scDeepHash: An automatic cell type annotation and cell retrieval method for large-scale scRNA-seq datasets using neural network-based hashing

Shihao Ma[1, 2, 3 *], Yanyi Zhang[1 *], Bohao Wang[1], Zian Hu[1], Jingwei Zhang[1], and Bo Wang[1, 2, 3 †]

[1]University of Toronto, [2]Peter Munk Cardiac Centre, [3]Vector Institute, [†]Senior Author, [*]These authors contribute equally to this work.

Abstract

Single-cell RNA-sequencing technologies measure transcriptomic expressions, which quantifies cell-to-cell heterogeneity at an unprecedented resolution. As these technologies become more readily available, the number of scRNA-seq datasets increases drastically. Prior works have demonstrated that bias-free, holistic single-cell profiling infrastructures are essential to the emerging automatic cell-type annotation methods. We propose scDeepHash, a scalable scRNA-seq analytic tool that employs content-based deep hashing to index single-cell gene expressions. scDeepHash allows for fast and accurate automated cell-type annotation and similar-cell retrieval. We also demonstrated the performance of scDeepHash by benchmarking it against current state of the art methods across multiple public scRNA-seq datasets.

1 Introduction

Single-cell RNA-sequencing technology quantifies transcriptomic expressions at single-cell level, leading to discoveries rooted in cell-to-cell heterogeneity. Cell-type annotation, or assignment of cell type labels to single cells, is an important step in analysing the RNA sequencing data [1]. With its wide and mature applications, scRNA-seq datasets have become increasingly and readily available in larger scale. The scale of scRNA-seq datasets has increased significantly in sample sizes, dimensions of features (genes), number of heterogeneous cell populations and inter-correlation between cell populations. Such explosive emergence of large-scale scRNA-seq data makes manual annotation no longer desirable due to the need of labor-intensive efforts, and demands for a more reliable and efficient automated cell-type annotation method for biological downstream analysis.

Existing cell-type annotation tools generally fall into three categories: annotation based on known marker genes; annotation based on similarity between the query dataset and the labeled reference dataset; supervised learning that labels query cells using a classifier trained from a labeled reference dataset. Yet, these methods cannot be easily adapted to annotate large-scale scRNA-seq datasets. Marker gene-based methods are labor intensive in practice and the results can have high variation. Similarity-based methods usually suffer from the trade-off between accuracy and speed. Machine learning-based methods can have inconsistent performance on different datasets and may be lack of interpretablity. We discuss the details of existing methods in Section 2.

In this work, we propose scDeepHash that efficiently encodes single-cell gene expressions using short, binary hash codes, and represents each distinct cell type using a 'hash bucket'. In the field of computer vision, deep hashing has proven its efficiency and accuracy in classifying and retrieving large-scale image datasets [2, 3, 4, 5]. With its hash code encodings of scRNA-seq data, scDeepHash can handle large-scale scRNA-seq datasets, and accurately annotate and retrieve cells.

Our main contributions are summarized as follows: **(1)** our high-quality hash codes well preserve the similarity information within the same cell type in the original gene expression space; **(2)** the short hash codes enable efficient binary computation for cell similarity; **(3)** in addition to cell-type annotations, scDeepHash can also perform similar-cell retrieval tasks. That is, given an annotated cell, scDeepHash can reliably and efficiently retrieve similar cells within a database for further analysis. To our best knowledge, this is the first time the idea of 'learning to hash' is applied to cell-type annotation and similar-cell retrieval. We demonstrate that scDeepHash achieves comparable performance or outperforms state of the art methods on accuracy and speed for both cell-type annotation and similar-cell retrieval on extensive large-scale scRNA-seq datasets.

2 Related Works

Existing tools for cell-type annotation can be categorized into three types: marker gene-based methods, similarity-based methods and machine learning-based methods.

Marker gene-based methods utilize biological prior knowledge for interpretable annotations. scRNA-seq data, after dimension reduction and clustering [6, 7], are manually annotated on cluster level by referencing the highly expressed genes with public cell-type marker gene databases such as CellMarker and PanglaoDB [8, 9]. An alternative approach is to incorporate the prior knowledge with a statistical inference framework. For example, SCINA [10] operates at cluster level by fitting a bimodal distribution to the expression of marker genes. However, the clustering-based gene selection is very sensitive to the clustering methods, choice of hyperparameters, and the number of cells in datasets, which can cause large variation in the downstream analysis [11, 12]. Another example is CellAssign [13] that includes a marker gene-based matrix within a probabilistic graphical model to perform annotation at single-cell level, which avoids the clustering step. However, its annotation performance is still largely dependent on the choice of canonical cell markers, which is often complex and labor intensive in practice [11, 12].

Similarity-based methods assign cell labels by measuring the similarities between query cells and reference datasets, which can be categorized into two types: cluster-to-reference, or single cell-to-reference. Examples like CIPR [14] and ClustifyR [15] fall into the former category, which unavoidably suffers from the limitations of clustering; for the latter category, notable tools include scmap [16], SingleR [17], and scMatch [18], which generally leverage statistical metrics (like Spearman and Pearson correlation) to determine the similarity between each query cell and reference public datasets like Blueprint [19] and FANTOM [20] for annotation. However, such methods can hardly balance the trade-off between accuracy and speed. Exceptionally fast inference can be achieved when each query cell is compared with the approximate referenced gene expression (like referenced cluster centroid in scmap-cluster [16]), but annotation accuracy will be significantly impaired; comparing cell to cell and then iteratively performing scoring aggregation per label like SingleR [17] can improve the annotation accuracy, but it is at the big cost of annotation speed.

Machine learning-based methods aim at overcoming the intrinsic variation of data and noise to find the invariant latent features through learning the model distribution of labeled datasets. Methods built on top of traditional classification methods such as Random Forest [21, 22], KNN [23, 24] and SVM [25, 11] can achieve decent results, but they are limited to application to cell-type annotation only. Recent works developed more complex deep learning architectures to increase models' capability. For example, scVI [26] uses a hierarchical Bayesian model based on a modification of Variational Autoencoder (VAE) to learn the invariant latent features, but its annotation performance will encounter sharp decrease on deeply annotated or highly inter-correlated datasets [27]; MARS [28] applies meta learning through joint annotated and unannotated experiments to learn the shared feature among similar cells. However, users cannot perform annotation until the successful joint training, which sharply slows down its annotation speed.

3 Methods

Suppose we are given the high dimensional single-cell RNA-seq data as $\mathcal{X} = \{(x_i, l_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^D$ denotes each cell and $l_i \in L$ denotes the responding cell label. The objective of scDeepHash is to learn a nonlinear hashing function that can map the input RNA-seq data into a K-bit Hamming space, where each cell be represented as a K-bit hash code $h_i \in \{0, 1\}^K$. As shown in Figure 1, scDeepHash consists of three sequential phases: (1) cell anchors generation, (2) hash function training, and (3) cell annotation or retrieval.

3.1 Cell Anchors Generation

The concept of cell anchors is originally proposed in scRNA-seq data integration by using a subset of cells to represent a biological state [29]. Inspired by this concept, we propose to use cell anchors in the Hamming space to represent heterogeneous cell types. Ideally, cell anchors should be mutually distant to preserve the dissimilarity of different cell types. The distance between two cell anchors a_1, a_2 can be defined using Hamming distance $D_H(a_1, a_2)$, which denotes the number of different bits between two hash codes. There are two different ways to obtain cell anchors: one is to directly learn from RNA-seq features, and the other is to pre-define the anchors with distant-enough Hamming distance. As shown in CSQ [30], the latter option is a better choice due to its better performance and simplicity. Thus, the first step of scDeepHash is to generate pre-defined cell anchors.

We leverage the properties of Hadamard matrix for our cell anchor generation. For a $K \times K$ Hadamard matrix $\mathcal{H}_K = [h_1, h_2, ..., h_K]$, $h_k \in \{-1, 1\}^K$, the pairwise Hamming distance of any two column vectors is exactly $D_H(h_i, h_j) = K/2$, $(i \neq j)$, which is desirable for the purpose of cell anchors generation. Thus, cell anchors can be obtained by sampling the column vectors and then converting them into Hamming space $\{0, 1\}^K$. We use *Sylvester's construction* [31] to construct a K-dimensional Hadamard matrix:

$$\mathcal{H}_K = \begin{bmatrix} \mathcal{H}_{2^{n-1}} & \mathcal{H}_{2^{n-1}} \\ \mathcal{H}_{2^{n-1}} & -\mathcal{H}_{2^{n-1}} \end{bmatrix} = \mathcal{H}_2 \otimes \mathcal{H}_{2^{n-1}}$$

where $K = 2^n$, $n \geq 1$, $\mathcal{H}_1 = [1]$, and $\otimes$ denotes the *Kronecker product* [32]. Let m denotes the number of cell types, we choose the n such that $2^{n-1} < m < 2^n$ to guarantee that we have enough column vectors in the Hadamard matrix to sample our cell anchors from.

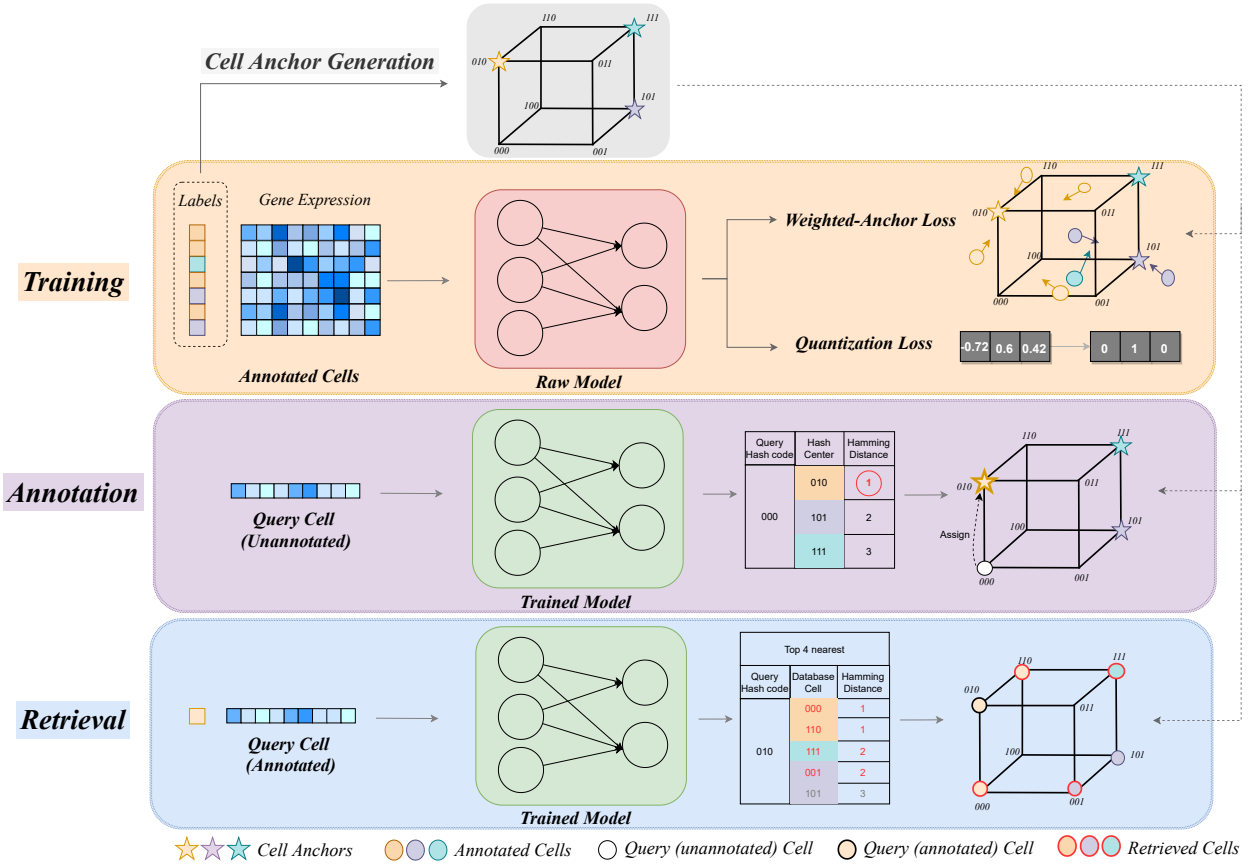

Figure 1: **The overview of scDeepHash**. scDeepHash consists of three sequential phases: (1) cell anchor generation (in grey): from the annotated cells' labels, cell anchors are generated leveraging properties of Hadamard Matrix (Sec. 3.1). The generated anchors will be used as basis for the training phase and annotation phase. (2) Hash function training (in orange): scRNA-seq dataset are used to supervisingly train the deep learning model for optimizing the targeted hashing function constrained with two specially-designed losses: weighted cell-anchor loss and quantization loss (Sec. 3.2). (3) Cell annotation (in purple) or retrieval (in blue): for annotation, unannotated scRNA-seq data will be fed through the trained model to generate the query cell's hash codes, which are used for cell-type annotation based on closest-hash center strategy (Sec. 3.3); for retrieval, annotated scRNA-seq data will also be passed through the trained model to generate the hash codes, which are utilized for retrieval of the top-k similar cells in database (Sec. 3.3).

3.2 Hash Function Training

Using deep learning to learn hash functions, also known as deep hashing, is a well-established methodology in the hashing community. As shown in DHN [4], DCH [33] and HashNet [5], the loss function is usually formulated as Maximum a Posterior (mAP) estimation, which is then split into two components using the Bayes' theorem. The first component is the likelihood function, which, in our context, aims at maximizing the probability of finding the corresponding cell anchor given a cell hash code. The second component is the prior distribution over hash codes, which aims at enforcing minimum information loss when quantizing continuous codes into discrete binary hash codes. In scDeepHash, we formulate the two losses as *Weighted Cell-Anchor Loss* and *Quantization Loss*.

Weighted Cell-Anchor Loss - L_{CA}. Define the generated hash code set as $\mathcal{H} = \{h_1, \ldots, h_n\}$, and the cell anchor set $\mathcal{A} = \{a_1, \ldots, a_n\}$, where h_i and a_i are the learned hash code and the corresponding cell anchor for cell i, respectively. Our training objective is to aggregate the generated hash codes of cell data around their corresponding cell anchors so that the difference between them is minimized. Since all the cell anchors are binary vectors of 0 and 1, we adopt the Binary Cross Entropy (BCE) loss to penalize the differences bit by bit.

The presence of rare cell types may induce a 'skewed' distribution of the available cell samples, which leads to poor annotation performance for the rare cell types. Thus, we introduce a weight coefficient based on the effective number of cells in each cell types to handle data imbalance [34]:

$$\alpha = (\frac{1-\beta}{1-\beta^{n_j}}), \tag{1}$$

where β is a hyper-parameter $\in \big[0, 1\big)$, and n_j is number of cell samples of j-th cell type. We then calculate the weight coefficient for each cell i using (1) and obtain $\alpha = \{\alpha_1, \ldots, \alpha_n\}$. Assuming the hash codes and the cell anchors are k-bits, the *Weighted Cell-Anchor Loss* is:

$$L_{CA} = \sum_{i=1}^{N} \alpha_i \sum_{k=1}^{K} \big[a_{ik} \cdot \log h_{ik} + (1 - a_{ik}) \log(1 - h_{ik})\big] \tag{2}$$

Quantization Loss - L_Q. For ease of optimization during training, continuous relaxation is needed to generate hash code h_i, and thus we apply a quantization loss L_Q for this purpose. Inspired by DCH [33], we used a Cauchy distribution prior to control the quantization error, which is defined as:

$$P(h_i) = \frac{\gamma}{\gamma + D_H(|h_i|, \mathbf{1})} \tag{3}$$

where γ is the scale parameter of Cauchy distribution and $\mathbf{1}$ is a vector of 1's. Formally, the quantization loss can then be defined as:

$$L_Q = \sum_{i=1}^{N} \log\big(1 + \frac{D_H(|h_i|, \mathbf{1})}{\gamma}\big). \tag{4}$$

Finally, combining (2) and (4), the final target loss function of scDeepHash is

$$\min_{\Theta} L = L_{CA} + \lambda L_Q, \tag{5}$$

where Θ is the parameter set for L, and λ is a hyper-parameter to calibrate the weight of quantization.

3.3 Cell Annotation & Retrieval

With a well-trained model that can extract high-dimensional cell features (genes) into binary hash code, scDeepHash can thus perform both the cell-type annotation and similar-cell retrieval in an accurate and efficient way.

Annotation - Each of the query cell is passed through the network to generate its corresponding hash code. After comparing the pairwise Hamming distance between the cell's hash code and each cell anchor, we choose the closest cell anchor as the query cell's assignment (Figure 1), which we call the Closest Cell Anchor **(CCA)** strategy . It is simple but very efficient since its computation complexity is $\mathbf{O(m)}$, where m is the number of cell types. This can ensure that the annotation can be applied to increasingly large-scale unannotated datasets without sacrificing the annotation speed.

Retrieval - All the cell hash codes within the database are first split into different 'hash buckets' based on their corresponding cell anchors. During retrieval, generated hash code of each query cell is used to compute the closest cell anchor, similar to the cell annotation process. Then, within the selected 'hash bucket', k nearest-neighbor strategy (Figure 1) is used to fetch the top k-nearest cells, where k is of users' choice. If k is larger than the size of the selected bucket, we move to the next closest cell anchor and repeat the same process. The retrieval computation complexity is $\mathbf{O(m + n)}$, where m is the number of cell types and n is the average number of cells in all 'hash buckets'.

3.4 Hyper-parameters Settings

Our neural network has five fully connected layers. Rectified linear unit (ReLU) function is used as activation function for all the layers, with an extra sign function to quantize the final network's outputs. The network is trained using stochastic gradient descent (Adam optimizer) with learning rate of 1×10^{-5}, weight decay of 5×10^{-4} and batch size of 128. The regularization constant λ is settled to 1×10^{-4} while the β in weight coefficient is set to 0.9999 through grid search. Number of hashing bits K is determined based on the number of cell types m, which is specific to post-processed datasets. The entire code base is implemented using PyTorch Lightning and is available at `https://github.com/bowang-lab/scDeepHash`.

4 Results

Table 1: Datasets Configuration

Dataset	n_{cells}	n_{genes}	$n_{\text{cell types}}$ (>10 cells)	Reference
Xin	1,449	33,889	4(4)	[35]
Baron Human	8,569	17,499	14(13)	[36]
TM	54,865	19,791	55(55)	[37]
AMB92	12,832	42,625	110(92)	[38]
Zheng 68K	68,579	20,387	11(11)	[39]

To demonstrate the superiority of scDeepHash in both cell annotation and retrieval, we perform separate experiments on five scRNA-seq datasets with the same pre-processing steps [27] based on the cell data 'complexity': (1) the number of cells, (2) the number of features (genes), (3) the annotation levels, and (4) the correlation between cell populations, each representing a different level of challenge to learn the underlying hash function mapping. As shown in Table 1, Xin [35] and BaronHuman [36] represent typical-sized scRNA-seq datasets (around 1.5k to 9k cells), while TM [37] and Zheng 68K [39] represents large scRNA-seq datasets (50k+ cells); both Xin and AMB92

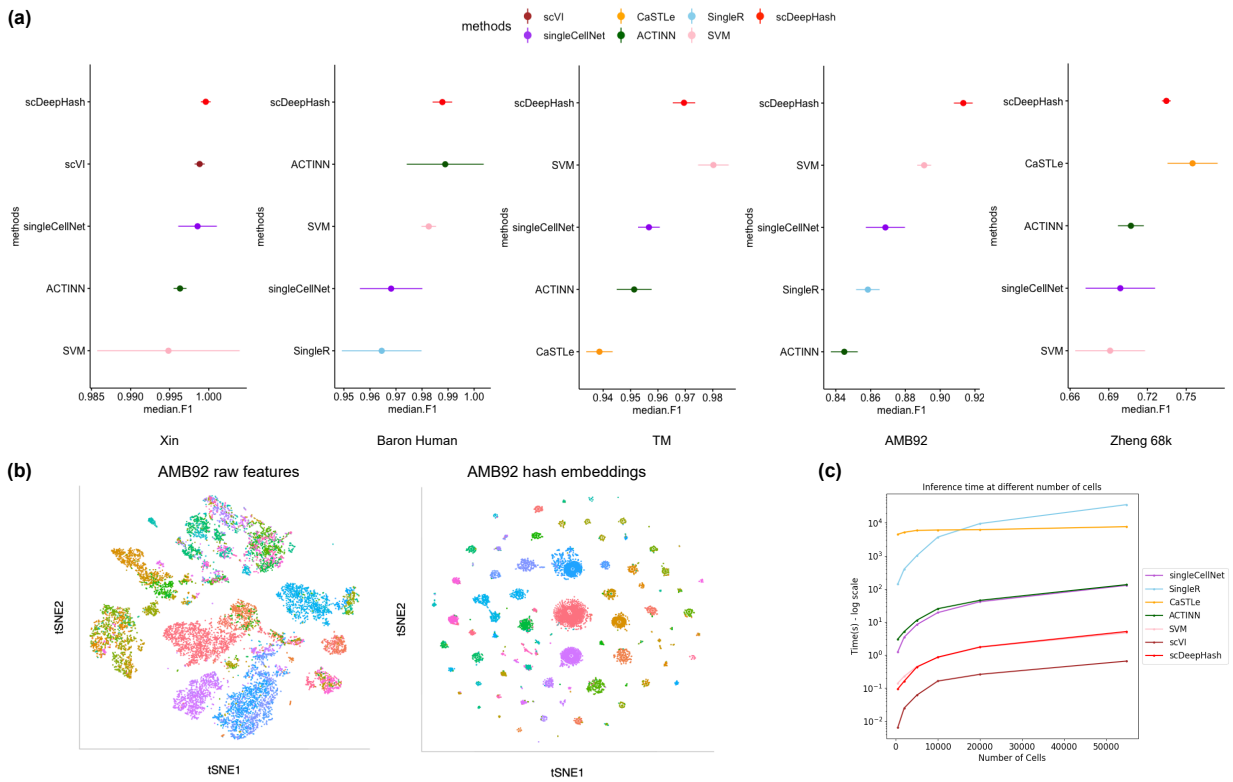

Figure 2: **Annotation experiments**. **(a)**: The annotation performance measured by Median F1 score on the five datasets across all benchmarked methods in boxplot. Each method is tested using the 5-fold cross validation. Aside from scDeepHash, for each dataset we only visualize the top-4 best performing methods. **(b)**: The before- and after-hashing cell data in t-sne visualization of scDeepHash on AMB92. **(c)**: Measured in seconds, the annotation speed on simulated TM dataset across all selected methods.

[38] have relatively high feature dimensions (33k+ genes), and AMB92 also represents a much more granularly annotated dataset with 92 post-processed cell populations; Zheng 68K is the most 'challenging' dataset for hash function learning because the samples are highly inter-correlated [27].

4.1 Performance on Cell-Type Annotation

We select top six state of the art for benchmarking according to the annotation performances on both accuracy and speed [27], including SingleCellNet [22], SingleR [17], ACTINN [40], CaSTLe [41], scVI [26] and SVM.

Annotation accuracy - Measured with F1-median score, Figure 2(a) shows that the annotation accuracy of all methods decreases on datasets with increasing number of cells and features, annotation levels, and correlation. On relatively 'easy' datasets like Xin and Baron Human, all methods perform relatively well. However, when applying on TM, competing methods start to experience a notable drop, with SingleR exhibiting a sharp drop below 90%, and scVI even to 0% (beyond model's capacity, and thus not display). In this case, scDeepHash's accuracy is only second to SVM with a negligible gap (97.3% compared with 98.0%). This shows that when scaling to cell datasets with much larger sample size and gene dimension, scDeepHash can still maintain a good

performance on cell-type annotation. This resonates with the contribution of Weighted Cell-Anchor loss and Quantization loss: high-dimensional and continuous features can be 'compressed' to the ones in discrete hash code while still maintaining the most distinguishable characteristics. As a learning-based method, more cell samples can better help learning the hashing mapping.

On the AMB92 dataset with 92 cell populations (after filtering) and 42k genes, scDeepHash ranks the best by a relatively big margin, surpassing the second best SVM for around 3% (92.0% to 89.1%). This shows that comparing with other state of the art, scDeepHash performs well on handling high-dimensional cell datasets with abundant cell populations. The t-sne visualization of the raw and hashed cell data distribution in Figure 2(b) further verifies such superior performance in an intuitive way: pre-defined cell anchors represent different cell types individually. Each cell anchor aggregates its corresponding hashed cell data into separate, isolated clusters on the hamming space. Based on the underlying methodology, the superiority of scDeepHash's annotation accuracy can be more evident with increasing number of cell populations and gene features.

On the Zheng 68k dataset, all methods encounter a sharp decrease in annotation accuracy. After analyzing the annotation results, we find that the decrease in the scDeepHash annotation accuracy is mainly due to the wrong annotation of the CD4+ T population, of which all the 4 sub-populations have below 40% of correct annotation across all methods. This resonates with the high inter-correlation among the Zheng 68k's cell populations, which makes it very difficult to annotate. Here, scDeepHash ranks second to CaSTLe with a gap of $\sim$1.6%. This is an interesting finding because the performance of CaSTLe is much worse on the other four benchmarked datasets. CaSTLe performs very heavy data pre-processing steps to extract the final 100 features for training on XGBoost, and one of the step is to remove the highly inter-correlated features [41]. We suspect that such step is especially helpful in datasets with abundant highly correlated genes, but for those datasets with less inter-correlation, simply extracting 100 genes of high-dimensional cell data ($\sim$20k original genes) may lose too many distinguishable features and lead to poor annotation performance.

It is worth to note that scDeepHash annotation accuracy is robustly stable on the five selected datasets, with the biggest fluctuation below 1.2%, while the other methods exhibit high variance on different folds within a single dataset. This means that as a learning-based method, scDeepHash can model the latent distribution of a given dataset very well despite the data's complexity, which helps broaden its application to more general use cases of scRNA-seq data with different characteristics.

Annotation Speed - We cross compare the annotation speed among the state of the art and see how they scale when the number of testing samples increases. To keep variables including feature sizes, cell populations, and cell correlation consistent, we choose TM as the candidate dataset and only scale the number of testing samples for variation.

Figure 2(c) shows the big discrepancy regarding the speed capacity of different methods. Excluding methods like SingleR, CaSTLe with extremely slow annotation (10k+ seconds for total TM annotation), other methods generally scale well. To annotate the whole TM dataset, scDeepHash only needs less than 3 secs, which is exceptionally fast. This echos the contribution of the fast computation and the fast labeling strategy described in Sec.3.3. As the fastest method, scVI's annotation time is a magnitude smaller than the second-ranking scDeepHash, but it is at the cost of huge sacrifice on annotation accuracy (Figure 2(c)) since it only uses 1-layer neural network for variational inference [26]. For scDeepHash, most of the annotation time is on passing the high-dimensional raw cell data through the 5-layer neural network for high-quality hash code generation.

Table 2: **Comparison of retrieval accuracy in mAP** against TM & AMB92 varying on top k, where k is based on the average number of ground truth of all cell population in TM and AMB92, respectively.

Method	AMB92			TM		
	mAP@50	mAP@100	mAP@200	mAP@100	mAP@500	mAP@1000
scmap-cell	0.743	0.704	0.663	0.928	0.888	0.864
CellFishing.jl	0.742	0.697	0.649	0.946	0.915	0.897
scDeepHash	**0.947**	**0.947**	**0.946**	**0.956**	**0.947**	**0.943**

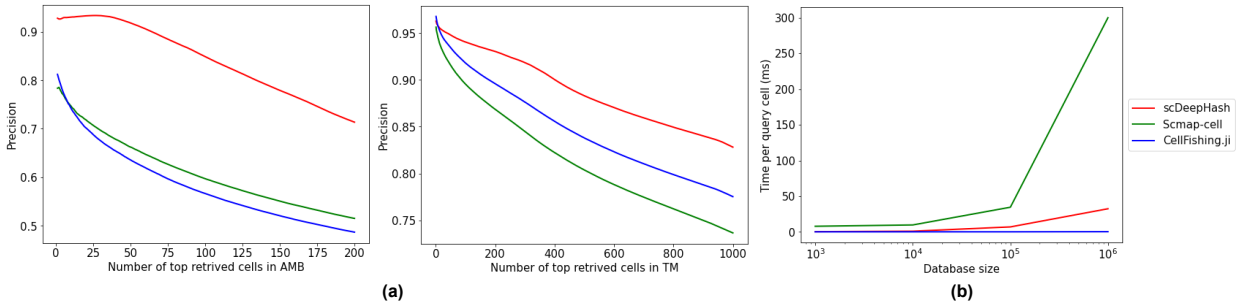

Figure 3: **P@N curves and Retrieval Speed Experiment**. **(a)**: Retrieval accuracy measured by P@N curves on AMB92 and TM datasets across three methods. Each query cell is used to retrieve the top-k most similar cells from their own databases. The average precision across all cell types present in query cells is calculated with respective to each k. **(b)**: Retrieval speed measured in millisecond by time per query, with varying database sizes.

Overall, when considering accuracy and speed as a whole, scDeepHash performs consistently the best against the rest of the state of the art on cell-type annotation.

4.2 Performance on Similar-Cell Retrieval

Here, we benchmark scDeepHash against two retrieval-based annotation methods: CellFishing.jl [42] and scmap-cell [16]. The accuracy of retrieval-based annotation methods is determined by ranking the similarities between query cells and all the reference cells in the database. To benchmark the retrieval accuracy and speed, we made a trivial modification on their final outputs, from returning a cell label to returning most similar cells in their own database for each query.

Retrieval Accuracy - Results in terms of Mean Average Precision (mAP) for cell data retrieval are given in Table 2, in which benchmarking is performed on two relatively complex datasets - AMB92 (with abundant cell types) and TM (with large sample size and high feature dimensions). We vary the top k retrieved cells from database, where k is selected based on the average number of ground truth of all cell population in the cell database, respectively (AMB92 is $\sim$90 on average, and TM is $\sim$600). We observe that scDeepHash achieves the best performance on similar-cell retrieval. Compared with the other two methods, scDeepHash exhibits a sharp mAP increase on both datasets, with more than 20% improvement on AMB92. Meanwhile, as the number of retrieval cells (k) increases, both other two competing methods encounter sharp decrease while scDeepHash can maintain a robustly stable retrieval performance. Such robustness to k means that scDeepHash optimizes the retrieval task and can prioritize the cells in query database from a global perspective, as opposed to the other counterparts that perform much worse in finding the

relatively 'less similar' ground truth given the query cell. Figure 3 (a) shows retrieval accuracy in Precision curves with respect to different numbers of returned samples (P@N). We can also find scDeepHash outperformed the other compared methods by a large margin.

Retrieval Speed - To guarantee the consistency of potential factors (feature size, cell types, etc), similarly, we use simulated TM datasets (upsampling the original TM to create huge database) for retrieval speed test. The experiment is conducted by retrieving top-10 similar cells given 10k query cells varying on the size of reference database. The average retrieval time per query cell is used as the metric to measure the retrieval speed. As shown in Figure 3(b), scDeepHash clearly outperforms scmap-cell, which conducts a linear scan through the entire database for each query cell. Compared with Cellfishing.ji, which uses multi-index hash (MIH) [43] to speed up the searching process, scDeepHash performs slightly slower when the size of reference database is gigantic due to the increased amount of hash codes that needed to be linear scanned through in the hash bucket. MIH divides bit vectors into shorter sub-vectors and indexes them separately into 'hash buckets'. The sub-vectors are later used for k-nearest neighbors search during query time. Even though such design can achieve significant speed improvement, it does sacrifice the retrieval accuracy since searching by sub-vectors is unreliable especially when the size of sub-vector is small.

5 Discussion & Conclusion

In this paper, we proposed a novel learning-to-hash methodology for cell-type annotation and similar-cell retrieval. We further quantitatively demonstrated that scDeepHash has a reliable and efficient performance on large-scale datasets compared with other state of the art. With the ever-increasing scale of scRNA-seq datasets, we believe the potential of scDeepHash is very promising and will keep exploring in this direction.

Our future exploration fall into the following three aspects. First, batch correction will be explicitly addressed. In the real-world application, training data and test data are usually generated under different protocols from different labs. Finding the intrinsic latent features and distributions is very useful in this case. Second, conveniently incorporating new cell data into the existing scDeepHash model is another interesting topic. With ever-increasing scRNA-seq data, new cell subtypes, or even new cell types, it can be time-consuming and cumbersome to iteratively incorporate novel cell data into the existing model with fast training efficiency. A worth-discovering topic is to study the incremental hashing methodology [44] or other online-learning methods to see how such capability can be fulfilled in scDeepHash. Third, the detection of novel cell type can be very useful for both cell-type annotation and similar-cell retrieval. It is highly possible that the true types of query cells are never seen by the underlying model, and thus forcing label assignment or retrieved cells will be unavoidably unreliable. Various methods including scPred [11], CHETAH [45] have already incorporated a similar mechanism into the classification task, while proper confidence boundary is still hard to achieve. We plan to look into this topic and see how to enable scDeepHash to reliably detect unseen cell types.

References

[1] Lähnemann, D. et al. Eleven grand challenges in single-cell data science. Genome biology **21**, 1–35 (2020).

[2] Tang, J., Li, Z. & Zhu, X. Supervised deep hashing for scalable face image retrieval. Pattern Recognition **75**, 25–32 (2018).

[3] Cui, H., Zhu, L., Li, J., Yang, Y. & Nie, L. Scalable deep hashing for large-scale social image retrieval. IEEE Transactions on Image Processing **29**, 1271–1284 (2020).

[4] Zhu, H., Long, M., Wang, J. & Cao, Y. Deep hashing network for efficient similarity retrieval. In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016).

[5] Cao, Z., Long, M., Wang, J. & Yu, P. S. Hashnet: Deep learning to hash by continuation. In Proceedings of the IEEE international conference on computer vision, 5608–5617 (2017).

[6] Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature Biotechnology **36**, 411–420 (2018).

[7] Luecken, M. D. & Theis, F. J. Current best practices in single-cell rna-seq analysis: A tutorial. Molecular Systems Biology **15** (2019).

[8] Zhang, X. et al. Cellmarker: a manually curated resource of cell markers in human and mouse. Nucleic Acids Research **47** (2018).

[9] Franzén, O., Gan, L.-M. & Björkegren, J. L. M. Panglaodb: a web server for exploration of mouse and human single-cell rna sequencing data. Database **2019** (2019).

[10] Zhang et al. Scina: Semi-supervised analysis of single cells in silico. Genes **10**, 531 (2019).

[11] Alquicira-Hernandez, J., Sathe, A., Ji, H. P., Nguyen, Q. & Powell, J. E. scpred: accurate supervised method for cell-type classification from single-cell rna-seq data. Genome Biology **20** (2019).

[12] Kiselev, V. Y., Andrews, T. S. & Hemberg, M. Challenges in unsupervised clustering of single-cell rna-seq data. Nature Reviews Genetics **20**, 273–282 (2019).

[13] Zhang, A. W. et al. Probabilistic cell-type assignment of single-cell rna-seq for tumor microenvironment profiling. Nature methods **16**, 1007–1015 (2019).

[14] Ekiz, H. A., Conley, C. J., Stephens, W. Z. & O'Connell, R. M. Cipr: a web-based r/shiny app and r package to annotate cell clusters in single cell rna sequencing experiments. BMC Bioinformatics **21** (2020).

[15] Fu, R. et al. clustifyr: An r package for automated single-cell rna sequencing cluster classification. F1000Research (2019).

[16] Kiselev, V. Y., Yiu, A. & Hemberg, M. scmap: projection of single-cell rna-seq data across data sets. Nature Methods **15**, 359–362 (2018).

[17] Aran, D. et al. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nature Immunology **20**, 163–172 (2019).

[18] Hou, R., Denisenko, E. & Forrest, A. R. R. scmatch: a single-cell gene expression profile annotation tool using reference datasets. Bioinformatics **35**, 4688–4695 (2019).

[19] Stunnenberg, H. G. & Hirst, M. The international human epigenome consortium: A blueprint for scientific collaboration and discovery. Cell **167**, 1897 (2016).

[20] Lizio, M. et al. Update of the fantom web resource: high resolution transcriptome of diverse cell types in mammals. Nucleic Acids Research **45** (2016).

[21] Cahan, P. et al. Cellnet: Network biology applied to stem cell engineering. Cell **158**, 903–915 (2014).

[22] Tan, Y. & Cahan, P. Singlecellnet: A computational tool to classify single cell rna-seq data across platforms and across species. Cell Systems **9** (2019).

[23] Wang, S. et al. Unifying single-cell annotations based on the cell ontology. BioRxiv (2019).

[24] Lin, Y. et al. scclassify: sample size estimation and multiscale classification of cells using single and multiple reference. Molecular Systems Biology **16** (2020).

[25] Wagner, F. & Yanai, I. Moana: A robust and scalable cell type classification framework for single-cell rna-seq data. BioRxiv (2018).

[26] Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. Nature Methods **15**, 1053–1058 (2018).

[27] Abdelaal, T. et al. A comparison of automatic cell identification methods for single-cell rna-sequencing data. Genome Biology (2019).

[28] Brbić, M. et al. Mars: discovering novel cell types across heterogeneous single-cell experiments. Nature Methods **17**, 1200–1206 (2020).

[29] Stuart, T. et al. Comprehensive integration of single-cell data. Cell **177**, 1888–1902 (2019).

[30] Yuan, L. et al. Central similarity quantization for efficient image and video retrieval. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020).

[31] Yarlagadda, R. K. & Hershey, J. E. A hadamard matrix. Hadamard Matrix Analysis and Synthesis 3–3 (1997).

[32] None. Kronecker product. Matrix Calculus and Kronecker Product with Applications and C++ Programs 55–107 (1997).

[33] Cao, Y., Long, M., Liu, B. & Wang, J. Deep cauchy hashing for hamming space retrieval. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018).

[34] Cui, Y., Jia, M., Lin, T.-Y., Song, Y. & Belongie, S. Class-balanced loss based on effective number of samples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019).

[35] Xin, Y., Kim, J. & Ikamoto, H. Rna sequencing of single human islet cells reveals type 2 diabetes genes. Cell Metabolism **24**, 608–615 (2016). URL https://www.sciencedirect.com/science/article/pii/S155041311630434X.

[36] Baron, M. et al. A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. Cell Systems **3**, 346–360.e4 (2016). URL https://www.cell.com/cell-systems/fulltext/S2405-4712(16)30266-6.

[37] Nicholas, S., Jim, K. & Norma F, N. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. Nature **562**, 367–372 (2018).

[38] Tasic, B. et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature **563**, 72–78 (2018).

[39] Zheng, G. X. Y. et al. Massively parallel digital transcriptional profiling of single cells. Nature Communications **8** (2017). URL https://www.nature.com/articles/ncomms14049.

[40] Ma, F. & Pellegrini, M. Actinn: automated identification of cell types in single cell rna sequencing. Bioinformatics (2019).

[41] Lieberman, Y., Rokach, L. & Shay, T. Castle – classification of single cells by transfer learning: Harnessing the power of publicly available single cell rna sequencing experiments to annotate new experiments. PLOS ONE **13**, 1–16 (2018). URL https://doi.org/10.1371/journal.pone.0205499.

[42] Sato, K., Tsuyuzaki, K., Shimizu, K. & Nikaido, I. Cellfishing. jl: an ultrafast and scalable cell search method for single-cell rna sequencing. Genome biology **20**, 1–23 (2019).

[43] Norouzi, M., Punjani, A. & Fleet, D. J. Fast exact search in hamming space with multi-index hashing. IEEE transactions on pattern analysis and machine intelligence **36**, 1107–1119 (2013).

[44] Wu, D., Dai, Q., Liu, J., Li, B. & Wang, W. Deep incremental hashing network for efficient image retrieval. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019).

[45] Kanter, J. K. D., Lijnzaad, P., Candelli, T., Margaritis, T. & Holstege, F. C. Chetah: a selective, hierarchical cell type identification method for single-cell rna sequencing. Nucleic Acids Research (2019).