Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 18 December 2021
Gene Expression

Subject Section

Imputation of Single-Cell RNA-Sequencing Data by Neural Collaborative Filtering

Bryce Blinn [1,*], Vadim Kudlay [1,*], Alexander Young [1,*], and Peter Zhu [1,*]

[1] Department of Computer Science, Brown University, Providence, 02912, USA

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Single-cell RNA sequencing suffers from dropout events that result in false, or "technical" zero expression values. Oftentimes, genes with low expression levels are simply not captured with current techniques. While deep learning models which attempt to detect these technical zeroes and impute values for them are beginning to emerge, none have yet leveraged the concept of collaborative filtering, originally developed as a means of making recommendations to users with similar sets of preferences. This paper seeks to implement two collaborative filtering architectures - a matrix factorization approach with nonlinear consideration of cell-wide counts as well as a multinomial variational autoencoder (Mult-VAE) solution which was originally proposed to reason specifically with count data.

Results: We evaluated NCFlix on simulated Splatter datasets with 2 and 6 datasets, as well as human PBMC data. While at this point in time NCFlix does not exceed the performance of DCA or the standard of the true counts, the model, particularly the Mult-VAE iteration, comes remarkably close, especially when run upon real biological data, with the PCA plot of imputed output having similar distributions and cohesion, and marginally worse spread.

1 Introduction

Single-cell RNA sequencing has revolutionized the field of genomics by permitting the analysis of gene expression at an unprecedented resolution, and is now considered the gold standard for defining cell types and their phenotypes. However, the data suffer from high fractions of observed zero values, known as 'dropout.' While some of these occurrences may be attributed to true absence of expression, many may arise from methodological noise. The efficiency with which instances of mRNA are captured, converted into cDNA, and amplified is unclear but can range anywhere from 10 to 40 percent (1), meaning some genes which are expressed at low levels may not be detected by sequencing technology. Moreover, transcriptional bursting, in which short pulses of high transcriptional activity are followed by longer silent refractory periods, and cell-cycle phases, lead to temporal fluctuation. These phenomena manifest in the form of artificial zeroes both systemic - for example, sequence-specific mRNA becoming degraded during cell lysis - and by chance.

Two approaches have generally been taken to handle the problem of sparsity: statistical models which quantify uncertainty by inherently capturing sparsity, sampling variation, and other noise; and methods which 'impute' values for technical zeroes. Since many applications are unable to handle sparse count data, the latter is becoming increasingly valuable.

Deep learning has become a powerful tool for genomics problems, from function and folding predictions from DNA sequences and histone modifications, to regulatory dependency modeling from chromatin, to, yes, de-noising and imputation. We have already seen its success in HiCPlus/hicGAN and DCA, but there is still significant room for improvement.

In this paper, we propose NCFlix, which is the first approach to utilize a deep learning architecture known as neural collaborative filtering (NCF) to impute scRNA-seq zeroes. Traditionally, NCF has found use in recommender systems such as mineral and environmental exploration, financial data, and electronic commerce, by virtue of the fact that it makes predictions about a 'user' by collecting information about the 'preferences' about many other users, operating under the assumption that when two users 'A' and 'B' have the same preference for one issue, 'A' is likely to also have the same preference as 'B' for another issue. Naturally, there

picture(0,0)(-35,0)(1,0)30 (0,35)(0,-1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,35)(0,-1)30 picture

is strong reason to believe that NCF will also be excellent for scRNA-seq imputation, wherein the 'users' are the cells and the 'preferences' are expression values for each gene, since cells tend to be clustered in cell types or cell states with similar expression profiles. scRNA-seq data also has the advantage of avoiding many of the issues that plague traditional NCF applications, such as the synonym problem (different users having the same 'name') and shilling attacks. We will apply NCF, and time permitting, variants utilizing NCF combined with matrix factorization (MF) and variational autoencoders (VAEs), on several large datasets of mouse brain/marrow cells and human PBMCs.

2 Related Work

2.1 Existing Methods of scRNA-seq Imputation

Current methods for imputation generally fall in one of the following categories:

- Model-based imputation methods liken the prediction of entries to a problem of selecting random samples from an associated random model. The current state-of-the-art is SAVER (3) and SAVER-X (4), which parameterize a negative binomial to fit the unique molecular identification (UMI) generation process and incorporates additional structures to boost predictive power.
- Data-smoothing models define a 'similarity' between cells and adjust the expression values based on the clustering or diffusion. An example is MAGIC (5), which uses diffusion across the nearest-neighbor graph.
- Data-reconstruction models define a latent space representation for the cells, and accomplish this with MF (e.g. principal component analysis, or PCA), or machine learning approaches, particularly deep learning. Instances of the former include pCMF (7), a probability count MF with the Poisson model, and instances of the latter include AutoImpute (an autoencoder model), DCA (deep count AE), DeepImpute (AE parallelized on gene subsets), and scVI (VAE) (8).

2.2 Existing Architectures for NCF

We are interested in selecting an NCF architecture and comparing its performance on this imputation problem to existing state-of-the-art methods. We can adapt any of the below architectures, none of which have yet been applied to scRNA-seq data, by treating our cells as 'users' and their genes as 'items' or 'issues.'

- NMF architectures (10) treat the embeddings of the users and the items as learned functions based on previous interactions.
- GNN-based architectures (11) treat the set of interactions between users and items as a graph and run message passing layers over this graph to learn embeddings over the users and items.
- GAN-based architectures (12) learn to predict user-item interactions by including a generator for likely user-item interactions as well as a generator for *unlikely* user-item interactions.
- VAE-based architectures (13) learn to encode interaction data of a single user into a vector samples from a Gaussian distribution and then decode this vector into a set of probabilities over all item interactions.

Recent evaluations in the field as detailed in the paper "A systematic evaluation of single-cell RNA-sequencing imputation methods" (2) have noted that many NCF formulations have trouble out-performing simpler techniques, such as knn-based recommendation approaches, on key baseline datasets (e.g. MovieLens and Amazon Music). One of the noted exceptions to this was Mult-VAE (13), which was found to significantly improve over simpler baseline techniques in some instances.

3 Methods

3.1 Simple Matrix Factorization

When considering the problem space of UMI counts, the problem environment is a matrix of counts. Each entry specifies the number of molecular identifiers observed for a specific gene type within a specific cell type; as such, the two dimensions of cell-type and gene-type are observed in the count matrix. For the sets of C cells and G genes types, the matrix of UMI counts $y \in \mathbb{R}^{C \times G}$ is thereby construct-able. This matrix is obtained via experimentation as discussed previously, but has many zero-value entries $x_{zero} = \{x \in \mathbb{N}^C \mathbb{N}^G \mid y(x) = 0\}$ that we would like to be imputed.

A common approach is to employ *matrix factorization* techniques to find low-rank matrix factors that can be used to generate a reasonable prediction matrix $\hat{y}$ that resembles y at its empirically-observed 'ground-truth' entries. These matrices - commonly referred to as U and V in some papers - is now commonly implemented as embeddings that map indices to vectors; thereby, we will use $E_G : \mathbb{N}^n \to \mathbb{R}^{n \times d}$ and $E_C : \mathbb{N}^n \to \mathbb{R}^{n \times d}$ to represent embedding layers to map cell and gene indices to d-dimensional vectors.

The core component of matrix factorization then involves optimizing E_c and E_g to minimize the difference between the known counts $y_{\bar{x}_{zero}}$ and their predicted analogues $(E_c(x_c)E_g(x_g)^T)_{\bar{x}_{zero}}$. The optimization trains the embeddings of E_c and E_g to have an element-wise relationship that is linear (multiplicative) in nature. The full end-to-end model formulation of basic matrix factorization (BMF) is then

$$E_g(x_c)E_g(x_g)^T + B_c(x_c) + B_g(x_g) \to \hat{y}_{BMF}(x_c, x_g) \sim y(x_c, x_g)$$

where B_c and B_g are optional $\mathbb{N}^n \to \mathbb{R}^n$ embedding layers that produce bias approximations for each class.

3.2 Simple NCF and Neural Matrix Factorization

In contrast to this, NCF tries to learn U and V in a way that does not force a linear relationship. It does so by training up the factor embeddings to be associated by some sort of learned similarity which can be reasoned with e.g. by a multi-layer perceptron. The resulting factors are thereby trained up to have a non-linear relationship which can be important in certain domains. The simplest variation of this using an n-layer MLP and designated embedding layers E_c and E_g can be trained end-to-end and defines the hypothesis as:

$$\begin{bmatrix} E_c(x_c) \\ E_g(x_g) \end{bmatrix} \to \text{Dense}_1 \to \cdots \to \text{Dense}_n \to \hat{y}_{NCF} \sim y$$

Variations of this approach have been introduced with various features. NeuMF organizes a generalized MF approach and a NCF approach in an ensemble setting by training up separate factor embeddings for both and deciding between the two results using an additional dense layer that reasons with the concatenation of two methods' outputs. In doing so, the technique can capture both linear and non-linear relationships in its prediction. Effectively, the NeuMF prediction is defined as following, where α is a hyperparameter dictating the weight of the MF result relative to the NCF approach:

$$\begin{bmatrix} y_{BMF} \\ y_{NCF} \end{bmatrix}^T \begin{bmatrix} \alpha \\ 1 - \alpha \end{bmatrix} \to Dense \to y_{NeuMF} \sim y$$

picture(0,0)(-35,0)(1,0)30 (0,-35)(0,1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,-35)(0,1)30 picture

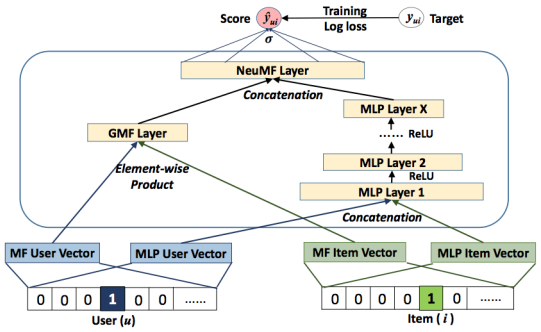

Fig. 1. Neural matrix factorization model, which combines matrix factorization and multi-layer perceptrons under the NCF framework

Our collaborative filtering approach straddles between NeuMF and the basic matrix factorization approach. Like normal matrix factorization, the gene features E_g and B_g are optimized directly to minimize the desired loss. However, the cell features are optimized as a non-linear function of x_g. This allows the network to enforce a multiplicative relationship between the cell and gene latent features while also forcing the input counts to contribute to the output prediction directly instead of merely in the back-propagation stage. On top of that, this allows a simple cell-level batching scheme where the number of cells per batch can fluctuate; this keeps the number of features per batch small (i.e. around 200 genes per cell in our dataset times the batch size) and allows us to speed up the training process. This new architecture follows

$$E'_c(x_c)E_g(x_g)^T + B'_c(x_c) + B_g(x_g) \;\rightarrow\; \hat{y}_{MF} \sim y$$

where E'_c and B'_c are specified via activated dense layers acting on x_c instead of directly-optimized embedding layers/lookup weights.

3.3 Multinomial Variational Auto-Encoder

Recall that a variational auto-encoder has two components. The first is an encoder component $\hat{f}_{enc}$ which converts from an input space to a learned bottle-necked prediction of parameters p. These parameters are then used to parameterize a random unit X (generally a gaussian) to associate the input instances to a distribution of random variates. A decoder $\hat{f}_{dec}$ is then trained to reconstruct the input from random variate realizations drawn from the parameterized distributions. In other words:

$$\hat{f}_{dec}(z \sim X(\hat{f}_{enc}(x))) \;\rightarrow\; \hat{f}(x) \sim f(x)$$

In contrast to this, Mult-VAE was proposed with count data in mind and seeks to predict the distributions that would generate the observed counts. Considering the problem of UMI counts under a Bayesian lens, we assume the gene observations converge to a per-cell discrete distribution as the number of observations approaches infinity. Thereby, the distribution of N_c gene observations in cell class c can be derived as a realization of the parameterized multinomial variable $\text{Mult}(N_c, \pi_c)$ where π_c is the parameters of cell class c's distribution. Mult-VAE aims to predict these parameters per class, so the output of the decoder is softmaxed to produce a discrete probability distribution. For the sake of our notation, assume that $\text{Mult}(N, < \pi_{c_0}, \ldots, \pi_{c_G} >) = < \text{Mult}(N, \pi_{c_0}), \ldots, \text{Mult}(N, \pi_{c_G}) >$. With this, the formulation of the problem local to a specific cell index x_c and gene index x_g becomes:

$$\text{Mult}(N_c, \hat{f}_{dec}(z_c \sim \mathcal{N}(\hat{f}_{enc}(x_c)))) \;\rightarrow\; \hat{y}_{MV}(x_c) \sim y(x_c)$$

Unfortunately, this formulation does assume that the predicted UMI counts sum to N_c. If we were to follow the original Mult-VAE paper, N_c would be the sum of UMI counts for the cell; this would train up a relationship that would work well for relational problems and thereby be good for recommender systems that return the top predictions as their deliverable hypothesis. In the case of UMI, the VAE components will still train to predict features that minimize the loss for the non-zero datasets, but it may limit the overall expressiveness as the multinomial is stretched to predict over the entire cell class instance. As such, we consider the incorporation of a bias that offsets the population count parameter of the multinomial. This bias is purely a factor of x_c, and as such we can predict this from $\hat{f}_{enc}$ directly using an additional optimizable structure, i.e. an additional bias embedding B_c of the cell class:

$$N'_c = \sum_{i \in 1..G} x_{ci} + B_c(x_c)$$

Of note, the reliance on the softmax layer does force this to be a problem defined as a function of the class index that has to make a prediction for all gene types. This does not pose a terrible problem for the optimization process as the training loss can be computed over the non-zero results. However, this can be a computational problem as the number of genes increases. It will be useful to consider measures to at least partially express the benefits of the softmax layer while defining $\hat{y}_{MV}$ as a function of x_c and x_g, which we are currently considering. Potential future plans could include incorporating an attention mechanism and defined embeddings similar to the previous techniques.

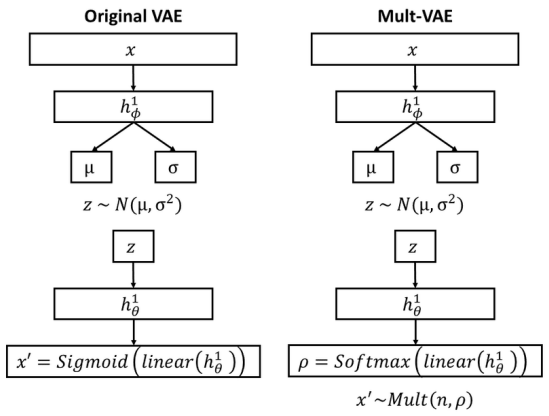

Fig. 2. Multinomial variational autoencoder

3.4 Data

For simulated scRNA-seq data, we used the Splatter R package, set at 2000 cells, 200 genes, and 2 or 6 cell types.

For real biological data, we selected human PBMCs from 10x Genomics (PBMC_G949), with a total of 68K cells and 11K genes at nonzero rate of 41%. We preprocessed the raw count matrix by filtering out cells with fewer than 200 expressed genes and genes expressed in fewer than 3 cells, log-normalizing the values, and selecting the top 1000 highly-variable genes.

picture(0,0)(-35,0)(1,0)30 (0,35)(0,-1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,35)(0,-1)30 picture

3.5 Methods Compared / Baselines

The baseline against which we will be comparing our model will be the Deep Count Autoencoder (DCA) (6), specifically with the default zero-inflated negative binomial (ZINB) loss function. This is the current best-performing matrix factorization approach to the scRNA-seq imputation problem, and so serves as a good comparison to our neural matrix factorization approach. If we can match or outperform DCA we can say with confidence that collaborative filtering is a promising direction to explore.

3.6 Evaluation

First, we ran our models on a simulated single-cell dataset generated from Splatter package, which is standard in papers such as DCA (6). Count data with dropout enabled was considered the input, while count data with dropout disabled was considered the ground truth.

Next, we applied our models to real biological data. Each of the input cells has a random 35% of its nonzero genes chosen as 'simulated' missing genes and their values set to zero, same as in DCA (6). Therefore the imputed values for the simulated missing genes can be considered as the predicted list Y, and the true values for those simulated missing genes considered as the ground truth T.

We evaluate our models by the same metrics as in DCA (6) - examination of recovered clusters by PCA/tSNE and Silhouette value. Because of the way Mult-VAE draws from a multinomial probability distribution, a direct comparison of imputed values to the true counts is unreasonable. Instead, we choose to examine whether or not we are capable of recovering cell types present in the true count matrix, from the imputed output, via clustering. This reflects the idea that the goal of imputation is to produce values for artifical zeros and restore the expression of a cell to the point where its original identity and relation to other neighboring cells can be recovered.

tSNE, like PCA, is a dimensionality reduction technique to visualize the clustering of cell expression on a two-dimensional plot. The tool converts similarities between data points to joint probabilities and minimizes the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

Silhouette value is a measure from -1 to 1 of how similar an object is to its own cluster (cohesion) compared to other clusters (separation) - a high positive value indicates a cell is well matched to its own cluster and not to any other cluster, while a low negative value indicates poor or ambiguous clustering. Briefly, silhouette value is calculated by

$$s(i) = \begin{cases} \frac{b(i)-a(i)}{max\{a(i),b(i)\}} & |C_I| > 1 \\ 0 & |C_I| = 1 \end{cases}$$

where

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i,j)$$

and

$$b(i) = min_{J \neq I} \frac{1}{C_J} \sum_{j \in C_J} d(i,j)$$

for every data point $i \in C_I$ cluster.

4 Results

4.1 Splatter Data

First, we trained our models simply to reconstruct the original, dropout-free input, in order to ensure that they were capable of learning a low-dimensional representation that preserved the clustering of cell types.

Fig. 3. PCA on reconstructed Splatter, 2 cell types

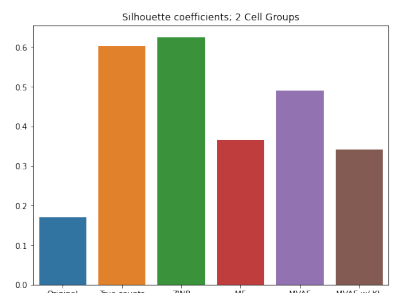

Fig. 4. Silhouette values on reconstructed Splatter, 2 cell types

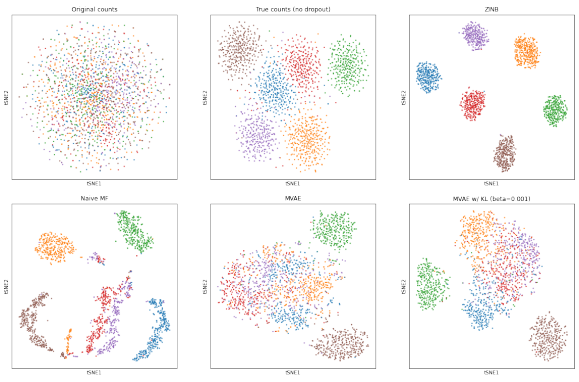

Fig. 5. PCA on reconstructed Splatter, 6 cell types

picture(0,0)(-35,0)(1,0)30 (0,-35)(0,1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,-35)(0,1)30 picture

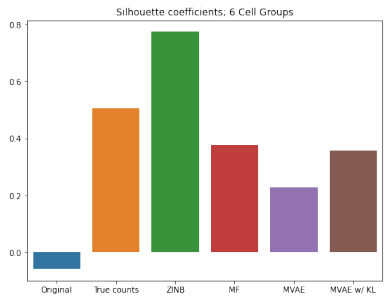

Fig. 6. Silhouette values on reconstructed Splatter, 6 cell types

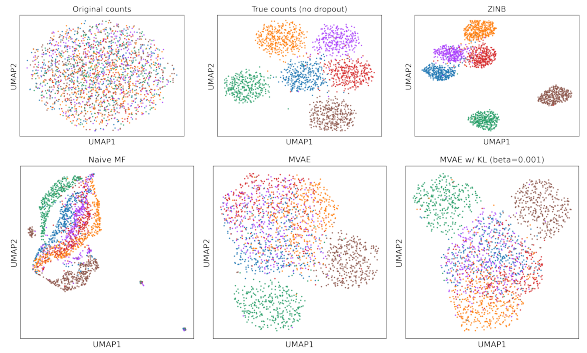

Fig. 9. PCA on imputed Splatter, 6 cell types

While the clustering on MF and Mult-VAE is not quite as tight as on DCA, they are nonetheless capable of capturing most if not all of the 2 and 6 cell types. This provides confidence that our architectures are operating as intended.

Next, we ran the true imputation task - that is, training our models to generate complete scRNA-seq matrices with imputed zero values from dropped-out data.

Fig. 7. PCA on imputed Splatter, 2 cell types

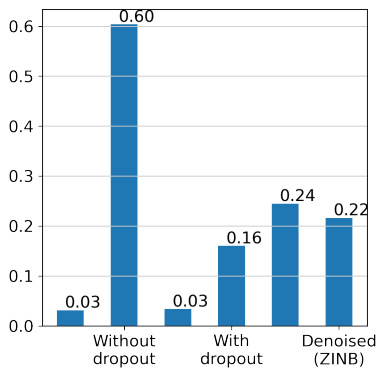

Fig. 8. Silhouette values and Pearson correlation on imputed Splatter, 2 cell types. The Silhouette bars are dropout data, original data, ZINB, MF, MVAE, and MVAE w/ KL, respectively.

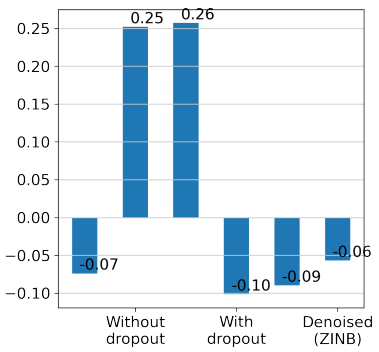

Fig. 10. Silhouette values on imputed Splatter, 6 cell types. The Silhouette bars are dropout data, original data, ZINB, MF, MVAE, and MVAE w/ KL, respectively.

Unusually, DCA is inconsistent with its performance on simulated Splatter datasets with two cell types - in the Figure 7 example it has poor separation, and the low Silhouette values reflect this. In these cases, both MF and MVAE outperform DCA, with MF having better separation and MVAE having better cohesion. Their silhouette values - 0.16, 0.24, and 0.22, respectively, are an order of magnitude larger than that of DCA.

On simulated Splatter datasets with six cell types, however, DCA returns to form, with all six cell types being recovered, and a Silhouette value superior even to that of the true counts. MVAE recovers at least two distinct cell types, but some of the others are poorly separated. MF looks qualitatively better, but the unusual elongation of the clusters and their close proximity to one other nonetheless contributes to a subpar Silhouette score.

What might be the reason for DCA's inconsistency: the DCA method itself, or the Splatter toolkit? The fact that DCA outperforms the true counts in terms of Silhouette value suggests Splatter's low number of cell types or its dropout methods may be at fault. Running the models with real biological data is a far more realistic scenario.

Finally we ran the true imputation task on the aforementioned PBMC_G949 dataset, which is much larger and more heterogenous than Splatter data. Note that because the PBMC_G949 cells are not manually labeled with cell types, and the DCA pipeline precludes clustering on the true counts, the coloration of the PCA in Figure 11 is derived from *post facto* Leiden clustering on a neighborhood graph with a standard 10 neighbors and 40 PCs, and reflect only the clarity and separation of PCA clusters.

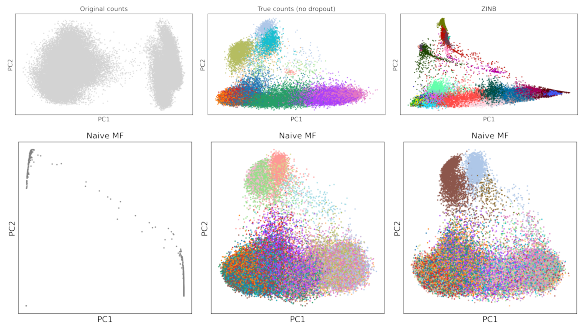

Fig. 11. PCA on imputed PBMC_G949

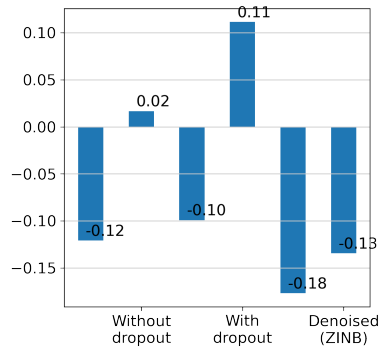

Fig. 12. Silhouette values on imputed PBMC_G949. The Silhouette bars are dropout data, original data, ZINB, MF, MVAE, and MVAE w/ KL, respectively.

On a true biological dataset we can see that both DCA and our model struggle to produce clusters with excellent separation, but this is to be expected with the number and variance in expression of real cells. Notably, the MVAE model's imputed output has a PCA plot extremely similar to that of DCA and the true counts, if only with a wider spread. While the Silhouette values in this case are a great deal less helpful, it is nonetheless good that MVAE is not so far away from that of DCA in terms of cohesion. (Simple MF fails to recover any of the cell types properly, so we may discount it.)

5 Discussion

We present a new approach to imputing zeroes in single-cell RNA-sequencing data using neural collaborative filtering.

The significance of this work lies primarily in the novel application of architectures which have not seen use in the realm of genomics, but which nonetheless have properties that make them extremely well-suited imputation in scRNA-seq; NCFlix represents progress on the long-intractable and inherent problem of dropout and sparsity in an extremely important and widespread sequencing approach.

While at this point in time NCFlix does not exceed the performance of DCA or the standard of the true counts, the model, particularly the Mult-VAE iteration, comes remarkably close, especially when run upon real biological data, with the PCA plot of imputed output having similar distributions and cohesion, and marginally worse spread. The fact that we can introduce technical zeros to real scRNA-seq data and generate an output resembling the original, and resembling the state-of-the-art, is encouraging.

Future experiments include hyperparameter tuning via grid search, a promising avenue considering the use of different beta values for KL clearly is capable of improving model performance; and the testing of more complex deep learning architectures such as RNNs and GANs, which meta-papers in the field of recommender systems have identified less consistent but nonetheless interesting.

6 Contributions

Vadim came up with the idea of using collaborative filtering, and he and Bryce did much of the heavy lifting in terms of getting the DCA and Splatter pipelines actually running. Alexander preprocessed the biological dataset, and got the pipeline running on that. Peter contributed to the writing of the paper. And of course, Professor Singh was responsible for their proper education and an excellent semester.

References

[1]Haque, A., Engel, J., Teichmann, S. & Lönnberg, T. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. (Springer Science,2017,8), https://doi.org/10.1186/s13073-017-0467-4

[2]Hou, W., Ji, Z., Ji, H. & Hicks, S. A systxematic evaluation of single-cell RNA-sequencing imputation methods. *Genome Biology*. (2020,8), https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02132-x

[3]Huang, M., Wang, J., Torre, E., Dueck, H., Shaffer, S., Bonasio, R., Murray, J., Raj, A., Li, M. & Zhang, N. SAVER: gene expression recovery for single-cell RNA sequencing. (Springer Science,2018,6), https://doi.org/10.1038/s41592-018-0033-z

[4]Wang, J., Agarwal, D., Huang, M., Hu, G., Zhou, Z., Ye, C. & Zhang, N. Data denoising with transfer learning in single-cell transcriptomics. *Nature Methods*. **16**, 875-878 (2019,9), https://doi.org/10.1038/s41592-019-0537-1

[5]Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A., Burdziak, C., Moon, K., Chaffer, C., Pattabiraman, D., Bierie, B., Mazutis, L., Wolf, G., Krishnaswamy, S. & Pe'er, D. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. (Elsevier BV,2018,7), https://doi.org/10.1016/j.cell.2018.05.061

[6]Eraslan, G., Simon, L.M., Mircea, M. et al. Single-cell RNA-seq denoising using a deep count autoencoder. Nat Commun 10, 390 (2019). https://doi.org/10.1038/s41467-018-07931-2

[7]Durif, G., Modolo, L., Mold, J., Lambert-Lacroix, S. & Picard, F. Probabilistic count matrix factorization for single cell expression data analysis. (Oxford University Press (OUP),2019,3), https://doi.org/10.1093/bioinformatics/btz177

[8]Lähnemann, D., Köster, J., Szczurek, E., McCarthy, D., Hicks, S., Robinson, M., Vallejos, C., Campbell, K., Beerenwinkel, N., Mahfouz, A., Pinello, L., Skums, P., Stamatakis, A., Attolini, C., Aparicio, S., Baaijens, J., Balvert, M., Barbanson, B., Cappuccio, A., Corleone, G., Dutilh, B., Florescu, M., Guryev, V., Holmer, R., Jahn, K., Lobo, T., Keizer, E., Khatri, I., Kielbasa, S., Korbel, J., Kozlov, A., Kuo, T., Lelieveldt, B., Mandoiu, I., Marioni, J., Marschall, T., Mölder, F., Niknejad, A., Raczkowski, L., Reinders, M., Ridder, J., Saliba, A., Somarakis, A., Stegle, O., Theis, F., Yang, H., Zelikovsky, A., McHardy, A., Raphael, B., Shah, S. & Schönhuth, A. Eleven grand challenges in single-cell data science. (Springer Science,2020,2), https://doi.org/10.1186/s13059-020-1926-6

[9]Elyanow, R., Dumitrascu, B., Engelhardt, B. & Raphael, B. netNMF-sc: Leveraging gene-gene interactions for imputation and dimensionality reduction in single-cell expression analysis. (Cold Spring Harbor Laboratory,2019,2), https://doi.org/10.1101/544346

[10]He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T. Neural Collaborative Filtering. *Proceedings Of The 26th International Conference On World Wide Web*.

picture(0,0)(-35,0)(1,0)30 (0,35)(0,-1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,35)(0,-1)30 picture

pp. 173-182 (2017), https://doi.org/10.1145/3038912.3052569

[11]Guo, Y. & Yan, Z. Recommended System: Attentive Neural Collaborative Filtering. *IEEE Access*. **8** pp. 125953-125960 (2020)

[12]Chae, D., Shin, J. & Kim, S. Collaborative Adversarial Autoencoders: An Effective Collaborative Filtering Model Under the GAN Framework. *IEEE Access*. **7** pp. 37650-37663 (2019)

[13]Liang, D., Krishnan, R., Hoffman, M. & Jebara, T. Variational Autoencoders for Collaborative Filtering. (2018)

[14]Basharat, Z., Majeed, S., Saleem, H., Khan, I. & Yasmin, A. An overview of algorithms and associated applications for single cell RNA-Seq data imputation. (Bentham Science Publishers Ltd.,2020,7), https://doi.org/10.2174/1389202921999200716104916

[15]Rendle, S., Krichene, W., Zhang, L. & Anderson, J. Neural Collaborative Filtering vs. Matrix Factorization Revisited. *Fourteenth ACM Conference On Recommender Systems*. pp. 240-248 (2020), https://doi.org/10.1145/3383313.3412488

[16]Dacrema, M., Boglio, S., Cremonesi, P. & Jannach, D. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Trans. Inf. Syst.*. **39** (2021,1), https://doi.org/10.1145/3434185

picture(0,0)(-35,0)(1,0)30 (0,-35)(0,1)30 picture

picture(0,0)(35,0)(-1,0)30 (0,-35)(0,1)30 picture