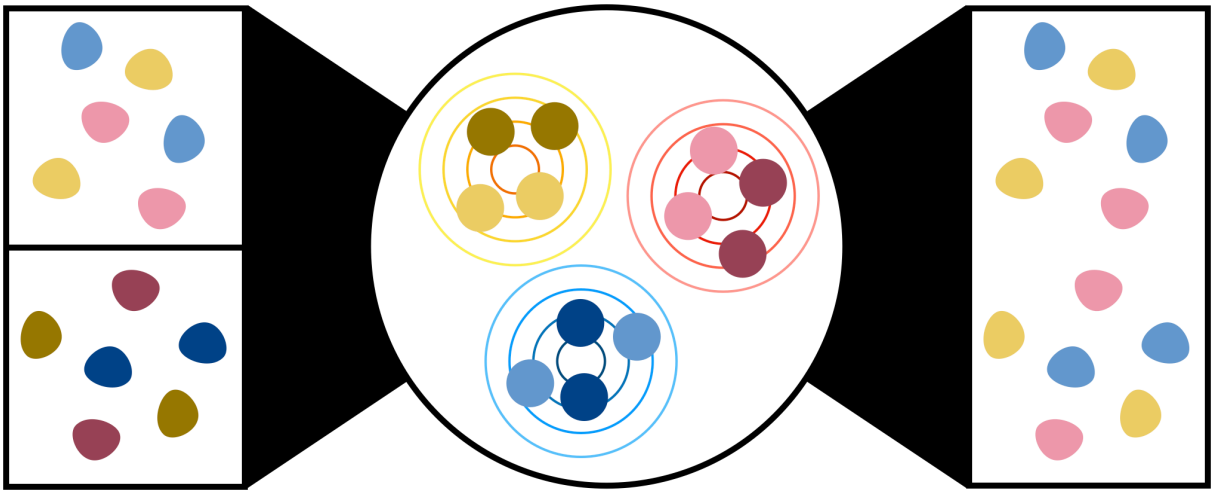
UNIVERSITY OF COPENHAGEN



PhD thesis

Representing and perturbing single-cell transcriptomes using deep generative modelling

Christopher Heje Grønbech

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen

Representing and perturbing single-cell transcriptomes using deep generative modelling

Christopher Heje Grønbech

PhD Thesis

Department of Biology
University of Copenhagen

Supervisors:

Ole Winther
Carl-Johan Ivarsson
Magnus Fontes

Christopher Heje Grønbech

Department of Biology
University of Copenhagen
Copenhagen
Denmark

Qlucore AB
Lund
Sweden

Supervisors:

Ole Winther
Department of Biology
University of Copenhagen
Copenhagen
Denmark

Carl-Johan Ivarsson
Qlucore AB
Lund
Sweden

Magnus Fontes
Institut Roche SAS
Boulogne-Billancourt
France

Summary

Throughout the last fifteen years, several methods and technologies to sequence the RNA of cells on an individual level have appeared. Sequencing RNA tells us about which RNA molecules were present in every single cell, and how many of each molecule. RNA is transcribed from genes in the genome of an organism, and it is either translated into proteins, and it perform other functions in the cell. A gene is said to be expressed in a cell, if the corresponding RNA molecule or protein is present in the cell. Therefore, by sequencing RNA, we can get an idea of which genes are expressed in a cell, how it functions, and what state it is in. This field is called single-cell transcriptomics, since the set of all the transcribed RNA molecules is known as transcriptome. Because there are many genes in a genome, we must also need many cells to model single-cell transcriptomes. This cause data sets to be both high-dimensional and large, which can be difficult to analyse.

Concurrently with the evolution of single-cell transcriptomics, machine-learning methods have also seen tremendous development with the advent of deep learning. Multiple methods have been successfully applied to large amounts of high-dimensional data, for instance, to embed data into lower-dimensional representations. Some deep generative models can be used for this, and as their name suggests, they can also be used to generate new data either unconditionally or given some constraint. Using these methods for single-cell transcriptomics could prove useful for reducing the dimensionality of the data as well as predicting how the data would look like in another state.

In my two research papers, this was achieved by using variational autoencoders (VAE) and adversarial autoencoders (AAE), which are separate, but related deep generative models.

For the VAE approach, four related conditional VAE models (scVAE-C) for integration and perturbation modelling were developed. In addition to a base model, another used a normal-mixture model, a third used a hierarchical structure, and fourth used both. All models performed better at integration than the state-of-the-art VAEs used for comparison, whereas only the nonhierarchical models were competitive with the best comparison methods. Overall, the normal-mixture VAE performed the best.

scAAE applies the AAE to single cell transcriptomics. This method was used to find representations useful for clustering. Because of their adversarial nature, AAEs can have difficulty in converging to an optimum. Other challenges include parts of the AAE overfitting or AAEs only generating one kind of data. These issues were addressed

by implementing an early-stopping scheme using a clustering metric as well as using noise models. This made scAAE successful and competitive with a state-of-the-art VAE.

Sammenfatning

Gennem de sidste femten år er adskillige metoder og teknologier udviklet, der kan sekventere RNA indholdet i individuelle celler. Sekventering af RNA fortæller os om hvilke RNA-molekyler, der var til stede i hver enkel celle. RNA transskriberes fra gener i genomet for en organisme, og det bliver enten oversat til proteiner, eller det bliver brugt til andre funktioner i cellen. Et gen siges at være udtryk i en cell, hvis det tilsvarende RNA-molekyle eller protein er tilstede i cellen. Ved at sekventere RNA kan vi derfor få en ide om hvilke gener, der er udtrykt i cellen, hvordan den fungererer, og hvilken tilstand den er i. Dette studie bliver for enkeltcellet transkriptomik, eftersom sættet af alle de transskriberede RNA-molekyler kaldes transkriptomet. Fordi der er mange gener i et genom, har vi derfor også behov for mange celler til at modellere enkeltcellet transkriptomik. Dette medfører, at datasæt er både højdimensionelle og store, hvilket kan være svært at analysere.

Samtidig med udviklingen af enkeltcellet transkriptomik, har maskinlæringsmetoder også oplevet en enorm udvikling med fremkomsten af dyb læring. Flere metoder er med succes blevet anvendt på store mængder højdimensionelle data, for eksempel for at indlejre data i lavere dimensionelle repræsentationer. Nogle dybe generative modeller kan bruges til dette, og som deres navn antyder, kan de også bruges til at generere nye data. Brug af disse metoder til enkeltcellet transkriptomik kan vise sig nyttig til at reducere dimensionaliteten af dataene samt forudsige, hvordan dataene ville se under andre betingelser.

I mine to forskningsartikler blev dette opnået ved at bruge variationelle autoencodere (VAE) og adversarielle autoencodere (AAE), som er separate, men relaterede dybe generative modeller.

Til VAE-tilgangen blev der udviklet fire relaterede betingede VAE-modeller (scVAE-C) til integration og perturbationsmodellering. Ud over en basismodel brugte en anden en normalmikstursmodel, en tredje brugte en hierarkisk struktur og en fjerde brugte begge dele. Alle modeller klarede sig bedre ved integration end de VAE'er, der blev brugt til sammenligning, hvorimod kun de ikke-hierarkiske modeller var konkurrencedygtige med de bedste sammenligningsmetoder. Samlet set klarede normalmikstur-VAE'en det bedst.

scAAE anvender AAE til at modellere enkeltcellet transkriptomer. Denne metode blev brugt til at finde repræsentationer nyttige til klyngedannelse. På grund af deres kontradiktoriske natur kan AAE'er have svært ved at konvergere til et optimum. Andre

udfordringer omfatter, at dele af AAE'en bliver overfittet, eller AAE'er, der kun genererer én slags data. Disse problemer blev løst ved at implementere en tidlig-stop-ordning ved hjælp af en klyngemåling samt ved at bruge støjmodeller. Dette gjorde scAAE succesrig og konkurrencedygtig med en avanceret VAE.

Publications

Included in thesis

Manuscripts

- Paper A: C. H. Grønbech, R. DeVries, E. Roellin, and O. Winther. scVAE-C: Latent-representation covariate removal and counterfactual expression profiles for single cells using conditional and hierarchical variational autoencoders. Appendix A.1, a
- Paper B: C. H. Grønbech, E. Molinaro, K. Natarajan, and O. Winther. scAAE: Adversarial autoencoders for single-cell gene expression data. Appendix A.2, b

Excluded from thesis

- J. N. Nissen, J. Johansen, R. L. Allesøe, C. K. Sønderby, J. J. A. Armenteros, C. H. Grønbech, L. J. Jensen, H. B. Nielsen, T. N. Petersen, O. Winther, and S. Rasmussen. Improved metagenome binning and assembly using deep variational autoencoders. *Nat. Biotechnol.*, 39(5):555–560, May 2021. DOI:10.1038/s41587-020-00777-4

Contents

Preface

This thesis is the culmination of my PhD programme with the title "Visualisation of Deep Learning on Biomedical Data for Improved Interpretability". This project was also a part of the Marie Skłodowska-Curie Innovative Training Network entitled Machine Learning Frontiers for Precision Medicine (MLFPM), and it was initially proposed by Magnus Fontes and Carl-Johan Ivarsson who both are founders of the biotech software company Qlucore in Sweden.

During my master's thesis, I had worked together with a fellow student, Maximillian Fornitz Vording, on a deep generative model for single-cell transcriptomics data with Ole Winther as supervisor. We both had exchanged physics for mathematical modelling, but by chance, for me at least, we essentially ended up in computational biology. After our master's programmes ended, our work resulted in a publication detailing our method, which was called scVAE.

When the opportunity later arose for me to join Qlucore and MLFPM as well as to work on the above project with Ole as supervisor, I gladly accepted. I intended to continue the work on scVAE to extend it to model multiple data sources and to use it for prediction. However, since I am a data scientist and not a bioinformatician, I was also connected with Kedar Nath Natarajan, who was an associate professor at the University of Southern Denmark studying single-cell transcriptomics among other things. Kedar had worked on a project researching an adversarial alternative to scVAE and similar methods. We started collaborating on this project, but developing and implementing this model proved more problematic than I initially thought, since training such models can be tricky.

In the end, through long days and nights, through trips back and forth between Denmark and Sweden, through failures and successes, through a pandemic, I ended up achieving results competitive with the state of the art for both projects as well as a paper manuscript for each of them. These manuscripts are both included in this thesis together with a synopsis of my research project as a whole.

With all that being said, I hope you find this thesis interesting and that it sheds sufficient light on my research project.

Sources

Both single-cell transcriptomics and deep generative modelling cover a lot of areas. Many technologies and analyses exists for single-cell transcriptomics, and deep generative modelling is comprised of a multitude of models. In writing this thesis, I have therefore tried to focus mostly on aspects of both that are relevant to my research project. For this purpose, I have mainly relied on the following sources for each chapter:

- Chapter 1:
  - Alberts et al. (2022, Chapter 1) for molecular cell biology.
  - Carlberg and Molnar (2016, Chapter 1) for gene expression.
  - Lowe et al. (2017) for transcriptomics technologies.
  - Lücken and Theis (2019) and Heumos et al. (2023) for transcriptomics analyses.
  - Tomczak (2024, Chapter 1) for deep generative modelling.
- Chapter 2:
  - Lafzi et al. (2018) for single-cell RNA sequencing.
  - Lücken and Theis (2019) and Heumos et al. (2023) for single-cell transcriptomics analyses.
- Chapter 3:
  - Tomczak (2024, Chapters 1, 2, 5, and 8) for deep generative modelling in general as well as for variational autoencoders and generative adversarial networks.
  - Bishop (2006, Chapters 1 and 9–12) for probabilistic modelling and machine-learning concepts in general.
- Chapter 4
  - Brendel et al. (2022) for deep-learning applications for single-cell RNA-sequencing data.
  - The scRNA-tools database (Zappia et al., 2018) for finding additional applications.

These sources are also cited more explicitly where applicable. I have assumed some prior knowledge about biology, machine learning, and probability theory, but if needed, the above sources are good starting points.

Sections 3.2 and 3.3 adapt some general aspects of the methods sections of Paper A and Paper B, respectively, but they have been more generalised and expanded, since I have tried to fill in details not covered in the papers.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 813533.

Acknowledgements

I would like to first express my gratitude to my supervisors, Ole, Carl-Johan, and Magnus, for their support and patience. Ole has been supervising me from the beginning of the work that has now resulted in this PhD thesis. I have appreciated our conversations, and I always felt energised after them. Carl-Johan welcomed me with open arms to Qlucore and helped me navigate administrative tasks that would have been strenuous otherwise. And Magnus is a busy man, but nevertheless he made time for meetings and gave valuable input.

Next, I wish to show appreciation to my other collaborators, Kedar, Emiliano Molinaro, Rachael DeVries, and Eljas Roellin, for their work and for helping me with the biological interpretation. I would also like to thank my former colleagues at Qlucore as well as the entire MLFPM network, especially my fellow students and my secondment hosts, Karsten Borgwardt and Krista Fischer. Maximillian deserves my thanks too, since without him, I would properly not have ended up working on this project.

Finally, I would like to thank my parents and my sister for their endless encouragement and for always being there for me.

Christopher Heje Grønbech
København
24 January 2025

1 Introduction

All known organisms are either single cells or consists of multiple cells. In more complex multicellular organisms, cells differentiate and specialise to have distinct functions such as blood cells transporting oxygen to an organism's cells. Biologists divide these specialised cells into different cell types based on function and shape. Cells of related cell types form tissues which in turn form organs which makes up the organisms themselves.

How a cell functions is determined by which genes are expressed in the cell. Genes encode information about an organism as sequences of DNA, and this information concerns how to synthesise proteins (protein-coding genes) or other gene products (non-coding genes). These syntheses start by transcribing the DNA sequences into different types of RNA molecules. Protein-coding genes are transcribed into messenger RNA that each carry a set of encoded instructions from the nucleus of the cell to a ribosome that can translate the instructions into a protein. There a many different kinds of proteins with different functions. Among other things, they can be enzymes, which catalyse chemical reactions, they can send, transport, or receive molecules around the body, they can regulate DNA transcription, or they can provide structure to cells. Non-coding genes are transcribed into RNA molecules that perform various functions such as helping in the production of proteins as well as regulating this same production. A gene is expressed in a cell if its gene product is present in the cell, and the abundance of its gene product determines its expression level.

The sequences of DNA of all genes as well as other DNA sequences are arranged in a certain way to form DNA molecules, and these genes make up the genome of an organism. Because the DNA sequences of genes can vary between organisms of the same species, each organism has a unique set of gene variations (alleles), which is its genotype. This difference in genotype together with differences in environmental factors and interactions between the two are the reason why traits of an organism, its phenotype, vary from another organism.

Each cell nucleus in a multicellular organism contains the genome of that organism, however, genes are still expressed differently from cell to cell. This is because of how gene expression is regulated in each cell, and this gene regulation leads to cells differentiating. As mentioned above, gene regulation could result from interference by proteins or RNA molecules. This kind of regulation arise from the genome, and regulatory regions even exists in the DNA sequences themselves. Even the structure of the

DNA molecules can hinder or facilitate the transcription of certain genes. The gene expression of a cell can also change because of environmental conditions such as diseases or other outside stimulation.

Biologists study the expressions of genes in cells (gene expression profiles) under normal conditions and compare them with cells in different states or at different times. In this way, biologists can learn and discover which genes result in what functions and how genetic, structural, and environmental factors among others regulate these genes. For a diseased individual, knowing which gene variations they have relevant to the disease, medical practitioners could then customise a treatment specific for that individual – a field called *personalised medicine*.

1.1 Gene-expression experiments

One way do measure the gene expression is to quantify the RNA transcripts in a cell – all together called the *transcriptome*. Since DNA is a more stable molecule than RNA, RNA is usually reverse transcribed to complementary DNA (cDNA), and methods used to quantify DNA can then also be used for RNA. At first, genes were detected or quantified by determining the RNA sequence (sequencing) of random transcripts in a sample of cells from an organism and mapping the sequences to the genome of the species or the organism. This was a slow process and required extensive manual labour.

As an alternative, methods that used probes to detect specific genes in a cell sample were developed. These probes bind to RNA or cDNA molecules matching the specified genes, and this makes the probes emit light, usually using fluorescence. In this way, the genes can be detected and the level of their expression can be quantified by measuring the amount of light emitted. One example is the reverse-transcription quantitative PCR (RT-qPCR) method (Kubista et al., 2006), which uses the polymerase chain reaction (PCR) to amplify cDNA and quantifying it by measuring the amplification.

One major disadvantage of most probing methods is that they can only investigate a small subset of genes at a time. One exception is the microarray (Nelson, 2001), which consists of thousands of microscopic spots, and each spot can contain a probe for a specific genes. Because of this, a microarray is able to detect thousands of genes at the same time – effectively quantifying a known transcriptome in that size range. Coupled with runtimes of one to two days per experiment, this is why microarray is a popular choice for transcriptomics. However, since probing methods only test for already known genes, they cannot be used to identify new genes.

Developments in sequencing DNA lead to methods able to do so in parallel on a massive scale, which is why such a method is categorised as massively parallel sequencing. This allowed RNA sequencing (RNA-Seq; Wang et al., 2009) to sequence the whole transcriptome of a cell sample in days or less. Gene expression could then be quantified by either aligning the RNA transcripts to a reference genome or to each other if no reference genome existed. Because of this, RNA-Seq is the other popular method for transcriptomics.

Until 2009, a significant drawback of RNA sequencing methods was that they could only quantify gene expression in a sample of cells and not individually (Aldridge and Teichmann, 2020). Since a cell sample could contain a mix of cells, the gene expression of the the cell sample would be an average of the these cells, and it would be difficult to identify specific kinds of cells. Probing methods had been to able to quantify the expression of specific genes of single cells using RT-qPCR and fluorescence *in situ* hybridisation (Levsky et al., 2002), but they were still limited in the number of genes they could detect at a time. As for sequencing methods, further advances of DNA and RNA sequencing resulted in techniques able to separate cells and sequence them individually (Sandberg, 2014). These are called single-cell RNA-sequencing (scRNA-seq) methods, and they lead to *single-cell transcriptomics* where gene expression of individual cells are studied.

1.2 Transcriptomics analyses

With gene expression quantified, the next step is usually to determine genes that are differentially expressed in (samples of) cells of different types or in different states by comparing their gene expression profiles. This is done with different kinds of statistical testing. Since there are usually thousands of genes, such analyses can be computationally cumbersome, so several methods have been employed to lower the number used in the analyses. Some methods rely on filtering the genes. This usually involves removing genes with little-to-no expression or genes with constant expression (called *housekeeping genes*) and only keeping genes with expression that varies the most between samples or cells. Others rely on combining the expressions of multiple sets of genes to a lower number of features that are used instead of the gene expression levels (Xiang et al., 2021). Such approaches are different kinds of dimensionality reduction, and they can only be used for analyses at the cell level. Both of these kinds of methods are of course not mutually exclusive.

A method often used for dimensionality reduction is principal component analysis (PCA) (Pearson, 1901). PCA linearly transforms data into a new representation whose components are computed based on how much variation they capture of the input data. These are called principal components, and the first principal component captures the most variation, the second principal component captures the second most variation, and so on. These principal components are all enforced to be orthogonal to each other, so they can be used as a coordinate system to visualise the data. PCA is generally used as the first step of other dimensionality reduction in scRNA-seq analyses (Heumos et al., 2023).

Another linear approach is factor analysis (Thurstone, 1935). This method supposes that multiple observed variables can be statistically represented by fewer unobserved variables (factors) than the number of observed ones. This is done by modelling the observed variables as noisy combinations of the factors, and these are learnt by fitting such a model to the data. These factors can then be used in place of the observed variables. Factor analysis have been used for dimensionality reduction for scRNA-seq

data (Pierson and Yau, 2015) and for other purposes for transcriptomic data from both microarrays and RNA sequencing (Hochreiter et al., 2006; Risso et al., 2014).

1.3 Deep learning and generative models

The autoencoder (Hinton and Zemel, 1993) is a machine-learning method that can be used to reduce dimensionality. This method uses artificial neural networks to model the data: one network called the *encoder* to transform data to a compressed or lower-dimensional representation and another network called the *decoder* to reconstruct the original data from this representation. Artificial neural networks (Bishop, 2006, Chapter 5), which took their inspiration from biological neural networks, consists of multiple layers of mostly nonlinear transformations. In machine learning, these are usually just called neural networks, and we will follow that convention in this thesis. Machine-learning methods in general have seen tremendous development and successful applications on large amounts of high-dimensional data in the last decade (Shao et al., 2022). Advancements in computational power using graphical processing units (GPUs), lead researchers in 2009 (the same year as the first use of scRNA-seq) being able to practically train neural networks several layers deep (Raina et al., 2009). Using these deep neural networks is referred to as *deep learning* (LeCun et al., 2015). Numerous deep-learning methods have been developed for transcriptomics (Cheng et al., 2024), and several autoencoders have been used for scRNA-seq data (Eraslan et al., 2018; Deng et al., 2019; Wang et al., 2019; Amodio et al., 2019).

The factors of factor analysis are also called latent variables, and factor analysis is a special case of a *latent-variable model* where both observed variables and factors are continuous (Bishop, 2006, Chapter 12). The latent-variable model itself is a kind of generative model. Generative models are so named because they can be used to generate new observations given a target value, for example, for a latent variable. This is in contrast to discriminative models, which can discriminate between target values given an observation. Combining generative models with deep-learning approaches has resulted in deep generative models (Tomczak, 2024).

One popular generative model is the variational autoencoder (VAE; Kingma and Welling, 2014; Rezende et al., 2014), which uses probabilistic modelling and neural networks to compress data to a latent representation as well as generate the original data from this representation. The similarity in structure to the autoencoder method is why the VAE is called an autoencoder. VAEs have been used to model both bulk RNA-Seq data (Way and Greene, 2017) and scRNA-seq data (Wang and Gu, 2018; Lopez et al., 2018; Ding et al., 2018).

The generative adversarial network (GAN; Goodfellow et al., 2014) is another popular generative model. GANs consist of two neural networks: One network generates samples from random noise trying to match real samples, and another network tries to discriminate between the generated samples and the real samples. These networks are then trained against each other usually until the generator reaches acceptable perform-

ance. GANs have also been used to analyse both bulk RNA-Seq data (Murchan et al., 2021) and scRNA-seq data (Gunady et al., 2019; Xu et al., 2020; Zhou et al., 2021).

Large language models (Tomczak, 2024, Chapter 11), or more generally foundation models, are a recently popular category of generative models. They can predict missing elements of a sequence based on the other elements of the sequence by paying special attention to multiple key elements. As their name alludes to, this models consists of a large number of parameters, so they also require a large amount of data for training. The generative pre-trained transformer (GPT) models (Vaswani et al., 2017; Radford et al., 2019; Brown et al., 2020) are examples of this kind of model. Large language models have also been used for single-cell transcriptomics (Shen et al., 2023; Chen et al., 2024).

1.4 Research objectives

The overall aim of my research project, which this thesis detail, was to explore and develop deep generative methods that would be effective at modelling single-cell transcriptomic data. Such methods should be able to learn lower-dimensional representations of gene expression profiles. The goal of these representation was for them to replace gene expression profiles themselves either to easier interpret visualisations of them or for improved performance in downstream analyses.

Various biological data such as clinical and technical information are also often included with transcriptomics data (Gagnon et al., 2022). This could, for instance, be because the transcriptomics data comes from multiple sources, for example, patients. Gene expression often varies depending on this kind of associated data. Therefore, another purpose of my research project was to investigate and develop methods that use this additional biological data when modelling single-cell transcriptomics data. One reason for this was to determine how well methods could remove the variation related to associated data from their learnt representations. This is referred to as *integration*, or *data correction* when done on the original data (Heumos et al., 2023). Another reason was to see if it was possible for such a method to take the gene expression profiles of cells under one set of conditions and generate artificial expression profiles for those same cells under different circumstances by only changing the associated biological data. For instance, one could provide gene expressions of cells in a diseased state to a method and use that method to predict the response in gene expression to different kinds of treatments. This is one part of perturbation modelling (Heumos et al., 2023).

In this thesis, I give an account of the outcome of my research project. I describe single-cell transcriptomics and deep generative modelling in relation to my research in Chapters 2 and 3. In Chapter 4, I present applications of deep generative modelling to single-cell transcriptomics, including models that I have developed as part of my research in collaboration with others. Finally, I conclude my thesis in Chapter 5 by summing up my findings and reasoning about how they can be used for further research.

2 Single-cell transcriptomics

The aim of single-cell transcriptomics is to quantify the different RNA transcripts in individual cells. Although this can be done using single-cell gene-probing methods for organisms with small genomes (for example, using microarrays), the most common method is single-cell RNA sequencing (scRNA-seq). In scRNA-seq, RNA transcripts of individual cells are sequenced and used to estimate how much each gene is expressed in each cell. This is in contrast to traditional transcriptomics techniques that examines the gene expressions of cells in bulk. Sequencing the transcripts in a pool of cells of different cell types mixes the gene expressions of all the cells together making it difficult to discern their individual functions. Single-cell transcriptomics has enabled several new discoveries (Aldridge and Teichmann, 2020).

2.1 Sequencing and quantification

There are many scRNA-seq methods, and most follow the same steps of extracting and isolating cells, preparing transcripts for sequencing, and the sequencing itself (Lafzi et al., 2018). These steps are described below.

Cells are first extracted from their environment such as a tissue, and they are then isolated. This is usually done by placing each cell into one of several small wells arranged on a plate (Ramsköld et al., 2012; Hashimshony et al., 2012; Gierahn et al., 2017) or capturing it in a microfluidic droplet (Klein et al., 2015; Macosko et al., 2015). Droplet-based methods can sequence many more cells simultaneously than methods using plates, but they are also more costly, so they are more popular in research, while plate-based methods are still used in clinical environments (Lafzi et al., 2018).

After the cells have been isolated, the transcripts are prepared for sequencing in a process called library construction. In each well or droplet, the cells are first lysed, and the desired RNA, typically mRNA, is isolated. The RNA transcripts in each cell are then reverse transcribed to cDNA, since it is more durable than RNA. Because cells do not contain enough RNA content to properly sequence, the produced cDNA is amplified using either PCR or *in vitro* transcription (IVT; Beckert and Masquida, 2011). The amplified cDNA is then divided into fragments of the same sequence length, since sequencing is limited by the length of the molecules. Finally, these transcript fragments are pooled together (multiplexed) and sequenced in parallel to allow high throughput.

The sequenced fragments are called reads. Because of the pooling of the fragments, the reads of all cells mixed up together. Tagging the reverse-transcribed cDNA in each cell with a cell-specific barcode before amplification, the reads are able to be associated with individual cells (demultiplexing). The reads can then be aligned to a reference genome to be able to estimate how many times a gene was read in each cell, thereby producing read-count data. In the absence of a reference genome, the reads can be aligned to each other to produce a transcriptome using *de novo* assembly (Raghavan et al., 2022).

Due to the amplification of cDNA, it is not possible to determine how many transcripts of a each gene were present in the original cell. This is usually solved by tagging the cDNA from each transcript with a unique molecular identifier (UMI) before amplification in addition to the cell barcode. By only counting unique reads, the number of times a gene was expressed in a cell is estimated instead, thereby producing a transcript-count data. Another way to account for the amplification is to add in so-called spike-in RNA, which is RNA of known sequence and quantify to use for normalisation during preprocessing.

To ease sequencing and alignment, many methods select only one of the end fragments of each transcript to sequence instead of sequencing all fragments of a transcript. This, however, prevents these methods from performing *de novo* transcriptome assembly, which is why some methods still perform full-length sequencing.

2.2 Characteristics of scRNA-seq data

The count data resulting from scRNA-seq present several challenges. Some of these challenges arise from the nature of the data, and some are caused by the processing before and during sequencing. One complication results from the number of genes. Genomes consists of thousands of genes in bacteria to tens of thousands of genes in complex multicellular organisms (Alberts et al., 2022). The human genome contains around 20 000 protein-coding genes and at least 17 000 noncoding genes as an example (Amaral et al., 2023). Because of this, scRNA-seq data are high-dimensional. This makes some analyses cumbersome or even impossible to perform.

Even with an abundance of genes, each cell typically only expresses about half of them (Carlberg and Molnar, 2016, Chapter 1). Since sequenced cells usually are of different cell types, it will also differ which subset of genes are expressed in each cell. Additionally, cells will also be in different states of the cell cycle or even dying resulting in more differences of gene expression (Lücken and Theis, 2019). This all results in scRNA-seq data being sparse, which can make it more difficult to discern patterns in the data. The differences in gene expression as well as the multilevel interactions between genes as part of gene regulation (Carlberg and Molnar, 2016, Chapter 1) further complicates pattern recognition.

Another source of sparseness is due to some RNA transcripts not being sequenced even though they were present in the cell (Lücken and Theis, 2019). These are called technical *dropout events*. Ambient RNA from other cells lysed during extraction of the

cells can also be captured together with a particular cell and be mistaken as coming from that cells (Heumos et al., 2023). Because of ambient RNA, empty droplets will still be measured as having some genes expressed, causing more sparseness. For droplet-based methods, it can also happen that no cell is captured in a droplet or that two or more cells (doublets or multiplets, respectively) are captured. As for doublets or multiplets, these can be mixtures of different cell types, making it difficult or even impossible to disentangle the gene expressions of each cell (Heumos et al., 2023).

As a result of the high dimensionality as well as the complexity of the data, a large number of cells is needed to be able to achieve enough statistical power to discern if a particular result is true or false (Angerer et al., 2017). Droplet-based methods enable sequencing thousands of cells at a time, and usually experiments are performed in batches to sequence even more cells, resulting in data sets consisting of up to millions of cells (Lafzi et al., 2018). As with high dimensionality, large data sets are can also be difficult to analyse.

scRNA-seq data will also include information about experiment batches as well as experimental details such as how and when cells were treated or stimulated. Additionally, clinical information like donor ID, age, sex, or disease state can be provided if cells differ in this way (Gagnon et al., 2022).

2.3 Preprocessing steps

Because of the complications mentioned in §2.2 as well as others mentioned below, scRNA-seq data needs to be preprocessed before performing most analyses, and this is done in multiple steps of quality control, normalisation, log-transformation, and data correction detailed below (Lücken and Theis, 2019).

2.3.1 Quality control

At first, low-quality cells and empty droplets as well as doublets and multiplets are removed based on certain quality-control metrics. Since the number of cells associated with a barcode can vary, these metrics are formulated in terms of the barcode (Lücken and Theis, 2019). The metrics include the total number of transcripts or reads per barcode (called the count depth or the library size), the number of genes detected per barcode, as well as the fraction of transcripts of reads from mitochondrial genes per barcode (Lücken and Theis, 2019). Barcodes with low count depth, few genes detected, and a high mitochondrial fraction are usually deemed to be low-quality cells or empty droplets (Lücken and Theis, 2019). On the other hand, barcodes with unusual high count depths are typically considered to be doublets or multiplets (Lücken and Theis, 2019). These can also be detected using dedicated methods (Germain et al., 2021; Kang et al., 2018). In addition to filtering cells, genes that are only detected in a few cells are also removed (Lücken and Theis, 2019).

2.3.2 Normalisation and log-transformation

Because not all RNA transcripts are evenly prepared and sequenced within an experiment batch, the gene expression will vary between even identical cells. Therefore, the counts of each cell are normalised by a certain *size factor*. This size factor can simply be the count depth of each cell or it can be more differentiated to account for the differences in, for instance, cell size between cell types or cell-cycle states. (Heumos et al., 2023)

For scRNA-seq methods that sequence the full length of transcripts, an additional normalisation is performed. Since the transcript fragments all have the same length, and since genes vary in sequence length (Alberts et al., 2022, Chapter 1), a gene that has a sequence twice as long as another, will also have about twice as many fragments aligned to it, if they are expressed in the same amount. Therefore, the number of reads aligned to each gene are normalised by the sequence length of that gene (Lücken and Theis, 2019).

After normalisation, the transcript count data is generally shifted by +1 and log-transformed: $\log(x + 1)$, where x represents a transcript count. This is to reduce the relationship between the mean and the variance of the data as well as the skewness of the data. Traditionally, gene expression levels were also expressed as logarithm fold changes, and many analyses expect that as their input. (Lücken and Theis, 2019)

2.3.3 Data correction

Combining data from multiple experiment batches complicates data sets further. This is because each experiment will be different due to random differences in the setup (Lücken and Theis, 2019). To remove this technical variation, several batch-correction methods exist (Büttner et al., 2019). Some of these methods rely on modelling the gene expression with the batch contribution as a covariate thereby regressing out its effect (Büttner et al., 2019).

In addition to integrating different batches of an experiment, different data sets can also be integrated. This is done when constructing single-cell atlases for instance (Hrovatin et al., 2025). However, the more different the compositions of the data sets are, the more difficult they will be to integrate because of their heterogeneity. Therefore, dedicated methods for integrating scRNA-seq data sets have been developed (Butler et al., 2018; Korsunsky et al., 2019).

Variation due to biological effects can also be controlled for using regression. This can, for example, be high mitochondrial content, which could be caused cell stress, or cell-cycle effects. However, one should be wary of removing biological effects, since they can be informative and used to explain other biological effects. Because of these interdependencies, covariates should always be regressed out together. (Lücken and Theis, 2019)

Data acquired using droplet-based methods, can also be corrected for ambient RNA. Since empty droplets would only contain ambient RNA, the gene expressions for these

can also be used to remove the contribution of ambient RNA to the gene expressions in other cells. (Heumos et al., 2023)

Methods also exist for correcting dropout events by inferring counts either from the data themselves or from reference data. These methods involve both determining whether a zero count is a dropout event or an actual zero as well as estimating the expression level in the former case. One should, however, exercise caution when inferring counts, since it could amplify noise in the data. (Lücken and Theis, 2019)

2.3.4 Feature selection

Because of the high dimension and the sparseness of the data, genes are typically filtered to only keep only the most relevant ones. Genes that are only expressed in a few cells have already been removed during quality control, but many genes generally remain. Since the purpose of transcriptomics is to explain differences in gene expression between cells, the most relevant genes are those that vary much from cell to cell. These genes are referred to as *highly variable genes*, and there exist several methods to select these (Satija et al., 2015; Zheng et al., 2017; Stuart et al., 2019). By only keeping highly variable genes, housekeeping genes are also removed. (Heumos et al., 2023)

2.3.5 Dimensional reduction

For analyses that do not require gene-level information (Lücken and Theis, 2019), scRNA-seq data can be further compressed using dimensionality-reduction methods. Analyses that could benefit from dimensionality reduction suffer from the *curse of dimensionality*. This phenomena occurs because the volume of a given data space grows exponentially with the dimensionality of the space, and to have enough data points to cover the volume, one would then need a exponentially large amount of data, otherwise the data become sparse. For analyses that rely on distances between data points, this becomes a problem, because if the data are sparse, the distances between them will be similar to each other. (Bishop, 2006, §1.4)

Even though data can be high dimensional, it can sometimes in reality occupy a much smaller space of a lower dimensionality. Some dimensionality-reduction methods attempt to find this space by projecting the high-dimensional data into a space of as few dimensions as possible while still retaining its overall structure. (Bishop, 2006, §1.4)

Since scRNA-seq data are sparse, methods to reduce its dimensionality for relevant analyses have been employed. Principal component analysis (PCA) is typically used, but factor analysis have also proven useful (§1.2). Dimensionality-reduction methods can also be used to visualise data and are discussed below.

2.4 Analyses

scRNA-seq data can be analysed in many ways, and analyses are performed on the level of genes, cells, or both (Lücken and Theis, 2019). They are usually implemented as software tools or libraries in the programming language R or Python and available

via Bioconductor or the Python Packaging Index (Zappia et al., 2018), respectively, or they are part of multifunctions software tools such as Seurat (Satija et al., 2015) and Scanpy (Wolf et al., 2018). Below are described types of analyses relevant to my research project.

2.4.1 Differential-expression analysis

To distinguish cells of different cell types or in different states and find genes that vary in expression between them, differential-expression analysis is performed. This consists of different statistical tests. To be able to capture as much biological variation as possible, it is recommended that differential-expression analyses are performed using the raw count data (that is, the data before normalisation and correction). The different covariates should instead be modelled directly in the analyses. (Lücken and Theis, 2019)

2.4.2 Visualisation

Gene expression data can be visualised in many ways such as using a heat map to show gene expression of each gene for each cell. However for scRNA-seq data, this is only be tenable for smaller data sets after feature selection. Therefore, to visualise the structure of the data, one of several dimensionality-reduction methods are used. PCA can be used for this task, but because of its linear nature, it is unable to capture more complex patterns in the data (Lücken and Theis, 2019). For this reason, nonlinear approaches are instead preferred.

Two of the most popular methods for visualising scRNA-seq data (Lücken and Theis, 2019) are t-distributed neighbourhood embedding (t-SNE; van der Maaten and Hinton, 2008) and uniform manifold mapping and projection (UMAP; McInnes et al., 2020). Both methods rely on constructing a network graph of its k-nearest neighbours (Fix and Hodges, 1951) by computing a distance matrix between the gene expression profiles of the cells using a specified distance metric. t-SNE uses this graph to embed the data into a two- or three-dimensional space aiming to preserve the local structure of the data, whereas UMAP also attempts to maintain the global structure. UMAP is also faster to run than t-SNE and able to scale to larger data sets (Becht et al., 2018).

Even though these methods are effective at reducing data to two or three dimensions, these representations should not be used for other analyses, since the intrinsic dimension of the data is most often higher (Lücken and Theis, 2019). Furthermore, since t-SNE only focuses on local similarity in the neighbourhood graph (van der Maaten and Hinton, 2008), distances between cells can be distorted, especially for cells fart apart in gene-expression space. A similar effect is observed for UMAP, despite its additional focus on global structure (McInnes et al., 2020).

2.4.3 Clustering

Another way to distinguish cells or to determine genes with differences in expression is to cluster them. Clustering can be performed on the gene-level, the cell-level, or even both (Lücken and Theis, 2019).

Genes are usually clustered using hierarchical clustering (Nielsen, 2016, Chapter 8). This kind of method works either from the top-down or the bottom-up. The top-down approach starts with one cluster that is split into smaller clusters based on differences between data points recursively until little difference remain in each cluster. Conversely, the bottom up approach works by collecting data points into clusters and clusters in to larger clusters based on similarity. Genes and cells can also be clustered together this way using biclustering (Fang et al., 2021).

On the cell level, dimensionality-reduced data are typically used. A simple approach to clustering is k-means clustering (Lloyd, 1982), which divide the data into k clusters in an iterative process. First, the centres of the clusters (the centroids) are initialised randomly. Then, data points are assigned to the cluster of the nearest centroid, and the centroid of each cluster is computed as the mean of the data points assigned to that cluster. This is repeated until equilibrium. For scRNA-seq, different distance metrics have been used (Wang et al., 2017; Haghverdi et al., 2018; Kim et al., 2019).

A more recent approach for cell-level clustering is community detection. This approach work on network graphs by finding structures whose members are densely connected. For scRNA-seq data, a k-nearest neighbours graph is used (Lücken and Theis, 2019). Two popular methods for community detection are the Louvain method (Blondel et al., 2008) and the Leiden method (Traag et al., 2019), which is an improvement upon the former method.

Once clusters have been formed, they can be annotated by *marker genes*. These are genes that are indicative of a certain cell type and can be looked up in reference database (Heumos et al., 2023).

2.4.4 Perturbation modelling

As mentioned in Chapter 1, one can infer gene functions by measuring similar cells under different conditions, which are also called perturbations. These perturbations can be investigated experimentally by measuring gene expressions from differently perturbed cells and predicting the perturbation from the gene expressions. Conversely, one can model the gene expression for unperturbed and perturbed cells and then try to predict perturbed gene expression profiles from unperturbed cells. This can, for example, by done using deep generative modelling. (Heumos et al., 2023)

3 Deep generative modelling

Suppose we have a set of observations, represented by a matrix $\mathbf{X}$, where each observation, represented by a N-dimensional column vector $\mathbf{x} \in \mathcal{X}^N$, have an associated label $y \in \mathcal{Y}$. $\mathbf{X}$ can be images, text, or experimental measurements such as gene expression profiles, and an appropriate data space $\mathcal{X}^N$ will be chosen depending on the kind of data. $\mathcal{Y}$ can, for instance, be a categorisation of certain kinds of subjects or objects such as cell types. If we want to predict the class y^* of some new observation $\mathbf{x}^*$, we can find a function $f(\mathbf{x})$ to do so directly: $y = f(\mathbf{x})$. Here, $f(\mathbf{x})$ is called a *discriminative function*, since it is a function to discriminate between classes. Since $f(\mathbf{x})$ maps a observation $\mathbf{x}$ directly to a class y, however, the model cannot express any uncertainty about this assignment.

Instead of a deterministic approach, we can instead model the probability distribution of y conditioned on $\mathbf{x}$, $p(y|\mathbf{x})$. By representing the probabilities of each class in $\mathcal{Y}$ given a observation $\mathbf{x}$ as a vector function $\boldsymbol{\pi}(\mathbf{x})$, we can, for example, parameterise $p(y|\mathbf{x})$ using a linear model: $\boldsymbol{\pi}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} + \boldsymbol{\epsilon}$ with a *weight matrix* $\mathbf{W}$, a *bias vector* $\mathbf{b}$, and some noise $\boldsymbol{\epsilon}$. The output of $\boldsymbol{\pi}(\mathbf{x})$ would then be a probability vector, whose components are between 0 and 1, inclusive, and all sum to 1. A neural network with K layers can also be used (Bishop, 2006, Chapter 5):

$$\boldsymbol{\pi} = \mathbf{a}_K, \quad \mathbf{a}_k = h_k(\mathbf{W}_k\mathbf{a}_{k-1} + \mathbf{b}_k), \quad \mathbf{a}_0 = \mathbf{x}, \tag{3.1}$$

where $\mathbf{a}_k$ and $h_k(\cdot)$ are the activation and the activation function of the kth layer of the neural network, respectively. We can then optimise the parameters of the model to obtain the most likely parameters that predict the correct label for a given observation. $p(y|\mathbf{x})$ is called a *discriminative model*.

However, observations are just that: observations about the world, and observations of the same phenomenon differ. No images are alike, a thousand words about an image are seldomly written in the same way, and cells of the same cell type can vary in gene expression. Because of this randomness, we treat the observations as samples $\mathbf{x}$ from a probability distribution of the data. To include the randomness of the observations in a model, we can model the observations together with the labels in a joint probability distribution $p(\mathbf{x}, y)$. Using the product rule, $p(\mathbf{x}, y)$ can be factorised in one of two ways:

$$p(\mathbf{x}, y) = p(y|\mathbf{x})\,p(\mathbf{x}) = p(\mathbf{x}|y)\,p(y). \tag{3.2}$$

Here, $p(\mathbf{x})$ and $p(y)$ are the marginal probability distributions of the observations and labels, respectively, and they can be obtained using the sum rule. As an example, the

marginal distribution of $\mathbf{x}$ can be obtained by summing over (or *marginalising out*) y:

$$p(\mathbf{x}) = \sum_{y} p(\mathbf{x}, y). \tag{3.3}$$

$p(\mathbf{x}, y)$ is called a *generative model*, because we can use it to generate new $(\mathbf{x}, y)$ pairs by sampling from it. By using the second factorisation in equation (3.2) and estimating $p(\mathbf{x}|y)$, we can even generate new observation given a certain class. With the first factorisation, we get a discriminative model that also models the uncertainty of the observation. We can again parameterise these models using linear models or neural networks, for instance, and then optimise the parameters, but because of the different factorisation and marginalisations, we can also estimate them from each other.

The marginal distribution $p(\mathbf{x})$ is an estimate of the data distribution of $\mathbf{x}$. However, since we only have a discrete number of observations or samples from the data distribution, we are, in fact, not directly estimating the *true* data distribution, but rather the *empirical* data distribution:

$$p_{\mathcal{D}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \delta(\mathbf{x} - \mathbf{x}_m), \tag{3.4}$$

where $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ are the observations, M is the number of observations, and $\delta(t)$ is the delta function. Thus, $p_{\mathcal{D}}(\mathbf{x})$ will be $\frac{1}{M}$ when $\mathbf{x}$ is an observation, and 0 otherwise. We can use $p(\mathbf{x})$ to better express how these observations are distributed in the data space, for example, where observations are more or less concentrated, as well as quantify the uncertainty of the observations. We also aim for our model to generalise well to unseen data, so we can use the marginal distribution $p(\mathbf{x})$ to estimate the probability of an unseen observation as well as its uncertainty.

The models above have been introduced in a supervised setting, where the labels y are freely available together with the observations $\mathbf{x}$. However, labels can be hard to come by, so the models can also be formulated without labels in the unsupervised setting. For generative models, this means only estimating $p(\mathbf{x})$, whereas one can, for example, introduce surrogate labels for the discriminative models. There is also the semi-supervised case, where only some observations have labels assigned. Additionally, a model being discriminative does also not prevent it from being generative or the reverse, as it can contain aspects of both as we saw above. As with all categorisation, it is useful being able to distinguish between items to be able to discuss related items separately and characterise them.

With that being said, there are also different kinds of generative models. By using deep neural networks to parameterise them, we enable these models to become more flexible in their expression. The result is deep generative models, and they can be divided into the following classes (Tomczak, 2024): *latent-variable models*, which use latent variables to model observations; *generative adversarial networks*, which use one neural network to generate samples and another to discriminate between generated

samples and observations; *energy-based models*, which model observations and optionally latent variables together using energy functions; *flow-based models*, which use invertible transformations to map observations to a latent representation and back again; *autoregressive models*, which predict observations or their components by other observation or components coming before them; *score-based models*, which model the score function, $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, rather than the $p(\mathbf{x})$, and *large language models* as described in §1.3.

The overall goal of my research project was to model data to learn a lower-dimensional representation and optionally use this representation to predict how data would look like given a perturbation. Therefore, generative models seem like a good fit. Latent-variable models, energy-based models, and flow-based models are all capable of representing observation using latent variables. However, flow-based models require the invertible transformations to be of the same dimensionality, and energy-based models are slow to compute, especially for high-dimensional data. Language models can also represent data in embeddings (Tomczak, 2024, Chapter 11.2), but since they require large amounts of data to train, they were not considered for this project. Latent-variable models remain, and one such model that has been proven to produce interpretable representations is the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014)), which is based on *variational inference* (Bishop, 2006, §10.1). A related model that instead is based on generative adversarial networks (Goodfellow et al., 2014) is the adversarial autoencoder (Makhzani et al., 2016).

I have used both variational autoencoders and adversarial autoencoders in my research, so I will focus on these models below. But since generative adversarial networks can also be seen as a latent-variable model, I will first outline latent-variable models.

3.1 Basics of latent-variable models

The aim of latent-variable models is to represent the observations $\mathbf{x} \in \mathcal{X}^N$ in terms of another variable $\mathbf{z} \in \mathcal{Z}^L$. Since $\mathbf{z}$ is not observed, it is called a latent variable. For the latent variable to capture higher-order features of $\mathbf{x}$, the dimension L of $\mathbf{z}$ is selected to be lower than dimension N of $\mathbf{x}$, $L < N$. Additionally, the latent space is usually chosen to be real valued, $\mathcal{Z} = \mathbb{R}$, and we will do so as well here. The joint probability distribution of $\mathbf{x}$ and $\mathbf{z}$, parameterised by $\boldsymbol{\theta}$, is factorised in the following way:

$$p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) = p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z}), \tag{3.5}$$

where $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ is the likelihood function, which measures how likely that $\mathbf{z}$ would give rise to an observation $\mathbf{x}$. $p_{\boldsymbol{\theta}}(\mathbf{z})$ is called the prior distribution of $\mathbf{z}$, since it it the distribution that $\mathbf{z}$ is assumed to follow without any prior information about the data. All distributions are not necessarily parameterised, but we leave the option open for them to be so. Since $\mathbf{z}$ is unobserved, we marginalising it out (by integration, since $\mathbf{z} \in \mathbb{R}^D$) to get the marginal likelihood function:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\, \mathrm{d}\mathbf{z} = \int p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{3.6}$$

Thus, we arrive at a probability distribution for $\mathbf{x}$ in terms of the latent variable $\mathbf{z}$ and the parameterisation $\boldsymbol{\theta}$.

To estimate $\boldsymbol{\theta}$, we can maximise the marginalised likelihood function over all observations, $p_{\boldsymbol{\theta}}(\mathbf{X})$, to find the parameters that most likely give rise to the observations. This is called maximum-likelihood estimation. By assuming that the observations are independent and identically distributed, $p_{\boldsymbol{\theta}}(\mathbf{X})$ can be factorised as $p_{\boldsymbol{\theta}}(\mathbf{X}) = \prod_m p_{\boldsymbol{\theta}}(\mathbf{x}_m)$, where $\mathbf{x}_m$ is an observation in the data set $\mathbf{X}$. The maximum-likelihood estimate of $\boldsymbol{\theta}$ is then

$$\boldsymbol{\theta}_{\mathrm{ML}} = \operatorname*{argmax}_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{X}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \prod_m p_{\boldsymbol{\theta}}(\mathbf{x}_m). \tag{3.7}$$

If $p_{\boldsymbol{\theta}}(\mathbf{x}_m)$ is small, the product will be even smaller, and this can cause $p_{\boldsymbol{\theta}}(\mathbf{X})$ to be too small to be represent by, for example, a computer. Since the logarithm is monotonically increasing function, it does not change the maximum. Therefore, the logarithm is usually computed instead:

$$\log p_{\boldsymbol{\theta}}(\mathbf{X}) = \sum_m \log p_{\boldsymbol{\theta}}(\mathbf{x}_m). \tag{3.8}$$

With $\boldsymbol{\theta}$ estimated, $\mathbf{z}$ can then be determined by factorising $p(\mathbf{x}, \mathbf{z})$ the other way and isolating $p(\mathbf{z}|\mathbf{x})$:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})}{\int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})\, \mathrm{d}\mathbf{z}}, \tag{3.9}$$

which is Bayes's rule, and where $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is called the posterior distribution, since it characterises the distribution of $\mathbf{z}$ after adapting information about the observations. This gives us a way to compute the most likely value for $\mathbf{z}$ given the data. However, this is not straightforward is most cases.

As an example, take the factor analysis as mentioned in §1.2. In factor analysis, there is a linear relationship between $\mathbf{x}$ and $\mathbf{z}$: $\mathbf{x} = \mathbf{W}\mathbf{z} + \mathbf{b} + \boldsymbol{\epsilon}$, where $\mathbf{W}$ is a weight matrix, $\mathbf{b}$ is bias vector, and $\boldsymbol{\epsilon}$ is some unknown noise. Usually, the noise is modelled as Gaussian noise, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{\Psi})$, and $p_{\boldsymbol{\theta}}(\mathbf{z})$ is a standard normal distribution, $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Thus, the parameters are $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \boldsymbol{\Psi})$, and we can determine these using maximum-likelihood estimation. For factor analysis, equation (3.9) only has an analytical solution when the Gaussian noise is isotropic $\boldsymbol{\Psi} = \sigma^2 \mathbf{I}$. This model is called probabilistic PCA, since it is similar to principal component analysis. (Tipping and Bishop, 1999)

3.1.1 Approximating the marginal likelihood

For factor-analysis models other than probabilistic PCA and for latent-variable models in general, the integral in equation (3.6) is intractable, and we have to resort to other methods to approximate a solution (Tomczak, 2024). One way is to formulate the integral as an expectation of $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with regards to $p_{\boldsymbol{\theta}}(\mathbf{z})$ and approximate the expectation

with a sum over R samples $\mathbf{z}^{(r)}$ from $p_{\boldsymbol{\theta}}(\mathbf{z})$:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x}) &= \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})\, \mathrm{d}\mathbf{z} = \mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})}[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \\ &\approx \frac{1}{R} \sum_{r} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(r)}),\ \mathbf{z}^{(r)} \sim p_{\boldsymbol{\theta}}(\mathbf{z}). \end{aligned} \tag{3.10}$$

This is called *Monte Carlo integration* (Bishop, 2006, Chapter 11), and usually few samples are needed for this approximation. With an approximation of $p_{\boldsymbol{\theta}}(\mathbf{x})$, we can use maximum-likelihood estimation to estimate $\boldsymbol{\theta}$. However, since we use $\mathbf{z}^{(r)} \sim p_{\boldsymbol{\theta}}(\mathbf{z})$ to compute $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(r)})$, we have to take into account where the masses of both distributions are concentrated. For instance, if most samples from $p_{\boldsymbol{\theta}}(\mathbf{z})$ are in a region where $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(r)})$ is low, we will underestimate $p_{\boldsymbol{\theta}}(\mathbf{x})$. We would then need to increase the number of samples to cover enough of $\mathcal{Z}^L$ to get a proper estimation of $p_{\boldsymbol{\theta}}(\mathbf{x})$. Even though $L < N$, L is still usually large, and then the curse of dimensionality can befall us.

The *expectation-maximisation algorithm* (Bishop, 2006, Chapter 9) is another method to determine $\boldsymbol{\theta}$. As the name alludes to, this method consists of two steps: expectation and maximisation, and these steps are alternated in an iterative manner. Instead of a single-observation joint distribution in equation (3.6), we consider the joint distribution over all observations $\mathbf{X}$ and the latent variables $\mathbf{Z}$, $p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z})$. In the expectation step, we then compute the expectation of $\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z})$ with regards to the posterior distribution $p_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{X})$ using parameters $\boldsymbol{\theta}^{(t)}$ at iteration t:

$$\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathrm{E}_{p_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{X})}[\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z})]. \tag{3.11}$$

Next, in the maximisation step, $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ is maximised with regards to $\boldsymbol{\theta}$ to estimate $\boldsymbol{\theta}^{(t+1)}$:

$$\boldsymbol{\theta}^{(t+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}). \tag{3.12}$$

These steps are alternated until convergence, and it can be shown to maximise the (marginal) log-likelihood function (Bishop, 2006, §9.4). The posterior distribution can be estimated using equation (3.9), but when the distributions use complicated functions such as neural networks, it becomes intractable to compute the posterior distribution. Additionally for large data sets, computing the expectation in $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ can also become cumbersome.

Another iterative approach for estimating $\boldsymbol{\theta}$ is *gradient descent* (Bishop, 2006, §5.2.4). In this method, the gradient of $p_{\boldsymbol{\theta}}(\mathbf{x})$ with regards to $\boldsymbol{\theta}$ is used to steps closer to the optimal value. There exists many different optimisation algorithms, but one popular technique is *stochastic gradient descent*. In each iteration, this technique only uses a random subset of observations to compute the gradient. This subset is commonly referred to as a batch or a mini-batch for smaller subset. To differentiate from experiment batches in single-cell transcriptomics, we will use the latter term in this thesis. This mini-batch could be just one observation, and usually it is ensured that all observations have been used to update the gradient before being used again. Because of this mini-batch scheme, stochastic gradient descent can easily scale to large amounts of data.

3.2 Variational autoencoder

The variational autoencoder (VAE; Kingma and Welling, 2014; Rezende et al., 2014) is a latent-variable model parameterised with neural networks, and it uses variational inference (Bishop, 2006, §10.1) to solve the integration in the computation of the marginal likelihood in equation (3.6). This results in an approximate posterior distribution, which stochastically compresses observations to latent representations. The likelihood function can then use the latent representations to stochastically reconstruct the the observations. Finally, the prior distribution is used to constrain the approximate posterior distribution. Because of the structure of compression and reconstruction, it was likened to an autoencoder with the approximate posterior distribution acting as a stochastic encoder and the likelihood function acting as a stochastic decoder. Though, compared to an autoencoder, which learns point estimates of latent representations and the reconstructed observations, the VAE learns probability distributions for both.

I will not derive the variational autoencoder in full here. For an example of that, I refer to Paper A (Appendix A.1), which concerns conditional and hierarchical VAEs for use with single-cell transcriptomics data. Below, I will instead outline some general aspects of VAEs.

3.2.1 Variational inference

To solve the integral in equation (3.6) when the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, variational inference introduces $q_\phi(\mathbf{z})$, which is called a *variational distribution*, and which is parameterised by ϕ. Since $p_\theta(\mathbf{x})$ does not depend on $\mathbf{z}$, the expectation of $p_\theta(\mathbf{x})$ with regards to $q_\phi(\mathbf{z})$ is just $p_\theta(\mathbf{x})$. Therefore, we can introduce $q_\phi(\mathbf{z})$ into the logarithm of the marginal likelihood function in the following way:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) = \mathrm{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x})] &= \int q_\phi(\mathbf{z}) \log p_\theta(\mathbf{x}) \, \mathrm{d}\mathbf{z} \\
&= \int q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \, \mathrm{d}\mathbf{z} \\
&= \int q_\phi(\mathbf{z}) \log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \frac{q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})} \right) \mathrm{d}\mathbf{z} \\
&= \int q_\phi(\mathbf{z}) \log \frac{q_\phi(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \, \mathrm{d}\mathbf{z} + \int q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} \, \mathrm{d}\mathbf{z} \\
&= \mathrm{KL}\Big[q_\phi(\mathbf{z}) \Big\| p_\theta(\mathbf{z}|\mathbf{x}) \Big] + \mathrm{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} \right].
\end{aligned}
\tag{3.13}
$$

Here, the first term in the last expression is the *relative entropy* or the *Kullback–Leibler divergence* (KL divergence; Kullback and Leibler, 1951) between the variational distribution and the posterior distribution. The Kullback–Leibler divergence $\mathrm{KL}[Q \| P]$ between two distributions, Q and P, measures how much more information is needed to model samples of Q when P is used instead, on average (Bishop, 2006, §1.6.1). The KL divergence is asymmetric as can be seen from equation (3.13), and $\mathrm{KL}[Q \| P] \geq 0$ with

equality only when $Q = P$. Because of this, and since the posterior distribution is assumed to be intractable, we can at least get a lower bound, $\mathcal{L}$, on $\log p_\theta(\mathbf{x})$:

$$\begin{aligned}\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}) &= \mathrm{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})}\right] \\ &= \mathrm{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) - \frac{q_\phi(\mathbf{z})}{p_\theta(\mathbf{z})}\right] \\ &= \mathrm{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}\Big[q_\phi(\mathbf{z}) \Big\| p_\theta(\mathbf{z})\Big].\end{aligned} \tag{3.14}$$

$\mathcal{L}(\theta, \phi; \mathbf{x})$ is called the *variational lower bound*. Since the marginal likelihood is also known as the *evidence* (Bishop, 2006, §3.4), $\mathcal{L}(\theta, \phi; \mathbf{x})$ is also referred to as the *evidence lower-bound objective* (ELBO). The first term in the last expression can be interpreted as a negative *reconstruction error* of $\mathbf{x}$, while the other term is another KL divergence.

To learn the variational distribution $q_\phi(\mathbf{z})$, it is traditionally conditioned on each observation, and therefore, we would need to estimate different parameters of this variational or *approximate posterior* for each observation, which can become impractical. Instead, $q_\phi(\mathbf{z})$ can be conditioned on $\mathbf{x}$, so we only need to learn one variational distribution for all observations: $q_\phi(\mathbf{z}|\mathbf{x})$. Conditioning on $\mathbf{x}$ in this way is called amortisation, since the cost of learning of the parameters is amortised over all observations. This kind of variational inference is therefore called *amortised variational inference* (Ganguly et al., 2023), which variational autoencoders generally use.

3.2.2 Tricks to training

By combining equations (3.7) and (3.14) with $q_\phi(\mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{x})$, we can now use maximum-likelihood estimation to determine θ using a variational autoencoder. When the variational lower bound is maximised, the reconstruction error is minimised, the KL divergence is minimised, or both are minimised. Minimising the reconstruction error in the absence of the KL divergence, the approximate posterior is unconstrained and can overfit to the observations. Therefore, the KL divergence can be seen as regularising the variational lower bound.

Since a VAE parameterises its distributions using neural networks, however, the reconstruction error and the KL divergence become intractable like the marginal likelihood (equation 3.6) itself. All this and nothing gained? Not quite. We turn to Monte Carlo integration again, but instead of sampling from the prior $p(\mathbf{z})$, samples now come from the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$. Since the approximate posterior is amortised by $\mathbf{x}$, we are more likely to draw samples of $\mathbf{z}$ where likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is high, thus more easily dodging the curse of dimensionality. Usually, one only needs one sample (Kingma and Welling, 2014)!

With Monte Carlo samples drawn, we then perform maximum-likelihood estimation using gradient descent. However, the gradient of the variational lower bound for ϕ is problematic to compute. This is because the expectation of the reconstruction error as well as the KL divergence are evaluated with regards to $q_\phi(\mathbf{z}|\mathbf{x})$, which is dependent

on $\boldsymbol{\phi}$. Instead, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be reparameterised such that the stochasticity is independent of $\boldsymbol{\phi}$. For $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, $\mathbf{z}$ could instead be obtained using $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$, where $g_{\boldsymbol{\phi}}(\cdot)$ is an appropriate function parameterised by $\boldsymbol{\phi}$, and where $\boldsymbol{\epsilon}$ is an independent random variable. This is called the *reparameterisation trick* (Kingma and Welling, 2014).

In addition to making it possible to train a VAE by using these techniques, they also make it easy to sample from the model as well.

3.2.3 Freedom of choice of distributions

Variational inference does not impose many limitations on the choice of distributions used for the likelihood function or the prior distribution, or the approximate posterior distribution. They only need to be tractable, and the approximate posterior distribution should be a good approximation of the true posterior distribution (Bishop, 2006, §10.1). Because variational autoencoders also model these distributions with neural networks, variational autoencoders are very flexible in their modelling. But with almost unlimited choice, how do we choose and how do we choose well?

Since the latent space is typically chosen to be real valued, and for simplicity, the prior $p_{\boldsymbol{\theta}}(\mathbf{x})$ is usually chosen to be a standard normal distribution: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. However, a problem can occur for the *aggregated approximate posterior*, which is the average approximate posterior over all observations:

$$q_{\boldsymbol{\phi}}(\mathbf{z}) = \frac{1}{M} \sum_{m} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_m). \tag{3.15}$$

If $q_{\boldsymbol{\phi}}(\mathbf{z})$ differs significantly from the prior, we risk sampling $\mathbf{z}$ from the prior with high probability in a region where $q_{\boldsymbol{\phi}}(\mathbf{z})$ is low, or the reverse. Thus, we would end up with many latent representations not matching those inferred from $\mathbf{x}$ and few that do. That would result in poor reconstructions and even poorer samples from the likelihood function. This is known as the *hole problem* (Rezende and Viola, 2018), since mismatched regions look like holes in visualisations of the latent representations. This mismatch between the prior and the approximate posterior is what the KL divergence measures. Since it is minimised during optimisation, the mismatch will also be minimised, and the approximate posterior will be constrained by the prior. The hole problem is more pronounced when the prior is a fixed distribution, and it can be alleviated by using more flexible and learnable priors such as mixture distributions (Tomczak, 2024, §5.4.1). Examples of these are a normal-mixture model (Dilokthanakul et al., 2016) and a variational mixture of posteriors (VampPrior; Tomczak and Welling, 2018). Approaches based on flows can also be used to achieve a flexible prior (Tomczak, 2024, §5.4.1.8).

To match the prior, the approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is also generally modelled using a normal distribution. Again for simplicity, this normal distribution has a diagonal covariance matrix, and the dependence on $\mathbf{x}$ is achieved by letting the mean and variance be functions of $\mathbf{x}$: $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \sigma^2_{\boldsymbol{\phi}}\mathbf{x}\mathbf{I})$. These distribution parameters are then both modelled using neural networks as in equations (3.1). Since

optimising the variational lower bound $\mathcal{L}(\theta, \phi; \mathbf{x})$ does not depend on $\mathbf{z}$, the quality of the latent representation is not necessarily maximised by this optimisation (Huszár, 2017). In fact, if the likelihood function is able to model $\mathbf{x}$ without $\mathbf{z}$, then it would ignore the approximate posterior, which would then be forced to match the prior because of the KL divergence (Rubenstein, 2019). This is called *posterior collapse* (Bowman et al., 2016). One way to avoid this is to use a more flexible distribution for the approximate posterior that more closely match the true posterior. Instead of using a diagonal covariance matrix for a normal distribution, we could use a full covariance matrix, for instance. Flow-based methods have also been developed to transform a diagonal normal distribution to more a complex distribution for use as an approximate posterior (Tomczak and Welling, 2016; van den Berg et al., 2018).

Since the likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$ is used to model $\mathbf{x}$, an appropriate distribution approximating the empirical distribution of $\mathbf{x}$ is chosen. The parameters of this distribution are then modelled as functions of $\mathbf{z}$ using neural networks. If it is not clear what the most appropriate distribution is, one can try out different distributions and choose the one that obtain the highest variational lower bound. One could even perform, for example, factor analysis using different distributions beforehand to select one or multiple candidates.

3.2.4 A variety of variational autoencoders

Because of their popularity, variational autoencoders exist in many different flavours to adapt to the local field of study (Gómez-Bombarelli et al., 2018; Cerri et al., 2019; Oibayashi et al., 2021) and to improve their performance overall (Burda et al., 2015; Higgins et al., 2017; Dai and Wipf, 2019). Some of these variations have already been mentioned in passing above. Here, I highlight a few more methods relevant to my research project. For variants applicably specifically to single-cell gene expression, see §4.1.

One purpose of my research project was to use additional information about the observations to better model the observations. This can, for example, be achieved by conditioning the VAE on this additional information, and a straightforward way to do this is the conditional variational autoencoder (CVAE; Sohn et al., 2015). In CVAE, the likelihood, the prior, and the approximate posterior are all conditioned on another observed variable $\mathbf{c}$, resulting in $p_\theta(\mathbf{x}|\mathbf{c},\mathbf{z})$, $p_\theta(\mathbf{z}|\mathbf{c})$, and $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{c})$. This also makes it possible to generate counterfactual samples by, for example, substituting $\mathbf{c}$ with $\mathbf{c}^*$ when sampling from the model.

Another variational-autoencoder model, Contrastive Mixture of Posteriors (CoMP; Foster et al., 2022), attempts to improve on CVAE by further requiring that the latent variable is independent of the other observed variable: $\mathbf{z} \perp \mathbf{y}$. This is done by adding a misalign penalty term to the variational lower bound. The misalign penalty term forces the aggregated approximate posterior given $\mathbf{c}$ to be equal to the aggregated approximate posterior not given $\mathbf{c}$. Others have proposed using the maximum mean discrepancy (Gretton et al., 2006) instead of the KL divergence (Louizos et al., 2015) or using a pre-trained unconditional VAE (Harvey et al., 2021).

In the pursuit of more interpretable latent representations, one avenue is hierarchical variational autoencoders. The idea being that higher-order latent variables would be able to capture more abstract aspects of the data. With two latent variables $\mathbf{z}_1$ and $\mathbf{z}_2$, a straightforward approach would be to infer them from the bottom-up: $\mathbf{z}_1$ from $\mathbf{x}$ and $\mathbf{z}_2$ from $\mathbf{z}_1$, and then generate the latent variables and the observation from the top-down: $\mathbf{z}_2$ to $\mathbf{z}_1$ to $\mathbf{x}$. However, it can be shown that this can lead to a collapse of the posterior for $\mathbf{z}_2$ (Tomczak, 2024, §5.5.2). Instead, several approaches have been developed that both infer and generate from the top-down (Kingma et al., 2016; Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020; Child, 2021).

3.3 Adversarial autoencoder

The adversarial autoencoder (AAE; Makhzani et al., 2016) is an autoencoder which uses its encoder and decoder to model probability distributions and regularises the distribution of the encoder to match an arbitrary distribution for the latent variable using generative adversarial networks. In this way, it is reminiscent of the VAE, which uses the Kullback–Leibler divergence to match an approximate posterior distribution (the encoder) to a prior distribution (an arbitrary distribution).

Here, I will describe elements of the adversarial autoencoders in general, and I refer to Paper B (Appendix A.2) for a more detailed presentation of AAEs. Paper B concerns adapting AAEs to single-cell gene expression data.

3.3.1 Generative adversarial networks

The idea behind generative adversarial networks (GANs; Goodfellow et al., 2014) is to produce point estimates of $\mathbf{x}$ from $\mathbf{z} \sim p_{\theta}(\mathbf{z})$ instead of estimating the parameters of the likelihood function. With a function $g_{\theta}(\mathbf{z})$ with parameters θ, we could achieve $\mathbf{x} = g_{\theta}(\mathbf{z})$ by expressing the likelihood function using the Dirac delta function (Tomczak, 2024, §8.2.2):

$$p_{\theta}(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - g_{\theta}(\mathbf{z})). \tag{3.16}$$

Modelling a distribution without specifying anything explicit about it is referred to as *implicit modelling*. Plugging the above expression into equation (3.6) for the marginal likelihood, we get:

$$p_{\theta}(\mathbf{x}) = \int \delta(\mathbf{x} - g_{\theta}(\mathbf{z}))\, p_{\theta}(\mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{3.17}$$

This is difficult to evaluate using Monte Carlo integration because of $p_{\theta}(\mathbf{x}|\mathbf{z})$. For maximum-likelihood estimation, we take the logarithm of $p_{\theta}(\mathbf{x}|\mathbf{z})$ for numerical stability, but it is not clear how to compute $\log \delta(\mathbf{x} - g_{\theta}(\mathbf{z}))$ (Tomczak, 2024, §8.2.2).

We could also use the change-of-variables formula to evaluate $p_{\theta}(\mathbf{x})$ (Tomczak, 2024, §4.1):

$$p_{\theta}(\mathbf{x}) = p_{\theta}(g_{\theta}^{-1}(\mathbf{x}))|\mathbf{J}_{g_{\theta}^{-1}}(\mathbf{x})|, \tag{3.18}$$

where $\mathbf{J}_{g_\theta^{-1}}$ is the Jacobian matrix of g_θ^{-1}, and this is the matrix of its first-order derivatives. Computing g_θ^{-1} and its derivatives can, however, be problematic when g_θ is a neural network. One could construct the neural network such that is invertible, which is the idea behind normalising flows (Tomczak, 2024, Chapter 4), but in general, maximum-likelihood estimation would be infeasible.

Instead, GANs solve another optimisation problem. We still have $\mathbf{x} = g_\theta(\mathbf{z})$, where $\mathbf{z} \sim p_\theta(\mathbf{z})$. In this context, g_θ is known as the generator. To evaluate the output of g_θ, we use another function $d_\omega(\mathbf{x})$ called the *discriminator*. This function is intended to determine the probability whether its input comes from the empirical data distribution, $p_\mathcal{D}(\mathbf{x})$. We can, for example, express this probability distribution using a Bernoulli distribution:

$$p_\omega(y\,|\,\mathbf{x}) = \mathrm{B}(y; 1, d_\omega(\mathbf{x})), \quad y = \begin{cases} 0, & \mathbf{x} = g_\theta(\mathbf{z}),\ \mathbf{z} \sim p_\theta(\mathbf{z}), \\ 1, & \mathbf{x} \sim p_\mathcal{D}(\mathbf{x}). \end{cases} \tag{3.19}$$

Other probability distributions can also be used (Mohamed and Lakshminarayanan, 2016).

The expectation of $\log p_\omega(y\,|\,\mathbf{x})$ with regards to the joint distribution $p(\mathbf{x}, y)$ of $(\mathbf{x}, y)$ pairs is then (Murphy, 2022, §19.3.6.2)

$$\begin{aligned} &\mathrm{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\log p_\omega(y\,|\,\mathbf{x})] \\ =&\frac{1}{2}\Big(\mathrm{E}_{p_\theta(\mathbf{z})}[\log p_\omega(y=0\,|\,\mathbf{x}=g_\theta(\mathbf{z}))] + \mathrm{E}_{p_\mathcal{D}(\mathbf{x})}[\log p_\omega(y=1\,|\,\mathbf{x})]\Big) \\ =&\frac{1}{2}\Big(\mathrm{E}_{p_\theta(\mathbf{z})}[\log(1 - d_\omega(g_\theta(\mathbf{z})))] + \mathrm{E}_{p_\mathcal{D}(\mathbf{x})}[\log d_\omega(\mathbf{x})]\Big) \\ =&\mathcal{L}_d(\omega, \theta; \mathbf{x}, \mathbf{z}), \end{aligned} \tag{3.20}$$

which is the objective function for the discriminator, and ω can be approximated using maximum-likelihood estimation while fixing the parameters θ of the generator.

At the same time, we also want the generator to produce samples that look like they come from $p_\mathcal{D}(\mathbf{x})$. We can do this by tricking the discriminator by setting $y = 1$ in $p_\omega(y\,|\,\mathbf{x})$ even though $\mathbf{x}$ is generated from $\mathbf{z}$:

$$\begin{aligned} \mathrm{E}_{p_\theta(\mathbf{z})}[\log p_\omega(y=1\,|\,\mathbf{x}=g_\theta(\mathbf{z}))] &= \mathrm{E}_{p_\theta(\mathbf{z})}[\log(d_\omega(g_\theta(\mathbf{z})))] \\ &= \mathcal{L}_g(\omega, \theta; \mathbf{z}), \end{aligned} \tag{3.21}$$

which is objective function for the generator. This is maximised with regards to θ, while the discriminator parameters ω remain fixed. Usually, the objective functions for the discriminator and for the generator are trained by alternating between them.

The generator could also be trained by minimising $p_\omega(y\,|\,\mathbf{x})$ with $y = 0$. This results in the same expression as the first term of $\mathcal{L}_d(\omega, \theta; \mathbf{x}, \mathbf{z})$. This leads us to framing the training of a GAN as a min–max optimisation problem, which is how it was originally formulated (Goodfellow et al., 2014):

$$\min_\theta \max_\omega \mathcal{L}(\theta, \omega; \mathbf{x}, \mathbf{z}) = \mathrm{E}_{p_\theta(\mathbf{z})}[\log(1 - d_\omega(g_\theta(\mathbf{z})))] + \mathrm{E}_{p_\mathcal{D}(\mathbf{x})}[\log d_\omega(\mathbf{x})], \tag{3.22}$$

where the conflict of the generator and the discriminator is clearly on display. This is called the adversarial loss. Both this formulation and the alternating scheme above, lead to the same solution, but the latter one was proved to be preferable in practice (Goodfellow et al., 2014).

In this way, generative adversarial networks can be used to learn a way to generate samples that look like the observations using another simpler distribution such as a Gaussian noise model. But since we have avoided computing the inverse of $g_{\theta}(\mathbf{x})$,GANs cannot be used to infer $\mathbf{z}$ from $\mathbf{x}$.

3.3.2 Encoder doubling as generator

The encoder of an adversarial autoencoder has two functions (Makhzani et al., 2016). The first is two serve as the encoder of a probabilistic autoencoder, where it models a probability distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$, which can be seen as an approximation of the posterior distribution (equation 3.9). To regularise the AAE, the aggregated approximate posterior (equation 3.15) is coerced to match a prior distribution of $\mathbf{z}$, $p_{\theta}(\mathbf{z})$. This is done using an adversarial network formed by the encoder functioning as generator as well as a separate discriminator. The purpose of the discriminator is to distinguish between samples from $q_{\phi}(\mathbf{z})$ and $p_{\theta}(\mathbf{z})$.

Paper B includes a short discussion of the difference of constraining $q_{\phi}(\mathbf{z}|\mathbf{x})$ in this way compared to how it is constrained in the variational autoencoder.

3.3.3 Three-stage training

In addition to training the autoencoder part of the adversarial autoencoder, the generator and the discriminator also need to be optimised. This results in three objective (loss) functions, which are optimised in turn. Since the encoder is present in all three loss functions, it plays an important role in the AAE. First, the autoencoder loss will force the encoder to produce samples that can be used to reconstruct specific observations. Second, the discriminator loss will help the discriminator to distinguish between the samples from the encoder and samples from the prior distribution of $\mathbf{x}$. Third, the generator loss will push the encoder towards producing samples from the prior distribution to mislead the discriminator.

The solution to equation (3.22) is a saddle point, and convergence of generative adversarial networks can therefore be difficult and take a long time to achieve if at all (Mescheder et al., 2018). Other problems for GANs include *mode collapse* (Salimans et al., 2016), where the generator only produces one kind of observation; *catastrophic forgetting* (Thanh-Tung and Tran, 2020), where the generator switch from mode to mode during training; as well as the discriminator overfitting (Yazici et al., 2020).

3.3.4 Alternative adversarial autoencoders

Several variations of the adversarial autoencoder exist. The authors of the AAE themselves present multiple formulations in different settings and applications: supervised

AAE, semi-supervised AAE, unsupervised clustering with AAE, and dimensionality reduction with AAE. Adversarial autoencoders have also been adapted to different domains (Kadurin et al., 2017; Jang et al., 2022; Lee et al., 2020) and modified (Creswell and Bharath, 2019; Jin et al., 2023). Other alternatives to generative adversarial networks than can encode $\mathbf{z}$ have also been proposed (Donahue et al., 2017; Dumoulin et al., 2016). GANs have also been combined with VAEs (Larsen et al., 2015; Plumerault et al., 2021).

4 Applications

With the success of both single-cell transcriptomics technologies and deep generative modelling in recent years, many models have been applied or specifically developed for the data resulting from these technologies (Brendel et al., 2022). Here, I will outline variational and adversarial approaches with a special focus on those used for mostly representation or perturbation of single-cell transcriptomics data. I will also present the methods I co-developed for my research project, scVAE-C (Paper A; Appendix A.1) and scAAE (Paper B; Appendix A.2), as well as their results.

These methods typically model gene-expression data as raw (read or transcript) counts, or as (standardised) log-transformed and size-factor normalised counts. As likelihood function for raw counts and size-factor normalised counts, a negative binomial distribution or a zero-inflated version is usually used. The idea behind using the zero-inflated distribution is to model the excess zeros introduced by dropout events (§2.2). For log-normalised counts, a normal distribution is normally used.

4.1 Variational ways

Many of the generative approaches are based on variational autoencoders. One of the first method using a VAE was VASC (Wang and Gu, 2018), which employed a standard architecture with the output transformed using zero inflation to model dropout events of log-normalised counts.

Another early method is scVI (Lopez et al., 2018). scVI models raw counts using a zero-inflated negative binomial distribution. It also models the size factor (§2.3.2) as a latent variable, allowing scVI to construct normalised counts from raw counts after it has been trained. The decoder of scVI can additionally be conditioned on covariates to enable integration as well as perturbation of count data. scVI have been proven to be an effective and popular method, and it has been used as basis for a range of other applications. CellAssign (Zhang et al., 2019) and scANVI (Xu et al., 2021) are able to annotate cells using a method conditioned on marker genes or a semi-supervised model, respectively. Solo (Bernstein et al., 2020) can detect doublets by classifying the latent representation. scAR (Sheng et al., 2022) was developed to remove ambient RNA by identifying cell-free droplets. contrastiveVI (Weinberger et al., 2023) uses two encoders for contrastive analysis and perturbation. Finally, MrVI (Boyeau et al., 2024) is

a hierarchical model for learning both a source-aware and a source-free representation for data from multiple sources, for example, patients.

Other methods experimented with modifying the variational lower bound. scvis (Ding et al., 2018) and VAE-SNE (Graving and Couzin, 2020) both added the loss for the t-SNE method to enable better visualisation and clustering of the data. To avoid posterior collapse, one approach (Kimmel, 2020) applied the β-VAE, where the KL divergence is scaled by hyperparameter β, to gene-expression data to improve the interpretability of the latent representations. Two other methods, MMD-VAE (Zhang, 2019) and DiffVAE (Bica et al., 2020), instead replaced the KL divergence by measuring the maximum mean discrepancy (Gretton et al., 2006) between the approximate posterior and the prior distributions. This was also used in one of the proposal (Louizos et al., 2015) for improving the CVAE (Sohn et al., 2015) as mentioned in §3.2.4.

Some methods investigated using different priors and approximate posteriors to better capture the data and achieve more interpretable latent representations. scVAE (Grønbech et al., 2020), MoE-Sim-VAE (Kopf et al., 2021), and DREAM (Jiang et al., 2023) all use normal-mixture distributions for both the prior and the approximate posterior to better model multiple modes in gene-expression data. These modes can, for instance, correspond to different cell types. DREAM also used a zero-inflated transformation like VASC, while scVAE enables using different count distributions, including zero-inflated distributions as well as a constrained Poisson distribution (Salakhutdinov and Hinton, 2009), as likelihood function. MoE-Sim-VAE takes another approach with a decoder for each component of the normal-mixture distributions. It then uses a mixture-of-experts model (Shazeer et al., 2017) to compute an average of the decoders (experts). This way different modes of the data can better be reconstructed. Additionally, it also adds a loss to the variational lower bound that forces better clustering of the latent representation.

Another method, scPhere (Ding and Regev, 2021), maps cells on a hypersphere using a von Mises-–Fisher distribution as approximate posterior and with a uniform prior over the hypersphere. The von Mises-–Fisher distribution can be related to the normal distribution (Tomczak, 2024, §5.4.2.3). scPhere uses these parameterisation to prevent the cells from concentrating together and allow for more meaningful latent representation and better clustering.

To improve the interpretability of their latent representations, alternative approaches use prior knowledge of some kind. SISUA (Trong et al., 2020) is a semi-supervised VAE which in addition to modelling single-cell gene-expression counts is also trained to reconstruct other covariate information to further constrain the latent representation. SSCVA (Gold et al., 2019), VEGA (Seninge et al., 2021), and pmVAE (Gut et al., 2021) all use gene sets for their model structure. SSCVA mask out connections in the neural networks of both the encoder and decoder based on gene sets, while VEGA only does so for the decoder, which is also linear. pmVAE instead uses separate VAEs for each gene set and optimises them together. siVAE (Choi et al., 2023) is a related model that forgoes gene sets by using two VAEs: one to model a gene-expression matrix across cells (as other methods) and another to model it across genes. It then combines the outputs

of the VAEs to reconstruct the original gene-expression matrix.

VAE-based methods have also been developed specifically for integrating cells from different batches or data sets, perturbing gene expression of cells, or both. SCALEX (Xiong et al., 2022) uses a decoder with domain-specific batch normalisation (Chang et al., 2019) to integrate data. scETM (Zhao et al., 2021) uses an embedded topic model (Dieng et al., 2020) as generative model with a batch-correction parameter to learn topic and gene embeddings integrated across batches. scGEN (Lotfollahi et al., 2019) is a standard VAE, which exploits that the learnt latent space is a data manifold (Kingma and Welling, 2014) by computing and using differences between cells from different sources to model perturbations of cells. CoupleVAE (Wu et al., 2024) uses two variational autoencoders whose latent spaces are coupled together also for pertubation modelling.

Transfer VAE (trVAE, Lotfollahi et al., 2020) is a combination of a conditional variational autoencoder and the Variational Fair Autoencoder (Louizos et al., 2015), which uses maximum mean discrepancy to align the approximate posterior for cells from different sources. As an example, this allows trVAE to integrate perturbed and control cells as well as predict the perturbed response for the control cells. scDisInFact (Zhang et al., 2024) is another method for both integration and perturbation. It has a shared encoder as well as separate encoders for each covariate, which one wishes to integrate or perturb. The latent representations are concatenated and put through a shared decoder. CoMP (Foster et al., 2022), which was outlined in §3.2.4, has also been used for integration and perturbation modelling of scRNA-seq data with great success.

4.1.1 scVAE-C

scVAE-C (Paper A; Appendix A.1) is a group of four different VAE models based on scVAE (Grønbech et al., 2020), and they implement a conditional framework in different ways to model single-cell gene-expression profiles. The base model is simply a combination of scVAE and CVAE (Sohn et al., 2015) with an unconditioned prior, because we want the latent variable to be independent of the conditions. Instead of a normal distribution, a second model uses a normal-mixture prior as well as a categorical and a normal distribution for the approximate posterior similar to the GMVAE model of scVAE. A third adds a hierarchical latent structure, and the last one combines both normal-mixture and hierarchical modelling. Like scVAE, the models support different count distributions as well as a normal distribution as likelihood function, so it can model most kinds of gene expression counts. In addition to using standard and zero-inflated versions of the negative binomial distribution, scVAE-C also supports constrained (Salakhutdinov and Hinton, 2009) and free-dispersion variations (Lopez et al., 2018).

We tested the four models in two ways. One way was to determine if the models could integrate gene expression profiles of cells from different sources in their latent space(s) by removing covariate effects. The other way find out if the models could then predict gene expression for counterfactually perturbed cells. For these tasks, we used a scRNA-seq data set consisting of peripheral blood mononuclear cells (PBMCs) from

Lupus patients (Kang et al., 2017) with about half of the cells having been stimulated by IFN-β. As comparison methods, we used scVI, MrVI (only for integration), CVAE, and CoMP. scVI was used, since it had proven uses for integration and perturbation, and because of its popularity, and MrVI was a good match for the hierarchical models. CoMP was chosen, since it had originally been used for the same tasks on the same data set, and it outperformed other methods such as trVAE in many metrics. Lastly, CVAE was used to compare with a standard conditional VAE for general use. Like other models, we used CVAE with an unconditional prior.

We found that the hierarchical scVAE-C models performed best at integrating the stimulated and control cells, and that the other scVAE-C models were not far off in performance. However, the base model did end up mixing the cells more than other scVAE-C models. The comparison methods all achieved worse results than the scVAE-C methods. Regarding perturbation modelling, the nonhierarchical methods performed close to the best comparison method, which was CoMP, using all genes for comparison. However, when only the top-100 differentially expressed genes were used, the base scVAE-C performed the best with the normal-mixture scVAE-C method, scVI, and CoMP in tow. The latter case is more difficult to model because of the more varied expression. Overall, the normal-mixture scVAE-C method was the best model, since it achieved good results at both tasks.

With reference to the issue of posterior collapse of VAEs (§3.2.3), it seems to have played a role in the performances of the scVAE-C models. During development of the normal-mixture scVAE-C models, we had a problem with the component distributions of the prior collapsing to one and the same happened with approximate posterior. We avoided this by factorising the approximate posterior into two distributions. A categorical approximate posterior was compared to the learnable mixture coefficients of the prior, and a normal approximate posterior was compared to component distributions of the prior. This enabled the model to learn both cell-population-specific and global mixture coefficients and components.

As for the hierarchical VAEs, even though these are implemented in the straightforward manner prone to posterior collapse, we avoided this by providing additional information to the second-order latent variable by conditioning it on a covariate. However, reconstructions from these models were worse, which might be because of the additional transformations between latent variables. Finally, in addition to conditioning the the scVAE-C models on their stimulation status, we also tried conditioning the likelihood functions of the scVAE-C models on the library size, instead of using a constrained negative binomial distribution. In this case, we saw that along with cells of different stimulation statuses integrating, cells of different cell types also became somewhat mixed. This could be because the decoder became too powerful, and the approximate posterior distribution collapsing.

4.2 Adversarial approaches

There exists few strictly adversarial autoencoders for single-cell transcriptomics modelling. AAE-SC (Wu et al., 2020) uses an AAE structure combined with an Improved Deep Embedded Clustering algorithm (Guo et al., 2017) to cluster the latent representation during training. IMAAE (Wang et al., 2023) integrates cells from multiple batches by selecting an anchor batch for each set of similar cells and training a single AAE to match the gene expression of the anchor batch for each set. DB-AAE (Ko and Sartorelli, 2024) uses an additional generator to encode the reconstruction of the prior mini-batch (§3.1.1). The discriminator then tries to distinguish between the two encoded latent representations. This results in better imputation and denoising of gene-expression data.

Cell BLAST (Cao et al., 2020) is an AAE, which encodes both a continuous and a categorical latent representation. These are then combined together with batch information for reconstruction. Three discriminators are used to distinguish if the continuous and categorical latent representations follow their respective prior distributions and to determine the batch of the latent representation. Cell BLAST can then project reference data into its learnt latent space and measure similarities between the reference and the original data to annotate cell types.

Other methods combined VAEs and AAEs, so the the latent representation is constrained to follow a prior distribution by both a KL divergence and an adversarial loss. DR-A (Lin et al., 2020) is one such model used for dimensionality reduction. It uses an additional adversarial network with the decoder as generator and a discriminator to distinguish between the original gene expression profiles and the reconstructed ones. scDREAMER (Shree et al., 2023) is based on DR-A and is able to integrate batches of cells. It features a classifier whose aim is to predict the batch of a given latent representation, and the encoder is trained to trick this classifier in addition to deceiving the discriminator. scDREAMER can also support supervised and semi-supervised settings using cell-type annotations.

GANs have also been used in other combination with autoencoders and VAEs. The compositional perturbation autoencoder (Lotfollahi et al., 2021) is an autoencoder, which uses discriminators to predict perturbations and covariate values for the latent representations while the encoder is trained to make that impossible. For reconstruction, the decoder is provided the the real perturbation and covariate values together with the latent representations. MichiGAN (Yu and Welch, 2021) uses a generator to reconstruct the data from the latent representation of a VAE separately from the decoder and a discriminator to differentiate between the generated and the original data. MichiGAN can then use latent-space arithmetics like scGen for perturbation modelling. UNAGI (Zheng et al., 2023) uses a VAE model followed by a GAN to compare the original and reconstructed data. UNAGI alternates between this and two other phases iteratively to be able to predict perturbed gene expression profiles. BatchAIF (Monnier and Cournède, 2024) is a conditional VAE serving as generator for a GAN with a discriminator to determine if its inputs are generated gene expression profiles or the

original ones. The conditional VAE is then conditioned on batch information to integrate cells across batches.

4.2.1 scAAE

scAAE (Paper B; Appendix A.2) is an adversarial autoencoder for embedding single-cell gene-expression data to enable better clustering using community-detection methods popular in single-cell transcriptomics. It addresses to problems common to generative adversarial networks: mode collapse and discriminator overfitting (§3.3.3). To avoid mode collapse, it employs an early-stopping scheme by clustering the data and evaluating the clustering regularly during training. By using a supervised metric, this model can also be used in semi-supervised manner. Even though community-detection clustering can be a costly operation, it is feasible to so by using the GPUs also used to train the model (RAPIDS Development Team, 2023). Discriminator overfitting is overcome by adding noise to the discriminator during training, and multiple noise models are supported. scAAE can also model raw and preprocessed counts by using count or continuous distributions.

scAAE was evaluated on multiple simulated data sets with clustering information as well as multiple PBMC data set of sizes varying from about 3000 cells to about 92 000 cells – all with cell-type annotations available. We then trained scAAE and evaluated its performance by how well clusterings of the latent representation matched the previously annotated grouping (clustering or cell-type annotation). scAAE was trained using both counts that were log-normalised and standardised as well as raw counts, and for the latter case, we tested different count distributions. We also investigated various noise models for the discriminator, and we used both unsupervised and supervised clustering metrics as well as the AAE losses for early stopping. For comparison, we used a baseline method, which uses the preprocessed counts for clustering, for both simulated and real data sets. We also modelled the real data sets using scVI to compare with a variational approach.

We demonstrated that scAAE had almost no problems finding good latent representations for simulated data sets, while it was competitive or better than scVI and the baseline method for the PBMC data sets. We also showed that adding discriminator noise could help with overfitting, and that early stopping using clustering metrics was necessary to find latent representation that could be clustered well. As for data preprocessing, we observed that using the standardised counts work best for the small data sets, while the large data sets were best modelled by the large data sets. When the data sets were modelled using raw counts, we found that zero-inflated distributions achieved the lowest reconstruction loss, but this did not result in latent representation with the best clusterings. Besides that, no clear pattern could be determined.

5 Conclusion

Single-cell transcriptomics involves large amounts of high-dimensional data with complex connections. Many kinds of techniques and analyses have been developed to makes sense of these data and to draw inferences about them for use in, for instance, medical applications. Deep generative modelling cover models that can learn probability distributions of the data themselves and that use many nonlinear transformations to be able to do fit complex data. Such models can scale to high dimensionality and huge data sets, and they have been used on wide variety of data modalities as well as applied to a vast assortment of problem areas. Therefore, it seemed obvious to apply deep generative modelling to single-cell transcriptomics as many have also done Chapter 4.

My research project, which has been detailed in this thesis, concerned itself specifically with how to use deep generative models to represent and perturb single-cell gene expression profiles either from the same source or from multiple sources. The goal of the representations was to both be easily to interpret, but also useful in downstream analysis. As for perturbation modelling, the focus was to predict counterfactual gene expression profiles given a certain perturbation. In this thesis, I have attempted to explain both single-cell transcriptomics and deep generative modelling as they pertain to my research project. I have also presented my research, which consisted of two papers (Appendix A) describing the methods scVAE-C (Paper A) and scAAE (Paper B), and outlined applications of multiple other related approaches. In the following, I will draw some final conclusions for my research project as well as give a few outlooks on where the research is headed.

5.1 Useful models for the right data

When modelling any kind of data, you first have to be sure that you are modelling the appropriate kind of data. Most data contain some level of noise or unwanted features, so to make them easier to model, they are preprocessed in various ways. But how much preprocessing to perform is needed? When preprocessing gene expression counts for single-cell transcriptomics, you have to be careful not throwing away too much important biological variation or the wrong kind of biological variation (see §2.4.1, for instance). For scAAE and scVAE-C, this was also an important factor in modelling gene expressions. There was a difference in preference of preprocessing the counts for scAAE between small and large data sets. Raw counts were preferred for large data sets, and

this might be because they needed more biological variation to find a representation that clustered the best. scVAE-C was only used to model raw counts, but two of the methods used for comparison modelled log-normalised counts. Both of these methods were bested by all scVAE-C models when removing covariate effect from gene expression profiles in their latent representations, while results were more mixed for perturbation modelling. This only serves to illustrate that one should be cautious when selecting the level of preprocessing and make sure that one's model is able to handle it well.

This leads us to choosing an appropriate model, and for deep generative models that means choosing an appropriate probability distribution for the data. For single-cell gene expression data, the negative binomial distribution or a zero-inflated variation is usually used for raw counts, while log-normalised counts are typically modelled using a normal distribution. When modelling raw counts using scAAE, we found no preferable counts distribution, except for zero-inflated distributions for the autoencoder loss. For scVAE-C, the models using negative binomial distributions outperformed the corresponding models using zero inflation as well scVI, which also used a zero-inflated negative binomial distribution. Svensson (2020) also showed that single-cell data are not zero inflated. We also found for scVAE-C that modelling gene expression profiles using a probability distribution designed to constrain the data to the library size worked better than simply conditioning the neural networks of the distribution parameters on that same quantity. Further research should investigate probability distributions specifically designed for single-cell transcriptomics data.

Specifically for latent-variable models and related methods, the structure of the latent variables as well as the selection of the prior and approximate posterior distributions are important. We observed for scVAE-C that the normal-mixture model performed the best overall, and we noted the factorisation of its approximate posterior allowed it to conform to different parts of the prior without collapsing. This enabled the model to have better global (prior) and cell-specific (approximate posterior) understanding of the multimodal data in its latent representation. For the hierarchical scVAE-C methods, we also found that by providing additional information to the second-order latent variable, the approximate posterior did not collapse, which is a problem for some hierarchical VAEs (§3.2.4). However, this did result in worse reconstructions. We did not experiment with the distributions for the latent variables of scAAE, but we did conclude that more flexible distributions might allow for latent representations resulting in improved clustering. Using more adaptable priors and approximated posteriors (§3.2.3) should be researched more for applications in single-cell transcriptomics.

Another decision to make is whether a certain model is at all appropriate for the problem you are attempting to solve. This is relevant for scVAE-C and scAAE, since both are optimising an objective (or multiple) different from the task at hand. scAAE optimises three different losses, which pull it in different direction, while the problem was to find latent representations that clustered better than the original data. This was addressed by separately clustering the embeddings during training, but it would better if this could have been part of the optimisation (Wu et al., 2020). Since community-

detection methods were used, this could prove challenging to implement because of the need to compute gradients of the neighbourhood graphs, but is not impossible to circumvent (Moor et al., 2020). scVAE-C was being used for integration and perturbation modelling, but these task was also not part of its optimisation. Instead, the optimisation objective was a balance between reconstruction of original data and regularisation of the latent variable. However, integration and perturbation modelling could be built into the model or the training scheme. Research have already shown how this could be done (Chapter 4), but achieving good results for these tasks should be examined further.

Another avenue for improvement is speed. Since it takes time to train VAEs and AAEs, for them to compare favourably to other faster dimensionality-reduction methods, the training needs to be expedited or the need for training needs to be alleviated or eradicated. Using fewer model parameters would make training faster, but we would need to ensure that we are not sacrificing too much flexibility of the models. As for the need for training, transfer learning uses pre-trained models to perform inference by only training using a few new observations (few-shot learning) or not training at all (zero-shot learning).

5.2 Towards modelling of multiomics

Other deep generative models have also been used to model single-cell gene expressions. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), which are related to variational autoencoders (Tomczak, 2024, §5.5.3), have been used for generation and perturbation modelling (Luo et al., 2024) and modelling of gene regulatory network (Zhu and Slonim, 2024). Language models and foundation models have also been used for multiple tasks each (Yang et al., 2022; Shen et al., 2023; Chen et al., 2024; Hao et al., 2024).

In single-cell transcriptomics, cells are usually disassociated from the tissue, but recently technologies to quantify transcripts *in situ* have appeared. This is called spatial transcriptomics. Variational and adversarial approaches have been developed to take advantage of this kind of data (Xu et al., 2023; Fang et al., 2023). A score-based generative model, SpatialScope (Wan et al., 2023), have also been proposed for integrating scRNA-seq data and spatial transcriptomics data.

However, transcriptomics only tell part of the story. One reason is that messenger RNA, which is typically what is sequenced in scRNA-seq, is only a proxy of the final gene product for protein-coding genes. This is because of post-transcriptional modification of mRNA before it is translated into proteins (Nachtergaele and He, 2017) and post-translation modifications of proteins after that (Carlberg and Molnar, 2016, Chapter 1). Thus, gaining information about protein expression levels and combining them with transcription levels, would provide a better representation of gene expression. CITE-seq (Stoeckius et al., 2017) and REAP-seq (Peterson et al., 2017) are two technologies for this purpose, and they can sequence and quantify surface proteins together with scRNA-seq. To get an even fuller representation, scATAC-seq (Buenrostro

et al., 2015; Cusanovich et al., 2015) provide quantification about chromatin accessibility, which plays a part in regulating gene expression. Other modalities of single-cell data or other *omics* data are also available, leading to the study of multiomics, where they are studied together, and this is becoming a popular area of research (Hasin et al., 2017; Subramanian et al., 2020).

To model multiple modalities, VAEs using product-of-experts approximate posteriors (Wu and Goodman, 2018) or mixture-of-experts approximate posteriors (Shi et al., 2019) have, for example, been proposed. Such models have been used to model single-cell multiomics data (Minoura et al., 2021; Zuo and Chen, 2021; Drost et al., 2024) in addition to other VAE approaches (Allesøe et al., 2023; Kalafut et al., 2023). Generative adversarial networks have also been used (Xu et al., 2022), especially in combination with standard autoencoders (Khan et al., 2023; Singh et al., 2023) or variational autoencoders (Yang et al., 2024; Cao et al., 2024). scGPT (Cui et al., 2024), which is a foundation model for single-cell data, has also recently been made available, and it can be adapted to different tasks. Other methods have also been proposed (Brombacher et al., 2022; Ballard et al., 2024).

Multimodal deep generative modelling of multiomics have clearly seen a lot of advancement, and training foundation models seems to be the direction we are heading in.

A Papers

A.1 Paper A: scVAE-C

Title

scVAE-C: Latent-representation covariate removal and counterfactual expression profiles for single cells using conditional and hierarchical variational autoencoders

Authors

Christopher Heje Grønbech, Rachael DeVries and Eljas Roellin, Ole Winther

Contributions

Christopher and Ole designed the study together. Christopher and Ole developed the methods with inputs from Rachael and Eljas, and Christopher implemented them. Rachael and Eljas set up the analyses with inputs from Christopher and Ole, and they implemented the analysis pipeline along with Christopher. Rachael and Christopher performed the analyses using scVAE-C, and Eljas performed the analyses using the comparison methods. Christopher wrote both the main text and supplementary materials with inputs from Ole, Rachael, and Eljas, except for the commentary of the dot plot (§3.2.3) and configuration searches for the comparison methods (Supplementary Text S6.5), which was both written by Eljas. All authors have also edited the report.

scVAE-C: Latent-representation covariate removal and counterfactual expression profiles for single cells using conditional and hierarchical variational autoencoders

Christopher Heje Grønbech[1,2], Rachael DeVries[2], Eljas Roellin[3], and Ole Winther[2,4,5]

[1]Qlucore, Sweden
[2]Bioinformatics Centre, Department of Biology, University of Copenhagen, Denmark
[3]Department of Biosystems Science and Engineering, ETH Zurich, Switzerland
[4]Centre for Genomic Medicine, Rigshospitalet, Copenhagen University Hospital, Denmark
[5]Section for Cognitive Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark

Last typeset on 22 January 2025

Abstract

Single-cell RNA-sequencing data sets often include covariates for clinical information, technical details, and quality metrics. These are usually used to preprocess the data by removing their effects before analysis. We present four methods that use covariates when modelling single-cell RNA-sequencing data by adding a conditional framework to a variational autoencoder using mixture models, hierarchical structure, both, or neither. By conditioning on a specific covariate, we show that these methods, especially the hierarchical ones, can regress out its effects from their compressed representations while still capturing biological variation. We also demonstrate that the methods, especially the nonhierarchical ones, can predict counterfactual gene expression profiles by changing the covariate value when generating data from their compressed representations. Additionally, we show that all methods outperform similar approaches at removing covariate effects and that the nonhierarchical methods generate counterfactual predictions at the same level or better as similar approaches. Overall, we find that our nonhierarchical mixture method achieves competitive performance in both kinds of analyses. Our methods are implemented in Python using the TensorFlow machine-learning library, and they are freely available as an update to the scVAE tool at `https://github.com/scvae/scvae`.

1 Introduction

Single-cell RNA-sequencing (scRNA-seq) enable analysis of cells at a higher resolution than previous approaches, and it has been increasing in use in recent years with around 2000 data sets published to date (Svensson et al., 2020, this database[1] is still being updated). scRNA-seq data sets consist of the

[1]Available at `https://nxn.se/single-cell-studies/gui`.

gene expression counts of typically thousands of cells or more across ten or more cell types (Svensson et al., 2020). The number of genes is usually around 20 000 in data sets of human or mouse cells, but not all genes are expressed in every cell (Andrews et al., 2020). This leads scRNA-seq data sets to be large, high-dimensional, and sparse in addition to being complex because of the varied expression across cell types. Several methods and tools are available to process and analyse these data sets. Most notably among these are Seurat (Hao et al., 2023) and Scanpy (Wolf et al., 2018), which both use various machine-learning methods. Deep generative methods, which use many-layered neural networks to model data in a probabilistic manner, have also been shown to be effective at modelling scRNA-seq data sets. Examples of such methods include scvis (Ding et al., 2018), VASC (Wang and Gu, 2018), scVI (Lopez et al., 2018), and scVAE (Grønbech et al., 2020), which all use variational autoencoders (VAE, Kingma and Welling, 2014; Rezende et al., 2014). VAEs model data by compressing them into lower-dimensional (latent) representations that can more easily be analysed or further processed as well as used to reconstruct the original data. For scRNA-seq data, these analyses could be for biological insights such as cell-type identification or gene expression profiling under alternative conditions.

In addition to gene expression, scRNA-seq data sets usually include information about cell types, technical details such experiment batches for each cell, and clinical information such as age, sex, and ethnicity for each donor (Gagnon et al., 2022). Quality metrics such as the sum of the gene expression counts for each cell (the library size) and fraction of mitochondrial RNA of each cell can also be computed from the data (Wolf et al., 2018). Many methods such as Seurat use this information as covariates when preprocessing data to remove batch effects or simply regress them out. scVI can use batch information or other covariates when generating the gene expression counts of cell to remove covariate effects in a similar manner. Including covariates in a scVI model can also be used to predict counterfactual expression profiles by replacing the covariate information. This could for instance predict how the gene expressions of a healthy cell would look if it was in a diseased state instead. Methods for counterfactual prediction include scGen (Lotfollahi et al., 2019), transfer VAE (trVAE, Lotfollahi et al., 2020), Compositional Perturbation Autoencoder (CPA, Lotfollahi et al., 2021), and Contrastive Mixture of Posteriors (CoMP, Foster et al., 2022).

Here, we present scVAE-C, which includes four modifications to the scVAE method, that can model gene expression counts for single cells using covariates in a conditional variational-autoencoder framework (CVAE, Sohn et al., 2015). The four methods consist of a base method, which is a combination of scVAE and CVAE, and a hierarchical method as well as normal-mixture versions of the two. We include hierarchical methods to be able to condition on covariates separately, and we use the normal-mixture methods, since they have been proven to better model inherent clusters of scRNA-seq gene-expression count data (Grønbech et al., 2020). For each method, the gene expression counts of a cell are conditioned on covariates both when compressing (or encoding) the cell into a latent representation (the inference process) and when reconstructing the gene expression counts of the cell by decoding this representation (the generative process). The methods support conditioning on multiple covariates at the same time, and they support both numerical and categorical covariates.

We demonstrate that the scVAE-C methods, particularly the hierarchical ones, can remove covariate effects from the latent representations, and how this affects the interpretability of these latent representations. We also show that the methods, particularly the nonhierarchical ones, are able to counterfactually predict gene expression profiles by changing the covariate values for cells when decoding the latent representations. We compare our methods with CVAE Sohn et al., 2015, CoMP Foster et al., 2022, scVI (Lopez et al., 2018), and MrVI (Boyeau et al., 2022). CVAE is a general implementation not specifically dedicated to biological data, while CoMP extends CVAE by adding a misalignment penalty term to the variational lower bound. The intention of this penalty is to enable CoMP to remove batch effects by ensuring independence of the latent representation and the covariates. Due to their more general approaches, CVAE and CoMP are trained with normalised scRNA-seq gene-expression counts, whereas scVAE-C and the other comparison methods are trained with raw gene-expression counts, which have been shown to improve performance in some cases (Eraslan et al., 2018; Lopez et al., 2018; Grønbech et al., 2020). MrVI is a hierarchical method based on scVI. It uses two latent representations in a hierarchy to remove different covariates from each latent representa-

tion. Like scVI, MrVI also only conditions on covariates when generating gene expression counts of cells, whereas scVAE-C and the other comparison methods also conditions on covariates during inference. We find that all scVAE-C methods outperform the comparison methods at latent-representation covariate removal, and that the nonhierarchical scVAE-C methods achieve competitive results with best comparison method for counterfactual prediction. The base mixture scVAE-C method attains the best overall performance.

Our contributions consists of four conditional VAE methods for modelling scRNA-seq gene expression profiles and covariates, analyses of their performances in removing covariate effects from their latent representations and in counterfactual prediction of expression profiles, as well as software implementations of the methods and the pipeline used for preprocessing and performing experiments and analyses. Compared to scVAE, scVAE-C adds a flexible conditional framework as well as an optionally hierarchical latent structure to both the standard and the normal-mixture VAEs of the scVAE method. None of the methods to which we compare scVAE-C supports the same level of functionality and customisability for conditioning on covariates.

2 Methods and materials

There are several ways to add a conditional framework to the scVAE method. We present the basis of our approach in this section along with a few additions using normal-mixture models and hierarchy. We also describe our analyses, the data we use, our experiment setup, and our software implementation.

2.1 Conditional generative process

The first approach is based on combining the standard scVAE method and the CVAE method (Sohn et al., 2015). We let $\mathbf{X}$ be an $M \times N$-matrix representing a data set of the gene expression counts of M cells and N genes. We also let $\mathbf{x}_m$ be a vector denoting the mth individual cell of the data set with each component x_{mn} corresponding to the gene expression count for the nth gene. The covariates for this cell are represented by a vector $\mathbf{c}_m$ (§2.6). The gene expression counts and covariates for each cell form our observations for that cell. For ease of notation in the following, we use $\mathbf{x}$ and $\mathbf{c}$ to symbolise the gene expression counts and covariates, respectively, of any individual cell in the data set.

We assume $\mathbf{x}$ to come from a probability distribution conditioned on $\mathbf{c}$: $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{c}, \boldsymbol{\theta}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c})$, which is parameterised by $\boldsymbol{\theta}$. By doing this, we let some of the variation in $\mathbf{x}$ be explained directly by $\mathbf{c}$ and let $\boldsymbol{\theta}$ model the rest. Since $N \gg 1$, we next introduce a stochastic variable $\mathbf{z}$ as a vector with fewer dimensions than $\mathbf{x}$ to represent a compressed representation of $\mathbf{x}$. We do this by also conditioning the probability distribution of $\mathbf{x}$ on $\mathbf{z}$. Modelling $\mathbf{x}$ in this way is called a latent-variable model, where $\mathbf{z}$ is the latent variable. We then end up with the following joint distribution for $\mathbf{x}$ and $\mathbf{z}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} \mid \mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z}), \tag{1}$$

where $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})$ is the likelihood function for $\mathbf{x}$ and $\mathbf{z}$ and $p_{\boldsymbol{\theta}}(\mathbf{z})$ is the prior distribution of $\mathbf{z}$. We do not condition $\mathbf{z}$ on $\mathbf{c}$, since we want the latent representation to not be dependent on the covariates. We do, however, list conditional priors for all scVAE-C methods in Supplementary Note S7.

By marginalising out $\mathbf{z}$ from the joint distribution, we get an expression for $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c})$:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) = \int p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{2}$$

Assuming that cells are independent and identically distributed, we can determine $\boldsymbol{\theta}$ by multiplying together the marginalised likelihood functions for all cells and use maximum-likelihood estimation:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} p_{\boldsymbol{\theta}}(\mathbf{x}_m \mid \mathbf{c}) = \arg\max_{\boldsymbol{\theta}} \sum_{m=1}^{M} \log p_{\boldsymbol{\theta}}(\mathbf{x}_m \mid \mathbf{c}), \tag{3}$$

where we take the logarithm of the marginalised likelihood functions for numeric stability.

Since the aim is to model raw count data, each gene expression count x_n is assumed to to follow its own negative binomial distribution (Oshlack et al., 2010): $x_n \sim \mathrm{NB}(\rho_n, \omega_n)$ (Supplementary Note S1). Assuming that the counts are also independent and identically distributed, the likelihood function is the product of these negative binomial distributions. By further assuming the prior distribution of $\mathbf{z}$ to be a unit multivariate normal distribution, this gives us the generative process for $\mathbf{x}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c}) = \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c}), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})), \tag{4a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}). \tag{4b}$$

Here, the distribution parameters $r_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})$ and $p_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})$ are given by $\mathbf{z}$ and $\mathbf{c}$ and modelled using neural networks as in Grønbech et al. (2020). We also investigate zero-inflated, constrained, and free-dispersion versions of the negative binomial distribution (Supplementary Note S1). The model graph for the generative process is shown in Figure 1a with solid arrows.

2.2 Conditional inference process

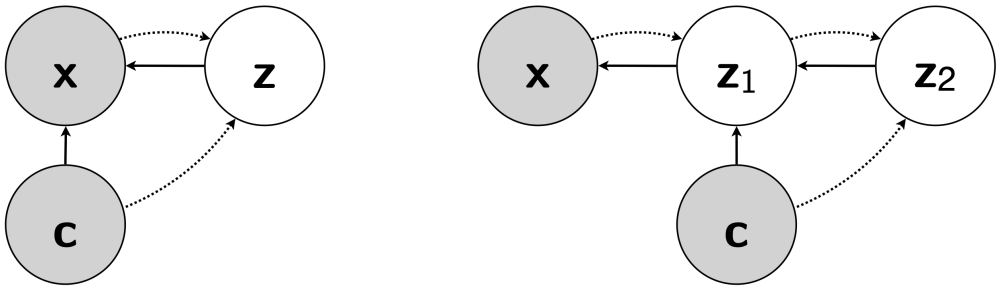
The marginalised likelihood in equation (2) becomes intractable as the number of dimensions of $\mathbf{z}$ increases. Usually this is solved by using the expectation-maximisation algorithm, but this requires the posterior distribution, which is intractable for more complicated functions like neural networks. Instead a variational Bayes optimisation method is used, by introducing an approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})$ parameterised by $\boldsymbol{\phi}$ and decomposing the marginal log-likelihood (taking the logarithm of equation 2) into two terms:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{c}) = \mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c}) \,\|\, p_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})] + \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} \,|\, \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}\right]. \tag{5}$$

Here, the first term is Kullback–Leibler (KL) divergence between the approximate and the true posterior distribution. The KL divergence of two distributions is only zero, when the distributions are equal, and positive otherwise, so the second term becomes a lower bound of $\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{c})$. Since we cannot evaluate the first term because of the intractable posterior, we focus on optimising the second one. This is called the variational lower bound, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c})$, and it can be further decomposed into two terms:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) = \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} \,|\, \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}\right] = \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c})] - \mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{z})], \tag{6}$$

where the first term is the expected negative reconstruction error. Monte Carlo integration is used to integrate over $\mathbf{z}$ by drawing R samples from $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})$, computing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c})$ for each sample and



(a) scVAE-CB. **(b)** scVAE-CH.

Figure 1. Model graphs of the generative and inference processes for two of the scVAE-C methods. Circles represent observed (grey) and latent (white) variables, and arrows indicate the directions of the generative (solid) and inference (dashed) processes.

averaging over the samples. The approximate posterior distribution is assumed to be a multivariate normal distribution to better resemble the prior distribution of $\mathbf{z}$, yielding the inference process for $\mathbf{z}$:

$$q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})\mathbf{I}). \tag{7}$$

Here, the distribution parameters are given by $\mathbf{x}$ and $\mathbf{c}$ and also modelled using neural networks. By letting the inference process be conditioned on $\mathbf{c}$, cells with similar covariates will more easily be recognised as being similar by the process. A reparameterisation trick is used to sample $\mathbf{z}$ for the Monte Carlo integration to allow backpropagation of the neural networks during optimisation (Kingma and Welling, 2014; Rezende et al., 2014). The model graph for the inference process is shown in Figure 1a with dashed arrows. Together, the conditional generative and inference processes form our *base scVAE-C* (scVAE-CB) method.

2.3 Reconstruction and counterfactual prediction

By computing the expected value of the likelihood function with respect to the approximate posterior distribution, we can reconstruct $\mathbf{x}$ as $\tilde{\mathbf{x}}$ using the scVAE-CB method:

$$\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}) = \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})}[\mathbf{x}]\Big] = \int \sum_{\mathbf{x}'} \mathbf{x}' \, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}, \mathbf{c}) \, q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \, \mathrm{d}\mathbf{z}. \tag{8}$$

This is effectively replacing the prior (equation 4b) in the generative process with the inference process (equation 7). By conditioning on $\mathbf{c}$, the inference process essentially removes the effect of the covariates from the cells, whereas the remaining generative process adds it back.

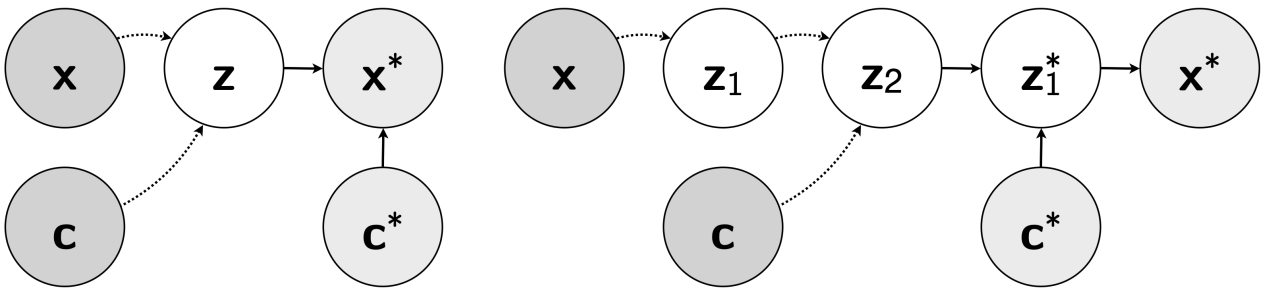
We can use this approach to produce counterfactual reconstructions $\mathbf{x}^*$. This counterfactual prediction is done by passing factual covariate values $\mathbf{c}$ to the inference process, and then passing the counterfactual covariate values $\mathbf{c}^*$ to the remaining generative process:

$$\mathbf{x}^*(\mathbf{x}, \mathbf{c}) = \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c}^*)}[\mathbf{x}]\Big] = \int \sum_{\mathbf{x}'} \mathbf{x}' \, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}, \mathbf{c}^*) \, q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \, \mathrm{d}\mathbf{z}. \tag{9}$$

This is illustrated in Figure 2a.

2.4 Conditional hierarchical scVAE

Another approach to conditional modelling with scVAE uses two stochastic latent variables, $\mathbf{z}_1$ and $\mathbf{z}_2$, arranged in hierarchical latent layers: $\mathbf{x}$ is encoded by $\mathbf{z}_1$, and $\mathbf{z}_1$ is encoded by $\mathbf{z}_2$. This method is our *hierarchical scVAE-C* (scVAE-CH) method, and it is similar to the M1+M2 method in Kingma



(a) scVAE-CB. **(b)** scVAE-CH.

Figure 2. Model graphs of counterfactual prediction for two of the scVAE-C methods. Circles represent observed (grey), latent (white), and counterfactual (light grey) variables. Arrows indicate the directions of the inference process (dashed) and the counterfactual generative process (solid).

et al. (2014). But whereas that method was trained in two separate steps, this method is trained in one. It uses the following generative process:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c})\, p(\mathbf{z}_2), \tag{10a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1) = \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}_1), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}_1)), \tag{10b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}), \boldsymbol{\sigma}^2_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c})\mathbf{I}), \tag{10c}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2) = \mathcal{N}(\mathbf{z}_2; \mathbf{0}, \mathbf{I}). \tag{10d}$$

By letting only $\mathbf{z}_1$ be conditioned on $\mathbf{c}$, we separate the generation of $\mathbf{x}$ in first latent layer from the addition of any covariate effects in the second latent layer. The inference process of this method has a similar separation:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}) = q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}), \tag{11a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c})\mathbf{I}), \tag{11b}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(\mathbf{x})\mathbf{I}). \tag{11c}$$

Here, the removal of covariate effects and the compression of $\mathbf{x}$ are separated in the two latent layers by conditioning only $\mathbf{z}_2$ on $\mathbf{c}$. All distribution parameters are modelled using neural networks as in the scVAE-CB method. The model graphs for both the generative and inference processes are shown in Figure 1b.

The variational lower bound for the method can computed as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) = \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c})}\right]. \tag{12}$$

This expression can be decomposed into reconstruction error and KL divergences as shown in Supplementary Note S2, which also lists derivations for reconstruction and counterfactual prediction. The counterfactual prediction is also illustrated in Figure 2b.

2.5 Conditional mixture scVAE methods and latent-layer conditions

Both the base and hierarchical conditional scVAE methods use normal distributions for the prior and approximate posterior distributions. To better model scRNA-seq data with several differing cell types, we can use normal-mixture models instead for these distributions as demonstrated in Grønbech et al. (2020), specifically, and in Kingma et al. (2014) and Dilokthanakul et al. (2016) in general. Supplementary Notes S3 and S4 describe the base *mixture scVAE-C* (scVAE-CM) and the *hierarchical mixture scVAE-C* (scVAE-CHM) methods, respectively.

The conditional hierarchical scVAE methods can also be further extended to support conditions specific to each latent layer. This enables each latent layer to focus on different covariates. For instance, one latent layer could focus on technical variation, and the other could focus on clinical information. The model setup for these extensions to the scVAE-CH and the scVAE-CHM methods are shown in Supplementary Notes S2.1 and S4.1, respectively.

Additionally, the model graphs of the generative and inference processes for all methods are illustrated in Supplementary Figure S1, and the model graphs of the counterfactual prediction for all methods are shown in Supplementary Figure S2.

2.6 Conditioning on covariates

Covariates in scRNA-seq data are usually represented as categorical (for example, experiment batches, donors, disease states) or numeric (for example, age, treatment dosage, fraction of mitochondrial RNA). scVAE-C supports conditioning on both of these. Numeric, both discrete and continuous, covariates are used as-is, while categorical covariates are one-hot encoded (Harris and Harris, 2013).

One-hot encoding is a way to represent unordered categories as binary vectors each with a length equal to the number of categories. Each component of the vectors then corresponds to a specific category, and it is 1 when the vector represents that category and 0 otherwise. When a variable, $\mathbf{x}$ or $\mathbf{z}$, is conditioned on one or multiple other variables including covariates, these are all concatenated into one vector.

2.7 Analyses

We compare scVAE-C methods directly with each other as well as other likelihood-based methods evaluated on the raw gene expression counts using the variational lower bound. However, this only measures how well each method can model the data. We also want to investigate how well the scVAE-C methods can remove covariate effects from latent representations as well as their performance in generating counterfactual expression profiles. This also allows us to compare scVAE-C with a wider range of methods. For details about hyperparameters used for the below analyses, see Supplementary Note S5.1.

2.7.1 Analysing latent-representation covariate removal

The local inverse Simpson's index (LISI, Korsunsky et al., 2019) is used to assess categorical covariate removal from the latent representations and the resulting integration of cells. LISI measures how well the neighbourhood around a single cell is mixed according to some categorical covariate such as experiment batch information. More precisely, it is the effective number of categories in the neighbourhood of the cell. Thus, the minimum is 1 and signifies no mixing, whereas the maximum equals the number of categories for the covariate in question and means that cells of different categories have mixed completely. We compute LISIs with respect to cell type as well as the categorical covariate used for conditioning. Following Korsunsky et al. (2019), we call the former cell-type LISI (cLISI) and the latter integration LISI (iLISI). To compute a single value for a population of cells (for example, a data set), we take the median of the population. Since one of the aims of the methods is to remove covariate effects, we want the median integration LISI to achieve the maximum value. However, cell types should not become more mixed than they already are, so we want the median cell-type LISI to be less than for the data set itself. As a baseline for the data set, we perform a truncated singular value decomposition (TSVD) of the normalised count data (see §2.8), and compute the LISI values using Euclidean distances between the decomposed representations, $\mathbf{h}$, of the cells. For latent representations, Euclidean distances are computed between the mean values of latent variable in question. As an example, we use $\boldsymbol{\mu}_\phi(\mathbf{x}, \mathbf{c})$ (see equation 7) for the scVAE-CB method.

We also visualise the latent representations using the UMAP dimensionality-reduction method (McInnes et al., 2020) to visually inspect the covariate removal and its effects on other relevant covariates. Again, we use the mean values of the latent variable in question for each latent representation to compute Euclidean distances.

2.7.2 Analysing counterfactual expression profiles

To evaluate the counterfactual prediction of the gene expression profiles (see equation 9 for the scVAE-CB method), we follow Lotfollahi et al. (2019) in performing regression analyses as well as visualising factual and counterfactual expression profiles. We show the analyses at the resolution of individual, previously published, expert-annotated cell types within a data set.

For the regression analyses, we first compute the mean gene expression levels of each gene across cells of the same cell type for both the data set as well as for the counterfactual predictions. For the data set, we only use cells of the same kind as the counterfactual predictions. We then compare the counterfactual mean gene expression levels of each gene to the factual ones for each cell type. We do this by performing a linear least-squares regression analysis using all genes and report the coefficient of determination, R^2, which is the square of the Pearson's correlation coefficient (see Supplementary Note S5.2 for calculation of statistics). Additionally, we compute the top-100 differentially expressed

genes (DEGs) using the Wilcoxon rank-sum test and perform the regression analyses again with only these genes.

We also compare the mean gene expression levels of specific marker genes for groups of counterfactually generated cells to real cells visually using a dot plot. This shows both the mean gene expression for each combination of cells and marker genes as well as the fraction of cells that expresses that gene in that combination. Since reconstructed counts from the methods are real numbers (Supplementary Note S1), we only consider a gene expressed in a cell if its expression count is greater than 0.5.

For these analyses, raw gene expression counts both from the data set as well as predicted by the methods are normalised and log-transformed (see §2.8).

2.8 Data set

We apply our methods to a scRNA-seq data set of peripheral blood mononuclear cells (PBMCs) from Lupus patients (Kang et al., 2017). This data set has been previously used in scRNA-seq benchmarking on probabilistic deep learning models (Lotfollahi et al., 2019, 2020). The data set contains gene expression counts for 14 053 genes and 13 576 cells divided between eight cell types. A little over half (7217) of the cells have been IFN-β stimulated and the rest (6359) are control cells. Concerning cell types, 32 % of the cells are CD14 monocytes, 31 % are CD4 T cells, and the rest of the cells are divided among the remaining six cell types. Stimulated cells also represent about half of the cells of each cell type, except for B, CD4 T, and CD8 T cells for which 57 % are stimulated cells (Supplementary Figure S3).

We filter the genes and only keep the 2000 most varying genes (Stuart et al., 2019). For the remaining genes and for all cells, we compute quality metrics such as the library size. Supplementary Figure S4 shows distributions of the library sizes for stimulated and control cells of each cell type. For each cell type, the library sizes of stimulated cells are on average higher than the control cells, especially for monocytes and dendritic cells. The library sizes for these cells are also significantly higher than for the other cells. For analyses and methods that cannot use the raw gene expression counts, before filtering the genes, we normalise all cells to have the same library size (the median library size of cells before normalisation) and log-transform the normalised counts using $\log(1 + x)$. Normalisation is performed, since the library size varies dependent how well the RNA content of each cell was quantified (Luecken and Theis, 2019), and we are not interested in this variation. Additionally, for computing LISI values for the data set, we follow Korsunsky et al. (2019) in also standardising the normalised counts to have zero mean and unit variance as well as performing L^2 normalisation followed by truncated SVD factorisation with 30 components. A UMAP embedding of this representation is shown in Supplementary Figure S5, which demonstrate the separation of stimulated and control cells for each cell type as well as the differences in library size. For the counterfactual-prediction analyses, we use the marker genes for IFN-β response reported in Butler et al. (2018).

2.9 Experiment setup

For the Kang data, we aim to integrate stimulated and control cells in a latent representation to demonstrate the covariate removal of the stimulation status. To assess the generation of counterfactual expression profiles, we predict expression profiles of stimulated cells for each cell type. For our purposes, the data set is shuffled and split into the three subsets: 60 % for a training set used for training models, 20 % for a validation set used for evaluation during training and early stopping, and the remaining 20 % for a test set used for evaluation after training. For counterfactual prediction, separate models are trained for each cell type by withholding the stimulated cells of that cell type from the training and validation sets. Stimulated cells of other types remain in the training and validation sets to allow the models to learn the relationship between stimulation and gene expression. The model for each cell type is then used to predict counterfactual expression profiles for control cells of that cell type if they had been stimulated. Results are averaged across all cell types.

All scVAE-C models use the same neural-network architecture, and we use feed-forward neural networks, so the order of genes and conditions does not matter. We do this for simplicity, and since

we do not truly know how the genes depend on each other. For each analysis and method, we perform configuration searches to test the different likelihood functions, conditioning on library size, as well as using dropout regularisation (Hinton et al., 2012). All models are trained five times with different random seeds resulting in five replicates for each model, and we report the median as well as the interquartile range. For more details about model setup, training procedure, configuration searches, and model selection, see Supplementary Note S6 as well as Supplementary Tables S1–S2.

We compare scVAE-C with four other methods: CVAE, CoMP, scVI, and MrVI (the last method is only used for latent-representation covariate removal). We perform a small configuration search for CVAE and CoMP and use the default configuration for scVI and MrVI, and they are also trained five times each. These are all VAE methods, and since scVI and MrVI use raw count data like scVAE-C, we can compare their variational lower bounds directly. CVAE and CoMP use normalised count data, so we can compare their variational lower bound to each other. We can compare all methods for the remaining analyses, however, because of some implementation details, we can only compare results for the test set. For consistency, we use the same notation for these methods as for the scVAE-C methods. See Supplementary Note S6.5 for more details about the comparison methods.

2.10 Software implementations

The scVAE-C methods have been implemented in Python as an update to the scVAE Python package, which is freely available online[2]. This implementation mainly uses the machine-learning software libraries TensorFlow (Abadi et al., 2015) as well as TensorFlow Probability (Dillon et al., 2017). All used software libraries are listed in Supplementary Table S3. In addition to the methods presented above, the software implementation also supports conditioning priors on covariates (Supplementary Note S7).

The preprocessing, experiment setup, and analyses have also been implemented in Python, but using the software libraries Scanpy (Wolf et al., 2018), seaborn (Waskom, 2021), and others (Supplementary Table S4). The source code for this implementation is also freely available available online[3].

3 Results

We have used the four scVAE-C methods to model the scRNA-seq data set described in §2.8. We have assessed how well the methods can remove the effects of a particular covariate from their latent representations and integrate the cells. We also examined how good the methods are at counterfactually predicting expression profiles for the removed covariate.

3.1 Latent-representation covariate removal

For each scVAE-C method, we first performed a configuration search, and then compared the optimal models to each other as well as the comparison methods both quantitively and qualitatively.

3.1.1 Likelihood functions, conditioned covariates, and dropout regularisation

The configuration search for each scVAE-C method consisted of testing different likelihood functions, two sets of conditioned covariates, as well as using dropout regularisation or not (Supplementary Figures S6–S9). In general, constrained negative binomial distributions performed the best – sometimes helped by having free dispersion, and zero-inflated distributions performed the worst. Conditioning on both the stimulation status and library size performed as well as or better than using the constrained distributions with stimulation status and condition, except for the median cell-type LISI. These models did indeed integrate stimulated and control cells, but they also mixed cells of different cell types significantly more than models only conditioned on stimulation status. Therefore, we excluded these models from further analyses in this section. Dropout regularisation improved the variational lower

[2] `https://github.com/scvae/scvae`
[3] `https://github.com/scvae/scvae-pipeline`

bound for all models, however, results were more mixed for the LISI medians: Overall, LISI medians improved, but the median integration LISI was generally worse for the scVAE-CB and scVAE-CM methods, and the median cell-type LISI was worse for the scVAE-CHM method.

3.1.2 scVAE-C methods integrated stimulated and control cells better than comparison

The results of the optimal models for each scVAE-C method are summarised in Table 1 along with the comparison methods. Supplementary Figure S10 also visualises these results, and Supplementary Table S5 summarises the configurations for the optimal scVAE-C models. The variational lower bounds of the scVAE-C methods were all close to each other, but on average the lower bounds became better as the complexity of the method increased. They were also all better than the variational lower bound for scVI, but worse than the one for MrVI, which is significantly higher.

According to the median integration LISI (see also Supplementary Figure S10b), the $\mathbf{z}_2$ latent representations of the hierarchical scVAE-C methods achieved the best integration of stimulated and control cells with close to the maximum value of 2. The other two scVAE-C methods had the next best integration. Additionally, the $\mathbf{z}_1$ latent representations of the hierarchical scVAE-C methods and MrVI all achieved a median iLISI equal to or close to 1, indicating that the learned latent representations separated cells with different stimulation status. These results were also significantly lower than the baseline truncated SVD method. The above is also evident from Figure 3, which shows the distributions of cell-level integration LISI values for each method. We see that the hierarchical scVAE-C methods were able to integrate all cells in their $\mathbf{z}_2$ representation to a higher degree than any of the other methods. Simultaneously, these methods were also able to separate most of the stimulated and control cells in their $\mathbf{z}_1$ representation better than any method (including the baseline method) other than MrVI in its $\mathbf{z}_1$ representation. The scVI, MrVI (for its $\mathbf{z}_2$ representation), CVAE, and CoMP methods had more difficulty integrating stimulated and control cells with scVI and CVAE leaving some cells completely separate. This is mainly because they were worse at mixing different stimulation states for CD14 monocytes and CD4 T cells, which are the most populous in the data

Table 1. Variational lower bounds and LISI medians for latent-representation covariate removal. The median and the interquartile range across replicate models of each method are reported. Arrows indicate the optimal direction, and the optimal results (based on the interquartile range) have been highlighted. For each metric, optimal models for the scVAE-C methods were selected using the metric itself, except for the median cell-type LISI for which the median integration LISI was used (Supplementary Note S6.4).

Method	$\mathcal{L}$ ↑	Representation	$\widehat{\text{iLISI}}$ ↑	$\widehat{\text{cLISI}}$ ↓
RAW COUNTS				
scVAE-CB	$-407.6\ ^{(-407.62-}_{-407.49)}$	$\mathbf{z}$	$1.869\ ^{(1.8661-}_{1.8697)}$	$1.071\ ^{(1.0702-}_{1.0712)}$
scVAE-CM	$-406.8\ ^{(-407.14-}_{-406.83)}$	$\mathbf{z}$	$1.876\ ^{(1.8748-}_{1.8811)}$	$\mathbf{1.010}\ ^{(1.0077-}_{1.0118)}$
scVAE-CH	$-406.4\ ^{(-406.49-}_{-406.38)}$	$\mathbf{z}_1$	$1.000\ ^{(1.0000-}_{1.0001)}$	$\mathbf{1.007}\ ^{(1.0054-}_{1.0081)}$
		$\mathbf{z}_2$	$1.925\ ^{(1.9117-}_{1.9282)}$	$1.013\ ^{(1.0122-}_{1.0196)}$
scVAE-CHM	$-406.3\ ^{(-406.42-}_{-406.33)}$	$\mathbf{z}_1$	$1.003\ ^{(1.0026-}_{1.0040)}$	$1.009\ ^{(1.0088-}_{1.0092)}$
		$\mathbf{z}_2$	$\mathbf{1.930}\ ^{(1.9288-}_{1.9328)}$	$\mathbf{1.009}\ ^{(1.0047-}_{1.0116)}$
scVI	$-415.5\ ^{(-415.86-}_{-415.46)}$	$\mathbf{z}$	$1.709\ ^{(1.6934-}_{1.7131)}$	$1.038\ ^{(1.0364-}_{1.0388)}$
MrVI	$\mathbf{-398.3}\ ^{(-398.49-}_{-398.01)}$	$\mathbf{z}_1$	$1.000\ ^{(1.0000-}_{1.0000)}$	$1.010\ ^{(1.0087-}_{1.0103)}$
		$\mathbf{z}_2$	$1.784\ ^{(1.7708-}_{1.7908)}$	$1.010\ ^{(1.0087-}_{1.0103)}$
NORMALISED COUNTS				
CVAE	$-861.0\ ^{(-862.15-}_{-860.91)}$	$\mathbf{z}$	$1.730\ ^{(1.7285-}_{1.7409)}$	$1.061\ ^{(1.0611-}_{1.0616)}$
CoMP	$\mathbf{-778.4}\ ^{(-779.67-}_{-777.40)}$	$\mathbf{z}$	$1.775\ ^{(1.7708-}_{1.7748)}$	$1.077\ ^{(1.0687-}_{1.0791)}$
TSVD		$\mathbf{h}$	1.031	1.022

set, than for most other cells (Supplementary Figure S11).

We also looked at the cell-type LISI for the optimal scVAE-C models (selected using the median iLISI) as well as for the comparison methods. All representations of the three hierarchical methods as well as the scVAE-CM method attained comparable cLISI medians lower than the baseline method, meaning that cell types were less mixed than for this baseline. The other methods, which only have one latent layer, mixed cells of different cell types in their latent representation somewhat more and worse than the baseline method – with the scVAE-CB and CoMP methods being the worst ones. This can also be seen by looking at the cell-level cLISI distributions for each method in Supplementary Figure S12 with distributions for single-latent-layer methods being less concentrated close to 1.

3.1.3 UMAP embeddings of latent representation hints at capture of biological variation

In addition to quantifying the results, we also inspected the latent representations visually. Figure 4 shows UMAP embeddings of the latent representations that were conditioned on stimulation status for the optimal scVAE-C method, scVAE-CHM, as well as scVI, MrVI, and CoMP coloured by stimulation status, cell type, and library size. UMAP embeddings of all latent representations for all scVAE-C methods and comparison methods are shown in Supplementary Figures S13–S15. All latent representations have clustered well according to cell type (less so for the scVAE-CB method, see Supplementary Figure S14), which suggests that biological variation is captured in the latent representations. This was also expressed by the cell-type LISI. We do observe that the clusters, especially the cluster of T and NK cells, in the $\mathbf{z}_2$ latent representations of the hierarchical scVAE-C methods are more loosely connected than in the latent representations for the other methods. We saw this same pattern in the $\mathbf{z}_2$ latent representations of all hierarchical scVAE-C models using dropout regularisation. These same models also had higher integration LISI medians than the corresponding models without dropout regularisation, but mostly worse cell-type LISI medians. This might imply that hierarchical scVAE-C methods reveal more substructure of each cluster than the other methods, and that this substructure is not necessarily congruent with the annotated cell types. But it could also simply be an artefact of the UMAP embedding arising from noise from two layers of dropout regularisation.

As for the stimulation status, the latent representations conditioned on that covariate for the scVAE-C methods all look to have been well integrated (Supplementary Figure S13), which was also evident from the integration LISI. We can also visually confirm that the comparison methods had difficulty integrating especially monocytes of different stimulation status, leading to a worse median iLISI than for the scVAE-C methods. Additionally, the $\mathbf{z}_1$ latent representations of the hierarchical

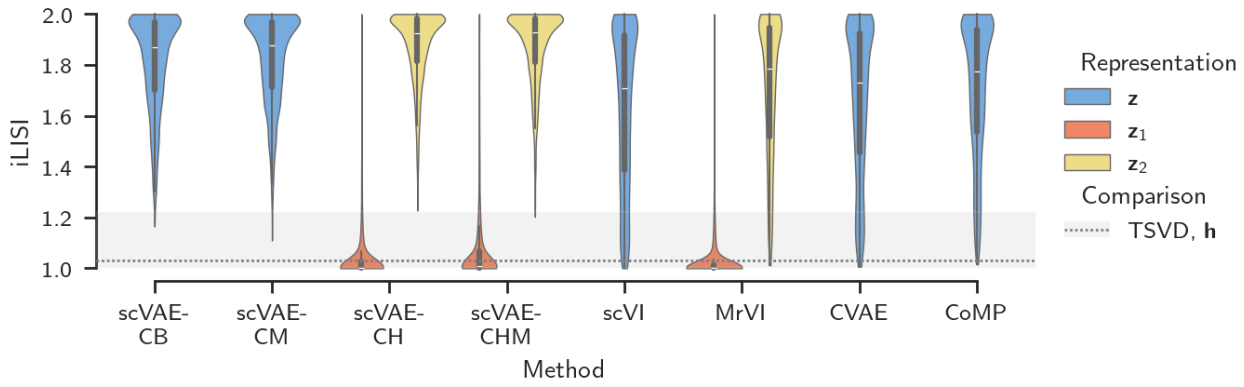


Figure 3. Distributions of cell-level integration LISIs for latent-representation covariate removal. For the latent representation(s) of each method, the kernel density estimate, median, and interquartile range are shown for the median-performing replicate model. Optimal models were used for the scVAE-C methods. The kernel density estimates were normalised to display with the same area. The median and the interquartile range for the decomposed representation of the baseline TSVD method are also shown.

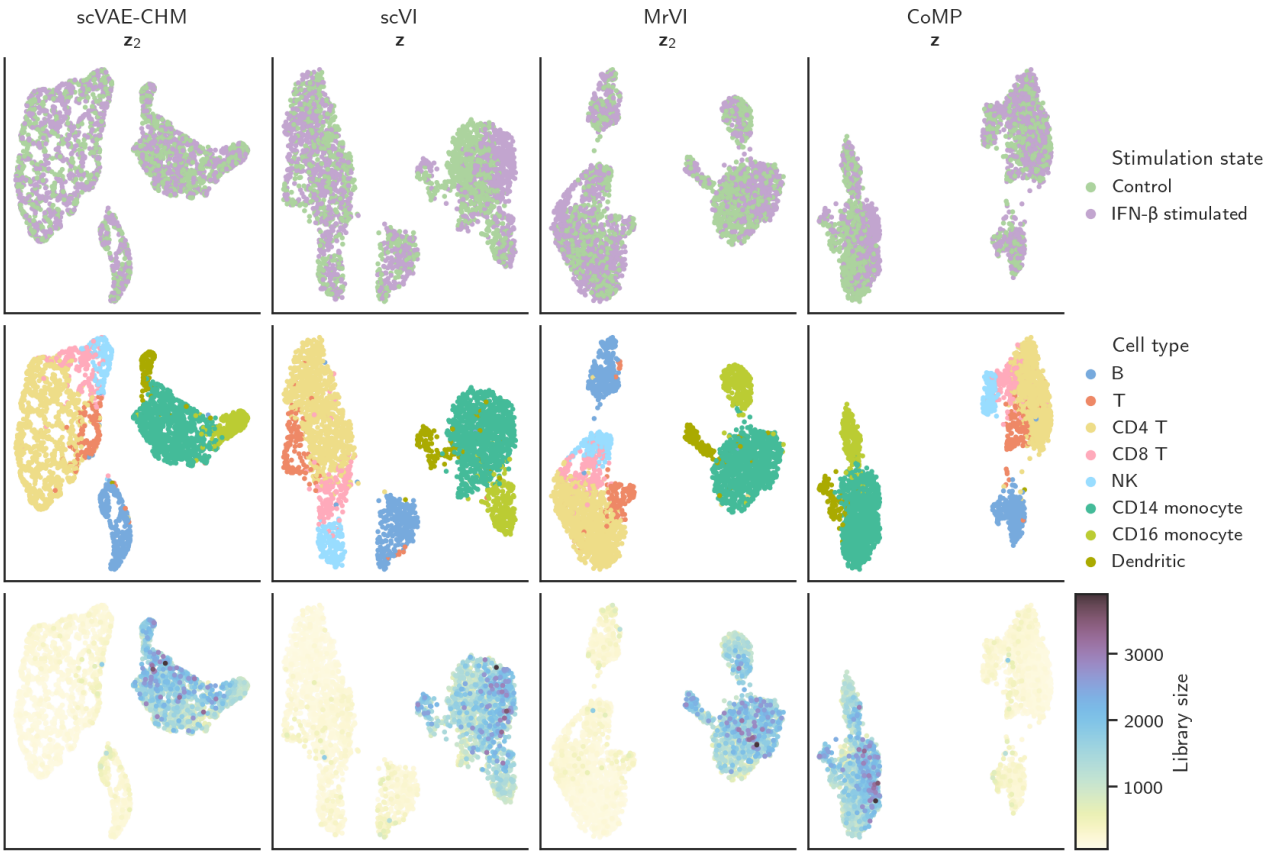


Figure 4. UMAP embeddings of latent representations for latent-representation covariate removal. Embeddings are shown for the median-performing replicate model of each method, and they are coloured by stimulation state, cell type, and library size. The optimal model for the scVAE-CHM method was selected using the median iLISI. For all latent representations for all methods, see Supplementary Figures S13–S15.

methods show the clear separation of stimulated and control cells as indicated by their iLISI values. Looking at the library-size colour-coding, we can see that cells with higher library sizes have clustered together. This is unsurprising, since these cells are monocytes and dendritic cells, which were already shown to cluster closely together. Comparing with the stimulation-status colour coding, we can also see the difference in library size between stimulated and control cells for these cells evident in the data set (Supplementary Figure S4). Cells with lower library sizes form two superclusters, which is likely one of the reason why conditioning on library size mixes cell types to a high degree.

3.1.4 Mostly small correlations between variational lower bound and local inverse Simpson's indices

Finally, we examined correlations between the variational lower bound and either of the LISI medians for the scVAE-C methods (Supplementary Figures S16). There seems to be a slightly positive correlation between the variational lower bound and the median integration LISI for the hierarchical scVAE-C methods, meaning that optimising the variational lower bound could lead to better integration for these methods. For the non-hierarchical scVAE-C methods, there does not seem to be any correlation. Regarding the median cell-type LISI, this seems to be negatively correlated with the variational lower bound for the scVAE-CB method and less so for its hierarchical equivalent, meaning that optimising the variational lower bound results in less mixing of cell types, in general. There does not seem to be any correlation for the scVAE-CM method. Do mind that we excluded the models conditioned on both stimulation status and library size, since they achieved significantly worse cLISI medians.

3.2 Counterfactual expression profiles

We again began by performing a configuration search for each scVAE-C method, followed by quantitive and qualitative comparisons of the optimal models and the comparison methods.

3.2.1 Likelihood functions, conditioned covariates, and dropout regularisation

The same configuration search as in §3.1 was performed for this analysis (Supplementary Figures S17–S19). For the mean variational lower bound (Supplementary Figures S17), we saw the same patterns for likelihood functions, condition sets, and dropout regularisation as for the variational lower bound in the analysis of the latent-representation covariate removal. This shows us that excluding some stimulated cells does not change the overall performance of the methods. Results are more mixed for the mean coefficients of determination using either all genes, $\overline{R^2_{\text{all}}}$, or the top-100 DEGs, $\overline{R^2_{\text{all}}}$ (Supplementary Figures S18–S19). Models with different likelihood functions performed comparably for all methods, but to a lesser extent for the hierarchical methods. Also conditioning on library size resulted in better mean coefficients of determination for the scVAE-CH and scVAE-CHM methods, but worse values for their nonhierarchical versions. Dropout regularisation improved $\overline{R^2}$ for the scVAE-CM method, but made the values for the other methods worse, especially for the hierarchical ones.

3.2.2 Nonhierarchical scVAE-C methods captured covariate effects competitively with best comparator

Table 2 shows the results for the optimal model for each scVAE-C method as well as for the comparison methods. The same results are also visualised in Supplementary Figure S20, and the configurations for the optimal scVAE-C models are summarised in Supplementary Table S6. We first note that the optimal mean variational lower bound for each scVAE-C method is similar to the corresponding method in §3.1, which is further evidence that leaving out some stimulated cells does not have a large effect on model performance. As for $\overline{R^2_{\text{all}}}$, CoMP performs the best on average, but CVAE, scVAE-CB, and scVAE-CM are all close to it in performance. scVAE-CB achieves the best $\overline{R^2_{\text{top}}}$ with scVAE-CM, scVI, and CoMP being close to its performance. For both coefficients of determination, the hierarchical scVAE-C methods perform the worst and also vary the most between replicates. We also found that these methods performed worse at reconstructing the true mean expression of stimulated and control cells using the models trained on all of the test set (Supplementary Figures S21–S22).

We also examined the variational lower bound and the regression analysis for each cell type that was used to withhold stimulated cells (Supplementary Figure S23). The relative performances in

Table 2. Mean variational lower bounds and mean coefficients of determination for counterfactual prediction. The median and the interquartile range across replicate models of method are reported. Arrows indicate the optimal direction, and the optimal results (based on the interquartile range) have been highlighted. For each metric, optimal models for the scVAE-C methods were selected using each metric itself (Supplementary Note S6.4).

Method	$\overline{\mathcal{L}}$ ↑	$\overline{R^2_{\text{all}}}$ ↑	$\overline{R^2_{\text{top}}}$ ↑
RAW COUNTS			
scVAE-CB	−407.6 (−407.88–−407.46)	0.918 (0.9163–0.9209)	**0.892** (0.8895–0.8928)
scVAE-CM	−406.6 (−406.61–−406.58)	0.918 (0.9165–0.9190)	0.883 (0.8812–0.8840)
scVAE-CH	−406.4 (−406.45–−406.38)	0.851 (0.8443–0.8552)	0.819 (0.8179–0.8268)
scVAE-CHM	**−406.1** (−406.17–−406.10)	0.863 (0.8523–0.8694)	0.826 (0.8144–0.8369)
scVI	−417.8 (−417.86–−417.73)	0.893 (0.8924–0.8936)	0.883 (0.8817–0.8838)
NORMALISED COUNTS			
CVAE	−847.6 (−847.69–−847.40)	0.919 (0.9176–0.9196)	0.862 (0.8622–0.8646)
CoMP	**−761.4** (−761.74–−761.37)	**0.925** (0.9244–0.9290)	0.880 (0.8759–0.8853)

variational lower bound across cell types are almost reversed for methods using normalised count data (CVAE and CoMP) and methods applied to raw count data (remaining methods). The cell types with the largest difference in performance are also the cell types with the largest shares of cells in the data set. So CVAE and CoMP are better at modelling the test set (with all stimulated cells) without having seen stimulated cells of these cell types in comparison to withholding them for other cell types. For the other methods, it is the reverse.

The distributions of the coefficients of determination across cell types used to withhold stimulated cells for each method are shown in Figure 5. R^2 was generally high for all cell types and methods (less so for the hierarchical scVAE-C methods) with low standard deviation, which means that the methods were able to capture the covariate effects. On average, R^2_{all} was also higher than R^2_{top} except in some cases. The coefficient of determination being higher for all genes than for the top-100 DEGs is to be expected, since R^2_{all} includes many genes not affected by IFN-β, and therefore their expressions are similar for stimulated and control cells. R^2_{top} were significantly lower than R^2_{all} for monocytes for most methods and dendritic cells in general. These cell types all have higher library sizes than the other cell types, and cells with high library sizes are also somewhat less represented in the data set (Supplementary Figures S3–S4).

The individual regression analyses for each cell type and method have been plotted in Supplementary Figures S24–S30. The top-five upregulated DEGs for each cell type have been highlighted, and all methods mostly predict expressions for these correctly. As alluded to by the low standard deviation of the coefficients of determination, the regression plots also show few outliers. The slopes of the linear regression for most cell types and methods are also close to 1, except for the CD14 monocytes and the CD4 T cells for the CVAE and CoMP methods Both methods predict counterfactual expression levels significantly less than for truly stimulated CD14 monocytes and somewhat less than for truly stimulated CD4 T cells. These cell types have the largest shares of cells (Supplementary Figure S3), and the methods both use normalised count data. So even though a linear regression did fit for these cell types and methods, the methods cannot predict the mean expression correctly, likely because some of the signal for each cell type have been removed – partly by withholding stimulated cells and partly by normalising the expression counts.

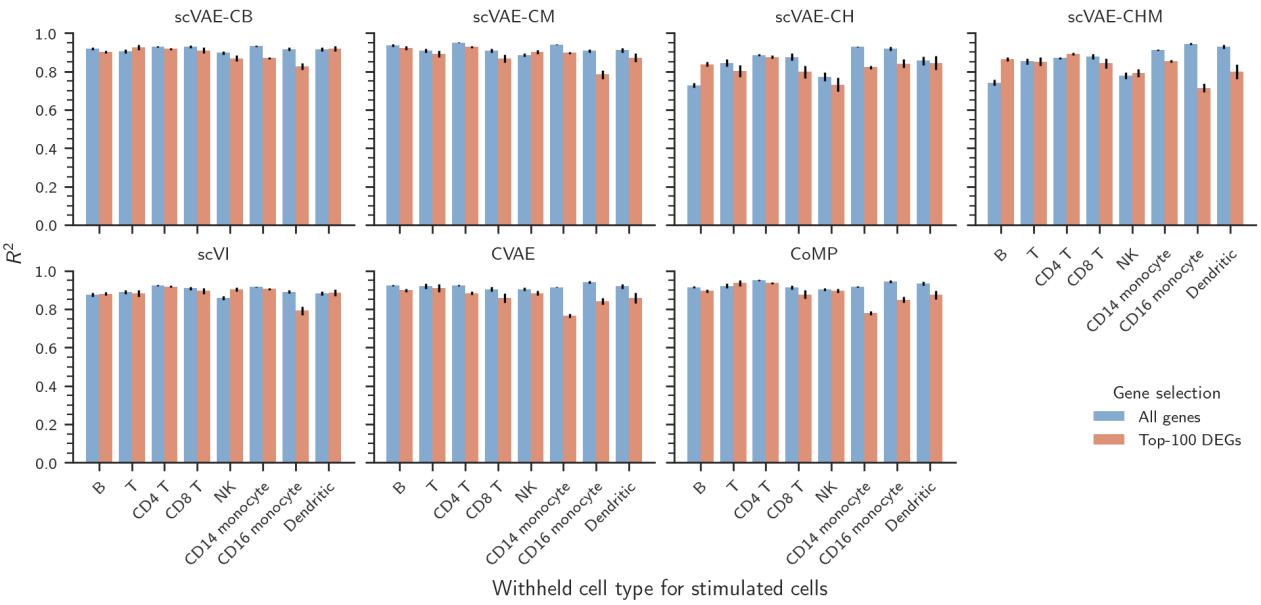


Figure 5. Distributions of coefficients of determination across withheld cell types for counterfactual prediction. For each cell type and method, the mean and the standard deviation are shown for the median-performing replicate model. Optimal models for scVAE-C methods were selected using the overall mean coefficient of determination for each gene selection.

3.2.3 Counterfactual expression profiles for key marker genes match factual expression profiles

For key marker genes, we also show a visual comparison of the counterfactual expression profiles to the factual ones using dot plots. In Figure 6, we show a dot plot of the mean expression profiles for factually unstimulated (control) cells, counterfactually stimulated cells, and factually stimulated cells of different cell types across a selection of key marker genes. The counterfactually stimulated cells were predicted from control cells using the best scVAE-C method, scVAE-CB. For the interferon-independent cell-type markers CD79A (B cells), CD3D (T cells), CCL5 (CD8 T and NK cells), GNLY (NK cells), FCGR3A (monocytes), S100A9 (CD14 monocytes), and HLA-DQA1 (dendritic and B cells), there are no visually evident differences in mean expression level and expressing cell fractions between the control and the stimulated cells. The model's inferred expression of these marker genes in stimulated cells matches the factual observations for all cell types. For the interferon-response genes, there is a strong expression in stimulated cells, which is absent in unstimulated cells. The model correctly predicts elevated levels of ISG15, IFI6, and IFIT1 across all cell types upon interferon stimulation. The model further predicts a strong increase in CXCL10, CXCL11, APOBEC3A in monocytes and dendritic cells. A more nuanced response of the DEFB1 gene in monocytes is also inferred as observed. While CCl8 is overestimated in CD16 monocytes and dendritic cells, it is correctly inferred to be elevated for CD14 monocytes, and correctly inferred to be not elevated in all other cell types. Dot plots for the other methods show similar tendencies and can be seen in Supplementary Figures S31–S37 together with the one for scVAE-CB.

3.2.4 Almost no correlations between variational lower bound and coefficients of determination

Lastly, we looked at correlations between mean coefficients of determination and mean variational lower bound for the scVAE-C methods (Supplementary Figures S38). Using either all genes or the top-100 DEGs, there seems to be a very slight negative correlation. This means that the mean variational lower bound is not useful for selecting models for counterfactual prediction, but if used, it might lead to worse results.

4 Discussion

In analysing latent-representation covariate removal and counterfactual prediction of gene expression profiles, it is clear that different approaches are needed to achieve the best performance in each analysis. We details these differences below.

4.1 Likelihood functions, conditioned covariates, and dropout regularisation

Using one of the constrained negative binomial distributions as likelihood function, conditional scVAE models were able to attain the best variational lower bound and the best LISI medians in most cases. However, there was no clear preference of likelihood function for counterfactual prediction – most negative binomial distributions performed comparably.

Conditioning on library size together with stimulation status, models performed similarly to or better than models using a constrained negative binomial distributions according to the variational lower bound, the integration LISI median, and the mean coefficients of determination for counterfactual prediction, except in a few cases. From this, it seems like we were able to achieve the same effects of a more complicated likelihood function by simply conditioning the likelihood function instead. However, the additional conditioning lead to cell-type LISI medians significantly higher than only conditioning on the stimulation status. Since the likelihood function is only evaluated on how well it can fit the actual scRNA-seq count data, it makes sense that reconstruction or counterfactual prediction perform the same for the two approaches, while their latent representations differ: Stimulated and control cells may have mixed, but so have cells of different cell types when conditioning on library size.

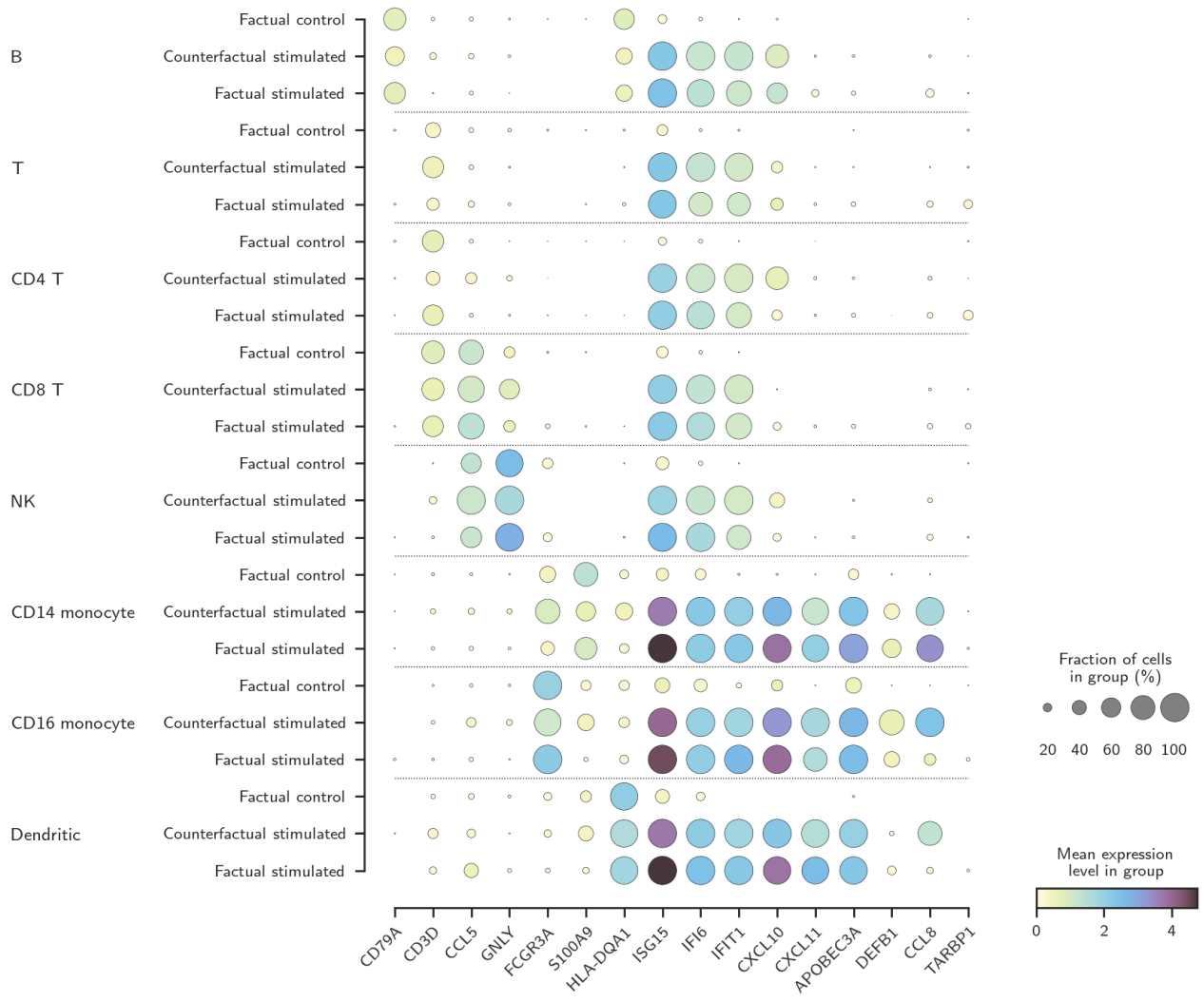


Figure 6. Dot plot of mean gene expressions for counterfactual prediction using the scVAE-CB method. For each cell type and marker gene, the mean gene expressions across factual control, counterfactually stimulated, and factually stimulated cells are shown. The counterfactually stimulated cells's expression profiles were generated by the median-performing replicate model. The optimal scVAE-CB model was selected by the mean coefficient of determination using all genes.

Dropout regularisation improved variational lower bounds, had mostly little effect on latent-representation covariate removal, and lead to mostly comparable counterfactual predictions for non-hierarchical scVAE-C methods and worse ones for the hierarchical versions.

4.2 scVAE-CB best at counterfactual prediction for most varying genes

The base scVAE-C method, scVAE-CB, integrated stimulated and control cells in the latent representation significantly better than the comparison methods. Mixing these cells, however, also led to cell types being mixed to a higher degree than any of the other methods except the CoMP method. The cell-type mixing was reduced for scVAE-CB when using dropout regularisation, but this lead to worse integration of stimulated and control cells.

Conversely, scVAE-CB and CoMP performed the best overall at counterfactually predicting the mean gene expression profiles of stimulated cells for each cell type. For all genes, CoMP performed the best with CVAE, scVAE-CB, and scVAE-CM close to its result. CVAE is similar in structure to scVAE-CB, so a similar performance is not unexpected. However, when using only the top-100 DEGs,

scVAE-CB achieved the best results with scVAE-CM, scVI, and CoMP being close in performance. As mentioned in §3.2, it is harder to predict expression levels for these genes, since they vary more between stimulated and control cells by design. So it is notable that scVAE-CB attained better counterfactual predictions for the most differentially expressed genes than the more complex CoMP method. Both CVAE and CoMP use normalised count data, so the reason why they perform worse for the top-100 DEGs might be because some of the variation is lost in the normalisation. CoMP's enforcement of independence of latent representation and covariates may still let it better encode the variation that is left than CVAE.

scVI is also similar in structure to scVAE-CB like CVAE, but scVI also uses raw count data like scVAE-CB, so one would expect it to perform similarly. However, since scVI by default does not condition on covariates during inference and only during generation, this explains its poor performance in latent-representation covariate removal. Additionally, the default choice of a free-dispersion zero-inflated negative binomial distribution, which was not among the best for scVAE-CB, explains why it did not perform as well as scVAE-CB for counterfactual prediction. The variational lower bound for scVI was also worse than to the corresponding scVAE-CB model in both analyses.

4.3 scVAE-CH and scVAE-CHM best at removing covariate effects in their latent representation

The hierarchical scVAE-C methods, scVAE-CH and scVAE-CHM, were able to simultaneously almost completely separate stimulated and control cells in its first latent representation and closely integrate them in its second latent representation. The separation of the two kind of cells was better than the baseline TVSD method and only bested by MrVI (in its first latent representation), and the integration was the best of the methods together with its normal-mixture version. At the same time, cells of different cell types were also kept separate in both latent representations, even more separate than the baseline method. Several other methods also achieved this, but they did not integrate stimulated and control cells as well.

In contrast, scVAE-CH and scVAE-CHM were the worst at counterfactual prediction and also varied the most between replicates. We also confirmed that the hierarchical scVAE-C methods were worse at reconstruction than the non-hierarchical ones. This is likely because of the two layers of the generative process in both methods resulting in more variation both in expression, but also between replicates. Both kinds of variation are then exacerbated when the methods are used to predict counterfactual expression, and variation in the neural networks introduced by dropout regularisation only worsens this further. These methods perform counterfactual prediction the best by also conditioning the first latent layer on library size and not by using a constrained negative binomial distribution like the nonhierarchical versions. This could be because the conditioning on stimulation state and library size is separate to each latent layer, so the first latent-layer transformation functions as a normalisation step similar to the preprocessing of the raw count data. MrVI most closely resembles scVAE-CH, and even though it performs better than scVI at integrating stimulated and control cells while keep cell types separate, it still worse at the integration than all scVAE-C methods. This is likely because the stimulation status of each cell is only used for generation and not inference like the scVI method.

4.4 scVAE-CM is the best overall method

The base mixture scVAE-C methods, scVAE-CM, perform comparably to its non-mixture version except in a few instances. scVAE-CM was significantly better than scVAE-CB at keeping cell types separate while integrating stimulated and control cells, and it was among the best methods at this task. However, scVAE-CM was worse than scVAE-CB at predicting counterfactual expressions for the top-100 DEGs, but it was still comparable to CoMP and scVI. Overall, scVAE-CM was the only method whose optimal models achieved competitive performance in all aspects of both analyses.

When also conditioning on library size and not only stimulation status, scVAE-CM models did perform worse at counterfactual prediction in contrast to other scVAE-C methods. We saw that this

lead to more mixing of cell types. Removing information about stimulation state and library size at the same time would lead to an even more compressed latent representation that is harder to counterfactually predict from for a method that expects multiple inherent clusters. A scVAE-CM model could collapse to a scVAE-CB model by letting all but one mixture coefficient go towards zero, but multiple modals might still fit a latent representation better, or early stopping might end the training of the model beforehand.

4.5 Unbalanced data set and differences in preprocessing it

The Kang data set presented several challenges with its uneven distribution of cell types and the large difference in expression between some of these cell types. This was especially the case for the monocytes and dendritic cells which have significantly higher average library sizes than the other cell types. Together, these cells are also somewhat underrepresented in the data set. Most methods had difficulty counterfactually predicting the stimulated expression of the most varying genes for these cell types. This might simply be because withholding stimulated cells of any of these cell types from a model during training, the model would have seen more examples of cells with low library sizes than with high library sizes.

Using raw or normalised scRNA-seq gene-expression counts was also a factor in how the methods performed. Normalising the gene expression counts for CVAE and CoMP means that these methods have less information to rely on for inference, but the counts seem to be easier to reconstruct – at least for cell types with similar average library size. Excluding stimulated CD14 monocytes, which are the second-most populous in the data set and one of the cell types with high library sizes, from training in the analysis of the counterfactual expression profiles, supports the last observation: The methods using the normalised counts are better at modelling the test set without having seen the withheld cells, whereas it harder for the ones using raw counts. This might be because of the difference in likelihood function with the former kind of methods using a normal distribution and the latter kind using some version of the negative binomial distribution. As for inference, with less signal in the normalised data, the methods would have to rely more on the covariates to encode the cells, which could be the reason why the stimulated and control cells are less integrated in their latent representations.

5 Conclusion

We have presented scVAE-C, which is a collection of methods for removing covariate effects from lower-dimensional (latent) representations and counterfactually predicting gene expression profiles. scVAE-C is based on the scVAE method (Grønbech et al., 2020), which include both a standard variational autoencoder and a normal-mixture version (GMVAE) for modelling scRNA-seq data. Our contributions were to add a conditional framework (Sohn et al., 2015) supporting multiple covariates, an optional hierarchical latent structure, and some modifications to the GMVAE version of scVAE. The methods we used for comparison each only support a different subset of this functionality, and none of them has the same flexibility in choosing covariates built-in. We found that all scVAE-C methods performed better than the comparison methods at removing covariate effects from their latent representations with the hierarchical scVAE-C methods achieving the best results. For counterfactual prediction, we observed that the nonhierarchical scVAE-C methods were competitive with the best comparison methods, while the hierarchical scVAE-C methods performed the worst. The base mixture scVAE-C method was the only method to achieve competitive results in both analyses.

The methods model scRNA-seq gene expression count data by conditioning on covariates, which allows the methods to represent the data without the effects of these covariates, since the methods did not have to attempt to model those effects themselves. Conditioning cells on covariates functioned for the latent representation similar to a preprocessing step of regressing out covariates, like some forms of batch correction, but since the covariate removal of the scVAE-C methods is built-in, biological variation is not necessarily lost. For the hierarchical scVAE-C methods, removing covariate effects can even be split between their two latent layers. Additionally, the generative processes of the methods

are able to reconstruct the gene expression profiles of the cells or even counterfactually predict profiles of the cells for different covariate values to various degrees of success.

We demonstrated this by applying the methods on the Kang data set (Kang et al., 2017), which consists of PBMCs from Lupus patients with one half of the cells having been IFN-β stimulated. By conditioning on the stimulation status, all scVAE-C methods, especially the hierarchical ones, were able remove its effects and integrate the two kinds of cells to a high degree and better than the methods we used for comparison: scVI (Lopez et al., 2018), MrVI (Boyeau et al., 2022), CVAE (Sohn et al., 2015), and CoMP (Foster et al., 2022). All scVAE-C methods but the base one were also able to keep the cell types almost completely separate – even more separate than a truncated SVD factorisation of the data. However, when the scVAE-C methods were also conditioned on the library sizes of the cells, cell types became somewhat mixed in the latent representations of the methods. Here, constrained negative binomial distributions provided a better way to account for the library size. As for counterfactual prediction, the base scVAE-C method performed best among the scVAE-C methods. It was able to counterfactually predict the mean expression profiles for stimulated cells of different cell types to a high degree comparable to the best comparison method, CoMP, especially when focusing on only the most differentially expressed genes. Whereas the base mixture scVAE-C method was close in performance to the base method, the hierarchical methods performed significantly worse at this task.

From these results, we observed a trade-off for the conditional scVAE methods between performing well at removing covariates from latent representations and producing counterfactual expression effectively. Looking at the variational lower bounds of the methods, they all consist of the sum of an expected negative reconstruction error and one of more negative KL divergences or similar terms. This is inherent to variational autoencoders: While the reconstruction error is optimised, the KL divergence acts as a regulariser ensuring that the model does not overfit. A lower reconstruction error signifies that a model is better able to reconstruct its input, and a lower KL divergence means the latent representation better follow the prior distribution for the latent variable in question. So the trade-off is built into the methods: For example, a decrease in reconstruction error can be partially offset by an increase in one or more KL divergences and still result in an overall higher variational lower bound. Complicating matters further, the reconstruction error does not take into account counterfactual values and the KL divergence(s) do not measure how cells are mixing according to some covariate. So one cannot assume that the variational lower bound or its components are good indicators for latent-representation covariate removal or counterfactual prediction. This was supported by the median integration and cell-type local inverse Simpson's indices measuring the mixing of cells as well as the mean coefficients of determination comparing counterfactual and factual expression both being almost independent of the variational lower bound. With this trade-off, the base mixture scVAE-C seems to achieve the best balance between the two analyses.

Because of the success of the covariate removal of the stimulation status, future work could investigate how well the scVAE-C methods perform at integrating experiment batches and even entire data sets. By treating different data sets as essentially other batches of a combined data set, the methods should be able to integrate the data sets in their latent representations. We would then like to examine if the methods are able to generate expression profiles of the cells without batch effects or for combining and integrating multiple data sets.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 813533 to C.H.G. R.DeV. and O.W.'s work were funded in part by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606). E.R. at contribution time (2022) was supported by the Novo Nordisk International Talent Program. O.W. acknowledge support from the Pioneer Center for AI, DNRF grant number P1.

Co-funded by
the European Union

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

T. S. Andrews, V. Y. Kiselev, D. McCarthy, and M. Hemberg. Tutorial: guidelines for the computational analysis of single-cell rna sequencing data. *Nature Protocols*, 16(1):1–9, Dec. 2020. ISSN 1750-2799. `DOI:10.1038/s41596-020-00409-w`.

P. Boyeau, J. Hong, A. Gayoso, M. Jordan, E. Azizi, and N. Yosef. Deep generative modeling for quantifying sample-level heterogeneity in single-cell omics. *bioRxiv*, 2022. `DOI:10.1101/2022.10.04.510898`.

A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420, 2018. `DOI:10.1038/nbt.4096`.

J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. D. Hoffman, and R. A. Saurous. Tensorflow distributions. *arXiv e-prints*, 2017. `arXiv:1711.10604`.

N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint*, Nov. 2016. `arXiv:1611.02648`.

J. Ding, A. Condon, and S. P. Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature Communications*, 9(1):2002, 2018. ISSN 2041-1723. `DOI:10.1038/s41467-018-04368-5`.

G. Eraslan, L. M. Simon, M. Mircea, N. S. Mueller, and F. J. Theis. Single cell RNA-seq denoising using a deep count autoencoder. *bioRxiv*, Apr. 2018. `DOI:10.1101/300681`.

A. Foster, A. Vezer, C. A. Glastonbury, P. Creed, S. Abujudeh, and A. Sim. Contrastive mixture of posteriors for counterfactual inference, data integration and fairness. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6578–6621. PMLR, 17–23 Jul 2022. `https://proceedings.mlr.press/v162/foster22a.html`.

J. Gagnon, L. Pi, M. Ryals, Q. Wan, W. Hu, Z. Ouyang, B. Zhang, and K. Li. Recommendations of scrna-seq differential gene expression analysis based on comprehensive benchmarking. *Life*, 12(6):850, June 2022. ISSN 2075-1729. `DOI:10.3390/life12060850`.

C. H. Grønbech, M. F. Vording, P. N. Timshel, C. K. Sønderby, T. H. Pers, and O. Winther. scVAE: Variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, 05 2020. ISSN 1367-4803. `DOI:10.1093/bioinformatics/btaa293`.

Y. Hao, T. Stuart, M. H. Kowalski, S. Choudhary, P. Hoffman, A. Hartman, A. Srivastava, G. Molla, S. Madad, C. Fernandez-Granda, and R. Satija. Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nature Biotechnology*, 2023. `DOI:10.1038/s41587-023-01767-y`.

D. M. Harris and S. L. Harris. *Digital Design and Computer Architecture*. Elsevier, 2013. ISBN 978-0-12-394424-5.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, July 2012. arXiv:1207.0580.

H. M. Kang, M. Subramaniam, S. Targ, M. Nguyen, L. Maliskova, E. McCarthy, E. Wan, S. Wong, L. Byrnes, C. M. Lanata, R. E. Gate, S. Mostafavi, A. Marson, N. Zaitlen, L. A. Criswell, and C. J. Ye. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology*, 36(1):89–94, Dec. 2017. DOI:10.1038/nbt.4042.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, 2014. https://iclr.cc/archive/2014/conference-proceedings/.

D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589, Montreal, Canada, 2014. Curran Associates, Inc.

I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-r. Loh, and S. Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 16(12):1289–1296, Nov. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0619-0.

R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018. ISSN 1548-7105. DOI:10.1038/s41592-018-0229-2.

M. Lotfollahi, F. A. Wolf, and F. J. Theis. scGen predicts single-cell perturbation responses. *Nature Methods*, 16(8):715–721, July 2019. DOI:10.1038/s41592-019-0494-8.

M. Lotfollahi, M. Naghipourfar, F. J. Theis, and F. A. Wolf. Conditional out-of-distribution generation for unpaired data using transfer VAE. *Bioinformatics*, 36(Supplement_2):i610–i617, Dec. 2020. DOI:10.1093/bioinformatics/btaa800.

M. Lotfollahi, A. K. Susmelj, C. De Donno, Y. Ji, I. L. Ibarra, F. A. Wolf, N. Yakubova, F. J. Theis, and D. Lopez-Paz. Learning interpretable cellular responses to complex perturbations in high-throughput screens. *bioRxiv*, 2021. DOI:10.1101/2021.04.14.439903.

M. D. Luecken and F. J. Theis. Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular Systems Biology*, 15(6), June 2019. ISSN 1744-4292. DOI:10.15252/msb.20188746.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv e-prints*, 2020. arXiv:1802.03426.

A. Oshlack, M. D. Robinson, and M. D. Young. From RNA-seq reads to differential expression results. *Genome Biol.*, 11(12):220, 2010. DOI:10.1186/gb-2010-11-12-220.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286. PMLR, 22–24 Jun 2014. http://proceedings.mlr.press/v32/rezende14.html.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902.e21, June 2019. ISSN 0092-8674. DOI:10.1016/j.cell.2019.05.031.

V. Svensson, E. da Veiga Beltrame, and L. Pachter. A curated database reveals trends in single-cell transcriptomics. *Database*, 2020, 11 2020. ISSN 1758-0463. DOI:10.1093/database/baaa073. baaa073.

D. Wang and J. Gu. VASC: Dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genomics, Proteomics & Bioinformatics*, 16(5):320–331, Oct. 2018. ISSN 1672-0229. DOI:10.1016/j.gpb.2018.08.003.

M. L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. DOI:10.21105/joss.03021.

F. A. Wolf, P. Angerer, and F. J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), Feb. 2018. ISSN 1474-760X. DOI:10.1186/s13059-017-1382-0.

A.1.1 Supplementary materials

SUPPLEMENTARY MATERIALS

scVAE-C: Latent-representation covariate removal and counterfactual expression profiles for single cells using conditional and hierarchical variational autoencoders

Christopher Heje Grønbech, Rachael DeVries, Eljas Roellin, and Ole Winther

Last typeset on 22 January 2025

S1 Negative binomial distribution and variations thereof

The negative binomial distribution can be defined in several ways. We use the one implemented by the TensorFlow Probability machine-learning software library (Dillon et al., 2017), which is used in the implementation of scVAE-C. In this formulation, the distribution models x successes of a sequence of Bernoulli (two-outcome) experiments given ρ failures and a probability of success for each Bernoulli experiment of ω:

$$\mathrm{NB}(x; \rho, \omega) = \frac{(x + \rho - 1)!}{x!(\rho - 1)!} \omega^x (1 - \omega)^r, \tag{S1}$$

This is a discrete probability function, but to better compute derivatives for optimisation during training of scVAE-C models, we let x be a real number, and the factorials are then computed using gamma functions, $\Gamma(x) = (x-1)!$. We also use different variations of the negative binomial distribution detailed below.

S1.1 Zero-inflated negative binomial (ZINB) distribution

A zero-inflated model supposes that there exists an excess amount of zeros in a population and models this subpopulation separately. This results in a mixture distribution, which for a zero-inflated negative binomial distribution can be formulated as

$$\mathrm{ZINB}(x; \rho, \omega, \epsilon) = \begin{cases} \epsilon + (1 - \epsilon)\mathrm{NB}(x; \rho, \omega), & x = 0, \\ (1 - \epsilon)\mathrm{NB}(x; \rho, \omega), & x > 0, \end{cases} \tag{S2}$$

where ϵ is the probability of excess zeros. Because of the sparsity of scRNA-seq data, this distribution has been proven to work favourably with this kind of data (Lopez et al., 2018).

S1.2 Constrained negative binomial (CNB) distribution

We were inspired to use this distribution by the constrained Poisson distribution (Salakhutdinov and Hinton, 2009), which was also used in Grønbech et al. (2020). A constrained zero-inflated negative binomial distribution is also the default choice for scVI Lopez et al. (2018) when not modelling the size factor as a latent variable. To apply the same idea to the negative binomial distribution, we first

need to reformulate the distribution using its mean. The mean ν of the negative binomial distribution as formulated in equation (S1) is

$$\nu = \frac{\rho\omega}{1-\omega}. \tag{S3}$$

By replacing ω in equation (S1), we can reformulate the negative binomial distribution as

$$\mathrm{NB}(x;\rho,\nu) = \frac{(x+\rho-1)!}{x!(\rho-1)!}\left(\frac{\nu}{\nu+\rho}\right)^x\left(\frac{\rho}{\nu+\rho}\right)^r. \tag{S4}$$

Next, we need to consider all counts x_n of a cell together as $\mathbf{x}$. We then constrain the mean $\boldsymbol{\nu}$ to be equal to the library size ℓ, which is the sum of these counts, by reparameterisation:

$$\boldsymbol{\nu} = \ell\boldsymbol{\tau}, \tag{S5}$$

where $\boldsymbol{\tau}$ is a probability vector, which means that all components are non-negative and their sum is 1. In component form, this yields the constrained negative binomial distribution:

$$\mathrm{CNB}(x;\rho_n,\tau_n) = \frac{(x+\rho_n-1)!}{x!(\rho_n-1)!}\left(\frac{\ell\tau_n}{\ell\tau_n+\rho_n}\right)^x\left(\frac{\rho_n}{\ell\tau_n+\rho_n}\right)^r, \quad \ell = \sum_n x_n, \quad \tau_n \geq 0, \quad \sum_n \tau_n = 1. \tag{S6}$$

S1.3 Free-dispersion negative binomial (FDNB) distribution

In the formulation of the negative binomial distribution using its mean, equation (S4), ρ is sometimes reparameterised as $\rho = \frac{1}{\alpha}$, and α is called the dispersion. In the main text, the parameters of a negative binomial distribution are dependent on the latent variable $\mathbf{z}$ and conditions $\mathbf{c}$:

$$\mathrm{NB}(x;\rho_{\boldsymbol{\theta}}(\mathbf{z},\mathbf{c}),\omega_{\boldsymbol{\theta}}(\mathbf{z},\mathbf{c})). \tag{S7}$$

Letting the dispersion be a free variable simply means that the dispersion parameter or equivalently the failure-count parameter ρ is independent of any variables and is learnt during training. This results in the following formulation:

$$\mathrm{FDNB}(x;\rho_{\boldsymbol{\theta}}(\mathbf{z},\mathbf{c}),\omega_{\boldsymbol{\theta}}). \tag{S8}$$

We were inspired to use this distribution by Lopez et al. (2018).

S1.4 Combining variations

Since the changes to the negative binomial distributions in the variations above are independent of each other, these variations can all be combined with expected abbreviations. For example, FDCNB stands for a free-dispersion constrained negative binomial distribution. All seven combinations as well as the standard distribution are used.

S2 Conditional hierarchical scVAE (scVAE-CH)

In the following are presented the expressions for this method (the generative and inference processes are reproduced for easy reference) as well as the derivation of some of these.

Generative process

$$p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}_1,\mathbf{z}_2 \mid \mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2,\mathbf{c})\, p(\mathbf{z}_2), \tag{S9a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1) = \prod_{n=1}^{N} \mathrm{NB}(x_n;\rho_n^{\boldsymbol{\theta}}(\mathbf{z}_1),\omega_n^{\boldsymbol{\theta}}(\mathbf{z}_1)), \tag{S9b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2,\mathbf{c}) = \mathcal{N}(\mathbf{z}_1;\boldsymbol{\mu}_{1,\boldsymbol{\theta}}(\mathbf{z}_2,\mathbf{c}),\boldsymbol{\sigma}^2_{1,\boldsymbol{\theta}}(\mathbf{z}_2,\mathbf{c})\mathbf{I}), \tag{S9c}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2) = \mathcal{N}(\mathbf{z}_2;\mathbf{0},\mathbf{I}). \tag{S9d}$$

Inference process

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c}) = q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}), \tag{S10a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c}) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c})\mathbf{I}), \tag{S10b}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{1,\boldsymbol{\phi}}(\mathbf{x})\mathbf{I}). \tag{S10c}$$

Variational lower bound

Following the derivation for a standard VAE method, we can separate the variational lower bound for the conditional hierarchical scVAE method into multiple terms:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c})}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})\, p(\mathbf{z}_2)}{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1) + \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} + \log \frac{p(\mathbf{z}_2)}{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})}\right]\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)] - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})}\right]\right] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}[\mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c}) \,\|\, p(\mathbf{z}_2)]].
\end{aligned} \tag{S11}$$

The first term is the expected negative reconstruction error, the second one is reminiscent of a negative KL divergence for $\mathbf{z}_1$, and the last one is the expected negative KL divergence for $\mathbf{z}_2$ with regards to $q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})$. We refer to the second term as a pseudo KL divergence.

Reconstruction

To reconstruct $\mathbf{x}$ for the conditional hierarchical scVAE method, we apply the same idea as for the base conditional scVAE method, but for each latent layer in sequence:

$$\begin{aligned}
\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)}[\mathbf{x}]\Big]\Big]\Big] \\
&= \iiint \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}_1')\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \,|\, \mathbf{z}_2, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S12}$$

Counterfactual prediction

As for the scVAE-CB method, to perform counterfactual prediction, we replace the factual covariate values $\mathbf{c}$ in the remaining generative process with counterfactual covariate values $\mathbf{c}^*$:

$$\begin{aligned}
\mathbf{x}^*(\mathbf{x}, \mathbf{c}, \mathbf{c}^*) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c}^*)}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)}[\mathbf{x}]\Big]\Big]\Big] \\
&= \iiint \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}_1')\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \,|\, \mathbf{z}_2, \mathbf{c}^*)\, q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S13}$$

S2.1 Latent-layer conditions

Instead of using the same set of covariates, $\mathbf{c}$, for both latent layers, we could use two subsets of covariates, $\mathbf{c}_1$ and $\mathbf{c}_2$, and condition on these separately. We show in the following how this is done, and what changes follow from this.

Generative process

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{c}_1, \mathbf{c}_2) = p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1, \mathbf{c}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c}_2)\, p(\mathbf{z}_2), \tag{S14a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1) = \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}_1, \mathbf{c}), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}_1, \mathbf{c})), \tag{S14b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}_2), \boldsymbol{\sigma}^2_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}_2)\mathbf{I}), \tag{S14c}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2) = \mathcal{N}(\mathbf{z}_2; \mathbf{0}, \mathbf{I}). \tag{S14d}$$

This is shown in Supplementary Figure S1c as solid arrows.

Inference process

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) = q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1), \tag{S15a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c}_2), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c}_2)\mathbf{I}), \tag{S15b}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}_1), \boldsymbol{\sigma}^2_{1,\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}_1)\mathbf{I}). \tag{S15c}$$

This is shown in Supplementary Figure S1c as dashed arrows.

Variational lower bound

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2)}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{c}_1, \mathbf{c}_2)}{q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2)}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1)] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2)}\right]\right] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}[\mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2) \,\|\, p(\mathbf{z}_2)]].
\end{aligned} \tag{S16}$$

Reconstruction

$$\begin{aligned}
\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2)}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1)}[\mathbf{x}]\Big]\Big]\Big] \\
&= \iiint \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}_1', \mathbf{c}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \mid \mathbf{z}_2, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S17}$$

Counterfactual prediction

$$\begin{aligned}
\mathbf{x}^*(\mathbf{x}, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_1^*, \mathbf{c}_2^*) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2^*)}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1^*)}[\mathbf{x}]\Big]\Big]\Big] \\
&= \iiint \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}_1', \mathbf{c}_1^*)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \mid \mathbf{z}_2, \mathbf{c}_2^*) \\
&\qquad \times q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S18}$$

This is illustrated in Supplementary Figure S2c.

S3 Conditional mixture scVAE (scVAE-CM)

A categorical stochastic latent variable y is added to the conditional scVAE method, and $\mathbf{z}$ is conditioned on y to better model the underlying structure in the latent representation. This can be seen as modelling the inherent clusters in the data. Below are shown the expression and derivations for this method.

Generative process

As in Grønbech et al. (2020), we use a normal-mixture model as the prior distribution for this method. So y is modelled using a categorical distribution, and the prior for $\mathbf{z}$, which is now a conditional prior, is a multivariate normal distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, y, \mathbf{z} \,|\, \mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y)\, p_{\boldsymbol{\theta}}(y), \tag{S19a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c}) = \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c}), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})), \tag{S19b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(y), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(y)\mathbf{I}), \tag{S19c}$$

$$p_{\boldsymbol{\theta}}(y) = \mathrm{Cat}(y; \boldsymbol{\pi}_{\boldsymbol{\theta}}). \tag{S19d}$$

The distribution parameters for the conditional prior for $\mathbf{z}$ are now given by y, and they are modelled by neural networks. The prior distribution for y is a categorical distribution with mixture coefficient $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is a free variable learnt during training, which makes the mixture model able to adjust the contribution of each component and even omit some components if they are not needed. The mixture coefficient for omitted components will tend toward zero. The model graph is shown in Supplementary Figure S1d as solid arrows.

Inference process

Unlike Grønbech et al. (2020), we separate the approximate posterior distributions for $\mathbf{z}$ and y:

$$q_{\boldsymbol{\phi}}(y, \mathbf{z} \,|\, \mathbf{x}, \mathbf{c}) = q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})\, q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c}), \tag{S20a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})\mathbf{I}), \tag{S20b}$$

$$q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c}) = \mathrm{Cat}(y; \boldsymbol{\pi}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})). \tag{S20c}$$

Here, all distribution parameters are given by $\mathbf{x}$ and $\mathbf{c}$ and modelled using neural networks. In Grønbech et al. (2020), $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})$ is also conditioned on y. However, we found that such a formulation only works for smaller and simpler data sets when we condition on covariates. The model graph is shown in Supplementary Figure S1d as dashed arrows.

Variational lower bound

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(y, \mathbf{z} | \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y, \mathbf{z} \,|\, \mathbf{c})}{q_{\boldsymbol{\phi}}(y, \mathbf{z} | \mathbf{x}, \mathbf{c})}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})\, q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y)\, p_{\boldsymbol{\theta}}(y)}{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})\, q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c}) + \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y)}{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})} + \log \frac{p_{\boldsymbol{\theta}}(y)}{q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})}\right]\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(y | \mathbf{x}, \mathbf{c})}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} | \mathbf{x}, \mathbf{c})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{c})]\Big] - \mathrm{E}_{q_{\boldsymbol{\phi}}(y | \mathbf{x}, \mathbf{c})}[\mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y)]] \\
&\quad - \mathrm{KL}[q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c}) \,\|\, p_{\boldsymbol{\theta}}(y)].
\end{aligned} \tag{S21}$$

The first term is the expected negative reconstruction error averaged over y, the second term is the negative KL divergence for $\mathbf{z}$ also averaged over y, and the last term is the negative KL divergence for y. Even though the reconstruction error is not dependent on y, we found that the method is more robust when evaluating the reconstruction error for each component of y and averaging over them. This makes this variational lower bound similar to the one for the GMVAE method in Grønbech et al. (2020).

Reconstruction

$$\begin{aligned}\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(y|\mathbf{x},\mathbf{c})}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x},\mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z},\mathbf{c})}[\mathbf{x}]\Big]\Big] \\ &= \int \sum_{y} \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})\, q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})\, \mathrm{d}\mathbf{z}.\end{aligned} \tag{S22}$$

Counterfactual prediction

$$\begin{aligned}\mathbf{x}^*(\mathbf{x}, \mathbf{c}, \mathbf{c}^*) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(y|\mathbf{x},\mathbf{c})}\Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x},\mathbf{c})}\Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z},\mathbf{c}^*)}[\mathbf{x}]\Big]\Big] \\ &= \int \sum_{y} \sum_{\mathbf{x}'} \mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}, \mathbf{c}^*)\, q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}, \mathbf{c})\, q_{\boldsymbol{\phi}}(y \,|\, \mathbf{x}, \mathbf{c})\, \mathrm{d}\mathbf{z}.\end{aligned} \tag{S23}$$

This is illustrated in Supplementary Figure S2d.

S4 Conditional hierarchical mixture scVAE (scVAE-CHM)

This is a combination of the conditional hierarchical scVAE and the conditional mixture scVAE methods. Only the second (inner-most) latent variable is modelled using a normal-mixture model. To highlight this, the categorical latent variable uses 2 as a subscript: y_2. The expressions and derivations for this method are shown in the following.

Generative process

$$p_{\boldsymbol{\theta}}(\mathbf{x}, y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2)\, p_{\boldsymbol{\theta}}(y_2), \tag{S24a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1) = \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}_1), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}_1)), \tag{S24b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c}) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}), \boldsymbol{\sigma}^2_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c})\mathbf{I}), \tag{S24c}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(y_2), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(y_2)\mathbf{I}), \tag{S24d}$$

$$p_{\boldsymbol{\theta}}(y_2) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\theta}}). \tag{S24e}$$

As for the other methods, all distribution parameters are modelled using neural networks, except for $\boldsymbol{\pi}_{2,\boldsymbol{\theta}}$ which is a free variable. The model graph is shown in Supplementary Figure S1e as solid arrows.

Inference process

$$q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c}) = q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}), \tag{S25a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c}) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{c}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{c})\mathbf{I}), \tag{S25b}$$

$$q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c}) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c})), \tag{S25c}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{1,\boldsymbol{\phi}}(\mathbf{x})\mathbf{I}). \tag{S25d}$$

Again, all distribution parameters are modelled using neural networks – no exceptions. The model graph is shown in Supplementary Figure S1e as dashed arrows.

Variational lower bound

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{c})}{q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{x}, \mathbf{c})} \right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c}) q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})\, p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2)\, p_{\boldsymbol{\theta}}(y_2)}{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})\, q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \Bigg[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Bigg[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Bigg[\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1) + \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})}{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \\
&\qquad\qquad + \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2)}{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})} + \log \frac{p_{\boldsymbol{\theta}}(y_2)}{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Bigg] \Bigg] \Bigg] \qquad \text{(S26)} \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} [\log p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \Bigg[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Bigg[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Bigg[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})} \Bigg] \Bigg] \Bigg] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} [\mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2)]] \Big] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} [\mathrm{KL}[q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c}) \,\|\, p_{\boldsymbol{\theta}}(y_2)]].
\end{aligned}
$$

Here, the first term is the expected negative reconstruction error while the other three terms are negative (pseudo) KL divergences for $\mathbf{z}_1$, $\mathbf{z}_2$, and y_2, respectively, with respect to relevant distributions. The second term is averaged over y even though the distributions are independent of y. As with the scVAE-CM method, we found that this made the method more robust.

Reconstruction

$$
\begin{aligned}
\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c})} \Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)} [\mathbf{x}] \Big] \Big] \Big] \Big] \\
&= \iiint \sum_{y_2} \sum_{\mathbf{x}'} (\mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}_1')\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \,|\, \mathbf{z}_2, \mathbf{c}) \\
&\qquad\qquad \times q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}))\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1. \qquad \text{(S27)}
\end{aligned}
$$

Counterfactual prediction

$$
\begin{aligned}
\mathbf{x}^*(\mathbf{x}, \mathbf{c}, \mathbf{c}^*) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x})} \Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Big[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})} \Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c}^*)} \Big[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1)} [\mathbf{x}] \Big] \Big] \Big] \Big] \\
&= \iiint \sum_{y_2} \sum_{\mathbf{x}'} (\mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \,|\, \mathbf{z}_1')\, p_{\boldsymbol{\theta}}(\mathbf{z}_1' \,|\, \mathbf{z}_2, \mathbf{c}^*) \\
&\qquad\qquad \times q_{\boldsymbol{\phi}}(\mathbf{z}_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(y_2 \,|\, \mathbf{z}_1, \mathbf{c})\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \,|\, \mathbf{x}))\, \mathrm{d}\mathbf{z}_1'\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1. \qquad \text{(S28)}
\end{aligned}
$$

This is illustrated in Supplementary Figure S2e.

S4.1 Latent-layer conditions

As with the conditional hierarchical scVAE method, the mixture one can also use a different subset of covariates for each latent layer (y_2 belongs to the $\mathbf{z}_2$ layer). This is done in the same way as in Supplementarty Text S2.1.

Generative process

$$
\begin{aligned}
p_{\boldsymbol{\theta}}(\mathbf{x}, y_2, \mathbf{z}_1, \mathbf{z}_2 \,|\, \mathbf{c}_1, \mathbf{c}_2) &= p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1, \mathbf{c}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}_1 \,|\, \mathbf{z}_2, \mathbf{c}_2)\, p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2)\, p_{\boldsymbol{\theta}}(y_2), \qquad \text{(S29a)} \\
p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}_1, \mathbf{c}_1) &= \prod_{n=1}^{N} \mathrm{NB}(x_n; \rho_n^{\boldsymbol{\theta}}(\mathbf{z}_1, \mathbf{c}), \omega_n^{\boldsymbol{\theta}}(\mathbf{z}_1, \mathbf{c})), \qquad \text{(S29b)}
\end{aligned}
$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}_2), \boldsymbol{\sigma}^2_{1,\boldsymbol{\theta}}(\mathbf{z}_2, \mathbf{c}_2)\mathbf{I}), \tag{S29c}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \mid y_2) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(y_2), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(y_2)\mathbf{I}), \tag{S29d}$$

$$p_{\boldsymbol{\theta}}(y_2) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\theta}}). \tag{S29e}$$

This is shown in Supplementary Figure S1f as solid arrows.

Inference process

$$q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) = q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1), \tag{S30a}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{c}_2), \boldsymbol{\sigma}^2_{2,\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{c}_2)\mathbf{I}), \tag{S30b}$$

$$q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{c}_2)), \tag{S30c}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1) = \mathcal{N}(\mathbf{z}_1; \boldsymbol{\mu}_{1,\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}_1), \boldsymbol{\sigma}^2_{1,\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}_1)\mathbf{I}). \tag{S30d}$$

This is shown in Supplementary Figure S1f as dashed arrows.

Variational lower bound

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2)}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y_2, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{c}_1, \mathbf{c}_2)}{q_{\boldsymbol{\phi}}(y_2, \mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}, \mathbf{c}_1, \mathbf{c}_2)}\right] \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1)] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2)}\right]\right]\right] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)}[\mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2) \parallel p_{\boldsymbol{\theta}}(\mathbf{z}_2 \mid y_2)]]\right] \\
&\quad - \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}[\mathrm{KL}[q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2) \parallel p_{\boldsymbol{\theta}}(y_2)]].
\end{aligned} \tag{S31}$$

Reconstruction

$$\begin{aligned}
\tilde{\mathbf{x}}(\mathbf{x}, \mathbf{c}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}_2)}\left[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}_1)}[\mathbf{x}]\right]\right]\right]\right] \\
&= \iiint \sum_{y_2} \sum_{\mathbf{x}'} (\mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}'_1, \mathbf{c}_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}'_1 \mid \mathbf{z}_2, \mathbf{c}_2) \\
&\qquad \times q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1))\, \mathrm{d}\mathbf{z}'_1\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S32}$$

Counterfactual prediction

$$\begin{aligned}
\mathbf{x}^*(\mathbf{x}, \mathbf{c}, \mathbf{c}^*) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)}\left[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_1 \mid \mathbf{z}_2, \mathbf{c}^*_2)}\left[\mathrm{E}_{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_1, \mathbf{c}^*_1)}[\mathbf{x}]\right]\right]\right]\right] \\
&= \iiint \sum_{y_2} \sum_{\mathbf{x}'} (\mathbf{x}'\, p_{\boldsymbol{\theta}}(\mathbf{x}' \mid \mathbf{z}'_1, \mathbf{c}^*_1)\, p_{\boldsymbol{\theta}}(\mathbf{z}'_1 \mid \mathbf{z}_2, \mathbf{c}^*_2) \\
&\qquad \times q_{\boldsymbol{\phi}}(\mathbf{z}_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(y_2 \mid \mathbf{z}_1, \mathbf{c}_2)\, q_{\boldsymbol{\phi}}(\mathbf{z}_1 \mid \mathbf{x}, \mathbf{c}_1))\, \mathrm{d}\mathbf{z}'_1\, \mathrm{d}\mathbf{z}_2\, \mathrm{d}\mathbf{z}_1.
\end{aligned} \tag{S33}$$

This is illustrated in Supplementary Figure S2f.

S5 Details regarding analyses

Additional information about the analyses are detailed below.

S5.1 Hyperparameters for analyses

To compute local inverse Simpson's indices, we use the Python package harmonypy[1], which is based on the R package Harmony (Korsunsky et al., 2019), and the default perplexity of 30 is used for these computations.

We use ScanPy (Wolf et al., 2018) to compute UMAP embeddings, and we use the default neighbourhood size of 15 and the default minimum distance of 0.5 for these computations.

S5.2 Computing statistics for the correlation of determination

To estimate the mean and standard deviation of correlation of determination, we use m-out-of-n bootstrapping with replacement (Bickel et al., 2011) where resampling only uses a subset of cells. As in Lotfollahi et al. (2019), we use a subset that is 80 % of the cell population used for evaluation and resample 100 times.

S6 Detailed experiment setup

Below different parts of the experiment setup are explained in more detail.

S6.1 Model setup

All neural networks consist of two intermediate (hidden) layers with 100 units each, while the number of units of the output layer depends on the parameter that is modelled. All continuous latent variables have a dimension of 10, so $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ layers have this number of units. For the mixture methods, we use 15 components to have about two components on average to model each of the Kang data set's eight cell types. Therefore, the $\boldsymbol{\pi}_{\boldsymbol{\phi}}$ layer has that many units, but since $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is a free variable and not fixed, a model is not forced to use all components (see Supplementary Note S3). The remaining distribution parameters are all for the likelihood function, so the number of units for the corresponding output layers are therefore equal to the number of genes.

Additionally, the intermediate layers use a rectifier function, $f_{\mathrm{ReLU}}(x) = \max(0, x)$, as activation function and employ batch normalisation (Ioffe and Szegedy, 2015) as well as dropout regularisation (Hinton et al., 2012) in some cases (see Supplementary Note S6.3). The parameter output layers all use activation functions appropriate to each parameter: the identity function; a sigmoid function, $f_{\mathrm{sigmoid}}(x) = \frac{1}{1+\mathrm{e}^{-x}}$ (Han and Moraga, 1995), for single probability parameters; the softplus function, $f_{\mathrm{softplus}}(x) = \log(1 + \mathrm{e}^{x})$ (Dugas et al., 2000), for parameters with only positive support; or the softmax function (Bridle, 1990) for normalising probability-vector parameters. Outputs are always clipped before any exponentiation to avoid numerical under- or overflow. This is all summarised in Supplementary Table S1.

S6.2 Training procedure

Models are trained using the Adam optimisation method (Kingma and Ba, 2014) using a mini-batch size of 64, a learning rate of 10^{-3}, and a gradient-clipping norm of 3. They are trained for 400 epochs with an early-stopping scheme as well as a warm-up scheme (Bowman et al., 2016; Sønderby et al., 2016) for the first 50 epochs. In the warm-up scheme, any (psuedo) KL divergence in the computation of the variational lower bound is multiplied by a variable β which is linearly increased from 0 to 1 during the warm-up.

S6.3 Configuration search

Configuration searches are performed for each method to determine the optimal combination of likelihood function, set of conditions, and dropout regularisation. For the likelihood function, we test the negative binomial distribution as well as the seven variations detailed in Supplementary Note S1. The constrained negative binomial distribution is one way to make the model depend directly on

[1] Available at `https://github.com/slowkow/harmonypy`

library size. Another way is to directly condition on library size (in addition to the integration covariate). We test this by either conditioning only on the integration covariate and using all negative binomial distributions or conditioning on both integration covariate and library size and using all non-constrained negative binomial distributions. How the different methods are conditioned are listed in Supplementary Table S2. Additionally, we test whether using a dropout rate of 0.1 or not for the intermediate layers improves the performance.

S6.4 Model selection

To select the optimal models for each method, we could just use the variational lower bound, since this was optimised during training. However, this does not necessarily lead to the best performance for all analyses. Instead, we use a metric relevant for each analysis. We use the median test iLISI for the analysis of latent-representation covariate removal. We only use this metric since we are interested in how the selected models perform in mixing other covariates. For the analysis of the counterfactual expression profiles, we use the mean test coefficients of determination: $\overline{R^2_{\text{all}}}$ for all genes and $\overline{R^2_{\text{top}}}$ when using only the top-100 DEGs. This is because we are interested in how models selected by these metrics perform against each other. As an exception, for reporting the optimal test variational lower bound, we still use the metric itself for model selection. Note that the metrics for each model are the result of taking the median over the five replicates trained for that model.

S6.5 Comparison methods

Because of their implementation, scVI and MrVI use a different combination of examples from the training and validation sets for training and validation for each replicate. Also the training–validation split is 80 % instead of 75 % (60 % of 80 % of the full data set), which the other methods are trained with. Therefore, we can only compare scVAE-C to these methods using the test set.

Since CVAE, CoMP, and scVI are VAEs with only one single latent layer, we denote their latent representation by $\mathbf{z}$. MrVI is hierarchical method with two latent layers: A stochastic latent variable $\mathbf{u}$ which models the data free of a specified covariate, and a deterministic latent variable $\mathbf{z}$ which is a sum of $\mathbf{u}$ and sources of variation specific to another covariate. Comparing the layers of MrVI with those of the hierarchical scVAE-C methods, we use $\mathbf{z}_1$ for MrVI's $\mathbf{z}$ latent variable and $\mathbf{z}_2$ for its $\mathbf{u}$ latent variable.

The hyperparameter optimisation of CVAE and CoMP was performed in two steps. First, a learning-rate sweep for values 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6} was performed using a neural-network architecture with one layer of 512 units and a latent dimension of 40. With the chosen learning rate, a sweep of neural-network architectures was performed. One, two, and three layers were tested with either 128, 256, or 512 units per layer and a latent dimension of either 10 or 40. For both CVAE and CoMP, this resulted in a learning rate of 10^{-3}, neural networks using one layer of 512 units, and a latent dimension of 40. For scVI, the default parameters were used: a learning rate of 10^{-3}, neural networks using one layer of 128 units, and a latent dimension of 10. For MrVI, the default parameters were also used: a learning rate of 10^{-2}, an encoder of two layers of each 128 units, two decoders with one layer of 32 units each, and a dimension of 10 for both latent variables. Early stopping was enabled for all comparison models.

For each final configuration, models were trained trained five times each with different random seeds. We used scVI with its zero-inflated negative binomial distribution with free dispersion similar to scVAE-C (Supplementary Note S1.3) as likelihood function. MrVI uses a standard negative binomial distribution, and CVAE and CoMP uses a normal distribution. By default, scVI and MrVI do not use conditioned covariates for inference, but only use them for generating gene expression counts. All comparison methods use the stimulation status as condition for the Kang data set. MrVI's $\mathbf{z}_1$ ($\mathbf{z}$) layer is used for conditioning, and the reconstruction is not conditioned on any additional covariate, since it can only be conditioned on categorical information, and no other relevant categorical covariates are available in the Kang data set.

S7 Conditioning priors on covariates

Below we show how to condition the prior distribution of all conditional scVAE methods on covariates $\mathbf{c}$.

S7.1 scVAE-CB

To condition the prior distribution of $\mathbf{z}$ for the scVAE-CB method on the covariates $\mathbf{c}$, we simply assume $\mathbf{z}$ to follow a normal distribution with parameters dependent on $\mathbf{c}$:

$$p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{c}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{c})\mathbf{I}), \tag{S34}$$

where the mean and standard deviation are given $\mathbf{c}$ and modelled using neural networks. This replaces $p_{\boldsymbol{\theta}}(\mathbf{z})$ in all relevant formulations for the scVAE-CB method.

S7.2 scVAE-CM

The conditional mixture scVAE method has two prior distributions: one for y and a conditional one for $\mathbf{z}$. For both, we again make the parameters dependent on $\mathbf{c}$:

$$p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y, \mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(y, \mathbf{c}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(y, \mathbf{c})\mathbf{I}), \tag{S35a}$$

$$p_{\boldsymbol{\theta}}(y \,|\, \mathbf{c}) = \mathrm{Cat}(y; \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{c})). \tag{S35b}$$

The parameters of the prior distribution of $\mathbf{z}$ are now also given by $\mathbf{c}$ in addition to y, and they are still modelled using neural networks. The probability vector $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is not a free parameter in this instance and instead given by $\mathbf{c}$ and also modelled using neural networks. $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y, \mathbf{c})$ and $p_{\boldsymbol{\theta}}(y \,|\, \mathbf{c})$ replaces $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, y)$ and $p_{\boldsymbol{\theta}}(y)$, respectively, in all relevant formulations for the scVAE-CM method.

S7.3 scVAE-CH

We condition the prior distribution of the conditional hierarchical scVAE method on $\mathbf{c}$ in the same way as for the scVAE-CB method (see Supplementary Note S7.1):

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, \mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(\mathbf{c}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(\mathbf{c})\mathbf{I}). \tag{S36}$$

Alternatively, when using latent-layer conditions, we can introduce another condition subset $\mathbf{c}_3$ and condition the prior on this instead:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, \mathbf{c}_3) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(\mathbf{c}_3), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(\mathbf{c}_3)\mathbf{I}). \tag{S37}$$

Either of these replace $p_{\boldsymbol{\theta}}(\mathbf{z})$ in all relevant formulations for the scVAE-CH method.

S7.4 scVAE-CHM

We combine the ways to condition the prior distributions of the conditional mixture and hierarchical scVAE methods on $\mathbf{c}$ to do the same for the conditional hierarchical mixture scVAE method (see Supplementary Notes S7.2–S7.3):

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2, \mathbf{c}) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(y_2, \mathbf{c}), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(y_2, \mathbf{c})\mathbf{I}), \tag{S38a}$$

$$p_{\boldsymbol{\theta}}(y_2 \,|\, \mathbf{c}) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\theta}}(\mathbf{c})). \tag{S38b}$$

Or alternatively with latent-layer conditions:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_2 \,|\, y_2, \mathbf{c}_3) = \mathcal{N}(\mathbf{z}_2; \boldsymbol{\mu}_{2,\boldsymbol{\theta}}(y_2, \mathbf{c}_3), \boldsymbol{\sigma}^2_{2,\boldsymbol{\theta}}(y_2, \mathbf{c}_3)\mathbf{I}), \tag{S39a}$$

$$p_{\boldsymbol{\theta}}(y_2 \,|\, \mathbf{c}_3) = \mathrm{Cat}(y_2; \boldsymbol{\pi}_{2,\boldsymbol{\theta}}(\mathbf{c}_3)). \tag{S39b}$$

Each set of equations replaces the corresponding set of prior distributions in all relevant formulations for the scVAE-CHM method.

S8 Supplementary Tables

Table S1. Properties of neural-network layers of different kinds. For output layers the parameter being modelled is used to represent the layer kind.

Kind	Layer count	Unit count	Activation	Normalisation	Dropout rate
Intermediate	2	100	Rectifier	Batch	Variable[a]
$\boldsymbol{\mu}$ (all)	1	10	Identify	None	None
$\boldsymbol{\sigma}^2$ (all)	1	10	Softplus	None	None
$\boldsymbol{\pi}_{\boldsymbol{\phi}}$ (all)	1	15	Softmax	None	None
$\boldsymbol{\rho}_{\boldsymbol{\theta}}$	1	n	Softplus	None	None
$\boldsymbol{\omega}_{\boldsymbol{\theta}}$	1	n	Sigmoid	None	None
$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$	1	n	Softmax[b]	None	None
$\boldsymbol{\tau}_{\boldsymbol{\theta}}$	1	n	Softmax	None	None

[a] Whether the intermediate layers use dropout regularisation depends on the configuration search (Supplementary Note S6.3.)

[b] The softmax function is not used to normalise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ over all n components, but to normalise ϵ_n and $1 - \epsilon_n$ (see equation S2) for each component n.

Table S2. Conditions for each method and each set of conditions. The first set of conditions uses only stimulation status, s, whereas the second set of conditions uses both stimulation status and library size, ℓ. Note that for the hierarchical methods we use the notation for condition subsets introduced in Supplementary Note S2.1.

Method	Stimulation status			Stimulation status, library size		
	$\mathbf{c}$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}$	$\mathbf{c}_1$	$\mathbf{c}_2$
scVAE-CB	s	—	—	s, ℓ	—	—
scVAE-CM	s	—	—	s, ℓ	—	—
scVAE-CH	—	s	None	—	ℓ	s
scVAE-CHM	—	s	None	—	ℓ	s

Table S3. Software libraries used by the scVAE-C implementation.

Software library	Purpose	Reference
TensorFlow	Modelling	Abadi et al. (2015)
TensorFlow Probability	Modelling	Dillon et al. (2017)
anndata	Input/output	Virshup et al. (2021)
pandas	Input/output	McKinney (2010)
Loompy	Input	`https://loompy.org`
seaborn	Visualisation	Waskom (2021)
Matplotlib	Visualisation	Hunter (2007)
scikit-learn	Visualisation	Pedregosa et al. (2011)
SciPy	Sparse matrices	Virtanen et al. (2020)
NumPy	Various	Harris et al. (2020)
natsort	Natural sorting	`https://github.com/SethMMorton/natsort`

Table S4. Software libraries used by the implementation of preprocessing, experiment setup, and analyses for scVAE-C.

Software library	Purpose	Reference
Scanpy	Input, computation, and visualisation	Wolf et al. (2018)
anndata	Input	Virshup et al. (2021)
pandas	Input/output and computation	McKinney (2010)
NumPy	Computation	Harris et al. (2020)
harmonypy	LISI computation	`https://github.com/slowkow/harmonypy`
seaborn	Visualisation	Pedregosa et al. (2011)
Matplotlib	Visualisation	Hunter (2007)
textalloc	Visualisation	`https://github.com/ckjellson/textalloc`
tol_colors	Visualisation	`https://personal.sron.nl/~pault/`

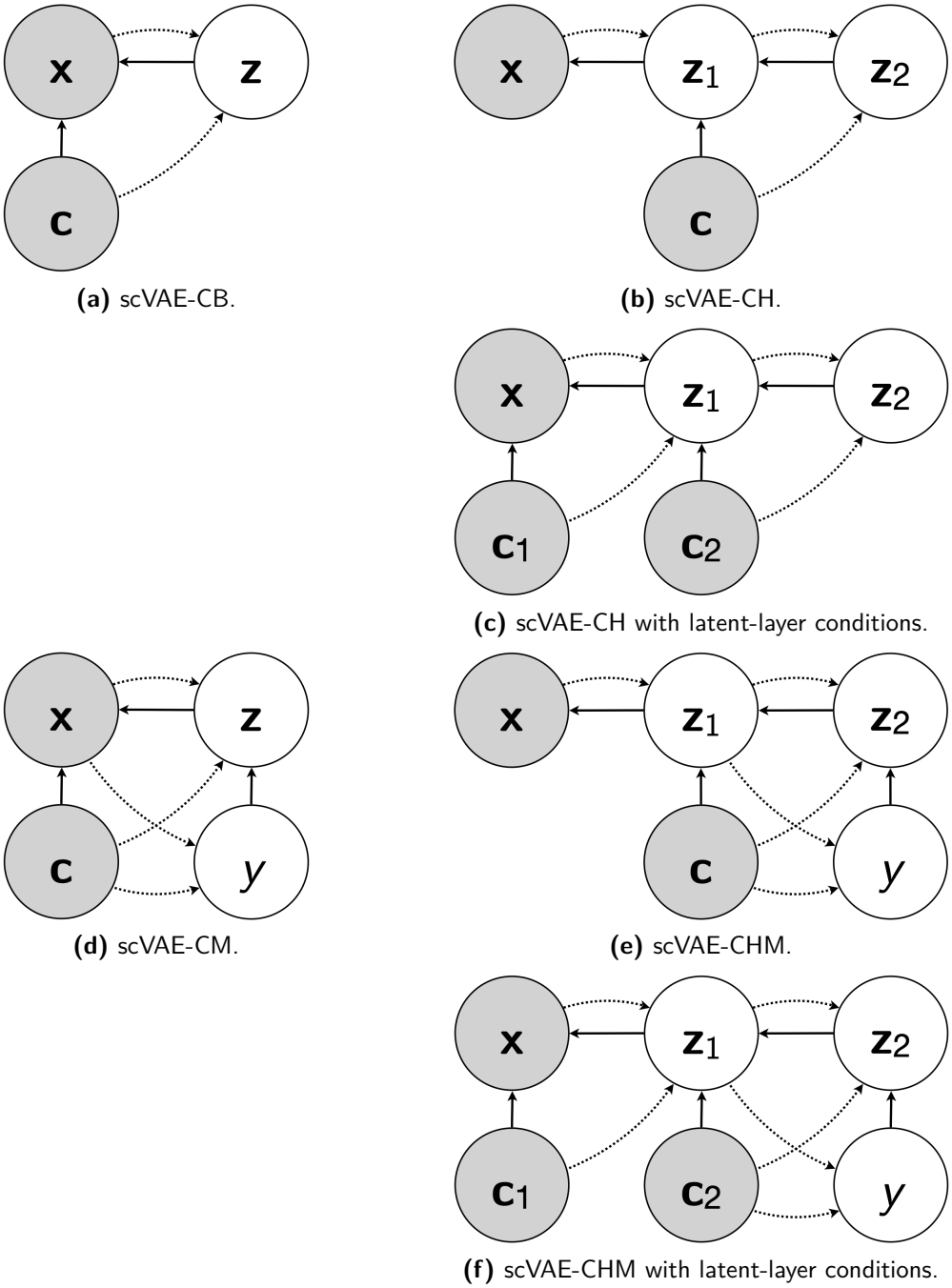
Table S5. Configurations of the optimal model of each scVAE-C method and for each metric for latent-representation covariate removal.

Method	Metric	Likelihood function	Conditioned covariates	Dropout rate
scVAE-CB	$\mathcal{L}$	CNB	Stimulation status	0.1
scVAE-CB	$\widehat{\text{iLISI}}$	FDCNB	Stimulation status	0
scVAE-CM	$\mathcal{L}$	CNB	Stimulation status	0.1
scVAE-CM	$\widehat{\text{iLISI}}$	FDNB	Stimulation status	0
scVAE-CH	$\mathcal{L}$	CNB	Stimulation status	0.1
scVAE-CH	$\widehat{\text{iLISI}}$	CNB	Stimulation status	0.1
scVAE-CHM	$\mathcal{L}$	CNB	Stimulation status	0.1
scVAE-CHM	$\widehat{\text{iLISI}}$	FDCZINB	Stimulation status	0.1

Table S6. Configurations of the optimal model of each scVAE-C method and for each metric for counterfactual prediction.

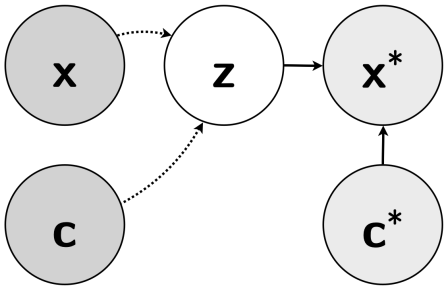
Method	Metric	Likelihood function	Conditioned covariates	Dropout rate
scVAE-CB	$\overline{\mathcal{L}}$	NB	Stimulation status, library size	0.1
scVAE-CB	$\overline{R^2_{\text{all}}}$	FDCZINB	Stimulation status	0
scVAE-CB	$\overline{R^2_{\text{top}}}$	ZINB	Stimulation status	0
scVAE-CM	$\overline{\mathcal{L}}$	CNB	Stimulation status	0.1
scVAE-CM	$\overline{R^2_{\text{all}}}$	FDCZINB	Stimulation status	0.1
scVAE-CM	$\overline{R^2_{\text{top}}}$	FDCNB	Stimulation status	0.1
scVAE-CH	$\overline{\mathcal{L}}$	CNB	Stimulation status	0.1
scVAE-CH	$\overline{R^2_{\text{all}}}$	NB	Stimulation status, library size	0
scVAE-CH	$\overline{R^2_{\text{top}}}$	NB	Stimulation status, library size	0
scVAE-CHM	$\overline{\mathcal{L}}$	CNB	Stimulation status	0.1
scVAE-CHM	$\overline{R^2_{\text{all}}}$	FDNB	Stimulation status, library size	0
scVAE-CHM	$\overline{R^2_{\text{top}}}$	FDNB	Stimulation status, library size	0

S9 Supplementary Figures

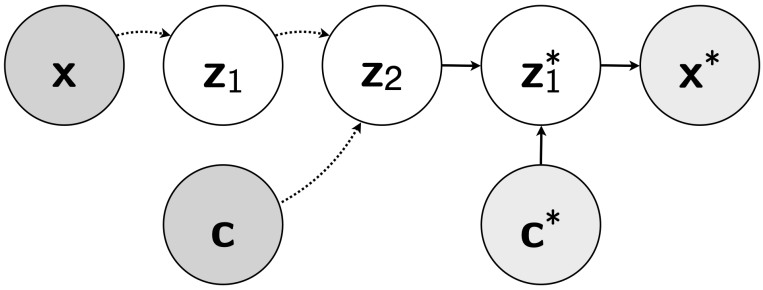


(a) scVAE-CB. **(b)** scVAE-CH. **(c)** scVAE-CH with latent-layer conditions. **(d)** scVAE-CM. **(e)** scVAE-CHM. **(f)** scVAE-CHM with latent-layer conditions.

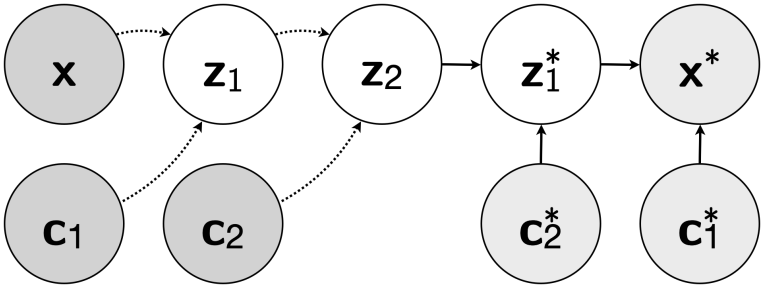
Figure S1. Model graphs of the generative and inference processes for the scVAE-C methods. Circles represent observed (grey) and latent (white) variables, and arrows indicate the directions of the generative (solid) and inference (dashed) processes.



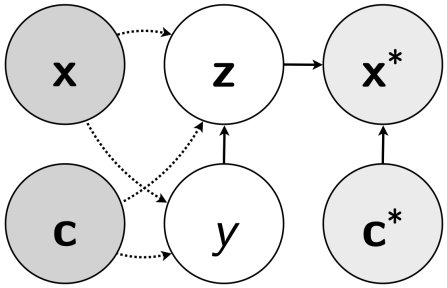
(a) scVAE-CB.



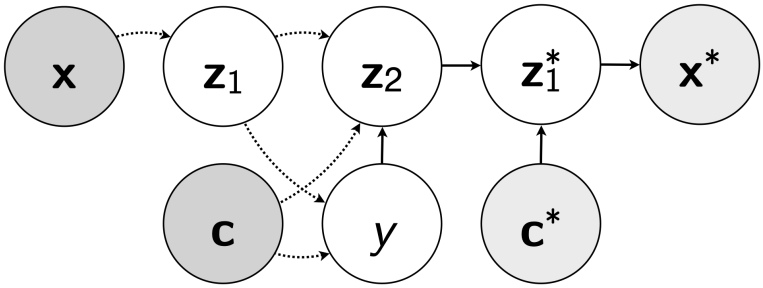
(b) scVAE-CH.



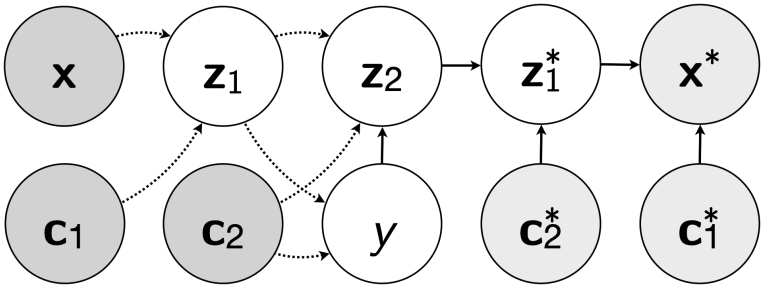
(c) scVAE-CH with latent-layer conditions.



(d) scVAE-CM.



(e) scVAE-CHM.



(f) scVAE-CHM with latent-layer conditions.

Figure S2. Model graphs of counterfactual prediction for the scVAE-C methods. Circles represent observed (grey), latent (white), and counterfactual (light grey) variables. Arrows indicate the directions of the inference process (dashed) and the counterfactual generative process (solid).

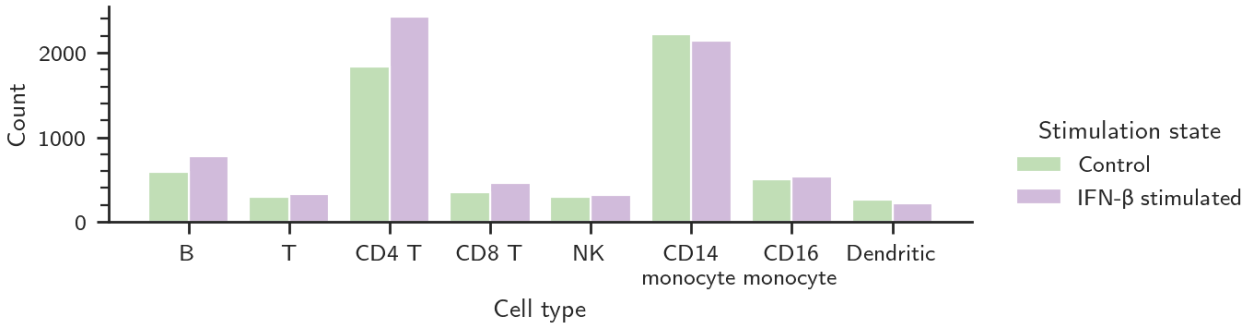


Figure S3. Histogram of stimulated and control cells of different cell types for the Kang data set.

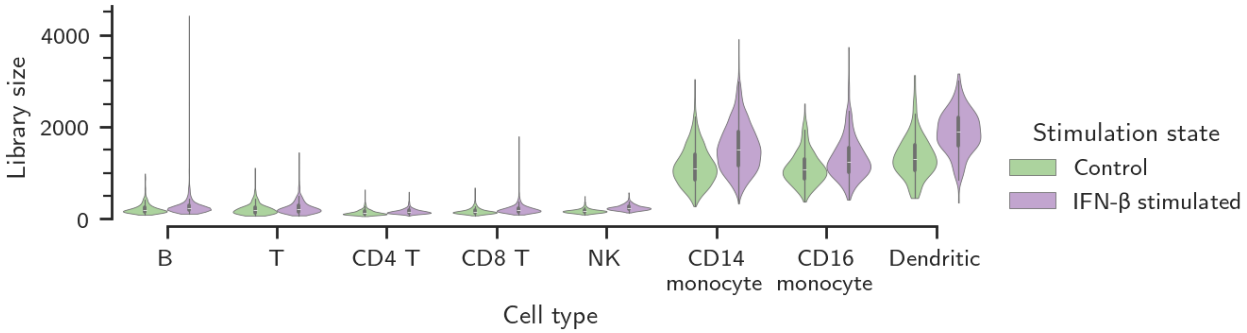


Figure S4. Distributions of library sizes for stimulated and control cells of different cell types for the Kang data set. The kernel density estimate, median, and interquartile range are shown for each group of cells. The kernel density estimates are normalised to display with the same width.

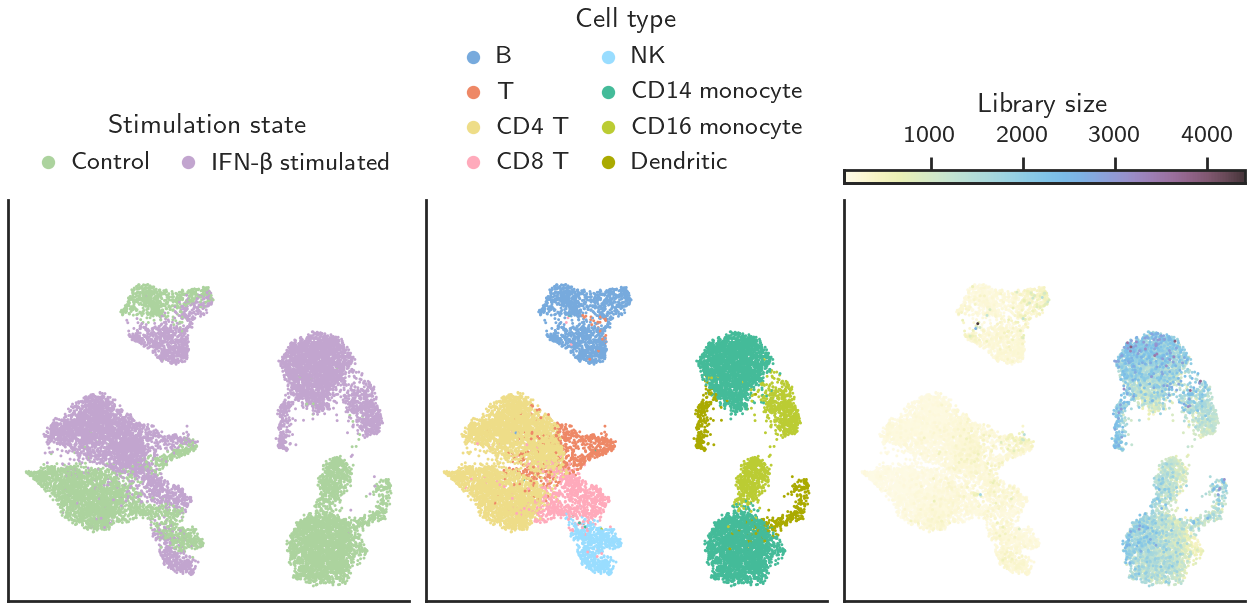


Figure S5. UMAP embedding of the truncated SVD representation of the Kang data set. The embedding is coloured by stimulation state, cell type, and library size.

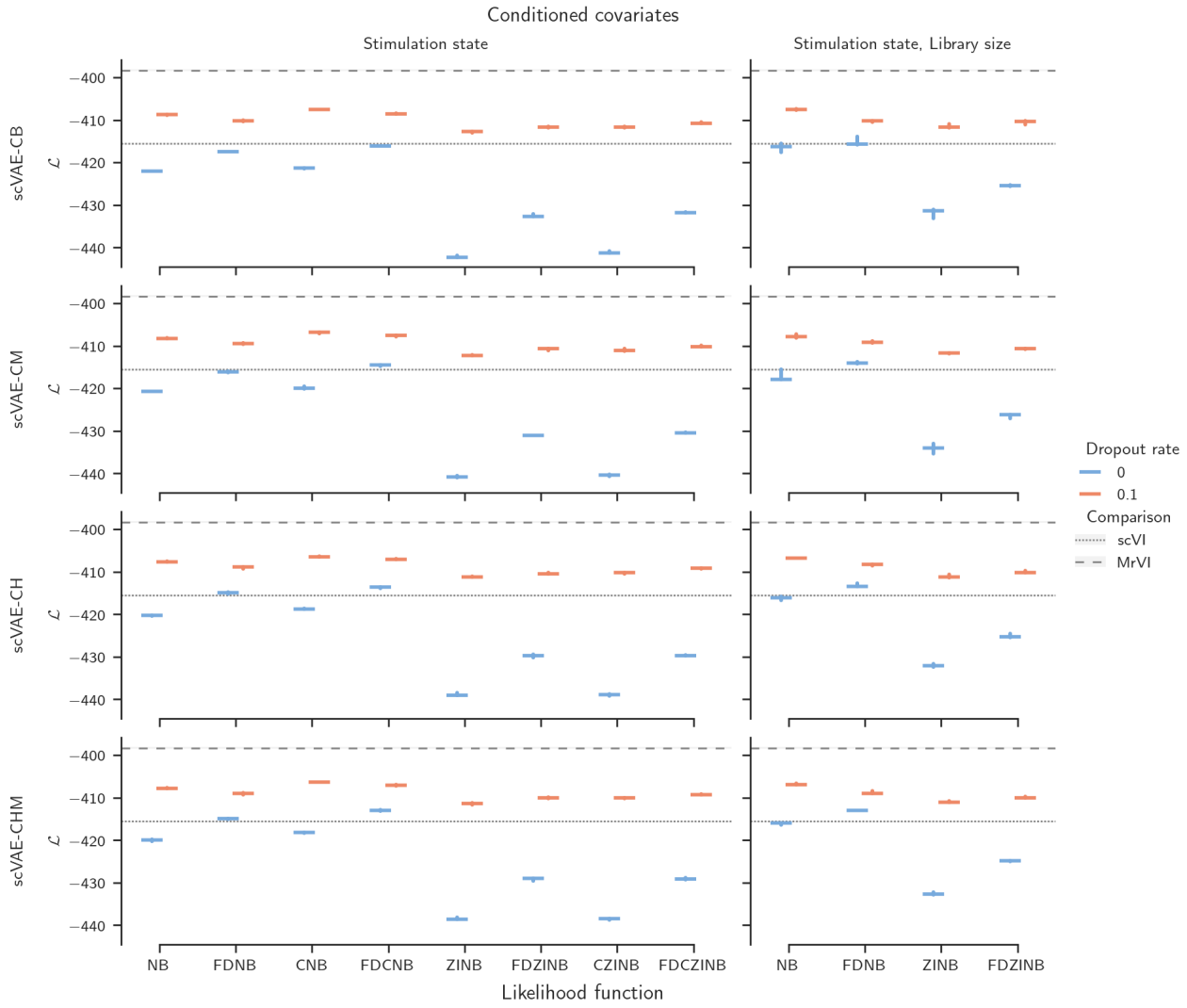


Figure S6. Variational lower bound for all scVAE-C models for latent-representation covariate removal. The median and the interquartile range across replicate models are shown. Results for the comparison methods are also displayed.

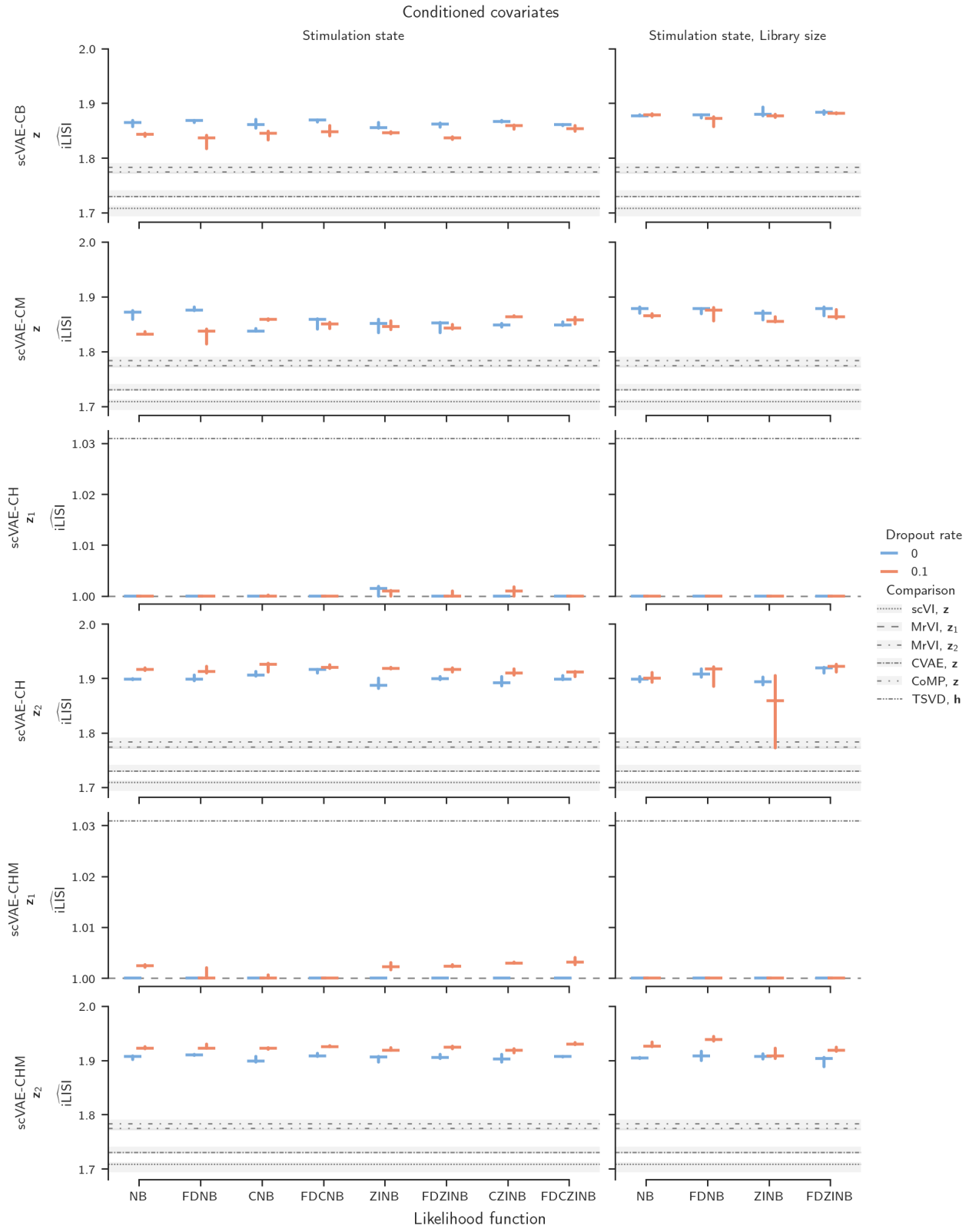


Figure S7. Median integration LISIs for all scVAE-C models for latent-representation covariate removal. The median and the interquartile range across replicate models are shown separately for the latent representation(s) of each method. Results for the comparison methods are also displayed.

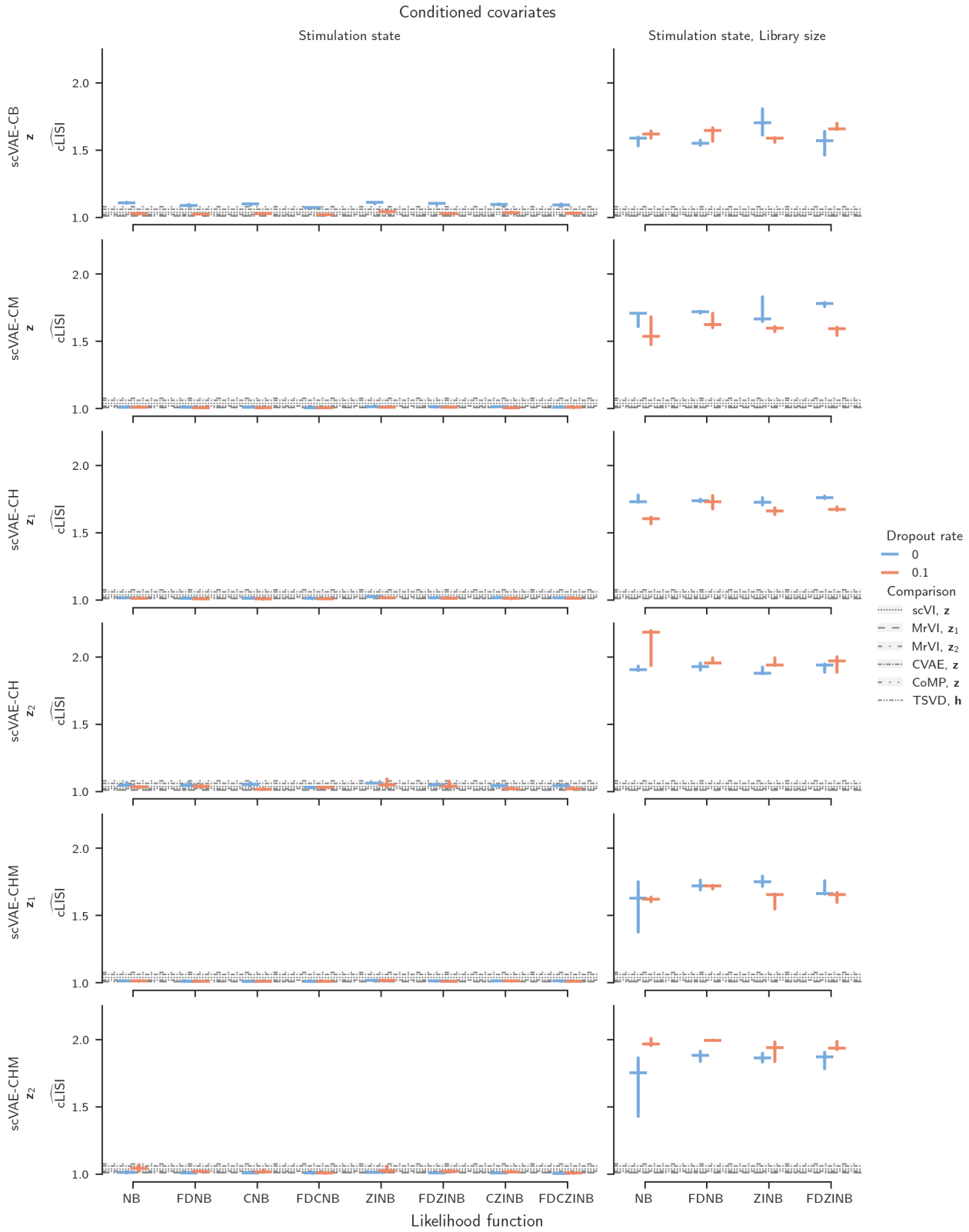


Figure S8. Median cell-type LISIs for all scVAE-C models for latent-representation covariate removal. The median and the interquartile range across replicate models are shown separately for the latent representation(s) of each method. See Supplementary Figure S9 for results for the models only conditioning on stimulation status. Results for the comparison methods are also displayed.

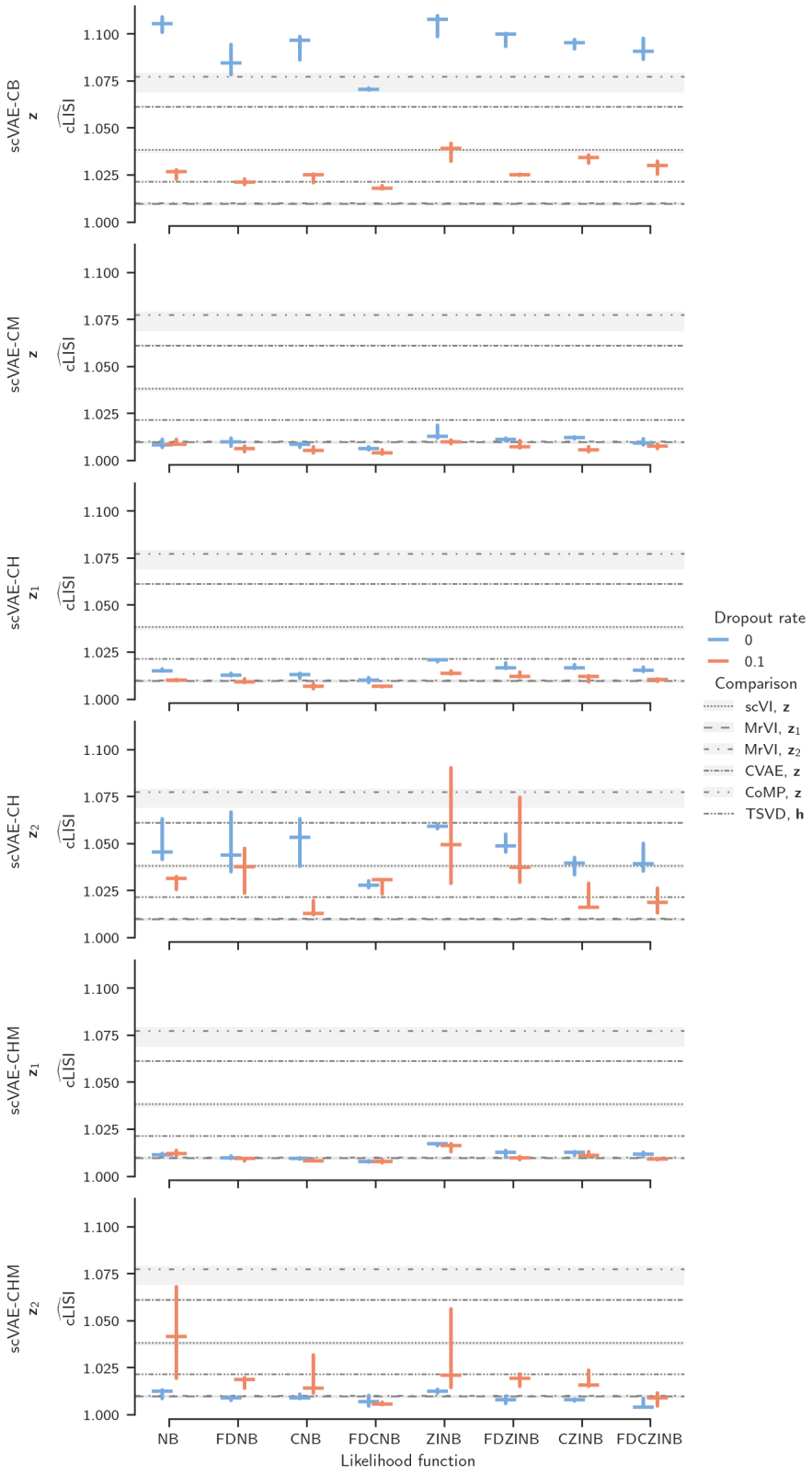
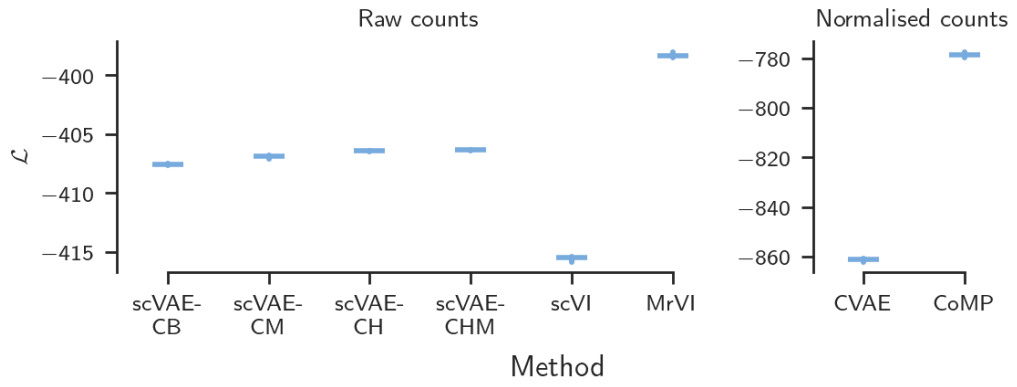
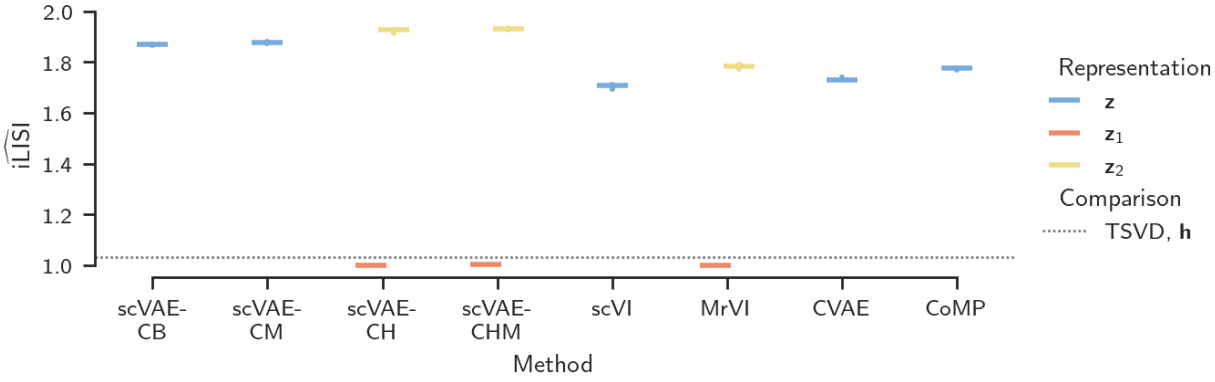


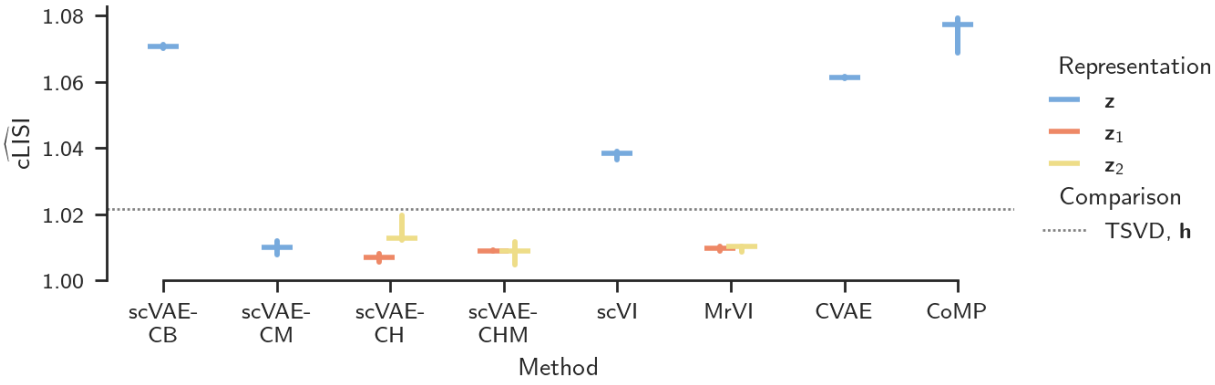
Figure S9. Median cell-type LISIs for all scVAE-C models only conditioned on stimulation status for latent-representation covariate removal. The median and the interquartile range across replicate models are shown separately for the latent representation(s) of each method. Results for the comparison methods are also displayed.



(a) Variational lower bound.



(b) Median integration LISI.



(c) Median cell-type LISI.

Figure S10. Metrics for latent-representation covariate removal. The median and the interquartile range across replicate models of each method are shown. For each metric, optimal models for the scVAE-C methods were selected using the metric itself, except for the median cell-type LISI for which the median integration LISI was used.

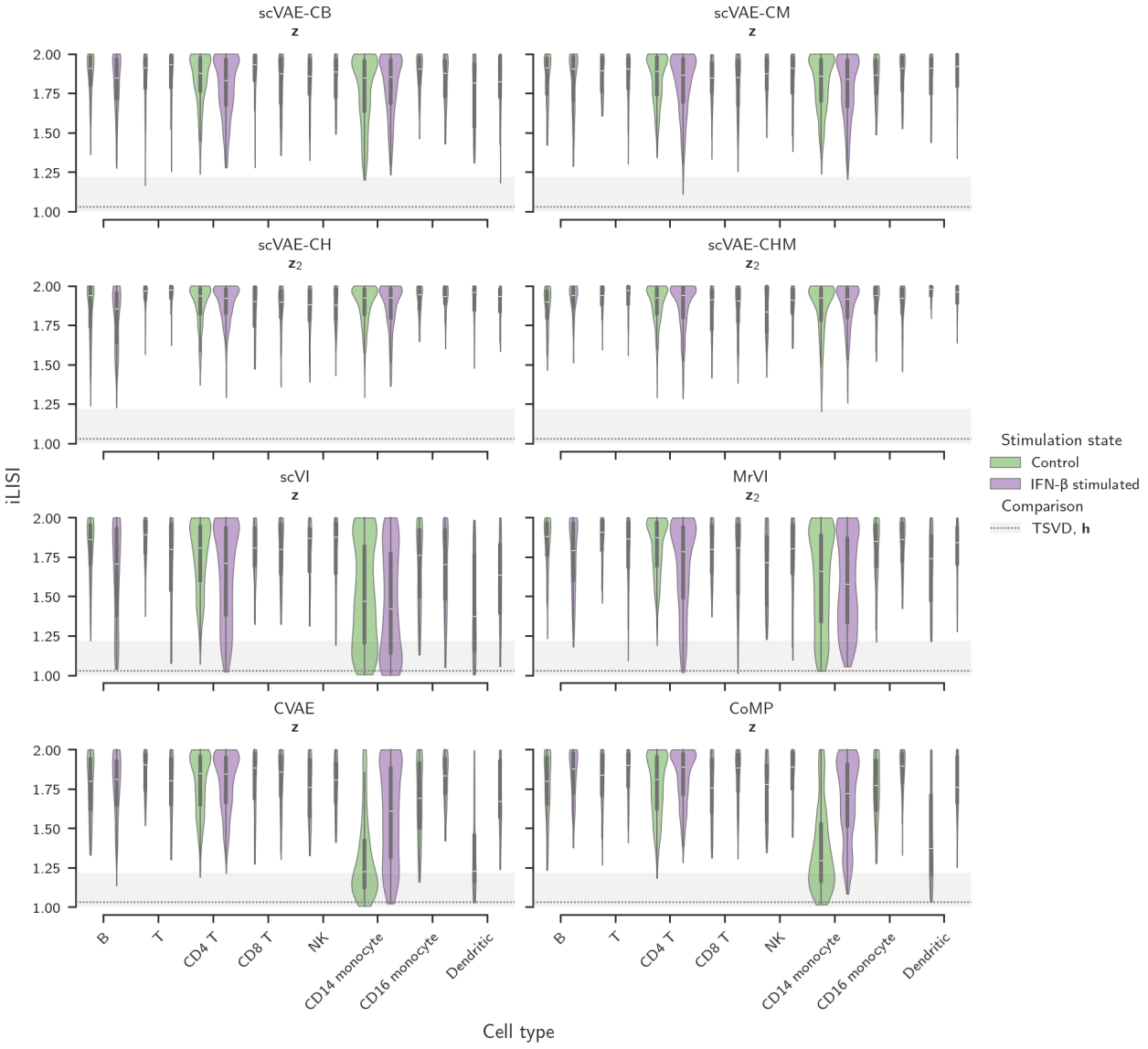
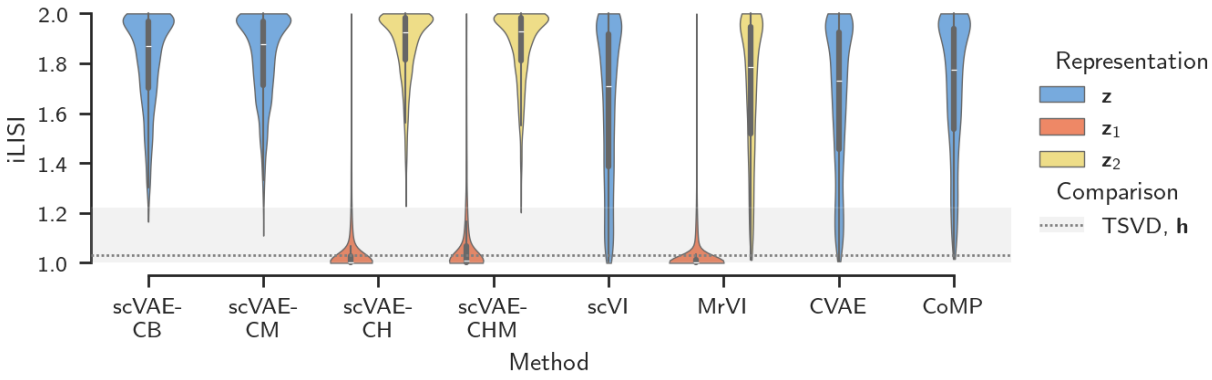
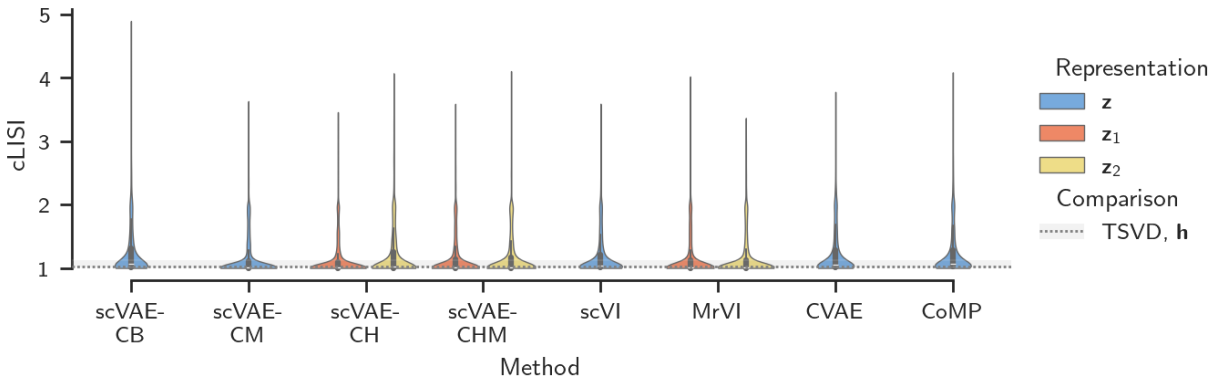


Figure S11. Distributions of cell-level iLISIs across cell types for latent-representation covariate removal. For the stimulation-state-conditioned latent representation of each method, the kernel density estimate, median, and interquartile range are shown for the median-performing replicate model. Optimal models were used for the scVAE-C methods. The kernel density estimates were normalised by the cell count of each cell group. The median and the interquartile range across all cells for the decomposed representation of the baseline TSVD method are also shown.



(a) Integration LISI (reproduced for easy comparison).



(b) Cell-type LISI.

Figure S12. Distributions of cell-level LISIs for latent-representation covariate removal. For the latent representation(s) of each method, the kernel density estimate, median, and interquartile range are shown for the median-performing replicate model. Optimal models were used for the scVAE-C methods. The kernel density estimates were normalised to display with the same area. The median and the interquartile range for the decomposed representation of the baseline TSVD method are also shown.

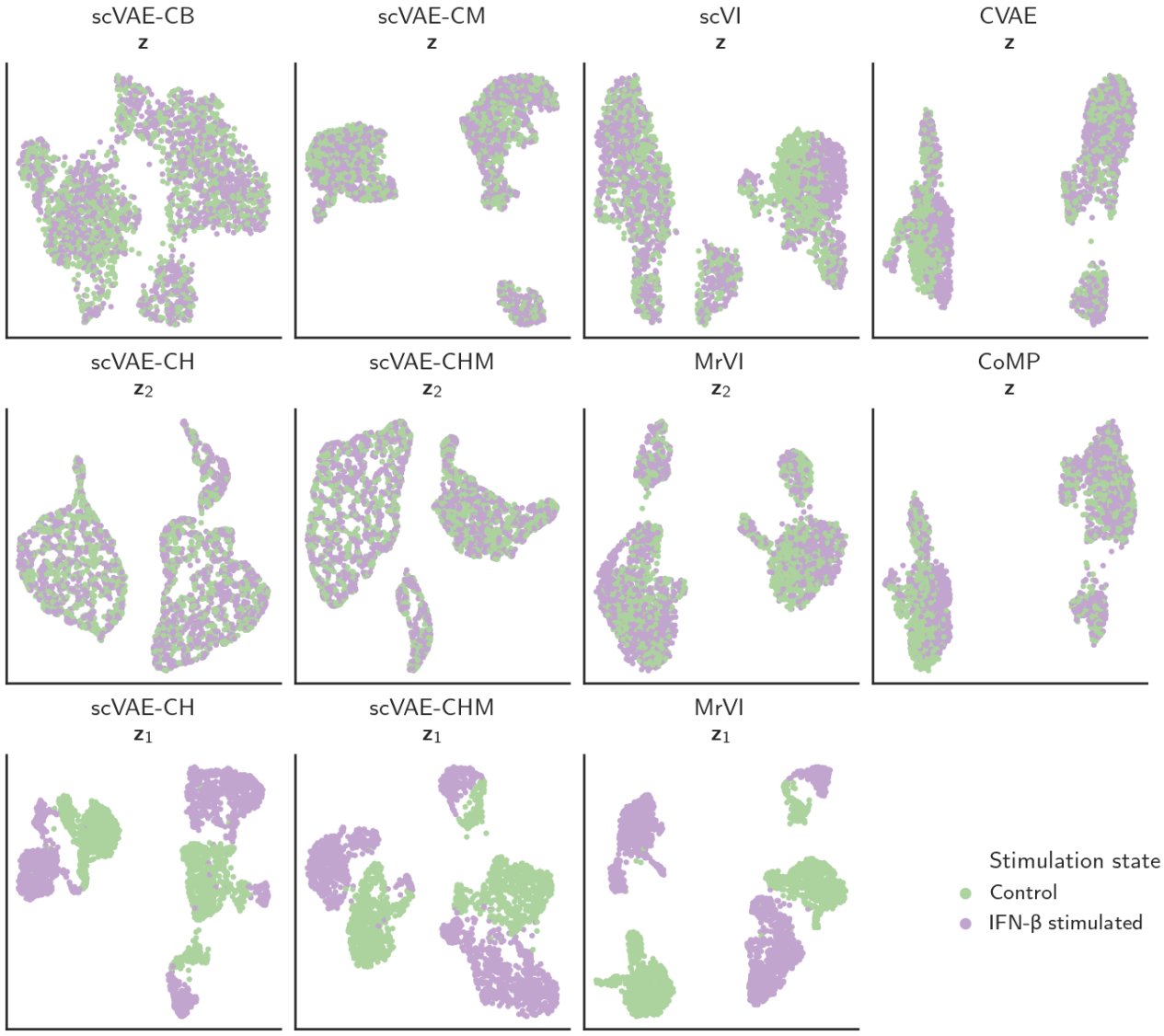


Figure S13. UMAP embeddings of latent representations for latent-representation covariate removal coloured by stimulation state. Embeddings are shown for the median-performing replicate model, and the optimal models for the scVAE-C methods were selected using the median iLISI.

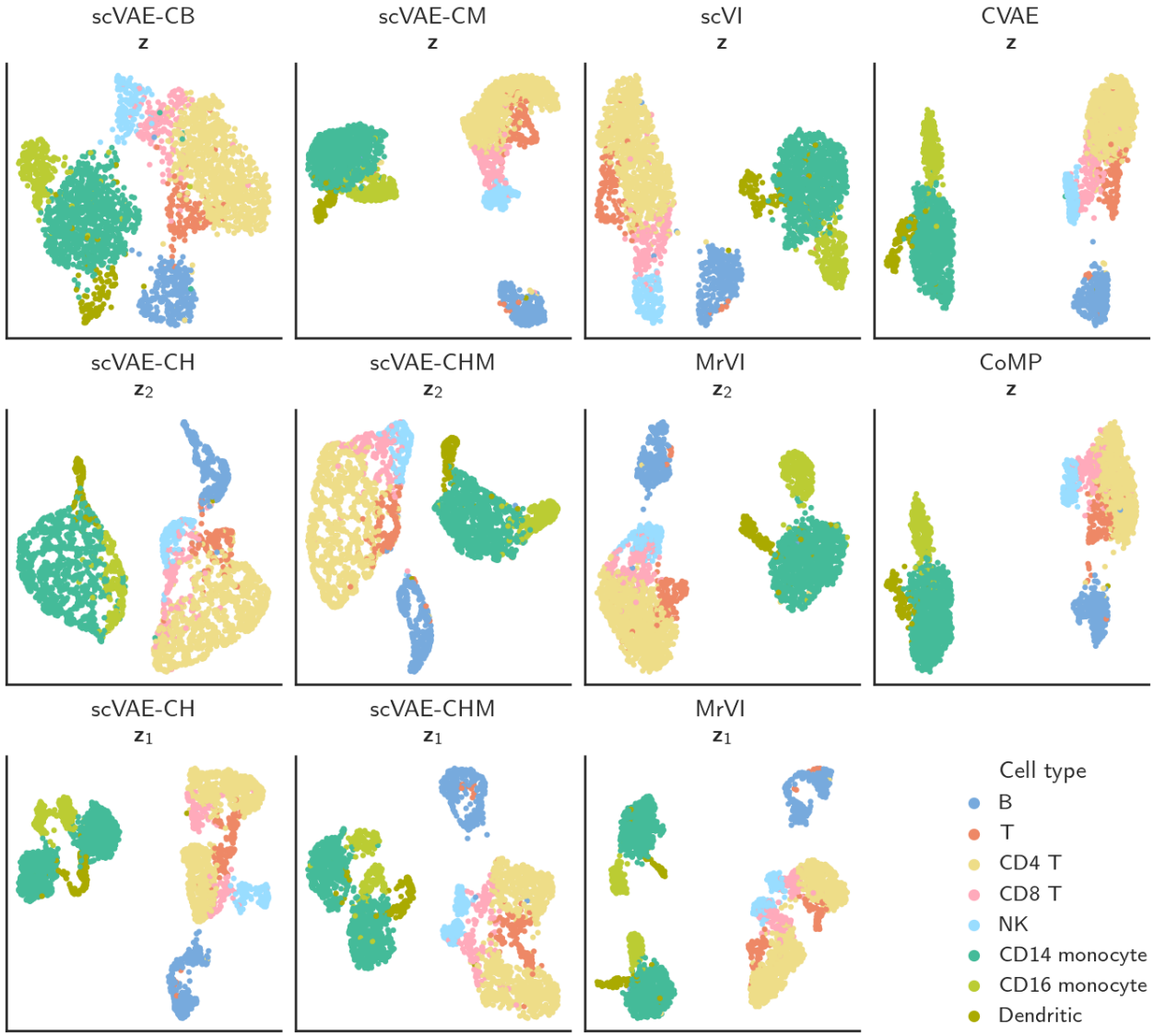


Figure S14. UMAP embeddings of latent representations for latent-representation covariate removal coloured by cell type. Embeddings are shown for the median-performing replicate model, and the optimal models for the scVAE-C methods were selected using the median iLISI.

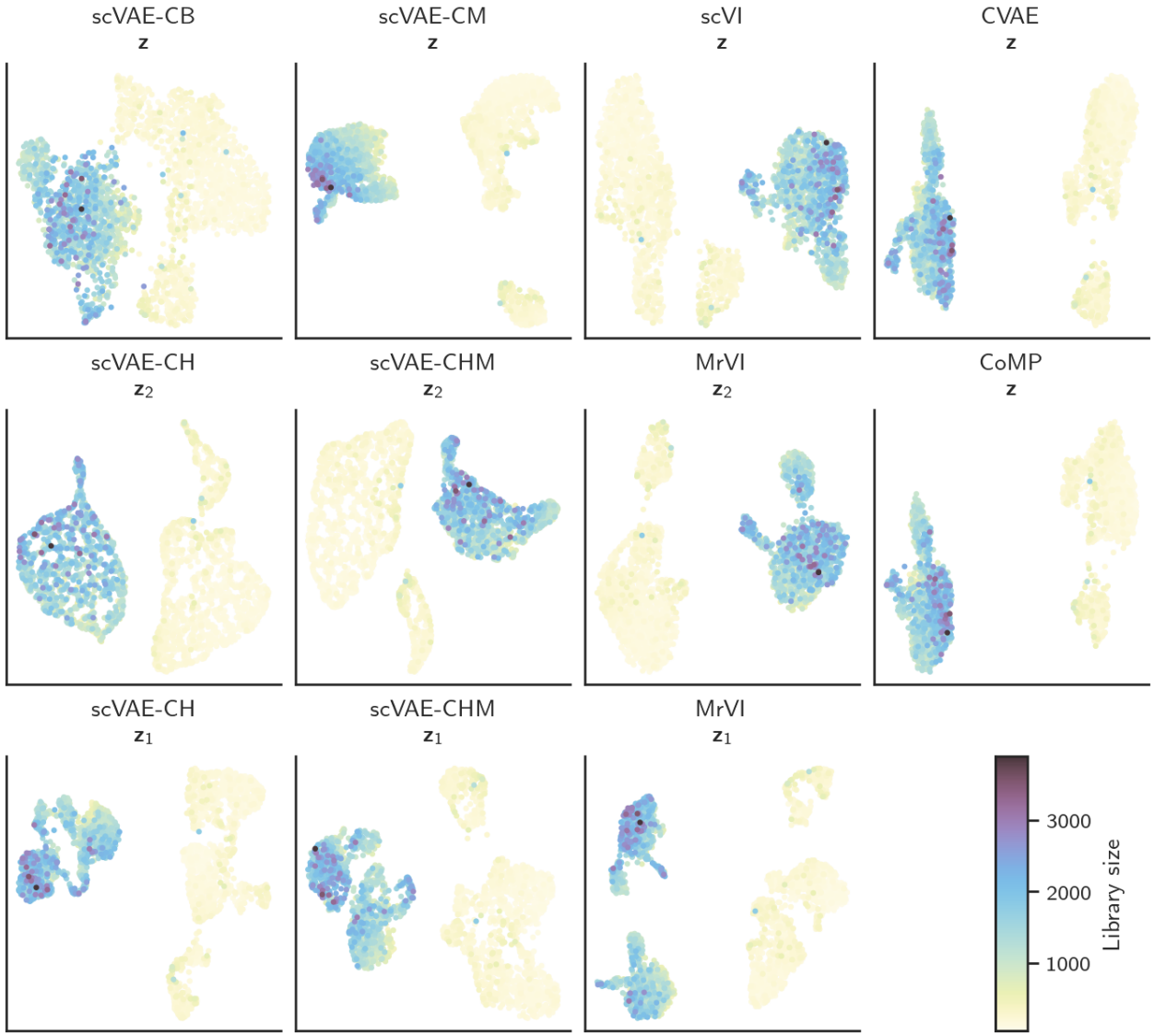
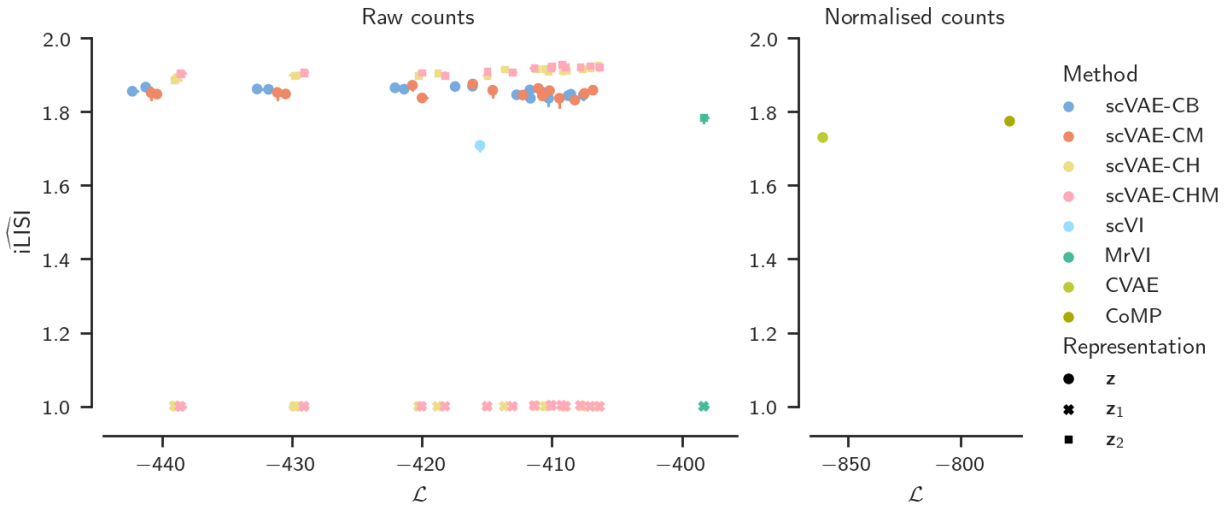
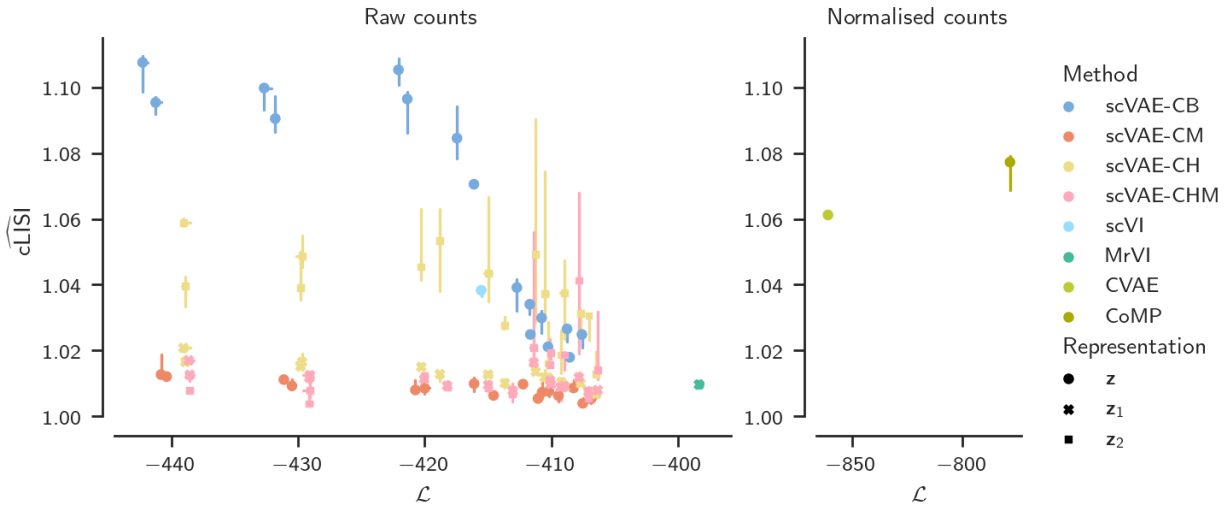


Figure S15. UMAP embeddings of latent representations for latent-representation covariate removal coloured by library size. Embeddings are shown for the median-performing replicate model, and the optimal models for the scVAE-C methods were selected using the median iLISI.



(a) Median integration LISI.



(b) Median cell-type LISI.

Figure S16. Relationship between LISI medians and variational lower bound for latent-representation covariate removal. The median and the interquartile range across replicate models are shown separately for the latent representation(s) of each method.

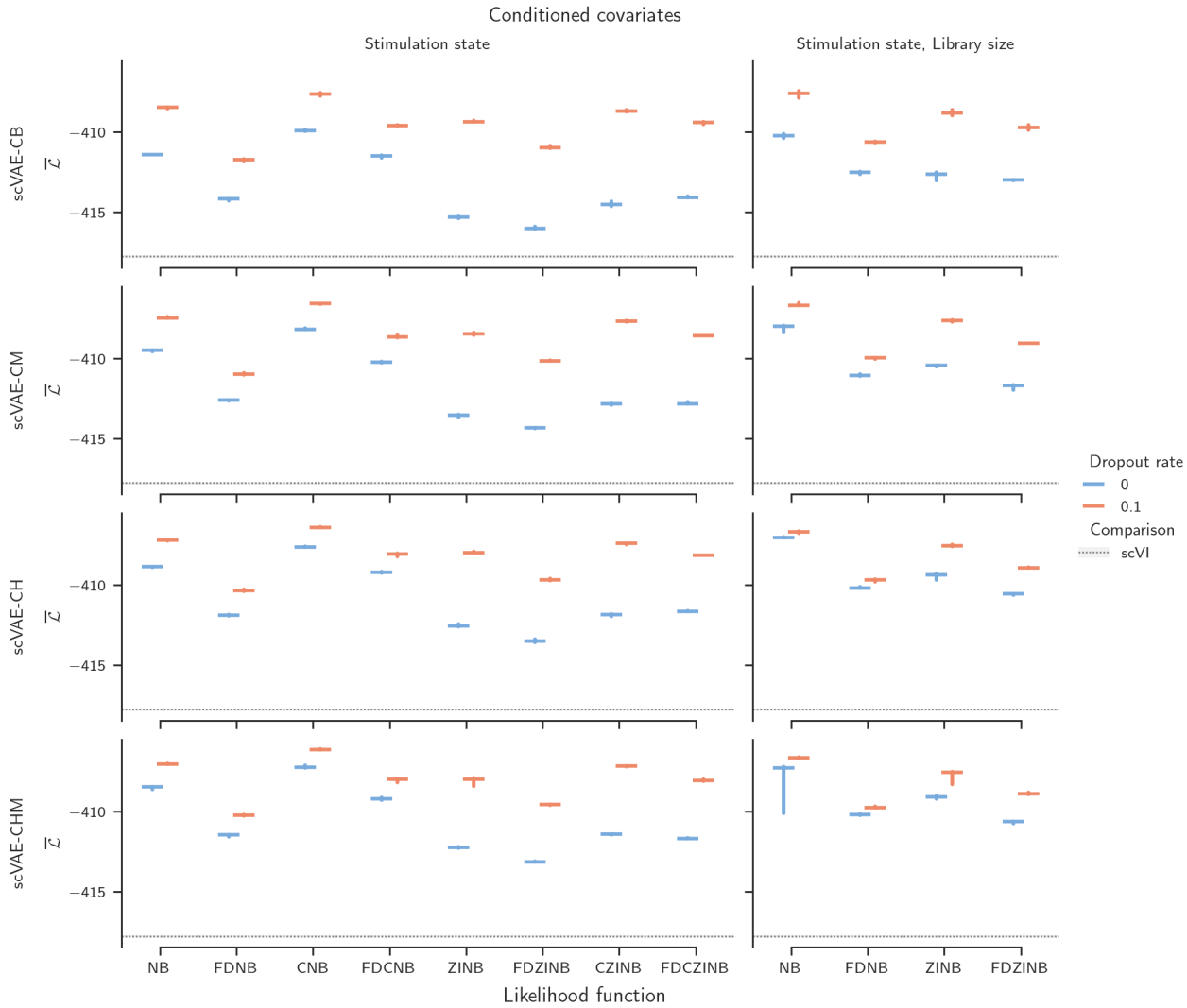


Figure S17. Mean variational lower bound for all scVAE-C models for counterfactual prediction. The median and the interquartile range across replicate models are shown. Results for the comparison methods are also displayed.

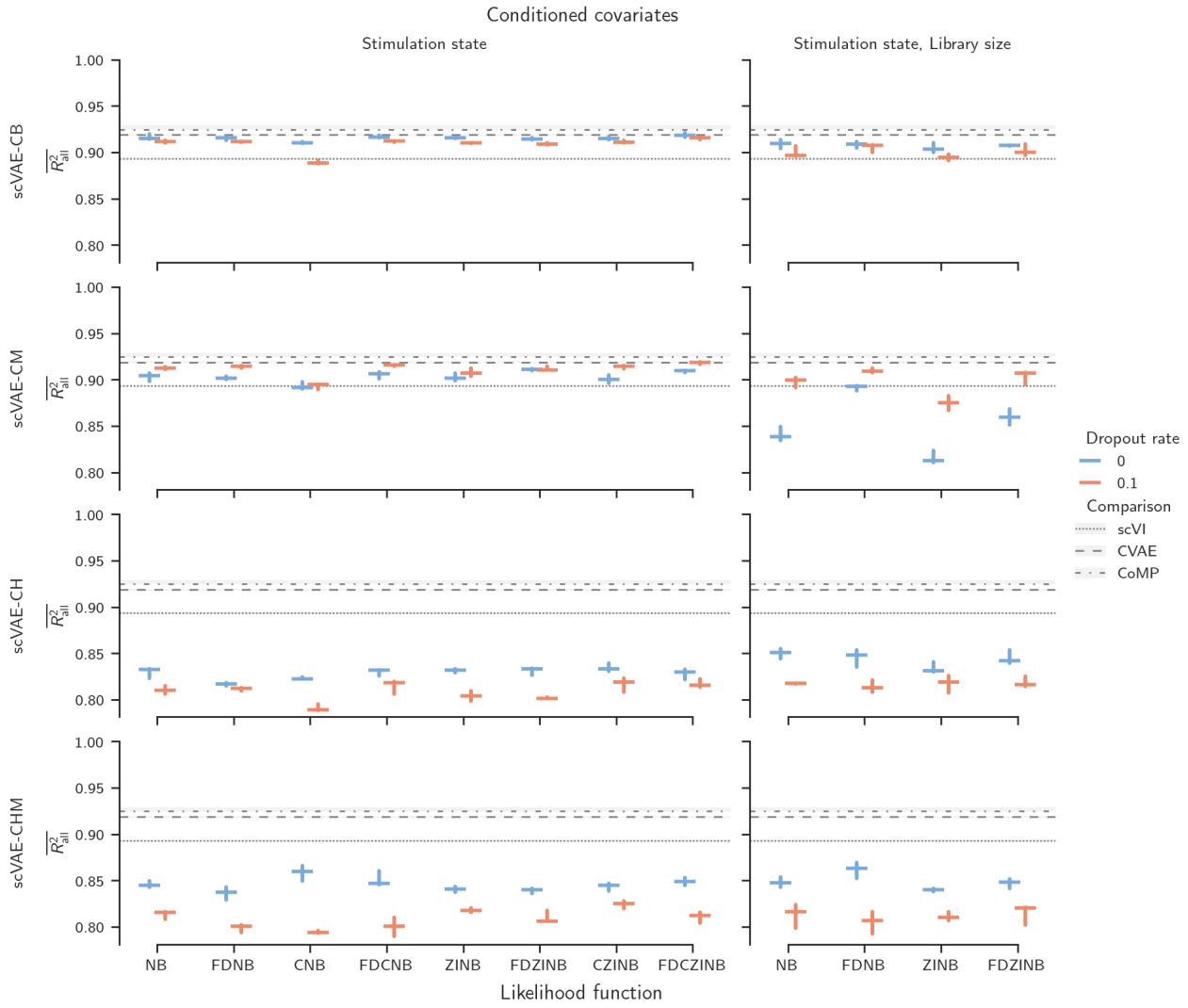


Figure S18. Mean coefficient of determination using all genes for all scVAE-C models for counterfactual prediction. The median and the interquartile range across replicate models are shown. Results for the comparison methods are also displayed.

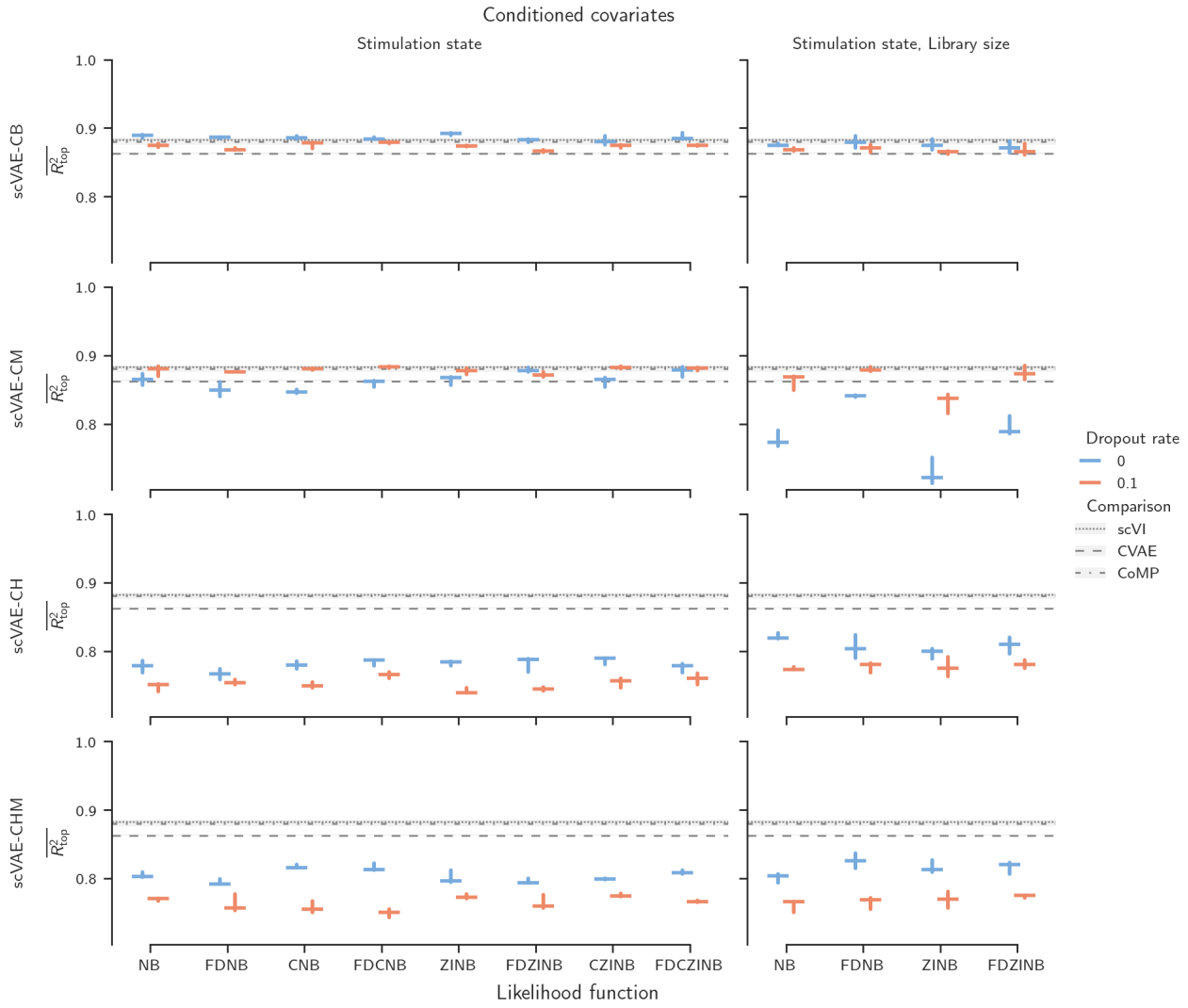
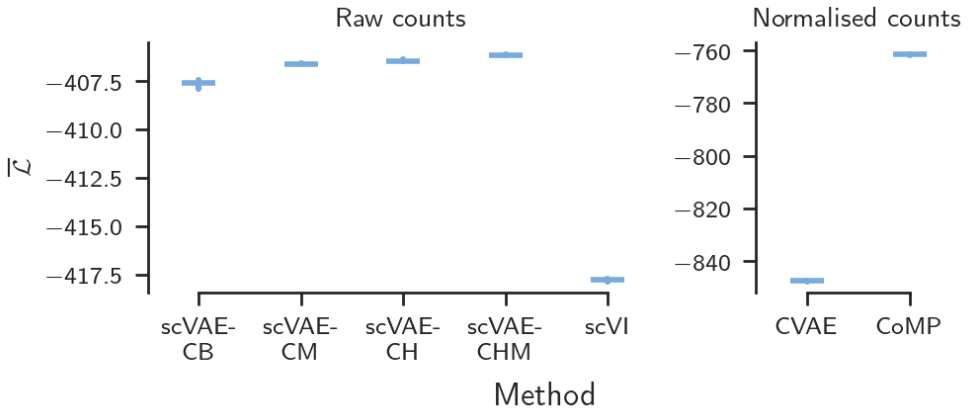
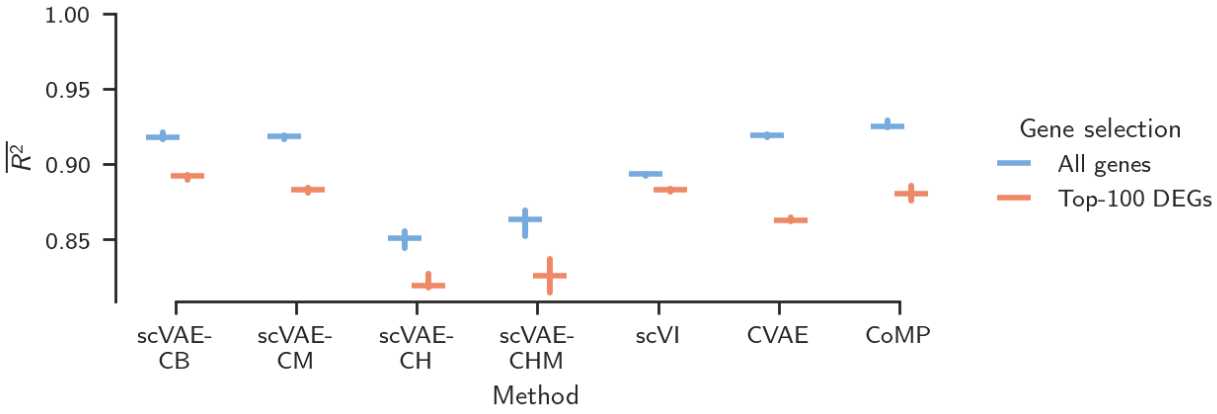


Figure S19. Mean coefficient of determination using the top-100 DEGs for all scVAE-C models for counterfactual prediction. The median and the interquartile range across replicate models are shown. Results for the comparison methods are also displayed.



(a) Mean variational lower bound.



(b) Mean coefficients of determination.

Figure S20. Metrics for counterfactual prediction. The median and the interquartile range across replicate models of each method are shown. For each metric, optimal models for the scVAE-C methods were selected the using the metric itself (separately for each gene selection for the mean coefficient of determination).

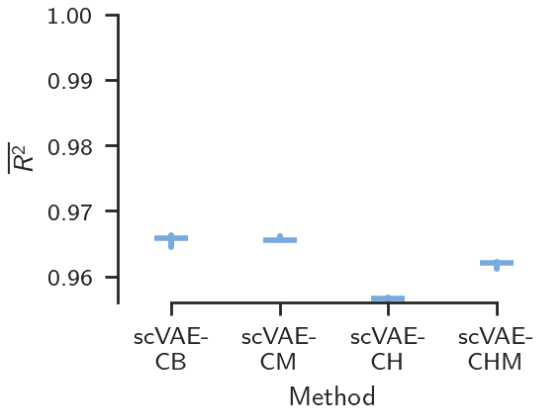


Figure S21. Optimal mean coefficient of determination for reconstruction. The median and the interquartile range across replicate models of each method are shown. Optimal models for the scVAE-C methods were selected by the metric itself. No cells were withheld during training, and mean coefficient of determination for each method were taken across the cell groups detailed in Supplementary Figure S22.

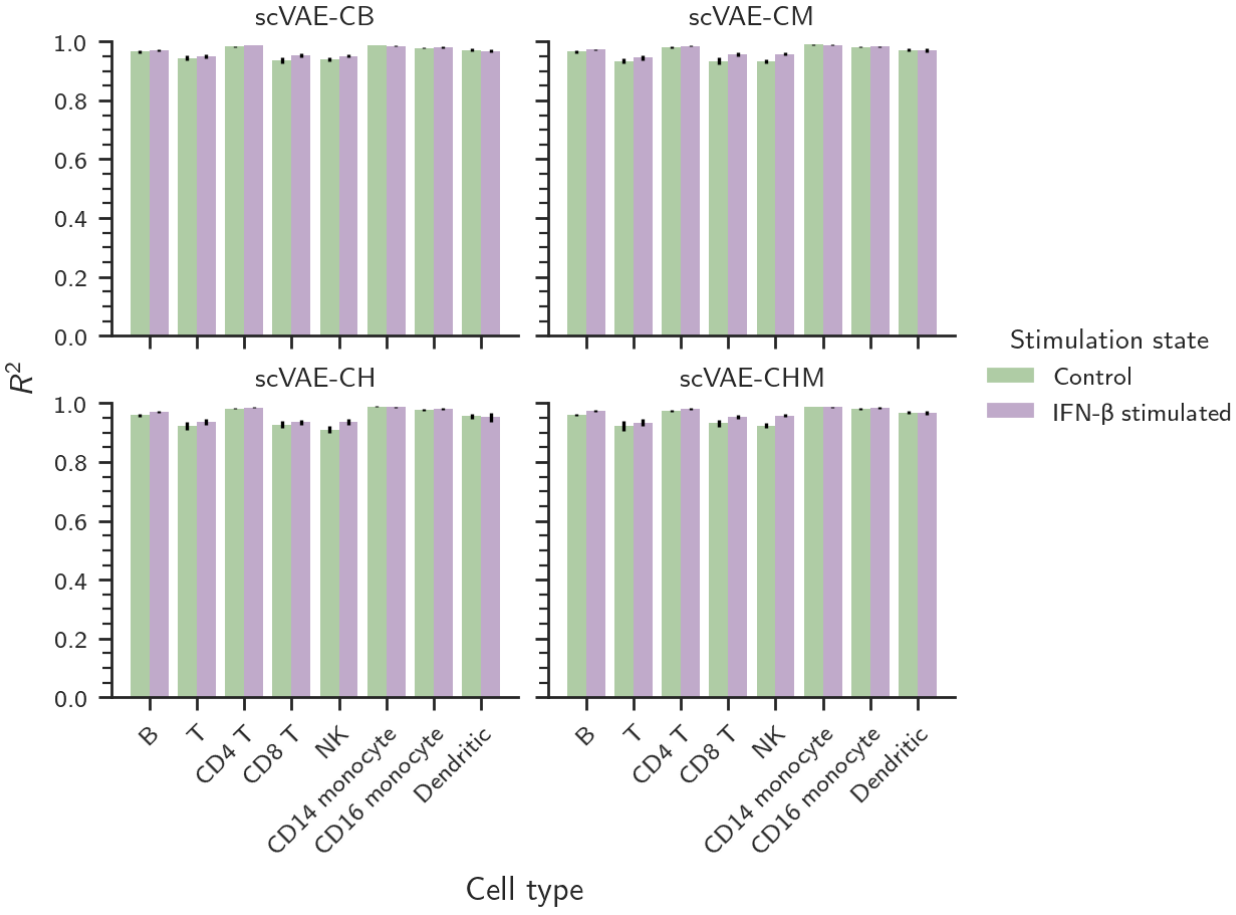
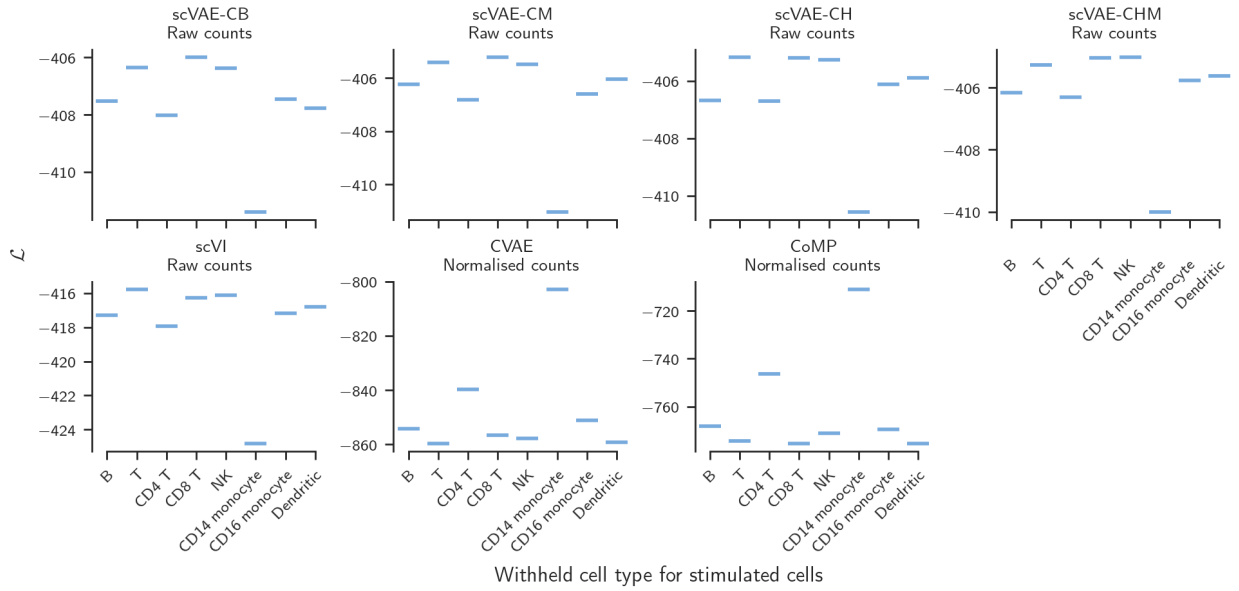
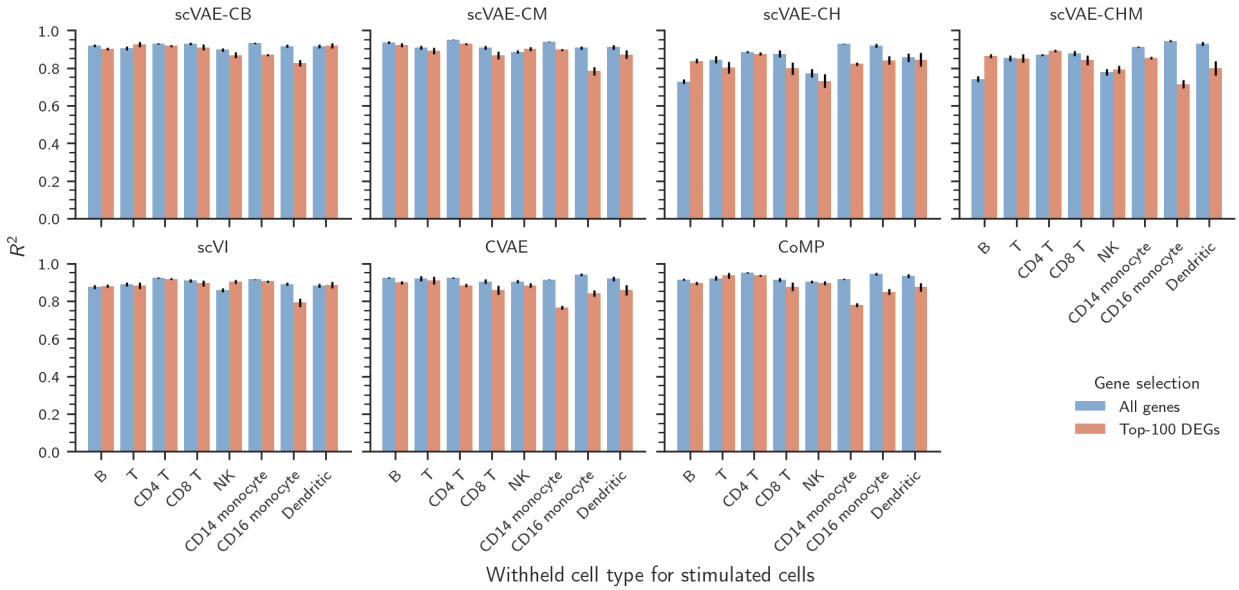


Figure S22. Distributions of coefficients of determination across cell type and stimulation state for reconstruction. All models used for reconstruction as the same as the models used for latent-representation covariate removal. As such, no cells were withhold during training. The coefficient of determination is computed by comparing the reconstructed mean gene expression level for each gene across each group of stimulated or control cells for each cell type to true mean expression level. The mean and the standard deviation are shown for each group of cells and each method using the median-performing replicate. Optimal models for the scVAE-C methods were selected by the metric itself.



(a) Variational lower bound.



(b) Coefficients of determination (reproduced for easy comparison).

Figure S23. Distributions of metrics across withheld cell types for counterfactual prediction. For each cell type and method, results are shown for the median-performing replicate model, and the mean and the standard deviation are displayed for the coefficients of determination. Optimal models for the scVAE-C methods were selected for each metric the using the metric itself (separately for each gene selection for the mean coefficient of determination).

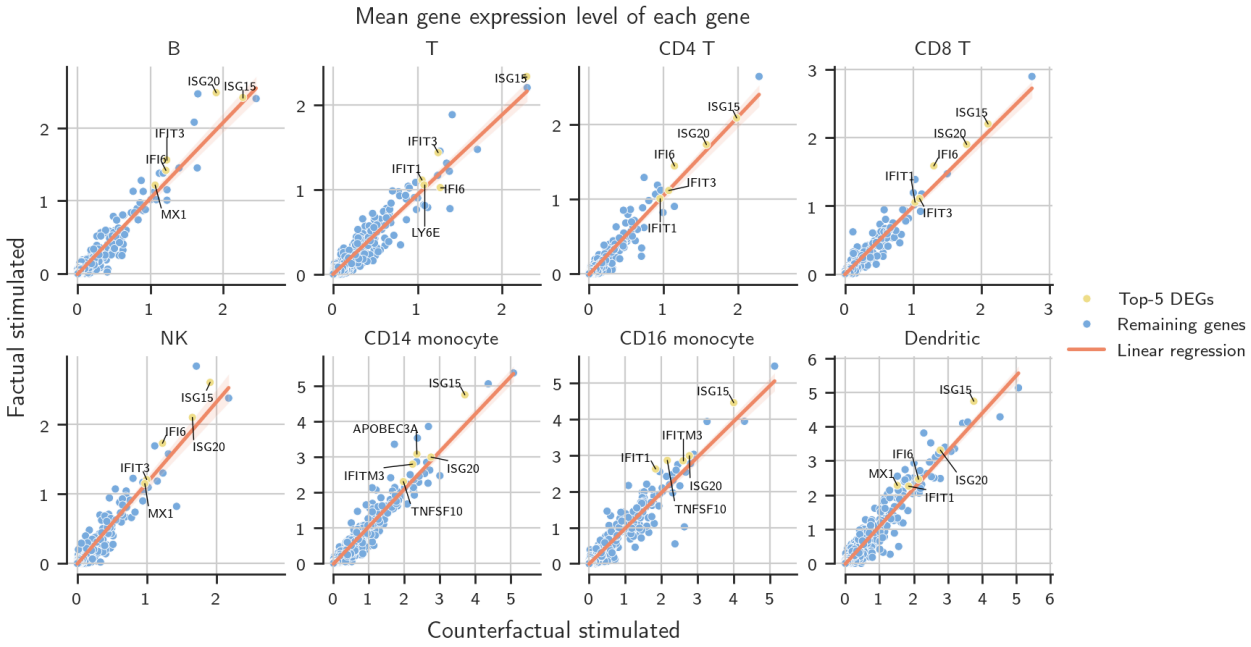


Figure S24. Regression plots for counterfactual prediction using scVAE-CB. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times. The optimal scVAE-CB model was selected by the mean coefficient of determination using all genes.

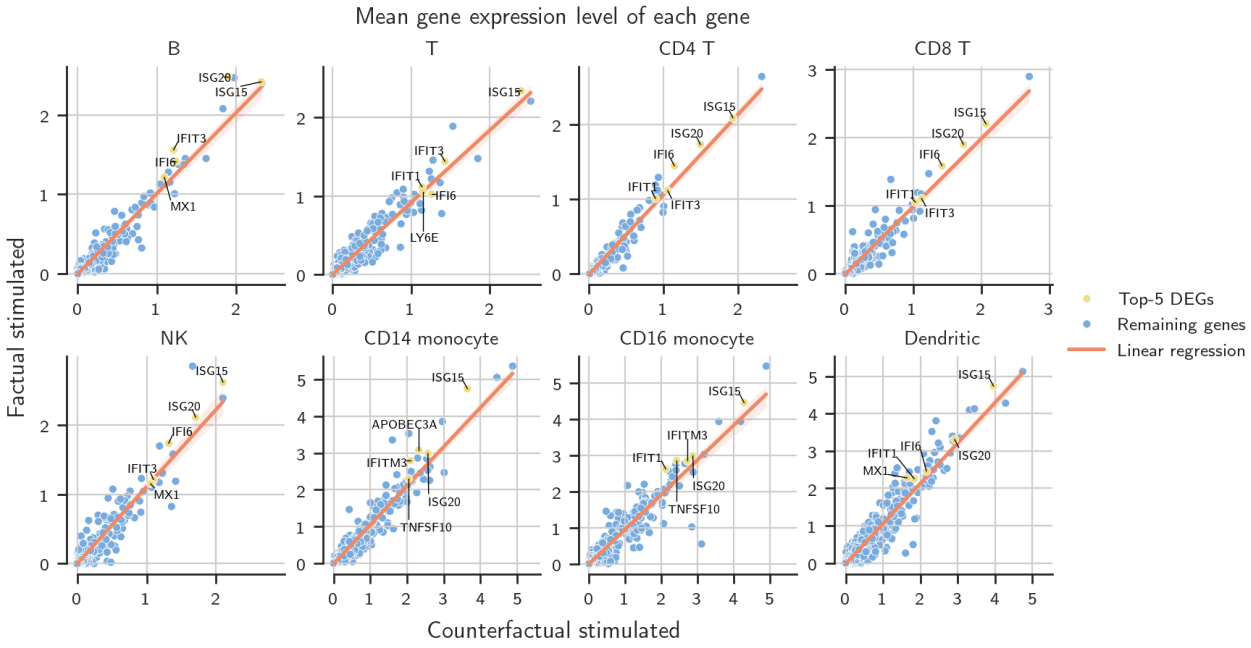


Figure S25. Regression plots for counterfactual prediction using scVAE-CM. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times. The optimal scVAE-CM model was selected by the mean coefficient of determination using all genes.

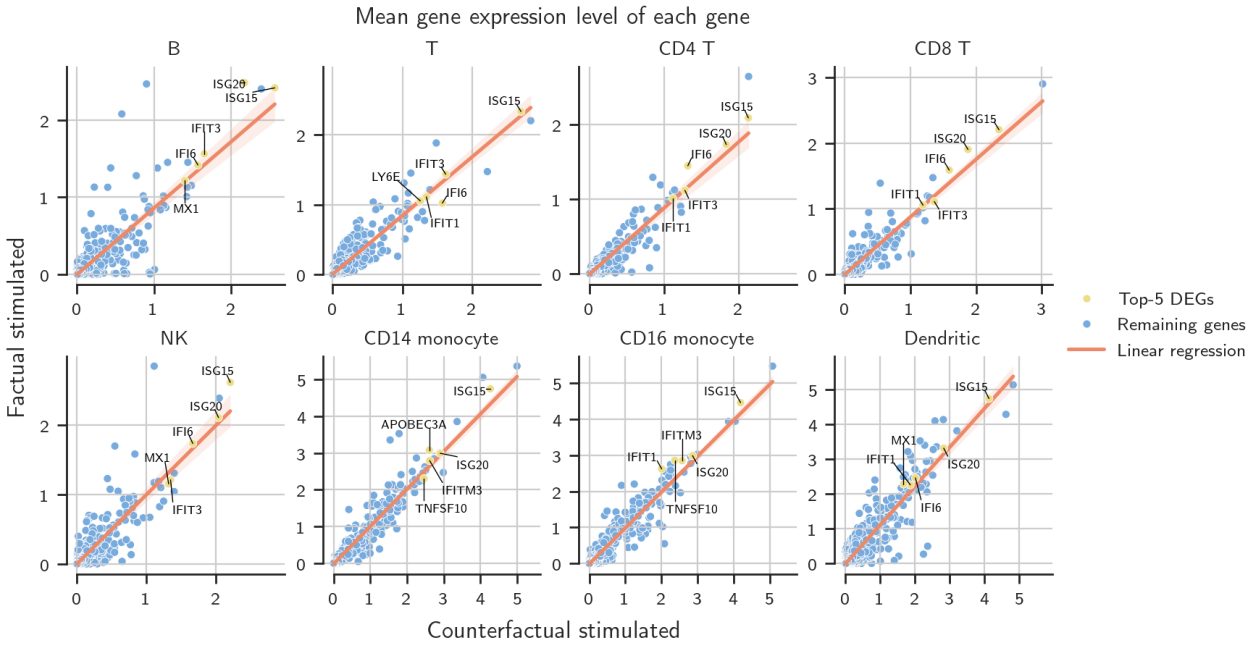


Figure S26. Regression plots for counterfactual prediction using scVAE-CH. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times. The optimal scVAE-CH model was selected by the mean coefficient of determination using all genes.

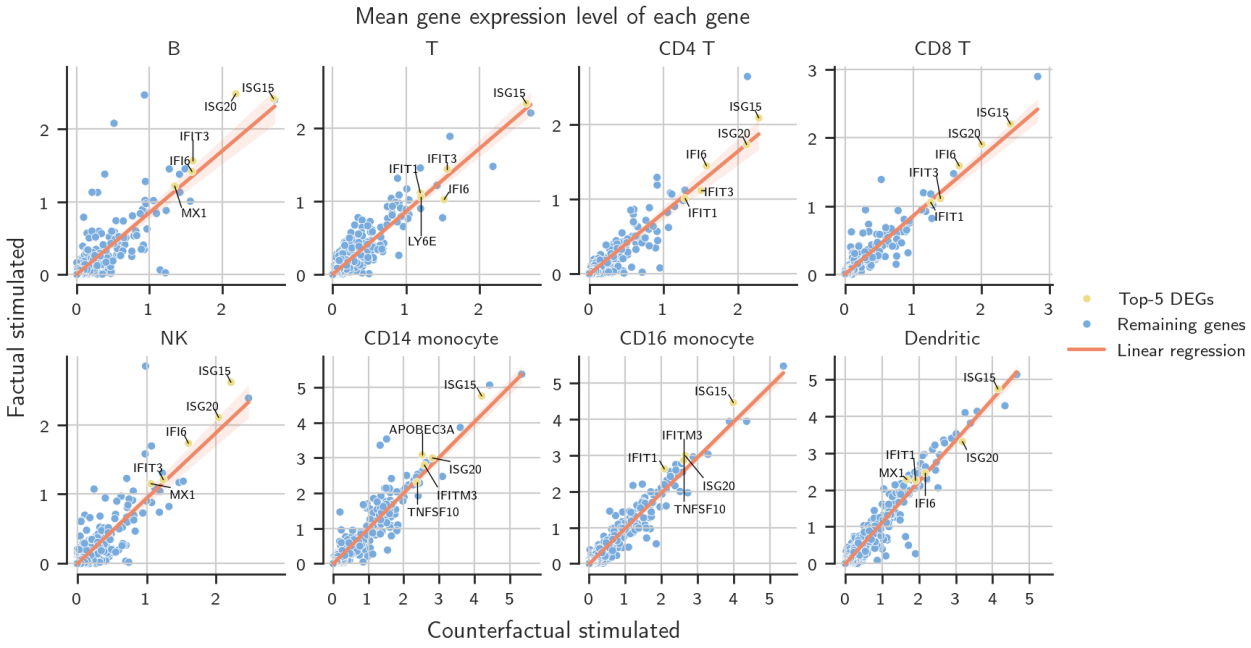


Figure S27. Regression plots for counterfactual prediction using scVAE-CHM. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times. The optimal scVAE-CHM model was selected by the mean coefficient of determination using all genes.

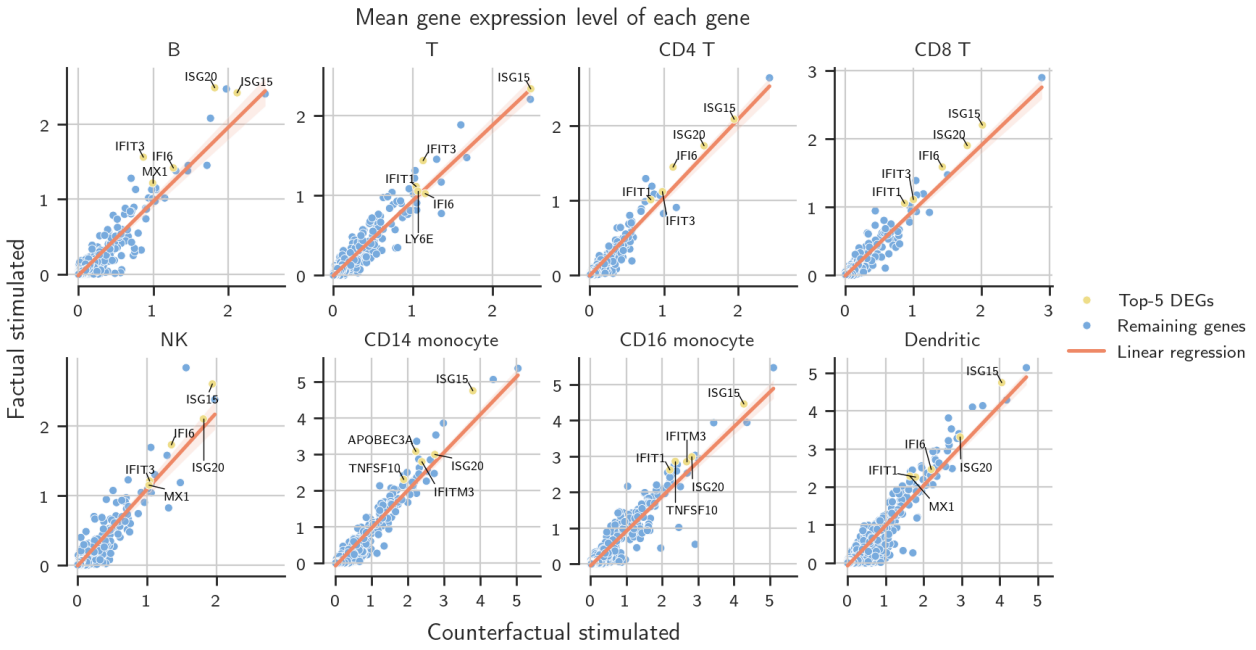


Figure S28. Regression plots for counterfactual prediction using scVI. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times.

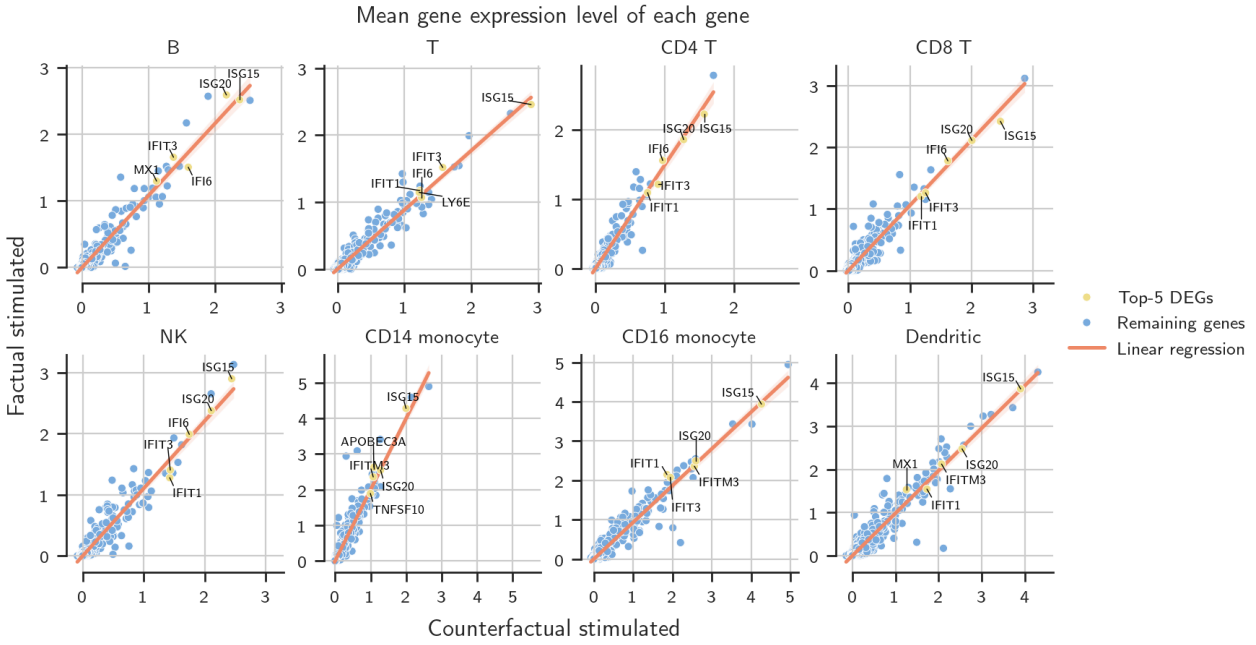


Figure S29. Regression plots for counterfactual prediction using CVAE. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times.

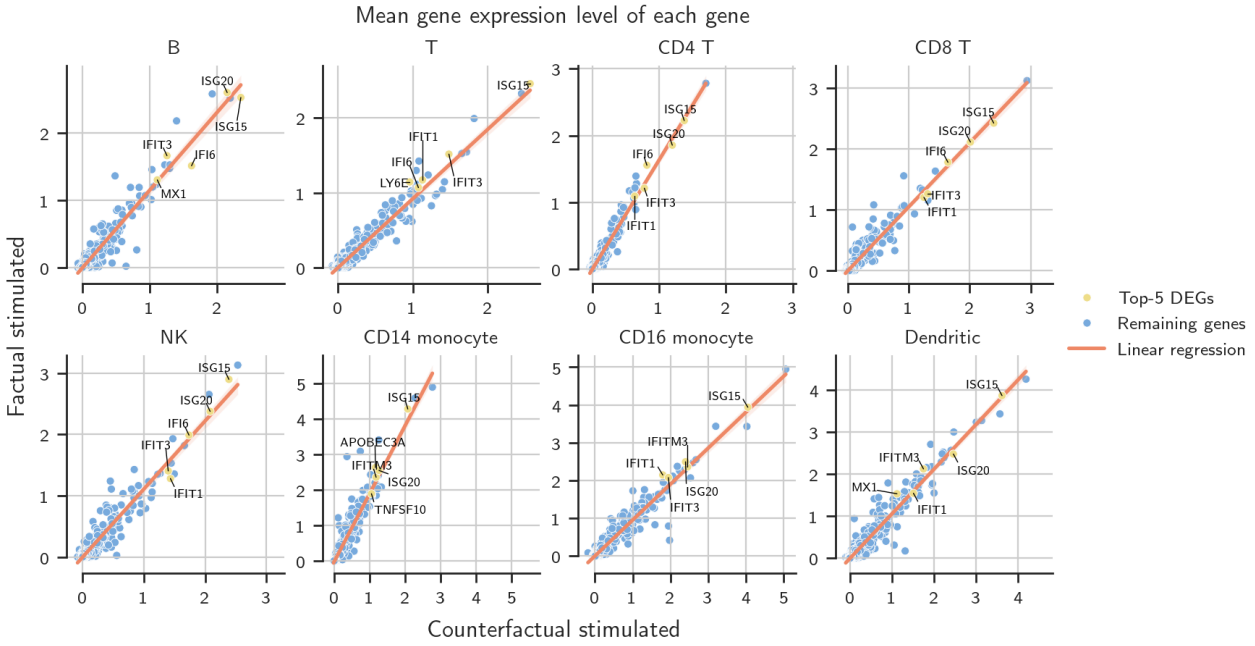


Figure S30. Regression plots for counterfactual prediction using CoMP. Results are shown for the median-performing replicate model. The top-5 DEGs for each cell type are highlighted. A 95 % confidence interval is shown for the linear regression estimated using bootstrapping by resampling 1000 times.

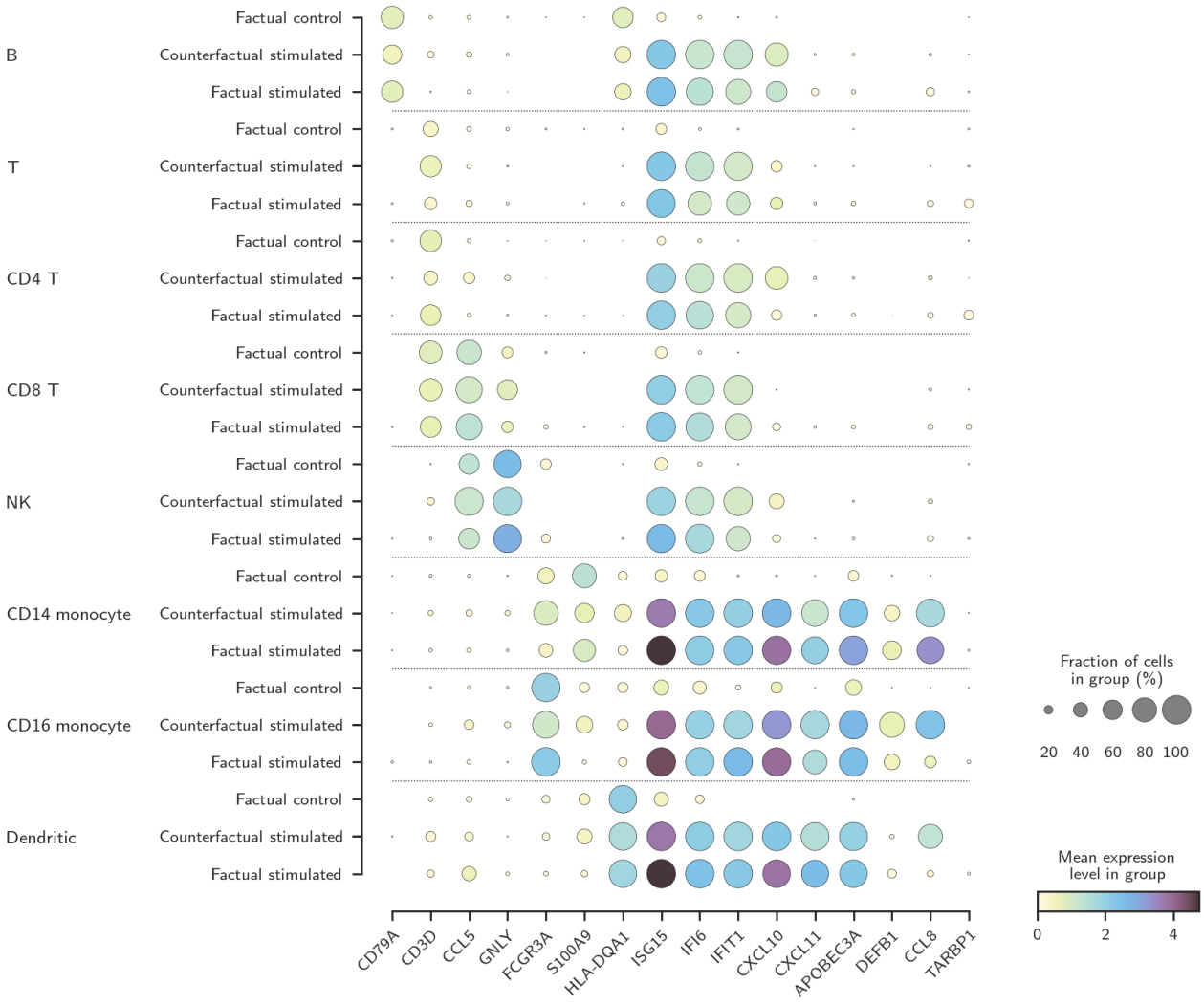


Figure S31. Dot plot of mean gene expressions for counterfactual prediction using scVAE-CB (reproduced for easy comparison). For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model. The optimal scVAE-CB model was selected by the mean coefficient of determination using all genes.

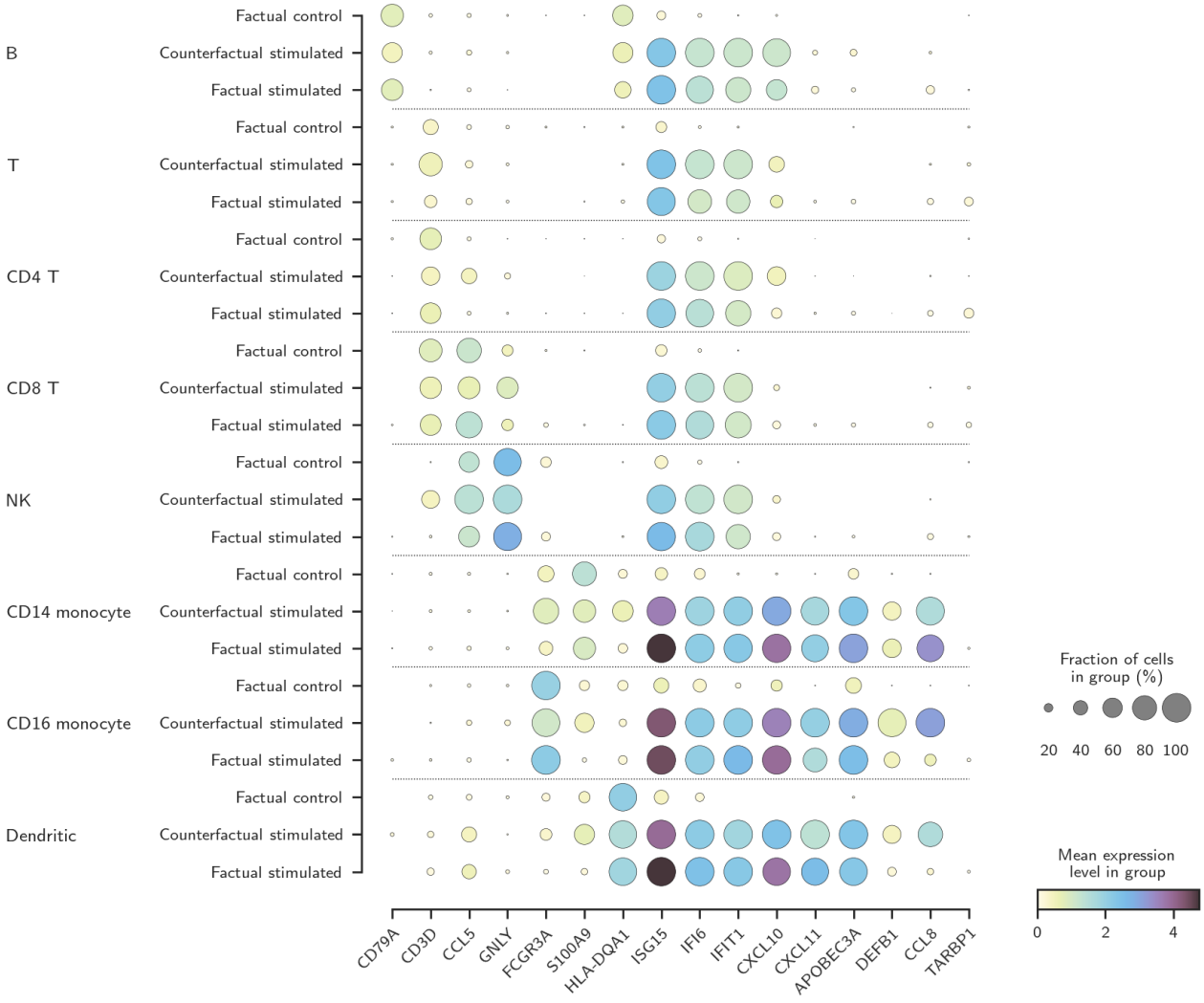


Figure S32. Dot plot of mean gene expressions for counterfactual prediction using scVAE-CM. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model. The optimal scVAE-CM model was selected by the mean coefficient of determination using all genes.

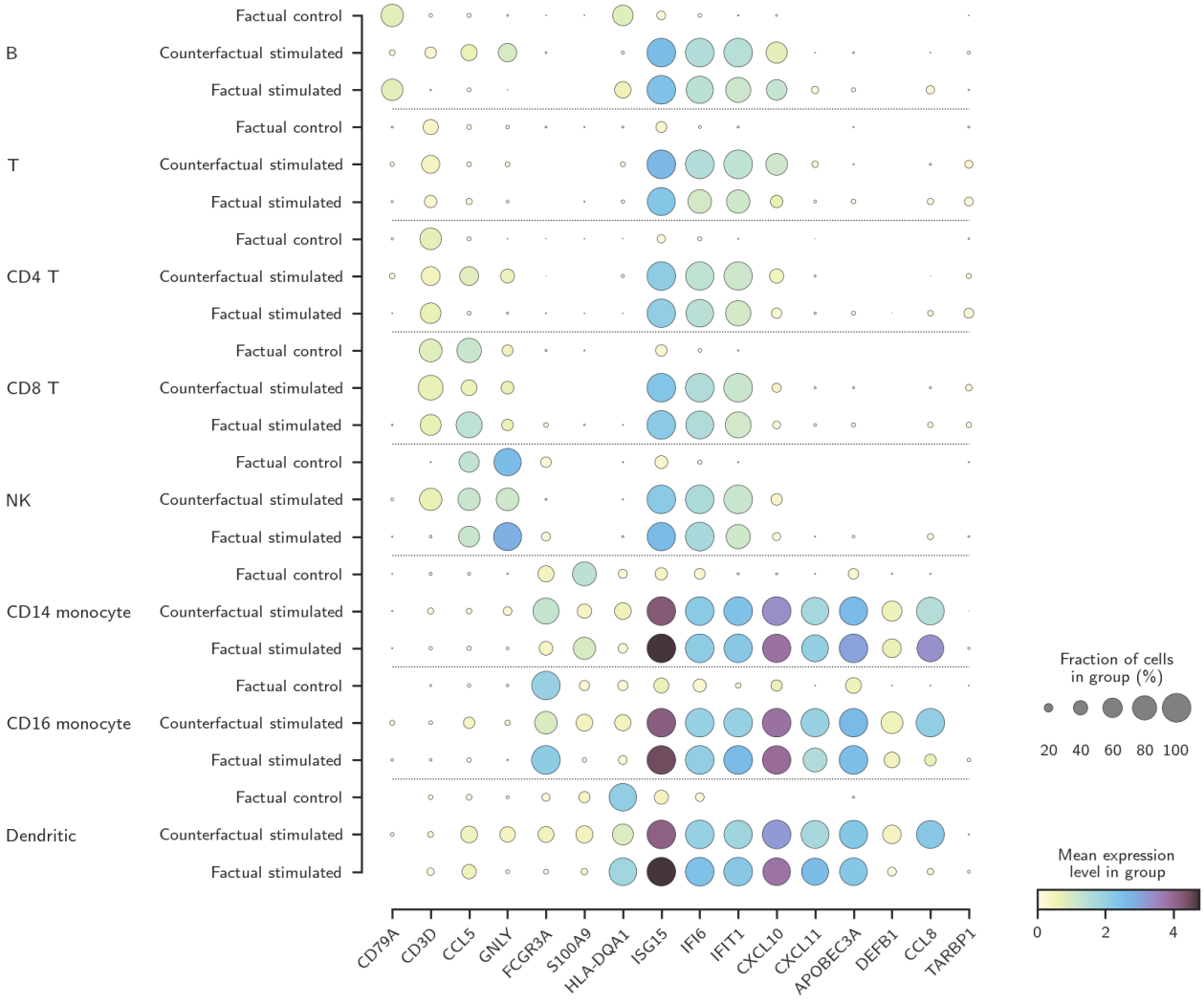


Figure S33. Dot plot of mean gene expressions for counterfactual prediction using scVAE-CH. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model. The optimal scVAE-CH model was selected by the mean coefficient of determination using all genes.

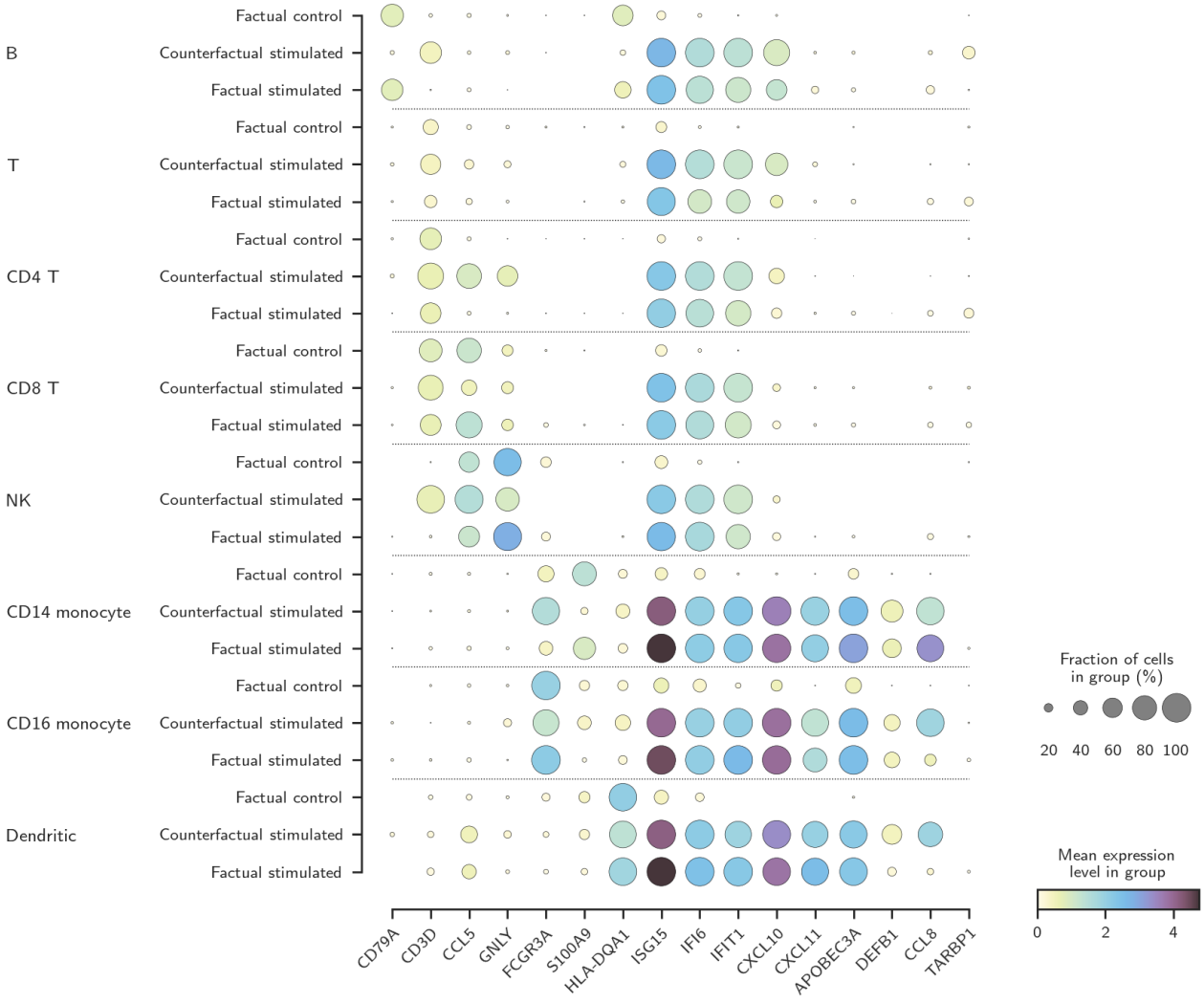


Figure S34. Dot plot of mean gene expressions for counterfactual prediction using scVAE-CHM. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model. The optimal scVAE-CHM model was selected by the mean coefficient of determination using all genes.

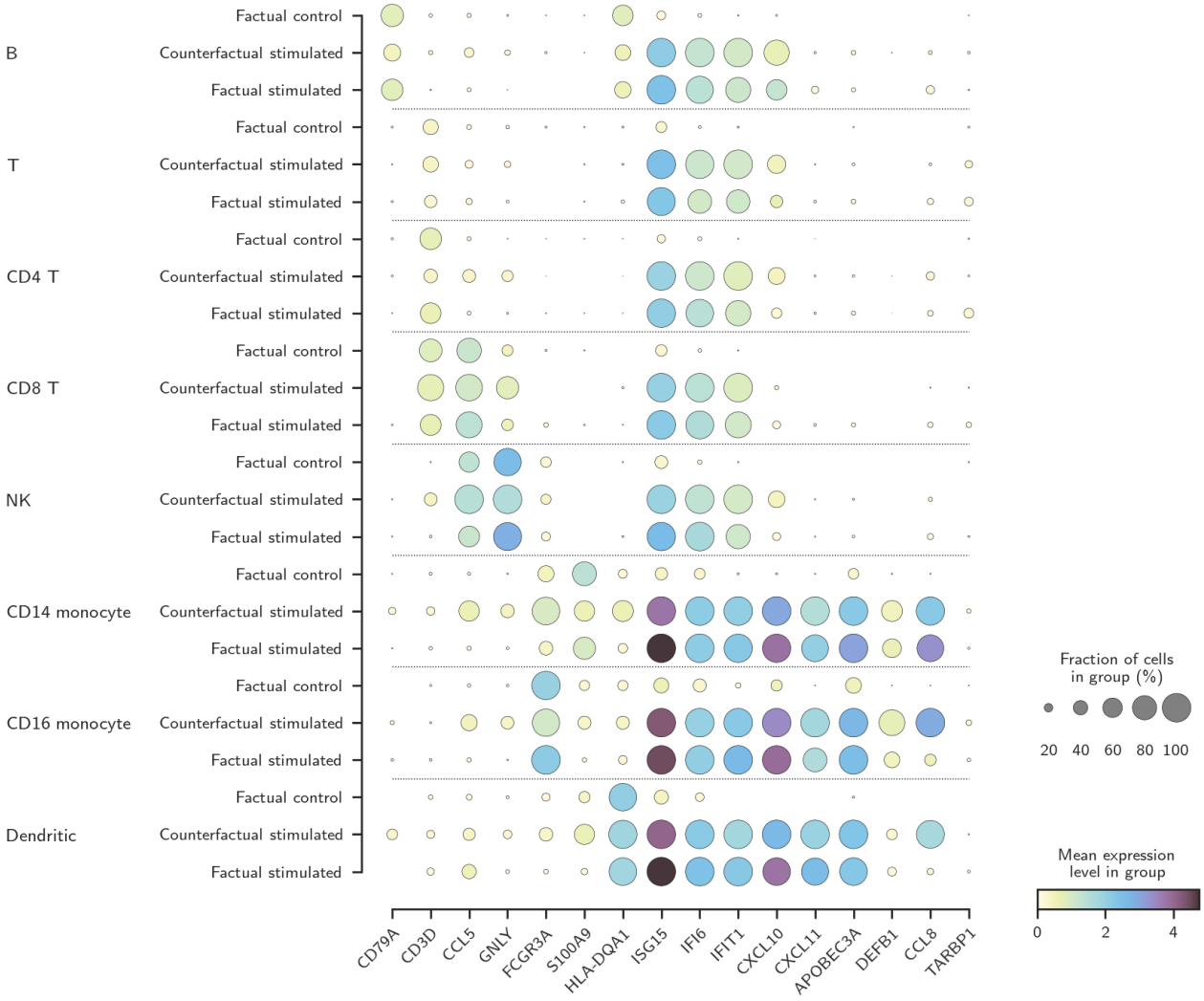


Figure S35. Dot plot of mean gene expressions for counterfactual prediction using scVI. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model.

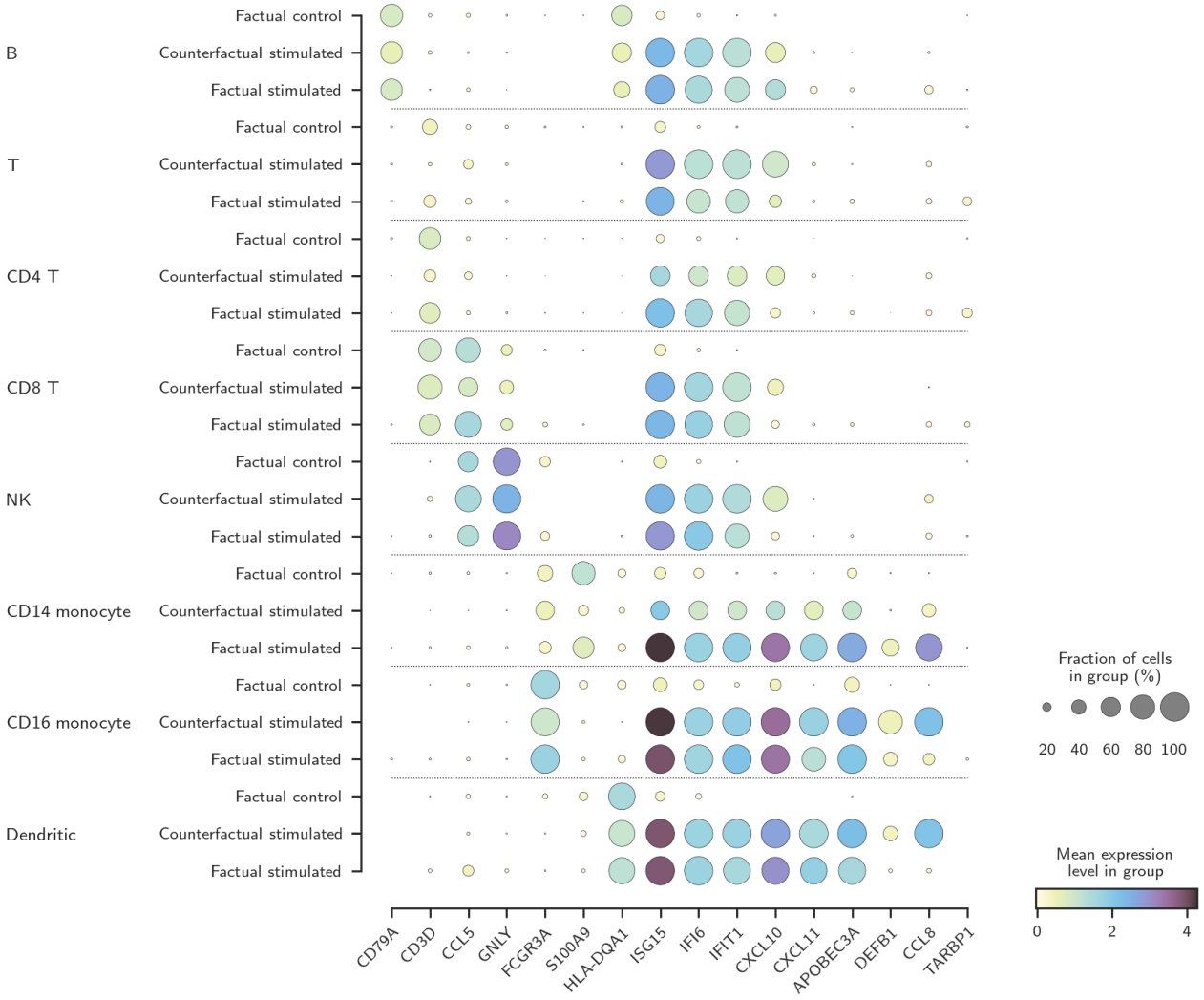


Figure S36. Dot plot of mean gene expressions for counterfactual prediction using CVAE. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model.

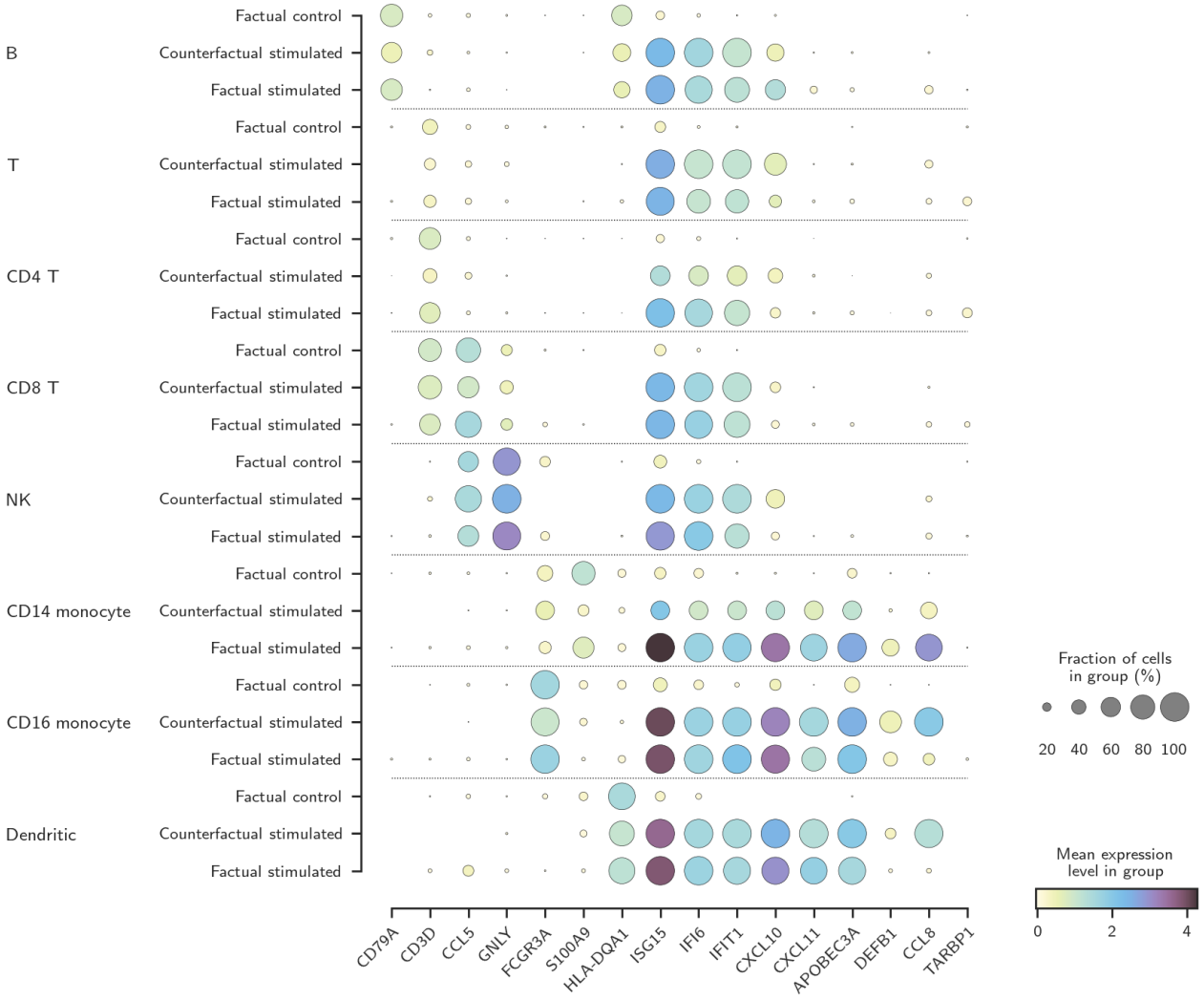
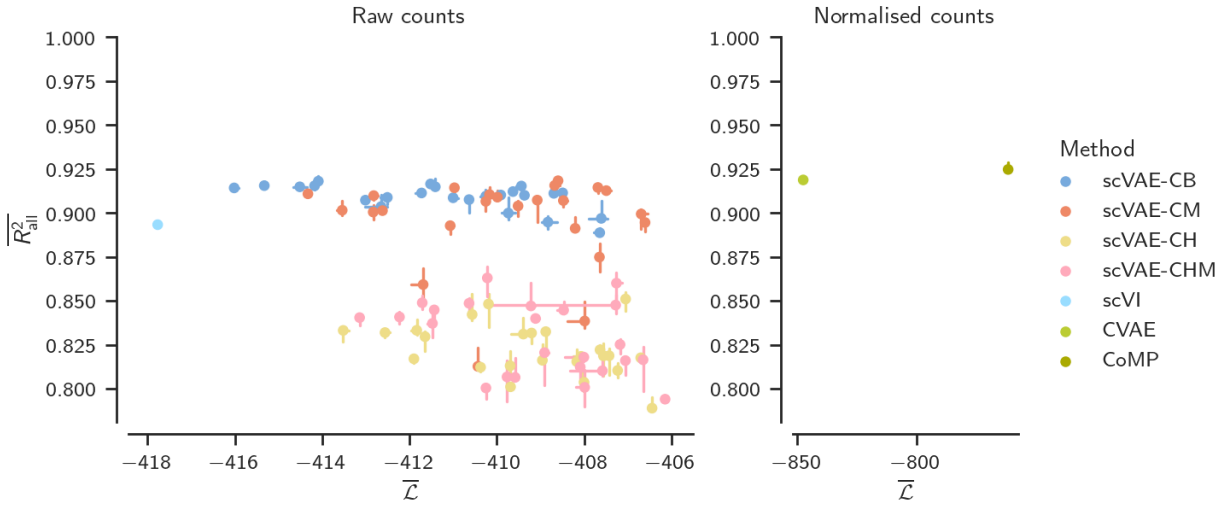
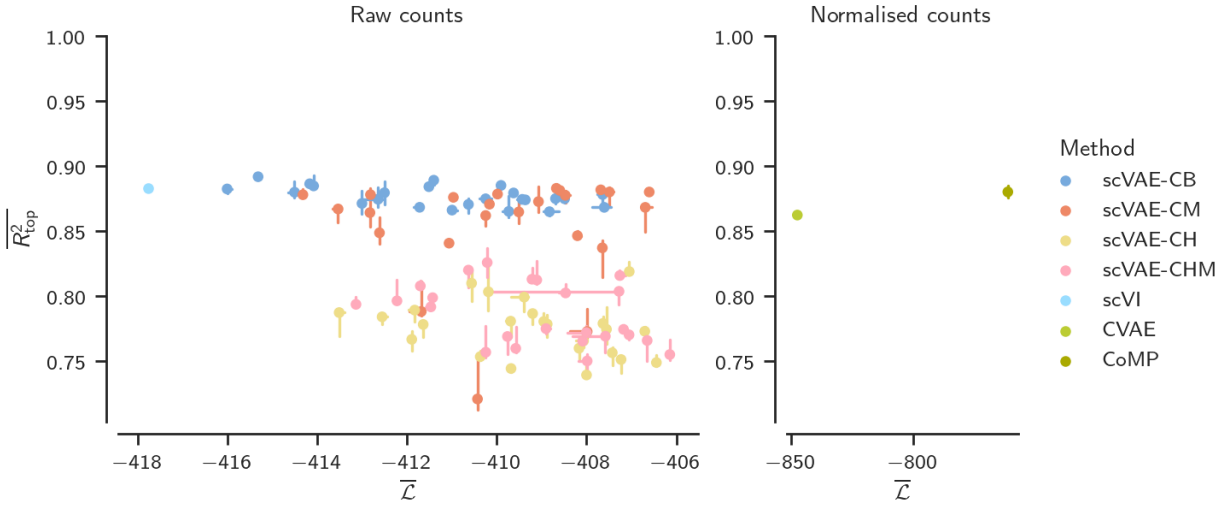


Figure S37. Dot plot of mean gene expressions for counterfactual prediction using CoMP. For each cell type and marker gene, the mean gene expressions across control, stimulated, and counterfactually stimulated cells are shown. The counterfactually stimulated cells were generated by the median-performing replicate model.



(a) Mean coefficient of determination using all genes.



(b) Mean coefficient of determination using the top-100 DEGs.

Figure S38. Relationship between mean coefficients of determination and mean variational lower bound for counterfactual prediction. The median and the interquartile range across replicate models are shown of each method.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

P. J. Bickel, F. Götze, and W. R. van Zwet. *Resampling Fewer Than n Observations: Gains, Losses, and Remedies for Losses*, pages 267—297. Springer New York, Nov. 2011. ISBN 9781461413141. `DOI:10.1007/978-1-4614-1314-1_17`.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint*, Jan. 2016. `arXiv:1511.06349`.

J. S. Bridle. *Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*, page 227–236. Springer Berlin Heidelberg, 1990. ISBN 9783642761539. `DOI:10.1007/978-3-642-76153-9_28`.

J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. D. Hoffman, and R. A. Saurous. Tensorflow distributions. *arXiv e-prints*, 2017. `arXiv:1711.10604`.

C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 451–457, Cambridge, MA, USA, 2000. MIT Press.

C. H. Grønbech, M. F. Vording, P. N. Timshel, C. K. Sønderby, T. H. Pers, and O. Winther. scVAE: Variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, 05 2020. ISSN 1367-4803. `DOI:10.1093/bioinformatics/btaa293`.

J. Han and C. Moraga. *The influence of the sigmoid function parameters on the speed of backpropagation learning*, page 195–201. Springer Berlin Heidelberg, 1995. ISBN 9783540492887. `DOI:10.1007/3-540-59497-3_175`.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. `DOI:10.1038/s41586-020-2649-2`.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, July 2012. `arXiv:1207.0580`.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. `DOI:10.1109/MCSE.2007.55`.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint*, Feb. 2015. `arXiv:1502.03167`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint*, Dec. 2014. `arXiv:1412.6980`.

I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-r. Loh, and S. Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 16(12):1289–1296, Nov. 2019. ISSN 1548-7105. `DOI:10.1038/s41592-019-0619-0`.

R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018. ISSN 1548-7105. DOI:10.1038/s41592-018-0229-2.

M. Lotfollahi, F. A. Wolf, and F. J. Theis. scGen predicts single-cell perturbation responses. *Nature Methods*, 16(8):715–721, July 2019. DOI:10.1038/s41592-019-0494-8.

W. McKinney. Data structures for statistical computing in Python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. DOI:10.25080/Majora-92bf1922-00a.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009. ISSN 0888-613X. DOI:10.1016/j.ijar.2008.11.006.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. *arXiv preprint*, May 2016. arXiv:1602.02282.

I. Virshup, S. Rybakov, F. J. Theis, P. Angerer, and F. A. Wolf. anndata: Annotated data. *bioRxiv*, Dec. 2021. DOI:10.1101/2021.12.16.473007.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. DOI:10.1038/s41592-019-0686-2.

M. L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. DOI:10.21105/joss.03021.

F. A. Wolf, P. Angerer, and F. J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), Feb. 2018. ISSN 1474-760X. DOI:10.1186/s13059-017-1382-0.

A.2 Paper B: scAAE

Title

scAAE: Adversarial autoencoders for single-cell gene expression data

Authors

Christopher Heje Grønbech, Emiliano Molinaro, Ole Winther, and Kedar Natarajan

Contributions

The initial project idea and conception was done by Kedar, and the initial method development and analysis for this report was done by Emiliano and Kedar. Christopher reimplemented the method with a then-new major version of the machine-learning library TensorFlow. Christopher have also continued the development of the method (with regular inputs from Kedar) and the implementation to improve the training and the evaluation of the method and the performance of the implementation. Regarding the results in the report, Kedar preprocessed the data sets, and Christopher trained all models and performed all analyses. As for the report itself, Kedar and Christopher wrote the introduction (§1) and the conclusion (§5), and I wrote the section on methods and materials (§2), the results section (§3), and the discussion (§4) – all with inputs from Kedar. Kedar and Ole edited the final report.

scAAE: Adversarial autoencoders for single-cell gene expression data

Christopher Heje Grønbech[1,2], Emiliano Molinaro[3], Ole Winther[2,4,5], and Kedar Natarajan[6,*]

[1]Qlucore, Sweden
[2]Bioinformatics Centre, Department of Biology, University of Copenhagen, Denmark
[3]SDU eScience Centre, University of Southern Denmark, Denmark
[4]Centre for Genomic Medicine, Rigshospitalet, Copenhagen University Hospital, Denmark
[5]Section for Cognitive Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark
[6]Department of Biotechnology and Biomedicine, Technical University of Denmark, Denmark
[*]Correspondence to kenana@dtu.dk

Last typeset on 20 January 2025

Abstract

Single-cell RNA-sequencing (scRNA-seq) has transformed our understanding of cell types, states, and regulatory networks governing health and diseases. The big-data multi-dimensional single-cell datasets require scalable deep-learning approaches for optimal representation, clustering, and fast runtimes to enable meaningful biological interpretation. Here we present scAAE, an adversarial autoencoder approach for representation-based learning from scRNA-seq data. Applying scAAE to simulated and real small and large datasets, we demonstrate an improved statistical performance in reference clustering, cell-type identification, and biologically meaningful interpretation. By designing an early-stopping clustering evaluation strategy, scAAE enables improved clustering and outcompetes other methods in benchmarks. scAAE is implemented in Python using the TensorFlow machine-learning library and is freely available at `https://github.com/Natarajanlab/scaae`.

1 Introduction

Approaches based on machine learning, especially deep learning, are rapidly being used in biology and provides unprecedented insights in different fields including microscopy, image processing, and others (Wu et al., 2019; Hallou et al., 2021; Ching et al., 2018). Machine-learning methods typically take an input data set and partition it into non-overlapping training and test sets. The training set is used to learn the data structures or patterns and then assessed on the test set. The learnt predictions can therefore be applied to any other data set of the same kind as the original one. Deep-learning methods can effectively take large datasets and find data substructures that are not easily interpretable and identifiable using conventional traditional approaches. These methods are based on artificial neural networks that use multiple layers to extract data structures or patterns within the input data. The aim is to capture different aspects of the data structures or patterns across individual layers and ascertain which features are the best descriptors.

Single-cell transcriptomics serves as a powerful tool to identify cell-states within populations of cells and to dissect underlying heterogeneity at high resolution. The emergence of single-cell genomics revolutionised our understanding of tissue complexity in normal and diseased tissues and large efforts are directed toward cataloguing all cell types across a number of organisms. Several deep-learning methods have been proposed and used for single-cell RNA-sequencing (scRNA-seq). The different applications and tools include cell-type identification, composition, and detection (CellAtlasSearch, Srivastava et al., 2018; CaSTLe, Lieberman et al., 2018; scQuery, Alavi et al., 2018; Dhaka, Rashid et al., 2019; scScope, Deng et al., 2019; scMatch, Hou et al., 2019; SAVER-X, Wang et al., 2019; rCASC Alessandrì et al., 2019; CellAssign, Zhang et al., 2019; scPred, Alquicira-Hernandez et al., 2019); batch-effect removal or merging (scVI, Lopez et al., 2018; scMerge, Lin et al., 2019; SAVER-X); visualisation (SIMLR, Wang et al., 2017; netSNE, Cho et al., 2018; VASC, Wang and Gu, 2018; scVI; scVAE, Grønbech et al., 2020); dimensionality reduction (ZIFA, Pierson and Yau, 2015; SIMLR; VASC; SWNE, Wu et al., 2018; scVAE); clustering (SIMLR, CaSTLe, scVAE); cell-type detection and perturbation response (scGen, Lotfollahi et al., 2019); imputation (ALRA, Linderman et al., 2018; TRANSLATE, Badsha et al., 2020); multi-omics (MAGAN, Amodio and Krishnaswamy, 2018); and multi-utility (scVI; SAUCIE, Amodio et al., 2019).

We have several contributions. First, we implement scAAE, an adversarial autoencoder (AAE, Makhzani et al., 2016) for scRNA-seq data. AAEs are based on generative adversarial networks (GAN, Goodfellow et al., 2014) and variational autoencoders (VAE; Kingma and Welling, 2014; Rezende et al., 2014) and perform inference by matching an encoding distribution of a hidden vector of the autoencoder with an arbitrary distribution. This enables an AAE to map its input to a compressed representation. Second, we demonstrate that scAAE is able to find a compressed representation that is optimal for clustering of scRNA-seq data sets. To do this, we implement an early-stopping scheme performing clusterings and evaluation of these clusterings during optimisation. This is enabled by recent developments in using GPUs for more machine-learning methods (RAPIDS, 2023). Third, we show that scAAE performs as good or better than scVI in most cases across both simulated and multiple real scRNA-seq data sets. Fourth, we provide scAAE as a publicly available software tool.

2 Methods and materials

We have developed, implemented, and made use of several methods and techniques to model gene expression counts of individual cells from scRNA-seq data as well as to cluster these cells. In this section, we describe these methods and techniques as well as how we assess them and which data sets we use for this task.

2.1 Adversarial autoencoders

We start by representing a single cell as a vector $\mathbf{x}$, and each of its components x_n corresponds to the gene expression count of gene n. To improve the clustering of cells, we want to find a representation that captures the essential covariation between genes across cells and cluster this representation instead. Such a representation $\mathbf{z}$ should have features that are (nonlinear) combinations of gene counts, and it should have fewer features than genes used for $\mathbf{x}$. Having fewer features enforces combinations of gene counts, and it also makes the clustering faster. We refer to this representation as a latent representation, and the function that encodes $\mathbf{x}$ to $\mathbf{z}$ we refer to as the encoder. To assess how accurately $\mathbf{z}$ represents $\mathbf{x}$, we also want to find a function that can decode $\mathbf{z}$ back to $\mathbf{x}$, and we refer to this as the decoder. The encoder and the decoder together form an autoencoder (Kramer, 1991, 1992). We model the encoder and the decoder as probability distributions and let $\mathbf{x}$ and $\mathbf{z}$ be stochastic variables. $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$ represents the encoder distribution parameterised by $\boldsymbol{\phi}$, and $p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z})$ represents the decoder distribution parameterised by $\boldsymbol{\theta}$.

Ensuring that the autoencoder generalises and does not overfit, we regularise it by enforcing the latent variable $\mathbf{z}$ to follow a certain latent distribution $p(\mathbf{z})$. This is achieved by integrating a generative adversarial network (GAN, Goodfellow et al., 2014) into the autoencoder. The purpose of a generative adversarial network is to generate samples from a desired data distribution and consists of

two neural networks: a generator and a discriminator. The generator is optimised to generate samples resembling data from the desired data distribution, and the discriminator is optimised to discriminate between generated samples and samples from the desired data distribution. This optimisation is done in alternating stages. For our purposes, a discriminator is added to the autoencoder with $\mathbf{z}$ as input either from the encoder distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ or from the latent distribution $p(\mathbf{z})$ as seen in Figure 1. The encoder of the autoencoder will also function as the generator of the generative neural network. In this way, the discriminator will force the encoder distribution conform to the latent distribution. This combination of autoencoder and generative neural network constitutes the adversarial autoencoder (Makhzani et al., 2016).

2.2 Choosing encoding to and regularisation of latent features

The encoder distribution is chosen to be a multivariate normal distribution with mean and variance as functions of $\mathbf{x}$:

$$q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x})\mathbf{I}), \tag{1}$$

where $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x})$ are modelled using deep neural networks. As an example, $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ can be modelled using a feed-forward neural network with K layers:

$$\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) = \mathbf{a}_K, \tag{2a}$$

$$\mathbf{a}_k = h_k(\mathbf{W}_k \mathbf{a}_{k-1} + \mathbf{b}_k), \tag{2b}$$

$$\mathbf{a}_0 = \mathbf{x}. \tag{2c}$$

Here, $\mathbf{a}_k$ is the output (activation) of the kth layer, $\mathbf{W}_k$ and $\mathbf{b}_k$ are the weights and bias of the kth layer and components of $\boldsymbol{\phi}$, and $h_k(\cdot)$ is an element-wise nonlinear transformation (activation function). The activation function of the Kth layer, $h_K(\cdot)$ is chosen as an appropriate function for the distribution parameter. In this way, the latent features become stochastic nonlinear combinations of the gene expression counts.

The discriminator is also modelled as a probability distribution and it is parameterised by $\boldsymbol{\omega}$:

$$p_{\boldsymbol{\omega}}(y \mid \mathbf{z}) = \mathrm{B}(y; 1, \tau_{\boldsymbol{\omega}}(\mathbf{z})), \tag{3}$$

where y is the verdict of the discriminator and is a binary variable signifying if a sample $\mathbf{z}$ comes from the latent distribution or if it is generated:

$$y = \begin{cases} 0, & \mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}), \\ 1, & \mathbf{z} \sim p(\mathbf{z}). \end{cases} \tag{4}$$

Since y is binary, a Bernoulli distribution $\mathrm{B}(1, \cdot)$ is used as the discriminator distribution, and $\tau_{\boldsymbol{\omega}}(\mathbf{z})$ is the probability of $y = 1$, and it is modelled using a deep neural network similarly to $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ in

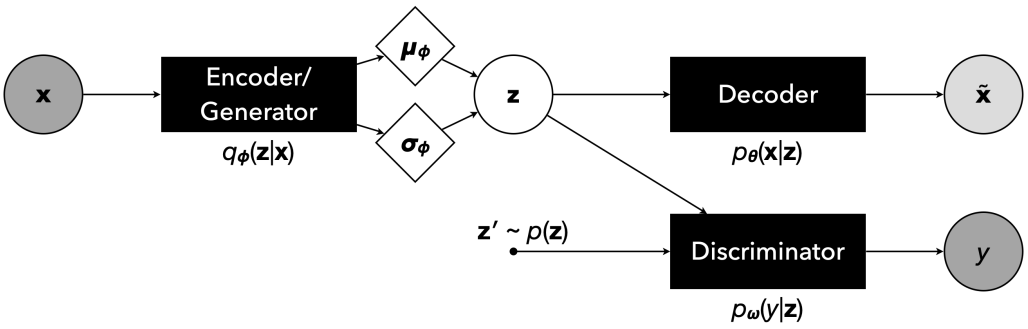


Fig. 1. Model graph for the adversarial autoencoder. Circles represent variables that are observed (dark grey), reconstructed (light grey), and latent (white). The rhombi symbolise deterministic variables such as the distribution parameters. The black rectangles denote neural networks, and arrows indicate input and output.

equation (2). To match the encoder distribution, the latent distribution, which $\mathbf{z}$ should follow, is chosen to be a multivariate standard normal distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}). \tag{5}$$

2.3 Modelling standardised and raw gene expression counts

The decoder distribution is chosen to be appropriate for the data it models, and we model both standardised and raw gene expression counts. For counts that have been log-normalised and standardised, a multivariate normal distribution with unitary covariance is used:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\nu}_{\boldsymbol{\theta}}(\mathbf{z}), \mathbf{I}), \tag{6}$$

where $\boldsymbol{\nu}_{\boldsymbol{\theta}}(\mathbf{z})$ is function of $\mathbf{z}$. The reconstruction $\tilde{\mathbf{x}}$ of $\mathbf{x}$ is the mean of $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$, and this is $\boldsymbol{\nu}_{\boldsymbol{\theta}}$ in this instance.

For raw counts, we test several discrete probability distributions. The simplest of these is the Poisson (P) distribution, which models a count with just an expected rate. Since $\mathbf{x}$ is a vector, we model a separate distribution for each component x_n and let the product of these be the decoder distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \prod_n \mathrm{P}(x_n; \lambda_n^{(\boldsymbol{\theta})}(\mathbf{z})), \tag{7}$$

where $\lambda_n^{(\boldsymbol{\theta})}(\mathbf{z})$ is the expected count rate of gene n and a function of $\mathbf{z}$. The rates for all genes compose the rate parameter vector $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z})$. We also test a negative binomial (NB) distribution to account for the overdispersion in gene expression data, since this distribution models both the expected count and the dispersion. To account for the sparsity of scRNA-seq data, we additionally test zero-inflated versions of the Poisson and negative binomial distributions (ZIP and ZINB, respectively). A zero-inflated distribution adds excess zeros to another distribution with a certain probability, which is modelled in addition to the parameters of the other distribution. For instance, a zero-inflated Poisson (ZIP) distribution is expressed as follows:

$$\mathrm{ZIP}(x; \pi, \lambda) = \begin{cases} \pi + (1-\pi)\mathrm{P}(x; \lambda), & x = 0, \\ (1-\pi)\mathrm{P}(x; \lambda), & x > 0, \end{cases} \tag{8}$$

where π is probability of excess zeroes. These distributions are used as the decoder distribution in the same way as the Poisson distribution, and their parameters are also functions of $\mathbf{z}$. All parameter vectors are modelled using deep neural networks similarly to the parameters are in §2.2. We use feed-forward neural networks for this, since we do not know how all genes depend on each other, and since assuming they are independent simplifies the computation.

2.4 Optimisation of adversarial autoencoders

The adversarial autoencoder is optimised using a stochastic-gradient-descent algorithm. In each optimisation step, different parts of the adversarial autoencoder are optimised in three phases. The first phase is the reconstruction phase, where the encoder and the decoder are optimised together to obtain the best reconstruction of the gene expression counts. The objective (or loss) function for the autoencoder for a single data point is computed as follows:

$$L_{\mathrm{AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = -\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})]. \tag{9}$$

Because the probability distribution parameters are modelled using neural networks, evaluating the expected value exactly becomes intractable. Instead, Monte Carlo sampling is used to provide an estimate:

$$L_{\mathrm{AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) \approx -\frac{1}{R} \sum_{r=1}^{R} \mathrm{E}_{\mathbf{z}^{(r)} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})} \Big[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}^{(r)})\Big], \tag{10}$$

where R is the number of Monte Carlo samples. To be able to backpropagate gradients when sampling from $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$, the reparameterisation trick is used (Kingma and Welling, 2014). The autoencoder loss (AEL) function is minimised by summing over all M cells $\mathbf{x}^{(m)}$ with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{m=1}^{M} L_{\mathrm{AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(m)}). \tag{11}$$

Next is the discriminator regularisation phase. Here the discriminator is optimised to distinguish between generated samples and samples from the latent distribution. This is done by evaluating the expected value of the discriminator distribution function with respect to first the encoder distribution and then the latent distribution:

$$\begin{aligned} L_{\mathrm{D}}(\boldsymbol{\omega}; \mathbf{x}) &= -\frac{1}{2}\Big(\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}[\log p_{\boldsymbol{\omega}}(y = 0 \mid \mathbf{z})] + \mathrm{E}_{p(\mathbf{z})}[\log p_{\boldsymbol{\omega}}(y = 1 \mid \mathbf{z})]\Big), \\ &\approx -\frac{1}{2R}\sum_{r_1=1}^{R} \mathrm{E}_{\mathbf{z}^{(r_1)} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}\Big[\log p_{\boldsymbol{\omega}}(y = 0 \mid \mathbf{z}^{(r_1)})\Big] \\ &\quad -\frac{1}{2R}\sum_{r_2=1}^{R} \mathrm{E}_{\mathbf{z}^{(r_2)} \sim p(\mathbf{z})}\Big[\log p_{\boldsymbol{\omega}}(y = 1 \mid \mathbf{z}^{(r_2)})\Big], \end{aligned} \tag{12}$$

In this way, for each cell $\mathbf{x}^{(m)}$ R samples are drawn from the encoder distribution and R samples are drawn from the latent distribution, and y is set depending on which distribution was sampled from according to equation (4). The discriminator loss (DL) function is also minimised in the same way as the autoencoder loss function but only with respect to the discriminator parameterisation $\boldsymbol{\omega}$.

The last phase is the generator regularisation phase, where the generator is optimised to generate samples that look like they are from the latent distribution. By evaluating the expected value of the discriminator distribution function with respect to the encoder distribution, but indicating to the discriminator that the latent distribution was used ($y = 1$), this is achieved:

$$L_{\mathrm{G}}(\boldsymbol{\phi}; \mathbf{x}) = -\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}[\log p_{\boldsymbol{\omega}}(y = 1 \mid \mathbf{z})] = -\frac{1}{R}\sum_{r=1}^{R} \mathrm{E}_{\mathbf{z}^{(r)} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}\Big[\log p_{\boldsymbol{\omega}}(y = 1 \mid \mathbf{z}^{(r)})\Big]. \tag{13}$$

The half factor in equation (12) makes the discriminator loss function comparable to this loss function. The generator loss (GL) function is also minimised as the other loss functions but only with respect to the encoder parameterisation $\boldsymbol{\phi}$.

2.5 Comparison of adversarial and variational autoencoders

A variational autoencoder is similar in structure to an adversarial autoencoder (Supplementary Figure S1). Like an AAE, an VAE also consists of an encoder and a decoder to map its input to a latent representation and back. The encoder and decoder are also modelled as probability distributions, $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$, and the latent representation is encouraged to follow a certain probability distribution, $p(\mathbf{z})$. The main difference between the two methods lies in how this regularisation is encouraged. Whereas an AAE uses an adversarial framework by introducing a discriminator to distinguish between samples from the encoder and the latent distributions, a VAE is variational Bayesian method. This results in the following single loss function for the VAE:

$$L_{\mathrm{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = -\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})). \tag{14}$$

Here, the first term is the same as the autoencoder loss in equation (9), and the second term is the Kullback–Leibler (KL) divergence between $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and $p(\mathbf{z})$. The KL divergence measures the similarity between two distributions and is only zero if the two are the same distribution.

The single loss function for the VAE means that the encoder and decoder are optimised at the same time and are forced to compromise between a latent representation that follows $p(\mathbf{z})$ and a

reconstruction that match the input. In contrast, an AAE has three loss functions that are optimised separately (albeit in consecutive sequence), where the decoding is decoupled from the regularisation. This enables the decoder and the discriminator to focus on each of their respective tasks, so the decoder can better accept a wider variety of samples from the encoder such as samples that also follow $p(\mathbf{z})$. This would allow that the adversarial autoencoder to produce a latent representation that more fully follow the chosen latent distribution than the variational autoencoder can, in general. By choosing an appropriate latent distribution, this could be advantageous when you want to achieve a latent representation that is more easily clustered – even for complex data such as scRNA-seq data. Makhzani et al. (2016) includes more discussion of adversarial and variational autoencoders with examples.

2.6 Clustering, evaluation, and visualisation of the latent representation

Our objective is to find a better representation of the data to cluster, but none of the optimised loss functions measures this. So in addition to evaluating those, we also cluster the latent representation and compute several clustering metrics. The clustering is performed using two similar clustering methods, the Louvain method (Blondel et al., 2008) and the Leiden method (Traag et al., 2019). The Louvain method is a method for finding communities in a network graph and has been used for single-cell analysis (Levine et al., 2015), and the Leiden method improves upon the Louvain method. A neighbourhood graph of the latent representation is used as the network graph, and it is computed as the Euclidean distances between the mean values of the encoder distribution, and these are simply $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$.

We measure both supervised and unsupervised clustering metrics of the clusterings of the latent representations. The supervised clustering metrics compare the clusterings to a reference clustering, and we evaluate the adjusted rand index (ARI; Hubert and Arabie, 1985) and the adjusted mutual information (AMI, Vinh et al., 2009). Both have a value of 1 for two identical clusterings and an expected value of 0. The unsupervised clustering metrics evaluate how well the clusterings partition the single-cell representation by measuring Euclidean distances based on gene expression counts. We compute the following three ones. The silhouette coefficient (SC; Rousseeuw, 1987) is a value between -1 and 1 with a high value signifying that similar data points are clustered together and that dissimilar data points are not. The Calinski–Harabasz index (CHI; Calinski and Harabasz, 1974) is bound below by 0, and higher values indicate that clusters are dense and well separated from each other. The Davies–Bouldin index (DBI; Davies and Bouldin, 1979) is also bounded below by 0, but conversely better divisions of the data achieve lower values.

Finally, we also evaluate the method by visualising the latent representation. This is done by plotting a UMAP embedding (McInnes et al., 2018) of the mean values of the encoder distribution – identical to what is used for clustering the latent representations.

2.7 Preventing model collapse with early stopping

There are several challenges when training a model in a generative-adversarial-network framework (Mescheder et al., 2018). One such challenge is that training an adversarial model can result in mode collapse, meaning that they generate similar output regardless of input (Goodfellow et al., 2014). To avoid this, some adversarial methods employ early stopping using a metric other than the generator or discriminator losses (Borji, 2018). For instance, GANs for image generation use metrics that compare the similarity of generated images to real images (Salimans et al., 2016; Heusel et al., 2018; Zhang et al., 2018).

Since our aim is to find a better representation to cluster, we can use one of the clustering metrics in §2.6 for early stopping by clustering the latent representation during training. Because of recent developments in using GPUs for data science in general (RAPIDS, 2023), it possible to perform the clustering after each epoch of training without slowing down the actual training significantly. We investigate both the unsupervised case where we do not have any cell-type information as well as a semi-supervised case where a separate data set with known cell types is used only for clustering and

evaluation of the clustering. For the unsupervised case we use the unsupervised clustering metrics, and for the semi-supervised case we use the supervised clustering metrics.

Because of the way the weights and biases of the neural networks are usually initialised, the latent representation will start out resembling a Gaussian hypersphere before splitting into clusters. This means that very few clusters will be found initially, which will result in unfairly good unsupervised clustering metrics. This is especially a problem for smaller data set, where fewer examples are seen each epoch. Therefore, early stopping is delayed by a certain amount of epochs at the beginning of training.

2.8 Noise methods for training the discriminator

Another problem with training GANs and related models is that the discriminator can overfit easily (Yazici et al., 2020). One way to fix this is to add noise to the discriminator during training (Feng et al., 2021), and we have explored several methods to do this. One method is label flipping (Yang et al., 2022), where label refers to the verdict y of the discriminator. This consists of changing the verdicts provided to the discriminator for a subset of generated and true samples from true to generated or generated to true:

$$y^* = \begin{cases} \neg y, & u \geq r, \\ y, & \text{otherwise}, \end{cases} \tag{15}$$

where u is drawn from a uniform distribution between 0 and 1, $\mathcal{U}_{[0,1]}$, and where $r \in [0, 1]$ is the rate of the label flipping. Another method is label smoothing (Szegedy et al., 2015), where uniform noise is added to or subtracted from the verdict to ensure $0 \leq y \leq 1$:

$$y^* = \begin{cases} y + u, & y = 0, \\ y - u, & y = 1, \end{cases} \tag{16}$$

where $u \sim \mathcal{U}_{[0,s]}$ with $s \in [0, 1]$ being the smoothing scale. The two last methods entail adding Gaussian noise to either the verdicts (label noise) or the gene expression counts (instance noise; Sønderby et al., 2016) with a standard deviation d.

2.9 Data sets

We use both simulated and real scRNA-seq data sets to test our method, and they are all listed in Table 1. The simulated data sets are generated by modelling noise and expression counts based on real scRNA-seq data distributions, varying the numbers of cells (3000–10 000 cells) and features (2000–3000 genes) as well as the *a priori* defined number of clusters (two, five, and ten cell types) obtained with real scRNA-seq datasets. We generated three data sets: SIM-2, SIM-5, and SIM-10, where the number signifies the number of clusters in the corresponding data set.

As for real data, we use scRNA-seq dataset from peripheral mononuclear blood cells (PBMC) that is widely benchmarked and is used as a gold standard for several studies (Zheng et al., 2017;

Table 1. Overview of scRNA-seq gene expression data sets.

Data set	Cell count	Gene count	HV gene count	Cell-type count
SIM-2	3 000	2 000	—	2
SIM-5	3 000	3 000	—	5
SIM-10	10 000	3 000	—	10
PBMC-3k	2 638	32 738	1838	8
PBMC-8k	7 475	33 694	1330	8
PBMC-10k	11 090	33 538	2145	12
PBMC-68k	68 551	32 738	1483	11
PBMC-92k	92 014	32 738	1570	9

Hao et al., 2021; Lopez et al., 2018). Specifically, we use 5 PBMC data sets of varying sizes from 10x Genomics (sources listed Supplementary Table S1): PBMC-3k, PBMC-8k, PBMC-10k, PBMC-68k, and PBMC-92k, where the number indicates the number of cells in thousands, approximately. Each PBMC data set is preprocessed in the same manner as for scRNA-seq data using ScanPy (Wolf et al., 2018). We first remove cells with few genes detected, followed by removing genes detected in less than 5 % of cells. Next, the data set is normalised to have the same library size for each cell and then log-transformed, followed by selecting highly variable (HV) genes (Stuart et al., 2019) and removing the remaining genes. Then, the original library size and the percentage of mitochondrial genes expressed in each cell is regressed out using linear regression (Satija et al., 2015), and finally the data set is standardised (to have zero mean and unit variance). We refer to the final preprocessed gene expression counts as standardised counts.

To evaluate the performance of our method, we need a reference clustering for each data set. For the simulated data set, we use the predefined cluster information. For the PBMC data sets, we use either the author-annotated clusters or specific cell-type labels, when available. For the smaller PBMC data sets (PBMC-3k, PBMC-8k, and PBMC-10k), author-annotated labels were not available. Therefore, we cluster the cells using the Louvain or Leiden community clustering with resolution parameters that result in reasonable clusters (that is, individual cell types), which are annotated using marker genes (Supplementary Table S2). The larger data sets (PBMC-68k and PBMC-92k) have been previously analysed (Zheng et al., 2017). The PBMC-92k data set is composed of nine different data sets of purified cell populations (listed in Supplementary Table S1), and we use these cell populations as the reference clustering. For the PBMC-68k data set, the PBMC-92k data set and another data set of a CD14+ monocytes and dendritic cells were used to correlate gene expression profiles and generate cell-type labels (Zheng et al., 2017), which we use as reference clustering.

2.10 Experiment setup

The data sets are each split into a training set, a validation set, and a test set. The simulated data sets and the small PBMC data sets use 80 % for the training set and 10 % for the validation and test sets, while the larger PBMC data sets use 60 % for the training set and 20 % for the two other sets. The models are trained using the training sets, early stopping and model selection are performed using the validation sets, and the models are evaluated using the test sets.

A number of configuration searches are performed to find the optimal configurations for each data set. The first is the architecture search for the deep neural networks for the encoder, the decoder, and the discriminator as well as the dimension of the latent representation. The encoder and decoder are tested with two-layer networks, and the discriminator is tested with networks of two to five layers. See Supplementary Table S3 for the specifics, which also specifies the standard neural network architecture used below. This network architecture search is performed with no added discriminator noise and using standardised gene expression counts. Next is the search for discriminator noise methods. Two experiments were run for each method in §2.8 as well as for a mixture of all methods: one with a small noise parameter value and one with a large one (Supplementary Table S4). The standard network architecture and standardised counts were used for this. The final configuration search is for the decoder distribution for raw counts where the four discrete probability distributions introduced in §2.3 are tested. These experiments are run using the standard network architecture and without a discriminator noise. For comparison, a reference model using the standard network architecture, standardised counts, and no discriminator noise has been included in the searches for noise methods and decoder distributions for raw counts.

Since the Louvain and Leiden methods often require parameter tuning, we perform a parameter search during clustering of the latent representation of the validation and test sets and report the best overall value for each clustering metric (Supplementary Table S5). Because parts of the clustering evaluation cannot be GPU-accelerated, we run the experiments for the configuration searches without early stopping and perform the clustering hyperparameter search at a certain interval (Supplementary Table S6). For each metric, we can then find the epoch with the most optimal value, effectively

employing early stopping with a step size of the evaluation interval.

For the PBMC data sets, we then train models with early stopping using each of the optimisation losses and clustering metrics. The models use combinations of the best configurations for each metric. For network architecture and noise methods, we choose the two most optimal ones. Since the autoencoder loss function and the unsupervised clustering metrics are computed using the gene expression counts, they depend on whether these have been log-normalised and standardised. As such, these metrics cannot be compared across standardised and raw counts. This is especially relevant for the autoencoder loss. Therefore, we choose the two most optimal discrete probability distributions for the decoder distribution when using raw counts, and we use the normal distribution for standardised counts. For clustering metrics, we perform a smaller clustering hyperparameter search at each epoch (Supplementary Table S5).

All experiments are run with three replicates, and results are reported for the run with the median metric. The loss functions were optimised using the Adam optimisation algorithm (Kingma and Ba, 2014) with a mini-batch size of 64, a learning rate of 10^{-4} for the autoencoder loss and of 10^{-5} for the discriminator and generator losses, and a learning decay rate of 10^{-6}. Batch normalisation (Ioffe and Szegedy, 2015) and dropout regularisation (Hinton et al., 2012) with a rate of 0.1 was also used. A leaky rectified linear function (Maas et al., 2013) with a negative slope coefficient of 0.2 is used as activation function for the hidden layers in the neural networks. For numeric stability, computation of the loss functions are performed in logarithmic space, and therefore most of the probability distribution parameters are modelled as logarithms, and as such an identity activation function is used for the output of the neural networks for these parameters. The only exceptions are for the standard deviation of normal distributions and the overdispersion parameter of the negative binomial distributions, which both use the softplus activation function (Dugas et al., 2000).

scAAE is compared to a baseline method as well as scVI. The baseline method consists of decomposing the standardised counts of the test set to 50 principal components and performing the same clustering hyperparameter search as for scAAE on these. For scVI, models are trained with default parameters, and the clustering hyperparameter search is performed using the latent representations of the models.

2.11 Software implementation

scAAE has been implemented in Python using the using the machine-learning software libraries TensorFlow (Abadi et al., 2015) for optimisation, RAPIDS (2023) for clustering, and scikit-learn (Pedregosa et al., 2011) for some evaluations. AnnData (Virshup et al., 2021) is used for reading scRNA-seq data, and Scanpy (Wolf et al., 2018) is used for preprocessing and visualisation. This implementation is open source and platform-independent, and the source code is freely available online[1].

3 Results

3.1 Simulated scRNA-seq data

We first tested our method using the three simulated scRNA-seq data sets with different number of cell populations: SIM-2, SIM-5, and SIM-10. UMAP embeddings of the test set of each data set can be seen in Supplementary Figure S2. The cell populations in these data sets are completely separated, and the baseline method also manages to obtain a test adjusted rand index of 1 for all three of them. scAAE models with various network architectures, discriminator noise methods, and decoder distributions were trained and evaluated on the data sets and early stopping was employed using different metrics. The test adjusted rand indices obtained by the median-performing run of these models and early-stopping metrics are plotted in Supplementary Figures S3–S11. Several results can be observed from these figures and are summarised below.

[1] `https://github.com/Natarajanlab/scAAE`

With any of the early-stopping metrics, a test adjusted rand index of 1 was obtained for all network architectures and for all noise methods for the three data sets except in a few cases. Using some of the loss metrics for early stopping, a few network architectures achieved a worse test ARI for the SIM-2 and the SIM-5 data sets (Supplementary Figures S3–S4). For the SIM-10 data set, about half of the network architectures and most of the noise methods resulted in a worse value with loss-metric early stopping (Supplementary Figures S5 and S8). Inspecting the latent representations of these unsuccessful models, we found that the correct number of groups are represented, but the clustering hyperparameter search was not extensive enough, resulting in groups being partitioned into two or sometimes multiple divisions (examples are shown in Supplementary Figures S12a and S12c). For the SIM-5 data set when using any of the unsupervised clustering metrics for early stopping, three network architectures also achieved a test ARI less than 1 (Supplementary Figure S4). For these network architectures, two groups formed one cluster in the validation latent representation early during training, and this representation achieved a higher value for the unsupervised clustering metrics than it did in later epochs (example shown in Supplementary Figure S12b). So for these experimenters using clustering metrics for early stopping vastly outperformed using loss metrics for the same. All network architectures and all noise methods with a worse test adjusted rand index did still achieve values above 0.6.

Models trained on the raw counts and using discrete probability distributions for the decoder distribution were mostly unsuccessful in attaining a test adjusted rand index of 1. All discrete probability distributions resulted in values between 0.3 and 0.6 for the SIM-2 data set and between 0.4 and 0.7 for the SIM-10 data set (Supplementary Figures S9 and S11, respectively). For the SIM-5 data set, however, Poisson distributions did achieve values of 1 when using some the metrics for early stopping (Supplementary Figure S10). In the remaining cases and for the negative binomial distributions, models obtained values between 0.7 and 0.9. Examining the latent representation of the unsuccessful models, the clusters are more oblong in the UMAP embeddings compared to models using the standardised counts, and the clustering algorithms are more prone to divide groups into multiple parts (examples shown in Supplementary Figures S12d–f).

3.2 PBMC data sets

We then tested our method using the five PBMC scRNA-seq data sets of varying sizes: PBMC-3k, PBMC-8k, PBMC-10k, PBMC-68k, and PBMC-92k. We performed various configuration searches for the different early-stopping metrics as with the simulated data sets, and we then trained models with the most optimal configurations and compared the performance of the early-stopping metrics.

3.2.1 Model selection

Models were trained with several network architectures, discriminator noise methods, and decoder distributions on the PBMC data sets using early stopping with the chosen metrics. The test adjusted rand indices obtained by the median-performing run of these models and early-stopping metrics are plotted in Supplementary Figures S13–S27. The results for both the test adjusted rand indices and the validation metrics used for early stopping and model selection are summarised in the following.

Almost all network architectures using any of the early-stopping metrics achieved comparable test adjusted rand indices for the PBMC-3k data set, but all obtained a value significantly higher than the baseline method (Supplementary Figure S13). Results were more mixed for the PBMC-8k and PBMC-10k data sets, ranging from significantly lower than the baseline to significantly higher for almost all of the early-stopping metrics (Supplementary Figures S14–S15). For the PBMC-68k and PBMC-92k data sets, using different network architectures resulted in test adjusted rand indices comparable to the baseline method with some lower than it (Supplementary Figures S16–S17). In general, models using smaller or mid-sized autoencoder architectures achieved the highest test ARIs, except for PBMC-10k data set where using any autoencoder architecture with a higher-dimensional representation performed the best. Using deeper discriminator networks improved the performance in most cases for all data sets.

When we used the autoencoder loss to stop the various network-architecture models early, we observed that the test ARI and the validation autoencoder loss was positively correlated to various degrees: strong for the PBMC-8k, -68k, and -92k data sets and less so for the other ones. Furthermore, the validation autoencoder loss was also positively correlated with the overall size of the autoencoder. This means that generally using the largest architectures results in the highest (optimal) validation autoencoder losses, but the lowest test ARIs, and the reverse. A smaller positive correlation between the test ARI and the early-stopping metric was also observed when using the validation discriminator loss, especially for the larger data sets. We did not find any other clearer patterns, except a positive correlation between the test ARI and both of the validation supervised clustering metrics for the PBMC-92k data set (Supplementary Figure S17).

Comparing models with added discriminator noise to the reference model without any added noise, we see that adding noise resulted in test ARIs either comparable to the reference model or higher than it in most cases for PBMC-3k, -8k, and -10k data sets (Supplementary Figures S18–S20). It proved especially beneficial for the PBMC-8k data set, where at least one model with added noise obtained a test ARI significantly higher than the baseline. However, none of the data sets had any favoured noise method. For the larger data sets, the noise methods had less of an effect with results being comparable to that of the reference model (Supplementary Figures S21–S22), and they also showed no preference for a noise method. As for the validation early-stopping metrics, using any of the noise methods only resulted in relatively minor changes compared to the reference model, and no discernible patterns were observed.

Regarding decoder distributions for raw counts, results were more varied across data sets. For the PBMC-3k data set, the test ARIs obtained by using discrete probability distributions are mostly comparable to those of the reference model using standardised counts, except a few being significantly lower for some early-stopping metrics (Supplementary Figure S23). For the PBMC-8k data set, the test ARIs of raw-read-count models are mostly higher than the reference values with some being significantly higher and the remaining being comparable (Supplementary Figure S24). For the PBMC-10k data set, it depends on the early-stopping metric whether the raw-read-count models performed better than, worse than, or comparable to the reference model (Supplementary Figure S25). For both the PBMC-68k and -92k data sets, using discrete probability distributions resulted in test ARIs that are comparable to or higher than reference model, except in a few cases (Supplementary Figures S26–S27). In contrast to the discriminator noise methods, using discrete probability distributions did result in relatively major changes in the early-stopping metrics compared to the reference model for some data sets, especially in the following cases: When using the autoencoder loss or the Calinski-Harabasz index for early stopping for all data sets as well as when using the other unsupervised clustering metrics for some of the data sets. This is to be expected, since these metrics are computed using the counts. We also note that when using the autoencoder loss for early stopping, the zero-inflated distributions achieved better results than the standard distributions for all data sets. Additionally, the negative binomial distributions perform better the Poisson distributions for all data sets, except for the PBMC-92k data sets for which it is the reverse. Other than those observations, no clear patterns can be discerned, and none of the data sets or the early-stopping metrics has a preferred decoder distribution for raw counts.

3.2.2 Early-stopping comparison

Finally, we tested models with combinations of the optimal network architectures, discriminator noise methods, and decoder distributions found for each early-stopping metric for the PBMC data sets. The test adjusted rand indices of the optimal model for each early-stopping metric are plotted in Figure 2 for all data sets. Since there is a significant difference between using standardised and raw counts, test adjusted rand indices for these are plotted separately. For comparison, the performances of scVI and the baseline method are also included in Figure 2. UMAP embeddings of the latent representations for the optimal scAAE models using either an unsupervised (scAAE-UC) or a supervised (scAAE-SSC) clustering metric for early stopping are plotted in Figure 3 for the small data sets and Figure 4 for the

large data sets. The UMAP embeddings of the standardised counts as well as the latent representation of scVI are also plotted in these figures.

For the PBMC-3k data set, all optimal models but two are significantly better than the baseline method and comparable to scVI as seen in Figure 2a. The remaining two still select models higher than or comparable to the baseline. The optimal scAAE-UC model (and overall model) was found using standardised counts and DBI for early stopping, and the optimal scAAE-SSC model was also obtained by using standardised counts, but AMI for early stopping. The latent representations for these, seen in Figure 3a, both have cell types clustered in a similar manner as the data-set embedding and the scVI latent representation: one B-cell cluster, one monocyte cluster, and one cluster of T and NK cells. However, these clusters are more separated for scVI than for scAAE.

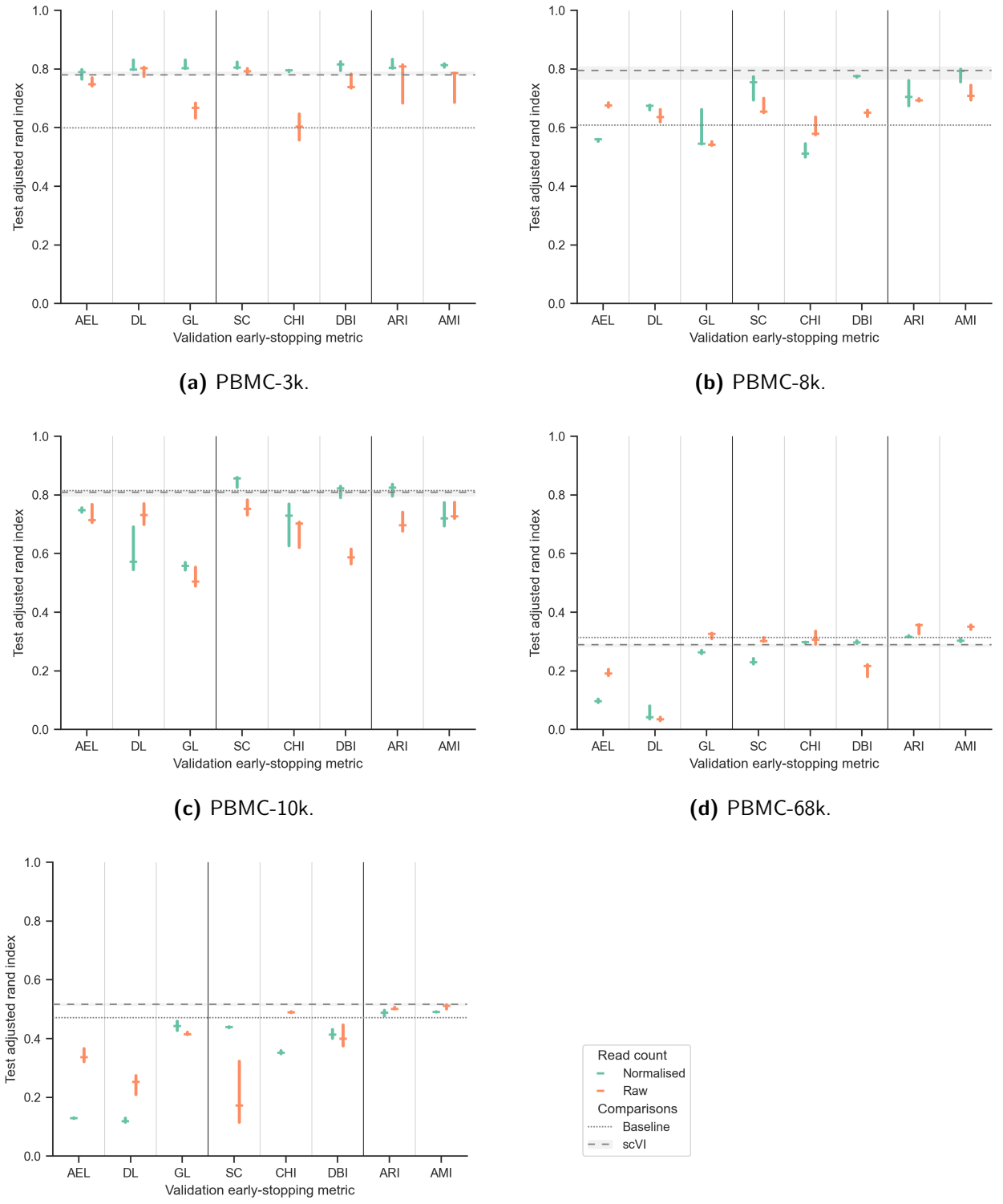
For the PBMC-8k data set, all optimal models are better than the baseline method except when using some of the optimisation metrics or CHI for early stopping and model selection as seen in Figure 2b. Models using standardised counts and either the silhouette coefficient, DBI, or AMI for early stopping are significantly better than the baseline and comparable to scVI. Using standardised counts resulted in the optimal scAAE models. The scAAE-UC model used DBI for early stopping, and the scAAE-SSC model used AMI, and this one also performed the best of the two. Figure 3b show the latent representations for these, and the cell types are clustered similarly to the data-set embedding. For the scVI latent representation, monocytes are separated from a cluster of T and NK cells, whereas these cell types are close together for scAAE.

For the PBMC-10k data set, only three optimal models are comparable to or higher than the baseline as seen in Figure 2c: models using standardised counts with the silhouette coefficient, DBI, or ARI for early stopping. scVI is slightly worse than the baseline. The optimal scAAE-UC model (and overall scAAE model) was obtained by using standardised counts and silhouette coefficient for early stopping, and it was better than both the baseline method and scVI. The optimal scAAE-SSC model also used standardised counts, but ARI for early stopping, and this was only slightly better on average than the baseline method and scVI. The latent representations for these can be seen in Figure 3c. They both show cell types clustered in a similar manner as the data-set embedding and the scVI latent representation.

For the PBMC-68k data set, most optimal models are comparable to the baseline method, which performs better than scVI, as seen in Figure 2d. Models using raw counts and any of the supervised clustering metrics for early stopping perform better than the baseline (and scVI). The models performing the worst used either the autoencoder or discriminator loss for early stopping. Using raw counts yielded the optimal scAAE models. The optimal scAAE-UC model used CHI for early stopping, while the optimal scAAE-SSC model, which also performed the best, used ARI. Also note that a scAAE model using the generator loss for early stopping performed better than the optimal scAAE-UC model. Figure 4a shows the latent representations for both. The one for the scAAE-SSC model have cell types in similar clusters as both the data-set embedding and the scVI latent representation. They all also show a subdivision of NK cells and of some of the T cells. The latent representation of the scAAE-UC model show more overlap between cell types and no subdivision of NK or of T cells.

For the PBMC-92k data set, five of the optimal models are comparable to or better than the baseline method as can be seen in Figure 2e. These models were stopped early using either supervised clustering metric for both standardised and raw counts as well as CHI for raw counts. The optimal scAAE models both used raw counts. The optimal scAAE-UC model was stopped early using CHI, and the scAAE-SSC model was stopped early using ARI and perform the best of the two. The latter model is also comparable to scVI. The latent representation for both, seen in Figure 4b, have cell types clustered similarly as the data-set embedding and the scVI latent representation, except that the NK cells and T cells are not completely separated for the optimal scAAE-UC model.

In addition to using the adjusted rand index for comparison, we also investigated the performance of the optimal models using the adjusted mutual information (Supplementary Figure S28), and we found similar results as for the ARI.



(a) PBMC-3k.

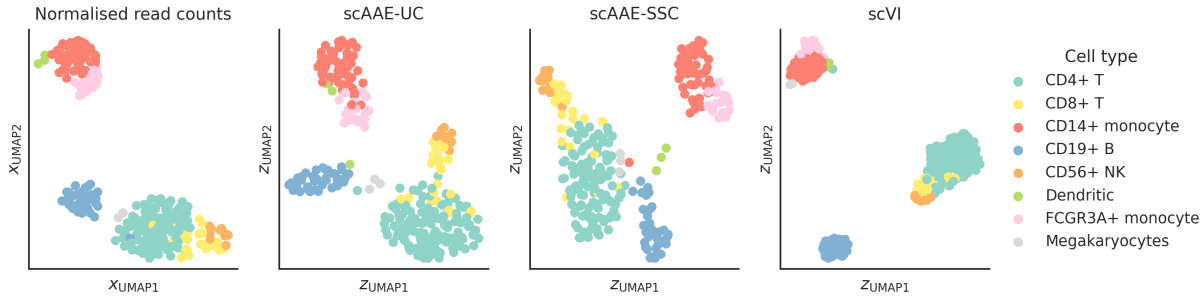
(b) PBMC-8k.

(c) PBMC-10k.

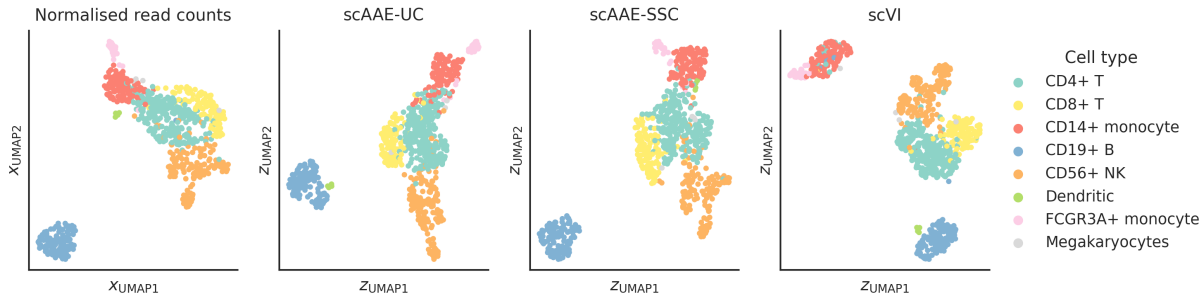
(d) PBMC-68k.

(e) PBMC-92k.

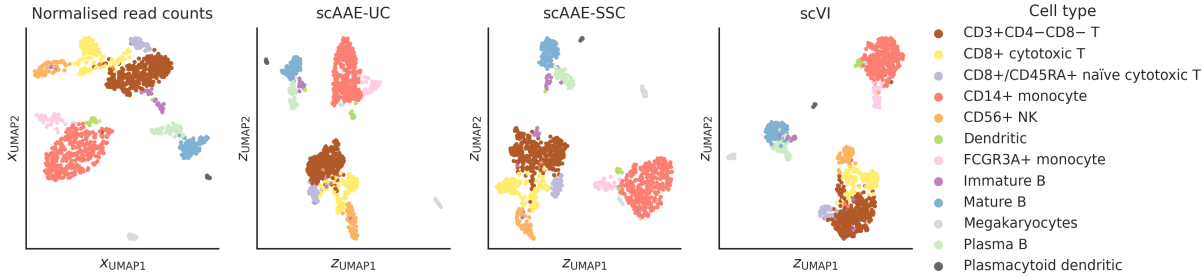
Fig. 2. The test adjusted rand indices for the optimal scAAE model selected by each early-stopping metric for all data sets. The performances of the baseline method and scVI are also shown. For the scAAE models and scVI, the median and the interquartile range for three replicates are visualised.



(a) PBMC-3k.



(b) PBMC-8k.



(c) PBMC-10k.

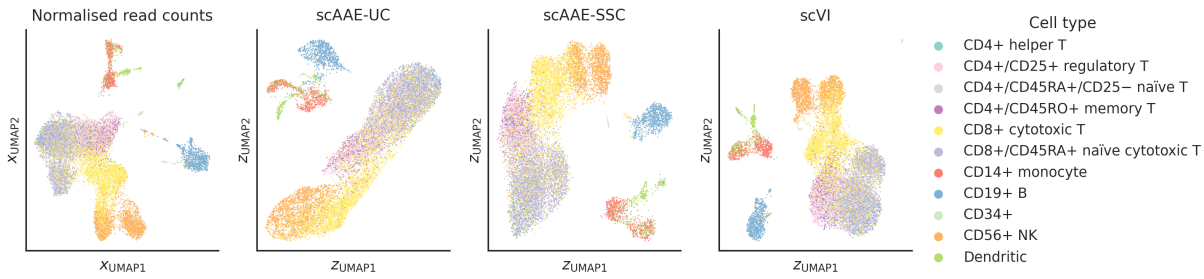
Fig. 3. UMAP embeddings of the small PBMC data sets using the standardised gene expression counts as well as the latent representations of the optimal scAAE-UC model, the optimal scAAE-SSC model, and scVI.

4 Discussion

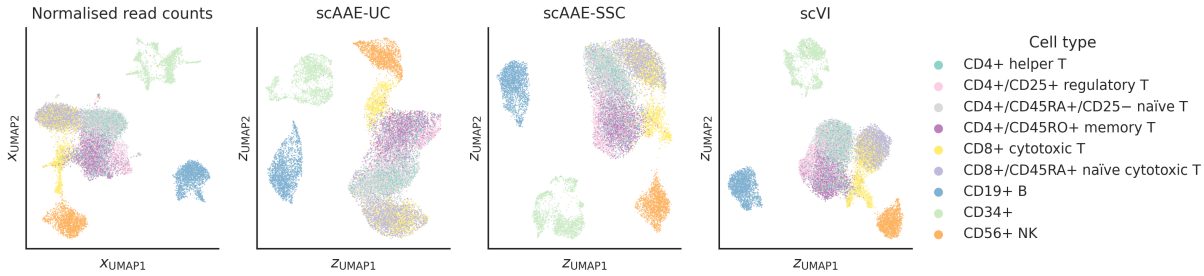
On the whole, scAAE models performed well on the smallest data sets (SIM-2, SIM-5, and PBMC-3k) and was comparable to or better than the baseline and scVI. For the mid-sized data sets (SIM-10, PBMC-8k, and PBMC-10k), scAAE models had more mixed results, but some were still able to best or equal the baseline and scVI. scAAE models for the largest data sets (PBMC-68k and PBMC-92k) mostly performed better or comparable to either the baseline, scVI, or both.

To evaluate scAAE, we used the same clustering hyperparameter search for all models and comparison methods as well as for all data sets. This performed well on the simulated data sets for most models. For some models using loss-metric early stopping cell groups were subdivided into multiple clusters because the models were stopped at a time where the extent of the hyperparameter search were insufficient. Finding potential subclusters are, however, more advantageous than multiple cell groups forming one cluster. This was the case for a few models when using any of the unsupervised clustering metrics for early stopping. These were not instances of mode collapse of the model, since the model recovered using other early-stopping metrics. Rather, this is simply a disadvantage of using an unsupervised clustering metric for early stopping, resulting in the models being stopped too early.

It is more difficult to evaluate the clustering hyperparameter search on the PBMC data sets, since



(a) PBMC-68k.



(b) PBMC-92k.

Fig. 4. UMAP embeddings of the large PBMC data sets using the standardised gene expression counts as well as the latent representations of the optimal scAAE-UC model, the optimal scAAE-SSC model, and scVI.

many of the cell types intermix. We did attempt to mitigate stopping too early for the early-stopping comparison by evaluating clustering metrics every epoch, requiring more epochs with no improvement before stopping, and delaying evaluation at the start of training (see §2.7). One can, however, still observe monocytes and T cells intermixing to a smaller extent for the optimal scAAE-UC model for the PBMC-68k and -92k data sets, whereas both the optimal scAAE-SSC model and scVI were able to better separate these cell types. A similar observation can be made for the PBMC-10k data set, but here neither of the optimal scAAE models is able to separate monocytes and T cells completely. This minor intermixing can also be seen in the data set itself, indicating that separating the cell types would be more difficult than for other data sets. However, scVI is able to separate the cell types completely. We also observed a difference in the general performance of the clustering hyperparameter search between the small PBMC data sets and the large ones. This is partly because the small data sets were clustered using either Louvain or Leiden clustering, which was also used in the clustering search, while the large data sets used other methods (see §2.9) Another reason is that more T cell subsets have been identified in the large data sets than in the small ones. These subsets overlap in expression making it more difficult to cluster them.

We also tested scAAE with different model configurations. Regarding the architecture of the model, small and mid-sized network architectures were preferred in most cases, and deeper discriminator networks generally improved overall performance, which advocate for more network layers over larger network layers. Additionally, data sets with more distinct cell types (for example, the PBMC-10k data set), favour models with higher-dimensional latent representations, which makes sense because of the increased complexity in representing the different cells. As for adding noise to the discriminator during training, using any of the noise methods improved results for all the mid-sized data sets for all or most early-stopping metrics. No improvement was observed for the smaller simulated data sets for which models had already achieved perfect clusterings, and most noise methods resulted in only minor improvements for the PBMC-3k data set. There was also no significant improvement for the larger data sets. One would assume that because of their size, these large data sets already are sufficiently noisy. Modelling raw gene expression counts and using discrete proba-

bility distributions for the decoder, however, proved better for the larger data sets. Modelling these data sets seems to benefit from the library size of individual cells, since the difference in library size is removed when normalising scRNA-seq data. However, no clear preference for a specific discrete probability distribution emerged, except when stopping early the autoencoder loss for PBMC data sets. In that case, the zero-inflated negative binomial distribution performed the best for all PBMC data set, except for the PBMC-92k data set for which the zero-inflated Poisson distribution attained the best autoencoder loss. For smaller data sets, the performance is comparable to or worse than using standardised gene expression counts. This may be because too much noise is included when modelling these with raw counts, or simply because of the difference in how the reference clustering was determined. However, the larger variability between cells is also reflected in the performance of the raw-read-count models: The performance of the replicates of these models vary more, in general, than when using standardised counts.

Comparing the different early-stopping metrics, the optimisation losses performed the worst in general. This was to be expected based on previous research on generative adversarial networks as mentioned in §2.7. One notable exception is an optimal model using a generator loss for early stopping achieving a better ARI than the optimal scAAE-UC model for the PBMC-68k data set. Since optimal models with generator-loss early stopping generally performed comparatively well on the large PBMC data sets, this could because of their size or because of how the reference clustering was created, which were different processes than for the smaller PBMC data sets. We also saw that a more optimal autoencoder loss leads to a less optimal test ARI or AMI, only enforcing why other metrics should be used for early stopping to prevent mode collapse. The clustering metrics we have tested as alternatives generally performed better than the optimisation losses with the supervised metrics outperforming the unsupervised ones on average, especially for the larger data sets. This is also unsurprising, since the clusters are compared to the actual cell types instead of the data itself. For the larger data sets, the results using the generator loss are comparable to using the unsupervised clustering metrics. This may be because clustering metrics and generator losses all evaluate how good the latent representation is, whereas the autoencoder and discriminator losses evaluates other aspects of the method. The only exception is for the PBMC-3k data set, where almost all configurations performed very well and at the same level as scVI, which speaks more to limits of modelling this data set using scAAE and scVI. The best overall early-stopping metrics were the Davies-Bouldin index for unsupervised models and adjusted mutual information for the semi-supervised models.

The optimal scAAE model for each PBMC data set were stopped after fewer epochs than a scVI model with similar network architecture, except for the PBMC-3k data set. However, the autoencoder, the discriminator, and the generator of the scAAE are optimised separately in each epoch, whereas scVI optimises its model weights only once. Moreover, when using clustering metrics for early stopping, the latent representation is also clustered multiple times after each epoch. This results in scAAE being slower to run than scVI.

As for scaling, since scAAE is optimised using a stochastic-gradient-descent algorithm, it scales well to large amounts of cells. However, performing clustering on a validation set during training – even when using RAPIDS for GPU acceleration – limits the validation set to around 25 000 cells for reasonable runtimes on modern hardware in our experience.

5 Conclusion

We show that adversarial autoencoders work perfectly on simulated data in almost all cases, especially when using clustering metrics for early stopping. We also show that AAEs work as good if not better than scVI on real-life data, especially when using supervised clustering metrics for early stopping. However, whereas scAAE fairs better when clustering latent representations in some cases, scVI usually obtains more visually separated latent representations. This agrees with the assertion that latent representations produced by AAEs more fully follow the latent representation than those by VAEs (see §2.5 and Makhzani et al. (2016)). To achieve more separation, we could investigate using a mixture model as the latent distribution for the AAE.

Since generative adversarial networks are difficult to train, we also demonstrate techniques to alleviate this problem. Adding noise to the discriminator helps in cases where data sets are not simple enough to easily cluster or not complex enough to include sufficient noise or variation on their own. One source of variation comes from using raw gene expression counts instead of log-normalised, standardised ones, and modelling these also improves the performance for larger data sets generally. We also show that early stopping is critical to obtaining a good latent representation of the data, and that using the optimisation losses of the AAE is inferior to using clustering metrics, especially for larger data sets. Additionally, we find that deeper mid-sized model architectures perform better than shallower larger ones, but that the dimension of the latent representation should still be large enough to represent the different cell types in a particular data sets.

Our objective was to investigate how AAEs can resolve scRNA-seq data and compare clusterings with ground truths, and we did not aim to resolve clusters further or improve the clustering methods themselves. As such, we did not perform extensive parameter tuning for this purpose. However, this should be done with any method, if the purpose is to find more clusters.

We have two main ideas on how to improve this method. One is to integrate the clustering into the model. Makhzani et al. (2016) demonstrate a way to do this with a fixed number of clusters, but we would like to investigate how network-based clustering methods such as the Louvain and Leiden methods would perform. The other is to optimise the autoencoder and generator losses together, since they (partly) update the same model weights. Both of these changes would allow the model to take more information into account when updating the weights to improve the performance, and to reduce the number of optimisation steps each epoch to speed up the training. This would also allow the method to cluster larger validation sets during training, since it would be clustering minibatches. The discriminator should, however, still be optimised separately to maintain the adversarial framework. Another way to speed up the training would be to model the latent representations deterministically as output of the encoder instead of modelling the parameters of an encoder distribution and sampling this distribution as mentioned in Makhzani et al. (2016).

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 813533 to C.H.G. The research in the K.N.N. lab is supported by the Villum Young Investigator grant (VYI#00025397) and DigitSTEM Initiative.

Co-funded by
the European Union

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

A. Alavi, M. Ruffalo, A. Parvangada, Z. Huang, and Z. Bar-Joseph. A web server for comparative analysis of single-cell RNA-seq data. *Nature Communications*, 9(1), Nov. 2018. ISSN 2041-1723. `DOI:10.1038/s41467-018-07165-2`.

L. Alessandrì, F. Cordero, M. Beccuti, M. Arigoni, M. Olivero, G. Romano, S. Rabellino, N. Licheri, G. De Libero, L. Pace, and R. A. Calogero. rCASC: reproducible classification analysis of single-cell

sequencing data. *GigaScience*, 8(9), Sept. 2019. ISSN 2047-217X. DOI:10.1093/gigascience/giz105.

J. Alquicira-Hernandez, A. Sathe, H. P. Ji, Q. Nguyen, and J. E. Powell. scPred: accurate supervised method for cell-type classification from single-cell RNA-seq data. *Genome Biology*, 20(1), Dec. 2019. ISSN 1474-760X. DOI:10.1186/s13059-019-1862-5.

M. Amodio and S. Krishnaswamy. MAGAN: Aligning biological manifolds. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 215–223. PMLR, 10–15 Jul 2018. https://proceedings.mlr.press/v80/amodio18a.html.

M. Amodio, D. van Dijk, K. Srinivasan, W. S. Chen, H. Mohsen, K. R. Moon, A. Campbell, Y. Zhao, X. Wang, M. Venkataswamy, A. Desai, V. Ravi, P. Kumar, R. Montgomery, G. Wolf, and S. Krishnaswamy. Exploring single-cell data with deep multitasking neural networks. *Nature Methods*, 16(11):1139—-1145, Oct. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0576-7.

M. B. Badsha, R. Li, B. Liu, Y. I. Li, M. Xian, N. E. Banovich, and A. Qiuyan Fu. Imputation of single-cell gene expression with an autoencoder neural network. *Quantitative Biology*, 8(1):78–94, Mar. 2020. ISSN 2095-4697. DOI:10.1007/s40484-019-0192-7.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct. 2008. DOI:10.1088/1742-5468/2008/10/p10008.

A. Borji. Pros and cons of GAN evaluation measures. *arXiv e-prints*, Feb. 2018. arXiv:1802.03446.

T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974. DOI:10.1080/03610927408827101.

T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. Xu, E. M. Cofer, C. A. Lavender, S. C. Turaga, A. M. Alexandari, Z. Lu, D. J. Harris, D. DeCaprio, Y. Qi, A. Kundaje, Y. Peng, L. K. Wiley, M. H. S. Segler, S. M. Boca, S. J. Swamidass, A. Huang, A. Gitter, and C. S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, Apr. 2018. ISSN 1742-5662. DOI:10.1098/rsif.2017.0387.

H. Cho, B. Berger, and J. Peng. Generalizable and scalable visualization of single-cell data using neural networks. *Cell Systems*, 7(2):185–191.e4, Aug. 2018. ISSN 2405-4712. DOI:10.1016/j.cels.2018.05.017.

D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, Apr. 1979. DOI:10.1109/tpami.1979.4766909.

Y. Deng, F. Bao, Q. Dai, L. F. Wu, and S. J. Altschuler. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nature Methods*, 16(4):311–314, Mar. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0353-7.

C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 451–457, Cambridge, MA, USA, 2000. MIT Press.

R. Feng, D. Zhao, and Z.-J. Zha. Understanding noise injection in gans. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3284–3293. PMLR, July 2021. https://proceedings.mlr.press/v139/feng21g.html.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014. https://papers.neurips.cc/paper/5423-generative-adversarial-nets.

C. H. Grønbech, M. F. Vording, P. N. Timshel, C. K. Sønderby, T. H. Pers, and O. Winther. scVAE: variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, May 2020. ISSN 1367-4811. DOI:10.1093/bioinformatics/btaa293.

A. Hallou, H. G. Yevick, B. Dumitrascu, and V. Uhlmann. Deep learning for bioimage analysis in developmental biology. *Development*, 148(18), Sept. 2021. ISSN 1477-9129. DOI:10.1242/dev.199616.

Y. Hao, S. Hao, E. Andersen-Nissen, W. M. M. III, S. Zheng, A. Butler, M. J. Lee, A. J. Wilk, C. Darby, M. Zagar, P. Hoffman, M. Stoeckius, E. Papalexi, E. P. Mimitou, J. Jain, A. Srivastava, T. Stuart, L. B. Fleming, B. Yeung, A. J. Rogers, J. M. McElrath, C. A. Blish, R. Gottardo, P. Smibert, and R. Satija. Integrated analysis of multimodal single-cell data. *Cell*, 2021. DOI:10.1016/j.cell.2021.04.048.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv e-prints*, June 2018. arXiv:1706.08500.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, July 2012. arXiv:1207.0580.

R. Hou, E. Denisenko, and A. R. R. Forrest. scMatch: a single-cell gene expression profile annotation tool using reference datasets. *Bioinformatics*, 35(22):4688–4695, Apr. 2019. ISSN 1367-4811. DOI:10.1093/bioinformatics/btz292.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec. 1985. DOI:10.1007/bf01908075.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint*, Feb. 2015. arXiv:1502.03167.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint*, Dec. 2014. arXiv:1412.6980.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, Apr. 2014. arXiv:1312.6114.

M. Kramer. Autoassociative neural networks. *Computers & Chemical Engineering*, 16(4):313–328, Apr. 1992. DOI:10.1016/0098-1354(92)80051-a.

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, Feb. 1991. DOI:10.1002/aic.690370209.

J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, E. ad D. Amir, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, R. Finck, A. L. Gedman, I. Radtke, J. R. Downing, D. Pe'er, and G. P. Nolan. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, July 2015. DOI:10.1016/j.cell.2015.05.047.

Y. Lieberman, L. Rokach, and T. Shay. CaSTLe – classification of single cells by transfer learning: Harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments. *PLOS ONE*, 13(10):e0205499, Oct. 2018. ISSN 1932-6203. DOI:10.1371/journal.pone.0205499.

Y. Lin, S. Ghazanfar, K. Y. X. Wang, J. A. Gagnon-Bartsch, K. K. Lo, X. Su, Z.-G. Han, J. T. Ormerod, T. P. Speed, P. Yang, and J. Y. H. Yang. scMerge leverages factor analysis, stable expression, and pseudoreplication to merge multiple single-cell RNA-seq datasets. *Proceedings of the National Academy of Sciences*, 116(20):9775–9784, Apr. 2019. ISSN 1091-6490. DOI:10.1073/pnas.1820006116.

G. C. Linderman, J. Zhao, and Y. Kluger. Zero-preserving imputation of scrna-seq data using low-rank approximation. *bioRxiv*, Aug. 2018. DOI:10.1101/397588.

R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, Nov. 2018. ISSN 1548-7105. DOI:10.1038/s41592-018-0229-2.

M. Lotfollahi, F. A. Wolf, and F. J. Theis. scGen predicts single-cell perturbation responses. *Nature Methods*, 16(8):715–721, July 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0494-8.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13. JMLR.org, 2013. https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016. arXiv:1511.05644.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv e-prints*, Feb. 2018. arXiv:1802.03426.

L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? *arXiv e-prints*, Jan. 2018. arXiv:1801.04406.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

E. Pierson and C. Yau. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology*, 16(1), Nov. 2015. ISSN 1474-760X. DOI:10.1186/s13059-015-0805-z.

RAPIDS Development Team. *RAPIDS: Libraries for End to End GPU Data Science*, 2023. https://rapids.ai.

S. Rashid, S. Shah, Z. Bar-Joseph, and R. Pandya. Dhaka: variational autoencoder for unmasking tumor heterogeneity from single cell genomic data. *Bioinformatics*, 37(11):1535–1543, Feb. 2019. ISSN 1367-4811. DOI:10.1093/bioinformatics/btz095.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, June 2014. PMLR. http://proceedings.mlr.press/v32/rezende14.html.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, Nov. 1987. DOI:10.1016/0377-0427(87)90125-7.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. *arXiv e-prints*, June 2016. arXiv:1606.03498.

R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*, 33(5):495–502, Apr. 2015. ISSN 1546-1696. DOI:10.1038/nbt.3192.

D. Srivastava, A. Iyer, V. Kumar, and D. Sengupta. CellAtlasSearch: a scalable search engine for single cells. *Nucleic Acids Research*, 46(W1):W141–W147, May 2018. ISSN 1362-4962. DOI:10.1093/nar/gky421.

T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902.e21, June 2019. ISSN 0092-8674. DOI:10.1016/j.cell.2019.05.031.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv e-prints*, Dec. 2015. arXiv:1512.00567.

C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. *arXiv e-prints*, Oct. 2016. arXiv:1610.04490.

V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), Mar. 2019. DOI:10.1038/s41598-019-41695-z.

N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, June 2009. DOI:10.1145/1553374.1553511.

I. Virshup, S. Rybakov, F. J. Theis, P. Angerer, and F. A. Wolf. anndata: Annotated data. *bioRxiv*, Dec. 2021. DOI:10.1101/2021.12.16.473007.

B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, and S. Batzoglou. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature Methods*, 14(4):414–416, Mar. 2017. ISSN 1548-7105. DOI:10.1038/nmeth.4207.

D. Wang and J. Gu. VASC: Dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genomics, Proteomics & Bioinformatics*, 16(5):320–331, Oct. 2018. ISSN 1672-0229. DOI:10.1016/j.gpb.2018.08.003.

J. Wang, D. Agarwal, M. Huang, G. Hu, Z. Zhou, C. Ye, and N. R. Zhang. Data denoising with transfer learning in single-cell transcriptomics. *Nature Methods*, 16(9):875–878, Aug. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0537-1.

F. A. Wolf, P. Angerer, and F. J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), Feb. 2018. DOI:10.1186/s13059-017-1382-0.

Y. Wu, P. Tamayo, and K. Zhang. Visualizing and interpreting single-cell gene expression datasets with similarity weighted nonnegative embedding. *Cell Systems*, 7(6):656–666.e4, Dec. 2018. ISSN 2405-4712. DOI:10.1016/j.cels.2018.10.015.

Y. Wu, Y. Rivenson, H. Wang, Y. Luo, E. Ben-David, L. A. Bentolila, C. Pritz, and A. Ozcan. Three-dimensional virtual refocusing of fluorescence microscopy images using deep learning. *Nature Methods*, 16(12):1323–1331, Nov. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0622-5.

R. Yang, D. M. Vo, and H. Nakayama. Stochastically flipping labels of discriminator's outputs for training generative adversarial networks. *IEEE Access*, 10:103644–103654, 2022. DOI:10.1109/access.2022.3210130.

Y. Yazici, C.-S. Foo, S. Winkler, K.-H. Yap, and V. Chandrasekhar. Empirical analysis of overfitting and mode drop in gan training. In *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, Oct. 2020. DOI:10.1109/icip40778.2020.9191083.

A. W. Zhang, C. O'Flanagan, E. A. Chavez, J. L. P. Lim, N. Ceglia, A. McPherson, M. Wiens, P. Walters, T. Chan, B. Hewitson, D. Lai, A. Mottok, C. Sarkozy, L. Chong, T. Aoki, X. Wang, A. P. Weng, J. N. McAlpine, S. Aparicio, C. Steidl, K. R. Campbell, and S. P. Shah. Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nature Methods*, 16(10):1007–1015, Sept. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0529-1.

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *arXiv e-prints*, Jan. 2018. arXiv:1801.03924.

G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.*, 8:14049, 2017. DOI:10.1038/ncomms14049.

A.2.1 Supplementary materials

SUPPLEMENTARY MATERIALS

scAAE: Adversarial autoencoders for single-cell gene expression data

Christopher Heje Grønbech, Emiliano Molinaro, Ole Winther, and Kedar Natarajan

Last typeset on 20 January 2025

S1 Supplementary Tables

Table S1. Sources for scRNA-seq gene expression data sets. The PBMC-92k data set has been compiled from cell-type-specific data sets, so sources for each of these are listed.

Data set	Cell type	Source[a]
PBMC-3k	—	`3-k-pbm-cs-from-a-healthy-donor-1-standard-1-1-0`
PBMC-8k	—	`8-k-pbm-cs-from-a-healthy-donor-2-standard-2-1-0`
PBMC-10k	—	`10-k-pbm-cs-from-a-healthy-donor-v-3-chemistry-3-standard-3-0-0`
PBMC-68k[b]	—	`fresh-68-k-pbm-cs-donor-a-1-standard-1-1-0`
PBMC-92k	CD8+	`cd-8-plus-cytotoxic-t-cells-1-standard-1-1-0`
PBMC-92k	CD8+/CD45RA+	`cd-8-plus-cd-45-r-aplus-naive-cytotoxic-t-cells-1-standard-1-1-0`
PBMC-92k	CD56+	`cd-56-plus-natural-killer-cells-1-standard-1-1-0`
PBMC-92k	CD4+	`cd-4-plus-helper-t-cells-1-standard-1-1-0`
PBMC-92k	CD4+/CD45RO+	`cd-4-plus-cd-45-r-oplus-memory-t-cells-1-standard-1-1-0`
PBMC-92k	CD4+/CD45RA+/CD25−	`cd-4-plus-cd-45-r-aplus-cd-25-naive-t-cells-1-standard-1-1-0`
PBMC-92k	CD4+/CD25+	`cd-4-plus-cd-25-plus-regulatory-t-cells-1-standard-1-1-0`
PBMC-92k	CD34+	`cd-34-plus-cells-1-standard-1-1-0`
PBMC-92k	CD19+	`cd-19-plus-b-cells-1-standard-1-1-0`

[a] Prepend `https://www.10xgenomics.com/resources/datasets/`.
[b] The source for the cell-type annotation is available at `https://github.com/10XGenomics/single-cell-3prime-paper`.

Table S2. Reference clustering details for PBMC data sets with no author-annotated clusters or cell-type labels.

Data set	Resolution	
	Louvain method	Leiden method
PBMC-3k	1.0	—
PBMC-8k	0.7	—
PBMC-10k	0.6	0.6

Table S3. Values investigated for the dimension ℓ of the latent variable as well as for the architecture of the deep neural networks of the encoder, the decoder, and the discriminator. The encoder has networks with two layers with H_1 in the first one and H_2 in the second one, and the networks in the decoder has the reverse architecture. The discriminator has networks with two to five layers with layer k having H_k units. For each row, four experiments is run with an increasing number of layers for the discriminator networks. The standard network architecture (with only two-layer discriminator networks) uses the values in bold.

ℓ	H_1	H_2	H_3	H_4	H_5
10	100	50	25	12	6
10	250	125	62	31	16
10	500	250	125	62	31
25	100	50	25	12	6
25	**250**	**125**	62	31	16
25	500	250	125	62	31
50	100	50	25	12	6
50	250	125	62	31	16
50	500	250	125	62	31

Table S4. Hyperparameter values investigated for the discriminator noise methods. The instance noise standard deviation is also decayed with a rate of 0.99.

Level	Flipping rate r	Smoothing scale s	Noise standard deviation d
Low	0.1	0.1	0.05
High	0.5	0.3	0.2

Table S5. Hyperparameter values investigated for clustering using the Louvain and Leiden methods. All values are used for configuration searches, and only bolded values are used early stopping.

Resolution	Neighbourhood size
0.4	5
0.6	10
0.8	**15**
1.0	20
1.2	25

Table S6. Training and evaluation options for each data set.

	Configuration search		Early stopping	
Data set	Epoch count	Clustering interval	Patience	Delay
SIM-2	200	20	—	—
SIM-5	200	20	—	—
SIM-10	200	20	—	—
PBMC-3k	200	20	25	10
PBMC-8k	200	20	25	10
PBMC-10k	200	20	25	10
PBMC-68k	50	10	25	1
PBMC-92k	25	5	25	1

S2 Supplementary Figures

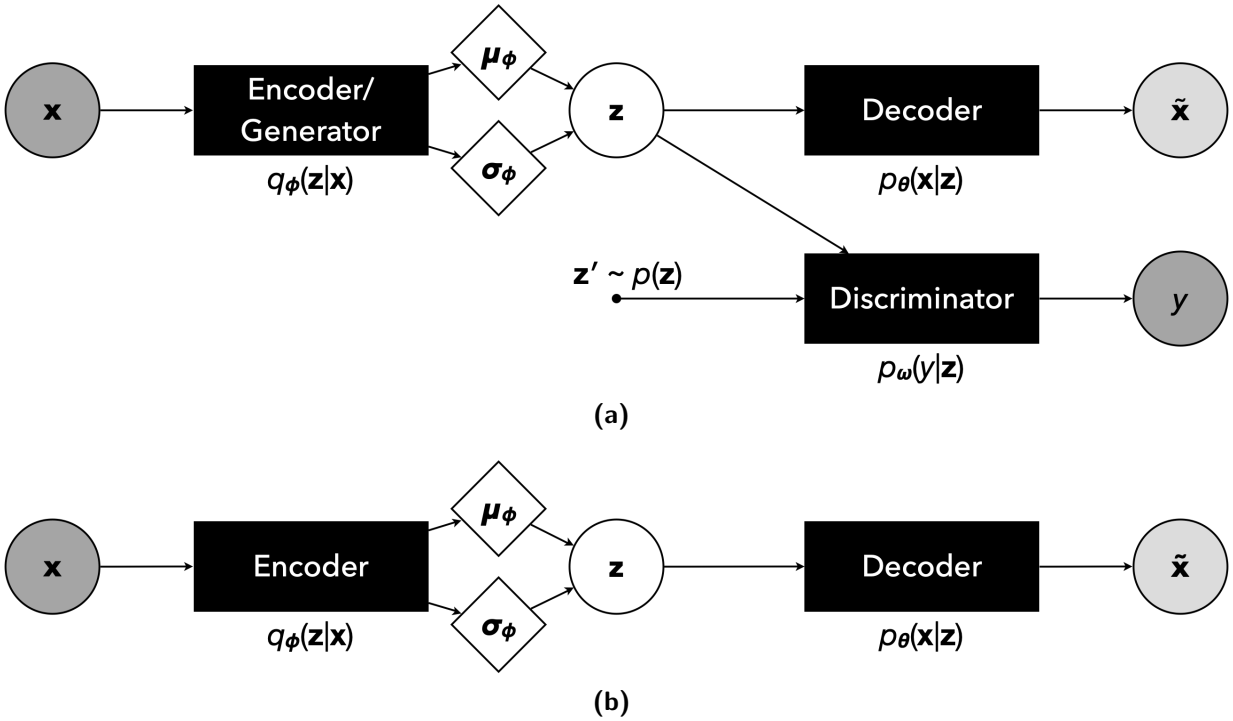
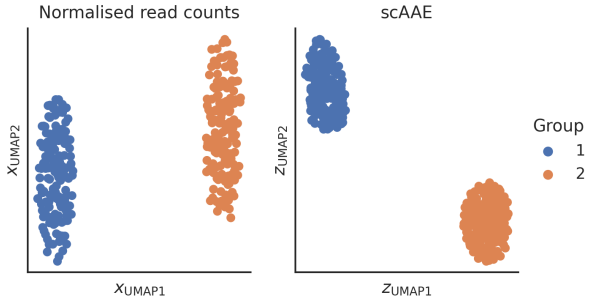
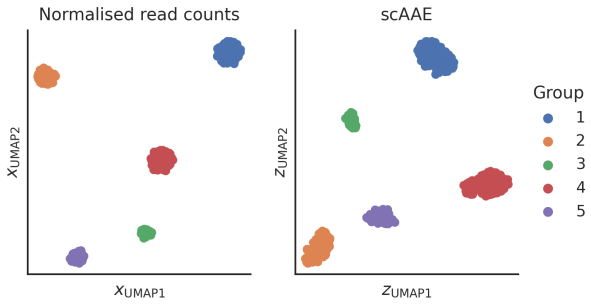


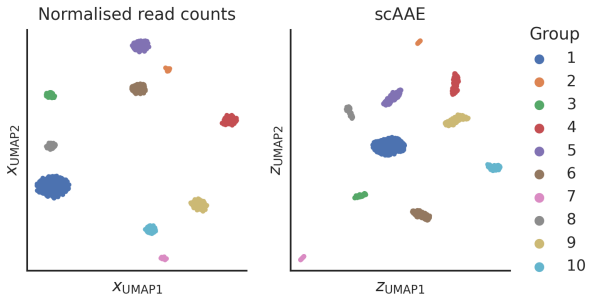
Fig. S1. Model graphs of (a) the adversarial autoencoder and (b) the variational autoencoder. Circles represent variables that are observed (dark grey), reconstructed (light grey), and latent (white). The rhombi symbolise deterministic variables such as the distribution parameters. The black rectangles denote neural networks, and arrow indicate input and output.



(a) SIM-2.



(b) SIM-5.



(c) SIM-10.

Fig. S2. UMAP embeddings of the test sets of the simulated data sets. The embeddings to the left use the standardised gene expression counts, and the embedding to the right use the latent representations of the reference model at the optimal validation autoencoder loss.

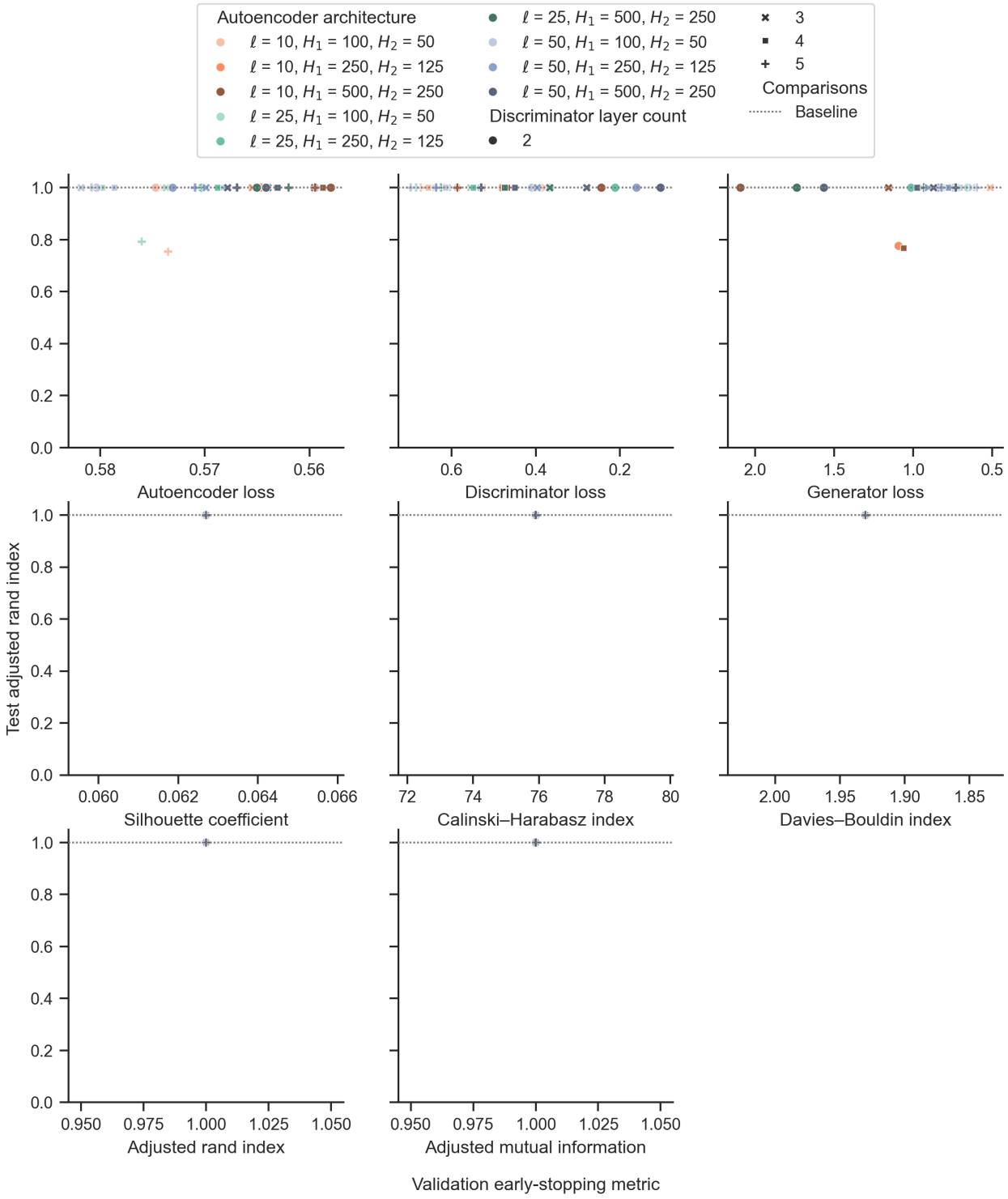


Fig. S3. Network-architecture search for the SIM-2 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

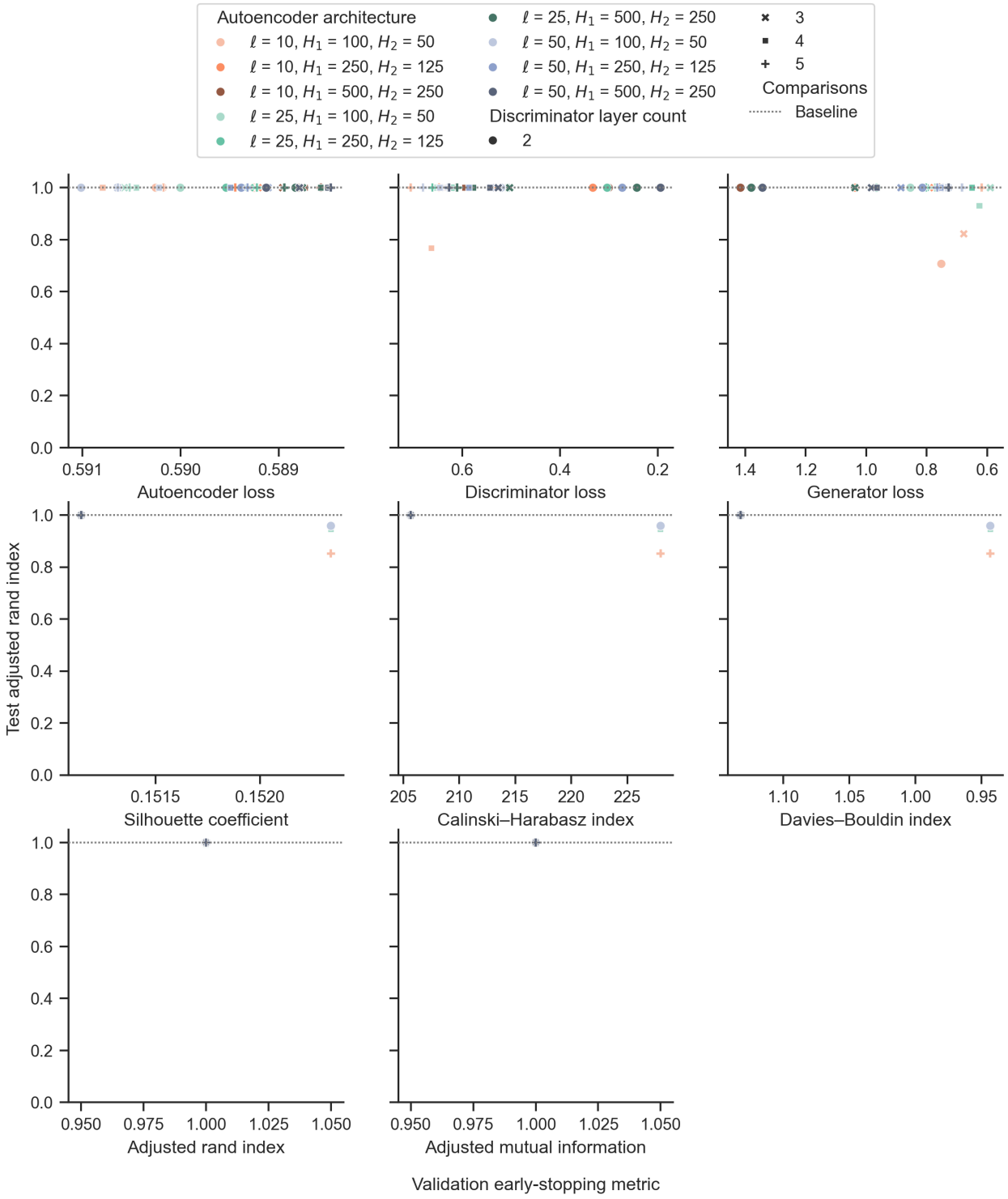


Fig. S4. Network-architecture search for the SIM-5 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

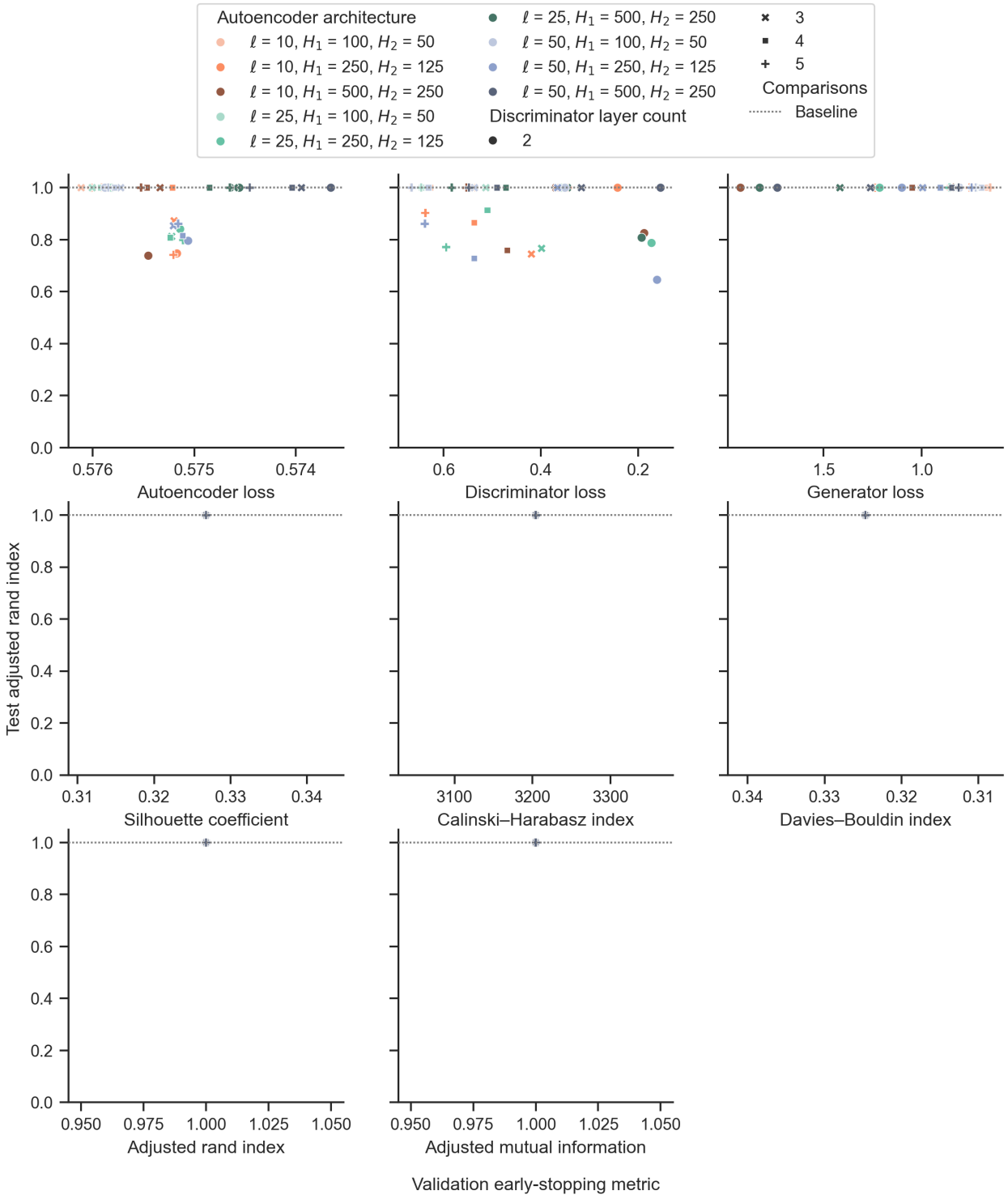


Fig. S5. Network-architecture search for the SIM-10 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

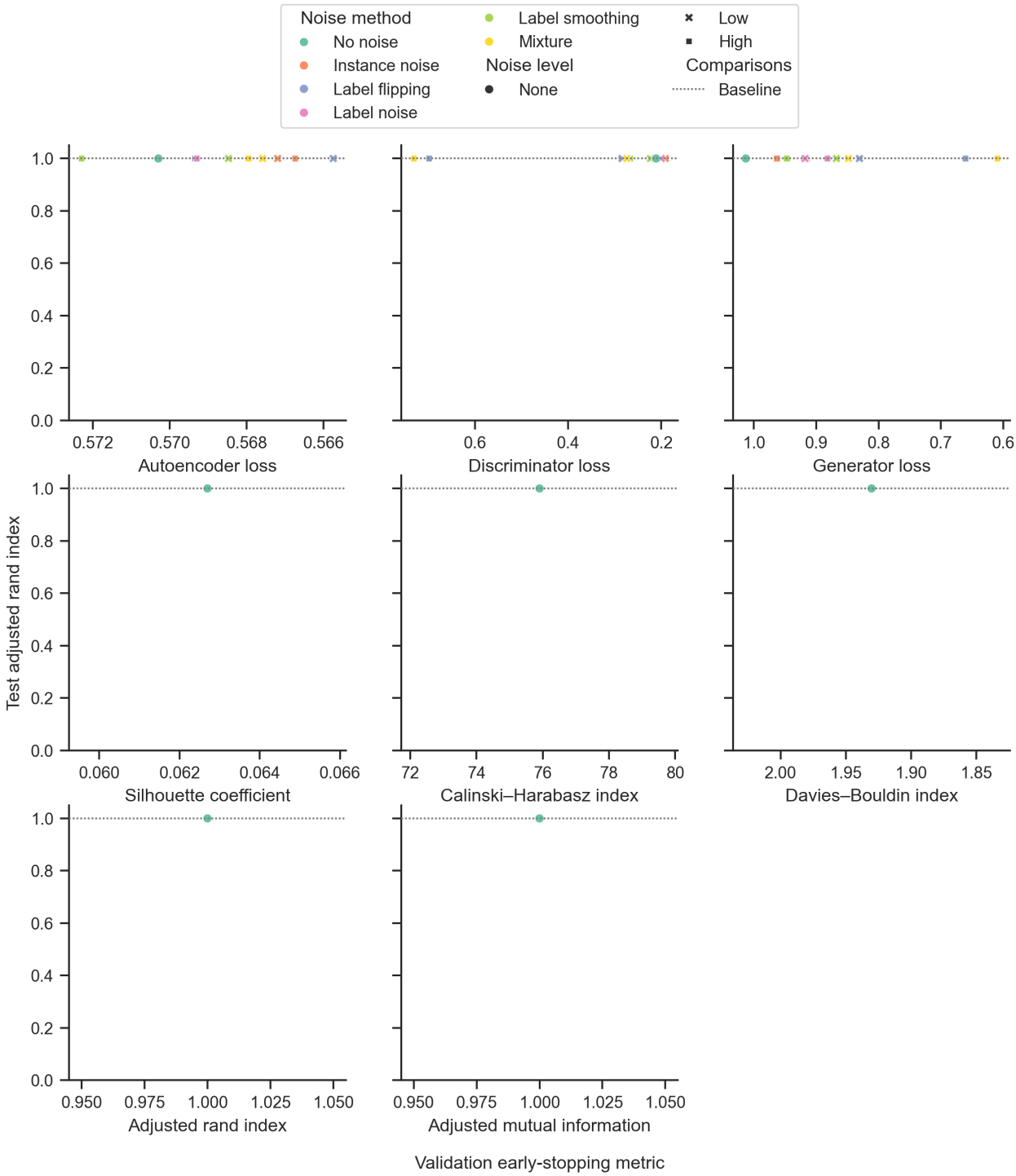


Fig. S6. Discriminator-noise-method search for the SIM-2 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

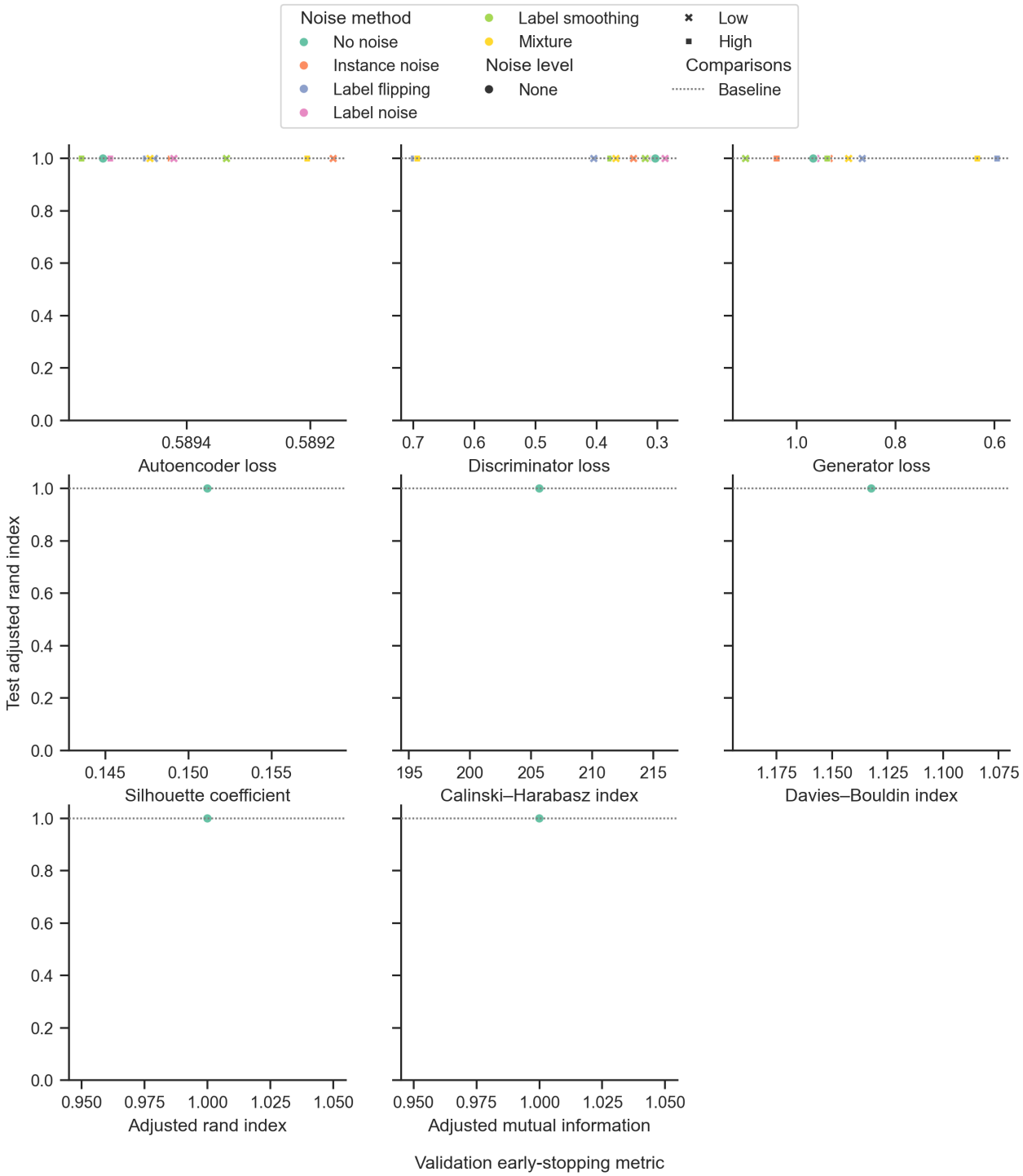


Fig. S7. Discriminator-noise-method search for the SIM-5 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

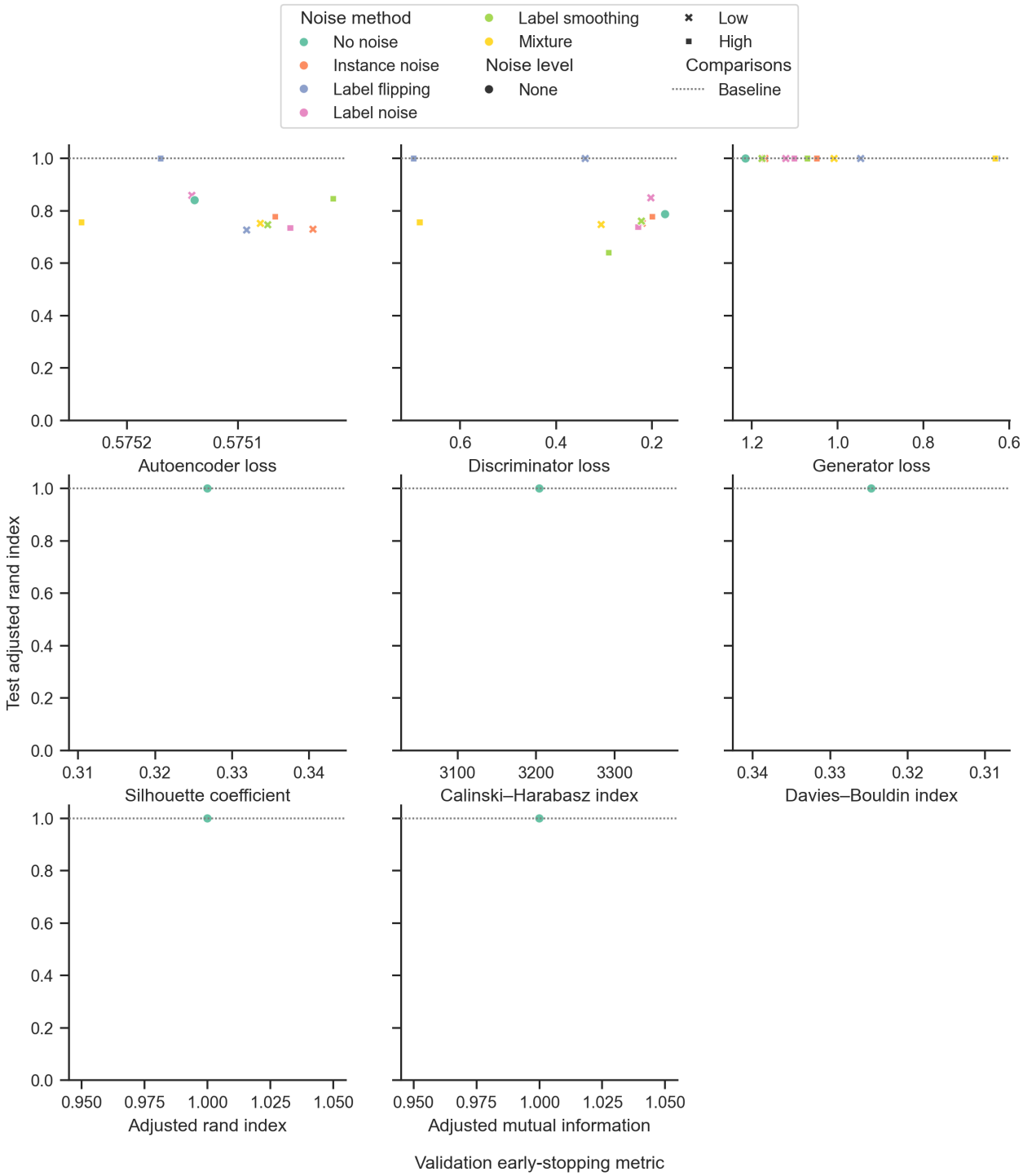


Fig. S8. Discriminator-noise-method search for the SIM-10 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

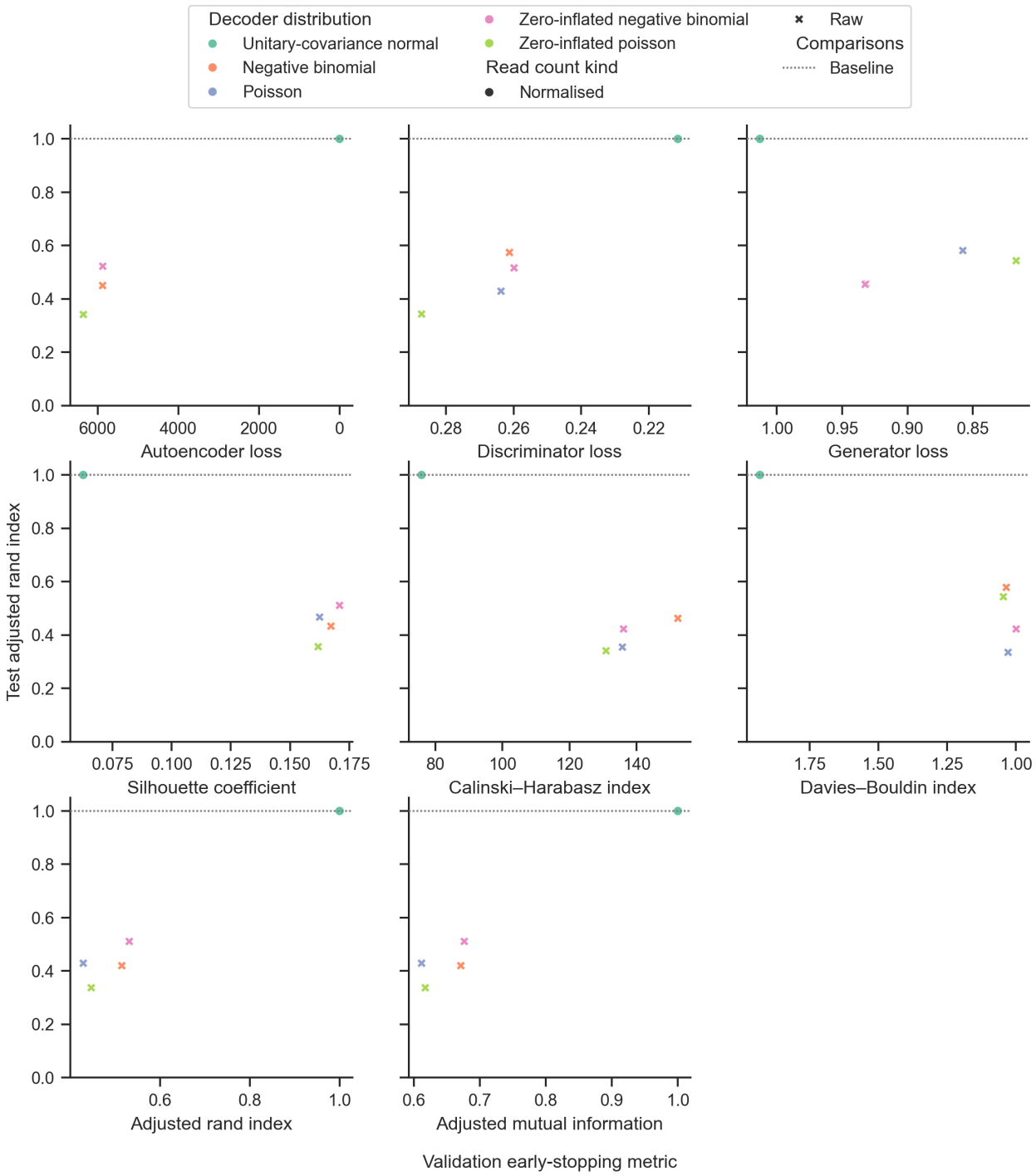


Fig. S9. Decoder-distribution search for the SIM-2 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

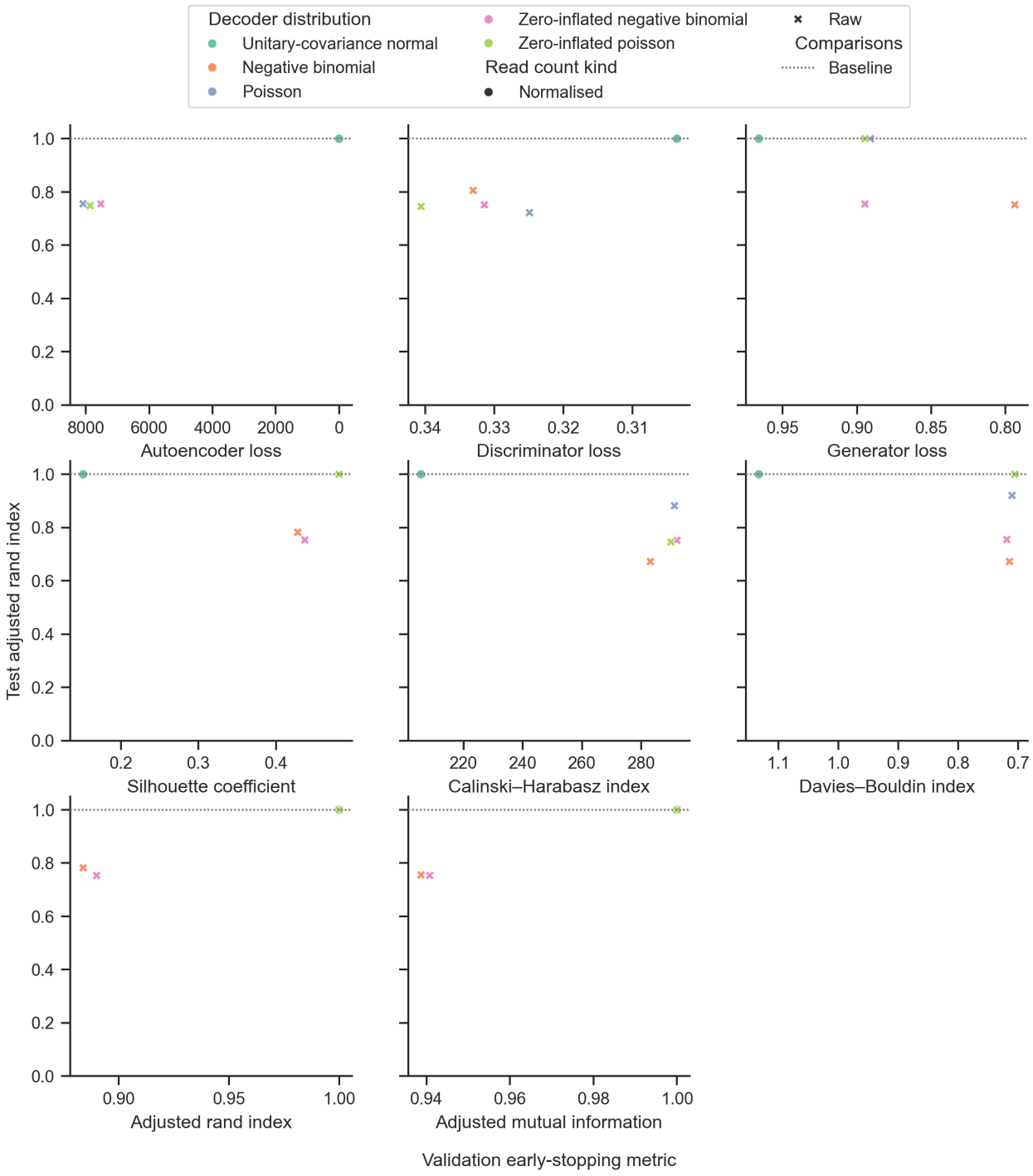


Fig. S10. Decoder-distribution search for the SIM-5 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

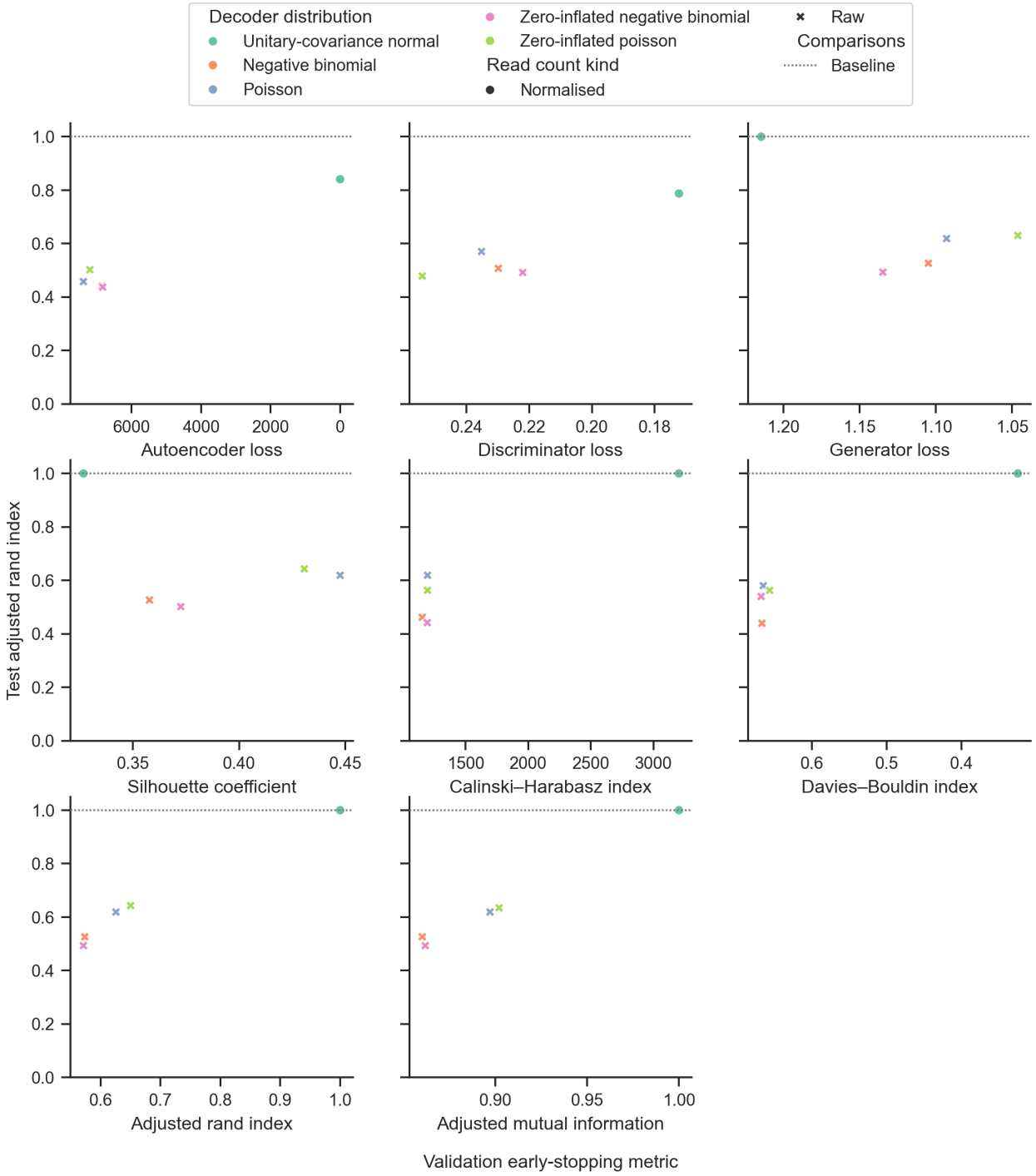
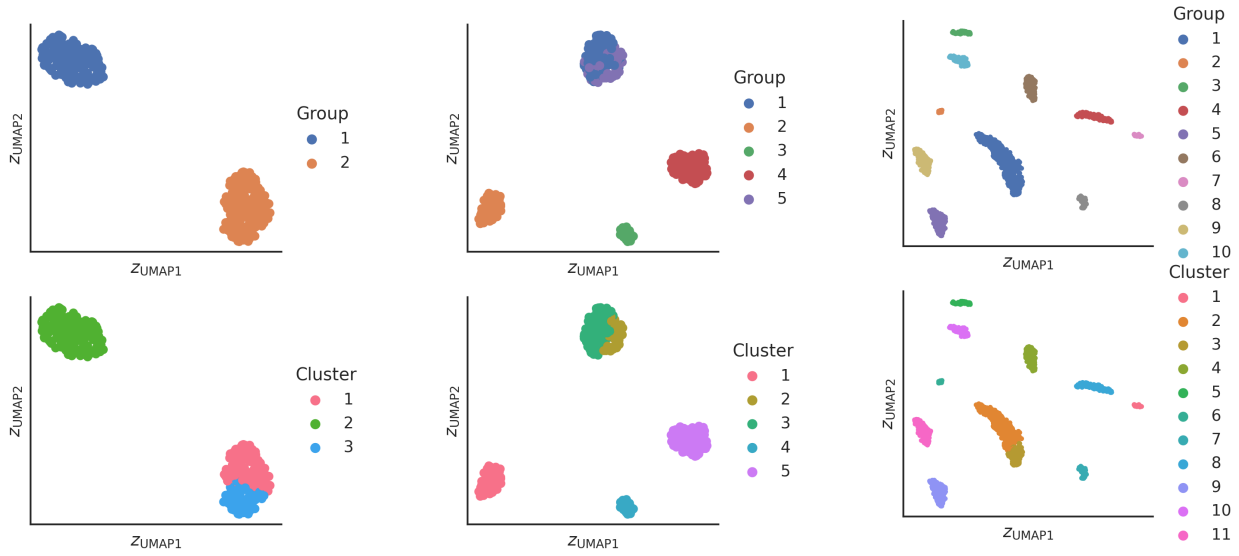
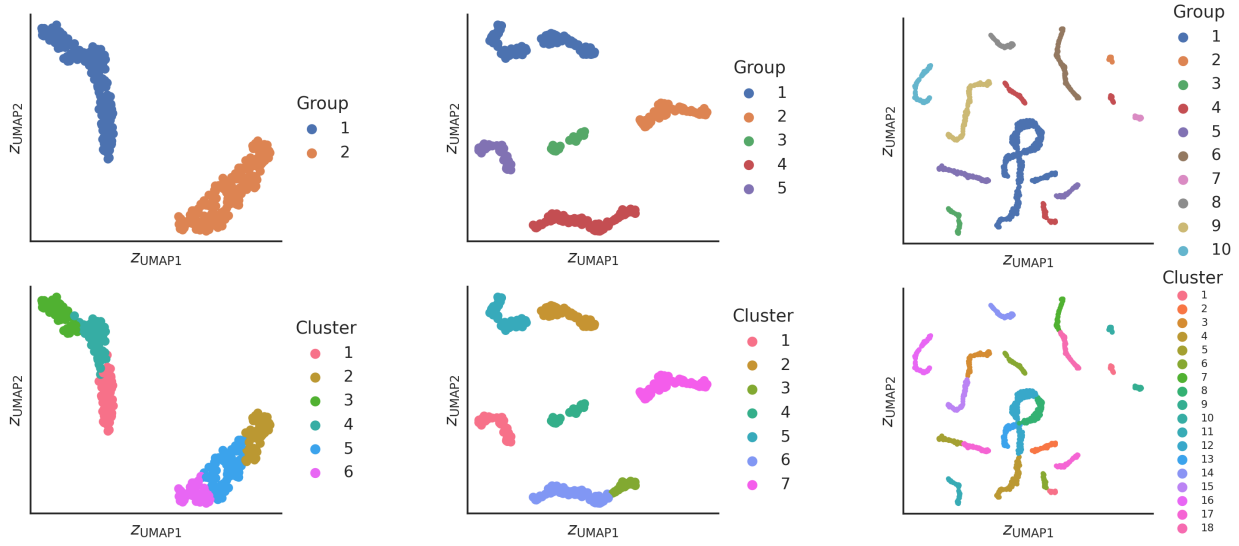


Fig. S11. Decoder-distribution search for the SIM-10 data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.



(a) Network architecture, SIM-2. **(b)** Network architecture, SIM-5. **(c)** Network architecture, SIM-10.



(d) Decoder distribution, SIM-2. **(e)** Decoder distribution, SIM-5. **(f)** Decoder distribution, SIM-10.

Fig. S12. UMAP embeddings of the latent representation of the simulated data sets for some models achieving a test adjusted rand index less than the baseline method. For each of the simulated data sets, an unsuccessful latent representation is shown for both the network-architecture search and the decoder-distribution search.

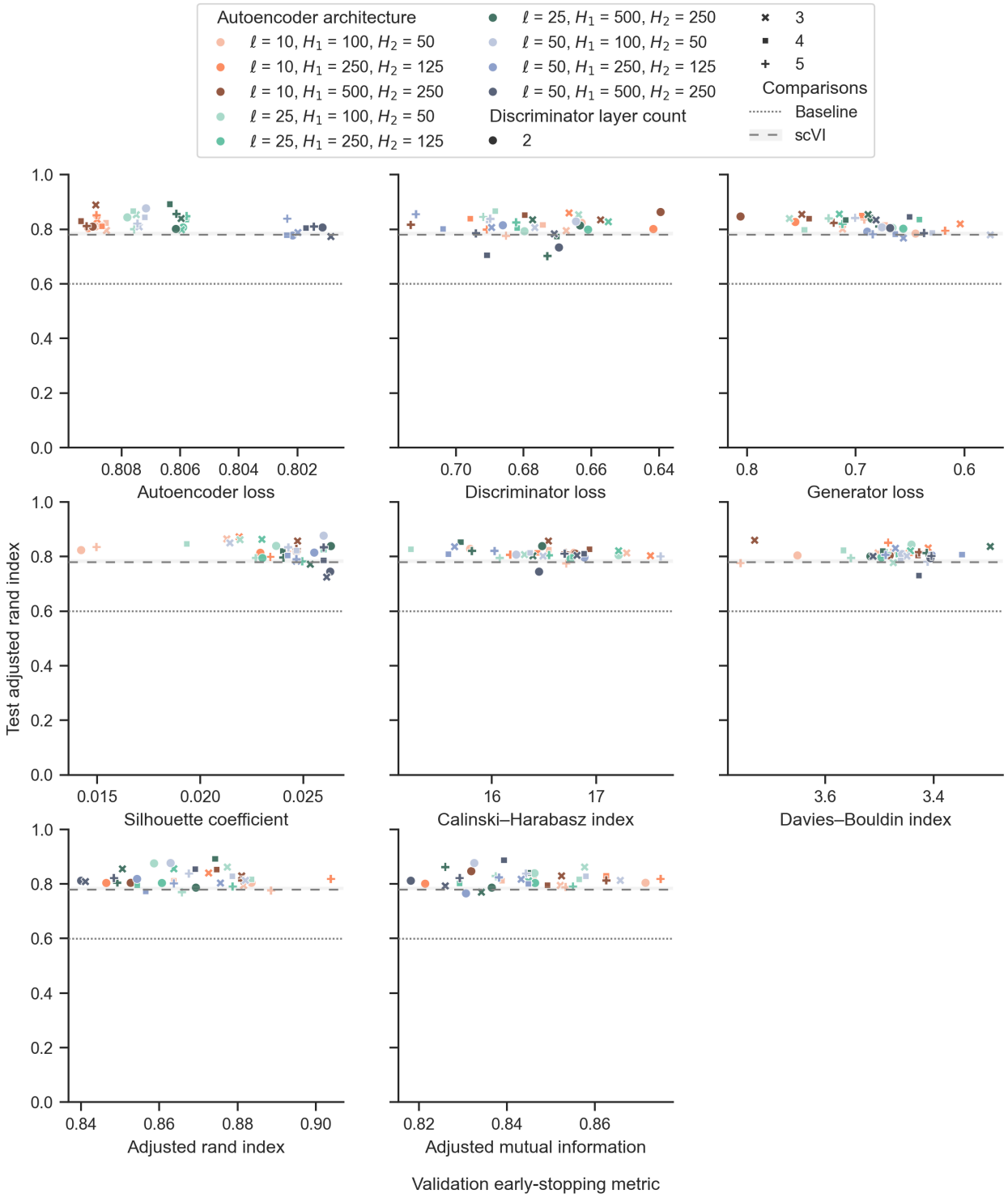


Fig. S13. Network-architecture search for the PBMC-3k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

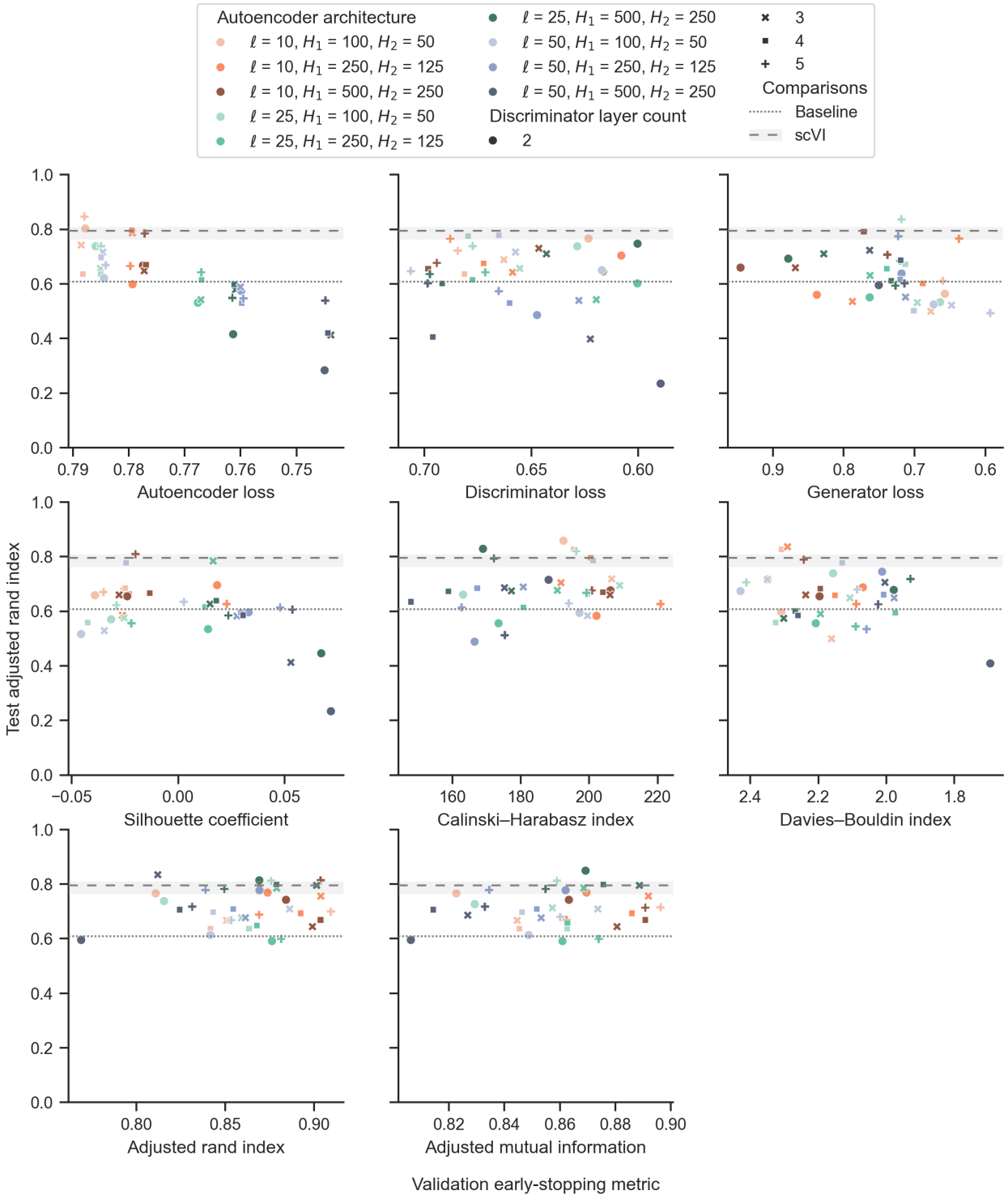


Fig. S14. Network-architecture search for the PBMC-8k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

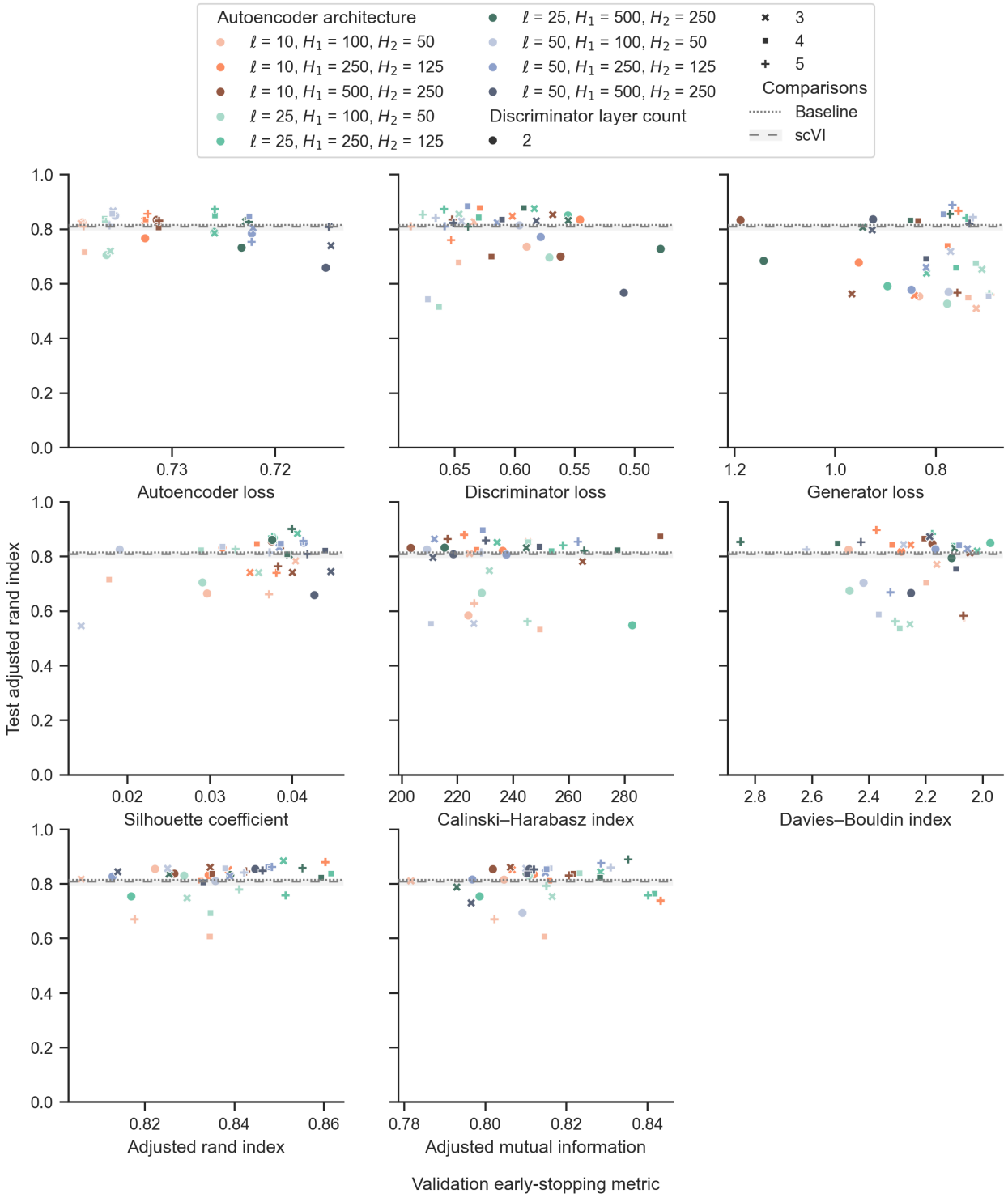


Fig. S15. Network-architecture search for the PBMC-10k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

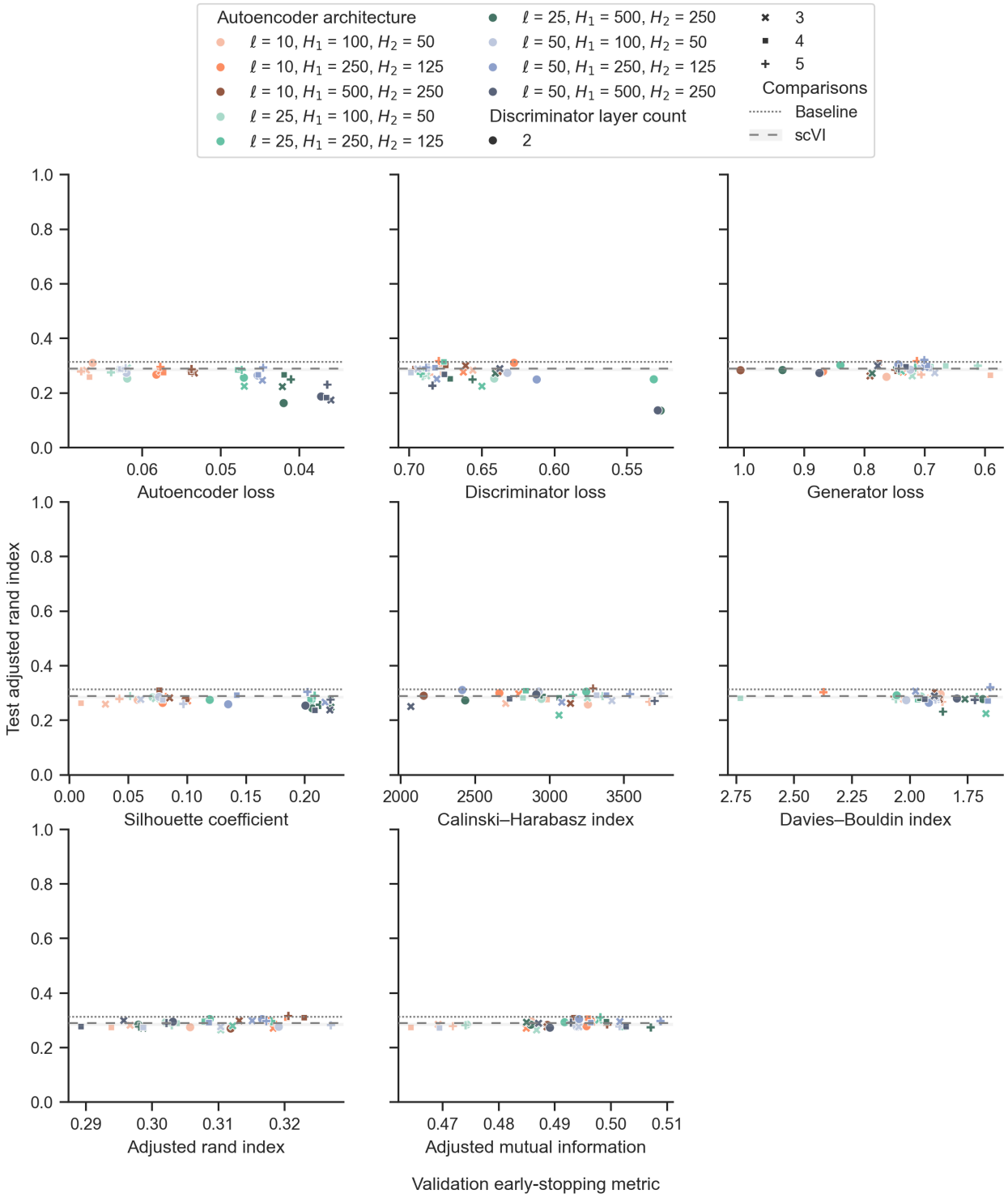


Fig. S16. Network-architecture search for the PBMC-68k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

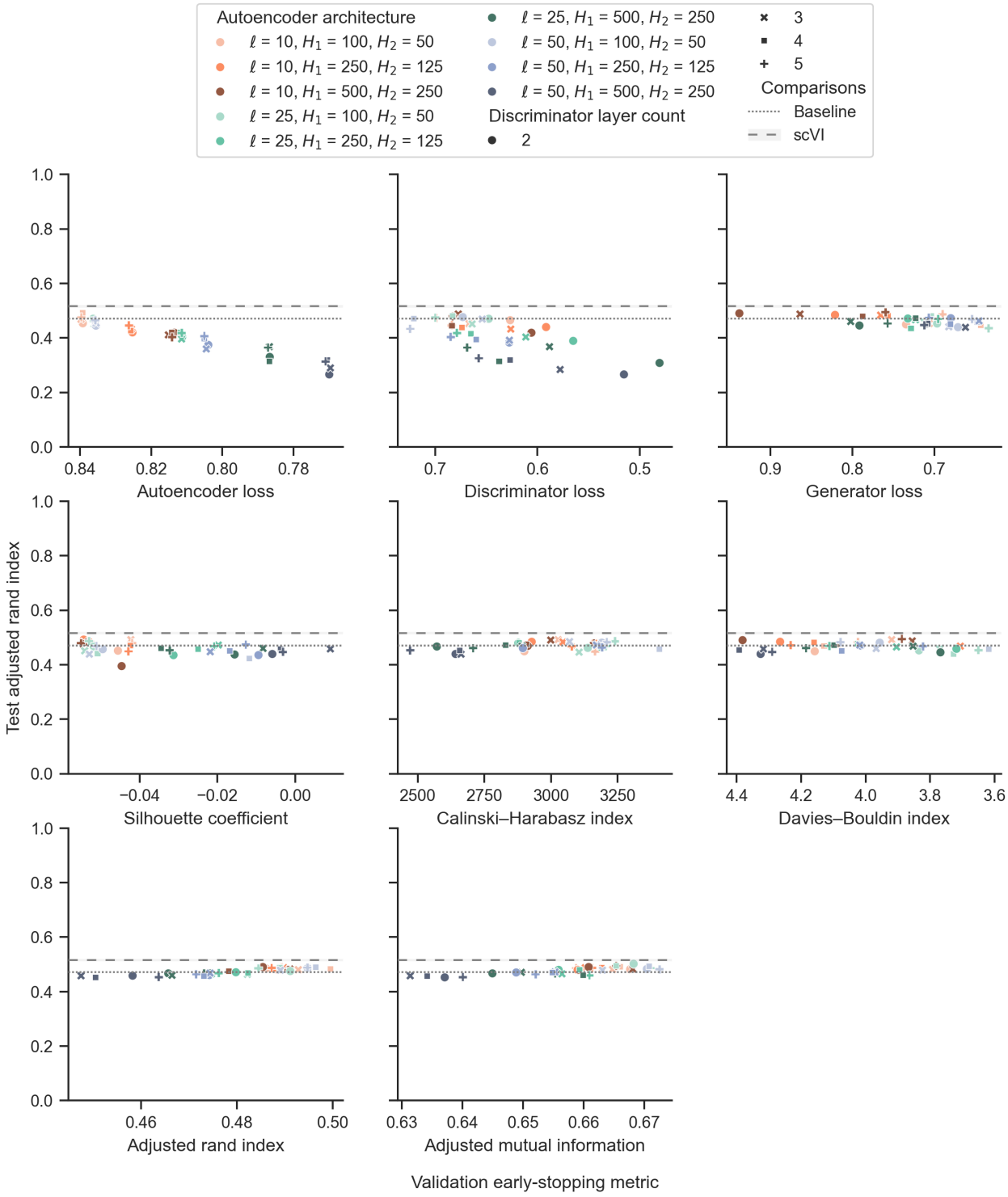


Fig. S17. Network-architecture search for the PBMC-92k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

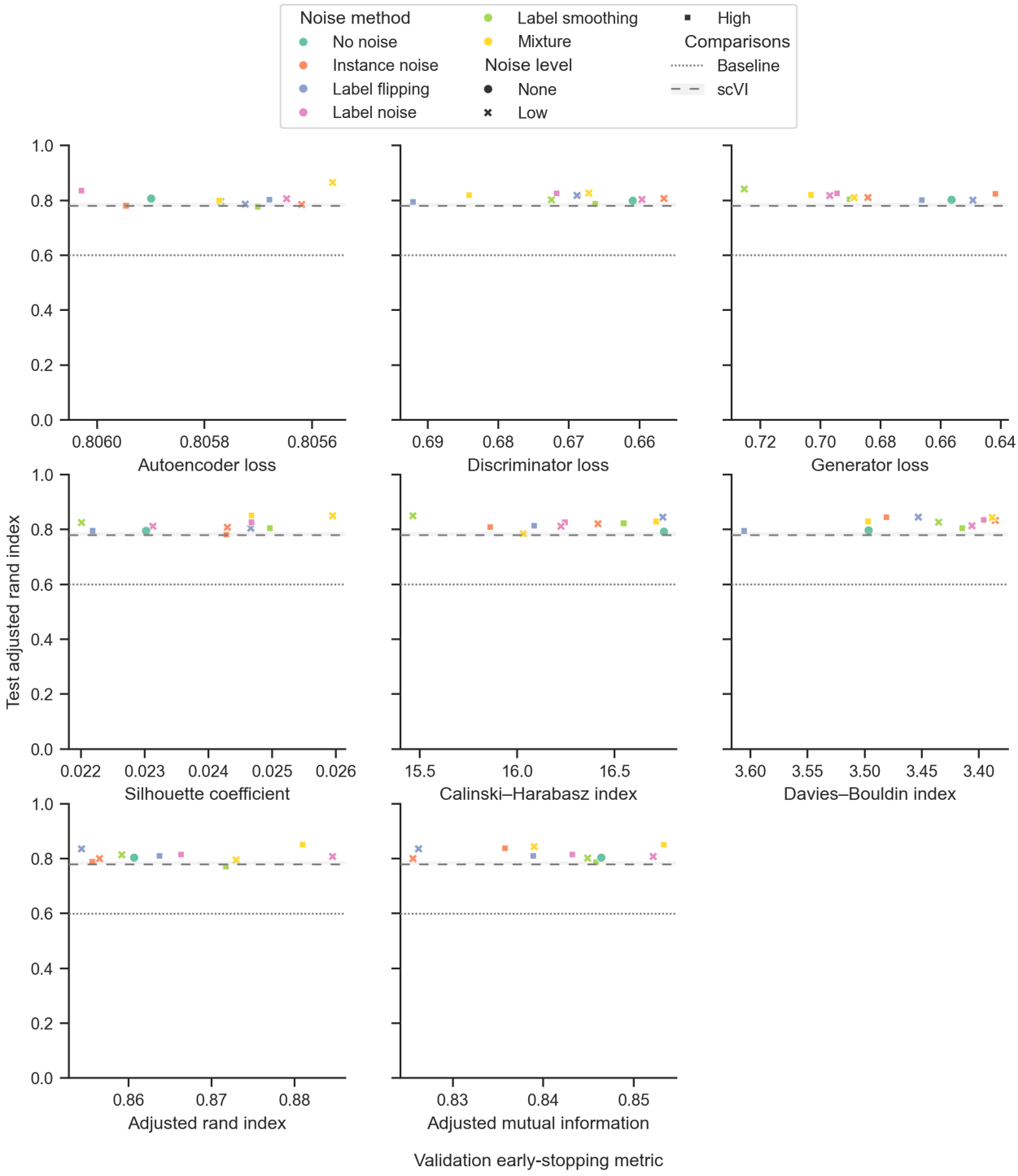


Fig. S18. Discriminator-noise-method search for the PBMC-3k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

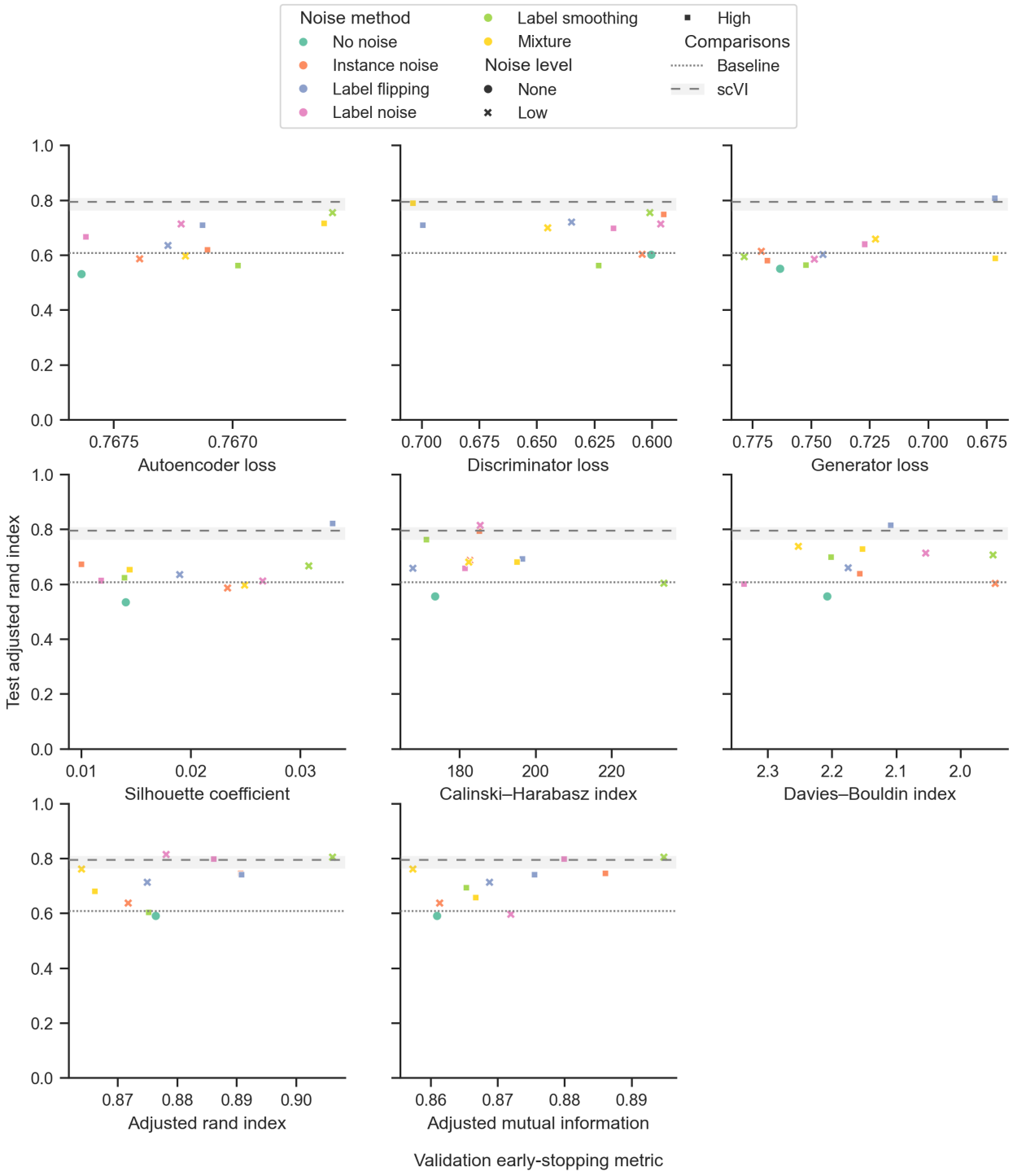


Fig. S19. Discriminator-noise-method search for the PBMC-8k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

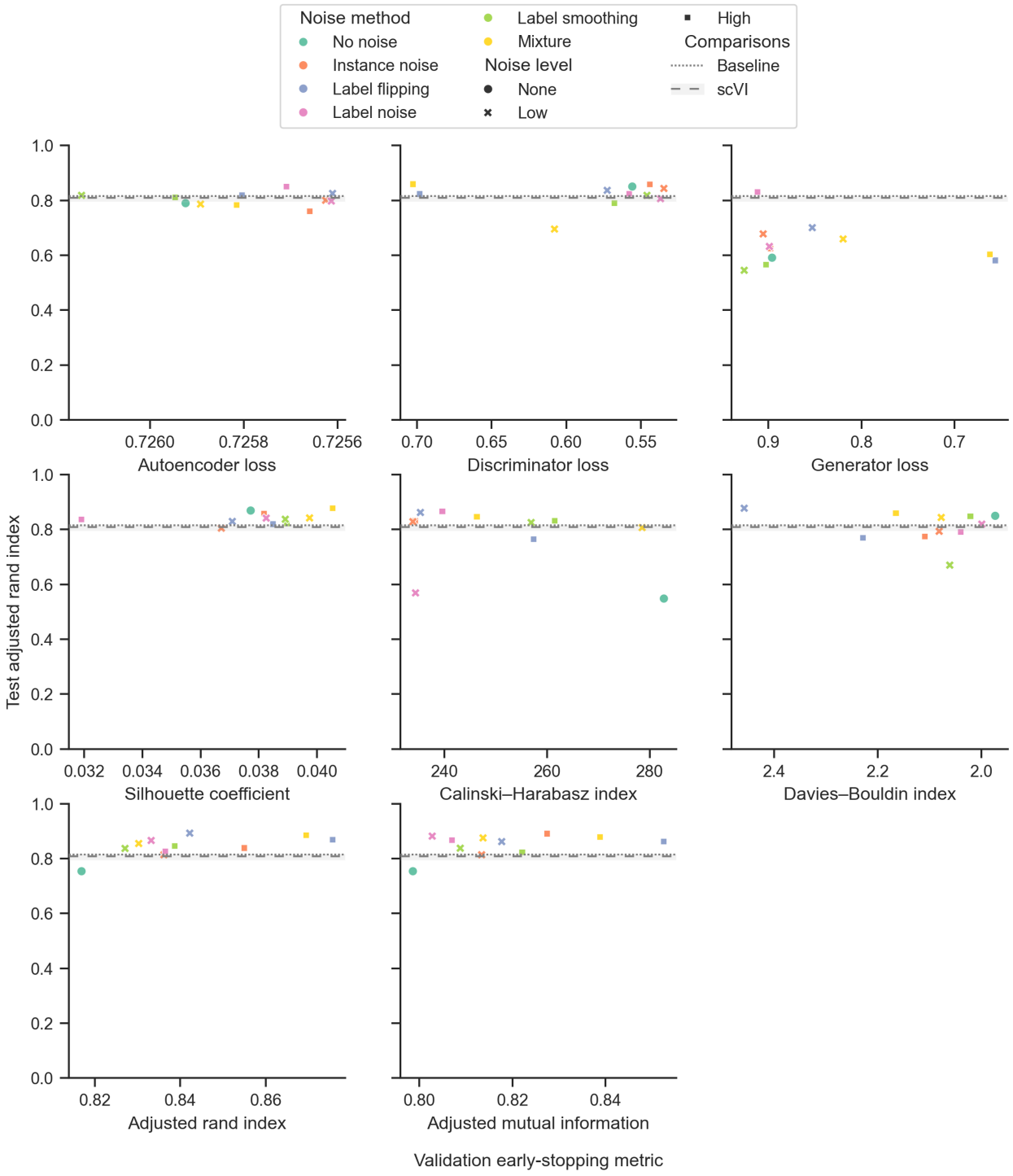


Fig. S20. Discriminator-noise-method search for the PBMC-10k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

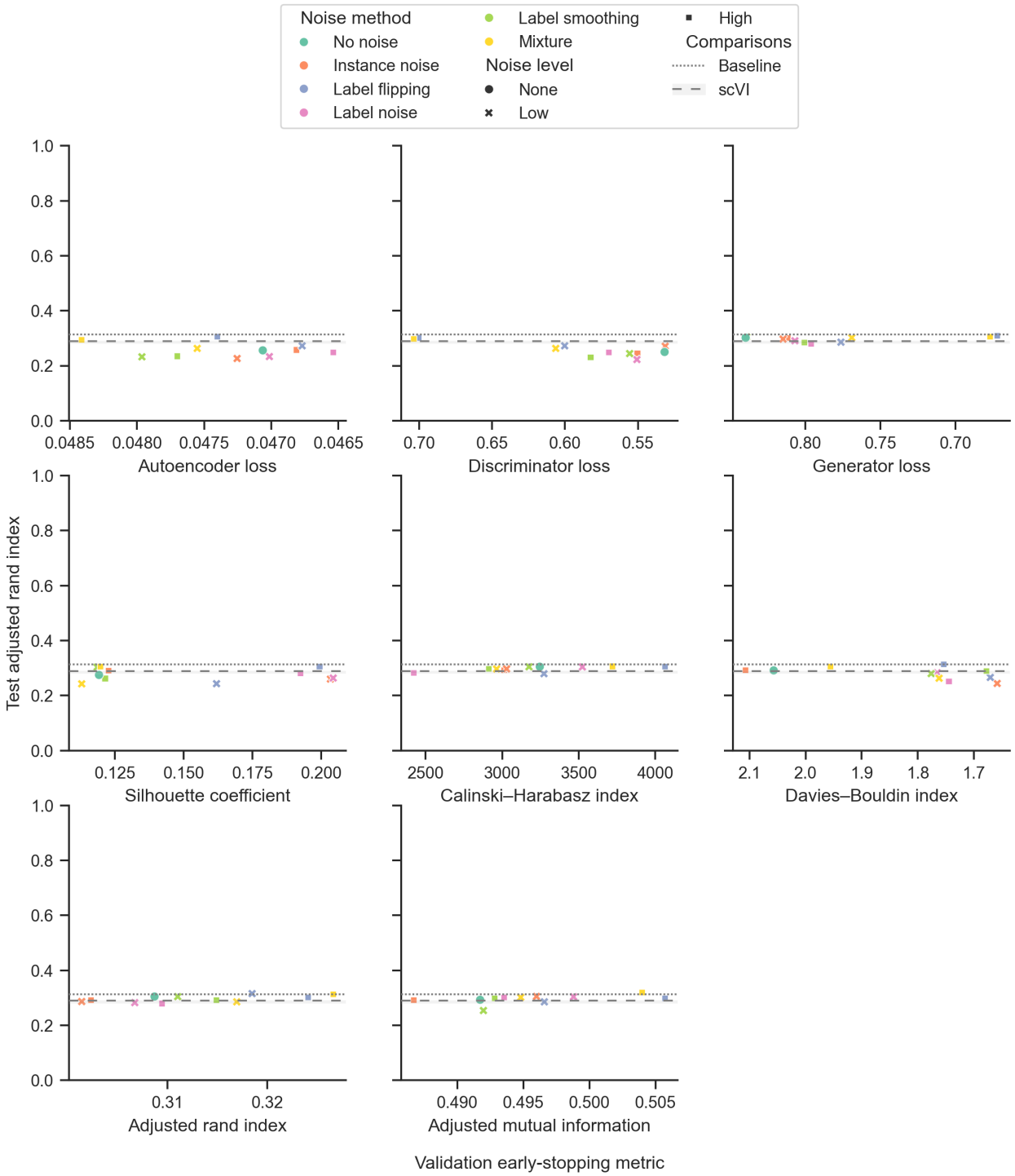


Fig. S21. Discriminator-noise-method search for the PBMC-68k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

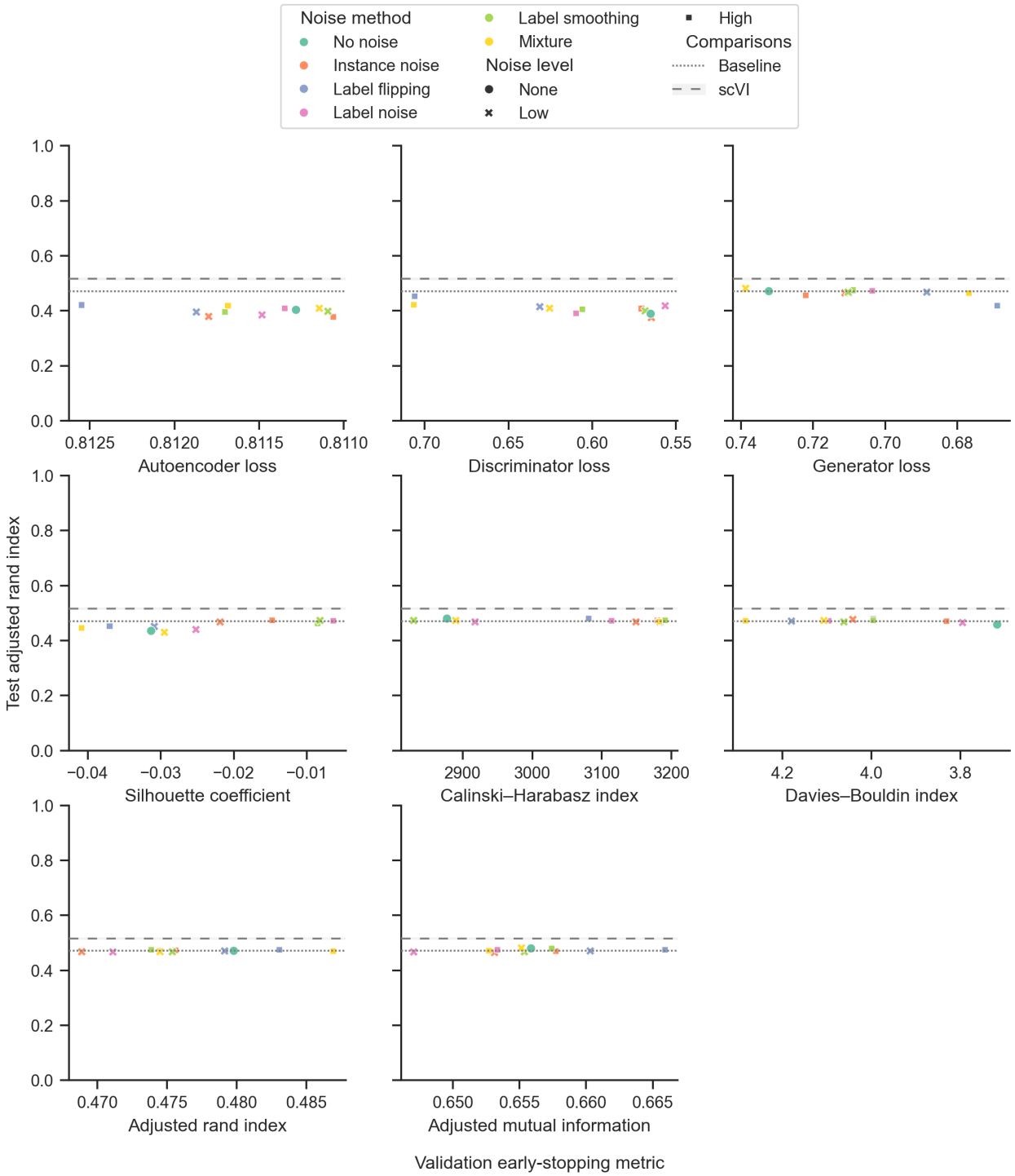


Fig. S22. Discriminator-noise-method search for the PBMC-92k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

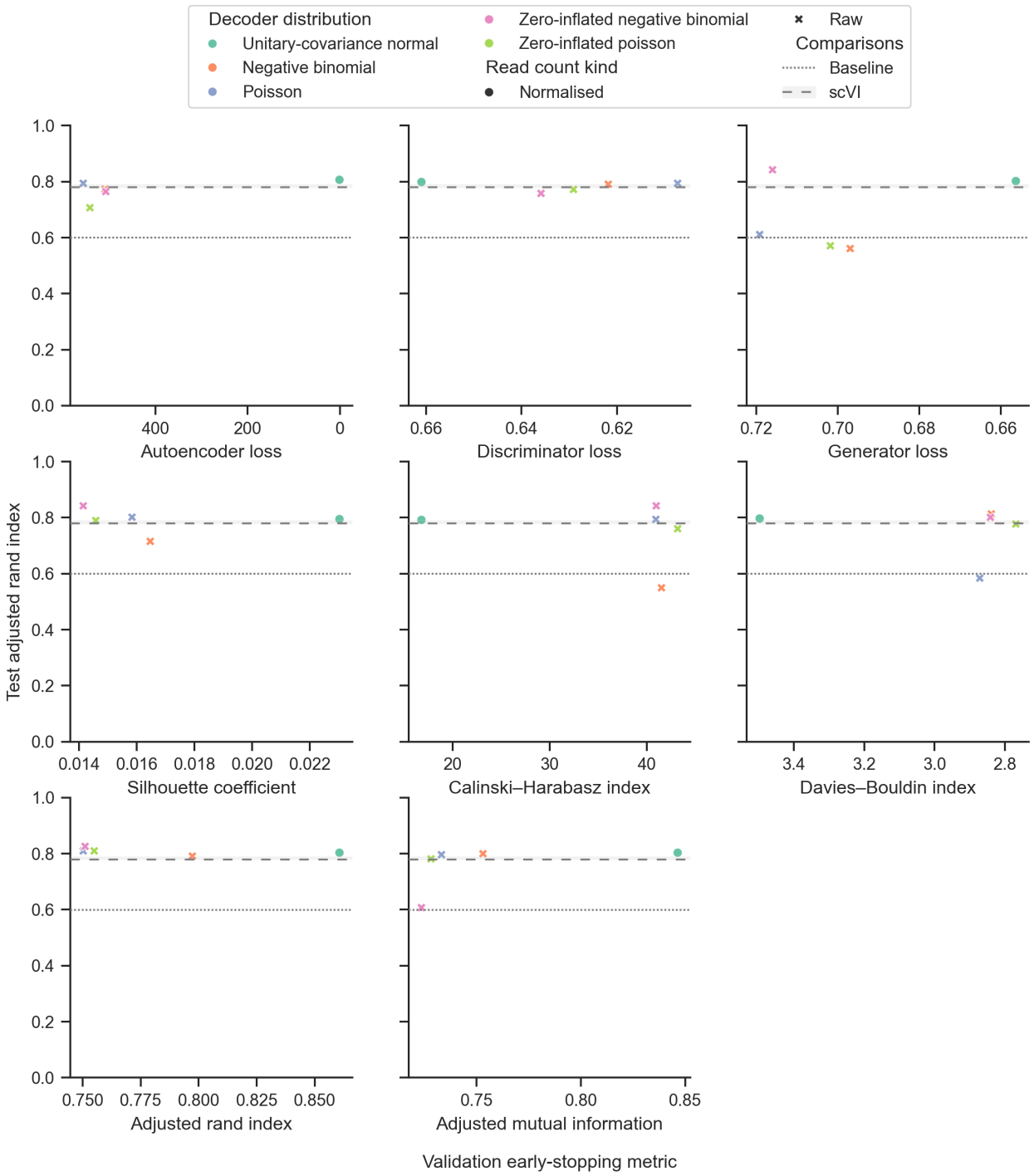


Fig. S23. Decoder-distribution search for the PBMC-3k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

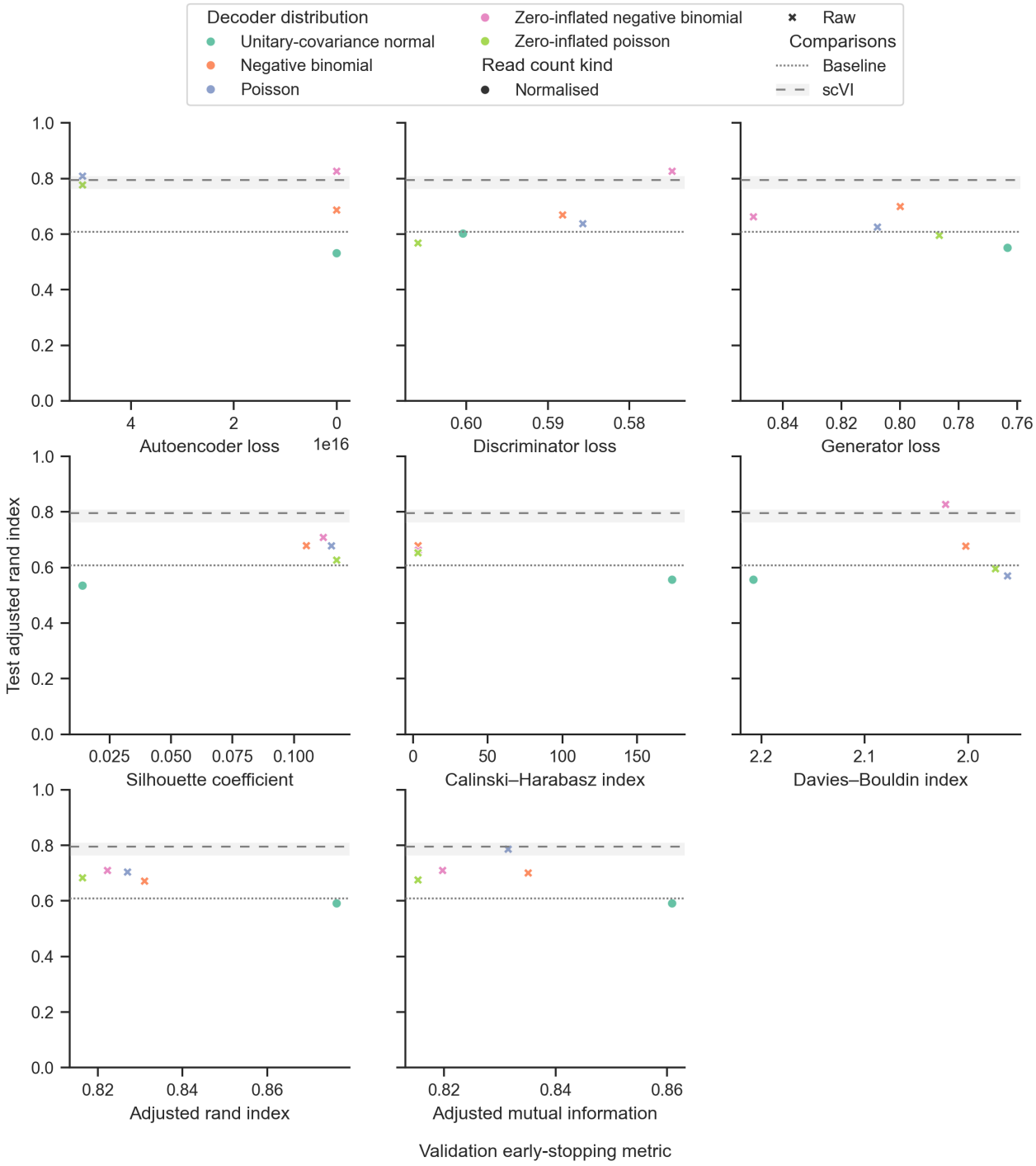


Fig. S24. Decoder-distribution search for the PBMC-8k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

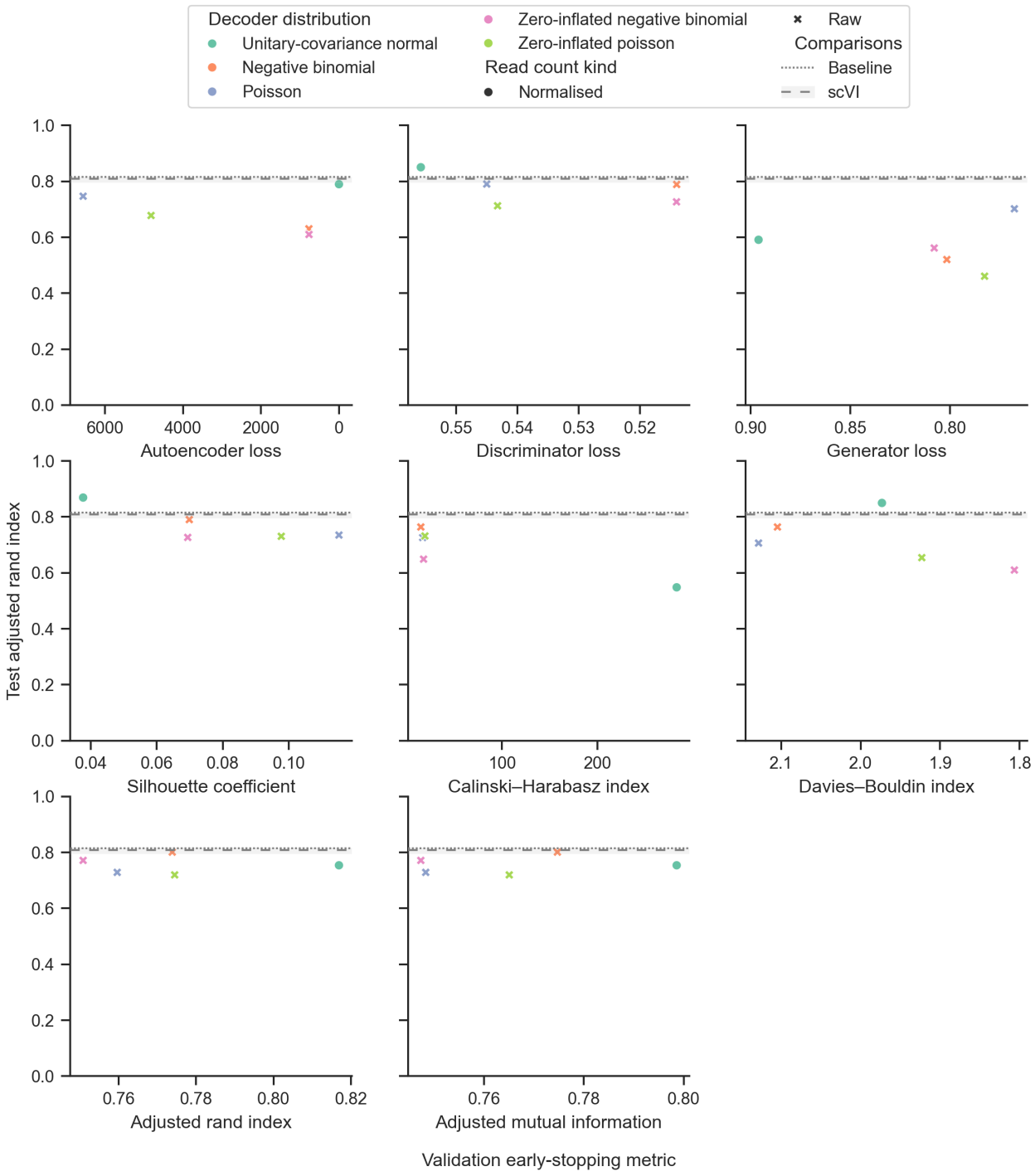


Fig. S25. Decoder-distribution search for the PBMC-10k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

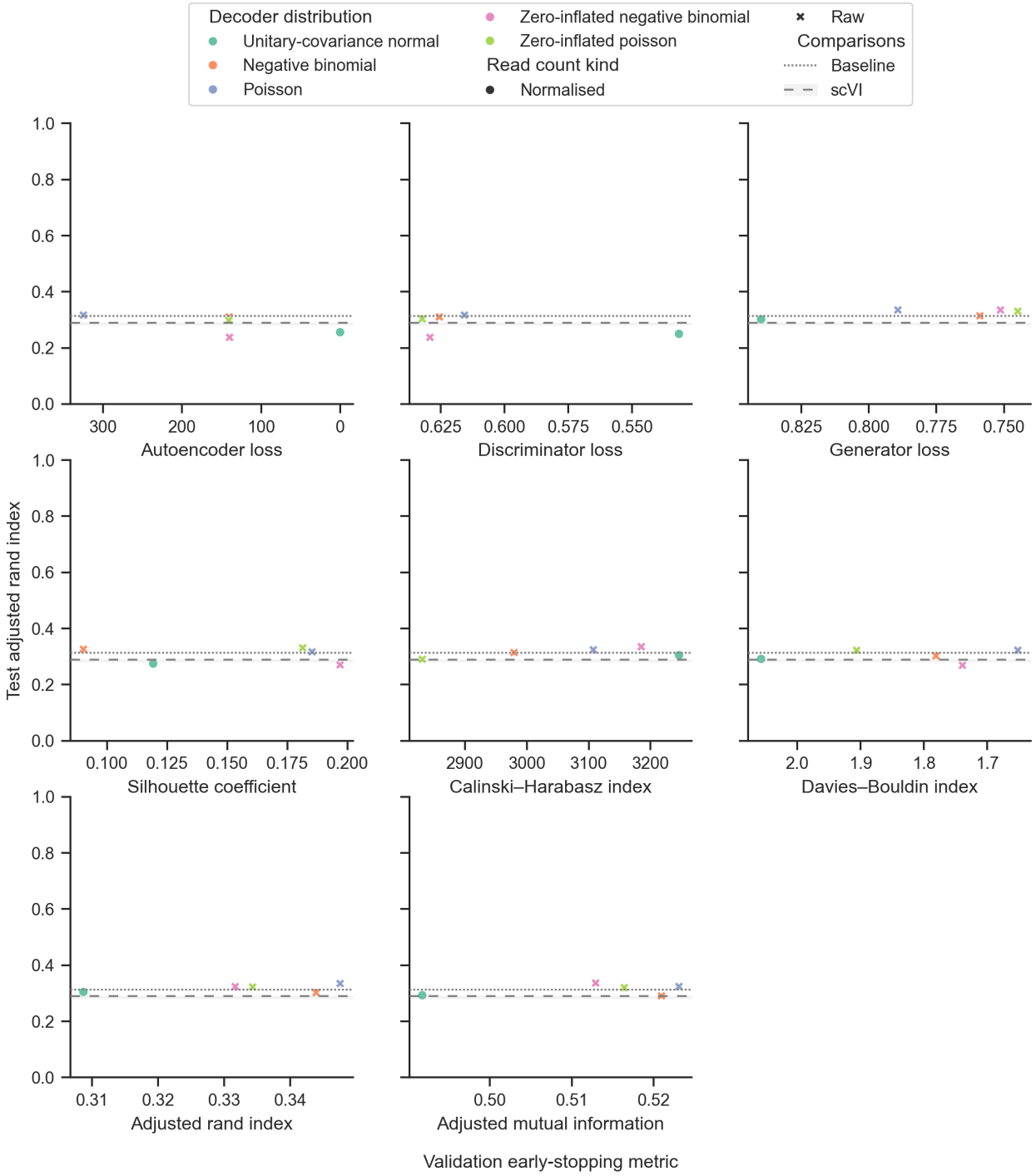


Fig. S26. Decoder-distribution search for the PBMC-68k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.

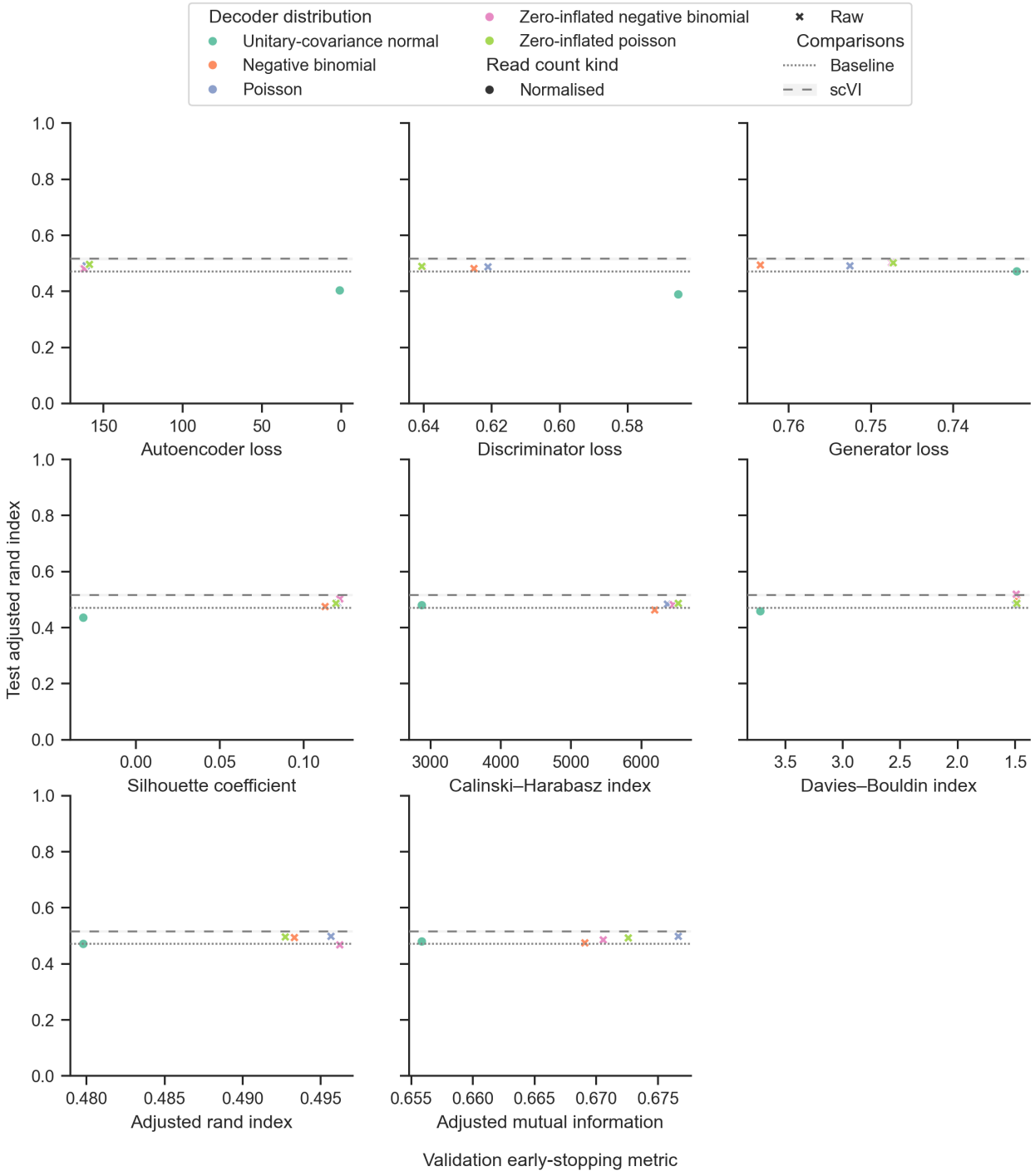
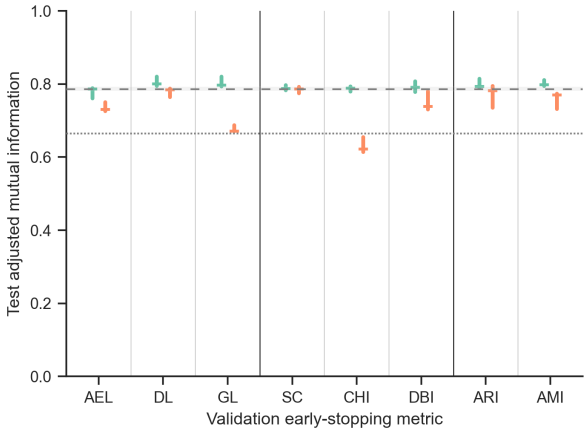
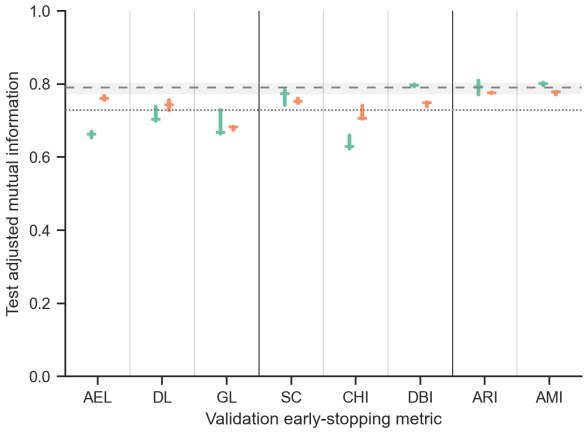


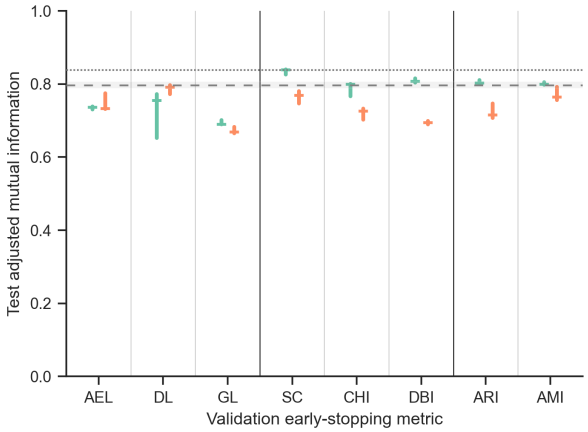
Fig. S27. Decoder-distribution search for the PBMC-92k data set using different metrics for early stopping. For each early-stopping metric, the test adjusted rand index is plotted against that particular validation metric for the median-performing run for all models included in the search. All validation metric axes are oriented such that optimal values are towards the right.



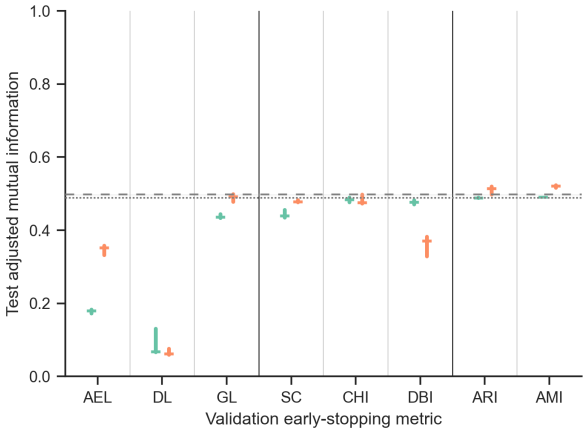
(a) PBMC-3k.



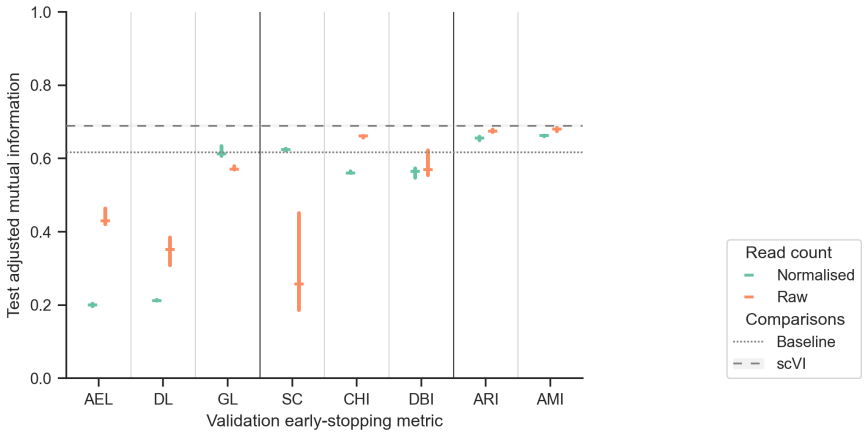
(b) PBMC-8k.



(c) PBMC-10k.



(d) PBMC-68k.



(e) PBMC-92k.

Fig. S28. The test adjusted mutual information for the optimal scAAE model selected by each early-stopping metric for all data sets. The performance of the baseline method and scVI is shown. For the scAAE models and scVI, the median and the interquartile range for three replicates is shown.

References

B. Alberts, R. Heald, A. Johnson, D. Morgan, M. Raff, K. Roberts, and P. Walter. *Molecular biology of the cell*. W. W. Norton & Company, New York, NY, seventh edition, July 2022. ISBN 978-0-393-88482-1.

S. Aldridge and S. A. Teichmann. Single cell transcriptomics comes of age. *Nat. Commun.*, 11(1):4307, Aug. 2020. DOI:10.1038/s41467-020-18158-5.

R. L. Allesøe, A. T. Lundgaard, R. Hernández Medina, A. Aguayo-Orozco, J. Johansen, J. N. Nissen, C. Brorsson, G. Mazzoni, L. Niu, J. H. Biel, C. Leal Rodríguez, V. Brasas, H. Webel, M. E. Benros, A. G. Pedersen, P. J. Chmura, U. P. Jacobsen, A. Mari, R. Koivula, A. Mahajan, A. Vinuela, J. F. Tajes, S. Sharma, M. Haid, M.-G. Hong, P. B. Musholt, F. De Masi, J. Vogt, H. K. Pedersen, V. Gudmundsdottir, A. Jones, G. Kennedy, J. Bell, E. L. Thomas, G. Frost, H. Thomsen, E. Hansen, T. H. Hansen, H. Vestergaard, M. Muilwijk, M. T. Blom, L. M. 't Hart, F. Pattou, V. Raverdy, S. Brage, T. Kokkola, A. Heggie, D. McEvoy, M. Mourby, J. Kaye, A. Hattersley, T. McDonald, M. Ridderstråle, M. Walker, I. Forgie, G. N. Giordano, I. Pavo, H. Ruetten, O. Pedersen, T. Hansen, E. Dermitzakis, P. W. Franks, J. M. Schwenk, J. Adamski, M. I. McCarthy, E. Pearson, K. Banasik, S. Rasmussen, S. Brunak, and IMI DIRECT Consortium. Discovery of drug-omics associations in type 2 diabetes with generative deep-learning models. *Nat. Biotechnol.*, 41(3):399–408, Mar. 2023. DOI:10.1038/s41587-022-01520-x.

P. Amaral, S. Carbonell-Sala, F. M. De La Vega, T. Faial, A. Frankish, T. Gingeras, R. Guigo, J. L. Harrow, A. G. Hatzigeorgiou, R. Johnson, T. D. Murphy, M. Pertea, K. D. Pruitt, S. Pujar, H. Takahashi, I. Ulitsky, A. Varabyou, C. A. Wells, M. Yandell, P. Carninci, and S. L. Salzberg. The status of the human gene catalogue. *Nature*, 622(7981):41–47, Oct. 2023. DOI:10.1038/s41586-023-06490-x.

M. Amodio, D. van Dijk, K. Srinivasan, W. S. Chen, H. Mohsen, K. R. Moon, A. Campbell, Y. Zhao, X. Wang, M. Venkataswamy, A. Desai, V. Ravi, P. Kumar, R. Montgomery, G. Wolf, and S. Krishnaswamy. Exploring single-cell data with deep multitasking neural networks. *Nature Methods*, 16(11):1139—-1145, Oct. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0576-7.

P. Angerer, L. Simon, S. Tritschler, F. A. Wolf, D. Fischer, and F. J. Theis. Single cells make big data: New challenges and opportunities in transcriptomics. *Curr. Opin. Syst. Biol.*, 4:85–91, Aug. 2017. DOI:10.1016/j.coisb.2017.07.004.

J. L. Ballard, Z. Wang, W. Li, L. Shen, and Q. Long. Deep learning-based approaches for multi-omics data integration and analysis. *BioData Min.*, 17(1):38, Oct. 2024. DOI:10.1186/s13040-024-00391-z.

E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.*, 37(1):38–44, Dec. 2018. DOI:10.1038/nbt.4314.

B. Beckert and B. Masquida. Synthesis of RNA by in vitro transcription. In *RNA*, Methods in molecular biology (Clifton, N.J.), pages 29–41. Humana Press, Totowa, NJ, 2011. DOI:10.1007/978-1-59745-248-9_3.

N. J. Bernstein, N. L. Fong, I. Lam, M. A. Roy, D. G. Hendrickson, and D. R. Kelley. Solo: Doublet identification in single-cell RNA-seq via semi-supervised deep learning. *Cell Syst.*, 11(1):95–101.e5, July 2020. DOI:10.1016/j.cels.2020.05.010.

I. Bica, H. Andrés-Terré, A. Cvejic, and P. Liò. Unsupervised generative and graph representation learning for modelling cell differentiation. *Sci. Rep.*, 10(1):9790, June 2020. DOI:10.1038/s41598-020-66166-8.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, Aug. 2006. ISBN 978-0387-31073-2.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008, Oct. 2008. DOI:10.1088/1742-5468/2008/10/P10008.

S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In S. Riezler and Y. Goldberg, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. DOI:10.18653/v1/K16-1002.

P. Boyeau, J. Hong, A. Gayoso, M. Kim, J. L. McFaline-Figueroa, M. I. Jordan, E. Azizi, C. Ergen, and N. Yosef. Deep generative modeling of sample-level heterogeneity in single-cell genomics. *bioRxiv*, May 2024. DOI:10.1101/2022.10.04.510898.

M. Brendel, C. Su, Z. Bai, H. Zhang, O. Elemento, and F. Wang. Application of deep learning on single-cell RNA sequencing data analysis: A review. *Genomics Proteomics Bioinformatics*, 20(5):814–835, Oct. 2022. DOI:10.1016/j.gpb.2022.11.011.

E. Brombacher, M. Hackenberg, C. Kreutz, H. Binder, and M. Treppner. The performance of deep generative models for learning joint embeddings of single-cell multi-omics data. *Front. Mol. Biosci.*, 9:962644, Oct. 2022. DOI:10.3389/fmolb.2022.962644.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. 2020. arXiv:2005.14165.

J. D. Buenrostro, B. Wu, U. M. Litzenburger, D. Ruff, M. L. Gonzales, M. P. Snyder, H. Y. Chang, and W. J. Greenleaf. Single-cell chromatin accessibility reveals principles of regulatory variation. *Nature*, 523 (7561):486–490, July 2015. DOI:10.1038/nature14590.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. 2015. arXiv:1509.00519.

A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat. Biotechnol.*, 36(5):411–420, May 2018. DOI:10.1038/nbt.4096.

M. Büttner, Z. Miao, F. A. Wolf, S. A. Teichmann, and F. J. Theis. A test metric for assessing single-cell RNA-seq batch correction. *Nat. Methods*, 16(1):43–49, Jan. 2019. DOI:10.1038/s41592-018-0254-1.

Y. Cao, X. Zhao, S. Tang, Q. Jiang, S. Li, S. Li, and S. Chen. scButterfly: a versatile single-cell cross-modality translation method via dual-aligned variational autoencoders. *Nat. Commun.*, 15(1):2973, Apr. 2024. DOI:10.1038/s41467-024-47418-x.

Z.-J. Cao, L. Wei, S. Lu, D.-C. Yang, and G. Gao. Searching large-scale scRNA-seq databases via unbiased cell embedding with cell BLAST. *Nat. Commun.*, 11(1):3458, July 2020. DOI:10.1038/s41467-020-17281-7.

C. Carlberg and F. Molnar. *Mechanisms of Gene Regulation*. Springer, Dordrecht, Netherlands, second edition, June 2016. ISBN 978-94-017-7740-7.

O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant. Variational autoencoders for new physics mining at the large hadron collider. *J. High Energy Phys.*, 2019(5), May 2019. DOI:10.1007/JHEP05(2019)036.

W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han. Domain-specific batch normalization for unsupervised domain adaptation. 2019. DOI:10.48550/arXiv.1906.03950, 1906.03950.

Y. Chen, H. Bian, L. Wei, J. Jia, X. Dong, Y. Li, Y. Zhao, X. Wu, C. Li, E. Luo, C. Xiao, M. Hao, and X. Zhang. Toward mastering the cell language by learning to generate. *bioRxiv*, Jan. 2024. DOI:10.1101/2024.01.25.577152.

Y. Cheng, S.-M. Xu, K. Santucci, G. Lindner, and M. Janitz. Machine learning and related approaches in transcriptomics. *Biochem. Biophys. Res. Commun.*, 724(150225):150225, Sept. 2024. DOI:10.1016/j.bbrc.2024.150225.

R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021. arXiv:2011.10650. https://openreview.net/forum?id=RLRXCV6DbEJ.

Y. Choi, R. Li, and G. Quon. siVAE: interpretable deep generative models for single-cell transcriptomes. *Genome Biol.*, 24(1):29, Feb. 2023. DOI:10.1186/s13059-023-02850-y.

A. Creswell and A. A. Bharath. Denoising adversarial autoencoders. *IEEE Trans. Neural Netw. Learn. Syst.*, 30(4):968–984, Apr. 2019. DOI:10.1109/TNNLS.2018.2852738.

H. Cui, C. Wang, H. Maan, K. Pang, F. Luo, N. Duan, and B. Wang. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. *Nat. Methods*, 21(8):1470–1480, Aug. 2024.

D. A. Cusanovich, R. Daza, A. Adey, H. A. Pliner, L. Christiansen, K. L. Gunderson, F. J. Steemers, C. Trapnell, and J. Shendure. Multiplex single cell profiling of chromatin accessibility by combinatorial cellular indexing. *Science*, 348(6237):910–914, May 2015. DOI:10.1126/science.aab1601.

B. Dai and D. Wipf. Diagnosing and enhancing VAE models. 2019. arXiv:1903.05789.

Y. Deng, F. Bao, Q. Dai, L. F. Wu, and S. J. Altschuler. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nature Methods*, 16(4):311–314, Mar. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0353-7.

A. B. Dieng, F. J. R. Ruiz, and D. M. Blei. Topic modeling in embedding spaces. *Trans. Assoc. Comput. Linguist.*, 8:439–453, Dec. 2020. DOI:10.1162/tacl_a_00325.

N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. 2016. arXiv:1611.02648.

J. Ding and A. Regev. Deep generative model embedding of single-cell RNA-Seq profiles on hyperspheres and hyperbolic spaces. *Nat. Commun.*, 12(1):2554, May 2021. DOI:10.1038/s41467-021-22851-4.

J. Ding, A. Condon, and S. P. Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.*, 9(1):2002, 2018. ISSN 2041-1723. DOI:10.1038/s41467-018-04368-5. https://doi.org/10.1038/s41467-018-04368-5.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017. arXiv:1605.09782.

F. Drost, Y. An, I. Bonafonte-Pardàs, L. M. Dratva, R. G. H. Lindeboom, M. Haniffa, S. A. Teichmann, F. Theis, M. Lotfollahi, and B. Schubert. Multi-modal generative modeling for joint analysis of single-cell T cell receptor and gene expression data. *Nat. Commun.*, 15(1):5577, July 2024. DOI:10.1038/s41467-024-49806-9.

V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. 2016. arXiv:1606.00704.

G. Eraslan, L. M. Simon, M. Mircea, N. S. Mueller, and F. J. Theis. Single cell RNA-seq denoising using a deep count autoencoder. *bioRxiv*, Apr. 2018. DOI:10.1101/300681.

Q. Fang, D. Su, W. Ng, and J. Feng. An effective biclustering-based framework for identifying cell subpopulations from scrna-seq data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2249–2260, 2021. DOI:10.1109/TCBB.2020.2979717.

Z. Fang, T. Liu, R. Zheng, J. A, M. Yin, and M. Li. stAA: adversarial graph autoencoder for spatial clustering task of spatially resolved transcriptomics. *Brief. Bioinform.*, 25(1), Nov. 2023. DOI:10.1093/bib/bbad500.

E. Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

A. Foster, A. Vezer, C. A. Glastonbury, P. Creed, S. Abujudeh, and A. Sim. Contrastive mixture of posteriors for counterfactual inference, data integration and fairness. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6578–6621. PMLR, 17–23 Jul 2022. https://proceedings.mlr.press/v162/foster22a.html.

J. Gagnon, L. Pi, M. Ryals, Q. Wan, W. Hu, Z. Ouyang, B. Zhang, and K. Li. Recommendations of scrna-seq differential gene expression analysis based on comprehensive benchmarking. *Life*, 12(6):850, June 2022. ISSN 2075-1729. DOI:10.3390/life12060850.

A. Ganguly, S. Jain, and U. Watchareeruetai. Amortized variational inference: A systematic review. *J. Artif. Intell. Res.*, 78:167–215, Oct. 2023. DOI:10.1613/jair.1.14258.

P.-L. Germain, A. Lun, C. Garcia Meixide, W. Macnair, and M. D. Robinson. Doublet identification in single-cell sequencing data using scDblFinder. *F1000Res.*, 10:979, Sept. 2021. DOI:10.12688/f1000research.73600.2.

T. M. Gierahn, M. H. Wadsworth, 2nd, T. K. Hughes, B. D. Bryson, A. Butler, R. Satija, S. Fortune, J. C. Love, and A. K. Shalek. Seq-Well: portable, low-cost RNA sequencing of single cells at high throughput. *Nat. Methods*, 14(4):395–398, Apr. 2017. DOI:10.1038/nmeth.4179.

M. P. Gold, A. LeNail, and E. Fraenkel. *Shallow Sparsely-Connected Autoencoders for Gene Set Projection*, pages 374–385. 2019. DOI:10.1142/9789813279827_0034.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014. https://papers.neurips.cc/paper/5423-generative-adversarial-nets.

J. M. Graving and I. D. Couzin. VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering. *bioRxiv*, July 2020. DOI:10.1101/2020.07.17.207993.

A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'06, page 513–520, Cambridge, MA, USA, 2006. MIT Press. arXiv:0805.2368.

C. H. Grønbech, R. DeVries, E. Roellin, and O. Winther. scVAE-C: Latent-representation covariate removal and counterfactual expression profiles for single cells using conditional and hierarchical variational autoencoders. Appendix A.1, a.

C. H. Grønbech, E. Molinaro, K. Natarajan, and O. Winther. scAAE: Adversarial autoencoders for single-cell gene expression data. Appendix A.2, b.

C. H. Grønbech, M. F. Vording, P. N. Timshel, C. K. Sønderby, T. H. Pers, and O. Winther. scVAE: Variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, 05 2020. ISSN 1367-4803. DOI:10.1093/bioinformatics/btaa293.

M. K. Gunady, J. Kancherla, H. C. Bravo, and S. Feizi. ScGAIN: Single cell RNA-seq data imputation using generative adversarial networks. *bioRxiv*, Nov. 2019. DOI:10.1101/837302.

X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1753–1759. AAAI Press, 2017. ISBN 9780999241103. https://www.ijcai.org/Proceedings/2017/243.

G. Gut, S. G. Stark, G. Rätsch, and N. R. Davidson. pmVAE: Learning interpretable single-cell representations with pathway modules. *bioRxiv*, Jan. 2021. DOI:10.1101/2021.01.28.428664.

R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.*, 4(2):268–276, Feb. 2018. DOI:10.1021/acscentsci.7b00572.

L. Haghverdi, A. T. L. Lun, M. D. Morgan, and J. C. Marioni. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.*, 36(5):421–427, June 2018. DOI:10.1038/nbt.4091.

M. Hao, J. Gong, X. Zeng, C. Liu, Y. Guo, X. Cheng, T. Wang, J. Ma, X. Zhang, and L. Song. Large-scale foundation model on single-cell transcriptomics. *Nat. Methods*, 21(8):1481–1491, Aug. 2024. DOI:10.1038/s41592-024-02305-7.

W. Harvey, S. Naderiparizi, and F. Wood. Conditional image generation by conditioning variational auto-encoders. 2021. arXiv:2102.12037.

T. Hashimshony, F. Wagner, N. Sher, and I. Yanai. CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification. *Cell Rep.*, 2(3):666–673, Sept. 2012. DOI:10.1016/j.celrep.2012.08.003.

Y. Hasin, M. Seldin, and A. Lusis. Multi-omics approaches to disease. *Genome Biol.*, 18(1), Dec. 2017. DOI:10.1186/s13059-017-1215-1.

L. Heumos, A. C. Schaar, C. Lance, A. Litinetskaya, F. Drost, L. Zappia, M. D. Lücken, D. C. Strobl, J. Henao, F. Curion, Single-cell Best Practices Consortium, H. B. Schiller, and F. J. Theis. Best practices for single-cell analysis across modalities. *Nat. Rev. Genet.*, 24(8):550–572, Aug. 2023. DOI:10.1038/s41576-023-00586-w.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. https://openreview.net/forum?id=Sy2fzU9gl.

G. E. Hinton and R. Zemel. Autoencoders, minimum description length and helmholtz free energy. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. https://proceedings.neurips.cc/paper_files/paper/1993/file/9e3cfc48eccf81a0d57663e129aef3cb-Paper.pdf.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. arXiv:2006.11239.

S. Hochreiter, D.-A. Clevert, and K. Obermayer. A new summarization method for affymetrix probe level data. *Bioinformatics*, 22(8):943–949, Apr. 2006. DOI:10.1093/bioinformatics/btl033.

K. Hrovatin, L. Sikkema, V. A. Shitov, G. Heimberg, M. Shulman, A. J. Oliver, M. F. Mueller, I. L. Ibarra, H. Wang, C. Ramírez-Suástegui, P. He, A. C. Schaar, S. A. Teichmann, F. J. Theis, and M. D. Luecken. Considerations for building and using integrated single-cell atlases. *Nat. Methods*, 22(1):41–57, Jan. 2025.

F. Huszár. Is maximum likelihood useful for representation learning?, May 2017. https://www.inference.vc/maximum-likelihood-for-representation-learning-2/. Accessed: 2025-01-22.

K. Jang, S. Hong, M. Kim, J. Na, and I. Moon. Adversarial autoencoder based feature learning for fault detection in industrial processes. *IEEE Trans. Industr. Inform.*, 18(2):827–834, Feb. 2022. DOI:10.1109/TII.2021.3078414.

J. Jiang, J. Xu, Y. Liu, B. Song, X. Guo, X. Zeng, and Q. Zou. Dimensionality reduction and visualization of single-cell RNA-seq data with an improved deep variational autoencoder. *Brief. Bioinform.*, 24(3), May 2023. DOI:10.1093/bib/bbad152.

Q. Jin, Y. Ma, F. Fan, J. Huang, X. Mei, and J. Ma. Adversarial autoencoder network for hyperspectral unmixing. *IEEE Trans. Neural Netw. Learn. Syst.*, 34(8):4555–4569, Aug. 2023. DOI:10.1109/TNNLS.2021.3114203.

A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, and A. Zhavoronkov. drugan: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular Pharmaceutics*, 14(9):3098–3104, 2017. DOI:10.1021/acs.molpharmaceut.7b00346. PMID: 28703000.

N. C. Kalafut, X. Huang, and D. Wang. Joint variational autoencoders for multimodal imputation and embedding. *Nat. Mach. Intell.*, 5(6):631–642, June 2023. DOI:10.1038/s42256-023-00663-z.

H. M. Kang, M. Subramaniam, S. Targ, M. Nguyen, L. Maliskova, E. McCarthy, E. Wan, S. Wong, L. Byrnes, C. M. Lanata, R. E. Gate, S. Mostafavi, A. Marson, N. Zaitlen, L. A. Criswell, and C. J. Ye. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology*, 36(1):89–94, Dec. 2017. DOI:10.1038/nbt.4042.

H. M. Kang, M. Subramaniam, S. Targ, M. Nguyen, L. Maliskova, E. McCarthy, E. Wan, S. Wong, L. Byrnes, C. M. Lanata, R. E. Gate, S. Mostafavi, A. Marson, N. Zaitlen, L. A. Criswell, and C. J. Ye. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nat. Biotechnol.*, 36(1):89–94, Jan. 2018. DOI:10.1038/nbt.4042.

S. A. Khan, R. Lehmann, X. Martinez-de Morentin, A. Maillo, V. Lagani, N. A. Kiani, D. Gomez-Cabrero, and J. Tegner. scAEGAN: Unification of single-cell genomics data by adversarial learning of latent space correspondences. *PLoS One*, 18(2):e0281315, Feb. 2023. DOI:10.1371/journal.pone.0281315.

T. Kim, I. R. Chen, Y. Lin, A. Y.-Y. Wang, J. Y. H. Yang, and P. Yang. Impact of similarity metrics on single-cell RNA-seq data clustering. *Brief. Bioinform.*, 20(6):2316–2326, Nov. 2019. DOI:10.1093/bib/bby076.

J. C. Kimmel. Disentangling latent representations of single cell RNA-seq experiments. *bioRxiv*, Mar. 2020. DOI:10.1101/2020.03.04.972166.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, 2014. arXiv:1312.6114. https://iclr.cc/archive/2014/conference-proceedings/.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4743–4751, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819. arXiv:1606.04934.

A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161 (5):1187–1201, May 2015. DOI:10.1016/j.cell.2015.04.044.

K. D. Ko and V. Sartorelli. A deep learning adversarial autoencoder with dynamic batching displays high performance in denoising and ordering scRNA-seq data. *iScience*, 27(3):109027, Mar. 2024. DOI:10.1016/j.isci.2024.109027.

A. Kopf, V. Fortuin, V. R. Somnath, and M. Claassen. Mixture-of-experts variational autoencoder for clustering and generating from similarity-based representations on single cell data. *PLoS Comput. Biol.*, 17(6):e1009086, June 2021. DOI:10.1371/journal.pcbi.1009086.

I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-R. Loh, and S. Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nat. Methods*, 16(12):1289–1296, Dec. 2019.

M. Kubista, J. M. Andrade, M. Bengtsson, A. Forootan, J. Jonák, K. Lind, R. Sindelka, R. Sjöback, B. Sjögreen, L. Strömbom, A. Ståhlberg, and N. Zoric. The real-time polymerase chain reaction. *Mol. Aspects Med.*, 27(2-3):95–125, Apr. 2006. DOI:10.1016/j.mam.2005.12.007.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22(1):79–86, Mar. 1951. DOI:10.1214/aoms/1177729694.

A. Lafzi, C. Moutinho, S. Picelli, and H. Heyn. Tutorial: guidelines for the experimental design of single-cell RNA sequencing studies. *Nat. Protoc.*, 13(12):2742–2757, Dec. 2018. DOI:10.1038/s41596-018-0073-y.

A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. 2015, 1512.09300.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. DOI:10.1038/nature14539.

D. Lee, H. Yu, X. Jiang, D. Rogith, M. Gudala, M. Tejani, Q. Zhang, and L. Xiong. Generating sequential electronic health records using dual adversarial autoencoder. *J. Am. Med. Inform. Assoc.*, 27(9):1411–1419, July 2020. DOI:10.1093/jamia/ocaa119.

J. M. Levsky, S. M. Shenoy, R. C. Pezo, and R. H. Singer. Single-cell gene expression profiling. *Science*, 297 (5582):836–840, Aug. 2002. DOI:10.1126/science.1072241.

E. Lin, S. Mukherjee, and S. Kannan. A deep adversarial variational autoencoder model for dimensionality reduction in single-cell RNA sequencing analysis. *BMC Bioinformatics*, 21(1):64, Feb. 2020. DOI:10.1186/s12859-020-3401-5.

S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–137, Mar. 1982. DOI:10.1109/TIT.1982.1056489.

R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, 15(12):1053–1058, 2018. ISSN 1548-7105. DOI:10.1038/s41592-018-0229-2.

M. Lotfollahi, F. A. Wolf, and F. J. Theis. scgen predicts single-cell perturbation responses. *Nat. Methods*, 16(8):715–721, Aug. 2019. DOI:10.1038/s41592-019-0494-8.

M. Lotfollahi, M. Naghipourfar, F. J. Theis, and F. A. Wolf. Conditional out-of-distribution generation for unpaired data using transfer VAE. *Bioinformatics*, 36(Suppl_2):i610–i617, Dec. 2020. DOI:10.1093/bioinformatics/btaa800.

M. Lotfollahi, A. K. Susmelj, C. De Donno, Y. Ji, I. L. Ibarra, F. A. Wolf, N. Yakubova, F. J. Theis, and D. Lopez-Paz. Learning interpretable cellular responses to complex perturbations in high-throughput screens. *bioRxiv*, Apr. 2021. DOI:10.1101/2021.04.14.439903.

C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel. The variational fair autoencoder. 2015. arXiv:1511.00830.

R. Lowe, N. Shirley, M. Bleackley, S. Dolan, and T. Shafee. Transcriptomics technologies. *PLoS Comput. Biol.*, 13(5):e1005457, May 2017. DOI:10.1371/journal.pcbi.1005457.

E. Luo, M. Hao, L. Wei, and X. Zhang. scdiffusion: conditional generation of high-quality single-cell data using diffusion model. *Bioinformatics*, 40(9), Sept. 2024. DOI:10.1093/bioinformatics/btae518.

M. D. Lücken and F. J. Theis. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.*, 15(6):e8746, June 2019. DOI:10.15252/msb.20188746.

L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. *BIVA: a very deep hierarchy of latent variables for generative modeling*. Curran Associates Inc., Red Hook, NY, USA, 2019. arXiv:1902.02102.

E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, J. J. Trombetta, D. A. Weitz, J. R. Sanes, A. K. Shalek, A. Regev, and S. A. McCarroll. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161 (5):1202–1214, May 2015. DOI:10.1016/j.cell.2015.05.002.

A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016. arXiv:1511.05644.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv e-prints*, 2020. arXiv:1802.03426.

L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? *arXiv e-prints*, Jan. 2018. arXiv:1801.04406.

K. Minoura, K. Abe, H. Nam, H. Nishikawa, and T. Shimamura. A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data. *Cell Rep. Methods*, 1(5):100071, Sept. 2021. DOI:10.1016/j.crmeth.2021.100071.

S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. 2016. arXiv:1610.03483.

L. Monnier and P.-H. Cournède. A novel batch-effect correction method for scRNA-seq data based on adversarial information factorization. *PLoS Comput. Biol.*, 20(2):e1011880, Feb. 2024. DOI:10.1371/journal.pcbi.1011880.

M. Moor, M. Horn, B. Rieck, and K. Borgwardt. Topological autoencoders. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7045–7054. PMLR, 2020. arXiv:1906.00722. http://proceedings.mlr.press/v119/moor20a.html.

P. Murchan, C. Ó'Brien, S. O'Connell, C. S. McNevin, A.-M. Baird, O. Sheils, P. Ó Broin, and S. P. Finn. Deep learning of histopathological features for the prediction of tumour molecular genetics. *Diagnostics (Basel)*, 11(8):1406, Aug. 2021. DOI:10.3390/diagnostics11081406.

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. http://probml.github.io/book1.

S. Nachtergaele and C. He. The emerging biology of RNA post-transcriptional modifications. *RNA Biol.*, 14(2):156–163, Feb. 2017. DOI:10.1080/15476286.2016.1267096.

N. J. Nelson. Microarrays have arrived: gene expression tool matures. *J. Natl. Cancer Inst.*, 93(7):492–494, Apr. 2001. DOI:10.1093/jnci/93.7.492.

F. Nielsen. *Introduction to HPC with MPI for data science*. Undergraduate Topics in Computer Science. Springer International Publishing, Basel, Switzerland, 1 edition, Feb. 2016. ISBN 978-3-319-21903-5.

J. N. Nissen, J. Johansen, R. L. Allesøe, C. K. Sønderby, J. J. A. Armenteros, C. H. Grønbech, L. J. Jensen, H. B. Nielsen, T. N. Petersen, O. Winther, and S. Rasmussen. Improved metagenome binning and assembly using deep variational autoencoders. *Nat. Biotechnol.*, 39(5):555–560, May 2021. DOI:10.1038/s41587-020-00777-4.

T. Oibayashi, T. Ueda, Y. Kimura, Y. Tohsato, and I. Nishikawa. Phenotype anomaly detection in early c. elegans embryos by variational auto-encoder. In *2021 IEEE 9th International Conference on Bioinformatics and Computational Biology (ICBCB)*. IEEE, May 2021. DOI:10.1109/ICBCB52223.2021.9459228.

K. Pearson. LIII. on lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.*, 2(11):559–572, Nov. 1901. DOI:10.1080/14786440109462720.

V. M. Peterson, K. X. Zhang, N. Kumar, J. Wong, L. Li, D. C. Wilson, R. Moore, T. K. McClanahan, S. Sadekova, and J. A. Klappenbach. Multiplexed quantification of proteins and transcripts in single cells. *Nat. Biotechnol.*, 35(10):936–939, Oct. 2017.

E. Pierson and C. Yau. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol.*, 16(1):241, Nov. 2015. DOI:10.1186/s13059-015-0805-z.

A. Plumerault, H. Le Borgne, and C. Hudelot. AVAE: Adversarial variational auto encoder. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI:10.1109/ICPR48806.2021.9412727.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

V. Raghavan, L. Kraft, F. Mesny, and L. Rigerte. A simple guide to de novo transcriptome assembly and annotation. *Brief. Bioinform.*, 23(2), Mar. 2022. DOI:10.1093/bib/bbab563.

R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA, June 2009. ACM. DOI:10.1145/1553374.1553486.

D. Ramsköld, S. Luo, Y.-C. Wang, R. Li, Q. Deng, O. R. Faridani, G. A. Daniels, I. Khrebtukova, J. F. Loring, L. C. Laurent, G. P. Schroth, and R. Sandberg. Full-length mRNA-Seq from single-cell levels of RNA and individual circulating tumor cells. *Nat. Biotechnol.*, 30(8):777–782, Aug. 2012. DOI:10.1038/nbt.2282.

RAPIDS Development Team. *RAPIDS: Libraries for End to End GPU Data Science*, 2023. https://rapids.ai.

D. J. Rezende and F. Viola. Taming VAEs. 2018. arXiv:1810.00597.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286. PMLR, 22–24 Jun 2014. arXiv:1401.4082. http://proceedings.mlr.press/v32/rezende14.html.

D. Risso, J. Ngai, T. P. Speed, and S. Dudoit. Normalization of RNA-seq data using factor analysis of control genes or samples. *Nat. Biotechnol.*, 32(9):896–902, Sept. 2014. DOI:10.1038/nbt.2931.

P. Rubenstein. Variational autoencoders are not autoencoders, Jan. 2019. http://paulrubenstein.co.uk/variational-autoencoders-are-not-autoencoders/. Accessed: 2025-01-22.

R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009. ISSN 0888-613X. DOI:10.1016/j.ijar.2008.11.006.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. 2016. arXiv:1606.03498.

R. Sandberg. Entering the era of single-cell transcriptomics in biology and medicine. *Nat. Methods*, 11(1):22–24, Jan. 2014. DOI:10.1038/nmeth.2764.

R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev. Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.*, 33(5):495–502, May 2015. DOI:10.1038/nbt.3192.

L. Seninge, I. Anastopoulos, H. Ding, and J. Stuart. VEGA is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. *Nat. Commun.*, 12(1):5684, Sept. 2021. DOI:10.1038/s41467-021-26017-0.

Z. Shao, R. Zhao, S. Yuan, M. Ding, and Y. Wang. Tracing the evolution of AI in the past decade and forecasting the emerging trends. *Expert Syst. Appl.*, 209(118221):118221, Dec. 2022. DOI:10.1016/j.eswa.2022.118221.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. 2017. arXiv:1701.06538.

H. Shen, J. Liu, J. Hu, X. Shen, C. Zhang, D. Wu, M. Feng, M. Yang, Y. Li, Y. Yang, W. Wang, Q. Zhang, J. Yang, K. Chen, and X. Li. Generative pretraining from large-scale transcriptomes for single-cell deciphering. *iScience*, 26(5):106536, May 2023. DOI:10.1016/j.isci.2023.106536.

C. Sheng, R. Lopes, G. Li, S. Schuierer, A. Waldt, R. Cuttat, S. Dimitrieva, A. Kauffmann, E. Durand, G. G. Galli, G. Roma, and A. de Weck. Probabilistic machine learning ensures accurate ambient denoising in droplet-based single-cell omics. *bioRxiv*, Jan. 2022. DOI:10.1101/2022.01.14.476312.

Y. Shi, N. Siddharth, B. Paige, and P. H. S. Torr. Variational mixture-of-experts autoencoders for multi-modal deep generative models. 2019. arXiv:1911.03393.

A. Shree, M. K. Pavan, and H. Zafar. scDREAMER for atlas-level integration of single-cell datasets using deep generative model paired with adversarial classifier. *Nat. Commun.*, 14(1):7781, Nov. 2023. DOI:10.1038/s41467-023-43590-8.

A. Singh, K. Biharie, M. J. T. Reinders, A. Mahfouz, and T. Abdelaal. scTopoGAN: unsupervised manifold alignment of single-cell data. *Bioinform. Adv.*, 3(1):vbad171, Nov. 2023. DOI:10.1093/bioadv/vbad171.

J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2256–2265. JMLR.org, 2015. arXiv:1503.03585.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

M. Stoeckius, C. Hafemeister, W. Stephenson, B. Houck-Loomis, P. K. Chattopadhyay, H. Swerdlow, R. Satija, and P. Smibert. Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods*, 14(9):865–868, Sept. 2017. DOI:10.1038/nmeth.4380.

T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, 3rd, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902.e21, June 2019. DOI:https://doi.org/10.1016/j.cell.2019.05.031.

I. Subramanian, S. Verma, S. Kumar, A. Jere, and K. Anamika. Multi-omics data integration, interpretation, and its application. *Bioinform. Biol. Insights*, 14:1177932219899051, Jan. 2020. DOI:10.1177/1177932219899051.

V. Svensson. Droplet scRNA-seq is not zero-inflated. *Nat. Biotechnol.*, 38(2):147–150, Feb. 2020. DOI:10.1038/s41587-019-0379-5.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3745–3753, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819. arXiv:1602.02282.

H. Thanh-Tung and T. Tran. Catastrophic forgetting and mode collapse in GANs. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2020. DOI:10.1109/IJCNN48605.2020.9207181.

L. L. Thurstone. *The Vectors of Mind*. The University of Chicago Press, Chicago, Illinois, 1935.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *J. R. Stat. Soc. Series B Stat. Methodol.*, 61(3):611–622, Sept. 1999. DOI:10.1111/1467-9868.00196.

J. Tomczak and M. Welling. Vae with a vampprior. In A. Storkey and F. Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 09–11 Apr 2018. https://proceedings.mlr.press/v84/tomczak18a.html.

J. M. Tomczak. *Deep Generative Modeling*. Springer International Publishing, Cham, Switzerland, second edition, 2024. ISBN 978-3-031-64086-5.

J. M. Tomczak and M. Welling. Improving variational Auto-Encoders using householder flow. 2016. arXiv:1611.09630.

V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Sci. Rep.*, 9(1):5233, Mar. 2019. DOI:10.1038/s41598-019-41695-z.

T. N. Trong, J. Mehtonen, G. González, R. Kramer, V. Hautamäki, and M. Heinäniemi. Semisupervised generative autoencoder for single-cell data. *J. Comput. Biol.*, 27(8):1190–1203, Aug. 2020. DOI:10.1089/cmb.2019.0337.

A. Vahdat and J. Kautz. Nvae: a deep hierarchical variational autoencoder. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. arXiv:2007.03898.

R. van den Berg, L. Hasenclever, J. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. arXiv:1803.05649.

L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 (86):2579–2605, 2008. http://jmlr.org/papers/v9/vandermaaten08a.html.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

X. Wan, J. Xiao, S. S. T. Tam, M. Cai, R. Sugimura, Y. Wang, X. Wan, Z. Lin, A. R. Wu, and C. Yang. Integrating spatial and single-cell transcriptomics data using deep generative models with SpatialScope. *Nat. Commun.*, 14(1):7848, Nov. 2023.

B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, and S. Batzoglou. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nat. Methods*, 14(4):414–416, Apr. 2017. DOI:10.1038/nmeth.4207.

D. Wang and J. Gu. VASC: Dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genomics, Proteomics & Bioinformatics*, 16(5):320–331, Oct. 2018. DOI:10.1016/j.gpb.2018.08.003. https://doi.org/10.1016/j.gpb.2018.08.003.

J. Wang, D. Agarwal, M. Huang, G. Hu, Z. Zhou, C. Ye, and N. R. Zhang. Data denoising with transfer learning in single-cell transcriptomics. *Nature Methods*, 16(9):875–878, Aug. 2019. ISSN 1548-7105. DOI:10.1038/s41592-019-0537-1.

X. Wang, C. Zhang, L. Wang, and P. Zheng. Integrating multiple single-cell RNA sequencing datasets using adversarial autoencoders. *Int. J. Mol. Sci.*, 24(6), Mar. 2023. DOI:10.3390/ijms24065502.

Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10(1):57–63, Jan. 2009. DOI:10.1038/nrg2484.

G. P. Way and C. S. Greene. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. In *Biocomputing 2018*, Singapore, Nov. 2017. World Scientific. DOI:10.1142/9789813235533_0008.

E. Weinberger, C. Lin, and S.-I. Lee. Isolating salient variations of interest in single-cell data with contrastiveVI. *Nat. Methods*, 20(9):1336–1345, Sept. 2023. DOI:10.1038/s41592-023-01955-3.

F. A. Wolf, P. Angerer, and F. J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, 19(1), Dec. 2018. DOI:10.1186/s13059-017-1382-0.

M. Wu and N. Goodman. Multimodal generative models for scalable weakly-supervised learning. 2018. arXiv:1802.05335.

Y. Wu, Y. Guo, Y. Xiao, and S. Lao. AAE-SC: A scRNA-seq clustering framework based on adversarial autoencoder. *IEEE Access*, 8:178962–178975, 2020. DOI:10.1109/ACCESS.2020.3027481.

Y. Wu, J. Liu, S. Liu, Y. Xiao, S. Zhang, and L. Li. CoupleVAE: coupled variational autoencoders for predicting perturbational single-cell RNA sequencing data. *bioRxiv*, Mar. 2024. DOI:10.1101/2024.03.05.583614.

R. Xiang, W. Wang, L. Yang, S. Wang, C. Xu, and X. Chen. A comparison for dimensionality reduction methods of single-cell RNA-seq data. *Front. Genet.*, 12:646936, Mar. 2021. DOI:10.3389/fgene.2021.646936.

L. Xiong, K. Tian, Y. Li, W. Ning, X. Gao, and Q. C. Zhang. Online single-cell data integration through projecting heterogeneous datasets into a common cell-embedding space. *Nat. Commun.*, 13(1):6118, Oct. 2022. DOI:10.1038/s41467-022-33758-z.

C. Xu, R. Lopez, E. Mehlman, J. Regier, M. I. Jordan, and N. Yosef. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. *Mol. Syst. Biol.*, 17(1):e9620, Jan. 2021. DOI:10.15252/msb.20209620.

H. Xu, S. Wang, M. Fang, S. Luo, C. Chen, S. Wan, R. Wang, M. Tang, T. Xue, B. Li, J. Lin, and K. Qu. SPACEL: deep learning-based characterization of spatial transcriptome architectures. *Nat. Commun.*, 14(1):7603, Nov. 2023. DOI:10.1038/s41467-023-43220-3.

Y. Xu, Z. Zhang, L. You, J. Liu, Z. Fan, and X. Zhou. scIGANs: single-cell RNA-seq imputation using generative adversarial networks. *Nucleic Acids Res.*, 48(15):e85, Sept. 2020. DOI:10.1093/nar/gkaa506.

Y. Xu, E. Begoli, and R. P. McCord. sciCAN: single-cell chromatin accessibility and gene expression data integration via cycle-consistent adversarial network. *NPJ Syst. Biol. Appl.*, 8(1):33, Sept. 2022. DOI:10.1038/s41540-022-00245-6.

F. Yang, W. Wang, F. Wang, Y. Fang, D. Tang, J. Huang, H. Lu, and J. Yao. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. *Nat. Mach. Intell.*, 4(10):852–866, Sept. 2022. DOI:10.1038/s42256-022-00534-z.

X. Yang, K. K. Mann, H. Wu, and J. Ding. sccross: a deep generative model for unifying single-cell multi-omics with seamless integration, cross-modal generation, and in silico exploration. *Genome Biol.*, 25(1):198, July 2024. DOI:10.1186/s13059-024-03338-z.

Y. Yazici, C.-S. Foo, S. Winkler, K.-H. Yap, and V. Chandrasekhar. Empirical analysis of overfitting and mode drop in gan training. In *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, Oct. 2020. DOI:10.1109/icip40778.2020.9191083.

H. Yu and J. D. Welch. MichiGAN: sampling from disentangled representations of single-cell data using generative adversarial networks. *Genome Biol.*, 22(1):158, May 2021. DOI:10.1186/s13059-021-02373-4.

L. Zappia, B. Phipson, and A. Oshlack. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. *PLoS Comput. Biol.*, 14(6):e1006245, June 2018. DOI:10.1371/journal.pcbi.1006245.

A. W. Zhang, C. O'Flanagan, E. A. Chavez, J. L. P. Lim, N. Ceglia, A. McPherson, M. Wiens, P. Walters, T. Chan, B. Hewitson, D. Lai, A. Mottok, C. Sarkozy, L. Chong, T. Aoki, X. Wang, A. P. Weng, J. N. McAlpine, S. Aparicio, C. Steidl, K. R. Campbell, and S. P. Shah. Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nat. Methods*, 16(10):1007–1015, Oct. 2019. DOI:10.1038/s41592-019-0529-1.

C. Zhang. Single-cell data analysis using MMD variational autoencoder for a more informative latent representation. *bioRxiv*, Apr. 2019. DOI:10.1101/613414.

Z. Zhang, X. Zhao, M. Bindra, P. Qiu, and X. Zhang. scDisInFact: disentangled learning for integration and prediction of multi-batch multi-condition single-cell RNA-sequencing data. *Nat. Commun.*, 15(1):912, Jan. 2024. DOI:10.1038/s41467-024-45227-w.

Y. Zhao, H. Cai, Z. Zhang, J. Tang, and Y. Li. Learning interpretable cellular and gene signature embeddings from single-cell transcriptomic data. *Nat. Commun.*, 12(1):5261, Sept. 2021. DOI:10.1038/s41467-021-25534-2.

G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.*, 8(1):14049, Jan. 2017. DOI:10.1038/ncomms14049.

Y. Zheng, J. C. Schupp, T. Adams, G. Clair, A. Justet, F. Ahangari, X. Yan, P. Hansen, M. Carlon, E. Cortesi, M. Vermant, R. Vos, L. J. De Sadeleer, I. O. Rosas, R. Pineda, J. Sembrat, M. Königshoff, J. E. McDonough, B. M. Vanaudenaerde, W. A. Wuyts, N. Kaminski, and J. Ding. UNAGI: Deep generative model for deciphering cellular dynamics and in-silico drug discovery in complex diseases. *Res. Sq.*, Dec. 2023. DOI:10.21203/rs.3.rs-3676579/v1.

X. Zhou, H. Chai, Y. Zeng, H. Zhao, and Y. Yang. scAdapt: virtual adversarial domain adaptation network for single cell RNA-seq data classification across platforms and species. *Brief. Bioinform.*, 22(6), Nov. 2021. DOI:10.1093/bib/bbab281.

H. Zhu and D. Slonim. From noise to knowledge: Diffusion probabilistic model-based neural inference of gene regulatory networks. *J. Comput. Biol.*, 31(11):1087–1103, Nov. 2024. DOI:10.1089/cmb.2024.0607.

C. Zuo and L. Chen. Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data. *Brief. Bioinform.*, 22(4), July 2021. DOI:10.1093/bib/bbaa287.