CMSC701, 2025, 1–4
doi:
UMD Computer Science Graduate Department: 13 May 2025
Scholarly Paper

Evaluating Implicit Neural Representations for Single Cell Sequencing Compression

Evan Guenterberg

Department of Computer Science, University of Maryland, College Park

Abstract

Single-cell RNA sequencing produces massive amounts of data. In order to make research using this technique more efficient, it is worthwhile to develop a compression strategy for the large, sparse matrices that it yields. This work evaluates the application of implicit neural representations (INRs) to this task, demonstrating that using sinusoidal activation functions with fully connected layers, as has been shown to be an effective compression strategy for images, is not a viable path forward in this area.

Key words: Implicit neural representation, single cell sequencing, sparse matrix compression

Introduction

Single cell sequencing

Single-cell RNA sequencing (scRNA-seq) is a growing technology that allows analysis of the RNA transcripts present in individual cells. By sequencing the RNA transcripts themselves, researchers gain knowledge of which genes are being expressed in the cell and at what level. See Jovic et al. [2022] for a survey of single-cell sequencing. This technology has already begun providing resources to treat diseases and unlock new areas of research, but it comes with its own set of challenges. One such challenge is the vast amount of data that scRNA-seq produces. The ultimate output of scRNA-seq is a large matrix representing the cells as columns and the gene expressions as rows. Typically, cells from the same organ are sequenced in a batch, meaning that there is likely to be similarity along the rows. However, these matrices tend to be extremely sparse. The typical storage method for single-cell sequencing data is as an hdf5 file, representing the matrix in the compressed sparse row (CSR) or compressed sparse column (CSC) format.

Implicit neural representations

Implicit neural representations (INRs) are a relatively recent technique used to store a signal using a neural network. By viewing the signal as a function from coordinates to values, it is possible to train a neural network to learn this function. Another intuition for this is that it is an extreme case of overfitting. A typical application of neural networks aims to avoid overfitting, where the network output conforms too closely to the training data. If the network overfits on the data to the point where it can reconstruct the data within some bound, then the network is an approximation of the data itself.

Implicit neural representations have been used for various types of data, ranging from audio signals, geometry, 3D scenes, partial differential equations, images, videos, and more. See Jiang et al. [2020], Park et al. [2019], Mildenhall et al. [2020], Dupont et al. [2022], Kim et al. [2023]. INR research has taken various forms. Early research was focused on learning a network that overfits to a set of data, such that a single network would represent a wide variety of images, scenes, or the like [Chen et al., 2020, see]. Another approach focuses on learning encoders and decoders that map the samples to a latent space and are able to reconstruct the data from the embedding in the latent space. The third approach, overfitting a network to a single sample, shows a promising path for data compression. The success of COIN [Dupont et al., 2021] showed that compressing images (essentially 2D matrices) is possible and competitive with JPEG in certain situations. The downside of COIN is that optimizing a neural network for a single image is not efficient - on the order of hours. The follow-up work, COIN++ [Dupont et al., 2022], improved the technique and showed that it transferred well to other modalities. This method is more complex, and involves meta-learning over a space of data. COOL-CHIC [Ladune et al., 2023] and its successors [Leguay et al., 2023, Kim et al., 2023] took this idea and ran with it, developing a library to encode and decode images and videos using INRs[1].

This work aims to apply simple INR techniques, along the lines of COIN, to single cell sequencing samples in order to evaluate the feasibility of using INRs for compression in this field. COIN-style INRs, though they have their inefficiencies, are a starting point for exploration in this area.

Approach

COIN-style INRs represent one sample of data as a function that takes the row and column indices as input and returns the value

[1] Available at `https://github.com/Orange-OpenSource/Cool-Chic`

© The Author 2025. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

at that point: $f : \mathbb{R}^2 \to \mathbb{R}^C$, where C is the number of channels in the signal. Training the network uses all possible coordinates as input, and the corresponding values at those coordinates as 'labels'. Rather than use integer rows and columns, COIN scales the indexes so that the rows are in the range $[-1, 1]$ and the columns are scaled by the same value, also centered around 0. This practice dates back to the SIREN paper [Sitzmann et al., 2020], which introduced sinusoidal activation functions to capture higher-resolution information in INRs, which COIN uses (and so does this work). This decision is not explicitly justified by either paper, but it likely has to do with the sinusoidal activation function. I also follow the scaling of indexes in this work.

Due to the sparsity of single cell sequencing data, I believed that it would not be effective to train an INR on the entirety of the data. Rather, the neural network can be fit to the nonzero elements of the matrix. This greatly reduces the volume of information the network needs to learn and means that the data has fewer boundaries with large transitions, which neural networks tend to struggle to approximate.

Although COIN is designed for relatively small images (they provide evaluation results on the Kodak dataset of 768 by 512 pixel images), C3 [Kim et al., 2023] shows that breaking an image into patches and creating an INR for each patch is a successful technique. Based on this, this work will focus on developing an INR for a subset of a single-cell sequencing matrix.

An auxiliary succinct data structure can then hold information about which entries in the matrix were zeros. Value lookup would be a two step process: examining the auxiliary data structure and retrieving the nonzero value from the INR if necessary. Notably, it is still a constant time operation, as the INR is essentially a series of fixed matrix operations.

Methods

A single dataset of single cell sequencing data was retrieved from 10x Genomics [10x Genomics, 2021]. The cells sampled were human peripheral blood mononuclear cells from a single healthy donor. Only a single dataset was used because this was intended to be an initial foray into the technique, to be supplemented with other datasets later, but for reasons that will become clear, this did not come to pass. This data consisted of a 587 by 36601 matrix, with corresponding information about the cells sequenced and the genes tallied. To simulate patching, 500 columns and 1000 rows of the matrix were randomly sampled. In practice, patches would be contiguous, but in this investigative case it was randomly sampled to preserve as many attributes (sparsity, value distribution, etc) from the whole matrix as possible. This patch contained 33564 nonzero entries.

The network architecture was chosen based on the work by Dupont et al. [2021] involving an architecture search which settled on a network of 10 hidden layers, each with the same number of nodes. Using more than 10 layers is not advised due to the vanishing gradient problem, where the early layers of the network do not contribute to the approximation. Each hidden layer consists of a fully connected layer and a sinusoidal activation function. The output layer consists of a fully connected layer with no activation function. In this case, since the values produced by single-cell sequencing are single integers, the output has one node. The input layer is two nodes, with no activation layer, representing the coordinates of each entry of the matrix.

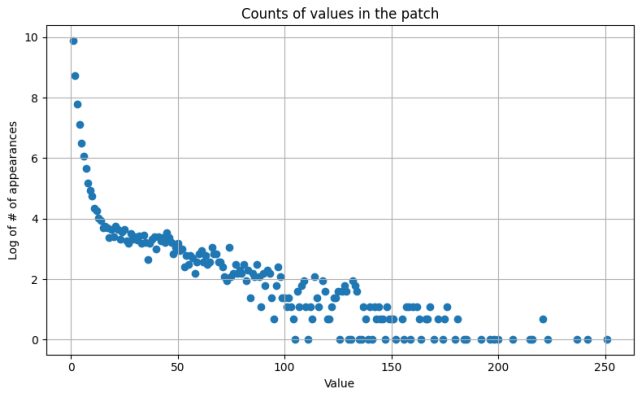


Fig. 1: Log of the frequency distribution of the values in the 500x1000 patch of single-celled sequencing data.

Various smaller numbers of nodes in the hidden layer were attempted, none of which produced usable results. Eventually I decided on 2400 nodes, which provided a trade-off between acceptable training time (less than one hour) and a large number of parameters (51,871,201). Maximizing parameter count for this work was desirable, because if producing an INR with a much greater size than the data proved to be impossible, then it can reasonably be concluded that INRs are a non-viable method of compression for this modality.

The loss function used for the INR is mean squared error (MSE), also known as the squared L2 norm. This is a widely used loss function, but it offers the desirable property that predicted values which differ from the expected values by greater than one are given more pressure to conform to the expected result during backpropagation. Since the values are rounded to the nearest integer after evaluation anyway, it is less important to correct values that round to the expected result.

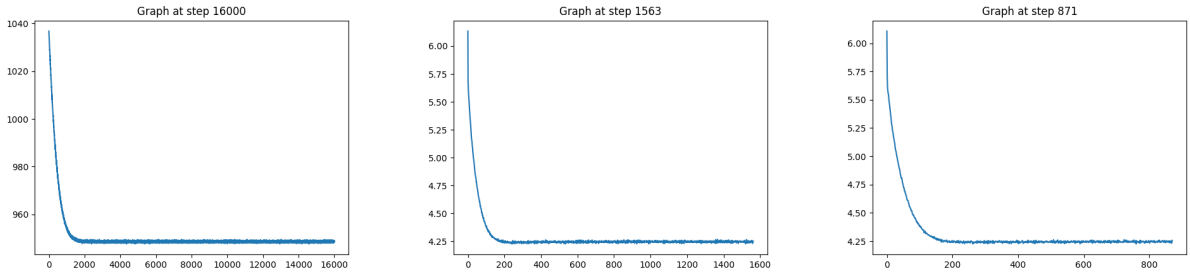
Three configurations were tested during training: training to match the matrix entries exactly without any normalization function, training to match the matrix entries divided by the maximum matrix entry (min-max scaling), and training to match the log of the matrix entries (logarithmic scaling). The two attempts at normalization follow the observations that the matrices COIN trains on are composed of values falling within the $[0, 1]$ range, and that the entries of the single-cell sequencing matrices follow a roughly exponential decay (see Figure 1).

Results

All configurations successfully converged (see Figure 2), indicating successful training and that the models had achieved a local maxima approximating the matrix.

The INRs were evaluated by accuracy of their predicted values after rounding. Unfortunately, no INR even approached a reasonable accuracy. See Table 1 for the values. Logarithmic normalization was substantially more effective than no normalization or min-max normalization, but it still managed to predict the right matrix value less than a quarter of the time.

I believe this to result from a combination of two factors. Primarily, single-cell sequencing data contains very few regions of homogeneity. Even if adjacent cells in the matrix have similar counts of gene expressions, that only extends horizontally and



(a) No normalization (b) Min-max normalization (c) Log normalization

Fig. 2: Convergence plots showing MSE per epoch of the three configurations. Note the smooth curve to a stable loss value in all three, indicating successful convergence of the neural network. The networks were not trained for the same number of epochs, but this is not a problem as the networks had converged to a stable state.

Accuracy results (%) on nonzero entries		
No normalization	Min-max normalization	Log normalization
3.32	4.83	23.28

Table 1. INR recreation accuracy results.

not vertically. Nearly all other successful use cases of INRs involve modalities that contain regions of homogeneity - images, geometries, 3D scenes all are focused on regions of similar values. Although improvements have been made in approximating high-resolution boundaries [see Sitzmann et al., 2020], these are still boundaries of relatively homogeneous regions. Secondarily, the unbounded nature of single-cell sequencing data means that the sharp boundaries present can be extremely steep. This contrasts with images, where the values are limited to the range $[0, 255]$.

Limitations

This project was limited to a single patch of 500 by 1000 entries from a single dataset. It is therefore conceivable that COIN-style INRs might accurately compress some scRNA-seq data, but not this sample. However this indicates that at the very least, COIN-style INRs cannot be a universal method of storing, let alone compressing, scRNA-seq data.

Future Work

Matrix reordering techniques were not explored in this work. Permuting either the rows or columns has no impact on the underlying data, as long as the labels are identically permuted. It may be successful to reorder the rows and columns of scRNA-seq data as a preparation step for INR fitting. Increasing homogeneity by grouping similar values together might improve performance following the principles discussed above, but it would be very difficult to guarantee that a reordering that ensures successful INR creation exists.

Advanced patching techniques are another potential avenue of improvement. Given the likelihood of similar gene expression values along the rows, using patches that extend the entire width of the matrix might increase homogeneity and therefore performance. This similarity is not guaranteed, especially given the fact that scRNA-seq samples can be combined by horizontal concatenation, so this approach is probably limited.

Conclusion

Due to the unbounded nature of the values stored and the lack of homogeneity, the sparse matrices generated by single-cell sequencing data are not amenable to compression via COIN-style INRs. The naive approach, considered foundational for the success of later works, was unable to reconstruct the underlying data even at a highly negative compression ratio. It is possible, but unlikely, that improvements may be found in matrix reordering techniques or other INR architectures such as the encoder-decoder model.

References

10x Genomics. 500 human pbmcs, 3' lt v3.1, chromium x, single cell immune profiling dataset by cell ranger v6.1.0, 2021. URL `https://www.10xgenomics.com/datasets/500-human-pbm-cs-3-lt-v-3-1-chromium-x-3-1-low-6-1-0`.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. *arXiv preprint arXiv:2012.09161*, 2020.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations, 2021. URL `https://arxiv.org/abs/2103.03123`.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. Coin++: Neural compression across modalities, 2022. URL `https://arxiv.org/abs/2201.12904`.

Chiyu Max Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework, 2020. URL `https://arxiv.org/abs/2005.01463`.

Dragomirka Jovic, Xue Liang, Hua Zeng, Lin Lin, Fengping Xu, and Yonglun Luo. Single-cell rna sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine*, 12(3):e694, 2022. doi: https://doi.org/10.1002/ctm2.694. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/ctm2.694`.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: High-performance and low-complexity neural compression from a single image or video,

2023. URL https://arxiv.org/abs/2312.02753.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. Cool-chic: Coordinate-based low complexity hierarchical image codec, 2023. URL https://arxiv.org/abs/2212.05458.

Thomas Leguay, Théo Ladune, Pierrick Philippe, Gordon Clare, and Félix Henry. Low-complexity overfitted neural image codec, 2023. URL https://arxiv.org/abs/2307.12706.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. URL https://arxiv.org/abs/1901.05103.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. URL https://arxiv.org/abs/2006.09661.