Deep Generative Model for Auto-annotation in Single-cell Analysis

Lin Zhu
zhulin1@shanghaitech.edu.cn

Jie Wang
wangjie1@shanghaitech.edu.cn

Abstract

With the increasing scale of single-cell RNA sequencing data, cell type annotation has to be automated. In this project, we implement a deep generative model for semi-supervised learning to encode the gene expressions of cells into the latent space and annotate the cell types based on the posterior inference. Compared with auto-annotation methods based on the raw data, our model can reduce time and still keep a high accuracy when handling large-scale data. Besides, our semi-supervised learning model can outperform the supervised method such as SVM when only a few labeled data is available in a dataset.

1. Introduction

Recent developments in high-throughput sequencing technologies enable the analysis of single-cell RNA data as an ascendant tool for analyzing complex tissues. Cell type identification is a common step in single-cell data analysis.

With a given dataset, the typical pipeline of cell type identification includes data processing, dimension reduction, clustering, and annotation. In the dimension reduction phase, linear methods like PCA and non-linear methods like t-SNE and UMAP are popular tools. The following clustering step is usually unsupervised and based on the similarity of gene expression profiles between cells in a low-dimensional space. The last annotation step usually involves inspecting marker genes of each cluster manually, which is biased and time-consuming. Besides, the lack of standardized processing and automation in a typical pipeline makes it inefficient for a growing data scale.

In this case, the cell type auto-annotation is necessary. In the real situation, it is common that only a subset of cells can be confidently labeled in a dataset while most auto-annotation methods require a large scale of labeled cells as a training set. This project will use the semi-supervised generative model to solve the auto-annotation problem in a single dataset with few labeled data. Specifically, we expect that using the gene expression profiles of cells and a few labels as input for our model can predict cell type for the rest unlabelled cells.

To evaluate our model, we will mainly consider the weighted accuracy because the ratio of different cell types is different in a dataset. The weighted accuracy refers to "the percent of cells that were correctly assigned to their correct label, averaging over all labels"[14].

2. Related Work

To overcome the shortages of the manual annotation methods, many auto-annotation methods have been proposed[1]. Generally, they can be broadly divided into two groups. The first group is using part of datasets labeled with correct cell populations to train the classifiers. Another group is prior knowledge-based, which requires either a marker gene set or a pre-trained classifier for selected cell populations[3].

Moana[12] uses a linear SVM to classify the cells. Scmapcell[5] adopts the kNN model. CaSTLe[6] and singleCellNet[10] are based on Random Forest. Although lots of methods use the traditional machine learning methods, the deep learning-based methods emerge in this field recently. ACTINN[8] applies a neural network with three fully connected layers to classify the cells. scVI[7] uses the VAE model to capture the latent embedding of gene expression profiles and annotation can be done based on the latent variances. However, scVI relies on down-stream tools to annotate the cell type and ACTINN requires sufficient training samples. SCANVI[14] is an extended version of scVI to annotate the cell type with semi-supervised learning.

Therefore, we try to propose a model based on neural network to annotate cell types of single-cell datasets with few labeled data. Semi-supervised learning is suitable for the situation that only a few labels are available. Besides, the work of scVI proved that generative model is a suitable method for single cell gene expression data.

3. Data

In this project, three datasets are used. Two of them are datasets of mouse cells form different tissues we curated manually and the other one is a simulation dataset adopted form SCANVI[14] with 20000 cells and 2000 genes. The specific information of the datasets we used can be found in Table 1.

Our dataset constructed based on the Tabula Muris[9] dataset (TM dataset), which is a compendium of single-cell transcriptomic data from the model organism Mus musculus that comprises more than 100,000 cells from 20 organs and tissues. Our protocol of dataset is based on the 10x Genomics.

For the TM dataset, we merged cells come from all tissues. While we merged them, we found some specific channels 10X_P8_N (N $\in$ [12,13,14,15]) have more than 700,000 cells, it didn't match the amount described in their document, so we didn't use these data. Meanwhile, we drop same cells when we merged. The total number of merged cells is 40,528, the number of genes is 23,433. Then we pre-process the dataset, we use the Scanpy[13] package to do quality control on our dataset to regress out some low-quality cells and genes, Scanpy is a popular single cell analysis tool in python. We filtered out cells which are detected less than 200 genes and genes which are detected from less than 3 cells. Besides, we remove cells whose total counts more than 7,000. After quality control, the shape of dataset becomes 18710 $\times$ 40479 (gene $\times$ cell). Furthermore, according to the requirement of following experiments, we filtered out the unlabeled cells. In this way, we constructed the TM Dataset 1 of size 18710 $\times$ 39264 (gene $\times$ cell) and the kinds of cell labels are 51. Additionally, by selecting the first 4999 high variable genes using Scanpy from TM dataset1 we get the TM Dataset 2.

Dataset	Shape (Gene $\times$ Cell)	Cell Types
TM Dataset 1	18710 $\times$ 39264	51
TM Dataset 2	4999 $\times$ 39264	51
Simulation	2000 $\times$ 20000	5

Table 1. The specific information of datasets.

After collecting and preprocessing the dataset, we use the louvain clustering method and UMAP tool to visualize the TM Dataset 1, which is shown in Fig 1.

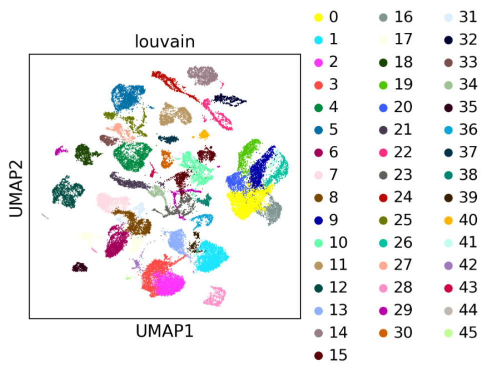

Figure 1. The clustering result of the TM Dataset 1 using louvain method and UMAP visualization.

4. Methods

We construct our method based on the SCANVI[14] model, which is a semi-supervised variational autoencoder (VAE) model. We further analysis the problems of VAE, we use the Maximum Mean Discrepancy (MMD)-based Optical Transport metrics to replace the KL divergence in VAE to evaluate the distance between model distribution and target distribution. This section will be divided into two parts, the first part will introduce the semi-supervised VAE model, and the second part will introduce the MMD-based method.

4.1. Semi-supervised VAE

4.1.1 Basic Model Architecture

Our semi-supervised generative model based on the architecture proposed by Kingma[4] and used in SCANVI[14]. It is a stacked generative model consisting of a vanilla VAE as M1 and a semi-supervised conditional VAE as M2. The architecture of model is shown in Fig 2.

The M1 takes the raw gene expression matrix as input. After it has been fully trained, we can use the the "encoder" to transfer the gene expression matrix x into the latent space defined by the z variable. The latent space defined by z should capture some useful features about the raw gene expression matrix such that it should be more separable. The latent space can be used for supervised learning with labeled cells as training datasets. With the supervised classifier, the label of unlabeled cells can be predicted. The generative model used in M1 is:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I}); \quad p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x};\mathbf{z},\boldsymbol{\theta}) \tag{1}$$

However, with only a tiny fraction of data labeled, the performance of supervised learning may not be very well. In order to make use of both labeled and unlabeled data, semi-supervised learning is necessary. Besides, the cell type information of labeled data has not been used in the M1 model which is a waste of label information. The M2 model is an extended VAE for semi-supervised learning, which takes the label information into account. The conditional VAE of M2 has a cell type variable c as a latent variable in addition to the continuous latent variable z. Since most of the labels are unknown, the model has a classifier to predict the missing labels based on the inferred posterior distribution $p(c|x)$. The generative model in M2 can be expressed as follow:

$$\begin{aligned} p(\mathbf{x}|c,\mathbf{z}) &= f(\mathbf{x};c,\mathbf{z},\theta) \\ p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|0,I) \\ p(c|\pi) &= \mathrm{Cat}(c|\pi) \\ p(\pi) &= \mathrm{SymDir}(\alpha), \end{aligned} \tag{2}$$

where c represents the cell type label and $\mathrm{Cat}(c|\pi)$ is the multinomial distribution.

The stacked generative model uses the latent representation $z1$ generated by M1 as the input of M2. In our model, both the encoders and the decoders are two-layer neural networks that consist of a fully connected layer with 128 nodes and an output layer with 10 nodes.

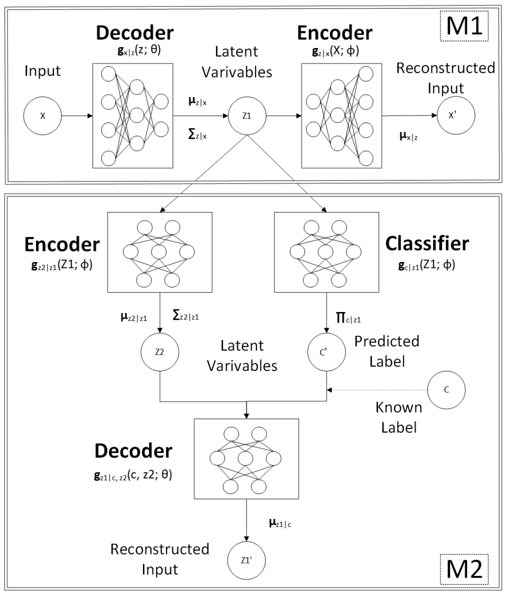

Figure 2. The architecture of stacked model.

4.1.2 Semi-supervised VAE

In addition to gene expression information and cell label information, our model has also included the batch information and library information, because gene expression data from different batches and different library sizes may contain sequencing technical noises. Therefore, We define the latent variance generating process of cell n as follow.

c_n represent the cell type of cell n and C is the number of cell types.

$$c_n \sim \text{Multinomial}(\frac{1}{C}) \tag{3}$$

u_n is a low-dimensional random vector describes cell n within its cell type.

$$u_n \sim \mathcal{N}(0, I) \tag{4}$$

Combining above cell type info encoded by c_n and u_n, we can get the new low-dimensional vector z_n. As commonly defined in variational autoencoders, z follows the standard multivariate normal prior so that it can be reparameterized into any arbitrary multivariate Gaussian random variable, which can simplify the calculation.

$$z_n \sim \mathcal{N}\left(f_z^\mu\left(u_n, c_n\right), f_z^\sigma\left(u_n, c_n\right)\right) \tag{5}$$

ℓ_n encodes a scaling factor of cell n. ℓ_μ, ℓ_σ are set to be the empirical mean and variance of the log-library size per batch. $\ell_\mu \ell_\sigma \in \mathbb{R}_+^B$ parameterize the prior for the scaling factor (on a log scale) and B denotes the number of batches.

$$l_n \sim \text{logNormal}\left(l_\mu, l_\sigma^2\right) \tag{6}$$

Conditional distribution $x_{ng}|z_n, l_n, c_n$ can be calculated with s_n representing the dataset information and $\theta \in \mathbb{R}_+^G$ encoding the gene specific inverse dispersion. $x_{ng}|z_n, l_n, c_n$ is zero-inflated negative binomial distribution.

$$\begin{gathered}
w_{ng} \sim \text{Gamma}\left(\int_w^g\left(z_n, s_n\right), \theta_g\right) \\
y_{ng} \sim \text{Poisson}\left(l_n w_{ng}\right) \\
h_{ng} \sim \text{ Bernoulli }\left(\int_h^q\left(z_n, s_n\right)\right) \\
x_{ng} = \begin{cases} y_{ng} & \text{if } h_{ng} = 0 \\ 0 & \text{otherwise} \end{cases}
\end{gathered} \tag{7}$$

As the equations show, the model considers the cell type, gene expression, batch size and library size when encoding gene expression into the latent space. This model will reduces the dimension of the input gene expression matrix and output a low-dimensional matrix as the latent encoding of gene expression values.

4.1.3 Posterior inference

With the latent variable $x_{ng}|z_n, l_n, c_n$, we can perform approximate inference use the stochastic gradients variational Bayes estimator by neural networks. The variational distribution factorizes are assumed as:

$$\begin{aligned}
& q_\Phi\left(c_n, z_n, l_n, u_n | x_n, s_n\right) = \\
& q_\Phi\left(z_n | x_n\right) q_\Phi\left(c_n | z_n\right) q_\Phi\left(l_n | x_n\right) q_\Phi\left(u_n | c_n, z_n\right)
\end{aligned} \tag{8}$$

Since the c_n can be observed or non-observed, the evidence lower bound (ELBO) can be defined as the sum of two variational lower bounds[4] $\text{ELBO} = \mathcal{L} + \mathcal{U}$.

The approximate posterior $q_\Phi\left(c_n | z_n\right)$ can be used as the classifier to annotate cell types.

4.2. WAE-MMD

After analysis the mechanism of VAE, we can find that it has some problems, and some papers also point them out. Google Brain proposed the Wasserstein Auto-Encoders (WAE) [11] in 2017, they analysis the problems of VAE. In the Fig 3, the bottom half represents data space, and the top half represents the latent space. The circles and squares in data space represent true data points and reconstructed data points respectively. The circles in latent space represent data distribution and triangles stands for the data points sampled from corresponding distribution, the solid circle refers to the prior Gaussian distribution and the dotted circles means encoded data distribution in latent space. The arrows refer to the conditional distributions. In VAE, every dotted circle is forced to match the solid circle. That is, to match P_Z, prior Gaussian distribution, which leads to the intersection of different encoded data distribution. The forced match will cause problems during reconstruction[11, 2]. To overcome this shortage, WAE uses a continues mixture distribution $Q_Z := \int Q(Z|X)dP_X$ to match P_Z, which will make the latent samples keep far away from each other. Consequently, it can get better reconstruction than VAE[11].

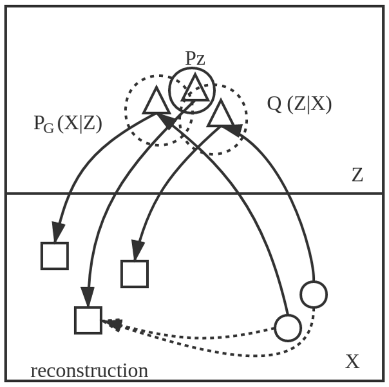

Figure 3. The simplified mechanism of VAE.

Both loss functions of VAE and WAE are constructed of two terms, the reconstruction loss and the regularizer penalizing discrepancy between prior distribution P_Z and distribution encoded by encoder Q[11]. The mainly difference between them is the regularizer loss. Different with the KL divergence of VAE, WAE considers from the Optimal Transport (OT) point of view to measure the distance between the true data distribution and a latent variable model. The OT cost can provide a much weaker topology, which is better than strong notions of distance, such as f-divergence, including KL divergence and JS divergence. WAE proposes two different regularizers, one is based on GANs, another one is based on the MMD. GANs-based method uses JS divergence and trains a discriminator in the latent space. However, we choose to use MMD-based method as our regularizer, because it can perform well when matching high-dimensional standard normal distribution P_Z. Meanwhile, it's a min-min optimization problem. Our adapted loss function as follow:

$$Loss = cl\_loss + recons\_loss + reg\_loss(MMD) \quad (9)$$

Like the loss function of the original SCANVI model, our loss function consists of classification loss (cl_loss), reconstruction loss ($recons\_loss$) and regularizer factor (reg_loss). We replace the prior KL divergence with MMD-based method as the regularizer factor.

As the paper[11] shows, the expression of MMD method is:

$$\begin{aligned}&\mathrm{MMD}_k\left(P_Z, Q_Z\right) = \\ &\left\| \int_{\mathcal{Z}} k(z,\cdot)dP_Z(z) - \int_{\mathcal{Z}} k(z,\cdot)dQ_Z(z) \right\|_{\mathcal{H}_k}\end{aligned} \quad (10)$$

k is a positive-definite reproducing kernel: $\mathcal{Z} \times \mathcal{Z} \to \mathcal{R}$, and $\mathcal{H}_k$ is the RKHS of real valued functions mapping $\mathcal{Z}$ to $\mathcal{R}$[11]. As WAE depicts in their paper, they found the Inverse Multiquadratics (IMq) kernel $k(x,y) = C/\left(C + \|x-y\|_2^2\right)$ is better than Radial Basis Function (RBF) kernel, so we adopt the IMq kernel in the MMD formula.

5. Experiments

We use the SVM model as our baseline and we compare the original semi-supervised model SCANVI with our adapted DGM4SCA model on different datasets. Table 2 shows the weighted annotation accuracy of different models on different datasets. The weighted annotation accuracy refers to calculating accuracy for each cell type first and average across cell types[14]. The description of datasets can be found in Data section. In our experiments, we sample the fixed-number (parameter CL in table 2) cells from each cell type in a dataset as labelled data and the remaining cells as unlabelled data.

Datasets\Models		SVM	SCANVI	DGM4SCA
TM Dataset 2 (HV genes)	CL=10	0.88	0.76	0.75
	CL=5	0.89	0.72	**0.73**
	CL=20	0.89	0.81	**0.83**
TM Dataset 1	CL=10	0.76	0.65	0.58
Simulation	CL=10	0.46	0.58	**0.59**

Table 2. The comparison of experimental results.

According to the Table 2, we can find that our DGM4SCA model get improved compared with SCANVI model on the majority of datasets. However, both SCANVI and DGM4SCA are worse than baseline on the TM dataset, and better than baseline on the simulation dataset. That is aroused by the difference between datasets. We analysis the datasets in Fig 4 and Fig 5, the x-axis represents different cell types and y-axis represents the number of cells. It can be observed that the small amount of cells take a dominant position in Fig 4, and Fig 5 conversely. In this case, if given same samples with labels, SVM works well on small-sized cells. Consequently, SVM can work well on TM dataset, but fail on big amount of cells with few labels. In contrast,

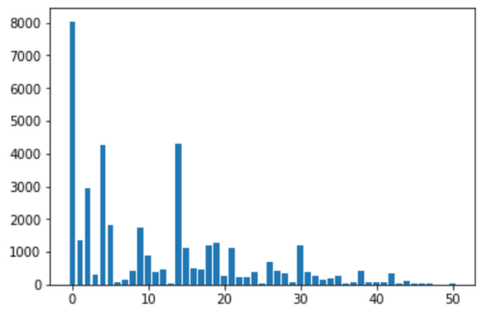
Figure 4. The number of cells distribution in TM dataset.

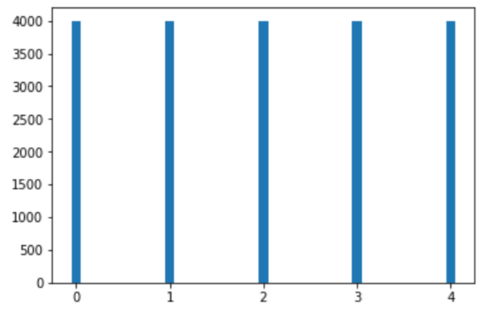
Figure 5. The number of cells distribution in simulation dataset.

our model can handle this problem and work well on simulation dataset.

It is worth noting that the big gap of time between SVM and our methods, and it will increase with the extent of scale of datasets. One advantage of our model is that it can benefit form the neural network to handle large scale of dataset. When the size of gene expression matrix increases form 2000 $\times$ 20000 to 18710 $\times$ 39264, the training time of SVM increases 640 times from 3 seconds to 32 minutes. However, our model only increases fourfold from 7 minutes to 30 minutes. As the size of dataset increasing, the time consuming by SVM increasing rapidly while the neural network has a better scalability.

Fig 6 and Fig 7 show the accuracy change on simulation dataset and TM dataset during 100 epochs. Fig 8 and Fig 9 show the curves of reconstruction loss during 100 epochs. As shown in the figures, the loss drops quickly and the accuracy increases rapidly which means that our model can converge in a short time. This is consistent with the conclusion in SCANVI that the model can converge quickly. Besides, given a larger dataset, the model can converge faster.

Fig 10 and Fig 11 visualize the latent space of simulation dataset and TM dataset. Compared with the raw gene expression data visualized in Fig 1, the latent space data separated more clearly by cell type. The distances of different cell types become larger and the distances of same cell types get smaller. This means our model captured the label-related features of the gene expression matrix and encoded

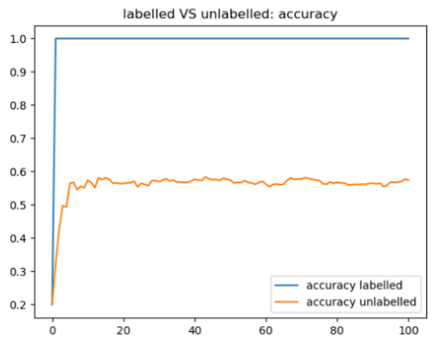

Figure 6. Accuracy change on simulation dataset.

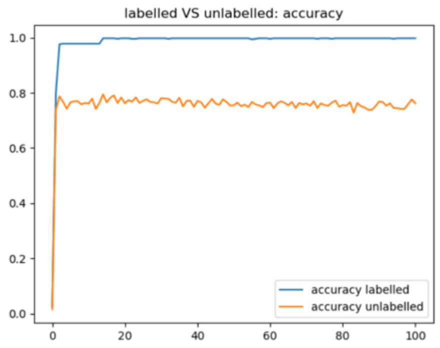

Figure 7. Accuracy change on TM dataset.

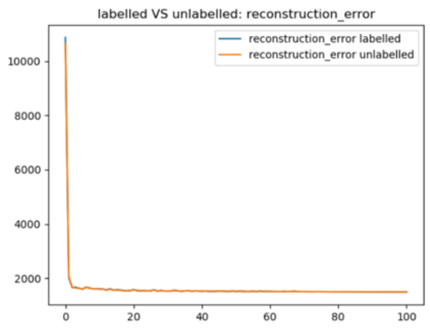

Figure 8. Reconstruction loss change on simulation dataset.

them to latent space efficiently.

6. Conclusion

In this project, we apply the deep generative model to auto-annotating cell type in single-cell data. Instead of supervised methods, our semi-supervised method focus on the datasets with a tiny fraction of labeled data.

As shown in the results, supervised methods represented by SVM cannot handle the dataset without sufficient label information. However, our semi-supervised method can make full use of both labeled and unlabeled data which shows more superiority on datasets lacking of la-

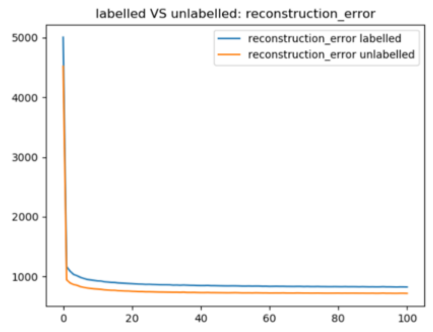

Figure 9. Reconstruction loss change on TM dataset.

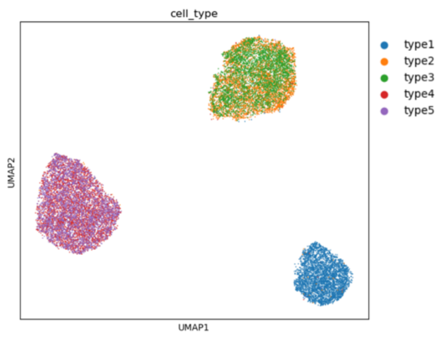

Figure 10. Visualization of latent space on simulation dataset.

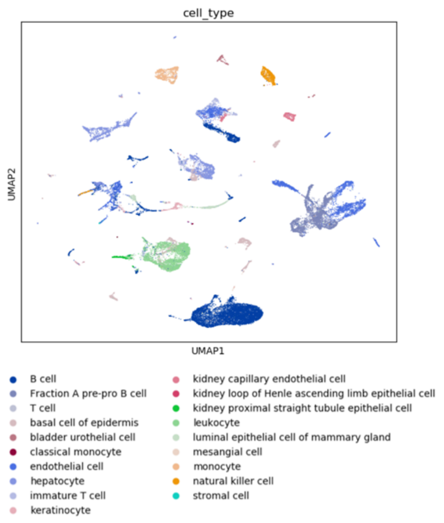

Figure 11. Visualization of latent space on TM dataset.

bels. As the accuracy in Table 2 shows, compared with the original VAE-based semi-supervised method represented by SCANVI, our WAE-based method improve the performance slightly. The increasing of accuracy indicts that MMD-based regularizer is useful.

In conclusion, the project proposed a MMD-extended deep generative model for semi-supervised learning for single-cell auto-annotation. Our model does outperform the original semi-supervised model SCANVI while it fails to achieve higher accuracy than SVM in unbalanced dataset such as our TM dataset. Fixed the problem of unbalanced dataset, the model may achieve a better performance.

7. Supplement

Data and code availability: `https://github.com/WJie12/DGM4SCA`.

References

[1] T. Abdelaal, L. Michielsen, D. Cats, D. Hoogduin, H. Mei, M. J. Reinders, and A. Mahfouz. A comparison of automatic cell identification methods for single-cell rna-sequencing data. *bioRxiv*, page 644435, 2019.

[2] O. Bousquet, S. Gelly, I. Tolstikhin, C. Simongabriel, and B. Schoelkopf. From optimal transport to generative modeling: the vegan cookbook. *arXiv: Machine Learning*, 2017.

[3] A. Ianevski, A. K. Giri, and T. Aittokallio. Fully-automated cell-type identification with specific markers extracted from single-cell transcriptomic data. *bioRxiv*, page 812131, 2019.

[4] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[5] V. Y. Kiselev, A. Yiu, and M. Hemberg. scmap: projection of single-cell rna-seq data across data sets. *Nature methods*, 15(5):359, 2018.

[6] Y. Lieberman, L. Rokach, and T. Shay. 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(10):e0205499, 2018.

[7] R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053, 2018.

[8] F. Ma and M. Pellegrini. Actinn: automated identification of cell types in single cell rna sequencing. *Bioinformatics*, 2019.

[9] N. Schaum, J. Karkanias, N. F. Neff, A. P. May, S. R. Quake, T. Wyss-Coray, S. Darmanis, J. Batson, O. Botvinnik, M. B. Chen, et al. Single-cell transcriptomics of 20 mouse organs creates a tabula muris: The tabula muris consortium. *Nature*, 562(7727):367, 2018.

[10] Y. Tan and P. Cahan. Singlecellnet: a computational tool to classify single cell rna-seq data across platforms and across species. *Cell systems*, 9(2):207–213, 2019.

[11] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. *arXiv: Machine Learning*, 2017.

[12] F. Wagner and I. Yanai. Moana: A robust and scalable cell type classification framework for single-cell rna-seq data. *BioRxiv*, page 456129, 2018.

[13] F. A. Wolf, P. Angerer, and F. J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):15, 2018.

[14] C. Xu, R. Lopez, E. Mehlman, J. Regier, M. I. Jordan, and N. Yosef. Harmonization and annotation of single-cell transcriptomics data with deep generative models. *bioRxiv*, page 532895, 2019.