Interpretable deep generative models for genomics

Yongin Choi[1,2,*], Gerald Quon[1,2,3,*]

[1]Biomedical Engineering Graduate Group, [2]Genome Center, [3]Department of Molecular and Cellular Biology, University of California, Davis, Davis, CA

*To whom correspondence should be addressed: gquon@ucdavis.edu

ABSTRACT

Deep neural networks implementing generative models for dimensionality reduction have been extensively used for the visualization and analysis of genomic data. One of their key limitations is lack of interpretability: it is challenging to quantitatively identify which input features are used to construct the embedding dimensions, thus preventing insight into why cells are organized in a particular data visualization, for example. Here we present a scalable, interpretable variational autoencoder (siVAE) that is interpretable by design: it learns feature embeddings that guide the interpretation of the cell embeddings in a manner analogous to factor loadings of factor analysis. siVAE is as powerful and nearly as fast to train as the standard VAE but achieves full interpretability of the embedding dimensions. We exploit a number of connections between dimensionality reduction and gene network inference to identify gene neighborhoods and gene hubs, without the explicit need for gene network inference. Finally, we observe a systematic difference in the gene neighborhoods identified by dimensionality reduction methods and gene network inference algorithms in general, suggesting they provide complementary information about the underlying structure of the gene co-expression network.

INTRODUCTION

Single cell genomic assays such as scRNA-seq and scATAC-seq measure the activity level of tens to hundreds of thousands of genomic features (genes or genomic regions), yielding high dimensional measurements of cells. Features tend to be inter-correlated: gene members of the same pathway, complex or module exhibit correlated expression patterns across cells[1], and proximal genomic regions covering the same regulatory elements or expressed genes are correlated in their accessibility patterns[2]. Key analysis tasks such as visualization[3], clustering[4], trajectory inference[5,6], and rare cell type identification[7,8] typically do not directly compute on the original features. Instead, they first perform dimensionality reduction (DR) to project cells from their high dimensional feature space to a lower dimensional "cell embedding space" consisting of a smaller set of embedding dimensions. Individual embedding dimensions capture distinct groups of correlated input features, and are often also correlated with biological factors such as case-control status[9], gender[10], and others[11]. Downstream tasks are then carried out on these embedding dimensions.

Given the central role of embedding dimensions in single cell analysis tasks, it is useful to be able to characterize which of the original input features contributed to the construction of each embedding dimension; that is, it is useful to "interpret" the embedding dimensions with respect to the input features. For example, in a visualization of a 2D cell embedding space based on scRNA-seq data, interpretation of the embedding dimensions would identify genes that may be responsible for variation in the transcriptome along different axes of the visualization (**Fig. 1**). Our ability to interpret embedding dimensions rests directly on the dimensionality reduction technique used to compute them. DR approaches can be categorized based on whether they use a linear or non-linear reduction framework. Linear DR frameworks are considered less powerful because they can typically be viewed as a specific, restricted implementation of a non-linear framework[12]. The advantage of linear methods such as Principal Components Analysis (PCA) is that they provide a quantitative estimate of the contribution of individual features towards each embedding dimension. In contrast, while non-linear methods such as UMAP, t-SNE and variational autoencoders (VAEs) produce better visualizations in which cells of the same type cluster together more closely (**Supplementary Fig. S1**), they are not interpretable. That is, they do not estimate the contributions of features to individual embedding dimensions, and therefore require more ad hoc, downstream analysis to gain intuition about the arrangement of cells in the visualization (e.g. understand why specific cells cluster). Beyond visualization, interpretability is an important property for other tasks in single cell analysis, such as the detection of genes and pathways driving variation in expression within or across cell types[13], and for identification of genes associated with cellular trajectories directly from visualization[14].

Here we propose a scalable, interpretable variational autoencoder (siVAE) that combines the non-linear DR framework of variational autoencoders (VAEs) with the interpretability of linear PCA. VAEs are non-linear DR methods that uses neural networks to infer a cell embedding space. siVAE is a variant of VAEs that additionally infers a feature embedding space for the genomic features (genes or genomic regions) that is used to interpret the cell embedding space, while being comparable in training time and power to a standard VAE. Compared to other approaches to achieving interpretable, non-linear DR, siVAE is either faster, generates better low dimensional representations of cells, or more accurately interprets the non-linear DR. We found that the feature embeddings that siVAE learns are useful for characterizing aspects of gene regulatory networks (GRN) while avoiding the challenging process of gene network inference[15]. More specifically, we show that the feature embeddings can be used to identify communities of co-regulated genes. siVAE can also find groups of co-regulated genes that are not readily identified by GRN methods, suggesting the two approaches to identifying co-regulated gene sets are complementary in their findings. Finally, we demonstrate siVAE identifies genes with high degree centrality more accurately than ranking genes by explicit node degree in a gene network, suggesting siVAE can be used to find central genes in the genome. Our framework for making VAEs interpretable is generalizable to other VAE-based frameworks. Given that VAEs have been applied to a wide range of genomics data modalities (epigenomics[16–18] and miRNA[19]) and analysis (visualization[20,21], trajectory inference[22], data imputation[23], and perturbation response prediction[24–26]), our work can therefore enable interpretability in a wide range of downstream applications of VAEs.

RESULTS

siVAE is a deep neural network consisting of two pairs of encoder-decoder structures, one for cells and the other for features (**Fig. 1a**). The cell-wise encoder-decoder learns to compress per-cell measurements $X_{c,:}$ (where X is a matrix of dimension $C \times P$) into a low dimensional embedding (z_c) of length K for visualization and analysis, similar to traditional VAEs implemented in single cell genomic applications and others[21,27,28]. We call the $C \times K$ matrix of embeddings $Z_{c,k}$ the siVAE score matrix, where the scores of cell c ($Z_{c,:}$) represent its position in the cell embedding space.

To facilitate interpretation of the cell state space, siVAE additionally implements a separate feature-wise encoder-decoder network (**Fig. 1a**) that learns to compress per-genomic features across the cells ($X_{:,f}$) into a low dimensional embedding (v_f) of length K, analogous to the cell-wise encoder-decoder. We call the $F \times K$ matrix of feature embeddings $V_{f,k}$ the siVAE loading matrix, where the loadings of feature f ($V_{f,:}$) represent its position in the feature embedding space. The cell- and feature-wise decoders together are used to generate the observed measurement $X_{c,f}$.

The strategy siVAE uses to achieve interpretation is best understood by briefly reviewing why probabilistic PCA (PPCA) and factor analysis are interpretable[29,30]. The underlying generative model behind PPCA can be thought of as similar to a VAE with a linear decoder, and the output of PPCA includes both a factor loading matrix $V_{f,k}$ and score matrix $Z_{c,k}$. In probabilistic PCA, the predicted expression of feature f in cell c ($X_{c,f}$) is assumed to be $V_{f,:}^T Z_{c,:}$, the dot product of the loadings for feature f and the scores of cell c. PPCA is therefore interpretable, because the larger the contribution of a feature f to a particular dimension k (indicated the magnitude of $V_{f,k}$), the more the measurement of feature f ($X_{c,f}$) is influenced by a cell's corresponding score in that dimension ($Z_{c,k}$). Conversely, when the magnitude of $V_{f,k}$ is small (or even 0), then the cell's corresponding score in that dimension ($Z_{c,k}$) does not influence $X_{c,f}$, the measurement of feature f in cell c. In this regard, we say that the PPCA model enforces correspondence between $Z_{c,k}$ and $V_{f,k}$, the cell and feature embedding at dimension k.

siVAE achieves interpretability of the siVAE scores $Z_{c,k}$ by adding a small interpretability regularization term to its objective function (see Methods). More specifically, this regularization term penalizes deviation between the observed measurement $X_{c,f}$, and the dot product of the corresponding siVAE scores and loadings ($V_{f,:}^T Z_{c,:}$). This small regularization term helps enforce some soft correspondence between dimension k of the cell scores, and dimension k of the feature loadings.

RESULTS – siVAE accurately generates low dimensional embeddings of cells

We first evaluated siVAE in the context of cell embedding space inference, where the goal is to generate low dimensional representations of cells in which cells of the same cell type cluster together. We benchmarked siVAE against other interpretable and non-interpretable dimensionality reduction approaches using a fetal liver cell atlas[31] consisting of 177,376 cells covering 41 cell types. We measured the accuracy of each approach in a 5-fold stratified cross validation framework by first using the training folds to learn a cell embedding space, followed by training of a k-NN ($k = 80$) classifier using the known cell type labels and cell coordinates within the embedding space. We then classified the held-out cells. We associate higher k-NN accuracy with a more accurate cell embedding space in which cells of the same type cluster together.

We compared siVAE against a classic VAE as well as LDVAE[29], where all three VAE frameworks used cell-wise encoder-decoders of the same size, and the VAE and siVAE use the same activation functions. Overall, we found siVAE's cell embedding space to be comparable in accuracy to classic VAEs, suggesting that the introduction of the siVAE feature-wise encoder-decoder does not affect siVAE performance in terms of its cell embedding space. 2D visualization of siVAE's cell embedding space reveals strikingly similarity to the cell embedding space of the classic VAE in that cells of the same type cluster together (**Fig. 2a**). Furthermore, siVAE is competitive in classification accuracy with a classic VAE on the fetal liver cell atlas (**Fig. 2b**).

siVAE therefore is competitive with VAEs in terms of generating cell embedding spaces, but has the additional benefit of interpretability, which we will explore below. In comparison, the LDVAE approach, which is interpretable like siVAE but performs linear DR, yields significantly lower classification accuracy (**Fig. 2b**) and generates visualizations in which different cell types mix together more prominently (**Fig. 2a**). LDVAE therefore gains interpretability at a cost to the accuracy of the cell embedding space.

We next constructed a set of model variants of siVAE in order to identify which aspects of siVAE lead to its superior performance over LDVAE (**Table 1)**. LDVAE is in principle similar to the classic VAE, with two key differences. First, the LDVAE decoder is restricted to use only linear activation functions in order to achieve interpretability; thus, LDVAE performs linear dimensionality reduction. scETM performs a similar dimensionality reduction, but further breaks down the loading matrix through tri-factorization[32]. Second, the LDVAE loss function uses a negative binomial or zero-inflated negative binomial distribution over the input features (genes), instead of the Gaussian distribution used in a classic VAE. In principle, the NB or ZINB observation model is a better fit for single cell transcriptomic data compared to a Gaussian distribution normally used on log transformed data [33,34]. We therefore constructed two variants of siVAE, siVAE-NB and siVAE-linear. siVAE-NB is identical to siVAE, except that it uses a negative binomial distribution for the observation layer while maintaining non-linear activation functions in its decoders to achieve non-linear DR. siVAE-linear is identical to siVAE, except that it restricts both the feature-wise and cell-wise decoder to use linear activation functions like LDVAE and does not implement the interpretability term. **Fig. 2b and Supplementary Fig. S2** shows that siVAE-NB performs worse than the corresponding model with the Gaussian distribution (siVAE), suggesting that using a NB output layer does not lead to a more accurate cell embedding space. siVAE-linear is more accurate than LDVAE (**Fig. 2b**), indicating that the feature-wise encoder-decoder of siVAE is overall beneficial to dimensionality reduction. However, siVAE-linear performs more poorly than siVAE, verifying the non-linear activation functions are beneficial to dimensionality reduction.

We also hypothesized that the interpretability term used in siVAE's loss function would degrade the quality of dimensionality reduction to an extent, as siVAE uses the regularization term to enforce correspondence between the individual dimensions of the feature and cell embedding spaces to achieve interpretability. We therefore constructed siVAE (γ=0), representing a siVAE model in which we turn off the regularization term by setting its weight γ to 0 and therefore disable interpretation, but keep the feature embedding space. From **Figure 2b**, we can see the small gap in classification performance between siVAE and VAE closes with siVAE (γ=0), showing that the intepretability of siVAE comes at a small cost in classification performance, though not nearly as large a cost as using linear dimensionality reduction, as evidenced by the poorer performance of siVAE-linear and LDVAE.

Finally, we repeated our classification experiments on three other image-based datasets (MNIST, Fashion-MNIST and CIFAR-10) to demonstrate our results in the fetal liver atlas generalize to data from different data modalities. We chose imaging datasets specifically to help validate our interpretation of the embedding spaces in the next section. Across these three image-based datasets, siVAE classification performance is within 1.6% of the VAE classification performance on average, again suggesting that interpretability comes at low cost to model performance. Interestingly, siVAE (γ=0) classification performance is actually slightly higher (1.4%) than the VAE on average (**Fig. 2c**), suggesting the feature-wise encoder-decoder improves performance.

RESULTS – siVAE interprets cell embedding spaces faster and more accurately than existing feature attribution approaches

Having shown siVAE generates cell embedding spaces competitive with classic VAEs, we next verified that the interpretations of the embedding dimensions output by siVAE are accurate. Again, we define an interpretation of the cell embedding space as a matrix of feature loadings (or more generally, attributions) $V_{f,k}$ of size $F \times K$, where F is the number of features (e.g. genes), K is the number of cell dimensions, and the magnitude of $V_{f,k}$ indicates the strength of association between cell state dimension k and feature f in the original data space.

In contrast to methods such as siVAE and LDVAE that construct interpretable cell embedding spaces by design, there are two competing types of approaches to feature attribution in the literature that can help interpret cell embedding spaces post-inference. First, siVAE feature embeddings are analogous to general neural network feature attribution methods that quantify how each output node of a neural network depends on each input node (feature) of the network[35], and include methods such as DeepLIFT[36], saliency maps[37], grad x input[36], integrated gradients[38], Shapley value[39] and others[35,40–44]. One of the strengths of these approaches is they can be applied to any trained neural network in principle, making them highly generalizable. Second, methods such as Gene Relevance[45] have been developed specifically to interpret cell latent spaces for any DR method including those not based on neural networks, and can be applied after cell embedding spaces are learned.

We first compared siVAE against Gene Relevance, using the neural network feature attribution methods as a gold standard as they have been extensively validated in other applications[46]. **Figure 3a** shows the mean pairwise correlation between the attributions of siVAE, Gene Relevance as well as three neural net feature attribution methods (saliency maps, grad x input, and DeepLIFT), where correlations have been averaged over each of the two feature dimensions that siVAE used to infer the cell embedding space for the fetal liver dataset. We see siVAE loadings are highly correlated with the neural net feature attribution methods (median Spearman correlation of 0.73, p=1.1e-15) with siVAE in striking agreement with DeepLIFT in particular (median Spearman correlation of 0.98, p=2.2e-16). In contrast, while Gene Relevance produced feature attributions that were consistent across their parameter selections (median Spearman correlation of 0.84, p=3.10e-22), they were poorly correlated with both neural net feature attribution methods (median Spearman correlation of 0.11, p=2.1e-6) and siVAE (median Spearman correlation of 0.14, p=3.9e-6). These results suggest Gene Relevance is less competitive with siVAE at interpreting cell embedding spaces of VAE architectures.

To provide a visual comparison of feature attributions, we trained siVAE to perform dimensionality reduction on a subset of the MNIST dataset consisting of black and white digits[47]. We focused on the digits 1 and 6 to ensure a human-visible separation of digits along individual embedding dimensions. Visualization of the cell embedding space confirms that images separate by digit as expected (**Fig. 3b**). We then ran all feature attribution methods to interpret each axis of **Fig. 3b** and found that again siVAE interpretations agreed more strongly with the neural net attribution methods (median Spearman correlation of 0.48, p=2.4e-27) compared to Gene Relevance (median Spearman correlation of 0.35, p=0.10). Visual inspection of the cell embedding space (**Fig. 3b**) suggest that the x-axis mainly captures variation within each of the 1 and 6 classes of digits, while the y-axis primarily distinguishes between the two digits, a finding consistent with the interpretations of the attribution methods (**Fig. 3c**). Additional experiments on the whole MNIST dataset with all digits also consistently show siVAE interpretations are in better agreement with feature attribution methods (median Spearman correlation of 0.89, p=1.07e-11) compared to Gene Relevance (median Spearman correlation of 0.12, p=0.14) (**Supplementary Fig. S4**).

For the above results, we applied the neural net attribution methods to the decoder of siVAE to generate the ground truth feature attributions. Previous work has suggested to apply attribution methods to the encoder to improve execution speed[28,48]. Here we found that running attribution methods on the siVAE encoder produce substantially different interpretations that are in strong disagreement with interpretations of the decoder (**Supplementary Fig. S5**) and inconsistent with the cell embedding space (**Fig. 3b**), suggesting interpretation of the encoder is not appropriate. These results make sense considering the primary role of the encoder is to compute an approximate posterior distribution of the latent embedding of each image (or cell), as opposed to the decoder, which is responsible for generating each image (or cell). Our results therefore suggest feature attributions should be applied to the decoder of VAEs instead of the encoder.

During our experiments on interpreting cell embedding spaces, it became evident that a number of neural network feature attribution approaches were computationally expensive to execute. Because these feature attribution methods perform calculations separately for either each embedding dimension or each output node of the network, their run time scales linearly with the number of embedding dimensions or features when run on VAE decoders[21]. The number of embedding dimensions is expected to be larger as the number of cells in the dataset grows, to accommodate more heterogeneity in the dataset; also, the number of features would be expected to be large for assays such as scATAC-seq that profile hundreds of thousands of genomic regions or more. We conjecture this problem of long execution time has not been previously reported in the literature because feature attribution methods are typically run on supervised neural networks to interpret class label predictions, and so the number of output nodes is traditionally very small, unlike generative models in genomics applications.

We therefore hypothesized that siVAE scales faster than the neural network attribution methods on larger single cell genomic datasets. To test this hypothesis, we assembled two datasets for execution time testing: the LargeBrainAtlas dataset published by 10x Genomics[49] consisting of 1.3 million brain cells and 27,998 genes measured with scRNA-seq, and the BrainCortex dataset[50] consisting of 8k cells and 244,544 genomic regions measured with SNARE-seq. We first compared the execution time of training siVAE on the full LargeBrainAtlas dataset, against the run time of training a VAE and individually running each of five neural network attribution methods (saliency maps, grad*input, DeepLIFT, integrated gradients and Shapley values) to the trained VAE. We found that siVAE achieved an execution time of 2.5 days, less than half of the fastest neural network attribution method (forward mode of saliency maps) (**Fig. 3d**).

To identify the most time-consuming aspects of feature attribution calculations for each method, we selected a subset of the LargeBrainAtlas dataset for varying the number of embedding dimensions from 20 to 512 and a subset of BrainCortex dataset for varying the number of features from 28k to 240k, to identify the speed bottlenecks. siVAE averaged 0.0073 days per embedding dimension (**Fig. 3e**) and 0.0027 days per 10k features (**Fig. 3f**), indicating siVAE execution time was robust to both the number of cells and features. On the other hand, we found the neural network attribution methods scale well when either the number of embedding dimensions or the number of input features is large, but not when they are both large. For example, DeepLIFT, Grad*Input (reverse-mode) and Saliency maps executed at 0.014, 0.0053, and 0.0012 days per embedding dimension respectively (**Fig. 3e**), but scaled poorly with respect to number of input features and executed at 2.9, 0.95, and 0.94 days per 10k features respectively (**Fig. 3f**). Switching Grad*Input and Saliency Maps to forward-mode led to fast execution times with respect to the number of input features (5.3e-4 and 6.5e-4 days per 10k features respectively (**Fig. 3f**)) but led to poor scaling with respect to the number of embedding dimensions (0.18 and 0.17 days per embedding dimension, respectively (**Fig. 3e**)). Slower attribution methods such as Integrated Gradients and Shapley Value were excluded due to their infeasible execution times. In summary, the neural network attribution methods scale poorly either with the number of embedding dimensions or the number of input features depending on whether forward- or reverse-mode is used. This therefore makes their execution time slow relative to siVAE if both the number of features and embedding dimensions is large.

RESULTS – Co-expressed genes cluster in the feature embedding space

Feature attributions, or factor loadings, of linear DR methods such as PCA have been exploited extensively in the literature to gain insight into the gene co-expression network (GCN); here we explore the extent to which the siVAE loading matrix can be leveraged to gain insight into GCN structure. GCNs are graphs in which nodes represent genes and edges represent co-expression of a pair of genes. A GCN captures co-variation in gene expression measurements between pairs (or more) of genes across a population of cells. GCNs are of interest because they can be used to identify (1) cell population-specific gene modules, representing groups of genes that are highly co-expressed and therefore are likely to function together in a cell type-specific manner, as well as (2) gene hubs, which are genes that are connected via an edge to many more genes than is typical in the network, and typically represent key functional genes in the cell[51,52]. While GCN inference is valuable for interrogating gene regulatory patterns in a cell, GCN inference is a notoriously difficult and error-prone task[53,54] because inferring them involves estimating many more parameters than the amount of data typically used to infer them[55].

In our application of dimensionality reduction in which features are all centered and scaled uniformly, the goal of DR methods is to learn (linear or non-linear) patterns of co-expression amongst the input features, that allow accurate reconstruction of the input data from low dimensional representations. It is therefore natural to ask whether a trained siVAE model could yield insight into gene co-expression network structure of the training data, without the need for explicit gene network inference. More specifically, we view the siVAE loading matrix that siVAE infers as a non-linear analog of the PCA loading matrix. Indeed, one can view probabilistic PCA[56] as a restricted form of a VAE in which all of the activation functions in the decoder are linear, no regularization is applied to the decoder weights, and the output distribution is an isotropic Gaussian.

Previous work has shown that eigengenes (genes captured by factor loadings of PCA) represent network modules in the gene co-expression network[57,58]. We hypothesized that siVAE genes captured by feature loadings of siVAE may also represent network modules, and that co-expressed genes in the training data are also proximal in the siVAE feature embedding space. To explore how a group of co-expressed genes are organized in the feature embedding space, we constructed a synthetic gene regulatory network consisting of five communities of 50 tightly correlated genes each, as well as an additional group of 50 independent, isolated genes (**Fig. 4a**). Each community follows a hub-and-spoke model in which a hub gene is connected to every other gene in the community, and each gene in the community is in turn only connected to the hub. No edges connect genes from different communities. Based on this gene network, we sampled a single cell gene expression dataset consisting of 5,000 cells and 300 genes (see Methods). The sampled expression matrix was used to train siVAE to embed genes in its feature embedding space.

We found that genes belonging to the same community co-localized in the feature embedding space, but interestingly, the hub nodes are embedded in distinct locations their corresponding community (**Fig. 4b**). Our interpretation of this observation is that given the limited capacity of the cell embedding space, siVAE tends to keep information specifically about each hub because of their high degree centrality and uses each hub to reconstruct the remaining nodes of their corresponding community. On the other hand, non-hub genes within the same community co-localize in the feature embedding space because the limited capacity of the VAE forces non-hub genes to be predicted similarly, given the retained information about the hub. Interestingly, the 50 independent, isolated genes in the network were clustered tightly but near the origin in the feature embedding space, whereas genes that are part of a community are clustered but located farther away from the origin. This is likely because of two reasons. First, the KL divergence term of the feature-wise encoder-decoder of siVAE will tend to draw genes towards the origin. Second, because isolated genes by definition do not co-vary with other genes, information about their expression pattern will tend to be lost during compression, leading the VAE to tend to predict the average expression level of that gene in the decoder (which will be 0, because of data centering). This in turn encourages the feature embedding to be at the origin because the interpretability term encourages the linear product of the feature embedding with the feature loadings to predict the gene's expression pattern, so if a feature is located at the origin in the feature embedding space, it will cause the predicted expression to be 0.

We also confirmed that co-expressed genes cluster in the feature embedding space using the fetal liver cell atlas data. Unlike the simulations above, for the fetal liver atlas there are no ground-truth gene regulatory networks available to use to identify truly co-expressed genes that are part of the same underlying gene communities. We therefore trained siVAE on the entire fetal liver atlas with 40 cell types, and considered marker genes of the same cell type[59] to be a ground truth set of co-expressed genes. We selected only the four cell types with available marker genes for visualization (**Fig. 4c**). In the resulting feature embedding space learned by siVAE, we see that markers of the same cell type tend to cluster in feature embedding space as expected (**Fig. 4d**). Our results overall suggest that co-expressed genes tend to co-localize in siVAE feature embedding space.

RESULTS – Gene hubs can be identified without explicit gene network inference

Our observation that hub genes in a community are treated differently by siVAE led us to hypothesize that we may be able to identify hub genes from a trained siVAE model without inferring a GRN. Hub genes are often identified after GRN inference because they play essential roles both in terms of the structure of the network and the genome itself, and are often targets of genetic variants associated with disease[60,61]. We reasoned that because hub genes are connected to so many other genes, siVAE is more likely to store the expression patterns of hub genes in the compressed representation (latent embedding) for use in reconstructing the rest of the gene expression patterns. We therefore hypothesized that we could identify hub genes as those genes that are well reconstructed by a trained siVAE model, because if siVAE captures variation in hub gene expression in the cell embedding space, it should also reconstruct the hub gene expression more accurately than other genes. We therefore used gene-specific reconstruction accuracy in the siVAE model as GRN-free measure of degree centrality. As a ground truth measure of gene centrality, we calculated each gene's individual ability to predict the expression levels of every other gene in the genome (see Methods), reasoning that a 'hub' gene should be predictive of many other genes in the network.

Figure 5a compares siVAE's estimate of gene centrality with gene centrality calculated on GRNs inferred using a number of existing GRN inference algorithms (see Methods). Overall, siVAE has the highest correlation between its predicted gene centrality and the ground truth centrality (Spearman rho=0.90, p = 2.2e-16), significantly larger than other approaches (median Spearman rho=0.36, p = 9.1e-11). When identifying hub genes in a network, it is typical to focus on the genes with highest predicted centrality. We found that the top 20 genes with highest predicted degree centrality for siVAE has mean ground truth degree centrality of 0.092. This compares favorably to the GRN inference methods, for whom the median of mean ground truth degree centrality is 0.074 for the top 20 most central genes identified by the GRN inference methods. **Figure 5b** illustrates the cumulative ground truth degree centrality of the top predicted hubs according to each method, and siVAE consistently selects genes with the largest cumulative degree centrality of all tested methods. These results in total suggest that using siVAE, we can identify high degree centrality genes more accurately than the more classic approach of first inferring a gene co-expression network before identifying high degree centrality genes.

RESULTS – Systematic differences in gene neighbors identified by dimensionality reduction and network inference methods

Finally, we explored the extent to which we could identify neighboring genes that share an edge in a GRN, without having to infer GRNs explicitly. Gene neighbors tend to share similar function[62], interact with one another[63] and/or belong to the same gene community[64]. Identification of gene neighbors therefore aids in identifying co-functional genes in the cell.

Here we hypothesized that we could identify gene neighbors directly using a trained siVAE model, instead of having to first infer an explicit GRN. GRN inference methods typically output edge weights between pairs of nodes in the network, where larger weights correspond to a greater chance the two nodes share an edge in the underlying GRN. For siVAE, we generate edge weights in two ways: (1) siVAE-Euc, the Euclidean distance of the two genes in the feature embedding space, where smaller distances correspond to closer proximity, and (2) siVAE-GRN, where we first sample a new scRNA-seq dataset from siVAE that matches the size of the training data, then run a GRN inference method (ARACNE, MRNET, CLR, and GRNBOOST2) on the sampled scRNA-seq dataset to calculate edge weights between genes. To quantitatively evaluate the accuracy of neighbor identification using each method, we measured the percentage variance explained of a given query gene when predicted by the expression levels of the nearest 20 genes ranked by edge weight to the query gene (see Methods). In our evaluations, we only consider the 152 query genes which were predicted to have high degree centrality across all tested methods (see Methods).

Overall, most methods identified neighbors that were equally predictive of the 152 query genes' expression levels (**Fig. 5c**). Excluding LDVAE and ARACNE, the median % variance explained for each method was 79.9% $\pm$ 0.84 s.d. **Supplementary Fig. S10** illustrates that excluding LDVAE and ARACNE, the pairwise difference in % variance explained between methods is only 0.013% on average. Notably, we observed lower % variance explained for LDVAE and ARACNE (on average, 77.2% variance explained, and 78.3% variance explained, respectively). The poorer results of LDVAE are consistent with our classification performance results above.

When considering the overlap in neighbors selected by different methods, it is striking how the dimensionality reduction methods cluster strongly (scVI, siVAE, LDVAE) and the GRN inference-based methods cluster strongly as well, with markedly less overlap between these two groups (**Fig. 5d)**. This is surprising in part because the neighborhood sets are all approximately of the same predictive performance (**Fig. 5c**), suggesting the DR methods are systematically identifying different neighbors that are as equally co-expressed as the set identified by the GRN methods. In particular, consider that siVAE-GRN involves identifying gene neighbors using the GRN inference methodology, but just applied to a siVAE-generated dataset (instead of the original training dataset). **Figure 5c** illustrates that under the siVAE-GRN neighborhood identification framework, the neighborhood genes are still much more similar to siVAE than to the GRN inference methods, suggesting the unique neighborhood identified by the DR methods is a property of the co-expression patterns that DR methods learn, and not due to the way in which neighborhood genes are identified. The poor overlap between the DR and GRN methods also holds true if we consider the average pairwise correlation in expression between neighbor sets, instead of considering overlap of genes (**Fig. 5e**). More specifically, the GRN-defined neighbor sets had higher average Pearson correlation amongst themselves (average Pearson correlation = 0.67, excluding ARACNE) compared to the average Pearson correlation coefficient among the neural net-based neighbor sets (average Pearson correlation = 0.46). There was also low average correlation between DR and GRN neighbor sets (average Pearson correlation = 0.39). Our results therefore suggest that since GRN- and dimensionality reduction-identified neighbor sets are systematically different but approximately equally predictive of neighboring genes, then both approaches should be used to find co-expressed genes in a network.

Discussion

Through the development of siVAE, we have mitigated one of the primary limitations of the interpretation of VAEs: the slow execution time of neural network feature attribution methods when the number of input features and embedding dimensions of the cell embedding space are both large. Single cell atlases are ever-increasing in size due to the dropping cost of single cell sequencing[65]. Also, there is rapidly increasing interest and development of multi-modal single cell assays such as SNARE-Seq[66], ECCITE-Seq[67], and SHARE-Seq[68] that measure multiple data modalities (RNA, ATAC) simultaneously and are yielding single cell measurements with up to hundreds of thousands of input features, which will then demand large cell embedding spaces to accurately capture covariation in input features. As such, we expect the importance of scalable, interpretable VAEs will continue to grow.

Our analysis has also demonstrated how interpretation of cell embedding spaces can lead to insight into the gene regulatory networks underlying the cell population siVAE is trained on. In addition to showing how co-expressed groups of genes can be identified, we also showed how we can identify hub genes, without inferring an explicit GCN. This is useful because GCN inference continues to be a highly challenging task, even in the era of large numbers of cells sequenced from single cell assays[15]. By comparing VAEs trained on different cell populations, it is likely possible to identify differential co-expression patterns between cell populations, also without explicit GCN inference.

A surprising observation we made was that the set of neighbors of a given gene with respect to the underlying GCN is systematically different between explicit GCN inference methods and the dimensionality reduction methods. This even holds true when a trained siVAE model was used to sample expression data that was then sent as input into a classic GCN inference method; in this scenario, the resulting GCN yielded neighbors that were similar to those identified directly from the DR methods. Our experiments further showed that both neighborhood sets are equally co-expressed with the query gene, suggesting at the least that accurate GCN construction should leverage both of these types of approaches to identifying gene neighbors. One possible explanation is that DR methods can learn to combine many genes into a single embedding dimension, whereas explicit GCN inference methods ultimately represent co-expression patterns as individual edges between only pairs of genes, and therefore are more limited in their capacity to represent higher order co-expression patterns.

While we have chosen the classic VAE framework upon which to build siVAE, our approach to introducing a feature-wise decoder and interpretability term is generalizable and can be applied to other general extensions of VAE, such as VAE-GANs, β-VAE among others [69,70]. With respect to genomic data modalities such as epigenomics, miRNA and scRNA-seq, methods such as RE-VAE[16], methCancer-gen[17], VAEMDA[19], scMVAE[18], scVI[21], Dr.VAE[25], scGen[26], and Dhaka[22] could also benefit from similar interpretability terms such as that used for siVAE. As such, our work on making VAEs interpretable can be applied across the diverse domains [21,27,71] that VAEs have and will be applied to in the future.

A related set of approaches to increasing interpretability of generative models focuses on disentanglement learning. In particular, methods such as InfoGAN[72], FactorVAE[48], DirVAE[73] and others[70,74,75] modify generative models such as the VAE to achieve disentangled representation by encouraging the individual cell dimensions to be statistically independent. They show that independence between cell dimensions oftentimes leads to more correspondence between individual cell dimensions and tangible factors such as width and rotation of digits for MNIST. However, we do not consider these model variants here because they do not provide contributions of individual features to cell dimensions. These approaches still require users to manually draw samples of points from the cell embedding space, reconstruct the input features from the cell dimensions, then use human intervention to manually inspect how variation across specific dimensions might correspond to human-interpretable factors of variation. However, the regularization terms that encourages disentanglement between the cell dimensions may be applied to siVAE. This would help remove the entanglement between cell dimensions such as the overlapping outlines of digits in siVAE loadings for the MNIST dataset.

ACKNOWLEDGEMENTS

This publication was made possible by an NIGMS funded predoctoral fellowship to YC (T32 GM007377). GQ was supported by NSF CAREER award 1846559. This project has been made possible in part by grant number 2019-002429 from the Chan Zuckerberg Foundation. This work was supported by the National Institute of Child Health and Human Development P50 HD103526.

METHODS

Model notation. We denote vectors as lower case, bold letters (e.g. $\boldsymbol{z}$). Matrices are upper case letters with two subscript indices (e.g. $X_{c,f}$). Constants are upper case letters with no subscripts (e.g. L).

Generative process of VAEs. siVAE is an extension of a classic variational autoencoder. Here we briefly review the generative process assumed by a standard VAE with L hidden layers in the decoder, and in which the hidden units of the last layer of the decoder are linearly transformed into the predicted mean of the Gaussian distribution over the observed data:

$$\boldsymbol{z}_{c,1} \sim N(0, I_K) \tag{1}$$
$$\boldsymbol{z}_{c,\ell} = \mu_\ell(\boldsymbol{z}_{c,\ell-1}),\ \ell = 2, \dots, L \tag{2}$$
$$X_{c,f} \sim N(\boldsymbol{v}_f^T \boldsymbol{z}_{c,L}, \sigma_d(\boldsymbol{z}_{c,L})) \tag{3}$$

$X_{c,f}$ is the input observed value for feature (e.g. gene) f and cell c (centered and scaled across all cells), where we assume there are F features and C cells in the training data. $\boldsymbol{z}_{c,1}$ is the embedding of cell c in the (latent) cell embedding space of the VAE, while $\boldsymbol{z}_{c,\ell}$ for $\ell > 1$ represent the activations of the hidden layer ℓ of the decoder for cell c. $\boldsymbol{v}_f$ is the vector of incoming weights to the predicted mean of the output node f of the VAE, while $\sigma_d(\cdot)$ is a one-layer function that predicts a non-negative scalar value representing variance. I_K is the identity matrix of rank K. $\mu_1(\cdot), \dots, \mu_L(\cdot)$ represent the parameterized activation functions of hidden layers $1, \dots, L$ of the cell-wise decoder, respectively.

Generative process of siVAE. The key idea behind siVAE is that we jointly infer cell-wise and feature-wise state spaces, and through regularization, loosely enforce correspondence between the cell and feature dimensions. Here, correspondence means variation in dimension k in the cell embedding space corresponds to observed variation in each feature f that is proportional to feature f's embedding coordinate in dimension k. Through correspondence, the feature embedding coordinates (`siVAE loadings') become analogous to factor loadings, and the cell embedding coordinates (`siVAE scores') become analogous to the factor scores of PCA. In siVAE, the feature and cell embeddings are sampled from different latent spaces and projected to higher dimensions through separate decoders, before combining to produce the means of the Gaussians (**Figure 1a**). The generative process assumed by siVAE is shown below:

$$\boldsymbol{z}_{c,1} \sim N(0, I_K) \tag{4}$$
$$\boldsymbol{z}_{c,\ell} = \mu_\ell(\boldsymbol{z}_{c,\ell-1}), \ell = 2, \dots, L \tag{5}$$
$$\boldsymbol{v}_{f,1} \sim N(0, I_K) \tag{6}$$
$$\boldsymbol{v}_{f,\ell} = \omega_\ell(\boldsymbol{v}_{f,\ell-1}), \ell = 2, \dots, L \tag{7}$$
$$X_{c,f} \sim N(\boldsymbol{v}_{f,L}^T \boldsymbol{z}_{c,L}, \sigma_d(\boldsymbol{z}_{c,L})) \tag{8}$$

Here $\boldsymbol{z}_{c,L}$, $\mu_\ell(\cdot)$, I_K and $\sigma(\cdot)$ are defined as above for VAEs. $\boldsymbol{v}_{f,1}$ is the latent embedding of feature f in the feature embedding space of siVAE, while $\boldsymbol{v}_{g,\ell}$ for $\ell > 1$ represent the activations of hidden layer ℓ of the feature-wise decoder for feature f. $\omega_1(\cdot), \ldots, \omega_L(\cdot)$ represent the activation functions of hidden layers $1, \ldots, L$ of the feature-wise decoder, respectively.

Comparing Equations 3 to Equations 6-8 illustrate that siVAE turns the last layer of weights leading to the Gaussian mean of the VAE into a non-linear transformation of the latent variables $\boldsymbol{v}_{f,1}$. siVAE can therefore be viewed as putting a prior over a single (last) layer of weights in the VAE. The matrix $V = [\boldsymbol{v}_{1,1}, \cdots, \boldsymbol{v}_{F,1}]^T$ encodes the siVAE loadings, while the matrix $Z = [\boldsymbol{z}_{1,1}, \cdots, \boldsymbol{z}_{C,1}]$ encodes the siVAE scores. Note that we can compute siVAE loadings and scores of other hidden layers ℓ as well, but in this paper, we focus on the latent space ($\ell = 1$).

Inference and training. We employ variational inference via a pair of encoder networks, $\psi(X_{:,f})$ for features and $\phi(X_{c,:})$ for cells, in a manner analogous to variational inference applied to VAEs. Note the input for the two encoders is different: $X_{:,f}$ is a vector of observations for a single feature f across all training cells, whereas $X_{c,:}$ is a vector of observations for a single cell c across all features. Our approximate posterior $q\left(\{\boldsymbol{v}_{f,1}\}_{f=1}^F, \{\boldsymbol{z}_{c,1}\}_{c=1}^C\right)$ factors as follows:

$$q\left(\{\boldsymbol{v}_{f,1}\}_{f=1}^F, \{\boldsymbol{z}_{c,1}\}_{c=1}^C\right) = \prod_{f=1}^{F} q(\boldsymbol{v}_{f,1}) \prod_{c=1}^{C} q(\boldsymbol{z}_{c,1}) \quad 9$$

$$q(\boldsymbol{v}_{f,1}) = N(\boldsymbol{v}_{f,1}; W_\psi^T \psi(X_{:,f}), \sigma_{e,\psi}(\psi(X_{:,f}))) \quad 10$$

$$q(\boldsymbol{z}_{c,1}) = N(\boldsymbol{z}_{c,1}; W_\phi^T \phi(X_{c,:}), \sigma_{e,\phi}(\phi(X_{c,:}))) \quad 11$$

We perform variational inference and learning by maximizing the expected lower bound function ℓ_{SIVAE}, where $\ell_{\text{KL}} = \text{KL}\left(q\left(\{\boldsymbol{v}_{f,1}\}_{f=1}^F, \{\boldsymbol{z}_{c,1}\}_{c=1}^C\right) \parallel p\left(\{\boldsymbol{v}_{f,1}\}_{f=1}^F, \{\boldsymbol{z}_{c,1}\}_{c=1}^C\right)\right)$.

$$\ell_{\text{SIVAE}} = \ell_{\text{KL}} + \mathbb{E}_{q(\boldsymbol{z}_{c,1}, \boldsymbol{v}_{f,1})}\left[\sum_c \sum_f \log N\left(X_{c,f}; \boldsymbol{v}_{f,L}^T \boldsymbol{z}_{c,L}, \sigma_d(\boldsymbol{z}_{c,L})\right)\right] \quad 12$$

$$-\gamma \mathbb{E}_{q(\boldsymbol{z}_{c,1}, \boldsymbol{v}_{f,1})}\left[\sum_c \sum_f \log N\left(X_{c,f}; \boldsymbol{v}_{f,1}^T \boldsymbol{z}_{c,1}, 1\right)\right] \quad 13$$

Interpretability term. The right-hand side of Equation 12 is analogous to the KL divergence and reconstruction loss terms of the original VAE lower bound function. The term in Equation 13, which we call the interpretability term, encourages the individual embedding dimensions of $\boldsymbol{v}_{f,1}$ and $\boldsymbol{z}_{c,1}$ to correspond to each other, by encouraging the linear products between $\boldsymbol{v}_{f,1}$ and $\boldsymbol{z}_{c,1}$ to approximate $X_{c,f}$. In our experiments, we set the penalty term $\gamma = 0.05$ to make the effect of the interpretability term small on the overall loss function.

Reducing dimensionality of input for feature-wise encoder-decoder. The size of input $X_{:,f}$ for feature-wise encoder-decoder increases with C, number of cells. To avoid the computational expenses of models with potentially millions of input nodes, we reduce the dimensionality of the input from C to C_{red} through either downsampling or PCA. For downsampled input, we randomly sample C_{red} cells while maintaining the ratio between the cell types which accounts for redundancy of information between cell types with similar gene expression patterns. For PCA input, we performed PCA without whitening on X^T, $G\ x\ C$ matrix and retained first C_{red} principal components resulting in X'^T, $G\ x\ C_{red}$ score matrix. The preservation of linear covariation with PCA is analogous to common usage of PCA before t-SNE or UMAP. In **Supplementary Fig. SXX**, we show training the feature-wise encoder-decoder with downsampled and PCA inputs both results in loss and clustering accuracy score comparable to that of model trained with the full data.

Training procedure for siVAE. We use a three-step training procedure to improve inference and learning, described in more detail in the Supplementary Materials:

- **Pre-train cell-wise encoder and decoder**. We first train the cell-wise encoder and decoder, similar to how a classic VAE is trained, by optimizing the Equation 12 component of ℓ_{SIVAE} with respect to $\{\mu_\ell, \sigma_d, \phi, \sigma_{e,\phi}, W_\phi\}$, and by treating the variables $\boldsymbol{v}_{f,L}$ as parameters to optimize to estimate $\tilde{\boldsymbol{v}}_{g,L}$. The input to the cell-wise decoder are the cell-wise data points $X_{c,:}$, and the output are the same data points $X_{c,:}$.
- **Pre-train feature-wise encoder and decoder**. We next train the parameters associated with the feature-wise encoder and decoder, namely $\{\omega_\ell, \psi, \sigma_{e,\psi}, W_\psi\}$, by training a VAE whose inputs are the data features $X_{:,f}$, outputs are $\tilde{\boldsymbol{v}}_{f,L}$ learned from the previous step, and whose encoder is defined by $\{\psi, \sigma_{e,\psi}, W_\psi\}$, and decoder parameterized by ω_ℓ, for $\ell = 1, \ldots, L-1$.
- **Train siVAE**. We finally train all model parameters $\{\mu_\ell, \sigma_d, \phi, \sigma_{e,\phi}, W_\phi, \omega_\ell, \psi, \sigma_{e,\psi}, W_\psi\}$ jointly by optimizing the full ℓ_{SIVAE} from Equation 12,13.

siVAE and VAE network design. For our experiments, identical neural net designs were used across the feature-wise and cell-wise encoders and decoders in siVAE. The architecture of the VAEs we compared against were matched to the architecture of the cell-wise encoder/decoders of siVAE. For MNIST and Fashion-MNIST, we set the architecture of the encoder to two hidden layers of sizes 512 and 128, and the decoder to two hidden layers of sizes 128 and 512. For all other datasets except the LargeBrainAtlas dataset, we set the architecture of the encoder to three hidden layers of sizes 1024, 512 and 128, and the decoder to three hidden layers of sizes 128, 512 and 1024. LargeBrainDataset, we trained an encoder with three hidden layers of sizes 2048, 1024, and 512, and the decoder with three hidden layers of sizes 512, 1024, and 2048. We used a latent embedding layer with size varying between 2, 5, 10 and 20 nodes for all imaging datasets. For the fetal liver atlas, we set the latent embedding layer size to be 2 for visualization and 64 for all other cases. In the timing experiment, we varied the latent embedding layer size between 20, 128 and 512 for the LargeBrainAtlas dataset, while setting the latent embedding layer size at 2 for the scATAC-Seq dataset. Additional implementation details as well as a table containing the above information on network design can be found in the Supplementary Materials (**Supplementary Table 1**).

siVAE and VAE model selection. We set model hyperparameters and optimization parameters by performing a hyperparameter search for the model with lowest total loss on the held-out data. For each model, we used the Adam optimizer for training, with a learning rate of either 0.0001, 0.001, or 0.01. We considered L2 regularization with scale factor λ of either 0.001 or 0.01. For imaging datasets, we set the number of embedding dimensions to 20. For genomic datasets, we used two embedding dimensions for models that were used for visualization, and otherwise considered sizes of 16, 32 and 64 for all other analyses.

siVAE model variants. To explore the role of different design choices of siVAE, we created several variants of the siVAE model described above. siVAE ($\gamma = 0$) removes the interpretability term in Equation 13 (by default, $\gamma = 0.05$). For comparison against LDVAE, whose decoder network ultimately predicts the parameters for negative binomial distributions instead of the parameters of a Gaussian distribution as implemented in siVAE and VAE, we also implemented both siVAE (NB) that predicts the parameters of a negative binomial distribution and VAE (linear) that is an identical implementation of LDVAE. siVAE (NB) is formulated as follows, where l_μ, l_σ parametrize the prior for scaling factor and are set to empirical mean and variance of the observed data:

$$l_c \sim \log\text{normal}(l_\mu, l_\sigma^2) \tag{14}$$
$$\rho_{c,f} = \text{softmax}\left(\boldsymbol{v}_{f,L}^T \boldsymbol{z}_{c,L}\right) \tag{15}$$
$$m_{c,f} \sim \text{Gamma}(\rho_{c,f}, \sigma_d(\boldsymbol{z}_{c,L})) \tag{16}$$
$$X_{c,f} \sim \text{Poisson}\left(l_c m_{c,f}\right) \tag{17}$$

$$\ell_{SIVAE_{NB}} = \ell_{\text{KL}} + \mathbb{E}_{q(\boldsymbol{z}_{c,1}, \boldsymbol{v}_{f,1})}\left[\sum_c \sum_f \log\text{Poisson}\left(X_{c,f}; l_c m_{c,f}\right)\right] \tag{18}$$

VAE (linear) is identical to siVAE (NB) except $\boldsymbol{v}_{f,L}^T$ is replaced by ϕ_f, an estimated parameter that matches the length of $\boldsymbol{z}_{c,L}$, thereby removing thefeature-wise encoder-decoder from the model. Finally, we implemented siVAE (linear), where the mean of the distribution over $X_{c,f}$ is directly predicted from linear multiplication of cell and feature embeddings. The reconstruction loss term corresponds to interpretability term, eliminating need for the latter.

$$X_{c,f} \sim N(\boldsymbol{v}_{f,1}^T \boldsymbol{z}_{c,1}, \sigma_d(\boldsymbol{z}_{c,1})) \tag{19}$$
$$\ell_{\text{SIVAE}_{\text{LINEAR}}} = \ell_{\text{KL}} + \mathbb{E}_{q(\boldsymbol{z}_{c,1}, \boldsymbol{v}_{f,1})}\left[\sum_c \sum_f \log N\left(X_{c,f}; \boldsymbol{v}_{f,1}^T \boldsymbol{z}_{c,1}, \sigma_d(\boldsymbol{z}_{c,1})\right)\right] \tag{20}$$

siVAE model availability. siVAE is implemented as a Python package and is available from PyPi (https://pypi.org/project/siVAE/).

LDVAE and scVI. We used LDVAE[29] and scVI[21] implemented in SCANPY[76] package available from PyPi. The architecture of the model was set to match that of the cell-wise encoder-decoder of siVAE, including the number of dimensions of the cell embedding space and the number of hidden layers, as well as the number of hidden nodes. Model optimization was performed by varying learning rate between 1e-2, 1e-3, and 1e-4 while the rest of the parameters were set to default.

Feature attribution methods. Two separate Python packages were used to compute neural network feature attributions in our experiments. We used the DeepExplain Python package that implemented all feature attribution methods (Saliency Maps, Grad*Int, DeepLIFT, IntGrad, Shapley Value) included in our experiments in reverse-mode[77]. We used the tensorflow-forward-ad Python package for computing Saliency Maps and Grad*Int in forward-mode[78]. In both cases, the package applies feature attribution between the target nodes and input nodes. For application of feature attributions on the decoder, the target nodes and input nodes were set to be the nodes of the output layer and latent embedding layer, respectively, for the cell-wise decoder. For application of feature attributions on the encoder, the target nodes and input nodes were set to be the nodes of the latent embedding layer and input layer, respectively, for the cell-wise encoder. By default, the DeepExplain package summarizes the attribution across all target nodes, so binary masks corresponding to a single target node were used per target node. Similarly, the tensorflow-forward-ad package summarizes attribution across all input nodes, so binary masks corresponding to a single input node were used per input node. Integrated Gradients and DeepLIFT require an additional parameter of input baseline, which represents a default null value that input values can be referenced against. We set this value to 0 equaling the mean value of gene expression after preprocessing.

For Gene Relevance, we used the published R package[45]. The method required the latent embeddings learned from siVAE as well as the raw count data corresponding to the embeddings. We also varied the number of neighborhoods (10, 100, 1000, and default).

Feature embeddings for feature attribution methods and Gene Relevance. All feature attribution methods tested here can output feature importance scores $\boldsymbol{s}_{f,c}$ that represents a vector of contributions of feature f to all cell dimension d for cell c. The Gene Relevance method[45] outputs partial derivatives in the same format. In contrast, siVAE loadings $\boldsymbol{v}_{f,1}$ represents a vector of contributions of feature f to all cell dimensions, summarized over all cells. To compare feature attribution methods to siVAE, we therefore need a procedure for converting the per-cell attributions $\boldsymbol{s}_{f,c}$ into a set of overall feature attributions $\boldsymbol{u}_f$ for each feature f with respect to all cell dimensions and that summarize across all cells, analogous to siVAE's loadings $\boldsymbol{v}_{f,1}$. To do so, we first construct a matrix $S_{d,f,c}$, containing all feature attributions for cell dimension d, cell c and feature f. For each cell dimension d, we apply PCA to the 2D matrix $S_{d,:,:}$ to extract the first principal component's loadings $\boldsymbol{u}_{:,d}$, a vector of length F that contains the contribution of each input feature f to cell dimension d. Repeating this process for each cell dimensions then concatenate the resulting vector results in matrix $\boldsymbol{U}_{f,d}$, whose rows $\boldsymbol{U}_{f,:}$ are analogous to siVAE's $\boldsymbol{v}_{f,1}$. Finally, we calculated Spearman correlation with two-sided test between the feature embeddings inferred through different approaches per dimension and reported the median values.

Datasets. A table summarizing the following datasets can be found in **Supplementary Table 2.**

Fetal liver atlas dataset processing. We obtained the fetal liver atlas[31] from ArrayExpress with accession code E-MTAB-7407 (https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-7407/) on 2020/06/10, in processed form. We normalized the count matrix to TP10K, then performed feature selection by retaining the top 2,000 highly variable genes, yielding 177,376 cells and 2,000 genes. We then downsampled the number of cells to 100,000 cells, while preserving the proportion of cells from each cell type. Genes were individually centered and scaled to unit variance. For visualization of the feature embeddings for the liver fetal atlas in **Figure 3b**, we obtained marker genes for four cell types (hepatocytes, Kupffer cells, NK/NKT cells, and MHC II positive B cells) that were available in the MSigDB database[79] (downloaded from http://www.gsea-msigdb.org/gsea/msigdb/collections.jsp#C8 on 2020/02/08). To account for the multiple subtype labels in the fetal liver dataset matching to a single cell type in marker gene set, we allowed many-to-one mapping by grouping multiple cell type labels in the dataset that corresponded to one of the cell types in marker gene database. The exact groupings are shown in **Supplementary Table**

3. We only visualized the cells with known marker genes, and genes that belonged to more than one marker gene set (shared across cell types as a marker) were discarded.

MNIST and Fashion-MNIST dataset processing. We obtained both datasets from the TensorFlow datasets web page on 2020/02/20. Images were flattened and centered and scaled to unit variance per feature across all images before input into the models.

CIFAR-10 dataset processing. We obtained CIFAR-10 from the TensorFlow datasets web page on 2020/02/20. We then subsampled the image classes to only the airplane and ship classes because other image classes require convolutional layers to achieve good classification performance, but here our goal was to benchmark VAEs. Images were flattened and centered and scaled to unit variance per feature across all images before input into the models. Color channels were concatenated and flattened.

LargeBrainAtlas dataset processing. We obtained the 1.3 Million Brain Cells dataset referred to as “LargeBrainAtlas” from the 10x Genomics website (https://support.10xgenomics.com/single-cell-gene-expression/ software/pipelines/latest/advanced/h5_matrices) on 2020/04/28. We normalized the count matrix to TP10K, then retained all genes. After, genes were individually centered and scaled to unit variance.

BrainCortex dataset processing. We obtained the BrainCortex dataset (GSE126074) from (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126074) on 2020/12/01. We performed quality control based on TSS enrichment and nucleosome signal which filtered the dataset down to 244,544 features.

Generation of simulated scRNA-seq datasets from a gene network. To explore the organization of genes in siVAE feature embedding space, we simulated scRNA-seq data where the correlations between genes is consistent with a specified gene co-expression network. We designed a gene co-expression network that consisted of five communities of 50 genes each, as well as an additional set of 50 isolated genes that are independently varying. Each community included a single hub gene that was connected to the other 49 genes in the community, in a hub-and-spoke model. No other genes in the community were connected to any other gene. All edge weights representing pairwise correlations between genes in the same community were set to 0.6. The adjacency matrix capturing the co-expression patterns between the 300 genes were converted to covariance matrix via the qpgraph R package[80], using the function qpG2Sigma with parameters rho=0.6. Afterwards, we used the resulting covariance matrix as input to a multivariate Gaussian distribution and sampled 5,000 cells for training with siVAE.

Cell type classification. The five-fold nested cross validation experiments reported in **Figure 1c** compares the performance of siVAE, VAE, and LDVAE on the fetal liver atlas dataset when matching their cell-wise encoder and decoder network designs. The number of embedding dimensions was fixed to be 2. After training using the training fold, the encoders were used to compute embeddings for the training and test datasets. We then used a k-NN ($k = 80$) classifier to predict labels of test cells based on the embeddings of the training and testing datasets. Similar five-fold nested cross validation experiments were performed on the imaging datasets (MNIST, Fashion MNIST, CIFAR-10). However, we allowed the model to individually select the number of embedding dimensions K from the set $\{2,5,10,20\}$ using the training fold. In addition, the number of clusters, k, was set to 15 as imaging datasets have far fewer classes than the fetal liver atlas dataset.

Execution time comparison experiments. We performed a series of experiments to compare siVAE training execution time against the combined execution time of VAE training and executing feature attribution methods. For Saliency Maps and Grad * Int, both forward and reverse modes were used. The majority of the feature attribution methods rely on taking the gradient of a single output nodes with respect to all input nodes using automatic differentiation in reverse-mode. For models with high number of output nodes, the operation becomes computationally infeasible. Using automatic differentiation in forward-mode allows gradient calculation of all output nodes with respect to a single input node but faces the same computational issue for models with high number of input nodes.

For the first experiment, we benchmarked using the 10x Genomics dataset. In the case of the feature attribution execution times, we extrapolated the execution time on the LargeBrainAtlas dataset from the execution time on 100,000 cells, due to time constraints and the fact that runtime of these methods should scale linearly with the number of cells. In contrast, execution times of siVAE are those measured on the full dataset. For the second experiment, we tested the effect of varying the either the number of embedding dimensions or the number of features on the execution time. As the execution time for two feature attribution methods (Integrated Gradients and Shapley Values) exceeded a realistic run time of 100 days, only the faster three methods (Saliency Maps, Grad * Int, and DeepLIFT) were used for the second experiment. For the 10x Genomics dataset, the number of embedding dimensions were set to 20, 100 and 500. Similar to the first experiment, siVAE was run on the entire set of 1.3 million cells, and the VAE+feature attribution approaches were run on 100,000 cells and then linearly interpolated to the full dataset size. For the scATAC-seq execution times, we varied the number of features by selecting the top n highly variable genomic regions, where n was set to either 28k, 120k or 240k. We used a single NVIDIA GeForce GTX1080 Ti GPU, Intel Core i5-6600K CPU, and 32 GB RAM for all experiments.

Estimating gene centrality using siVAE. We reasoned that the expression patterns of genes with high degree centrality would be most likely to be retained by siVAE during dimensionality reduction, because those genes could be used to reconstruct the expression patterns of the many other genes connected to them. If so, then the hub genes would also be likely to be the genes whose expression patterns are reconstructed with the lowest error. We therefore define gene centrality for siVAE as the negative reconstruction error of siVAE on each individual gene during training.

Estimating gene centrality using GCN inference methods. The GCN inference methods tested here all output pairwise weights between genes, where larger weights indicated higher confidence in a pairwise edge in the underlying GCN. We therefore measured each query gene's degree centrality for GCN inference methods by averaging the weights between the query gene and every other gene in the network.

Estimating the ground truth gene centrality. To compute the accuracy of siVAE-based gene centrality and GCN-based gene centrality, we generated ground truth gene centrality estimates as follows. We reasoned that a well-connected gene with high degree centrality would be highly co-expressed with many other genes in the genome, either in a linear or non-linear way. One way to quantitatively measure the degree of co-expression of a single query gene to all other genes is to measure how well the query gene can predict the expression level of all other genes in the genome. Therefore, our ground-truth gene centrality is defined as the percentage of variance explained by a query gene, with respect to all other genes in the genome. To measure percentage variance explained, for each gene in the genome, we trained a neural network consisting of a single input node (corresponding to the query gene expression), 3 hidden layers with 128, 512, and 1024 nodes, and a final output layer of 2000 nodes for all remaining genes in the genome. The percentage variance explained per gene was measured as $1 - \mathrm{Var}(X_{:,g} - \hat{X}_{:,g})/\mathrm{Var}(X_{:,g})$ where $\hat{X}_{:,g}$ is a vector of expressions for $X_{:,g}$ predicted by siVAE. We then averaged the percentage variance explained over all predicted genes, for all cells in a dataset, and refer to this quantity as the ground truth gene centrality.

Identifying gene co-expression network neighbors from siVAE models. GCN inference methods typically output a weighted adjacency matrix that indicates the strength of co-expression of every pair of genes, which in turn could be used to find the closest neighbors of every gene in the genome based on the largest values of the adjacency matrix. In our experiments in **Figure 5c**, we used siVAE to also identify the closest co-expression neighbors of every gene in the genome, using two different approaches based on leveraging the feature embedding space. In our first approach ('distance-based network neighbors'), we used Euclidean distance in the feature embedding space as a measure of distance between two genes in the network; the k nearest neighbors of a given query gene were defined as the k genes with shortest distance to the query gene. Our second approach, termed the 'GCN-based network neighbors', involved passing a matrix of feature embeddings to a GCN inference method (CLR) as input in place of the typical gene expression matrix input, in order to infer a classic gene co-expression network. From this gene co-expression network, we extracted the nearest neighbors of every gene according to the strategy described below for GCN inference methods.

Identifying gene co-expression network neighbors using GCN inference methods. For GCN inference methods, we used the output adjacency matrix to identify the closest 20 neighborhood genes per target gene based on largest pairwise weights for each gene.

Benchmarking gene neighborhoods. We computed the accuracy of both siVAE to the GCN inference methods in terms of their ability to identify neighbor genes that are co-expressed. To do so, we applied each method to identify the 20 closest neighbor genes to the query gene. We then defined a prediction task in which the 20 neighbor genes were used as input to a neural network to predict the expression of the target query gene. We used a fully connected neural network consisting of 3 hidden layers each with 16, 8, and 4 nodes, in addition to the input layer (with 20 nodes corresponding to the 20 closest neighbors), and the output layer consisting of a single node for the query gene. Accuracy was defined as the percentage variance explained with respect to the query gene. We compared siVAE to the GCN methods based on a set of 152 query genes, which were identified by taking the intersection of siVAE and each GCN inference method's top 500 highest centrality genes, to ensure that the query genes were of high degree centrality (and therefore should have many neighbor genes).

Quantifying overlap in gene neighborhoods between siVAE and other methods. We also used two different strategies to gauge the overlap in the gene neighborhoods predicted by siVAE and each GCN inference method, defined as the 20 closest genes to every query gene. For percentage overlap, we measured the percentage of genes that overlapped between any two sets of neighborhood genes. For mean correlation, we measured the Pearson correlations with two-sided test of gene expression between every pair of genes between two neighborhood gene sets for the same query gene, then averaged the 20*20 = 400 correlation values together to compute mean correlation.

Model	**Interpretability Reg.**	**Decoder**	**Observation model**
siVAE	Yes	Non-Linear	Gaussian
siVAE ($\gamma = 0$)	No	Non-linear	Gaussian
siVAE-linear	No	Linear	Gaussian
siVAE-NB	Yes	Non-linear	Negative Binomial
VAE	NA	Non-linear	Gaussian
scVI	NA	Non-linear	Negative Binomial
LDVAE	NA	Linear, single	Negative Binomial
VAE (linear)	NA	Linear, single	Negative Binomial

Table 1. List of model variations with corresponding features. Usage of the interpretability term only applies to siVAE and its variants. A linear decoder is composed of the same number of layers as the non-linear decoder unless specified as single, in which case the latent embedding layer is directly transformed to an output layer.

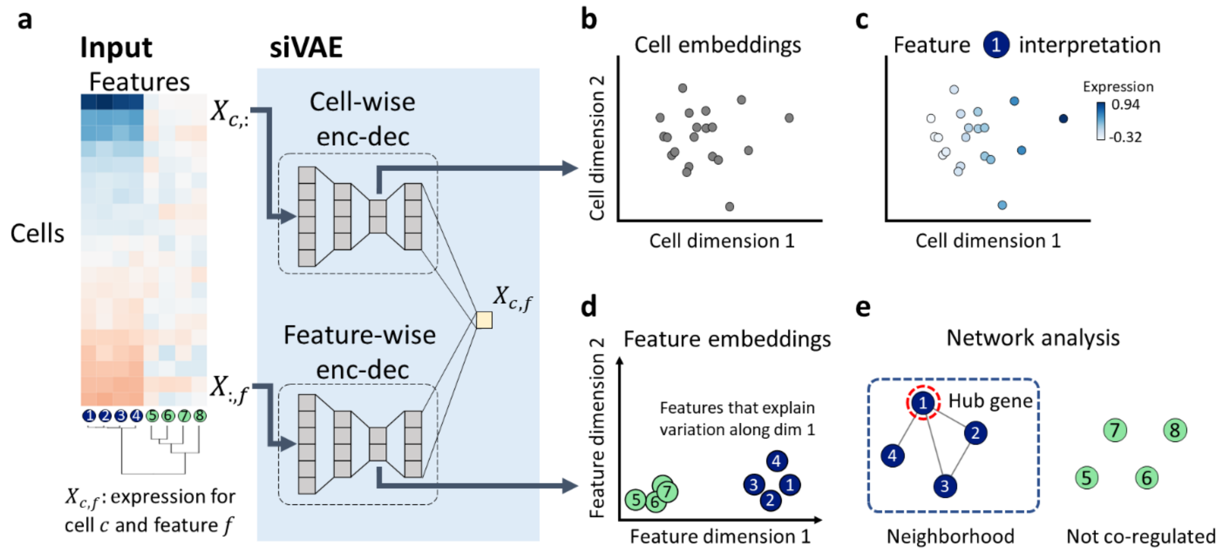


Figure 1: The siVAE model for inferring interpretable representations of single cell genomic data. (**a**) The input to siVAE is a cell by feature matrix; shown here is a synthetic gene expression matrix of eight genes, four of which are tightly regulated (genes 1,2,3, 4), and the other four of which vary independently (5,6,7, 8). siVAE is a neural network consisting of a pair of encoder-decoders, that jointly learn a cell embedding space and feature embedding space. The "cell-wise encoder-decoder" acts like a traditional VAE, where the input to the encoder is a single cell c's measurement across all input features ($X_{c,:}$). The cell-wise encoder uses the input cell measurements to compute an approximate posterior distribution over the location of the cell in the cell embedding space. The "feature-wise encoder-decoder" takes as input measurements for a single feature f across all input training cells ($X_{:,f}$). The feature-wise encoder uses the input feature measurements to compute an approximate posterior distribution over the location of the feature in the feature embedding space. The decoders of the cell-wise and feature-wise encoder-decoders combine to output the expression level of feature f in cell c ($X_{c,f}$). (**b**) Visualization of the cell and feature embedding space learned from the gene expression matrix in (a). Note the embeddings of genes 1, 2, 3 and 4 all have large magnitudes along dimension 1 but not dimension 2, suggesting genes 1, 2, 3 and 4 explain variation in the cell embedding space along dimension 1. Genes 5, 6, 7, and 8 sit at the origin of the feature embedding space, suggesting they do not co-vary with other features. (**c**) The expression patterns of gene 1 are overlaid on the cells in the cell embedding space. Gene 1 clearly increases in expression when inspecting cells from left to right, consistent with the feature embedding space that shows Gene 1 having large loadings on dimension 1. (**d**) In contrast, Gene 5's expression does not have a clear pattern of variation with respect to position of the cell in cell embedding space, consistent with Gene 5's location close to the origin in the feature embedding space. (**e**) A trained siVAE model can be used to identify hubs and gene neighbors in a gene co-expression network, without the need to explicitly infer a co-expression network.

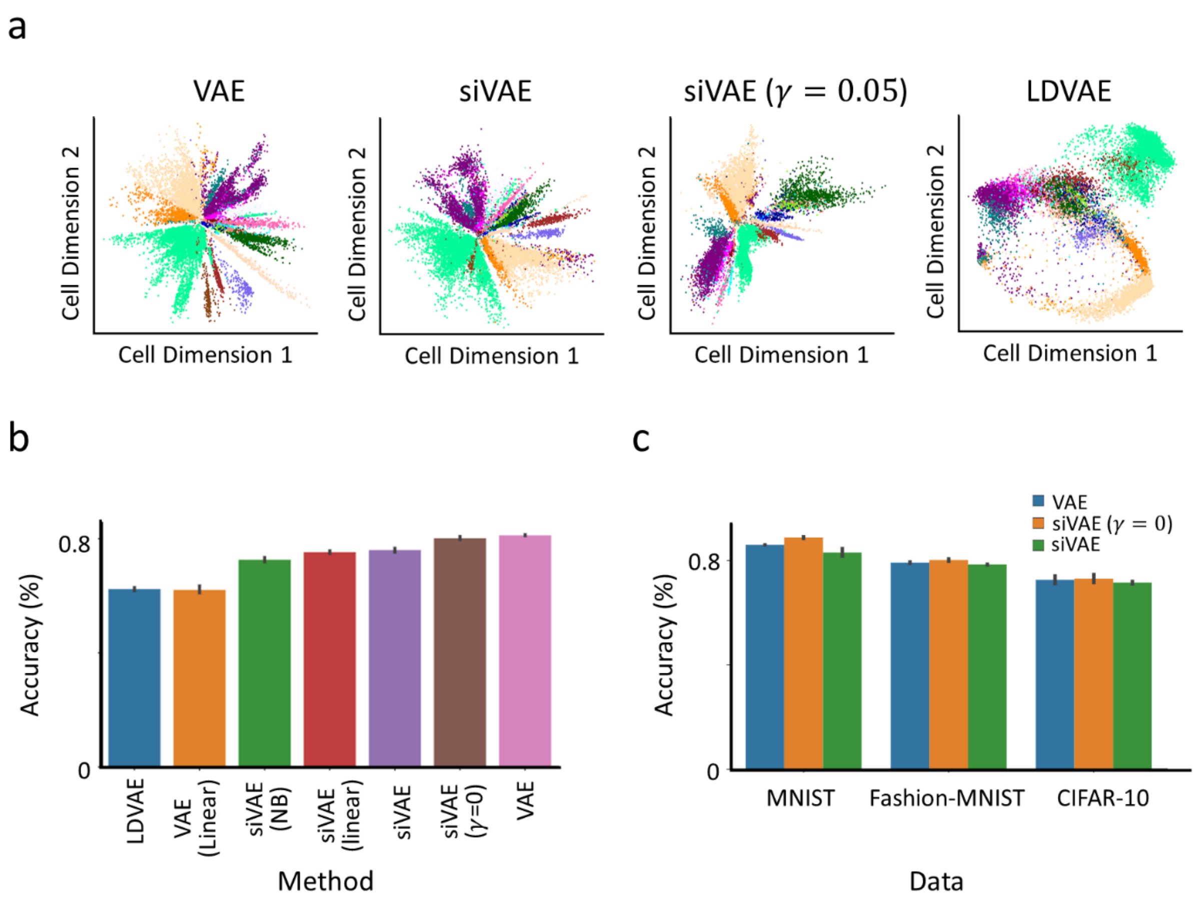


Figure 2: Accuracy evaluation of siVAE's cell embedding spaces. (**a**) 2D visualization of the inferred cell embedding spaces of a classic VAE, siVAE, a variant of siVAE in which the interpretability regularization term is removed ($\gamma = 0$) and LDVAE. Each point represents a cell and is colored according to cell type. (**b)** Barplot indicating the accuracy of a k-NN ($k = 80$) classifier predicting the cell type labels of single cells based on their inferred position in the cell embedding space inferred by siVAE and other methods trained on the Fetal Liver Atlas dataset. (**c**) Same as (a), but classification was performed on three imaging datasets (MNIST, Fashion-MNIST and CIFAR-10).

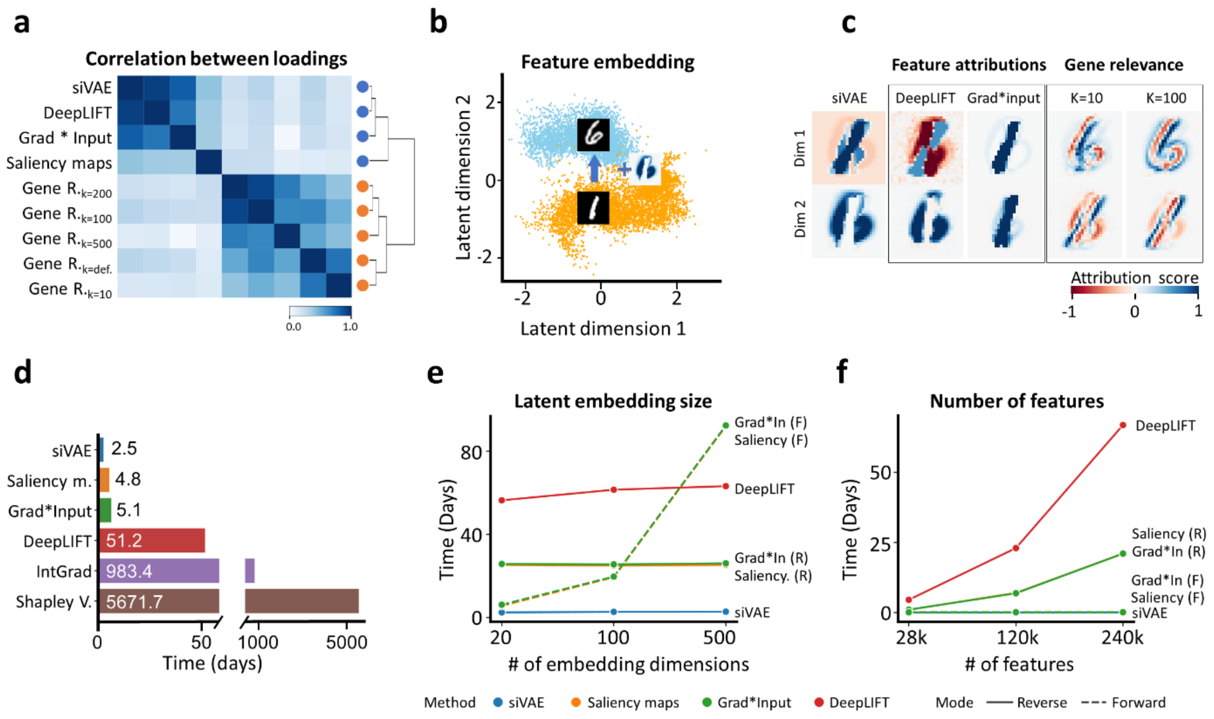


Figure 3: siVAE yields accurate and fast interpretations. (**a**) Heatmap indicating the mean pairwise correlation between the interpretations (loadings) of siVAE, Gene Relevance as well as three neural net feature attribution methods (saliency maps, grad*input, and DeepLIFT), where correlations have been averaged over each of the 2 embedding dimensions for the fetal liver atlas dataset. (**b**) Scatterplot of feature embeddings inferred by siVAE trained on the one and six digits from the MNIST dataset. (**c**) Visualization of interpretations learned for each of the two dimensions (axes) from (b), for siVAE, Gene Relevance, DeepLIFT and grad*input. (**d**) Barplot indicating the time required to train siVAE versus training a classic VAE and applying feature attribution methods on the LargeBrainAtlas dataset. (**e**) Line plot indicating the time required to train siVAE and feature attribution methods on the LargeBrainAtlas dataset when the number of embedding dimensions for siVAE is varied, and the number of features is fixed at 28k. (**f**) Line plot indicating the time required to train siVAE and feature attribution methods on the scATAC-Seq dataset when the number of features is being varied, while the number of embedding dimensions is fixed at 20.

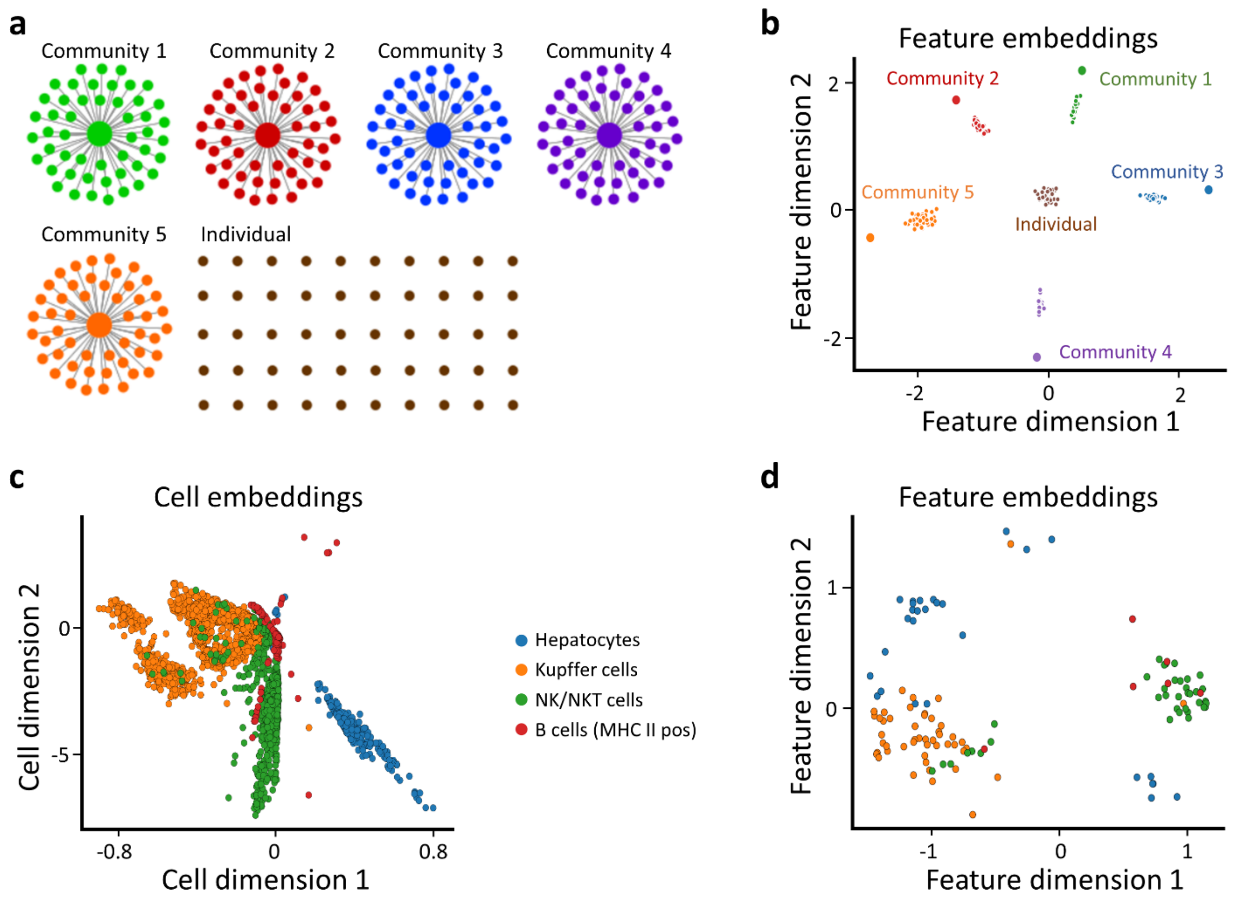


Figure 4: Co-expressed genes tend to co-localize in the siVAE feature embedding space. (**a**) The gene co-expression network used to simulate single cell expression data. The network consists of five tightly correlated groups of 50 genes each, along with 50 isolated genes. (**b**) Scatterplot of the feature embeddings produced from siVAE trained on the dataset simulated from the network in (a). Each point represents a gene, colored and labeled by the community it belongs to in (a). (**c**) Scatterplot of the cell embedding space inferred by siVAE trained on the fetal liver atlas dataset. Each point represents a cell and is colored based on its pre-defined cell type. (**d**) Scatterplot of feature embeddings inferred by siVAE trained on fetal liver atlas dataset. Each point represents a marker gene and is colored based on its prior known association to a cell type.

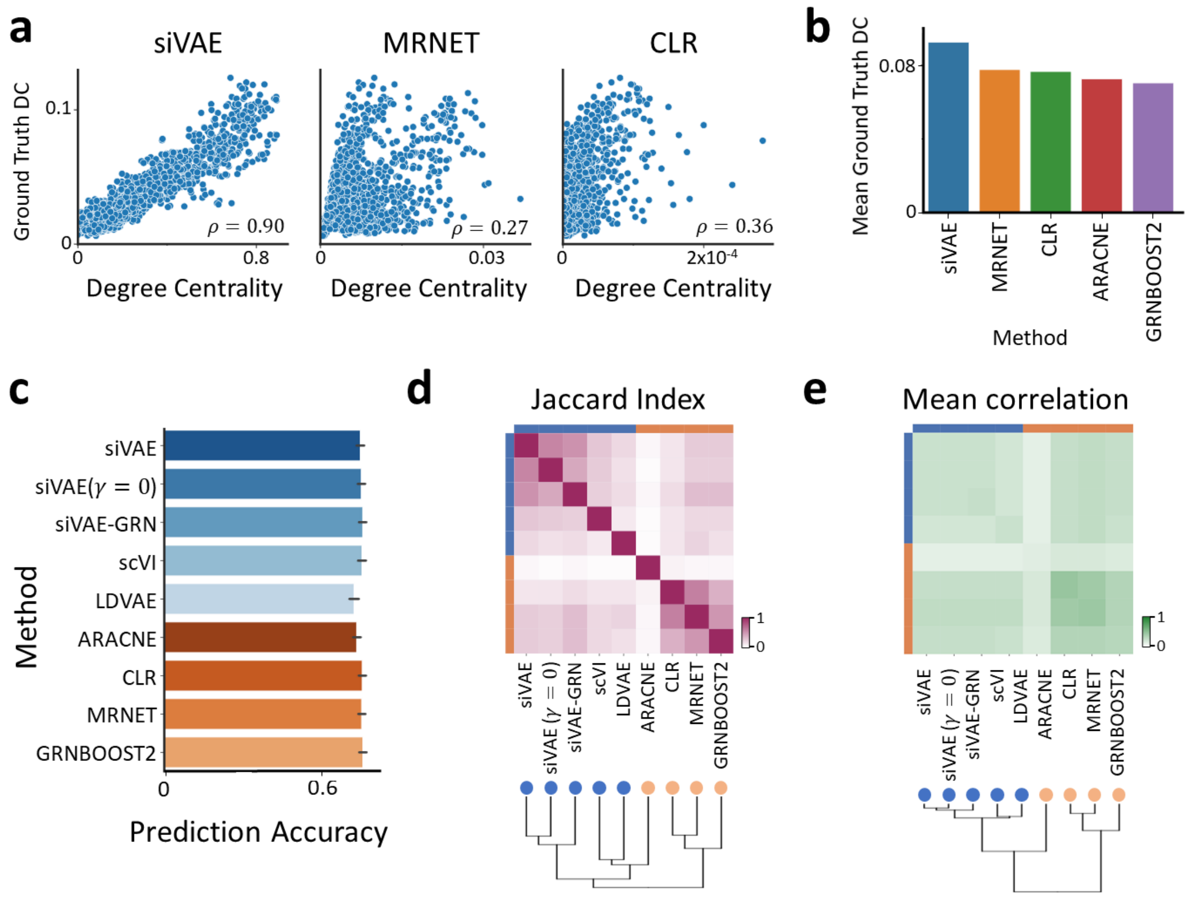


Figure 5: siVAE can be used to gain insight into gene co-expression network structure without explicit network inference. (**a**) Scatterplot showing the correlation between ground truth degree centrality and predicted degree centrality, based on using siVAE training performance, or by computing node degree when a network is inferred using the MRNET or CLR algorithms. (**b**) Average true degree centrality of the top 50 genes predicted to have highest degree centrality across different methods. (**c**) Bar plot indicating the prediction accuracy (% variance explained) of the neighborhood gene sets when predicting each query gene, averaged over the top 152 query genes with highest predicted degree centrality across all tested methods in the fetal liver atlas dataset. Blue bars denote methods based on dimensionality reduction, while orange bars indicate methods based on explicit gene regulatory network inference. (**d**) Heatmap indicating the pairwise Jaccard index (overlap) between neighborhood genes identified by pairs of methods. (**e**) Heatmap indicating the mean pairwise correlation in expression between pairs of methods with respect to their neighborhood genes. Each box indicates average of the pairwise correlation matrix where the columns and rows correspond to neighborhood genes identified by two methods.

1. Wang, J. *et al.* Single-Cell Co-expression Analysis Reveals Distinct Functional Modules, Co-regulation Mechanisms and Clinical Outcomes. *PLOS Comput. Biol.* **12**, e1004892 (2016).
2. Chepelev, I., Wei, G., Wangsa, D., Tang, Q. & Zhao, K. Characterization of genome-wide enhancer-promoter interactions reveals co-expression of interacting genes and modes of higher order chromatin organization. *Cell Res.* **22**, 490–503 (2012).
3. Cakir, B. *et al.* Comparison of visualization tools for single-cell RNAseq data. *NAR Genomics Bioinforma.* **2**, lqaa052 (2020).
4. Andrews, T. S. & Hemberg, M. Identifying cell populations with scRNASeq. *Mol. Aspects Med.* **59**, 114–122 (2018).
5. Saelens, W., Cannoodt, R., Todorov, H. & Saeys, Y. A comparison of single-cell trajectory inference methods. *Nat. Biotechnol.* **37**, 547–554 (2019).
6. Cannoodt, R., Saelens, W. & Saeys, Y. Computational methods for trajectory inference from single-cell transcriptomics. *Eur. J. Immunol.* **46**, 2496–2506 (2016).
7. Jiang, L., Chen, H., Pinello, L. & Yuan, G.-C. GiniClust: detecting rare cell types from single-cell gene expression data with Gini index. *Genome Biol.* **17**, 144 (2016).
8. Bergen, V., Lange, M., Peidli, S., Wolf, F. A. & Theis, F. J. Generalizing RNA velocity to transient cell states through dynamical modeling. *Nat. Biotechnol.* **38**, 1408–1414 (2020).
9. McFarland, J. M. *et al.* Multiplexed single-cell transcriptional response profiling to define cancer vulnerabilities and therapeutic mechanism of action. *Nat. Commun.* **11**, 4296 (2020).
10. Peng, M. *et al.* Single-cell transcriptomic landscape reveals the differences in cell differentiation and immune microenvironment of papillary thyroid carcinoma between genders. *Cell Biosci.* **11**, 39 (2021).
11. Sun, S., Zhu, J., Ma, Y. & Zhou, X. Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. *Genome Biol.* **20**, 269 (2019).

12. V.S, S. & Surendran, S. A Review of Various Linear and Non Linear Dimensionality Reduction Techniques. in (2015).

13. Stuart, T. *et al.* Comprehensive Integration of Single-Cell Data. *Cell* **177**, 1888-1902.e21 (2019).

14. Chen, H. *et al.* Single-cell trajectories reconstruction, exploration and mapping of omics data with STREAM. *Nat. Commun.* **10**, 1903 (2019).

15. Nguyen, H., Tran, D., Tran, B., Pehlivan, B. & Nguyen, T. A comprehensive survey of regulatory network inference methods using single cell RNA sequencing data. *Brief. Bioinform.* **22**, bbaa190 (2021).

16. Hu, R., Pei, G., Jia, P. & Zhao, Z. Decoding regulatory structures and features from epigenomics profiles: A Roadmap-ENCODE Variational Auto-Encoder (RE-VAE) model. *Methods* **189**, 44–53 (2021).

17. Choi, J. & Chae, H. methCancer-gen: a DNA methylome dataset generator for user-specified cancer type based on conditional variational autoencoder. *BMC Bioinformatics* **21**, 181 (2020).

18. Zuo, C. & Chen, L. Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data. *Brief. Bioinform.* **22**, bbaa287 (2021).

19. Zhang, L., Chen, X. & Yin, J. Prediction of Potential miRNA–Disease Associations Through a Novel Unsupervised Deep Learning Framework with Variational Autoencoder. *Cells* **8**, 1040 (2019).

20. Wang, D. & Gu, J. VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder. *Genomics Proteomics Bioinformatics* **16**, 320–331 (2018).

21. Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nat. Methods* **15**, 1053–1058 (2018).

22. Rashid, S., Shah, S., Bar-Joseph, Z. & Pandya, R. Dhaka: variational autoencoder for unmasking tumor heterogeneity from single cell genomic data. *Bioinformatics* **37**, 1535–1543 (2021).

23. Qiu, Y. L., Zheng, H. & Gevaert, O. Genomic data imputation with variational auto-encoders. *GigaScience* **9**, giaa082 (2020).

24. Jarada, T. N., Rokne, J. G. & Alhajj, R. SNF–CVAE: Computational method to predict drug–disease interactions using similarity network fusion and collective variational autoencoder. *Knowl.-Based Syst.* **212**, 106585 (2021).

25. Rampášek, L., Hidru, D., Smirnov, P., Haibe-Kains, B. & Goldenberg, A. Dr.VAE: improving drug response prediction via modeling of drug perturbation effects. *Bioinformatics* **35**, 3743–3751 (2019).

26. Lotfollahi, M., Wolf, F. A. & Theis, F. J. scGen predicts single-cell perturbation responses. *Nat. Methods* **16**, 715–721 (2019).

27. Kusner, M. J., Paige, B. & Hernández-Lobato, J. M. Grammar Variational Autoencoder. in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* 1945–1954 (JMLR.org, 2017).

28. Kinalis, S., Nielsen, F. C., Winther, O. & Bagger, F. O. Deconvolution of autoencoders to learn biological regulatory modules from single cell mRNA sequencing data. *BMC Bioinformatics* **20**, 379 (2019).

29. Svensson, V., Gayoso, A., Yosef, N. & Pachter, L. Interpretable factor models of single-cell RNA-seq via variational autoencoders. *Bioinformatics* **36**, 3418–3421 (2020).

30. Buettner, F., Pratanwanich, N., McCarthy, D. J., Marioni, J. C. & Stegle, O. f-scLVM: scalable and versatile factor analysis for single-cell RNA-seq. *Genome Biol.* **18**, 212 (2017).

31. Popescu, D.-M. *et al.* Decoding human fetal liver haematopoiesis. *Nature* **574**, 365–371 (2019).

32. Zhao, Y., Cai, H., Zhang, Z., Tang, J. & Li, Y. Learning interpretable cellular and gene signature embeddings from single-cell transcriptomic data. *Nat. Commun.* **12**, 5261 (2021).

33. Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S. & Theis, F. J. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.* **10**, 390 (2019).

34. Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S. & Vert, J.-P. A general and flexible method for signal extraction from single-cell RNA-seq data. *Nat. Commun.* **9**, (2018).

35. Janizek, J. D., Sturmfels, P. & Lee, S.-I. Explaining Explanations: Axiomatic Feature Interactions for Deep Networks. *ArXiv200204138 Cs Stat* (2020).

36. Shrikumar, A., Greenside, P. & Kundaje, A. Learning Important Features Through Propagating Activation Differences. *ArXiv170402685 Cs* (2017).

37. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *ArXiv13126034 Cs* (2013).

38. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic Attribution for Deep Networks. in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* 3319–3328 (JMLR.org, 2017).

39. Lundberg, S. M. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. in *Advances in Neural Information Processing Systems 30* (eds. Guyon, I. et al.) 4765–4774 (Curran Associates, Inc., 2017).

40. Tsang, M. *et al.* Feature Interaction Interpretability: A Case for Explaining Ad-Recommendation Systems via Neural Interaction Detection. *ArXiv200610966 Cs Stat* (2020).

41. Ribeiro, M. T., Singh, S. & Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1135–1144 (Association for Computing Machinery, 2016). doi:10.1145/2939672.2939778.

42. Smilkov, D., Thorat, N., Kim, B., Viégas, F. & Wattenberg, M. SmoothGrad: removing noise by adding noise. *ArXiv170603825 Cs Stat* (2017).

43. Selvaraju, R. R. *et al.* Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *ArXiv161002391 Cs* (2016).

44. Fong, R. & Vedaldi, A. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *2017 IEEE Int. Conf. Comput. Vis. ICCV* 3449–3457 (2017) doi:10.1109/ICCV.2017.371.

45. Angerer, P., Fischer, D. S., Theis, F. J., Scialdone, A. & Marr, C. Automatic identification of relevant genes from low-dimensional embeddings of single-cell RNA-seq data. *Bioinformatics* **36**, 4291–4295 (2020).

46. Adebayo, J. *et al.* Sanity Checks for Saliency Maps. in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* 9525–9536 (Curran Associates Inc., 2018).

47. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

48. Dwivedi, S. K., Tjärnberg, A., Tegnér, J. & Gustafsson, M. Deriving disease modules from the compressed transcriptional space embedded in a deep autoencoder. *Nat. Commun.* **11**, 856 (2020).

49. 10X Genomics. 1.3 million brain cells from E18 mice. https://www.10xgenomics.com/blog/our-13-million-single-cell-dataset-is-ready-to-download (2017).

50. 10X Genomics. Single Cell ATAC Datasets. https://support.10xgenomics.com/single-cell-atac/datasets.

51. Goymer, P. Why do we need hubs? *Nat. Rev. Genet.* **9**, 651–651 (2008).

52. Liu, Y. *et al.* Identification of Hub Genes and Key Pathways Associated With Bipolar Disorder Based on Weighted Gene Co-expression Network Analysis. *Front. Physiol.* **10**, 1081 (2019).

53. Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A. & Murali, T. M. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nat. Methods* **17**, 147–154 (2020).

54. Barbosa, S., Niebel, B., Wolf, S., Mauch, K. & Takors, R. A guide to gene regulatory network inference for obtaining predictive solutions: Underlying assumptions and fundamental biological and data constraints. *Biosystems* **174**, 37–48 (2018).

55. Banf, M. & Rhee, S. Y. Enhancing gene regulatory network inference through data integration with markov random fields. *Sci. Rep.* **7**, 41174 (2017).

56. Tipping, M. E. & Bishop, C. M. Probabilistic Principal Component Analysis. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **61**, 611–622 (1999).

57. Langfelder, P. & Horvath, S. Eigengene networks for studying the relationships between co-expression modules. *BMC Syst. Biol.* **1**, 54 (2007).

58. Foroushani, A. *et al.* Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications. *BMC Med. Genomics* **10**, 16 (2017).

59. Liberzon, A. *et al.* Molecular signatures database (MSigDB) 3.0. *Bioinformatics* **27**, 1739–1740 (2011).

60. Koschützki, D. & Schreiber, F. Centrality Analysis Methods for Biological Networks and Their Application to Gene Regulatory Networks. *Gene Regul. Syst. Biol.* **2**, GRSB.S702 (2008).

61. Nair, J., Ghatge, M., Kakkar, V. V. & Shanker, J. Network Analysis of Inflammatory Genes and Their Transcriptional Regulators in Coronary Artery Disease. *PLoS ONE* **9**, e94328 (2014).

62. van Dam, S., Võsa, U., van der Graaf, A., Franke, L. & de Magalhães, J. P. Gene co-expression analysis for functional classification and gene–disease predictions. *Brief. Bioinform.* bbw139 (2017) doi:10.1093/bib/bbw139.

63. Guo, Y., Alexander, K., Clark, A. G., Grimson, A. & Yu, H. Integrated network analysis reveals distinct regulatory roles of transcription factors and microRNAs. *RNA* **22**, 1663–1672 (2016).

64. Liesecke, F. *et al.* Improved gene co-expression network quality through expression dataset down-sampling and network aggregation. *Sci. Rep.* **9**, 14431 (2019).

65. Wetterstrand, K. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP).

66. Chen, S., Lake, B. B. & Zhang, K. High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. *Nat. Biotechnol.* **37**, 1452–1457 (2019).

67. Mimitou, E. P. *et al.* Multiplexed detection of proteins, transcriptomes, clonotypes and CRISPR perturbations in single cells. *Nat. Methods* **16**, 409–412 (2019).

68. Ma, S. *et al.* Chromatin potential identified by shared single cell profiling of RNA and chromatin. *bioRxiv* 2020.06.17.156943 (2020) doi:10.1101/2020.06.17.156943.

69. Larsen, A. B. L., Sønderby, S. K., Larochelle, H. & Winther, O. Autoencoding beyond pixels using a learned similarity metric. *ArXiv151209300 Cs Stat* (2016).

70. Burgess, C. P. *et al.* Understanding disentangling in β-VAE. *ArXiv180403599 Cs Stat* (2018).

71. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *ArXiv13126114 Cs Stat* (2014).

72. Chen, X. *et al.* InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. in *Proceedings of the 30th International Conference on Neural Information Processing Systems* 2180–2188 (Curran Associates Inc., 2016).

73. Joo, W., Lee, W., Park, S. & Moon, I.-C. Dirichlet Variational Autoencoder. *ArXiv190102739 Cs Stat* (2019).

74. Kim, M., Wang, Y., Sahu, P. & Pavlovic, V. Bayes-Factor-VAE: Hierarchical Bayesian Deep Auto-Encoder Models for Factor Disentanglement. *ArXiv190902820 Cs Stat* (2019).

75. Chen, R. T. Q., Li, X., Grosse, R. & Duvenaud, D. Isolating Sources of Disentanglement in VAEs. in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* 2615–2625 (Curran Associates Inc., 2018).

76. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).

77. Ancona, M., Ceolini, E., Öztireli, C. & Gross, M. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. *ArXiv171106104 Cs Stat* (2018).

78. Ren, M. tensorflow-forward-ad. *Github*.

79. Subramanian, A. *et al.* Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. U. S. A.* **102**, 15545–15550 (2005).

80. Castelo, R. & Roverato, A. Reverse Engineering Molecular Regulatory Networks from Microarray Data with qp-Graphs. *J. Comput. Biol.* **16**, 213–227 (2009).

Supplementary materials for interpretable deep generative models for genomics

Yongin Choi[1,2,*], Gerald Quon[1,2,3,*]

[1]Biomedical Engineering Graduate Group, [2]Genome Center, [3]Department of Molecular and Cellular Biology, University of California, Davis, Davis, CA

*To whom correspondence should be addressed: gquon@ucdavis.edu

Table of Contents

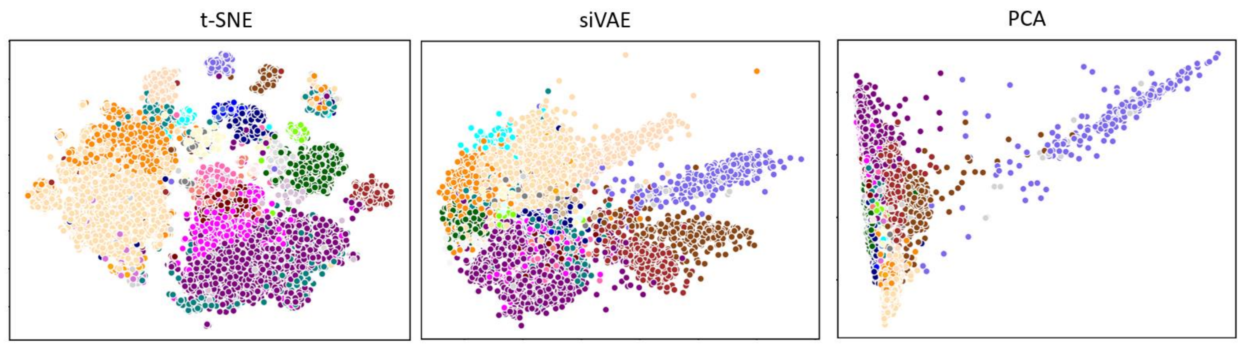


Fig. S1: Non-linear dimensionality reduction methods generate cell embedding spaces in which cells of the same cell type cluster more tightly. Scatterplots shows embedding spaces learned using t-SNE, siVAE, and PCA trained on fetal liver atlas datasets. Cells are colored based on cell type.

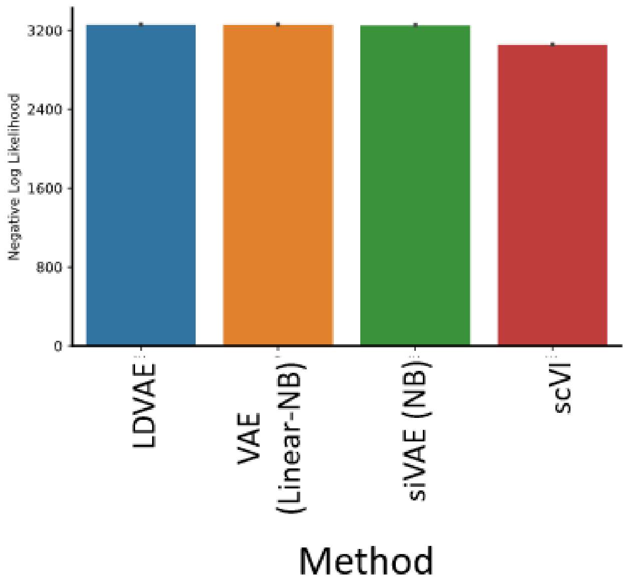


Fig. S2: Negative log likelihoods achieved by different methods on the fetal liver atlas dataset. Bar plot indicates the negative log likelihood (nll) for different models that use a negative binomial distribution as the output layer of the neural network.

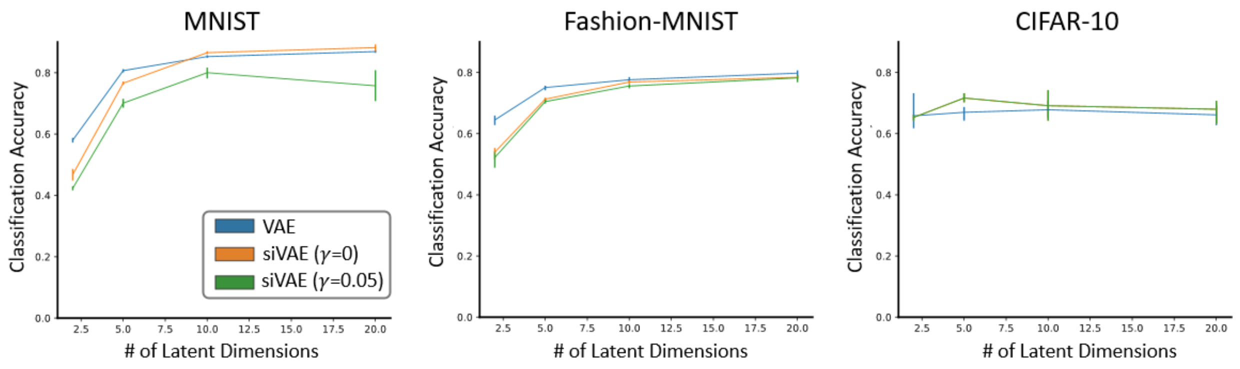


Fig. S3: Classification experiments on three imaging datasets. Line plots indicate classification accuracy of methods while varying number of embedding dimensions trained on three imaging datasets. Classification was performed on the embeddings of each model using k-nearest neighbors in a 5-fold cross validation framework.

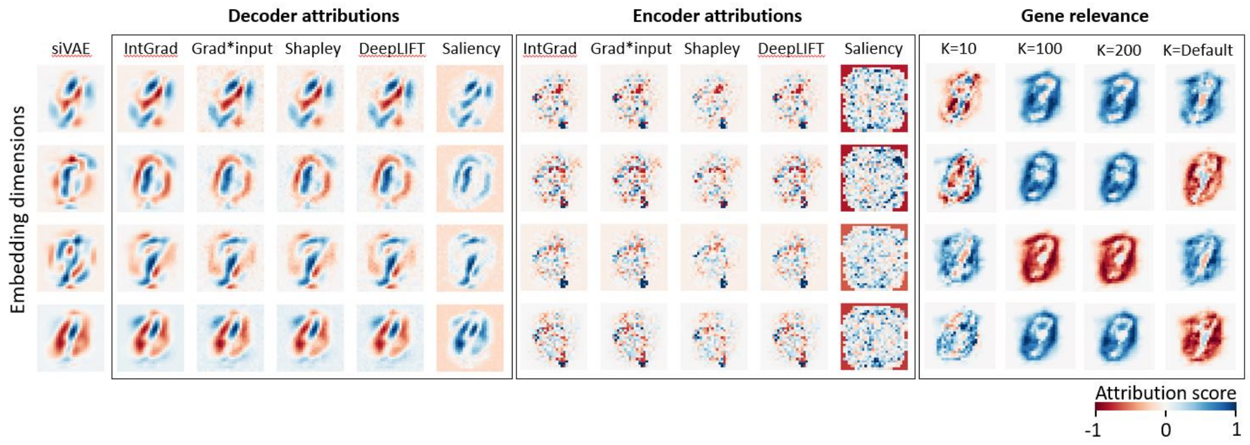


Fig. S4: Visualization of feature attributions across methods. Figure shows the visualization of feature attributions (embeddings for siVAE) for different methods when trained on the MNIST dataset with all digits. Individual image represents the feature attributions (or embeddings for siVAE) for one embedding dimension. Attribution score represents the contribution of individual feature (pixel) to each embedding dimension. Feature attribution methods and gene relevance scores were computed on the trained siVAE model to make them comparable.

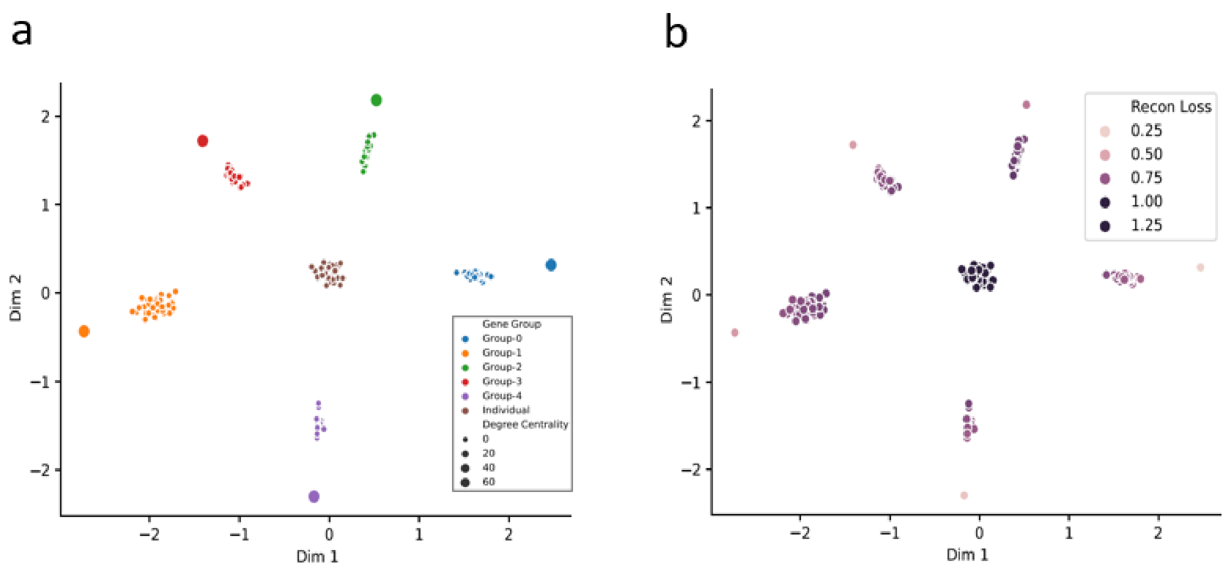


Fig. S5: Organization of the siVAE feature embedding space when trained on a simulated dataset. Scatter plots show the feature embeddings of siVAE when trained on a dataset simulated from a hypothetical genome containing 300 genes, and in which the underlying gene network consists of five communities of co-regulated genes, and one group of disconnected nodes. (**a**) Nodes are colored based on which community they originate from. (**b**) Nodes are colored based on their reconstruction loss per gene after training.

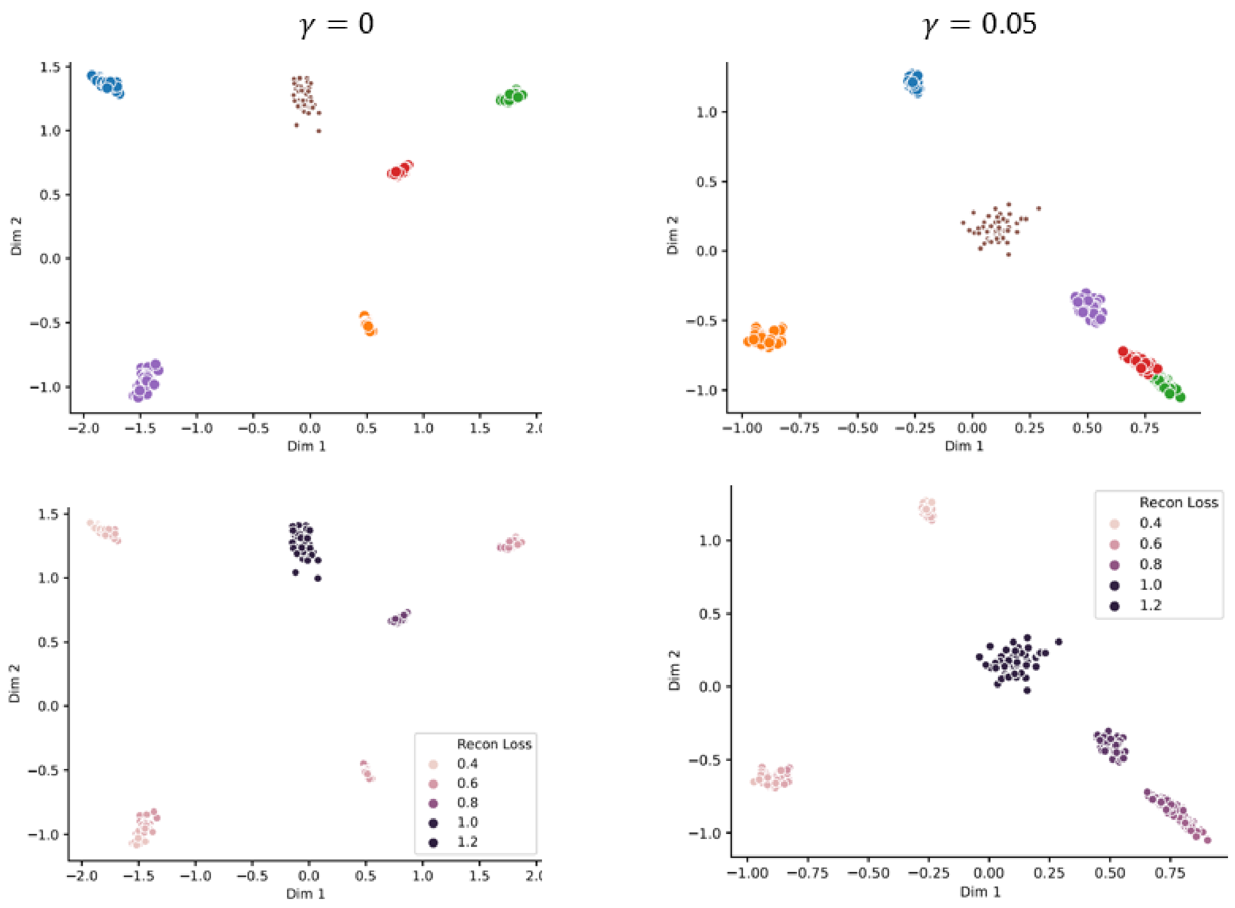


Fig. S6: Organization of the siVAE feature embedding space when trained on a simulated dataset. Scatter plots show the feature embeddings of siVAE when trained on a dataset simulated from a hypothetical genome containing 300 genes, and in which the underlying gene network consists of five communities of co-regulated genes, and one group of disconnected nodes. Scatterplots show the feature embeddings of siVAE with ($\gamma = 0.05$) and without ($\gamma = 0$) the interpretability term. Top row: nodes are colored based on which community they belong to. Bottom: nodes are colored by reconstruction error after training.

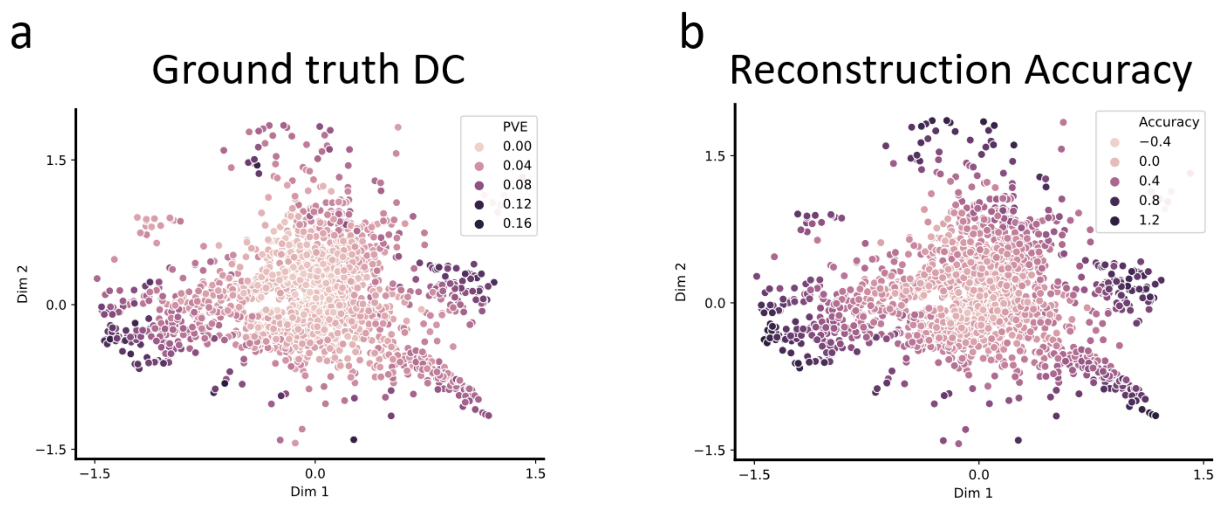


Fig. S7: Prediction of degree centrality of features based on their siVAE reconstruction accuracy. Scatter plot shows feature embeddings of siVAE trained on fetal liver dataset where each point represens a single input feature. **(a)** Scatterplot is overlaid with ground truth degree centrality of each gene, where the ground truth is computed as the percent variance explained (PVE) when a query gene was used to predict the rest of genes in the transcriptome. **(b)** Features are colored based on how well reconstructed they were after training by siVAE. Note the close concordance with (a).

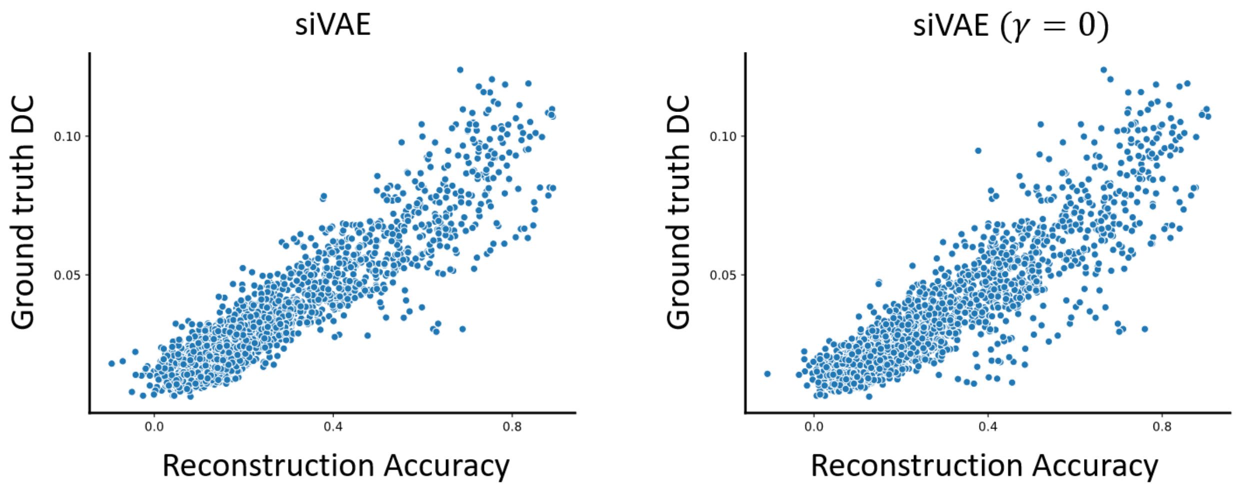
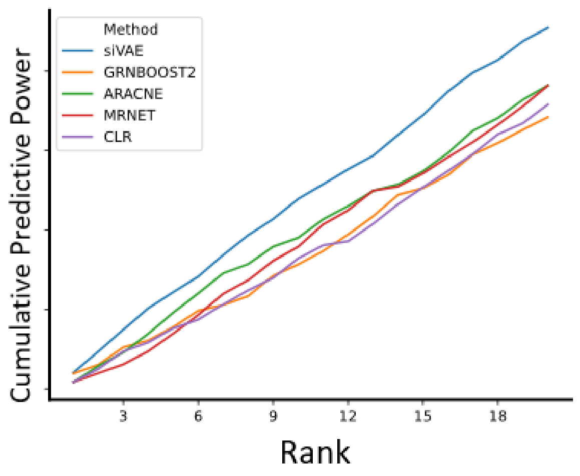


Fig. S8: High correlation between ground truth degree centrality (DC) and siVAE-estimated reconstruction accuracy. Scatter plot shows the correlation between predicted degree centrality (measured as reconstruction accuracy) and ground truth degree centrality for both siVAE model with and without interpretability term. Based on the strong correlation with ground truth degree centrality in both cases, we use reconstruction accuracy to predict degree centrality for siVAE.



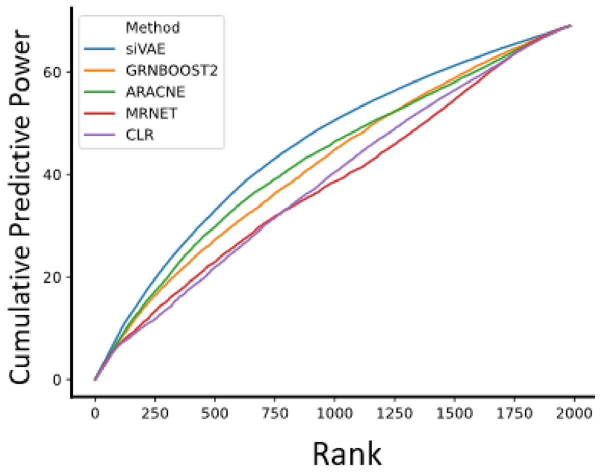


Fig. S9: siVAE predicts genes with higher degree centrality compared to other methods. Line plot indicates the cumulative ground truth degree centrality of the top 20 (a) and 2000 (b) genes ranked in decreasing order of largest predicted degree centrality by each method.

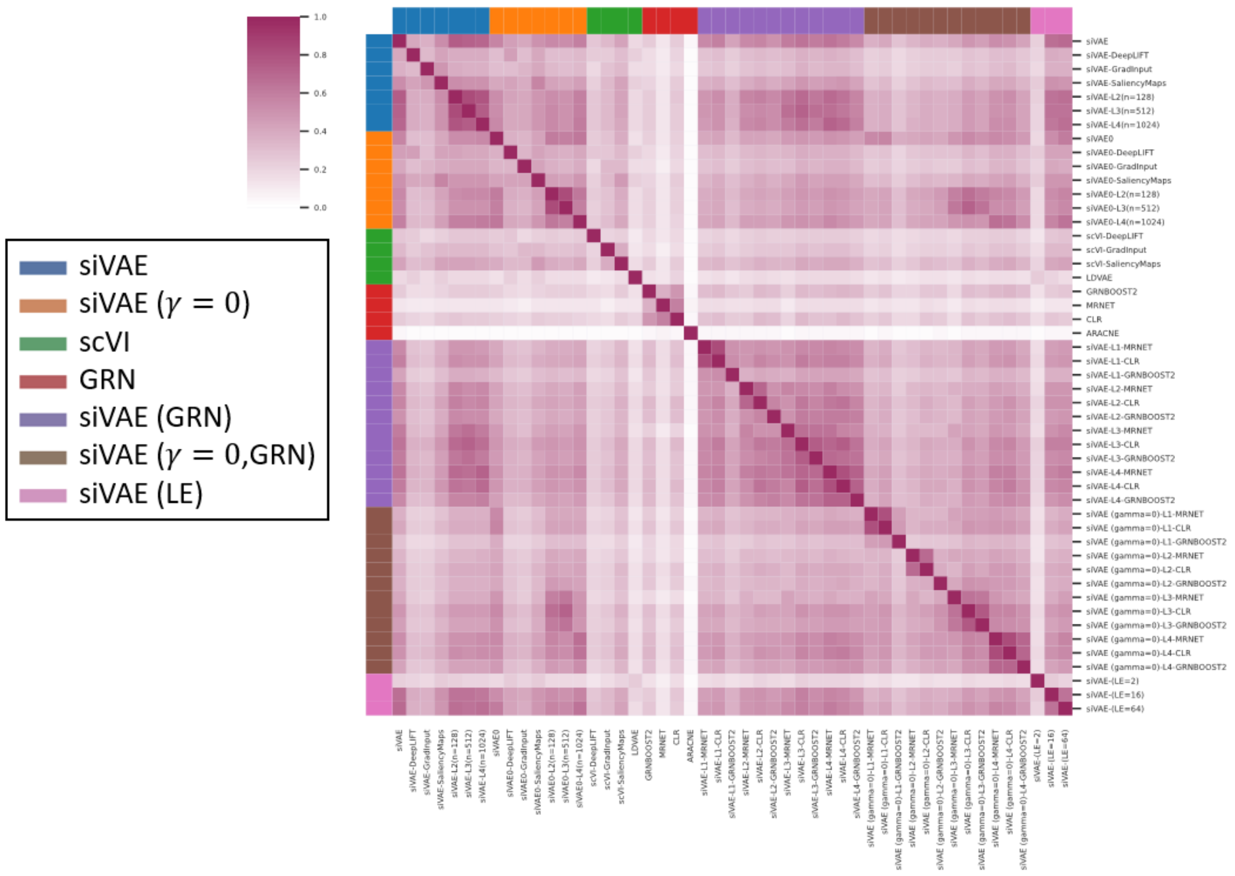


Fig. S10: Overlap between neighborhood genes identified by different methods. Heatmap indicates the Jaccard index quantifying the overlap between the neighbor genes detected by each method.

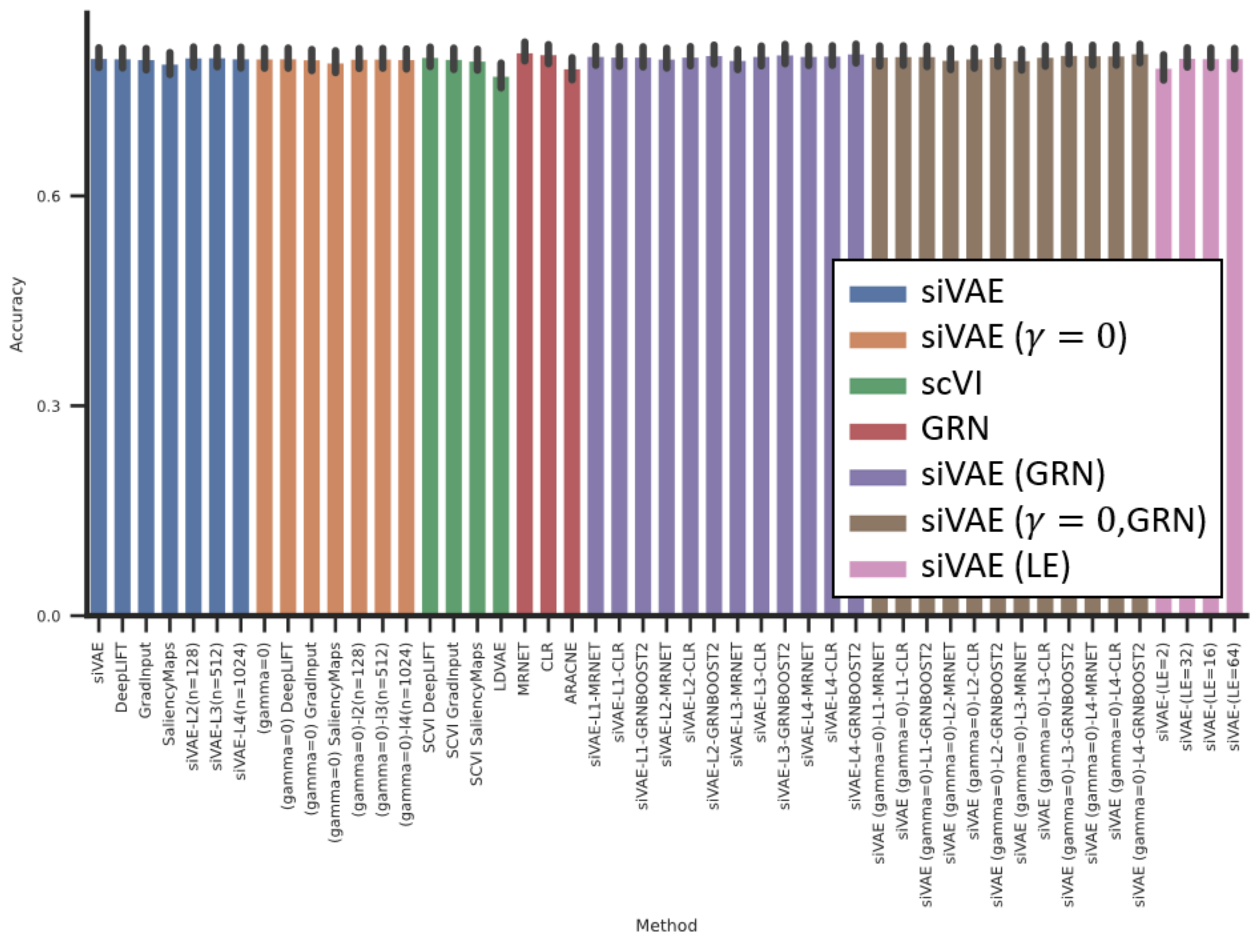


Fig. S11: Both dimensionality reduction based approaches and explicit GCN inference based approaches predict neighborhood genes that equally explain the expression of query genes. Bar plot indicates the prediction accuracy (% variance explained) of the neighborhood gene sets when predicting each query gene, averaged over the top 152 query genes with highest predicted degree centrality across tested methods. Blue bars denote methods based on siVAE, and orange bars denote methods based on siVAE without interpretability term. Green bar denotes methods based on applying feature attribution to scVI. Red bars indicate GCN inference based methods. Purple and brown bars denote approaches where GRN inference was used on data that was passed through siVAE with and without interpretability term respectively. Finally, pink bar denotes method based on siVAE with varying numbers of embedding dimensions.

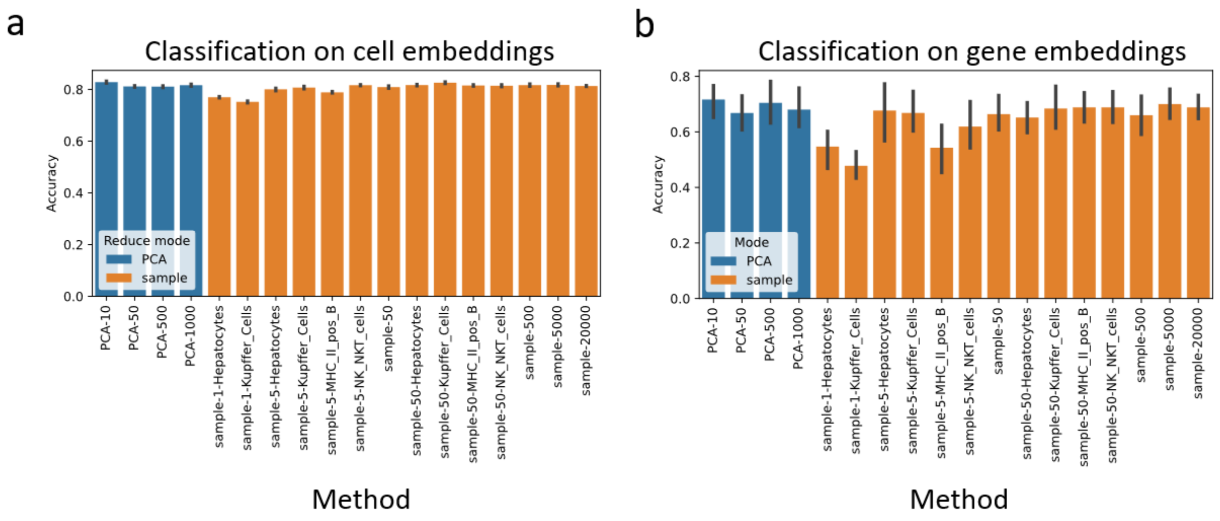


Fig. S12: Training with reducing dimensionality of input for feature-wise encoder-decoder yields comparable clustering accuracy. Bar plots indicate classification accuracy of methods while training with different processing of the input for feature-wise encoder decoder on the Fetal Liver Atlas dataset. Classification was performed on both cell and gene embeddings of each model using k-nearest neighbors in a 5-fold cross validation framework. Blue bars indicate accuracy from using PCA to reduce number of input features whereas orange bars indicate accuracy from using downsampling to reduce number of input features.

Dataset name	Encoder architecture	# of latent dimension	Decoder architecture
MNIST	512-128	{2,5,10,20}	128-512
Fashion-MNIST	512-128	{2,5,10,20}	128-512
CIFAR-10	1024-512-128	{2,5,10,20}	128-512-1024
Tabula Muris	1024-512-128	{2}	128-512-1024
Fetal liver atlas	1024-512-128	{2,64}	128-512-1024
1.3 Million Brain Cells	2048-1024-512	{20,128,512}	2048-1024-512
scATAC-Seq	1024-512-128	{20}	1024-512-128

Supplementary Table 1. List of models trained on different datasets. For encoder and decoder architectures, each number separated by dash indicates the number of nodes for a single layer. # of latent dimensions indicates set of all numbers of latent dimensions that were used throughout the experiments.

Dataset name	# Class	# Sample	# Feature
MNIST	10	60,000	784
Fashion-MNIST	10	60,000	784
CIFAR-10	2	10,000	3,072
Tabula Muris	3	1,766	2,660
Fetal liver atlas	40	100,000	2,000
1.3 Million Brain Cells	NA	1,308,421	27,998
scATAC-Seq	NA	8,000	244,544

Supplementary Table 2: List of datasets that were used in our experiments.

Cell type category	Cell type	Gene sets
Hepatocytes	Hepatocytes	Aizarani_Liver_C11_Hepatocytes_1 Aizarani_Liver_C30_Hepatocytes_4 Aizarani_Liver_C17_Hepatocytes_3 Aizarani_Liver_C14_Hepatocytes_2
Kupffer cells	Kupffer cells	Aizarani_Liver_C6_Kupffer_Cells_2 Aizarani_Liver_C2_Kupffer_Cells_1 Aizarani_Liver_C31_Kupffer_Cells_5 Aizarani_Liver_C25_Kupffer_Cells_4 Aizarani_Liver_C23_Kupffer_Cells_3
B cells (MHC II Positive)	Pro B cell Pre B cell Pre pro B cell	Aizarani_Liver_C34_MHC_II_pos_B_cells Aizarani_Liver_C38_Resident_B_cells_3 Aizarani_Liver_C8_Resident_B_cells_1 Aizarani_Liver_C22_Resident_B_cells_2
NK/NKT cells	NK Mono-NK Mac NK	Aizarani_Liver_C28_NK_NKT_cells_6 Aizarani_Liver_C1_NK_NKT_cells_1 Aizarani_Liver_C12_NK_NKT_cells_4 Aizarani_Liver_C5_NK_NKT_cells_3 Aizarani_Liver_C3_NK_NKT_cells_2 Aizarani_Liver_C18_NK_NKT_cells_5

Supplementary Table 3. Mapping of gene sets and cell type labels in fetal liver datasets to cell type category.