorowa, F. K. Hamey, B. Pijuan Sala, E. Diamanti, M. Shepherd, E. Laurenti, N. K. Wilson, D. G. Kent, and B. Göttgens, "A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation," *Blood, The Journal of the American Society of Hematology*, vol. 128, no. 8, pp. e20–e31, 2016.

[29] X. Han, R. Wang, Y. Zhou, L. Fei, H. Sun, S. Lai, A. Saadatpour, Z. Zhou, H. Chen, F. Ye *et al.*, "Mapping the mouse cell atlas by microwell-seq," *Cell*, vol. 172, no. 5, pp. 1091–1107, 2018.

[30] T. M. Consortium *et al.*, "Single-cell transcriptomics of 20 mouse organs creates a tabula muris," *Nature*, vol. 562, no. 7727, pp. 367–372, 2018.

[31] K. Shekhar, S. W. Lapan, I. E. Whitney, N. M. Tran, E. Z. Macosko, M. Kowalczyk, X. Adiconis, J. Z. Levin, J. Nemesh, M. Goldman *et al.*, "Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics," *Cell*, vol. 166, no. 5, pp. 1308–1323, 2016.

[32] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck *et al.*, "Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets," *Cell*, vol. 161, no. 5, pp. 1202–1214, 2015.

[33] F. A. Wolf, P. Angerer, and F. J. Theis, "Scanpy: large-scale single-cell gene expression data analysis," *Genome biology*, vol. 19, no. 1, pp. 1–5, 2018.

[34] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy *et al.*, "Science forum: the human cell atlas," *elife*, vol. 6, p. e27041, 2017.

[35] A. Saunders, E. Z. Macosko, A. Wysoker, M. Goldman, F. M. Krienen, H. de Rivera, E. Bien, M. Baum, L. Bortolin, S. Wang *et al.*, "Molecular diversity and specializations among the cells of the adult mouse brain," *Cell*, vol. 174, no. 4, pp. 1015–1030, 2018.

[36] A. B. Rosenberg, C. M. Roco, R. A. Muscat, A. Kuchina, P. Sample, Z. Yao, L. T. Graybuck, D. J. Peeler, S. Mukherjee, W. Chen *et al.*, "Single-cell profiling of the developing

mouse brain and spinal cord with split-pool barcoding," *Science*, vol. 360, no. 6385, pp. 176–182, 2018.

[37] L. Zhang, Z. Li, K. M. Skrzypczynska, Q. Fang, W. Zhang, S. A. O'Brien, Y. He, L. Wang, Q. Zhang, A. Kim *et al.*, "Single-cell analyses inform mechanisms of myeloid-targeted therapies in colon cancer," *Cell*, vol. 181, no. 2, pp. 442–459, 2020.

[38] X. Wang, Y. He, Q. Zhang, X. Ren, and Z. Zhang, "Direct comparative analyses of 10x genomics chromium and smart-seq2," *Genomics, Proteomics & Bioinformatics*, 2021.

[39] M. Efremova, M. Vento-Tormo, S. A. Teichmann, and R. Vento-Tormo, "Cellphonedb: inferring cell–cell communication from combined expression of multi-subunit ligand–receptor complexes," *Nature protocols*, vol. 15, no. 4, pp. 1484–1506, 2020.

[40] H. T. N. Tran, K. S. Ang, M. Chevrier, X. Zhang, N. Y. S. Lee, M. Goh, and J. Chen, "A benchmark of batch-effect correction methods for single-cell rna sequencing data," *Genome biology*, vol. 21, no. 1, pp. 1–32, 2020.

[41] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[44] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. Gool, "Wasserstein divergence for gans," in *ECCV*, 2018.

[45] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, vol. 4, 2019.

[46] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and B. McWilliams, "The shattered gradients problem: If resnets are the answer, then what is the question?" in *International Conference on Machine Learning*. PMLR, 2017, pp. 342–350.

[49] J. Ernst, P. Kheradpour, T. S. Mikkelsen, N. Shoresh, L. D. Ward, C. B. Epstein, X. Zhang, L. Wang, R. Issner, M. Coyne *et al.*, "Mapping and analysis of chromatin state dynamics in nine human cell types," *Nature*, vol. 473, no. 7345, pp. 43–49, 2011.

[50] S. M. Cohen, J. Brennecke, and A. Stark, "Denoising feedback loops by thresholding—a new role for micrornas," *Genes & development*, vol. 20, no. 20, pp. 2769–2772, 2006.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[54] F. Paul, Y. Arkin, A. Giladi, D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, D. Winter, D. Lara-Astiaso, M. Gury, A. Weiner *et al.*, "Transcriptional heterogeneity and lineage commitment in myeloid progenitors," *Cell*, vol. 163, no. 7, pp. 1663–1677, 2015.