Simulating Latent Single-Cell Gene Expression of COVID-19 Patients with Variational Autoencoder and Implications for Augmenting Classification Models

Romel Baral, Jeremy Fisher, Simon Levine and Krithika Veerappan

02-512/712 Fall 2020

1 Abstract

We present a Variational Autoencoder-based simulation of single cell gene expression from healthy and COVID-19 patient PBMC cells with the aim of addressing class-imbalanced data. To demonstrate the utility of this simulation, we build a synthetic dataset seeded from rare cell types. We then train a classifier of ventilation severity and demonstrate that, by augmenting with simulated data, predictive performance does not decline significantly. Moreover, based on phylogeny tree analysis, the simulated rare cell data does not differ dramatically from the original distribution, as desired. Regardless of these mixed results, we consider our simulation as a solid baseline and a promising future direction to aiding in single cell research of COVID-19.

2 Introduction

The most common COVID-19 test, nasal viral swabbing, cannot adequately determine a proper course of treatment by itself [4]. Aside from physician evaluations, researchers have proposed using machine learning algorithms on other data modalities, such as X-rays, to determine who should be given a ventilator [20]. However, studies that model COVID-19 disease severity routinely suffer from limited, low-quality or imbalanced datasets. These are crucial issues to address as these models become essential to determining courses of treatment and allocating ventilators in the United States [16].

COVID-19 is a vascular disease [11]. Therefore, research is on-going to establish which cell types most affect the peripheral immune response. For example, a recent study found that monocytes and lymphocytes do not increase pro-inflammatory cytokines, which is often associated with more severe COVID-19 cases [17]. The study primarily used dimensionality reduction and graph-based clustering to create a cell atlas for COVID-19 [17]. This study also noted that their sample data size was not very large, and that this was one of the limitations of the study [17].

Indeed, there are pronounced class (that is, cell type) imbalances in most preliminary single cell datasets such as the ones found on `https://www.covid19cellatlas.org` (see Figure 1). In addition to certain immunocytes generally being more populous than other ones, healthy and sick donors are expected to have varying immunocytes distributions due to the disease response itself.

It is common that class imbalance leads to diminished performance in classification tasks. To address this, we introduce a Variational Autoencoder (VAE) simulation to generate synthetic single cell data from healthy and COVID-19 patients. VAEs are widely used in unsupervised learning tasks [2] and have achieved stunning success in a variety of simulation contexts [1, 5, 6]. More recently, VAEs have been applied to genomic contexts as well [9, 12]. Because we want to build a simulation where we can produce synthetic examples from different cell types, it is important that we have a simple latent distribution to draw from. But we must also train this efficiently on large single cell datasets. VAEs fulfill these requirements.

In this paper, we investigate the implementation and uses of such a simulation. To demonstrate its efficacy, we trained a classifier to predict the origin of real and simulated single-cell gene expression: that is, whether it derived from a healthy patient, an admitted COVID-19 patient or a COVID-19 patient that

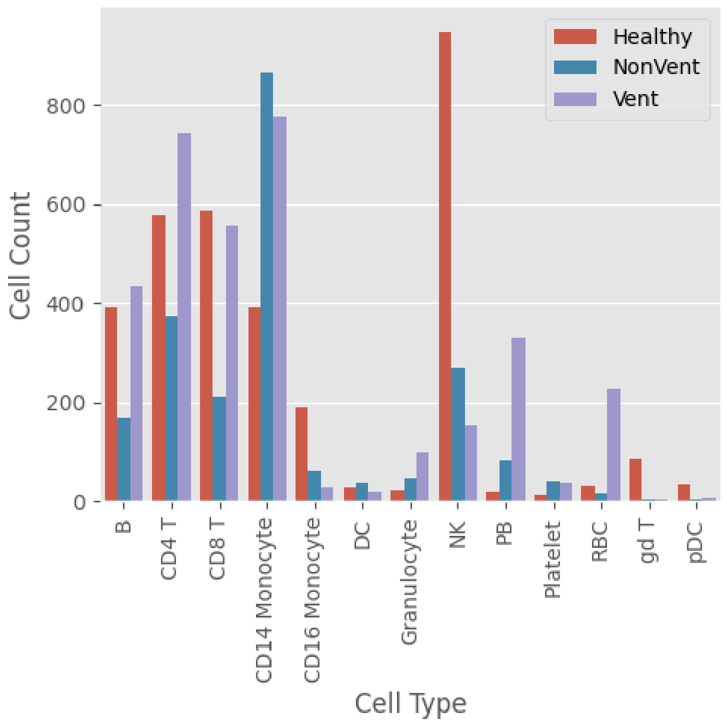

Figure 1: Class Imbalance in the Peripheral Blood Mononuclear Cells Dataset from the Blish Lab

required a ventilator. We also provide a phylogenetic analysis demonstrating that the synthetic data do not differ dramatically from the original data.

3 Data and Methods

3.1 Dataset

We used the Peripheral Blood Mononuclear Cell (PBMC) dataset from the Blish lab because (1) COVID-19 is a vascular disease, (2) it contained both cell type (monocytes, CD4+, CD8+, etc) and ventilator use as per-cell features, and (3) it was drawn from healthy and sick donors [17].

3.2 Pre-processing

Our chosen dataset was already pre-processed and provided by the Blish lab [17]. For thoroughness, we also implemented a pre-processing pipeline for other, unprocessed raw scRNA datasets. This follows closely to standard normalization steps such as used in the ScanPy package [1]. In our case, we started by setting the minimum number of genes per cell to 200 and the minimum number of cells per cell to five. This removes genes that are seen in very few cells and as such would be less relevant. Next, we filtered out genes that had counts less than 2500 and filtered out mitochondrial noise at a five percent threshold.

We filtered out the highly variable genes and regressed out the effects on the data that arose from the cell counts and the amount of mitochondrial genes that were expressed in the cell. Finally, we removed examples greater than ten standard deviations from the mean and scaled the remaining data to unit variance [10, 18].

The pre-processed data is then given directly to the VAE. We skipped the common step of leveraging PCA-based dimensionality reduction because VAEs themselves have a similar functionality: that is, the encoder itself should learn a low-dimensional representation.

[1] `https://scanpy-tutorials.readthedocs.io/en/latest/pbmc3k.html`

3.3 Simulating Single Cell Data

3.3.1 Training the VAE Latent Variable Simulation

We assume gene expression derives from a set of latent factors, denoted $z = \{z_1, \ldots, z_n\}$. This casts our problem as Expectation Maximization:

$$\text{argmax}_z p(z|x) \text{ where } p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

where x is some observation (i.e., normalized gene expression for a single cell) in our dataset. However, $p(x) = \int p(x|z)p(z)dz$ is intractable. Hence, we approximate $p(z|x)$ with a tractable distribution $y \sim q(z|x)$, such as the multivariate normal with a diagonal covariance matrix. Then, we iteratively improve estimates for z by minimizing the expected log

$$\sum_y Pr(y|x, \bar{z}) \text{ log } Pr(x, y|z)$$

Variational Autoencoders (VAEs) are an increasingly common Expectation Maximization algorithm for simulation. It works on a principle of compressing and reconstructing vector inputs. This produces an encoder (or recognition model) ϕ, which maps inputs into a tractable latent space. Furthermore, a decoder (or generative model) ψ maps samples in the latent distribution back into the original vector space. In our case, we minimize expected log by minimizing Kullback–Leibler divergence (assuming the distribution q is parametrized by θ):

$$\begin{aligned} \text{argmin}_\theta \; KL(q_\theta || p(z|x)) &= \mathbb{E}_{q_\theta} \text{ log } q_\theta(z|x) - p(z|x) \\ &= -\mathbb{E}_{q_\theta}\left[\text{ log } \left(\frac{p(x,z)}{q_\theta(z|x)} \right) \right] + \text{ log} p(x) \\ &= -\mathcal{L}_\theta(x) + \text{ log } p(x) \text{ where } \mathcal{L}_\theta(x) = \mathbb{E}_{q_\theta} \text{ log } \left(\frac{p(x,z)}{q_\theta(z|x)} \right) \end{aligned}$$

Because $\mathcal{L}_\theta$ is non-negative, maximizing $\mathcal{L}$ improves $p(x)$ and $q(x)$. Note, also, we can reformulate it as

$$\mathbb{E}_q \left[\text{log } p(x|z) \right] - KL\left(q(z|x) || p(z) \right)$$

This formulation clarifies the relationship to Expectation-Maximization. Furthermore, it is an efficient method for doing so because it trains itself entirely with by gradient descent with this objective. (Albeit not end-to-end, requiring the "reparameterization trick" that allows the network to retain log-variance and mean as parameters [8].) Therefore, it is more appropriate – compared to compared to MCMC, say – for large datasets common in the scRNAseq literature.

3.3.2 Using VAE to augment the original dataset

We use our VAE simulation in the following manner to assess its efficacy:

- Build a classifer, M, with D, where D is the original dataset
- Using the training data, we determine the following:

$$\mu_{minority} = \frac{1}{|D'|} \sum_{i=1}^{|D'|} \phi(x_i) \text{ where } D' = \{x \in D \mid x \text{ is a member of the minority class}\}$$

for all minority cell types

- Build a classifer, M', with D and S
- Build a synthetic dataset $S = \{x \sim \mathcal{N}(\mu_{\text{minority}}, 1)\}$ such that, when merged with D, the minority and majority classes are more balanced. Specifically, if any class has a representation of less than 100 cells in the single cell dataset, then synthetic cells are simulated such that the overall count of cells is 100.
- Compare the performance of M and M' with classification metrics (accuracy, F1 score, etc)

3.4 Choosing an Appropriate Latent Space

It in non-obvious how many axes of gene expression are required to describe in a lower dimensional space the variety of expression of thousands of genes and dozens of cell type. We denote the dimensionality of this latent space as λ and proceed by using maximum-likelihood estimation and Newton-Raphson to determine it empirically.

We pose the objective function as the likelihood of observing hold-out single-cell expression profiles. For a variational autoencoder, the likelihood function is defined

$$C(\lambda) = \prod_{i=0}^{n} \mathcal{N}(\psi_\lambda(z_i) | \mu = 0, \sigma = 1)$$

or, equivalently,

$$y = ln \cdot C(\lambda) = \sum_{i=0}^{n} ln \cdot \mathcal{N}(\psi_\lambda(x_i)) = \frac{-n}{2} ln(2\pi) - \frac{1}{2}\sum_{i=1}^{n} \psi_\lambda(m_i)$$

for some dataset $\{x_1, ..., x_n\}$ where each z is a gene expression profile for a single cell not seen during the training process. To perform this optimization, we minimize the negative log-likelihood with Newton-Raphson by finding a zero of

$$y' = \frac{d(ln \cdot C)}{d\lambda} = \frac{1}{2}\left(\sum_{i=1}^{n} \frac{d\psi_\lambda(z_i)}{d\lambda}\right)$$

Lacking an expression for $\frac{d\psi_\lambda}{d\lambda}$, we use the Taylor approximations

$$y' \approx \frac{1}{2}\left(\sum_{i=1}^{n} \frac{f_{\lambda+\Delta\lambda}(m_i) - f_{\lambda-\Delta\lambda}(m_i)}{2\Delta\lambda}\right)$$

$$y'' \approx \frac{1}{2}\left(\sum_{i=1}^{n} \frac{f_{\lambda+\Delta\lambda}(m_i) + f_{\lambda-\Delta\lambda}(m_i) - 2f_\lambda(m_i)}{\Delta\lambda^2}\right)$$

We also must make a few modifications to the algorithm to fit the dimensions of the problem. λ is an integer, so we let $\Delta\lambda = 1$ and coerce $x - \Delta x \in \mathbb{Z}$. Furthermore, the domain of ψ is positive, non-zero integers fewer than the number of genes. To address that the Newton-Raphson step can suggest a λ not in this domain, we introduce a "temperature" coefficient. Each time the Newton-Raphson algorithm produces an invalid λ, this temperature, t, is halved and the step is repeated. In all, the parameter tuning algorithm is:

Algorithm 1: Modified Newton-Raphson Algorithm

```
Input: $y'$; $y''$; $\lambda_0$; $N$, the number of genes in the dataset; $t$, the temperature coefficient
Output: $\lambda$
$i \leftarrow 1$
$\lambda \leftarrow \lambda_0$
while $i \leq 10$ do
    $\lambda_{\text{candidate}} \leftarrow \left\lfloor \lambda - t\left(\frac{y'(\lambda)}{y''(\lambda)}\right) \right\rfloor$
    if $0 < \lambda_{candidate} < N$ then
        $\lambda \leftarrow \lambda_{\text{candidate}}$
        $i \leftarrow i + 1$
    else
        $t \leftarrow \frac{t}{2}$
```

3.5 Classifier

We tried various kinds of machine learning classifiers to classify the data (both existing data and the generated data). We chose the classifier with the best accuracy as it might help us in comparison more than the others.

For the chosen classifier, we implemented a deep neural network to accomplish the task of classifying single cell read count features against ventilation status classes (healthy, sick without requiring a ventilator and sick requiring a ventilator). Through a multilayer perceptron, we learn a function $\hat{\vec{y}} = f(\vec{x}) = \arg\max_y \Pr(Y = y|X)$, such that the neural network is capable of inferring the most probable ventilation class label(s) $\hat{y}$ given an instance of single cell reads X. As a result, we present a model capable of separating clinical labels from single cell read counts across cell types, useful for benchmarking the utility of generative data from the learned, optimized variational autoencoder.

3.5.1 Architecture

Specifically, a multilayer perceptron was implemented using PyTorch. The network is composed of an initial linear layer of size 26361, one for each distinct single-cell read feature in our data, with 2048 logits as output size.

A rectified linear unit (ReLU; $\max(0, x)$) was used as the activation function, followed by a dropout layer[2] with the intention of regularizing the learned network to decrease bias at the expense of increased variance.

Two fully connected layers follow, each also using ReLU and dropout as activation and regularization components, respectively, again with a hidden size of 2048 neurons. A final linear layer computes class probabilities, for logits[3] $\hat{\vec{y}}$ such that $|\hat{\vec{y}}| = 3$.

3.5.2 Hyperparameters

The network is trained for 10 epochs each for both cases of baseline (single-cell only) data and augmented (single-cell and simulated) data. Note that the model is re-instantiated, such that weights are re-initialized per Kaiming uniform. Dropout probabilities were set to 10%, with dropout disabled during validation and testing phases. A learning rate of 0.001 was used with an Adam optimizer. Mini-batch sizes were set to 256. Note again that during a training/test/validation run using simulated data, we oversample from those classes belonging to cell types with fewer than 100 cells per class.

3.5.3 Training

We train our model with Adam-optimized stochastic gradient descent and back-propagation of predicted-real divergence gradients through the fully connected network.

During training, we compute the logarithm of the soft-max of the outputted network logits [4]. The most probable predicted class for a given feature input x is then yielded from the argument corresponding to the maximum predicted logit in this vector. [5] Next, the mean-reduced negative log-likelihood loss is obtained[6] across the predicted class and ground-truth class label [7].

Validation and test loops are identical to this training loop, except mini-batches are not stochastically ingested, dropout is disabled, gradient computation is disabled, and backpropagation is disabled. Upon termination of each epoch (training included), Accuracy, F1, Recall, and Precision are computed [8].

3.6 Phylogenetic Analysis

Next we wanted to analyze the biological importance of our simulated sample. So, we constructed two phylogenetic trees by the UPGMA[9] algorithm [19]: one on the original data and another on the original

[2]Dropout implies randomly setting elements of the input tensor $\vec{x}$ to zero, or $\vec{x} \to \vec{x}' = [x'_1, ..., x'_N]$ where $x'_i = 0$ iff $z \sim \mathcal{U}(0,1) < p_{\text{dropout}} = 0.10$, else $x'_i = x_i$. Note that during training, outputs are scaled by $\frac{1}{1-p}$.

[3]$y_i \in \{\text{ventilated, non-ventilated, healthy}\}$

[4]$\hat{\vec{y}}' = \text{LogSoftMax}(\hat{\vec{y}}) = \log\left(\frac{exp(\hat{y}_i}{\sum_j \exp(\hat{y}_j)}\right)$

[5]$\hat{y}_{\text{class}} = \arg\max \hat{\vec{y}}$

[6]Note that our implementation is exactly equivalent to taking the cross-entropy loss between most probable logits and ground-truth labels.

[7]$\mathcal{L}(\hat{\vec{y}}, y_{\vec{\text{class}}}) = \sum_{n=1}^{N} \left(\frac{l_n}{\sum_n \vec{w}_{\vec{y_n}}}\right)$, such that $l_n = -\vec{w}_{\vec{y_n}} \vec{x}_{n,y_n}$ Note that we weight classes equally in this training implementation, such that $\vec{w_c} = 1 \; \forall$ classes $c \in \{\text{ventilated, non-ventilated, healthy}\}$.

[8]$Precision = \frac{TP}{\sum TP, FP}$; $Recall = \frac{TP}{\sum TP, FN}$; $F1 = 2\frac{Precision * Recall}{\sum Precision,\ Recall}$; $Accuracy = \frac{\sum TP, TN}{\sum(TP, TN, FP, FN)}$

[9]Unweighted Pair Group Method with Arithmetic mean

data augmented with simulated data. Because of the large size of both original and generated datasets, tree construction would require much more computational resources than we had, rendering it infeasible. So we created various subsampled datasets by randomly sampling a percentage (say, $x\%$) of original dataset and, separately the same percentage from the simulated data concatenated with the original dataset.

Given a subset of data, we calculate the distance metric between the data points. Then we aggregate the data points based on the distance and we keep continuing this process until we have a single group. We compare these two trees on on euclidean distance, false positive bipartition number (not to be confused with rate) and difference in bipartition number [14, 15].

4 Results

Iteration	f	f'	f''	Temperature	λ_n	λ_{n+1}
1	695.3	4.6	25.0	1	32	31
2	703.2	-3.5	-8.6	1	31	30
3	702.4	8.6	-15.7	1	30	30
4	702.4	8.6	-15.7	1	30	30
5	702.4	8.6	-15.7	1	30	30
6	702.4	8.6	-15.7	1	30	30
7	702.4	8.6	-15.7	1	30	30
8	702.4	8.6	-15.7	1	30	30
9	702.4	8.6	-15.7	1	30	30
10	702.4	8.6	-15.7	1	30	30

Table 1: Modified Newton-Raphson

Starting from a guess of $\lambda = 32$ latent dimensions, the Modified Newton-Raphson algorithm converged on 30 latent dimensions, as demonstrated in Table 1.

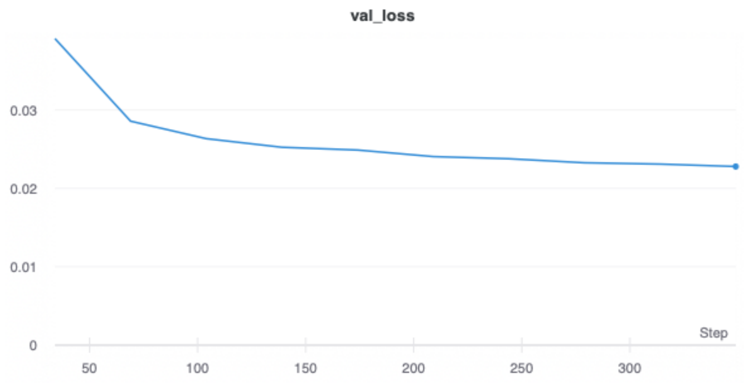

Figure 2: Validation Loss Curve for the VAE

The validation loss decreased steadily in the VAE over the course of 10 epochs, as demonstrated in Figure 2. The validation loss in the classifier failed to converge (Figure 3).

The classifier achieved high accuracy on the Validation dataset split. Augmenting with synthetic data diminished overall performance slightly, but we did not observe significant drop in accuracy in any of the previously underrepresented cell types (Figure 4).

We found that our chosen metrics for the phylogenetic analysis grew linearly with respect to the subsampling rate, as demonstrated in Figure 5. We also note that the slope is relatively shallow for the Euclidean distance curve. We also notice that the False positive number and bipartition number difference values

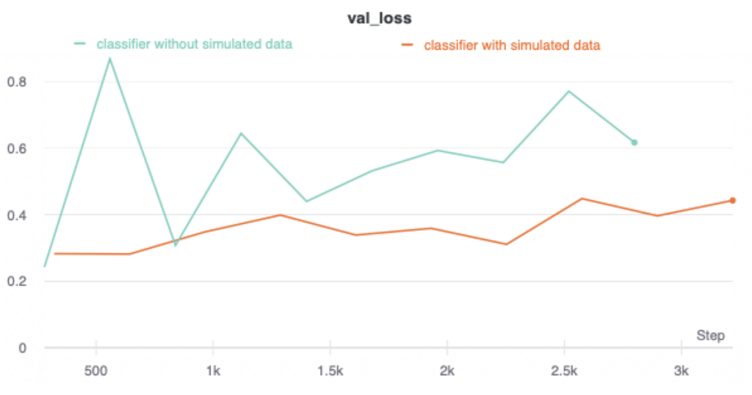

Figure 3: Validation Loss Curve for the Classifier

	Accuracy	F1	Recall	Precision
MLP	0.92	0.92	0.92	0.92
MLP with Synthetic Data	0.90	0.90	0.90	0.90

Table 2: Validation Metrics for Classifier. Note that we collected micro-class average benchmarks, and as such, the metrics are equal within runs.

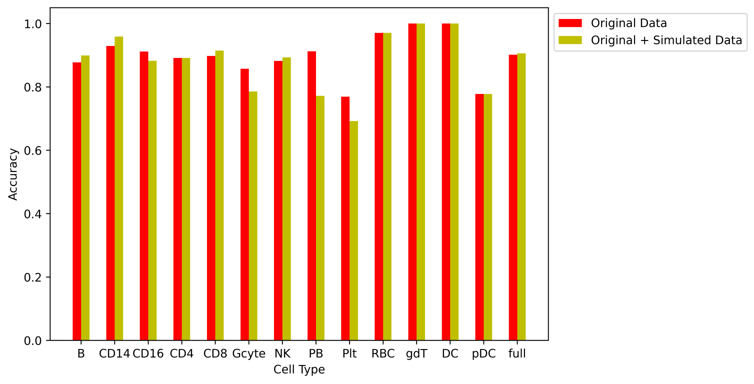

Figure 4: Accuracy of the classifier evaluated across different cell types.

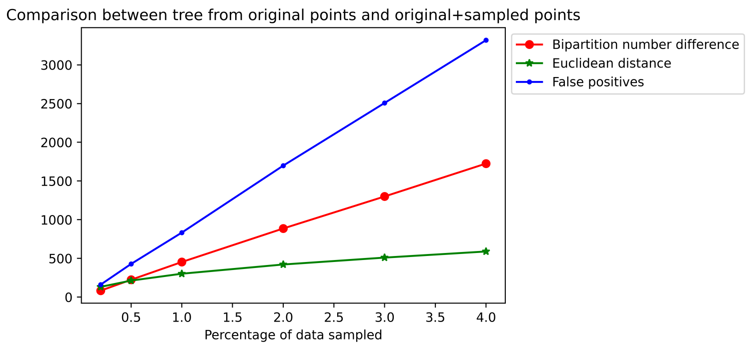

Figure 5: Comparison between trees constructed from original data and original and synthetic data

are bounded by simulated sample count, which is the desired effect as it shows that the difference in phlogenetic trees are bounded above by the added samples. In other words, this verifies the hypothesis that

VAE-generated data belongs to the same distribution as the real data.

5 Discussion

In this study, we introduced a novel simulation for single cell gene expression and assessed its utility in a clinically-important context.

Our implementation of Newton-Raphson converged immediately on a solution that had a higher negative log likelihood than our initial guess. Regardless of our initialization for λ_0 and t, we observed a similar pattern. Ultimately, we settled on the result of $\lambda = 30$. We expect this algorithm has little utility, given the granularity of the problem and the fact that the parameter was constrained to be a non-zero natural number.

Validation loss on our simulattion decreased smoothly, as expected for a deep learning application that converges on a locally optimal solution. This gives us confidence that Variational Autoencoders were a useful and appropriate architecture for simulation.

1,359 cells were generated from the VAE simulation, as per the algorithm described the Methods section. We were surprised by the classification results. First, the classifier achieved an excellent performance in either case, despite that the validation loss curves did not decrease or converge. UMAP projections indicated these classes were likely already highly separable and perhaps achieved locally optimal performance very quickly (see Figure 6). Appending synthetic data tended to diminish performance, albeit only slightly. From this, we conclude that the simulation or the algorithm that uses the simulation needs additional research and modifications.

With regards to the phylogenetic analysis, we did not observe any exponential pattern in our chosen metrics with respect to the subsampling rate, which suggests that our synthetic dataset closely resembled the original dataset.

In the future, we expect that an algorithm adapted to learning integer hyperparameters, such as Population Based Training [7], would more robustly choose an appropriate latent space dimensionality. Regardless, our results on the validation loss curve point to the simulation itself achieving a good balance between a simple latent variable distribution and reconstruction. Therefore, we believe the algorithm used to correct class imbalance is more culpable than the simulation itself in reducing the classifier performance. Further investigation into the appropriate amount of synthetic data to include is therefore warranted. Finally, we feel that adversarial game-theoretic approaches such as Generative Adversarial Networks could serve as an alternative generative model, given sufficient computational resources.

6 Appendix A

We strove to make our simulation reproducible using Data Version Control (DVC). The code and instructions for doing so is available publicly at: `https://github.com/simonlevine/singlecell-sim-by-VAE`

7 Appendix B: Supplementary Figures

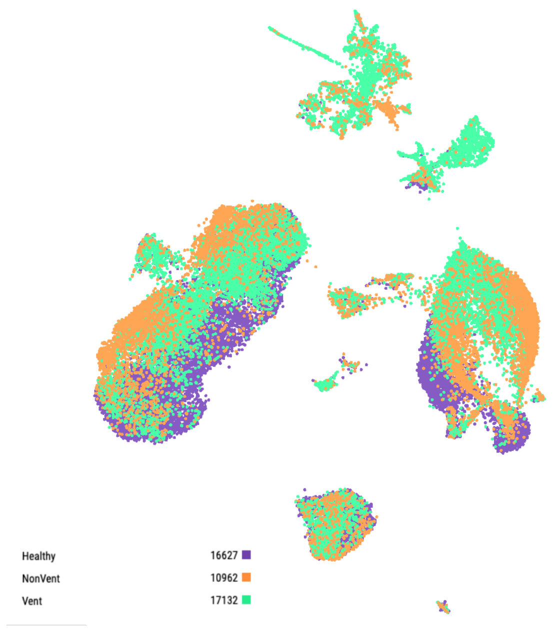

Figure 6: UMAP Visualization of Single-Cell Data, Colored by Ventilation Status. Note plausible linear separability of classes

References

[1] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[2] Carl Doersch. Tutorial on variational autoencoders, 2016.

[3] Mauro Camara Escudero. Variational auto-encoders and the expectation-maximization algorithm, Jul 2020.

[4] Centers for Disease Control and Prevention. Covid-19 (coronavirus disease) test for current infection, 2020.

[5] Chengshuai Li, Shuai Han, and Jianping Xing. Vce: Variational convertor-encoder for one-shot generalization, 2020.

[6] Ruizhe Li, Xiao Li, Guanyi Chen, and Chenghua Lin. Improving variational autoencoder for text modelling withtimestep-wise regularisation. In *COLING*, 2020.

[7] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[8] Sayak Paul. Reparameterization trick in variational autoencoders, 2020.

[9] Sabrina Rashid, Sohrab Shah, Ziv Bar-Joseph, and Ravi Pandya. Dhaka: variational autoencoder for unmasking tumor heterogeneity from single cell genomic data. *Bioinformatics*, 02 2019. btz095.

[10] Rahul Satija, Jeffrey A. Farrell, David Gennert, Alexander F. Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*, 33(5):495–502, May 2015.

[11] Hasan K. Siddiqi, Peter Libby, and Paul M Ridker. Covid-19 – a vascular disease. *Trends in Cardiovascular Medicine*, 2020.

[12] Nikola Simidjievski, Cristian Bodnar, Ifrah Tariq, Paul Scherer, Helena Andres Terre, Zohreh Shams, Mateja Jamnik, and Pietro Liò. Variational autoencoders for cancer data integration: Design principles and computational practice. *Frontiers in Genetics*, 10:1205, 2019.

[13] A.K Subramanian. Pytorch-vae. `https://github.com/AntixK/PyTorch-VAE`, 2020.

[14] Jeet Sukumaran and Mark T. Holder. DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571, 04 2010.

[15] Javier Igea Victor Soria-Carrasco, Gerard Talavera and Jose Castresana. The k tree score: quantification of differences in the relative branch length and topology of phylogenetic trees.

[16] Douglas White and Bernard Lo. A framework for rationing ventilators and critical care beds during the covid-19 pandemic, 2020.

[17] Aaron J. Wilk, Arjun Rustagi, Nancy Q. Zhao, Jonasel Roque, Giovanny J. Martínez-Colón, Julia L. McKechnie, Geoffrey T. Ivison, Thanmayi Ranganath, Rosemary Vergara, Taylor Hollis, Laura J. Simpson, Philip Grant, Aruna Subramanian, Angela J. Rogers, and Catherine A. Blish. A single-cell atlas of the peripheral immune response in patients with severe covid-19. *Nature Medicine*, 26(7):1070–1076, Jul 2020.

[18] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1):15, Feb 2018.

[19] Dega Ravi Kumar Yadav and Gunes Ercal. A comparative analysis of progressive multiple sequence alignment approaches using upgma and neighbor join based guide trees. *International Journal of Computer Science, Engineering and Information Technology*, 5(3/4):1–9, 2015.

[20] Mehmet Yamac, Mete Ahishali, Aysen Degerli, Serkan Kiranyaz, Muhammad E. H. Chowdhury, and Moncef Gabbouj. Convolutional sparse support estimator based covid-19 recognition from x-ray images, 2020.