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

Bryce Blinn, Vadim Kudlay, Alexander Young, Peter Zhu

CSCI 2952G, Fall 2021

## 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 [Haq+17], 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 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 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 [Hua+18] and SAVER-X [Wan+19], which parameterize a negative binomial to fit the 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 [Dij+18], 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 [Dur+19], 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) [Läh+20].

We take a moment to examine a prominent non-negative matrix factorization (NMF) technique, netNMF-sc, since MF can find excellent use in combination with collaborative filtering. It utilizes an existing gene co-expression network to interpolate the results by learning designated gene and cell-type factor low-rank matrices via vanilla NMF optimization [Ely+19].

### 2.2 Existing Architectures for Neural Collaborative Filtering

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 by, as stated previously, treating our cells as 'users' and their genes to be 'items' or 'issues.'

- NMF architectures [He+17] treat the embeddings of the users and the items as learned functions based on previous interactions.
- GNN-based architectures [GY20] 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 [CSK19] 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 [Lia+18] 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" [Hou+20] 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 [Lia+18], 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 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 is then

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

where $B_c$ and $B_g$ are optional $\mathbb{N}^n \rightarrow \mathbb{R}^n$ 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:

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

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 $\alpha$ is a hyperparameter dictating the weight of the MF relative to the NCF approach:

$$\begin{bmatrix} y_{_{MF}}(x_c, x_g) \\ y_{_{NCF}}(x_c, x_g) \end{bmatrix}^T \begin{bmatrix} \alpha \\ 1-\alpha \end{bmatrix} \rightarrow Dense \ \rightarrow \ y_{_{NeuMF}}(x_c, x_g) \sim y(x_c, x_g)$$

## 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 bottlenecked 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 a 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 $\pi_c$ 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 $\text{Mult}(N, < \pi_{c_0}, \dots, \pi_{c_G} >) = < \text{Mult}(N, \pi_{c_0}), \dots, \text{Mult}(N, \pi_{c_G}) >$. 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. Current plans include incorporating an attention mechanism and and define embeddings similar to the previous techniques.

## 3.4 Data

The following biological datasets were selected for potential training and testing:

1. Mouse cell atlas dataset (GEO accession number GSE108097): 405,796 cells from 51 tissues were sampled and sequenced by Microwell-seq; top 1K, 2K, 5K, and 10K highly-variable genes selected.

2. Embryonic mouse brain from 10X Genomics: 1,308,421 cells sequenced by Cell Ranger 1.3.0 protocol; top 1K highly-variable genes selected.
3. The MAGIC_mouse bone marrow: 2,576 cells & 16,114 genes at multiple differentiation stages of hematopoiesis. N.B. Original data is highly sparse.
4. Human PBMCs from 10x Genomics (PBMC_G949). 68K cells, with 1K genes selected; nonzero rate of 41%.

### 3.5 Methods Compared / Baselines

Due to the success of netNMF-sc as a MF method for scRNA-seq imputation and MF's aforementioned relevance as a method used in tandem with NCF, as well as the industry-standard use of LATE [Bad+20] and autoencoder methods like it, these are the two baselines with which we are going to compare NCFlix.

### 3.6 Evaluation

For real biological data, we randomly split the collection of n cells into $n_{train}$ training cells and $n_{val}$ validation cells. The training cells are used to train the desired models. Each of the validation cells, on the other hand, has a random $p\%$ of its nonzero genes chosen as 'simulated' missing genes and their values set to zero. Therefore the imputed values for the simulated missing genes from the $n_{val}$ cells can be considered as the predicted list $Y$, and the true values for those simulated missing genes considered as the ground truth $T$.

For extremely sparse data like (3), we may enhance the dynamic changes in gene expression and the correlation structure among genes by applying an existing baseline imputation method like MAGIC [Dij+18], providing a ground truth that contains sharply-defined nonlinear patterns.

In any case, imputation accuracy can be defined as the normalized distance between the imputed log count entries and log count ground truth entries. Normalized error can be defined in terms of $y$ and $\hat{y}$ as

$$\mathcal{L}(y, \hat{y}) = \frac{||y - \hat{y}||}{||y||_0}$$

.

## 4 Expected Results

We expect at least one of our models, most likely the NMF or VAE variant, since they are more complex and capable of capturing the necessary dependencies, to outperform netNMF-sc and LATE by the evaluation metric of normalized reconstruction error. In particular we would also like to look at how the weights in our NMF model compare to those from the netNMF-sc model.

It will be interesting to see if the phenomenon of increasingly complex architectures failing to match up to simpler models, as described in [Dac+21], can be seen with an input like scRNA-seq matrices.

## 5 Timeline

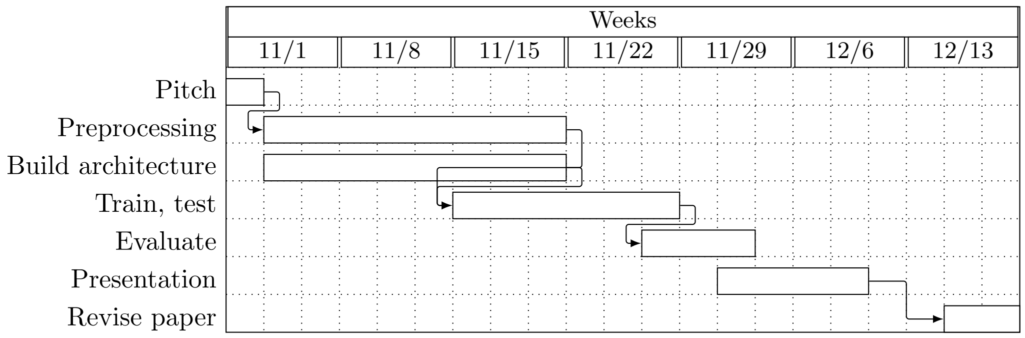

Figure 1: Planned project tasks and timeline

We plan on dividing the workload of the project relatively equally to ensure an equal learning experience, though members may leverage their comparative advantage to some extent i.e. Bryce and Vadim focusing more on the nuts and bolts of the DL architecture, or Alex paying more attention to processing and representation of biological data.

## 6 Discussion

We present a new approach to imputing zeroes in single-cell RNA-sequencing data using neural collaborative filtering. In other words, we expect our models to learn relationships between different cells and the genes likely to be expressed within each, allowing us to predict expected gene expression from erroneous zero values in our scRNA-seq data with superior performance to existing MF and autoencoder models.

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; if we are successful in developing a model with superior accuracy and minimal error in comparison with baselines, this would represent a breakthrough for the long-intractable and inherent problem of dropout and sparsity in an extremely important and widespread sequencing approach.

We expect to face difficulty in determining which architecture best captures the properties of RNA-seq data, as it is useful to picture RNA-seq data as a graph of connections between cells and expressed genes, but such a graph would inherently be incomplete.

We also expect some trouble in the evaluation of the performance of our data, both because it is difficult to determine good ground truths as well as parse the difference between imputation of technical zeroes - a desirable outcome - and imputation of true zeroes - an undesirable outcome.

Future experiments which may lie outside the scope of this project may include the implementation of more complex deep learning architectures such as RNNs and GANs, simply because meta-papers in the field of recommender systems have identified that their associated papers are often not replicable, or otherwise the architectures are outperformed by more simple, non-DL methods.

## References

[Haq+17] Ashraful Haque et al. "A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications". In: 9.1 (Aug. 2017). DOI: 10.1186/s13073-017-0467-4. URL: https://doi.org/10.1186/s13073-017-0467-4.

[He+17] Xiangnan He et al. "Neural Collaborative Filtering". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. ISBN: 9781450349130. DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569.

[Dij+18] David van Dijk et al. "Recovering Gene Interactions from Single-Cell Data Using Data Diffusion". In: 174.3 (July 2018), 716–729.e27. DOI: 10.1016/j.cell.2018.05.061. URL: https://doi.org/10.1016/j.cell.2018.05.061.

[Hua+18] Mo Huang et al. "SAVER: gene expression recovery for single-cell RNA sequencing". In: 15.7 (June 2018), pp. 539–542. DOI: 10.1038/s41592-018-0033-z. URL: https://doi.org/10.1038/s41592-018-0033-z.

[Lia+18] Dawen Liang et al. *Variational Autoencoders for Collaborative Filtering*. 2018. arXiv: 1802.05814 [stat.ML].

[CSK19] Dong-Kyu Chae, Jung Ah Shin, and Sang-Wook Kim. "Collaborative Adversarial Autoencoders: An Effective Collaborative Filtering Model Under the GAN Framework". In: *IEEE Access* 7 (2019), pp. 37650–37663. DOI: 10.1109/ACCESS.2019.2905876.

[Dur+19] Ghislain Durif et al. "Probabilistic count matrix factorization for single cell expression data analysis". In: 35.20 (Mar. 2019). Ed. by Inanc Birol, pp. 4011–4019. DOI: 10.1093/bioinformatics/btz177. URL: https://doi.org/10.1093/bioinformatics/btz177.

[Ely+19] Rebecca Elyanow et al. "netNMF-sc: Leveraging gene-gene interactions for imputation and dimensionality reduction in single-cell expression analysis". In: (Feb. 2019). DOI: `10.1101/544346`. URL: `https://doi.org/10.1101/544346`.

[Wan+19] Jingshu Wang et al. "Data denoising with transfer learning in single-cell transcriptomics". In: *Nature Methods* 16.9 (Sept. 2019), pp. 875–878. ISSN: 1548-7105. DOI: `10.1038/s41592-019-0537-1`. URL: `https://doi.org/10.1038/s41592-019-0537-1`.

[Bad+20] Md. Bahadur Badsha et al. "Imputation of single-cell gene expression with an autoencoder neural network". In: 8.1 (Jan. 2020), pp. 78–94. DOI: `10.1007/s40484-019-0192-7`. URL: `https://doi.org/10.1007/s40484-019-0192-7`.

[GY20] Yanli Guo and Zhongmin Yan. "Recommended System: Attentive Neural Collaborative Filtering". In: *IEEE Access* 8 (2020), pp. 125953–125960. DOI: `10.1109/ACCESS.2020.3006141`.

[Hou+20] Wenpin Hou et al. *A systematic evaluation of single-cell RNA-sequencing imputation methods.* Aug. 2020. URL: `https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02132-x`.

[Läh+20] David Lähnemann et al. "Eleven grand challenges in single-cell data science". In: 21.1 (Feb. 2020). DOI: `10.1186/s13059-020-1926-6`. URL: `https://doi.org/10.1186/s13059-020-1926-6`.

[Dac+21] Maurizio Ferrari Dacrema et al. "A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research". In: *ACM Trans. Inf. Syst.* 39.2 (Jan. 2021). ISSN: 1046-8188. DOI: `10.1145/3434185`. URL: `https://doi.org/10.1145/3434185`.