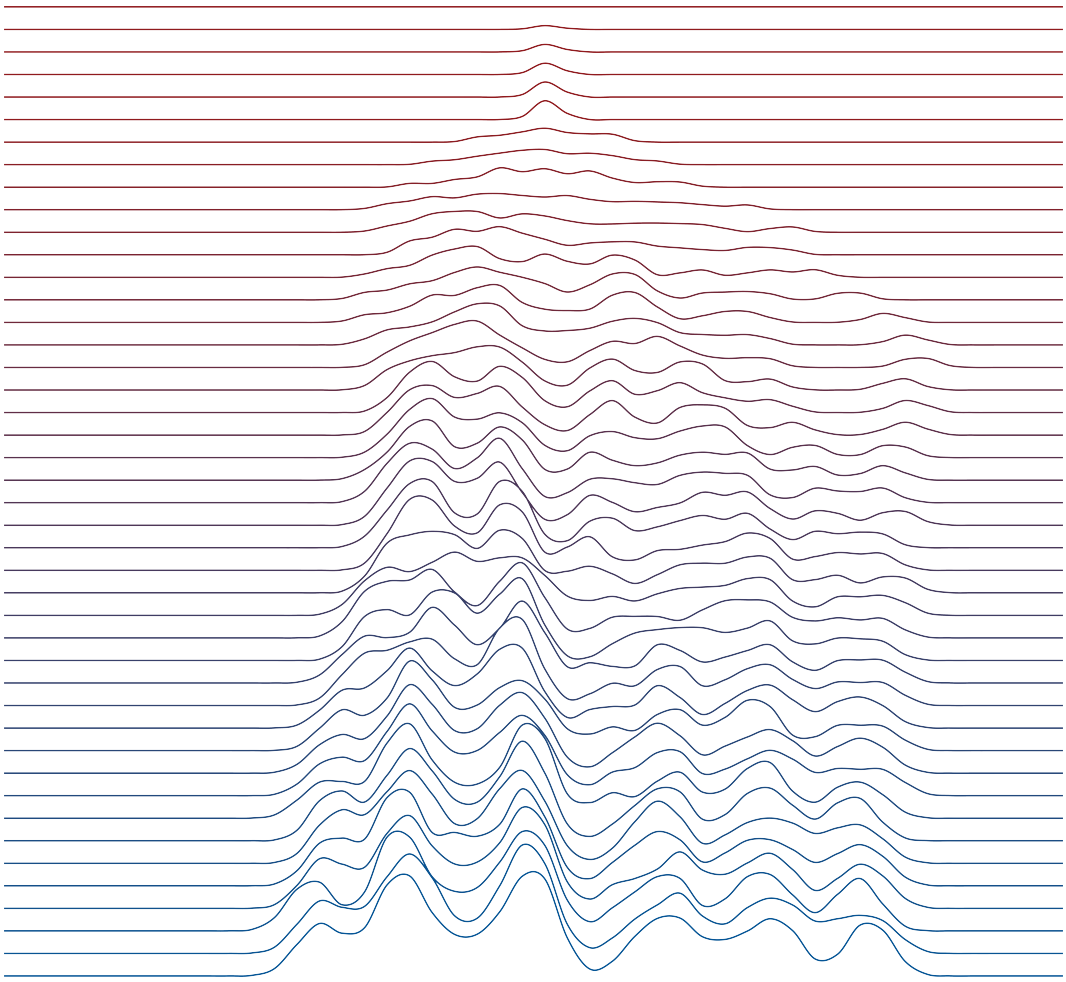
UNIVERSITY OF COPENHAGEN
FACULTY OF HEALTH AND MEDICAL SCIENCES



The Deep Generative Decoder

Development and Application to single-cell Data

A PhD Thesis by Viktoria Schuster

This thesis has been submitted to the Graduate School of Health and Medical Sciences, University of Copenhagen 31 October 2023.

The title figure was created by Jonas Sindlinger. It depicts scaled probability densities in the first dimension of a Gaussian mixture model over training time. The data were generated by Viktoria Schuster.

The Deep Generative Decoder: Development and Application to single-cell Data

Viktoria Schuster

PhD Thesis

Advisor: Anders Krogh

Center for Health Data Science,
Faculty of Health and Medical Sciences,
University of Copenhagen

31 October 2023

Supervisor: Anders Krogh
Professor, Department of Computer Science, University of Copenhagen.
Head of Center, Center for Health Data Science, University of Copenhagen.

Co-Supervisor: Tune H. Pers
Associate Professor, Novo Nordisk Foundation Center for Basic Metabolic Research, University of Copenhagen.

Preface

This PhD was carried out in the Krogh group at the Center for Health Data Science (HeaDS) from 1 November 2020 until 31 October 2023. The Krogh group, shared between HeaDS and the Department of Computer Science (DIKU), is also part of the Center for Basic Machine Learning Research in Life Science (MLLS), funded by the Novo Nordisk Fonden. The project and external stay were funded by the Faculty and Graduate School of the Department of Health and Medical Sciences. The PhD was supervised by principal supervisor Anders Krogh and co-supervisor Tune H. Pers.

The change of research environment took place in the second year of the PhD project, between 3 October and 23 December 2022 in Sarah Teichmann's group at the Wellcome Sanger Institute, Hinxton, Cambridgeshire, UK.

The thesis is comprised of three manuscripts. Two are published and one was under review at the time of submission. One additional manuscript was written as a collaborative project during the PhD, though it is not included in this work.

31 October 2023, Copenhagen, Denmark

Acknowledgements

First and foremost, I would like to express my gratitude to my principal supervisor **Anders Krogh** for great ideas, freedom to grow, and support when needed. I have enjoyed working with him immensely, especially our morning chats about new ideas and science in general.

I also want to thank **Anders** and **Thilde Bagger Terkelsen** for bringing me to the Center for Health Data Science (HeaDS). It is the best workplace I can imagine, with amazing colleagues and plenty of cake. In my time here, I have found wonderful friends. So, a special shout-out goes to HeaDS: **Jennie, Alex, Chloé, Andreas, Iñigo, Diana, Tugce, Valentina, Henrike, Thilde**, and **Anders**. **Iñigo Prada-Luengo** deserves some special recognition for saving my nerves and taking over the bulk project when I went to the UK.

Furthermore, I would like to acknowledge the people at the Center for Basic Machine Learning Research in Life Science (MLLS). It is a great machine-learning community and place for dialogue in Copenhagen. Special thanks here goes to **Kristoffer Stensbo-Smidt** for being a great collaborator and just such a nice person.

Another special thanks goes to **Sarah A. Teichmann** and her group at the Sanger Institute, particularly **Emma Dann**, for a wonderful collaboration that resulted in a great project.

I could not have grown as much as I did, both as a person and a researcher, without all the wonderful people I have met along the way. I have been very lucky to receive a ton of support and enlightenment. This especially includes **Anders**, **Jennifer A. Bartell**, and **everyone** mentioned so far. I would also like to thank my co-supervisor **Tune H. Pers** for feedback on my research progress. On that note, I would like to acknowledge all the wonderful people who have donated their time to read my thesis drafts, give feedback, and help me translate: **Anders, Diana, Thilde, Iñigo, Valentina, Andreas, Kristoffer**, and **Jonas**.

My friends and family have been a vital part of my journey and provided support, fun distractions, and serious mental health improvements. My absolute biggest supporter has been **Jonas Sindlinger**. Without him, I would have gone completely crazy - in general, with every deadline approaching, and especially during the final months of my PhD.

Summary (English)

Lower-dimensional representations offer profound insight into underlying structures and dynamics of complex and high-dimensional data. This can facilitate interpretation and knowledge transfer, which has made representation learning a vital aspect of generative modeling. There is immense potential in using generative models for the challenging tasks often encountered when working with biological and health-related data. A good example is single-cell data. Single-cell technologies provide unique snapshots of biological processes inside the cell and this information is critical to advancing medical research. However, the data is often sparse, high-dimensional, and noisy. Given the constraints and the extensive potential of such data, our primary objective was to develop a generative representation learning approach that can efficiently model small but high-dimensional datasets and provide a powerful foundation for downstream applications, while ensuring user-friendliness to promote widespread applicability.

We designed an approach that increases data efficiency by learning representations directly instead of through inference networks, called encoders. These are convenient but necessitate a higher number of parameters and thus training data. In situations with limited data, this can prove counterproductive. By learning representations directly and using flexible and more complex prior distributions, we aimed to increase the expressiveness of the generative model and the resulting representation. This adjustment can prove instrumental in capturing intricate patterns and variations embedded in biological datasets. We named our approach the "Deep Generative Decoder" (DGD) and applied it to various single-cell datasets, featuring gene expression and chromatin accessibility. The DGD not only excelled in modeling the data but also demonstrated superior clustering capabilities compared to existing models. The added modeling flexibility presents a more powerful and robust foundation for downstream analysis. We showcased the latter through *in silico* experiments of gene silencing, which resulted in the identification of regulatory regions linked to specific genes. This discovery was solely based on paired observations of gene expression and chromatin accessibility, emphasizing the model's potential to infer biological relationships from brief snapshots of the cellular state.

In conclusion, our work seeks to bridge a vital gap in generative representation learning for high-dimensional, sparse datasets. We believe the DGD can make significant contributions to biological and health-related research, paving the way for deeper, more insightful explorations.

Resumé (Dansk)

Repræsentationer af lavere dimensionalitet kan give indsigt i de underliggende dynamikker af data, som er kompleks og højdimensionel. Dette kan fremme forståelse og fortolkning af resultater, hvilket har gjort læring af repræsentationer en afgørende del af generativ modellering. Der er et markant potentiale i at bruge generative modeller for de udfordringer, som ofte opstår med biologiske og sundhedsrelaterede datasæt. Et godt eksempel er enkeltcelledata. Enkeltcelleteknologier giver unikke snapshots af biologiske processer, og informationen fra disse teknologier er afgørende for at fremme medicinalforskning. Dog er data ofte støjet, højdimensionelt og tyndt. Vores mål, givet både begrænsningerne og potentialerne af disse data, har været at udvikle en metode til generativ læring af repræsentationer, som effektivt kan modellere små, højdimensionelle datasæt og give et kraftfuldt fundament til efterfølgende anvendelser; samtidigt med, at brugervenlighed sikres.

Vi har designet en tilgang, der øger dataeffektiviteten ved at lære repræsentationer direkte frem for at bruge inferensnetværk (enkodere). Enkodere er praktiske, men kræver et højere antal af parametre og dermed mere træningsdata. Dette kan blive et problem i situationer med begrænset data. Ved at bruge fleksible og mere komplekse priorfordelinger såvel som at lære repræsentationerne direkte, har vi øget udtryksevnen af den generative model og de tilsvarende repræsentationer. Denne ændring kan være afgørende for at fange indviklede mønstre og variationer indlejret i de biologiske datasæt. Vi har navngivet metoden "Deep Generative Decoder" (DGD) og har anvendt den på forskellige enkeltcelledatasæt, som indeholder data om både genekspression og kromatintilgængelighed. Modellen har vist overlegne resultater for både rekonstruktion og clustering sammenlignet med eksisterende metoder. Den nye fleksibilitet ved modellering giver et stærkere og mere robust grundlag for efterfølgende analyser. Vi har demonstreret dette gennem *in silico* eksperimenter af gensilencing, hvilket har identificeret regulatoriske regioner forbundet med specifikke gener. Denne opdagelse var udelukkende baseret på genekspression og kromatintilgængelighed, hvilket fremhæver modellens potentiale til at aflede biologiske sammenhænge fra blot snapshots af cellulære tilstande.

Som konklusion søger vores arbejde at overkomme et væsentligt hul i generative repræsentationer for højdimensionelle, tynde datasæt. DGD fylder dette hul og kan bidrage yderligere til biologisk og sundhedsrelateret forskning ved at bane vejen for dybere og mere indsigtsfulde opdagelser.

List of Publications

V. Schuster and A. Krogh†, "A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder", *Entropy*, vol. 23, no. 11, p. 1403, October 2021.

V. Schuster and A. Krogh†, "The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data", *Bioinformatics*, vol. 39, no. 9, September 2023.

V. Schuster, E. Dann, A. Krogh†, S. A. Teichmann†, "multiDGD: A versatile deep generative model for multi-omics data", *bioRxiv preprint*, 2023.

Other relevant publications

The following publication was a collaborative effort at the Center for Health Data Science.

I. Prada-Luengo*, **V. Schuster***, Y. Liang*, T. Terkelsen, V. Sora, A. Krogh†, "N-of-one differential gene expression without control samples using a deep generative model", *bioRxiv preprint*, 2023.

† Corresponding author
* These authors contributed equally.

Contents

1 | Generative Modeling and Representation Learning

1.1 Introduction

Generative modeling aims to describe unknown data distributions by finding approximate data-generating functions. Given that the approximation is sufficient, a generative model can synthesize new data. The year 2022 has seen an explosion of large generative models trained on vastly available, unlabeled, and thus "cheap" data [1]. Applied to domains such as text, image, video, and audio, generative models can for example answer questions (albeit not always grounded in truth), create art from a single sentence, and write code [1]. Generative models have also shown great potential in the medical field [2, 3, 4], although nowhere close to the extent of text and image generation yet. There are many tasks of interest in healthcare, such as disease diagnosis [2], risk assessment [2], event prediction [2], cancer classification [2], and molecule (drug) design [5, 6, 7]. Thanks to the increasing accessibility of data, these tasks have seen great advancement through deep learning [2]. However, health data present unique challenges and often possess at least one of the following characteristics: high-dimensionality, heterogeneity, sparsity, irregularity (i.e. missing data), temporal dependency, or lack of harmonization (i.e. inconsistencies stemming from different ontologies) [2]. In addition, existing data often lack quantity, diversity (in terms of individuals), and quality in order for deep learning approaches to be effective [8]. Many methods developed are very task-specific and rely on relatively simple neural network (NN) architectures [2]. The trends do, however, seem to shift towards more powerful implementations and generative models [4, 5, 6].

Powerful generative models are typically based on NNs, and their space of architectures and protocols is vast. Even though the Transformer [9] has become a predominant architecture in many fields [1, 10], it may not be the optimal choice in every case. Plenty of methods have been developed over the last two decades, all of them with their own trade-offs [11]. We can try to classify existing methods into four categories [12], although the distinction is often difficult, and so-called hybrid models combine methods from different categories [11]. These four categories are *autoregressive, flow-based, latent variable* and *energy-based* models [12]. Part of their core objectives are to model processes (*autoregressive*) [13], estimate exact densities (*flow-based*) [14], and find underlying data distributions (*latent variable* [15], *energy-based* for unnormalized distributions [11]). These objectives are

represented in their perspective on the data variable $\mathbf{x} \in X$, and its marginal probability $p(\mathbf{x})$ [12]. Autoregressive models treat the data as a sequence of variables and describe the probability as a product of conditional probabilities [12]. These represent likelihoods of observations given previous observations in the sequence. Flow-based models make use of the change of variable formula and are trained through invertible functions [12]. Energy-based models view the marginal probability from the perspective of physics, making use of energy functions [12]. Lastly, latent variable models assume the presence of an underlying generative variable [16]. We will focus on latent variable models due to their relation to representation learning, which will be outlined in the following section.

1.2 Latent Variable Models and Representation Learning

The idea behind latent variable models (LVMs) is that complex data distributions are generated from simpler, often lower-dimensional random variables and functions mapping this *latent* or *underlying* variable into data space. Because we assume the latent variable to be of a simpler form than the data distribution, we tend to approximate it with simple and well-behaved distributions [15]. As a result, LVMs naturally lend themselves to representation learning, where the goal is to extract lower-dimensional views of the data that are easier to interpret and use for downstream tasks [15]. Deep LVMs combine this goal with computational efficiency by learning a NN as the generative function $f : Z \to X$. This function maps the underlying (random) variable Z to the observed data space X. We can express this in a probabilistic generative process as done in [16]:

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) \\ \mathbf{x} &\sim p(\mathbf{x} \mid \mathbf{z}) \ . \end{aligned} \tag{1.1}$$

Here, values $\mathbf{z}$ of the random variable Z are drawn from the distribution $p(\mathbf{z})$, often called the prior. Values of the observed variable X are generated from the conditional distribution $p(\mathbf{x} \mid \mathbf{z})$, the likelihood. Because the generative process of a NN is a function parameterized by θ, the likelihood is given as $p(\mathbf{x} \mid \mathbf{z}, \theta)$ or $p_\theta(\mathbf{x} \mid \mathbf{z})$. For simplicity, the parameter dependency will be left out here. The objective of many generative processes is given by the marginal probability of the data

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} \\ &= \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \ , \end{aligned} \tag{1.2}$$

where $p(\mathbf{x}, \mathbf{z})$ presents the joint probability of observed and latent variable [16]. This is an important part of probabilistic generative modeling, as it introduces uncertainties about the latent variable [16]. However, computing the marginal probability is often intractable in deep learning. As a result, many approaches revert to approximations or sampling [16].

LVMs are closely related to representation and manifold learning. They all aim to explain important variation and structure of the data in a typically lower-dimensional space. This makes LVMs and representation learning subjects of interest across many research domains and practical applications. In the Krogh group, our interests lie in simple, yet elegant and efficient solutions to generative representation learning for biological data. Simple solutions mean looking at what is absolutely necessary. In a LVM, this boils down to the representation and a generative function mapping from representation to data space. When this generative function is a NN architecture, it is typically called a generator or decoder.

1.3 A Review of Encoder-less Latent Variable Models

The simplest formulation of a LVM can be described by a latent variable or representation $\mathbf{z}$, usually of lower dimensionality than data $\mathbf{x}$, and a mapping that transforms samples in representation space into samples from the complex data space. This view is more common in traditional dimensionality reduction algorithms [17]. However, these often do not generalize well on the increasingly complex tasks we attempt to solve [15]. It is quite common to introduce a mapping from data to representation space for the sake of efficient inference and approximation of the distribution over $\boldsymbol{z}$ [15]. This mapping is usually referred to as an encoder. In the deterministic case, the combined use of encoder and decoder is called an autoencoder (AE) [18]. Deterministic here means that there is no assumption of a distribution over $\mathbf{z}$, unlike in probabilistic approaches. One of the most prominent examples of probabilistic LVMs is the variational autoencoder (VAE) [19, 20, 21].[1] In recent years, several approaches have been proposed that do not require encoders [22, 23, 24, 25, 26, 27], with more or less success in their application. Their motivations have been to reduce the constraints on representations [22] and the difficulties associated with common training processes [25], to improve parameter efficiency of the whole network [23], and to make handling incomplete data straightforward [22]. Navigating the field of encoder-less models can be difficult. The space of generative models is vast and there are no consistent naming conventions between subfields. It is important to note here that while popular methods like diffusion models [28] and normalizing flows [29, 30] include no encoders either, they do not inherently qualify as LVMs learning a lower-dimensional representation. They can, however, be combined with other LVM approaches into hybrid models [11]. The space of encoder-less LVMs is rather limited and has resulted in few methods that are powerful, flexible, and simple. It becomes even smaller when only considering approaches with high capacity, i.e. non-linear mappings with the potential to describe complex functions. None of the encoder-less methods discussed here have truly found widespread application. This emphasizes the need for further investigation.

[1] The VAE is introduced in Methodology Section 3.2.

The term "encoder-less" representation learning brings up the question of how to learn representations without the data as input. Most approaches without encoders learn the representations as another set of parameters [22, 24, 25]. For every data sample $\mathbf{x}_i$ used for training, there exists a designated representation vector $\mathbf{z}_i$ which will be updated during training to maximize the training objective.[2] The optimization of these representation vectors is straightforward. During backpropagation [31, 32], gradients for each parameter are calculated based on the training loss. As a result, representations as parameters receive updates automatically [22]. The biggest differences between existing encoder-less LVMs are the type of mapping, optimization approach, choice of prior over $\mathbf{z}$ and initialization [22, 23, 24, 25, 27]. Most powerful approaches utilize neural networks as mappings, which are then referred to as either generators or decoders [22, 23, 24, 25, 27]. However, Gaussian process (GP) [33] LVMs (GP-LVMs) [26] have been a relatively popular method among encoder-less LVMs [34], also in the field of biology [35, 36, 37]. Gaussian process latent variable models (GP-LVMs) pair the inference of latent variables with the non-parametric mapping of a Gaussian Process [26]. In their original form, they enable the optimization of representations through marginalization (i.e. integrating out) of the mapping [26]. A graphical model of classical GP-LVMs [34] is included in Figure 1.1 among the other encoder-less LVMs discussed in this section. Some of these approaches are not considered generative models, like alternating back-propagation (ABP) [22] and the original GP-LVM [26]. They can, however, be transformed into generative models given a prior over $\mathbf{z}$ [23]. In terms of optimization, one can either choose to maximize the "complete-data" likelihood $p(\mathbf{x}, \mathbf{z})$ [22, 23] or the observed-data likelihood $p(\mathbf{x})$ [22, 26].

[2] For more details on parameter estimation and training objective see Section 3.1.

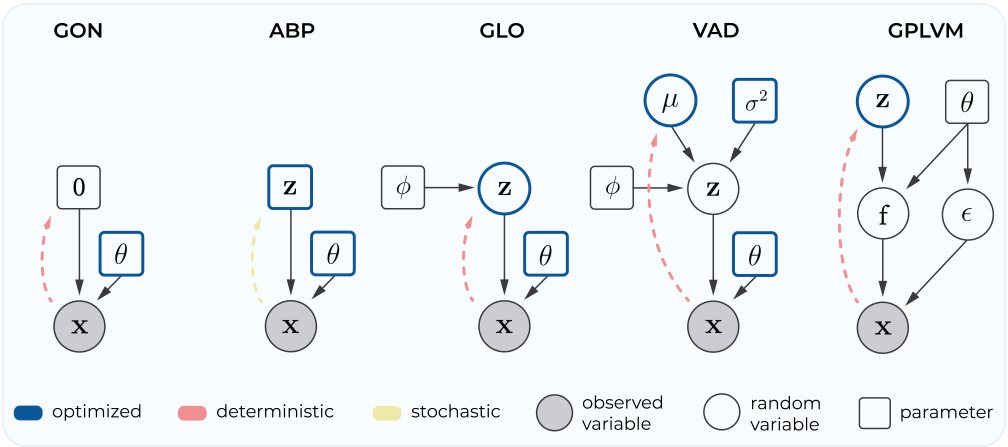


Figure 1.1: Encoder-less LVMs. Simplified graphical models of encoder-less LVMs, ordered by their complexity from left to right. Circles present random variables, without fill for unobserved and with grey fill for observed variables. Boxes depict parameters. Blue outlines represent learned parameters or variables, the latter thus also being parameters. Black arrows present directed dependencies. Dashed arrows stand for updates to the latent representation, with the color representing the type of update. Deterministic refers to gradient-based updates, stochastic refers to sampling-based approaches. The abbreviated names of the depicted approaches are shown above each graphical model.

The latter requires the latent variable to be integrated or sampled "out", but provides robustness with respect to uncertainty about $\mathbf{z}$. Many of the presented methods employ variational inference (VI) [38, 39] for that purpose, and use either standard Gaussian priors [23, 24] or post hoc learned priors [40]. The original ABP follows the sampling approach and performs stochastic optimization [22], which has been accelerated by later work running shorter sampling steps with optimal transport [41]. Most of the models depicted in Figure 1.1 initialize the representations $\mathbf{z}$, or in the case of the Variational Auto-Decoder (VAD) [24] the variational means $\boldsymbol{\mu}$, as random Gaussian samples [22, 24, 25]. They also use this initialization at inference, when representations for new samples are learned [22, 24, 25]. A unique alternative, the Gradient Origin Network (GON) [23], initializes representations as zero vectors (at *origin*) and updates them via one *single* gradient step [23]. Both the Generative Latent Optimization (GLO) approach [25] and the original GP-LVM have, among other initialization schemes, performed principal component analysis (PCA) [25, 26]. GP-LVMs, despite all their improvements in terms of scalability and flexibility [34, 42, 43], are very sensitive to the initialization of the latent vectors, which should be close enough to the underlying manifold [26]. Several extensions now include an encoder due to the inefficiency of sampling [44].

Sticking to simple Gaussian priors or deterministic representations presents one of the key limitations of all these approaches. Representations become limited in terms of expressiveness, and may not generalize well to tasks outside the realm of the investigated image domain. Encoder-less LVMs present a key research topic in the Krogh group. As discussed above, existing methods show strong deficiencies, and attempts at improving on these often result in overly complex concepts. Nevertheless, encoder-less models have the potential to be versatile and useful for many downstream tasks, which makes it worthwhile to investigate alternative approaches. They reduce the number of trainable parameters and can handle missing data naturally.

2 | Single-cell Transcriptomics: A Window into the Cell

2.1 Introduction

The human body is a complex system made of millions of cells. Malfunctions in this system, such as aging and disease, can often be traced back to abnormalities at the cellular level. For this reason, it is important to understand the underlying mechanisms and dynamic states of cells. To gain insight into specific cellular mechanisms, we can capture snapshots of the cellular state and cell-cell communication. Measurements of different parts of the cell create windows into the state of the genome, epigenome, and transcriptome (Figure 2.1A). All these parts are connected in the central dogma of biology (Figure 2.1B), and help us understand why and how things go wrong.

The transcriptome comprises the ribonucleic acid (RNA) molecules of a cell, which are produced during gene expression. Measuring it at a single point in time has been made possible and feasible by the invention of high-throughput sequencing technologies [45]. These are collectively referred to as next-generation sequencing (NGS). In RNA sequencing (RNA-seq), the RNA is isolated, converted into complementary DNA (cDNA), sequenced, and then aligned to a reference genome [46]. After alignment, sequencing reads are quantified to obtain gene counts for each measured cell (Figure 2.1C). Changes to the cellular state include changes to transcription, which is why transcriptomics data provides valuable insight. Until roughly a decade ago, data were mainly generated by bulk sequencing. As the name suggests, bulk samples contain multiple cells from a given tissue and are sequenced together. However, technological advances now allow us to capture gene expression at single-cell level [45]. The capacity and quality to which we can process cells using single-cell RNA sequencing (scRNA-seq) has increased rapidly [45, 47]. Technologies now are dominated by microfluidics approaches, isolating hundreds of thousands or even millions of cells [48] in tiny droplets [49, 50, 51], and giving them individual barcodes [52]. It has also been made possible to simultaneously measure multiple snapshots together, which we then call *modalities*. Many additional technologies are arising that can include even more modalities [53]. The so-called multi-omics data have become very popular since their inception [53] and present the state of the art in many fields [54]. Measuring the transcriptome alongside the epigenome or proteome has become common practice [55]. This first combination of gene expression and chromatin accessibility will

be the focal point of this work. Chromatin accessibility is commonly measured by the Assay for Transposase-Accessible Chromatin (ATAC) [56]. It can provide additional insight into accessible regions in the chromatin and regulatory elements [55], which control cell-type-specific gene expression patterns and dynamics [57]. Accessible regions are typically pooled into *peaks*, which are reported as dataset-specific features [56, 58]. Continuous efforts are being made to provide an extensive collection of cell-level omics data from the whole body [45, 59]. Such a holistic view could provide us with a deeper understanding of the processes of aging, disease, and development [45]. Single-cell transcriptomics has already enabled several new insights into cell subtyping and discovery [45, 60], cell differentiation and gene regulation [54, 61], and disease- and immune-related genes and cell types [45, 54, 60, 61, 62, 63].

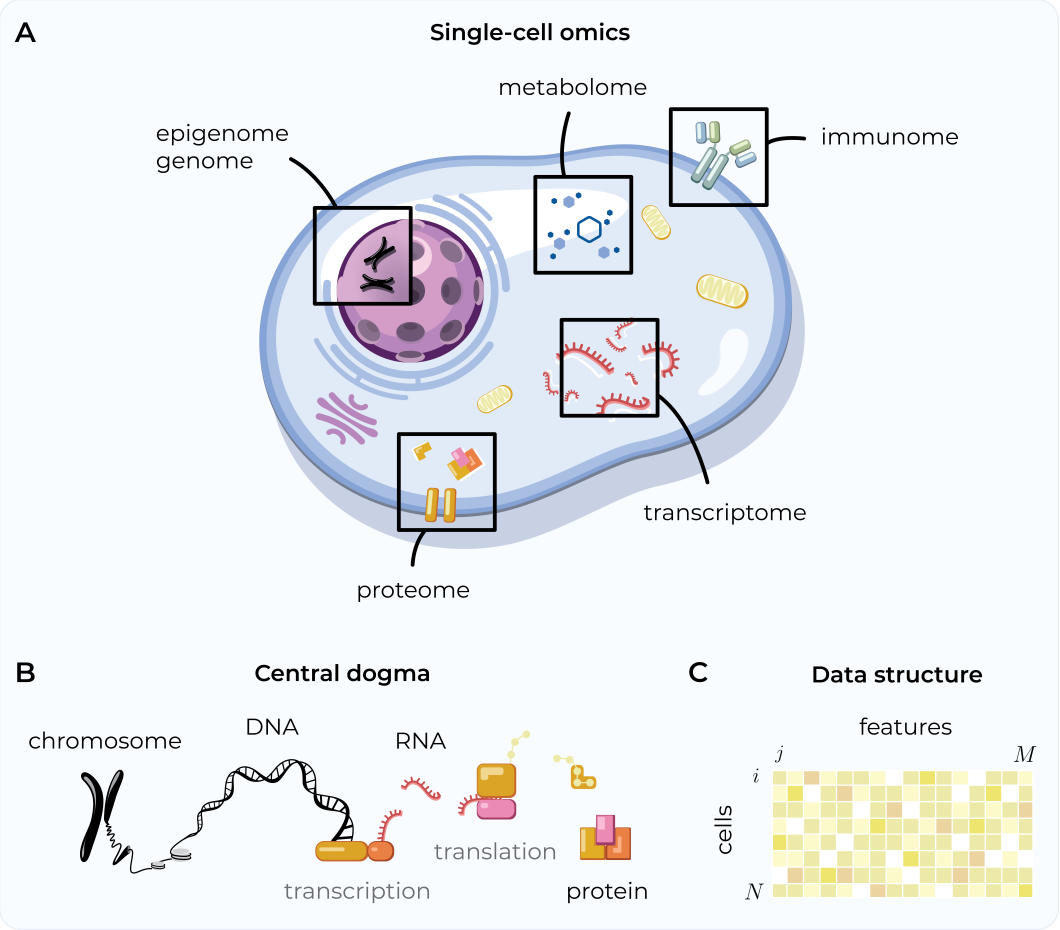


Figure 2.1: An overview of single-cell modalities. A Simplified illustration of single-cell modalities as windows into different structural and functional aspects of the cell. **B** Simplified illustration of the central dogma of biology. Deoxyribonucleic acid (DNA) is compactly organized in chromosomes. It is transcribed into RNA, which is translated into peptide sequences. These peptide sequences fold into proteins, which fulfill many important functionalities in the cell. **C** Illustration of a single-cell *omics* count matrix. Cells present samples as N rows. The modality's features present the columns with a dimensionality of M. Cells and features are indexed by i and j, respectively.

2.2 Computational Methods

Despite their resolution, high-throughput single-cell sequencing technologies only capture a fraction of the molecules present in a cell [64], and there are various sources of systematic and random noise [47, 48, 55]. This noise and uncertainty about the observed measurements makes data preprocessing a crucial step [55] and has led to a preference for probabilistic models in data analysis [47]. The advancements in single-cell technology have gone hand in hand with the development of computational methods to tackle preprocessing, the integration of datasets, and analysis [47]. As of recently, there were over 1400 tools available for the mentioned tasks [55, 65]. This large space of potential methods complicates workflow design [55]. Projects like the scverse [66] in Python and Bioconductor [67] in R promise to improve the user experience by providing a framework for compatible software for various tasks [66]. If we want to make the most out of the increasing amount of data available, deep learning will become an essential aspect of working with it. Foundational models especially have gained significant attention within the single-cell research community. Given this desire, representation learning is likely to assume a pivotal role in method development. In this area, the space of available methods shows a lot of potential for improvement. Before diving into the current state of this field and its effect on downstream data analysis, I will briefly introduce some of the most common components of data preprocessing.

2.2.1 Preprocessing

Single-cell data offer higher resolution than bulk sequencing data at the cost of more noise and sparsity [48]. Several sources of noise are outlined in a brief introduction to single-cell sequencing in Methodology Section 3.4. Low-quality samples and some sources of noise are removed during quality control, the first step in most preprocessing procedures. A single preprocessing and analysis pipeline does not exist, due to the vast space of possible downstream tasks [55]. Many preprocessing workflows do however include normalization of the data and feature selection after quality control. Normalization typically brings all cells to the same *library size*, which is the total number of all transcripts j measured in a cell i (Figure 2.1C). In feature selection, oftentimes only highly variable genes are selected as informative features. Normalization and feature selection can be prohibitive to downstream tasks and may have to be considered to be skipped or moved to later steps when needed [55].

2.2.2 Data Integration

Data integration has become an important subject to the development of computational methods, as it can help create larger, harmonized datasets. We can integrate data based on cell identity and unique features. Different transcriptomics datasets can easily be integrated by aligning the cell counts according to the shared set of genes. When multiple modalities are measured simultaneously, integration becomes more complex [68]. Modalities can be aligned according to cell identity, which is called paired integration [55, 68].

Diagonal integration is often referred to as unpaired integration, as it contains only one modality for each cell. Mosaic data presents a mix of paired and unpaired data [55]. Paired data integration naturally is a much simpler problem to solve [55]. Either way, integration is not just trivially achieved by concatenating matrices. Differences between datasets, for example bias from different sequencing equipment, may result in a significant fraction of the heterogeneity in the data [48]. This can prevent us from identifying important, more subtle patterns. As a result, computational methods alleviating these biases have become a vital aspect of data integration.

For a long time, most methods for data integration were based on k-nearest neighbors, Bayesian models, and matrix factorization [55]. There have been several challenges, including missing data, the combination of statistical models for different modalities, overfitting due to small datasets, noise disentanglement, and scalability [68]. Some of them still remain even in the newest methodologies. By now, there are many different integration approaches available for both transcriptomics and multi-omics data, dominated by matrix factorization, network-based approaches, or deep learning [55, 69, 70]. Because of the massive increase in data volume and complexity, deep learning approaches are becoming increasingly intriguing [71, 47].

Application of Representation Learning

Most deep learning approaches applied to data integration focus on representation learning. The goal is to achieve lower-dimensional, more interpretable embeddings (an often used term for representations here), and to provide "denoised" or "corrected" samples [55, 68]. Autoencoders (AEs) and variational autoencoders (VAEs) are the most prevalent methods applied [68]. In a recent benchmark of 68 single-cell integration methods, deep learning-based approaches (scANVI [72], scVI [73]) and one matrix factorization-based approach (Scanorama [74]) were among the overall best performing embedding methods for transcriptomics data [75]. They tested various evaluation metrics in terms of scalability, usability, and batch effect removal. However, all existing methods for transcriptomics and multi-omics integration have their trade-offs [70, 75]. One important limitation to note is that even some of the best integration methods favor feature selection of highly variable genes [75], which limits the capability to study more fine-grained distinctions between cells. Additionally, the objective to "denoise" data has led to underperformance of many deep learning models by accepting strong decreases in prediction variability as noise removal. Plenty of noise propagates errors into the data, making analysis of downstream feature-level interactions difficult [48]. But since we cannot yet sufficiently disentangle sources of noise [48], we are in danger of missing important cell heterogeneity this way.

2.2.3 Analysis

Because of the curse of dimensionality, lower-dimensional representations are vital to single-cell data analysis [47, 55]. The methodologies discussed above for data integration have been instrumental in improving data correction and analysis including imputation, batch correction, cell type identification, and trajectory analysis [48]. They not only allow us to perform analysis on larger, harmonized datasets, but they also provide more interpretable latent representations. Their use, however, has so far been limited. Deep learning methods such as VAEs are mostly used to integrate datasets, denoise counts, and perform cell type identification [68]. Their representations are mainly used as a basis for 2D visualization [47] and subsequent exploratory data analysis [76]. Analysis based on 2D visualizations can be misleading because employed methods do not preserve distances or relationships within the data [76]. Specific analyses of interest such as trajectories, differential expression, gene set enrichment, cell composition changes, perturbation effects, and cell-cell communication are still computationally limited [55]. This is also true for biological relationships we hope to learn from other modalities than the transcriptome. Candidate *cis*-regulatory elements, for example, are typically identified through enriched peaks in cell clusters [57]. Linking *cis*-regulatory elements to target genes is still underexplored, and mostly identified using correlation-based approaches that struggle with the sparse and noisy data [55].

Altogether, even though the space of existing tools is vast, there is a lot of room for improvement and further development. Likely, many new discoveries from single-cell data will depend on powerful, interpretable latent variable models (LVMs) that can incorporate and support advanced computational analysis.

3 | Methodology

The main methodology consists of the Deep Generative Decoder (DGD) and its implementations, which were developed as part of this work. The DGD is described in the Contributions Chapter 5.4. Here, I will briefly revisit concepts and methods relevant to the understanding of this thesis. Detailed methodologies for the contributions can be found in manuscripts II and III.

3.1 Parameter Estimation: MLE vs. MAP

Neural networks (NNs) are handy models to use when the underlying processes are not known mathematically, and the relationships between or within the data are believed to be highly non-linear. In general terms, NNs present complex non-linear functions f_θ with parameters $\theta \in \Theta$ for which we try to find "optimal" parameters θ^* such that they sufficiently approximate the data [77]. Finding these parameters is commonly referred to as *learning* or *training*. The first thing we need is an objective function $\mathcal{L}$, which evaluates how well the model fits (or explains) the data given the current parameters. It also allows us to compute gradients that update θ, in order to improve our model with respect to this objective [77]. In very simple settings, the objective may be represented as minimizing the distance between the observed data and the model output. We can generalize this objective as maximizing the probability of the data, $\mathbf{x}$, given our model $f_\theta : p(\mathbf{x} \mid \theta)$. Maximizing this objective with respect to the parameters,

$$\theta^*_{\text{MLE}} = \arg\max_\theta \mathcal{L} = \arg\max_\theta \; p(\mathbf{x} \mid \theta) \tag{3.1}$$

is called maximum likelihood estimation (MLE) [77]. In this section, the likelihood is written in notation $p(\mathbf{x} \mid \theta)$ (instead of $p_\theta(\mathbf{x})$ as in the rest of this work) to highlight the origins of the two estimation approaches discussed here. MLE assumes that the parameters all weigh in equally, i.e. that they are uniformly distributed [77]. If this is not the case, we need to include a prior over the parameters $p(\theta)$ as a regularization. The optimization objective thus becomes

$$\arg\max_\theta \mathcal{L} = \arg\max_\theta \; p(\mathbf{x} \mid \theta)\, p(\theta) \; . \tag{3.2}$$

Making use of the Bayes' theorem and the fact that $p(\mathbf{x})$ is independent of the parameters, this can be written as

$$\theta^*_{\text{MAP}} = \arg\max_{\theta} p(\mathbf{x} \mid \theta)\, p(\theta) = \arg\max_{\theta} p(\theta \mid \mathbf{x}) \,. \tag{3.3}$$

The objective is now to maximize the posterior distribution $p(\theta \mid \mathbf{x})$, which is why θ^*_{MAP} is called a maximum *a posteriori* (MAP) estimate [77]. A common way of performing this estimation is by maximizing the average value of that objective with respect to a randomly drawn set from the data

$$\mathcal{L}(\theta) = \mathbb{E}_{p(\mathbf{x})}\mathcal{L}(\theta, \mathbf{x}) \tag{3.4}$$

by updating the parameters θ via the gradient of $\mathcal{L}$ [77]. A simple algorithm for updating the parameters via the gradient is stochastic gradient descent (SGD) [78]. This is also the basis for many modern optimization schemes.

3.2 Variational Autoencoders

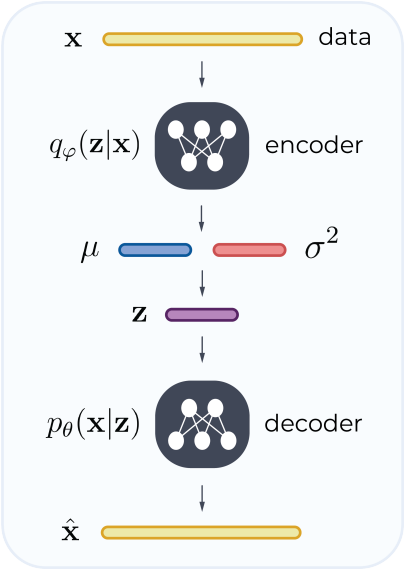


Figure 3.1: Schematic of the classic VAE. Data is passed through the encoder, which produces a sample-specific mean and variance vector. These are used in the reparameterization trick to sample the representation $\mathbf{z}$, which is fed to the decoder to reconstruct the data.

The manuscripts included in this thesis do not present research in the application or extension of variational autoencoders (VAEs). However, we have drawn parallels and performed experimental comparisons of our work to the VAE and relevant implementations.

The VAE is a probabilistic generalization of the deterministic autoencoder (AE) [19]. An AE consists of two mappings implemented as NNs [18]. The first, referred to as the encoder, maps from data space to a usually lower-dimensional latent space. In probabilistic terms, this can be written as $p(\mathbf{z} \mid \mathbf{x})$. The second one maps back to data space. It is called the decoder with likelihood $p(\mathbf{x}|\mathbf{z})$. The generative process involving the decoder was introduced in Equation 1.1. The inference process represented by the encoder aims to describe the posterior $p(\mathbf{z} \mid \mathbf{x})$ [19]. With only the observed values $\mathbf{x}$ known, the posterior

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p(\mathbf{x})} \tag{3.5}$$

cannot be computed. The marginal probability $p(\mathbf{x})$ is intractable due to the integral in Equation 1.2 [38]. Variational inference (VI) [38, 39] was introduced as a general approximation technique in this case and is not specific to just the VAE. Its name is derived

from the approximation of the posterior $p(\mathbf{z} \mid \mathbf{x})$. Besides optimizing the reconstruction of the observed data through $p(\mathbf{x} \mid \mathbf{z})$, VI aims to find a distribution $q(\mathbf{z} \mid \mathbf{x})$ that is as similar as possible to $p(\mathbf{z} \mid \mathbf{x})$ [38, 39]. The search space is limited to distributions that are of simpler form than the true posterior and parameterizable. This is called a variational distribution family [38]. The second objective of a VAE is thus to find parameters φ of this distribution so that $q_\varphi(\mathbf{z} \mid \mathbf{x})$ approximates the true posterior $p(\mathbf{z} \mid \mathbf{x})$ as well as possible [38, 39]. With the variational approximation, the log marginal likelihood can be expressed as

$$\log p(\mathbf{x}) = \mathbb{E}_{z \sim q_\varphi(\mathbf{z} \mid \mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}[q_\varphi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] + \mathrm{KL}[q_\varphi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})] \,. \quad (3.6)$$

For details of this derivation see [16]. The first term presents the negative reconstruction error of the model. The second term is a distance measure between the aggregate approximate posterior and the prior, given by the Kullback-Leibler (KL) divergence. The third term is another KL divergence, but of the approximate posterior and the true posterior, which we cannot compute [16]. Because the KL divergence is always non-negative, the first two terms serve as a lower bound, a proxy, to the log marginal probability and are called the evidence lower bound (ELBO) [19]

$$\log p(\mathbf{x}) \geq \mathbb{E}_{z \sim q_\varphi(\mathbf{z} \mid \mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}[q_\varphi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] \,. \quad (3.7)$$

Since the ELBO is tractable for simple variational distributions and priors, it is used as the training objective. In VAEs, $q_\varphi(\mathbf{z} \mid \mathbf{x})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$ are parameterized by NNs with parameters φ and θ, respectively [19]. The prior $p(\mathbf{z})$ is typically chosen to be Gaussian. This original setup is often referred to as the *classic* or *vanilla* VAE. A Gaussian prior is convenient, as it leads to a simple KL divergence when both the approximate posterior and the prior are Gaussian [19]. In order to calculate the expectation in Equation 3.7, samples from the approximate posterior are generated by separating the parameters and random factors. Representations

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \quad (3.8)$$

are sampled from parameterized means $\boldsymbol{\mu}$ and covariances $\boldsymbol{\sigma}^2$ through a random noise sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{I})$. This so-called reparameterization trick circumvents backpropagating through random nodes [19]. A summary of the VAE is depicted in Figure 3.1.

Beyond the classic VAE

Many extensions and improvements to the original VAE framework have been made since its publication [11]. More complex priors have been added [11], for example by using more complex distributions [79, 80], energy-based approaches as hybrid models [81], score matching [82, 83], or normalizing flows [84, 85] in representation space. Complex priors

can also be achieved by diving into non-Euclidean space [86], or by building hierarchical VAEs [87] with conditional representations [85, 88, 89]. Advancements have also been made in closing the inherent approximation gap of VI [11], for example through sampling-based methods as in semi-amortized VAEs [90]. These are just some examples of the continuous research on VAEs.

Implementations and Applications

In all our works, we either implemented the VAE (and its encoder-less variation, the Variational Auto-Decoder (VAD) [24]), or applied existing models in order to compare usability and performance. These include scVI [73] for single-cell transcriptomics and MultiVI [91] for single-cell multi-omics data.

3.3 Gaussian Mixture Models

Mixture models are a combination of simple distributions that can be seen as another type of generative model [77]. They are often applied to unsupervised clustering of continuous data. Even though they are tractable and can be very expressive, they are computationally inefficient at large data dimensionalities [77]. A mixture model generally consists of K distributions with mixture weights π_k and a likelihood $p_\phi(\boldsymbol{z} \mid k)$ parameterized by ϕ [77]. Here, data values are denoted as $\boldsymbol{z}$ because mixture models will in this work only be discussed for modeling representations. The generative process of a mixture model is to first sample from the categorical prior over the components k, and then to generate observations $\boldsymbol{z}$ given the parameters of the kth component [77].

$$p_\phi(\boldsymbol{z}) = \sum_{k=1}^{K} p(k)\, p_\phi(\boldsymbol{z} \mid k) = \sum_{k=1}^{K} \pi_k\, p_{\phi_k}(\boldsymbol{z}). \tag{3.9}$$

For a Gaussian mixture model, we simply plug in the probability density of a multivariate Gaussian for $p_{\phi_k}(\boldsymbol{z})$ so that

$$p_{\phi_k}(\boldsymbol{z}) = \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3.10}$$

The mixture weights π_k, means $\boldsymbol{\mu}_k$ and covariances $\boldsymbol{\Sigma}_k$ represent the parameters of the Gaussian mixture model (GMM) [77]. In manuscripts II and III, these parameters are learned and regularized by priors of their own. Details can be found in the methodologies of the corresponding papers.

3.4 Single-cell Sequencing

Even though applying single-cell sequencing technologies was not part of this work, a basic understanding of the process and potential sources of error are important for working with this type of data. Thus, I will briefly describe relevant steps and biases involved in single-cell ribonucleic acid (RNA) and Assay for Transposase-Accessible Chromatin (ATAC) sequencing.

Cell Isolation

The data used in this work are based on popular microfluidics approaches [45, 55], capturing single cells and barcode beads together in small droplets [49, 50, 51, 52]. Sometimes, droplets remain empty or capture more than one cell (so-called doublets). These samples are removed later in quality control [55, 92].

Library Preparation

The beads that are captured with the cells contain adaptors, primers, and additional identification tags [49, 50, 51]. Adaptors and primers ensure deoxyribonucleic acid (DNA) immobilization, amplification, and sequencing [93]. Typical additional tags are cell-specific barcodes and so-called unique molecular identifiers (UMIs) that label each individual target molecule captured before amplification [46, 93, 94]. It is important to note that we only observe a fraction of the target molecules in each cell. Typically, high-throughput sequencing captures between 5 and 20 percent of present RNA molecules [64]. As mentioned above, library preparation is based on DNA. For single-cell RNA sequencing (scRNA-seq), RNA molecules from the cell lysate are reverse-transcribed into complementary DNA (cDNA) before fragmentation and tagging [93]. ATAC-seq makes use of an engineered version of an enzyme recognizing sequence motifs found all over the genome and cuts out the corresponding DNA segments [56, 95].

From Sequencing to Count Matrix

After tagging, DNA fragments from either scRNA-seq or scATAC-seq are amplified using polymerase chain reaction (PCR) and processed via next-generation sequencing (NGS) [46, 93, 95]. After sequencing, quality control and alignment to the reference genome are performed [58, 92]. Some of the biases and sources of noise in the data have been mentioned already. There are additional concerns that have to be addressed in quality control, for example dead cells and their *ambient* RNA contaminating other samples [92], artifacts, amplification bias, and sequencing errors [46]. Quality control is an important step and strongly influences downstream analysis, which is discussed in more detail in other works [92]. The high-quality reads aligned to the reference genome are counted by individual cell barcodes and UMIs [46, 93]. The aligned sequences are thus converted into features, dependent on the modality measured. RNA-seq counts are mapped to corresponding genes and provided as cell-by-gene expression count matrices [92]. Chromatin

accessibility data is not as standardized [55]. Here, it is common to perform *peak calling* [56, 58]. It describes a computational method that identifies accumulated fragments on the aligned genome. These *peaks* present the features in ATAC-seq data [56, 58]. When measuring multiple modalities in one cell [96], there are new biases to be considered. Protocols for the separation and access to one modality can negatively affect the quality of the other, leading to trade-offs and potentially enlarged differences between data processed in different batches [58].

3.5 Modeling Count Data with Negative Binomials

There are many uncertainties linked to single-cell count data [48]. As a result, it is common to approximate these counts with the Negative Binomial distribution which fits the count distributions well [97]. This probabilistic objective enables us to make assumptions about estimated counts in a feature- and dataset-specific context. As an example, we can look at the two different Negative Binomial distributions shown in Figure 3.2. They have the same mean, but very different profiles because of distinct variances. The probability masses depicted make it clear that predictions are not equally likely given the underlying distributions. The likelihood is commonly modeled as

$$p_\theta(x_{i,j}) = \mathcal{NB}(x_{i,j} \mid \mu_{i,j}, r_j). \tag{3.11}$$

with $x_{i,j}$ as the count of cell i and feature j and a feature-specific dispersion factor r_j, which is inversely correlated with the variance.

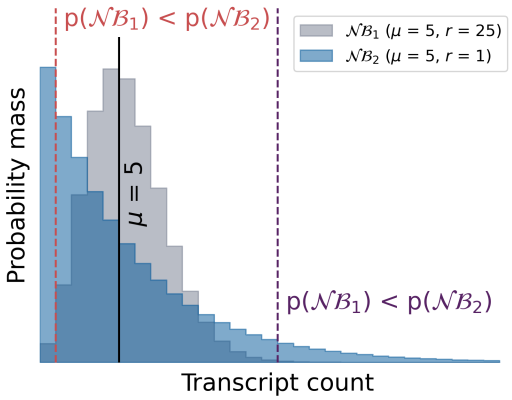


Figure 3.2: Simulation of gene expression distributions. Probability masses from two Negative Binomial distributions are plotted against the sampled counts. The distributions share the mean (black line), but exhibit different probability masses due to distinct dispersion factors/variances.

4 | Objectives

Gene expression data and other cellular modalities promise to reveal a multitude of cellular mechanisms. Representation learning naturally lends itself to such data for several reasons. It can help uncover relevant features and patterns, integrate data types with a lot of heterogeneity, transfer extracted knowledge, and enhance predictive models. Even though more and more data have become available along with tools to analyze them, we are still lacking powerful and generalizable methods. In order to alleviate this problem, we need to address existing limitations and provide methods that

i. improve the learning of low-dimensional representations and the data-generating process of high-dimensional data, and

ii. facilitate their usability in order to lower the barrier for data-driven research.

We aimed to accomplish these goals by developing a method that

i. is effective for high-dimensional datasets of potentially small sample size,

ii. provides complex latent distributions for structured data, and

iii. is simpler and more robust in its optimization.

Once these aims were accomplished, we wished to apply the resulting method to single-cell datasets. We believe such a method can create value in this field by providing more informative representations and models that support various downstream tasks. This could lead to new insights into driver genes and pathways associated with disease, regulatory mechanisms, and prediction of genetic perturbations and disease trajectories.

5 | Contributions

The manuscripts presented in this work address the outlined objectives in chronological order. First, we developed the foundation of our approach in a deterministic setting by demonstrating the power of a decoder neural network (NN) from a manifold perspective (Paper I). Secondly, we expanded into probabilistic representations to form our main methodology: the Deep Generative Decoder (DGD) (Paper II). This paper also contains an application to single-cell expression data. Building on this, the third and last manuscript of this thesis extends the method to include multiple single-cell modalities and datasets (Paper III). It also showcases an example of the DGD's potential contributions to improve downstream data analysis. The first section describes the theoretical principles of the DGD and the motivation behind it.

5.1 Theoretical Foundation

As with NNs, there likely exist multiple equally good representations $\mathbf{z}$ for any given dataset $\mathbf{x}$, deeply intertwined with the parameter estimates θ^*. Permutations

$$f_{\theta_1^*}(\mathbf{z}_1) = \cdots = f_{\theta_n^*}(\mathbf{z}_n) \tag{5.1}$$

present equally good solutions to the approximation of $\mathbf{x}$. With representations being part of these permutations, we can view them as another set of parameters. This changes the decomposition of the joint probability of data and parameters

$$p(\mathbf{x}, \mathbf{z}, \theta) = \begin{cases} p_\theta(\mathbf{x}, \mathbf{z})\, p(\theta) & \text{for } \mathbf{z} \text{ as part of the data} \\ p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, p(\theta) & \text{for } \mathbf{z} \text{ as parameters.} \end{cases} \tag{5.2}$$

In variational inference (VI) [38, 39], representations are seen as part of the full data and are generated by an encoder, which leads to intractability [38].[1] Treating representations as data-independent parameters and learning them directly bypasses this problem. In order to find expressive approximations of the underlying generative variable Z, we introduce a prior over the representation space with parameters ϕ. Because this prior is not

[1] Discussed briefly in Methodology Section 3.2.

uniform, our optimization scheme becomes a maximum *a posteriori* (MAP) estimation.[2] The generative process can be described as a chain of two processes

$$\begin{aligned} &\text{A) } \mathbf{x} \sim p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z}) \\ &\text{B) } \mathbf{z} \sim p(\mathbf{z}, \phi) = p_\phi(\mathbf{z})\, p(\phi) \end{aligned} \tag{5.3}$$

and the overall objective becomes the log joint probability [98]

$$\log p(\mathbf{x}, \mathbf{z}, \theta, \phi) = \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\phi(\mathbf{z}) + \log p(\phi)\ . \tag{5.4}$$

With representations being learned as parameters, we no longer need an encoder network as is typical in other approaches, and lose no flexibility in architectural choices for the decoder. This theory presents the foundation for the development of the DGD and all applications presented in this work.

5.2 Paper I

A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder

Status: Published - Entropy (2021)

In our first work, we present an alternative formulation to encoder-less representation learning, closely connected to alternating back-propagation (ABP) [22] and Generative Latent Optimization (GLO) [25]. In these three works inspired by manifold learning, representations are learned as trainable parameters, removing the need for an inference network (the encoder) [22, 25, 99]. This is achieved by creating them as a look-up table (sometimes called a *codebook*) with each data point $\mathbf{x}_i$ receiving an assigned representation vector $\mathbf{z}_i$ (Figure 5.1). Our encoder-less model differs from ABP in the optimization procedure, using faster gradient-based optimization instead of sampling-based approaches [22]. In addition, it is more general than GLO, because it does not assume any specific topology of the representation space. GLO typically assumes the representations to lie on the unit sphere and requires remapping after each update [25]. In this work our model is a deterministic one, meaning that the representations are not approximated by any distribution and cannot be sampled (as is the case for Gradient Origin Networks (GONs)). However, our proposed method is simpler than ABP or GLO, and more expressive than GONs [23]. This makes our approach a solid basis for later work.

Unlike these other minimalist encoder-less latent variable models (LVMs) focusing primarily on the reconstruction and generation of image data [22, 25], we additionally explored the effect of encoder-less training on data efficiency and representation quality to a small

[2] Introduced in Methodology Section 3.1.

extent. Without an encoder, fewer data points are needed to learn a mapping between low-dimensional representation and data space [99]. The corresponding representations as a set of learnable parameters are furthermore no longer restricted by the complexity of the encoder. On simulated data, we have demonstrated that our representations are capable of recovering underlying variables of the generative process [99]. As a solution to the slower inference step in our approach (compared to autoencoders (AEs)), we propose that an encoder can be trained *post hoc*. This leads to both fast inference as well as higher-quality representations and is nearly as data-efficient as representations and decoder alone [99].

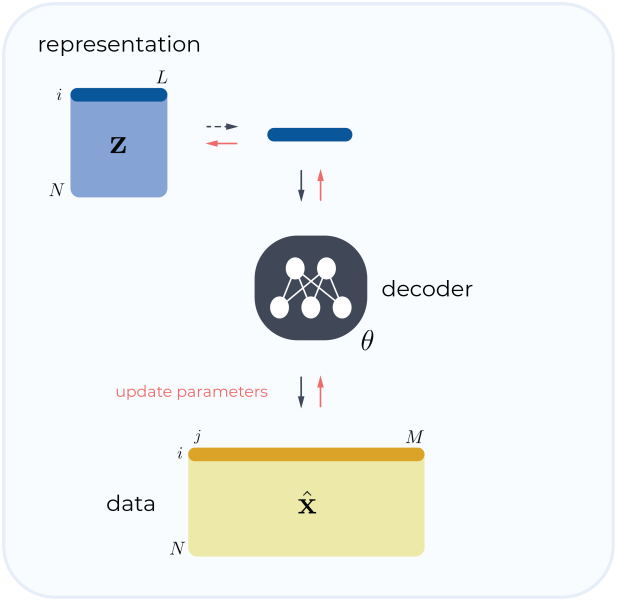


Figure 5.1: Schematic of the deterministic prequel to the DGD from paper I. The representations are taken from an $N \times L$ parameter tensor, the codebook, and fed into the decoder according to sample index i. N presents the number of samples, L the dimensionality of the representation space, and M the dimensionality of data $\mathbf{x}$. The data is reconstructed from the decoder predictions. Parameters ($\mathbf{z}$ and θ) are updated according to the gradients derived from the reconstruction loss.

5.3 Paper II

The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data

Status: Published - Bioinformatics (2023)

After having laid the foundation for our approach with the simple deterministic case, we extended it to the probabilistic formulation by incorporating a prior distribution over representation space. Including a prior distribution brings several advantages. Firstly, it regularizes the representations, potentially providing improved generalization. Secondly, it opens up new possibilities in terms of applications as a generative model.

In the Deep Generative Decoder (DGD), as we call our approach, representations remain sample-wise point estimates [98]. A Gaussian mixture model (GMM) is selected as a prior over representation space (Figure 5.2) for two reasons. A GMM creates a more complex and expressive distribution compared to a multivariate Gaussian. Additionally, a more structured distribution fits our assumptions about relationships in single-cell data. Since we still treat representations as point estimates, we do not need variational approximations or bounds to the marginal log likelihood. We instead compute the joint probability of data and parameters exactly and perform MAP estimation [98]. With the first version of this manuscript, we experienced that this objective is much more robust than a simple variational autoencoder (VAE) when it comes to model search through hyperparameter optimization [100]. This first version, however, was limited to image data and insufficient in terms of contributions to the research field, as classic VAEs no longer represent the state-of-the-art in image generation.

We thus shifted focus towards investigating the effects of the encoder and complexity of the prior, and extended our work to include an application to single-cell gene expression data. Removing the encoder in the VAE (resulting in the Variational Auto-Decoder (VAD) [24]) yielded similar results compared to the DGD with a standard Gaussian as a non-parametric prior [98]. This showed that the encoder can have a negative impact on performance. We demonstrated that on both image and gene expression data, a more complex prior improved data likelihoods and interpretability of the learned latent representations [98]. Additionally, the single-cell gene expression application achieved superior performance compared to the industry standard and one of the best-performing methods at that time [75], scVI [73], in both count modeling as well as representation clustering, especially for small datasets [98]. We provided this application as a Python package called scDGD.

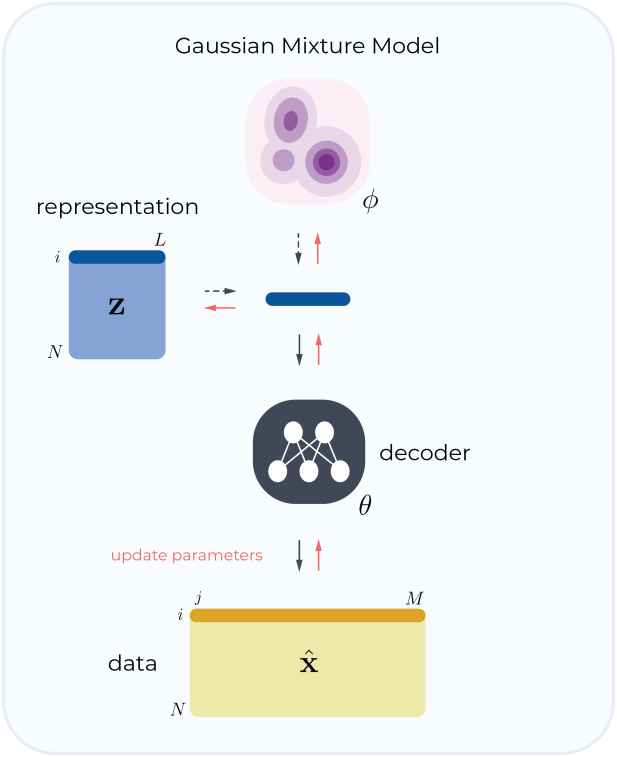


Figure 5.2: Generalized schematic of the Deep Generative Decoder from paper II. The design builds upon the deterministic model in paper I. The representations $\mathbf{z}$ of shape $N \times L$ (number of samples times latent dimensionality) are fed into the decoder according to sample index i. M presents the dimensionality of data $\mathbf{x}$. The model is extended by the addition of a prior over representation space. This is here shown as the Gaussian Mixture Model with parameters ϕ. The data is reconstructed from the decoder predictions. Losses are computed according to the objective given in Equation 5.4.

5.4 Paper III

multiDGD: A versatile deep generative model for multi-omics data

Status: Under review - Nature Methods
Preprint available on bioRxiv (2023)

The DGD should excel at tasks in which the number of data features is very large compared to the number of samples because a larger data space typically increases the number of NN parameters to train. In theory, this makes the DGD perfectly suited for high-dimensional multi-omics data. Including multiple modalities adds complexity to the choice of architecture. Modalities could be modeled in multiple ways, from using completely separate decoders with only the representation space being shared, to joining the modalities into one feature space and utilizing a single decoder. We decided on a hierarchical setup with a partially shared decoder and separate output networks [101]. This presents a parameter-efficient solution with enough individual capacity for each modality (Figure 5.3B). Here, the reduction in parameters becomes especially apparent as we do not need multiple encoders like comparable models [91, 102, 103, 104, 105, 106, 107]. Our output networks return parameters of modality-specific distributions for each feature. Because we have worked with gene expression and chromatin accessibility data, both modalities were modeled as Negative Binomials [101].[3]

An important contribution of this manuscript is that the model can incorporate multiple latent models (Figure 5.3A). A latent model here refers to a set of learnable representations and a corresponding GMM. Multiple latent models can help disentangle the representation space. Disentangled representations separate different sources of variation in the data, which can make them more interpretable [15]. To that effect, we retained the unsupervised latent model introduced in paper II as the *basal* model and added a supervised latent model [101]. This supervised latent model is used to disentangle confounding factors, here referred to as *covariates*, from the basal representation. We demonstrated this for two exemplary covariates: different sites data had been processed at (leading to technical variations called *batch effects*), and developmental stages of mouse embryos [101]. Modeling batch effects this way preserves information on the covariate and enables the integration of new batches into pre-trained models [101]. Integration of unseen covariates previously had to be done with architectural surgery and fine-tuning [108].

Aside from having demonstrated state-of-the-art performance on the reconstruction and clustering of gene expression and chromatin accessibility data [101], we also introduced preliminary results on emergent properties of multiDGD. By performing *in silico* perturbations on selected genes (Figure 5.3C), we were able to identify potential regulatory regions, of which some could be validated by experimental data [101]. This investigation is promising, as it demonstrates new horizons for the usability of powerful generative models in this field.

[3] See Methodology Section 3.5.

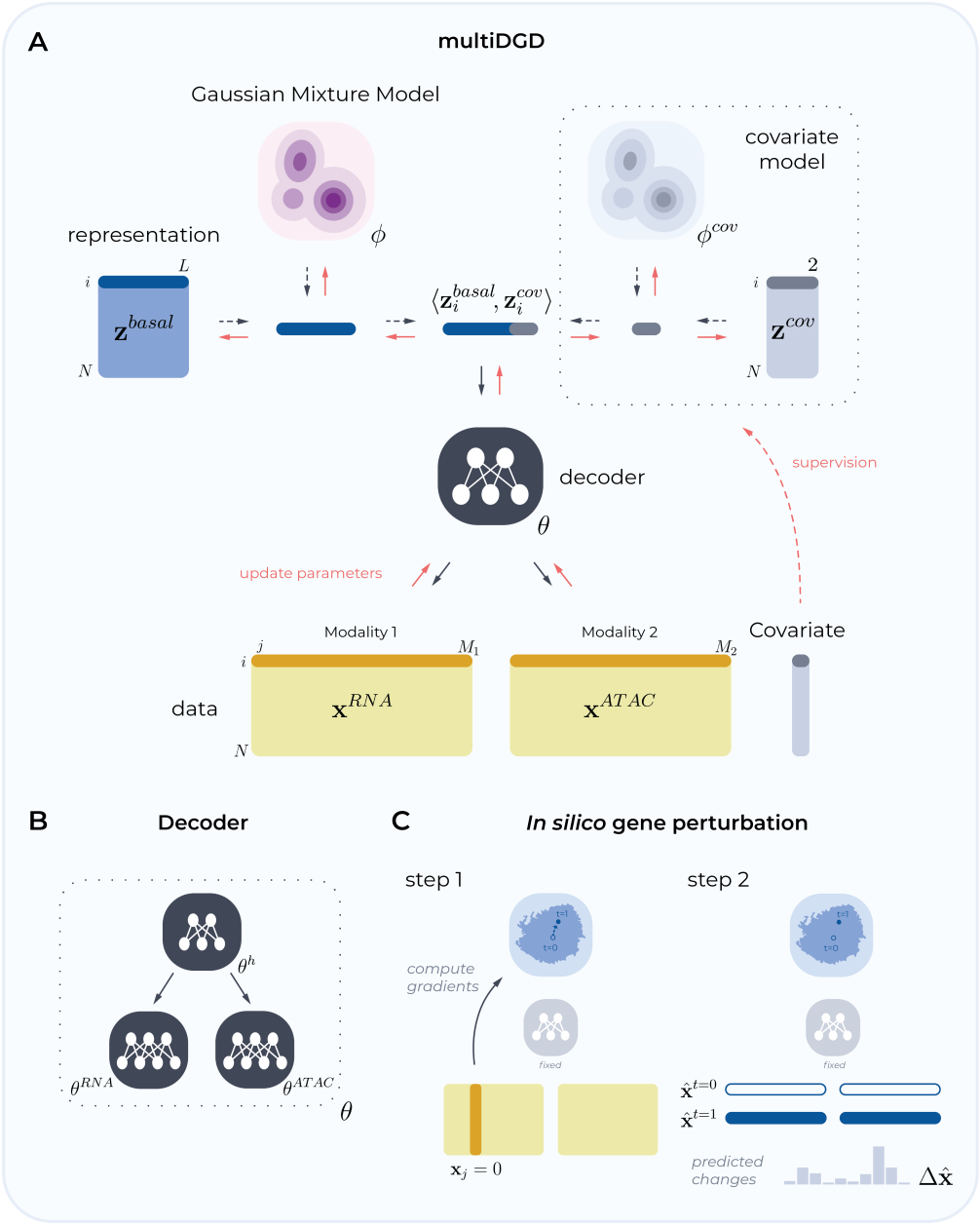


Figure 5.3: Structure of multiDGD and its functionalities from paper III. multiDGD is an extension of the DGD that can model multiple single-cell modalities in data space and covariates in representation space. This figure was adapted from figures in paper III [101]. For reoccurring notation, see Figures 5.1 or 5.2. **A** Besides the usual unsupervised representation ($\mathbf{z}^{basal}$), multiDGD models other covariates in a supervised manner. For this purpose, the representation space is expanded by another latent model (representation-GMM pair, ($\mathbf{z}^{cov}$, ϕ^{cov})). These parameters are estimated by assigning one GMM component to each category in the defined covariate. **B** In order to model multiple modalities, we created a hierarchical decoder. It consists of a shared network (θ^h) with the representation as input, and subsequent modality-specific networks (θ^{RNA}, θ^{ATAC}) that take the output of (θ^h) as input. **C** Sketch of the two-step gene-to-peak functionality via *in silico* gene perturbations. Firstly, a single step ($t = 1$) in representation space is taken with respect to gradients computed from silencing a given gene $\mathbf{x}_j$. Secondly, sample-specific changes in chromatin accessibility are computed as a difference between original ($t = 0$) and perturbed ($t = 1$) sample.

6 | Discussion

6.1 Accomplishing the Goals of the PhD Project

6.1.1 Suitability for small high-dimensional Datasets

In all three manuscripts, we have (more or less extensively) demonstrated that the Deep Generative Decoder (DGD) is more data-efficient than comparable variational autoencoder (VAE) architectures for small datasets [98, 99, 101]. In the experimental setups so far, the VAE's reconstruction performance was preferable in the limit of very large datasets for similar architectures [98]. The DGD, however, could enable the use of larger decoders, thus shifting this relationship and allowing for much better performance. It is also important to consider alternative origins to the under-performance of VAE-based implementations for single-cell data as observed for scVI [98] and MultiVI [101]. In the cases in which the number of samples is small, under-performance may arise from the common and more traditional practice of keeping decoders and encoders small, to avoid overfitting and posterior collapse [109].[1] Realistically, the DGD's applicability has to be considered and placed into perspective with every new task. For smaller single-cell datasets with many features, however, the DGD has demonstrated robust performance preferable to other deep learning methods.

When it comes to experimental design, there are of course more methods than those we included in our comparisons. For example, in all our work we have mainly focused on the comparison to variational inference (VI). This was motivated by the prevalence of VAEs in biological applications, where we see a major use case for the DGD. Our comparisons to VAEs could also have been more extensive. In our investigation into the performance of DGD, Variational Auto-Decoder (VAD) and VAE in paper II, we only compare the three methods for the case in which the prior is a standard Gaussian, but do not compare Gaussian mixture VAEs directly to the fully realized DGD. In the application to single-cell transcriptomics, we indirectly compared scDGD to a Gaussian mixture VAE through scVI performance [98]. Still, including such a comparison along with a Gaussian mixture VAD implementation would have been a valuable investigation. Some of our conclusions draw from the fact that modeling is improved when removing the encoder and adding a more complex prior, which could have been done for the VAD as well. This leaves us

[1] Posterior collapse happens when the variational distribution matches the (uninformative) prior [109].

with the question of whether a Gaussian mixture VAD would not perform equivalently to or better than the DGD.

6.1.2 Inclusion of complex Priors for expressive Representations

We have made it straightforward to include theoretically any (differentiable) prior in our generative model. In practice, we have implemented a trainable Gaussian mixture model (GMM). The GMM has been helpful in providing intrinsic unsupervised clustering in the representation space of gene expression and multi-omics data [98, 101]. Whether Gaussians with their exponential shape and high-density areas around the means are the best choice for a base distribution is something we are aiming to look into. The reason for this is that representations may be more easily stuck in local minima, and the structure may not be ideal to account for intermediary states between cell types in differentiation. Another aspect we need to address is the initialization of GMM component means. We suspect that this random initialization may be a source of strong clustering variability seen in manuscript III [101].

From the learned representations we could see that "smart" initialization choices in representation space are crucial to model performance, representation clustering, and training time. Based on unpublished experiments on image data, it became clear that initializing at origin (all zero vectors) rather than using random samples or dimensionality reduction methods presented the most unbiased and robust choice.

6.1.3 Simplicity and Robustness

We have produced results suggesting that the DGD's training performance is less susceptible to the choice of hyperparameters than the VAE [100]. However, we have so far only applied the DGD to a handful of data types. The optimization behavior could change given strongly different data, with for example much fewer features. This could have a large effect since the DGD is technically a hybrid model, whose objectives can be difficult to balance. In this work, however, hyperparameters and training scheduling translated well from the image domain to single-cell count data and required only small changes.

A known limitation of the DGD is that due to the lack of an encoder, inference of new data points is not as efficient. We have to perform a few steps of optimization on the new latent vectors. Depending on the application, however, an encoder can be trained *post hoc* to alleviate this limitation. This was tested in the deterministic case and resulted in a better overall performance than a traditional autoencoder (AE) [99]. We intended to provide a model that is both simpler to understand and to use. We have been able to get rid of the choice of encoder architecture, the often difficult-to-handle Kullback-Leibler (KL) divergence, and the related fixes to training scheduling. As discussed above, there are some caveats to the simplicity of the DGD. Most of them, like balancing objectives during training and integrating new data efficiently, have been solved for the tasks investigated in this work.

6.2 Limitations of scDGD and multiDGD

The modeling of count data is commonly achieved through parameterized Negative Binomial distributions [97]. Using the cell-specific library size as a scaling factor has been a popular choice combined with Softmax activation of the neural network (NN) output. It also outperformed other scaling methods in our investigations. Library size scaling can be interpreted as modeling the fraction of total expression contributed by each gene. From our perspective, however, the modeling of count data requires more thorough investigation. In our experiments on *in silico* perturbations with multiDGD, the parameterized Negative Binomial with library size scaling has revealed itself to be prohibitive, as it does not allow the total expression of the cell to change. One option for adjustment could be to model the library size as well. However, this is not our only concern regarding the way we model count data. It initially seemed logical to use the probability mass of a feature-specific Negative Binomial distribution as the reconstruction objective by accounting for knowledge about the general statistics of expression or accessibility counts. This means that we learned a feature-specific dispersion factor r_j and predicted scaled means $\mu_{i,j}$ for each feature j and cell i. For features with generally low dispersion factors (meaning that the mode of the distribution is approaching zero), this leads to underestimation of the predicted counts. These kinds of count patterns present the majority of the features in single-cell gene expression data, and practically all in chromatin accessibility data. We plan to investigate this behavior and potential alternatives in future work.

We have also encountered issues in learning representations with multiDGD from unpaired data. Unlike MultiVI [91] and similar models [103, 104, 105, 106, 107], we do not learn modality-specific representations that are averaged or annealed before decoding. This would encourage representations from single modalities to be close together. In order to alleviate this issue, we are planning to investigate training schemes that improve the integration of unpaired data, such as augmentation through masking [110], including distance terms, or adversarial training [104].

7 | Conclusion and Perspectives

Altogether, we have developed a generative modeling approach that is data-efficient and well-suited for high-dimensional biological data. On the tasks and data types included in this PhD project, the Deep Generative Decoder (DGD) has demonstrated its potential to be more robust in model selection and training than comparable variational autoencoders (VAEs), thanks to the direct estimation of representations and the resulting training objective. We have shown its usefulness through improved data modeling and representation clustering for single-cell expression and multi-omics data. Besides advances in performance and covariate modeling, we have demonstrated the DGD's potential to capture biologically relevant relationships from the data. We are particularly interested to see how much regulatory information we can uncover from multiDGD models and what types of regulatory mechanisms are detected. We further believe that the disentanglement of latent representations can be extended to many other data aspects, such as disease states. These and other potential functionalities of the DGD may play a critical role in future single-cell data analysis and streamlining computational workflows.

The DGD could additionally have the potential to provide higher-quality representations, although this has not been properly studied. Investigating the quality and interpretability of representations in the DGD is part of our objectives for future research. In my personal opinion, interpretability and explainability are immensely important for machine learning in biology and medicine. Models that cannot explain their decisions will (or at least should) not be deployed into real-world applications where they affect people's lives. For this purpose, I believe it is necessary to gain expertise in this field, which I hope to do in the future.

Bibliography

[1] R. Gozalo-Brizuela and E. C. Garrido-Merchan, "ChatGPT is not all you need. A State of the Art Review of large Generative AI models," Jan. 2023, arXiv:2301.04655 [cs].

[2] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018.

[3] Z. Zhao, J. C. Ye, and Y. Bresler, "Generative Models for Inverse Imaging Problems: From mathematical foundations to physics-driven applications," *IEEE Signal Processing Magazine*, vol. 40, no. 1, pp. 148–163, Jan. 2023.

[4] Z. Guo, J. Liu, Y. Wang, M. Chen, D. Wang, D. Xu, and J. Cheng, "Diffusion Models in Bioinformatics: A New Wave of Deep Learning Revolution in Action," Feb. 2023, arXiv:2302.10907 [cs, q-bio].

[5] C. Bilodeau, W. Jin, T. Jaakkola, R. Barzilay, and K. F. Jensen, "Generative models for molecular discovery: Recent advances and challenges," *WIREs Computational Molecular Science*, vol. 12, no. 5, p. e1608, 2022.

[6] D. M. Anstine and O. Isayev, "Generative Models as an Emerging Paradigm in the Chemical Sciences," *Journal of the American Chemical Society*, vol. 145, no. 16, pp. 8736–8750, Apr. 2023.

[7] B. Baillif, J. Cole, P. McCabe, and A. Bender, "Deep generative models for 3D molecular structure," *Current Opinion in Structural Biology*, vol. 80, p. 102566, Jun. 2023.

[8] A. Sharma and L. Palaniappan, "Improving diversity in medical research," *Nature Reviews Disease Primers*, vol. 7, no. 1, pp. 1–2, Oct. 2021.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[10] X. Amatriain, A. Sankar, J. Bing, P. K. Bodigutla, T. J. Hazen, and M. Kazi, "Transformer models: an introduction and catalog," Feb. 2023.

[11] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, Nov. 2022.

[12] J. M. Tomczak, "Why Deep Generative Modeling?" in *Deep Generative Modeling*, J. M. Tomczak, Ed. Cham: Springer International Publishing, 2022, pp. 1–12.

[13] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," in *Advances in Neural Information Processing Systems*, vol. 13. MIT Press, 2000.

[14] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[16] J. M. Tomczak, "Latent Variable Models," in *Deep Generative Modeling*, J. M. Tomczak, Ed. Cham: Springer International Publishing, 2022, pp. 57–127.

[17] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Computer Science Review*, vol. 40, p. 100378, May 2021.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[19] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2013.

[20] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic Variational Inference," *Journal of Machine Learning Research*, vol. 14, no. 40, pp. 1303–1347, 2013.

[21] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Jun. 2014, pp. 1278–1286.

[22] T. Han, Y. Lu, S.-C. Zhu, and Y. N. Wu, "Alternating back-propagation for generator network," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. San Francisco, California, USA: AAAI Press, Feb. 2017, pp. 1976–1984.

[23] S. Bond-Taylor and C. G. Willcocks, "Gradient Origin Networks," Oct. 2020.

[24] A. Zadeh, Y.-C. Lim, P. P. Liang, and L.-P. Morency, "Variational Auto-Decoder: A Method for Neural Generative Modeling from Incomplete Data," Jan. 2021, arXiv:1903.00840 [cs, stat].

[25] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam, "Optimizing the Latent Space of Generative Networks," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 600–609.

[26] N. Lawrence, "Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2003.

[27] H. Bendekgey, G. Hope, and E. B. Sudderth, "Unbiased Learning of Deep Generative Models with Structured Discrete Representations," Jun. 2023, arXiv:2306.08230 [cs, stat].

[28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," Oct. 2020.

[29] E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Communications in Mathematical Sciences*, vol. 8, no. 1, pp. 217–233, Mar. 2010.

[30] E. G. Tabak and C. V. Turner, "A Family of Nonparametric Density Estimation Algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013.

[31] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, Jun. 1976.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[33] C. Williams and C. Rasmussen, "Gaussian Processes for Regression," in *Advances in neural information processing systems 8*. Cambridge, MA, USA: MIT Press, 1996, pp. 514–520.

[34] P. Li and S. Chen, "A review on Gaussian Process Latent Variable Models," *CAAI Transactions on Intelligence Technology*, vol. 1, no. 4, pp. 366–376, Oct. 2016.

[35] F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle, "Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells," *Nature Biotechnology*, vol. 33, no. 2, pp. 155–160, Feb. 2015.

[36] B. Velten, J. M. Braunger, R. Argelaguet, D. Arnol, J. Wirbel, D. Bredikhin, G. Zeller, and O. Stegle, "Identifying temporal and spatial patterns of variation from multimodal data using MEFISTO," *Nature Methods*, vol. 19, no. 2, pp. 179–186, Feb. 2022.

[37] N. Kumasaka, R. Rostom, N. Huang, K. Polanski, K. B. Meyer, S. Patel, R. Boyd, C. Gomez, S. N. Barnett, N. I. Panousis, J. Schwartzentruber, M. Ghoussaini, P. A. Lyons, F. J. Calero-Nieto, B. Göttgens, J. L. Barnes, K. B. Worlock, M. Yoshida, M. Z. Nikoli , E. Stephenson, G. Reynolds, M. Haniffa, J. C. Marioni, O. Stegle, T. Hagai, and S. A. Teichmann, "Mapping interindividual dynamics of innate immune response at single-cell resolution," *Nature Genetics*, vol. 55, no. 6, pp. 1066–1075, Jun. 2023.

[38] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An Introduction to Variational Methods for Graphical Models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, Nov. 1999.

[39] M. J. Wainwright and M. I. Jordan, "Graphical Models, Exponential Families, and Variational Inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, Nov. 2008.

[40] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural Discrete Representation Learning," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[41] D. An, J. Xie, and P. Li, "Learning Deep Latent Variable Models by Short-Run MCMC Inference with Optimal Transport Correction," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 15 410–15 419.

[42] G. Gundersen, M. Zhang, and B. Engelhardt, "Latent variable modeling with random features," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 1333–1341.

[43] M. M. Zhang, G. W. Gundersen, and B. E. Engelhardt, "Bayesian Non-linear Latent Variable Modeling via Random Fourier Features," Jun. 2023, arXiv:2306.08352 [cs, stat].

[44] V. Lalchand, A. Ravuri, and N. D. Lawrence, "Generalised GPLVM with Stochastic Variational Inference," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. PMLR, May 2022, pp. 7841–7864.

[45] S. Aldridge and S. A. Teichmann, "Single cell transcriptomics comes of age," *Nature Communications*, vol. 11, no. 1, p. 4307, Aug. 2020.

[46] M. L. Metzker, "Sequencing technologies — the next generation," *Nature Reviews Genetics*, vol. 11, no. 1, pp. 31–46, Jan. 2010.

[47] P. V. Kharchenko, "The triumphs and limitations of computational methods for scRNA-seq," *Nature Methods*, vol. 18, no. 7, pp. 723–732, Jul. 2021.

[48] D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, N. Beerenwinkel, A. Mahfouz, L. Pinello, P. Skums, A. Stamatakis, C. S.-O. Attolini, S. Aparicio, J. Baaijens, M. Balvert, B. d. Barbanson, A. Cappuccio, G. Corleone, B. E. Dutilh, M. Florescu, V. Guryev, R. Holmer, K. Jahn, T. J. Lobo, E. M. Keizer, I. Khatri, S. M. Kielbasa, J. O. Korbel, A. M. Kozlov, T.-H. Kuo, B. P. Lelieveldt, I. I. Mandoiu, J. C. Marioni, T. Marschall, F. Mölder, A. Niknejad, L. Raczkowski, M. Reinders, J. d. Ridder, A.-E. Saliba, A. Somarakis, O. Stegle, F. J. Theis, H. Yang, A. Zelikovsky, A. C. McHardy, B. J. Raphael, S. P. Shah, and A. Schönhuth, "Eleven grand challenges in single-cell data science," *Genome Biology*, vol. 21, no. 1, p. 31, Feb. 2020.

[49] L. Mazutis, J. Gilbert, W. L. Ung, D. A. Weitz, A. D. Griffiths, and J. A. Heyman, "Single-cell analysis and sorting using droplet-based microfluidics," *Nature Protocols*, vol. 8, no. 5, pp. 870–891, May 2013.

[50] 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*, vol. 161, no. 5, pp. 1202–1214, May 2015.

[51] 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*, vol. 161, no. 5, pp. 1187–1201, May 2015.

[52] V. Svensson, R. Vento-Tormo, and S. A. Teichmann, "Exponential scaling of single-cell RNA-seq in the past decade," *Nature Protocols*, vol. 13, no. 4, pp. 599–604, Apr. 2018.

[53] S. Ogbeide, F. Giannese, L. Mincarelli, and I. C. Macaulay, "Into the multiverse: advances in single-cell multiomic profiling," *Trends in Genetics*, vol. 38, no. 8, pp. 831–843, Aug. 2022.

[54] F. Ginhoux, A. Yalin, C. A. Dutertre, and I. Amit, "Single-cell immunology: Past, present, and future," *Immunity*, vol. 55, no. 3, pp. 393–404, Mar. 2022.

[55] L. Heumos, A. C. Schaar, C. Lance, A. Litinetskaya, F. Drost, L. Zappia, M. D. Lücken, D. C. Strobl, J. Henao, F. Curion, H. B. Schiller, and F. J. Theis, "Best practices for single-cell analysis across modalities," *Nature Reviews Genetics*, vol. 24, no. 8, pp. 550–572, Aug. 2023.

[56] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf, "Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position," *Nature Methods*, vol. 10, no. 12, pp. 1213–1218, Dec. 2013.

[57] S. Preissl, K. J. Gaulton, and B. Ren, "Characterizing cis-regulatory elements using single-cell epigenomics," *Nature Reviews Genetics*, vol. 24, no. 1, pp. 21–43, Jan. 2023.

[58] F. Yan, D. R. Powell, D. J. Curtis, and N. C. Wong, "From reads to insight: a hitchhiker's guide to ATAC-seq data analysis," *Genome Biology*, vol. 21, no. 1, p. 22, Feb. 2020.

[59] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, H. Clevers, B. Deplancke, I. Dunham, J. Eberwine, R. Eils, W. Enard, A. Farmer, L. Fugger, B. Göttgens, N. Hacohen, M. Haniffa, M. Hemberg, S. Kim, P. Klenerman, A. Kriegstein, E. Lein, S. Linnarsson, E. Lundberg, J. Lundeberg, P. Majumder, J. C. Marioni, M. Merad, M. Mhlanga, M. Nawijn, M. Netea, G. Nolan, D. Pe'er, A. Phillipakis, C. P. Ponting, S. Quake, W. Reik, O. Rozenblatt-Rosen, J. Sanes, R. Satija, T. N. Schumacher, A. Shalek, E. Shapiro, P. Sharma, J. W. Shin, O. Stegle, M. Stratton, M. J. T. Stubbington, F. J. Theis, M. Uhlen, A. van Oudenaarden, A. Wagner, F. Watt, J. Weissman, B. Wold, R. Xavier, N. Yosef, and Human Cell Atlas Meeting Participants, "The Human Cell Atlas," *eLife*, vol. 6, p. e27041, Dec. 2017.

[60] D. Morgan and V. Tergaonkar, "Unraveling B cell trajectories at single cell resolution," *Trends in Immunology*, vol. 43, no. 3, pp. 210–229, Mar. 2022.

[61] R. Elmentaite, C. Domínguez Conde, L. Yang, and S. A. Teichmann, "Single-cell atlases: shared and tissue-specific cell types across human organs," *Nature Reviews Genetics*, vol. 23, no. 7, pp. 395–410, Jul. 2022.

[62] M. Tamaddon, M. Azimzadeh, P. Gifani, and S. M. Tavangar, "Single-cell transcriptome analysis for cancer and biology of the pancreas: A review on recent progress," *Frontiers in Genetics*, vol. 14, 2023.

[63] R. K. Perez, M. G. Gordon, M. Subramaniam, M. C. Kim, G. C. Hartoularos, S. Targ, Y. Sun, A. Ogorodnikov, R. Bueno, A. Lu, M. Thompson, N. Rappoport, A. Dahl, C. M. Lanata, M. Matloubian, L. Maliskova, S. S. Kwek, T. Li, M. Slyper, J. Waldman, D. Dionne, O. Rozenblatt-Rosen, L. Fong, M. Dall'Era, B. Balliu, A. Regev, J. Yazdany, L. A. Criswell, N. Zaitlen, and C. J. Ye, "Single-cell RNA-seq reveals cell type–specific molecular and genetic associations to lupus," *Science*, vol. 376, no. 6589, p. eabf1970, Apr. 2022.

[64] J. Ding, X. Adiconis, S. K. Simmons, M. S. Kowalczyk, C. C. Hession, N. D. Marjanovic, T. K. Hughes, M. H. Wadsworth, T. Burks, L. T. Nguyen, J. Y. H. Kwon, B. Barak, W. Ge, A. J. Kedaigle, S. Carroll, S. Li, N. Hacohen, O. Rozenblatt-Rosen, A. K. Shalek, A.-C. Villani, A. Regev, and J. Z. Levin, "Systematic comparison of single-cell and single-nucleus RNA-sequencing methods," *Nature Biotechnology*, vol. 38, no. 6, pp. 737–746, Jun. 2020.

[65] L. Zappia and F. J. Theis, "Over 1000 tools reveal trends in the single-cell RNA-seq analysis landscape," *Genome Biology*, vol. 22, no. 1, p. 301, Oct. 2021.

[66] I. Virshup, D. Bredikhin, L. Heumos, G. Palla, G. Sturm, A. Gayoso, I. Kats, M. Koutrouli, B. Berger, D. Pe'er, A. Regev, S. A. Teichmann, F. Finotello, F. A. Wolf, N. Yosef, O. Stegle, and F. J. Theis, "The scverse project provides a computational ecosystem for single-cell omics data analysis," *Nature Biotechnology*, vol. 41, no. 5, pp. 604–606, May 2023.

[67] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang, "Bioconductor: open software development for computational biology and bioinformatics," *Genome Biology*, vol. 5, no. 10, p. R80, Sep. 2004.

[68] R. Argelaguet, A. S. E. Cuomo, O. Stegle, and J. C. Marioni, "Computational principles and challenges in single-cell data integration," *Nature Biotechnology*, vol. 39, no. 10, pp. 1202–1215, Oct. 2021.

[69] P. Rautenstrauch, A. H. C. Vlot, S. Saran, and U. Ohler, "Intricacies of single-cell multi-omics data integration," *Trends in Genetics*, vol. 38, no. 2, pp. 128–139, Feb. 2022.

[70] S. Stanojevic, Y. Li, A. Ristivojevic, and L. X. Garmire, "Computational Methods for Single-cell Multi-omics Integration and Alignment," *Genomics, Proteomics & Bioinformatics*, vol. 20, no. 5, pp. 836–849, Oct. 2022.

[71] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, vol. 12, no. 7, p. 878, Jul. 2016.

[72] 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," *Molecular Systems Biology*, vol. 17, no. 1, p. e9620, Jan. 2021.

[73] R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef, "Deep generative modeling for single-cell transcriptomics," *Nature Methods*, vol. 15, no. 12, pp. 1053–1058, Dec. 2018.

[74] B. Hie, B. Bryson, and B. Berger, "Efficient integration of heterogeneous single-cell transcriptomes using Scanorama," *Nature Biotechnology*, vol. 37, no. 6, pp. 685–691, Jun. 2019.

[75] M. D. Luecken, M. Büttner, K. Chaichoompu, A. Danese, M. Interlandi, M. F. Mueller, D. C. Strobl, L. Zappia, M. Dugas, M. Colomé-Tatché, and F. J. Theis, "Benchmarking atlas-level data integration in single-cell genomics," *Nature Methods*, vol. 19, no. 1, pp. 41–50, Jan. 2022.

[76] T. Chari and L. Pachter, "The specious art of single-cell genomics," *PLOS Computational Biology*, vol. 19, no. 8, p. e1011288, Aug. 2023.

[77] K. P. Murphy, *Probabilistic Machine Learning*, ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: MIT Press, 2022.

[78] J. L. Doob, "Stochastic Processes," *Chapman and Hall, London*, p. 654 pp., Nov. 1953.

[79] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised Learning with Deep Generative Models," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[80] J. Tomczak and M. Welling, "VAE with a VampPrior," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1214–1223.

[81] B. Pang, T. Han, E. Nijkamp, S.-C. Zhu, and Y. N. Wu, "Learning Latent Space Energy-Based Prior Model," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 21 994–22 008.

[82] A. Vahdat, K. Kreis, and J. Kautz, "Score-based Generative Modeling in Latent Space," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 11 287–11 302.

[83] D. P. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational Diffusion Models," Nov. 2021.

[84] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[85] A. Vahdat and J. Kautz, "NVAE: A Deep Hierarchical Variational Autoencoder," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 19 667–19 679.

[86] D. Kalatzis, D. Eklund, G. Arvanitidis, and S. Hauberg, "Variational Autoencoders with Riemannian Brownian Motion Priors," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 5053–5066.

[87] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder Variational Autoencoders," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[88] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther, "BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[89] R. Child, "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images," Oct. 2020.

[90] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush, "Semi-Amortized Variational Autoencoders," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 2678–2687.

[91] T. Ashuach, M. I. Gabitto, R. V. Koodli, G.-A. Saldi, M. I. Jordan, and N. Yosef, "MultiVI: deep generative model for the integration of multimodal data," *Nature Methods*, pp. 1–10, Jun. 2023.

[92] M. D. Luecken and F. J. Theis, "Current best practices in single-cell RNA-seq analysis: a tutorial," *Molecular Systems Biology*, vol. 15, no. 6, p. e8746, Jun. 2019.

[93] F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B. B. Tuch, A. Siddiqui, K. Lao, and M. A. Surani, "mRNA-Seq whole-transcriptome analysis of a single cell," *Nature Methods*, vol. 6, no. 5, pp. 377–382, May 2009.

[94] T. Kivioja, A. Vähärautio, K. Karlsson, M. Bonke, M. Enge, S. Linnarsson, and J. Taipale, "Counting absolute numbers of molecules using unique molecular identifiers," *Nature Methods*, vol. 9, no. 1, pp. 72–74, Jan. 2012.

[95] J. D. Buenrostro, B. Wu, H. Y. Chang, and W. J. Greenleaf, "ATAC-seq: A Method for Assaying Chromatin Accessibility Genome-Wide," *Current Protocols in Molecular Biology*, vol. 109, no. 1, pp. 21.29.1–21.29.9, 2015.

[96] 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*, vol. 348, no. 6237, pp. 910–914, May 2015.

[97] V. Svensson, "Droplet scRNA-seq is not zero-inflated," *Nature Biotechnology*, vol. 38, no. 2, pp. 147–150, Feb. 2020.

[98] V. Schuster and A. Krogh, "The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data," *Bioinformatics*, vol. 39, no. 9, p. btad497, Sep. 2023.

[99] V. Schuster and A. Krogh, "A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder," *Entropy*, vol. 23, no. 11, p. 1403, Nov. 2021.

[100] V. Schuster and A. Krogh, "The Deep Generative Decoder: MAP estimation of representations (V1)," Oct. 2021, arXiv:2110.06672 [cs] version: 1.

[101] V. Schuster, E. Dann, A. Krogh, and S. A. Teichmann, "multiDGD: A versatile deep generative model for multi-omics data," Aug. 2023, bioRxiv 2023.08.23.554420.

[102] Y. Lin, T.-Y. Wu, S. Wan, J. Y. H. Yang, W. H. Wong, and Y. X. R. Wang, "scJoint integrates atlas-scale single-cell RNA-seq and ATAC-seq data with transfer learning," *Nature Biotechnology*, vol. 40, no. 5, pp. 703–710, May 2022.

[103] S. G. Stark, J. Ficek, F. Locatello, X. Bonilla, S. Chevrier, F. Singer, Tumor Profiler Consortium, G. Rätsch, and K.-V. Lehmann, "SCIM: universal single-cell matching with unpaired feature sets," *Bioinformatics*, vol. 36, no. Supplement_2, pp. i919–i927, Dec. 2020.

[104] K. D. Yang, A. Belyaeva, S. Venkatachalapathy, K. Damodaran, A. Katcoff, A. Radhakrishnan, G. V. Shivashankar, and C. Uhler, "Multi-domain translation between single-cell imaging and sequencing data using autoencoders," *Nature Communications*, vol. 12, no. 1, p. 31, Jan. 2021.

[105] C. Zuo and L. Chen, "Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data," *Briefings in Bioinformatics*, vol. 22, no. 4, p. bbaa287, Jul. 2021.

[106] C. Zuo, H. Dai, and L. Chen, "Deep cross-omics cycle attention model for joint analysis of single-cell multi-omics data," *Bioinformatics*, vol. 37, no. 22, pp. 4091–4099, Nov. 2021.

[107] 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 Reports Methods*, vol. 1, no. 5, p. 100071, Sep. 2021.

[108] M. Lotfollahi, M. Naghipourfar, M. D. Luecken, M. Khajavi, M. Büttner, M. Wagenstetter, Z. Avsec, A. Gayoso, N. Yosef, M. Interlandi, S. Rybakov, A. V. Misharin, and F. J. Theis, "Mapping single-cell data to reference atlases by transfer learning," *Nature Biotechnology*, vol. 40, no. 1, pp. 121–130, Jan. 2022.

[109] Y. Wang, D. Blei, and J. P. Cunningham, "Posterior Collapse and Latent Variable Non-identifiability," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 5443–5455.

[110] N. B. Ipsen, P.-A. Mattei, and J. Frellsen, "not-MIWAE: Deep Generative Modelling with Missing not at Random Data," Oct. 2020.

Notation

$\mathbf{x}$, X	Data points, data variable
f	Generator function
$\mathbf{z}$, Z	Representations, latent variable
p	Probability distributions, densities and masses
i, N	Sample index, number of (data) samples
θ	Neural Network (decoder) parameters
ϕ	Prior parameters
q	Variational distribution
φ	Encoder parameters
μ	Mean
σ, σ^2	Standard deviation, variance
ϵ	(Gaussian) random noise
$\hat{\mathbf{x}}$	Reconstructed data
KL	Kullback-Leibler divergence
$\mathcal{N}$	Gaussian / Normal distribution
I	Identity matrix
$\mathcal{L}$	Objective function or loss
Θ	Parameter space
$\mathbb{E}$	Expectation
t	Time point or state
η	Step size / learning rate
∇	Nabla operator
k, K	GMM component index, number of components
π	Mixture weights
Σ	Covariance matrix
L	Dimensionality of the representation
M	Dimensionality of the data
$\mathcal{NB}$	Negative Binomial distribution
r	Dispersion factor
j	Feature index

Abbreviations

ABP	Alternating Back-propagation
AE	Autoencoder
ATAC	Assay For Transposase-Accessible Chromatin
cDNA	Complementary DNA
DGD	Deep Generative Decoder
DNA	Deoxyribonucleic Acid
ELBO	Evidence Lower Bound
GLO	Generative Latent Optimization
GMM	Gaussian Mixture Model
GON	Gradient Origin Network
GP	Gaussian Process
GP-LVM	Gaussian Process Latent Variable Model
KL	Kullback-Leibler
LVM	Latent Variable Model
MAP	Maximum *A posteriori*
MLE	Maximum Likelihood Estimation
NGS	Next-generation Sequencing
NN	Neural Network
PCA	Principal Component Analysis
PCR	Polymerase Chain Reaction
RNA	Ribonucleic Acid
RNA-seq	RNA Sequencing
scRNA-seq	Single-cell RNA Sequencing
SGD	Stochastic Gradient Descent
UMI	Unique Molecular Identifier
VAD	Variational Auto-Decoder
VAE	Variational Autoencoder
VI	Variational Inference

List of Figures

8 | Paper I

A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder

Viktoria Schuster[1] **and Anders Krogh**[1,2,*]

[1] Center for Health Data Science, University of Copenhagen, Blegdamsvej 3B, 2200 Copenhagen, Denmark
[2] Department of Computer Science, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark

* Corresponding author: Anders Krogh (akrogh@di.ku.dk)

Publication details

This paper was published in *Entropy* on 25 October 2021.

Supplementary data are available at *Entropy* online under DOI https://doi.org/10.3390/e23111403.

entropy

MDPI

Article

A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder

Viktoria Schuster [1] **and Anders Krogh** [1,2,*]

[1] Center for Health Data Science, University of Copenhagen, 2200 Copenhagen, Denmark; viktoria.schuster@sund.ku.dk
[2] Department of Computer Science, University of Copenhagen, 2100 Copenhagen, Denmark
* Correspondence: akrogh@di.ku.dk

Abstract: Autoencoders are commonly used in representation learning. They consist of an encoder and a decoder, which provide a straightforward method to map n-dimensional data in input space to a lower m-dimensional representation space and back. The decoder itself defines an m-dimensional manifold in input space. Inspired by manifold learning, we showed that the decoder can be trained on its own by learning the representations of the training samples along with the decoder weights using gradient descent. A sum-of-squares loss then corresponds to optimizing the manifold to have the smallest Euclidean distance to the training samples, and similarly for other loss functions. We derived expressions for the number of samples needed to specify the encoder and decoder and showed that the decoder generally requires much fewer training samples to be well-specified compared to the encoder. We discuss the training of autoencoders in this perspective and relate it to previous work in the field that uses noisy training examples and other types of regularization. On the natural image data sets MNIST and CIFAR10, we demonstrated that the decoder is much better suited to learn a low-dimensional representation, especially when trained on small data sets. Using simulated gene regulatory data, we further showed that the decoder alone leads to better generalization and meaningful representations. Our approach of training the decoder alone facilitates representation learning even on small data sets and can lead to improved training of autoencoders. We hope that the simple analyses presented will also contribute to an improved conceptual understanding of representation learning.

Keywords: representation learning; manifold learning; neural networks; autoencoders

Citation: Schuster, V.; Krogh, A. A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder. *Entropy* **2021**, *23*, 1403. https://doi.org/10.3390/e23111403

Academic Editors: Fabio Aiolli and Mirko Polato

Received: 31 August 2021
Accepted: 21 October 2021
Published: 25 October 2021

1. Introduction

The original article on backpropagation is called "Learning Internal Representations by Error Propagation" [1], and indeed, learning in neural networks can be viewed as the learning of intermediate representations in the different layers. The neural network thus performs a transformation of the input through a series of these internal representations. In representation learning (reviewed in [2,3]) the objective is to use these learned representations for other applications as they can for instance map discrete high-dimensional input samples like text to a Euclidian space of lower dimensionality and hopefully learn (or preserve) relatedness by assigning similar representations to related examples. Feed-forward autoencoders, whose objective is to reproduce the input on the output layer, are often used for unsupervised (or "self-supervised") representation learning, although one can of course learn representations even when inputs and targets differ.

In the related field of manifold learning, the objective is likewise to find a representation of samples. It is assumed that the high-dimensional data lie on a lower-dimensional manifold, and the aim is to construct a map of the (training) data on such a manifold. Principal component analysis (PCA) is the simplest and most used method, in which the linear subspace explaining most of the variation in the data is found. The main difference between manifold learning and representation learning is that in the latter one obtains an

encoder that maps from the input space to the representation space and a decoder that maps from the representation space back to the input space. Additionally, the objective function in manifold learning often takes neighbourhood relations into account, whereas samples in standard neural network training are treated independently. The relationship between representation learning and manifold learning has been extensively discussed in works such as [2,4,5].

Autoencoders were originally introduced in [1], with image compression [6] and speech recognition [7] as some of the first applications. In [8] it was shown that simple auto-encoding feed-forward neural networks, with a single hidden layer and optimized to reproduce real-valued input on the output, will converge to the principle components subspace in the hidden layer. Since then, there has been significant progress with respect to regularization and robustness of autoencoders, such as using noisy inputs to train denoising autoencoders [4] and making the contractive autoencoders aiming at more robust encoding [9]. The relation between autoencoders and manifold learning has been discussed before, see e.g., [4,5].

Here, we took up a simple but not too well-known view on autoencoders and representation learning from a manifold learning perspective. We showed that the decoder maps from the representation space to a manifold in the input space. We then showed how a decoder can be trained without an encoder by optimizing the representations of the training data directly together with the weights of the decoder similar to manifold learning. In training, we optimized towards a manifold that has the minimum distance between the training points and their projections onto the manifold. Learning of a representation along with an autoencoder has been introduced before as predictive sparse decompositions [10], and others have even made the step of separating representation and the decoder from the encoder [11]. We wished to take up this view on representation learning and show its benefits in terms of simplicity, performance, and data-efficiency.

We derived expressions for the number of samples needed to specify an encoder and a decoder and showed that in most situations, the decoder is much better specified than the encoder. The theoretical predictions were confirmed on three different data sets. We demonstrated how training of the decoder alone performs much better than a standard autoencoder on small data set sizes, while performance on larger data sets is still better but very close to the autoencoder. We further showed that the learned representation can be useful for downstream tasks and that specific solutions can potentially be derived better from the decoder than the autoencoder.

This study was basic and straightforward mathematically, and it is our hope that it will help readers build intuition about decoders, encoders, and autoencoders in this manifold learning perspective.

2. Encoder-Free Representation Learning in Theory

2.1. The Linear Case

Assume the data are n-dimensional real vectors. In PCA, a linear subspace is obtained in which the first basis vector points in the direction of highest variance, the second points in the direction of highest variance of the subspace orthogonal to the first, and so on. When PCA is used for manifold learning and dimensionality reduction, the first $m < n$ principal components are used, and the data points are projected onto this linear subspace of dimension m, which we call the principal subspace.

If the basis vectors of the principal subspace are called $\vec{w}_i$, the projection of a point $\vec{x}$ can be written as $\sum_{i=1}^{m} z_i \vec{w}_i$, where $\vec{z}$ is the m-dimensional representation of $\vec{x}$ and $z_i = \vec{x} \cdot \vec{w}_i$. The z vectors are analogues of the representations in representation learning or manifold learning.

If we assume that data have mean zero, the linear subspace is also the one which has the smallest mean distance to the data. Therefore, the principal subspace can be found by minimizing the mean distance between the data points and their projections,

$$\sum_k ||\vec{x}^k - \sum_{i=1}^{m} z_i^k \vec{w}_i||^2, \tag{1}$$

where k indexes the data points. (We do not give the detailed proof, but it is relatively straight-forward to expand the square in (1) and using $z_i^k = \vec{x}^k \cdot \vec{w}_i^k$ to show that minimizing (1) corresponds to maximizing the variance $\sum_{i,k}(\vec{x}^k \cdot \vec{w}_i)^2$ when requiring that the $\vec{w}_i$s are orthonormal.) This will not normally give an orthonormal basis, but vectors w will still span the principal subspace. We can recognize this as a linear "neural network" with weights w, corresponding to the decoder part of a linear autoencoder, mapping from a representation z to a point x in input space. It is a classic result that a linear autoencoder will learn the principal subspace [8].

Note that (1) does not have a unique solution. The weights and representations can be scaled arbitrarily ($wx = (w/s)(zs)$ for a constant s), and it is invariant to permutations of the order of vectors and rotations within the subspace in general. One could apply normalization or impose other constraints on the solution to limit the freedom.

2.2. Non-Linear Decoders

Assume now that we have a non-linear mapping that maps a representation z to a point $g_w(z)$ in input space (we dropped the vector arrows for ease of notation). We assume also that g_w is a continuous function that depends on some parameters w and possibly some form of regularization. We assumed $m < n$, g_w defines a manifold in input space of dimension m (or lower), onto which all points in the representation space are mapped.

Above, we saw how one can obtain the principal subspace by minimizing the distance between the data points and their projections. We can do the same for a non-linear decoder. The idea is then to find the manifold defined by g_w that minimizes the mean distance (or the loss) between the training points and their projections onto the manifold, $L(x, g_w(z))$, where L is the loss function and z and w are parameters. See Figure 1 for an illustration. (Here "projection" means the point on the manifold that we map a point to, so it is not used in a strict mathematical sense).

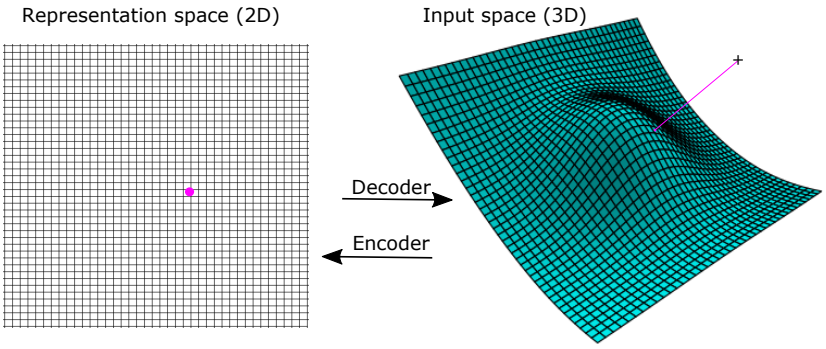


Figure 1. The decoder maps from the low-dimensional representation space to a manifold in input space. Here the representation space is 2D, and the input space is 3D. The representation of a point in input space is the point in representation space that maps to the nearest point on the manifold. Here, it is illustrated in 3D, where the point in representation space to the left maps to the point on the manifold that is closest to the point at the cross.

Usually, internal representations are learned implicitly by adapting the weights of the neural network, but there is nothing stopping us from treating them as parameters and learning them explicitly together with the weights (see e.g., [12]). A loss function $L(x, \hat{x})$ used to train an autoencoder, where $\hat{x}$ is the autoencoder output, can be used with internal

representations. The loss for input x would be $L(x, g_w(z))$, where the representation z is unknown and found together with the weights by minimizing the total loss. So, for each of N training examples, $(x^1 \dots x^N)$, the representation is optimized during training, so the total loss

$$\sum_k L(x^k, g_w(z^k)) \tag{2}$$

would be minimized with respect to both w and the vectors $z^1 \dots z^N$. This can be done by any optimization technique, such as gradient descent. Gradients with respect to z are simply the back-propagated errors that we are calculating anyways for weight optimization. As in the linear case described above, there are scaling and symmetry invariances, so the solution is not unique. It is therefore advisable to constrain the weights (for example, via a small weight decay).

If the decoder is a single layer, there is only one optimal solution (apart from scaling and symmetry operations), and the optimization problem is convex, i.e., it is very easy to learn, and it would essentially recover the PCA if the layer is fully connected. With a multi-layer decoder, one could learn the internal representations layer-by-layer using convex optimization, but this is unlikely to lead to an overall optimal solution and was not considered further here.

Once the weights of the decoder and training set representations are determined by training, how can we use it for new samples, e.g., in a test set? One possibility is to do the same optimization by gradient descent with fixed weights to find the optimal representation for the new sample. This was our approach when reporting results of the decoder alone. A disadvantage of this approach is that there may be local minima trapping the optimization, and thus it may not give a globally optimal representation. In our experiments, we did not observe problems relating to this. An alternative is to train an encoder, as we discuss later.

One disadvantage of optimizing the representations directly within the decoder training loop is that each sample only receives one gradient update per epoch, whereas the weight updates are based on gradients from the whole data set. One could therefore suspect a large number of epochs is needed for the representations, but this was not a major issue in our experiments. If the training set is very large, one could train the decoder on a sub-sample and afterwards train an autoencoder on the whole dataset (see below).

2.3. The Number of Samples Needed to Train the Decoder

In an idealized noise-free case, the decoder should fulfil $x = g_w(z)$ for all N training examples. If g is a linear function, there exists an exact solution for w and z if the number of weights C_d plus the number of parameters for the representation, Nm, is larger than the number of constraints Nn: one equation per example (k) per output unit (i), $x_i^k = g_w(z^k)$. We defined the load as the number of constraints per parameter,

$$\alpha_d = \frac{Nn}{C_d + Nm} = \frac{n}{m}\left(1 + \frac{C_d}{Nm}\right)^{-1}. \tag{3}$$

If the load is below 1, the system is under-determined, and if it is above 1, it is over-determined (if the training vectors are linearly independent).

For non-linear decoders, such as multi-layer perceptrons, we expect the above to be approximately true, although C_d should be replaced by an "effective" number of parameters, which we call the network complexity. Formally, we defined this complexity as the number of training examples needed to train a unique model apart from scaling and symmetry operations. There is currently no theory to calculate this complexity, and we approximated it by the number of weights. The number of weights is an upper bound for the effective number of parameters, and therefore, the load, as defined above, will generally be underestimated, and we are thus less likely to apply too large a model to the given data.

In most realistic situations, we would want the system to be over-determined, i.e., $\alpha_d > 1$, because otherwise the representations and the decoder would adapt to the noise (alternatively, the system can be regularized). When the number of weights is less than Nm, the load is essentially the input dimension divided by the representation dimension, the compression rate n/m, which is normally much larger than 1 when the objective is to find a low-dimensional representation of high-dimensional data. When $n >> m$, it is generally quite easy to obtain high loads. (For $m > n$, the load is always below 1 and is only relevant with sparsity constraints or other regularization.) Another way to interpret this is that you would be wise to either construct your neural network decoder to have a relatively large α_d or use some form of regularization.

We have seen that one can learn a manifold in input space using a neural network decoder, and it is therefore a form of manifold learning. The mathematical properties of the manifold are determined entirely by the neural network decoder, and the learned manifold may be further constrained by regularization. If the decoder is over-determined ($\alpha > 1$), the training data cannot normally be contained in the manifold exactly, and minimization of the loss will find the manifold that is closest to the training data (closest in the sense of having minimum loss).

2.4. The Encoder and Autoencoder

We see that in theory the encoder is not really needed for learning low-dimensional representations. However, finding the representation of a new data point x requires a minimization of $L(x, g_w(z))$ in z with fixed weights in the decoder. Although this can easily be done, it is often desired to have an encoder instead that maps directly from input space to representation space. In an autoencoder, the encoder puts additional constraints on the representations, meaning that it will not necessarily learn the same representations as the decoder would recover on its own.

In the framework of manifold learning, the encoder can only make things worse, because with the chosen parametrization of our manifold (the decoder) and the chosen distance metric (the loss function), the representations found using only the decoder are optimal—they minimize the loss and the distance to the manifold. In representation learning, one could instead view the encoder as a possibility to impose constraints on the representations as a sort of regularization. However, regardless of the encoder, reconstructions will lie on a manifold defined by the decoder.

In principle, the encoder could be of such high complexity that it could learn almost exactly the "true" decoder representations. If it had high enough complexity, we know that the "true" representations would be found, because they minimize the loss. However, when the encoder complexity is very high, it is again likely that we will not have enough data. Instead, once the representations are estimated (along with the decoder), the encoder can principally be trained using the learned representations as targets. For an encoder, the load is therefore

$$\alpha_e = \frac{mN}{C_e} \tag{4}$$

when the representations of the training examples are fixed (Nm equations/constraints on the weights). Interestingly, there is no simple relation between this load and the decoder load. If the encoder and decoder have the same complexity $C_e = C_d$ and $m < n$, we will have from (3) and (4) that

$$\alpha_d = \frac{n}{m}\left(1 + \alpha_e^{-1}\right)^{-1} = \frac{n}{m}\frac{\alpha_e}{1 + \alpha_e}. \tag{5}$$

For small α_e, α_d is n/m times larger, and when α_e is large, α_d is essentially equal to n/m. Often, the compression factor n/m is in the hundreds, and the decoder would thus be "hundreds of times more well-specified" than the encoder; in many situations, one would have plenty of examples to learn the decoder but not enough for the encoder.

The above analysis is a little simplified. Since the assumption is that the training data lie close to an m-dimensional manifold in input space, the data were not scattered randomly, and it is likely that the encoder can be trained on a smaller number of samples in this region of sample space. In other words, it may be possible to train the encoder with a small load for points in or close to the manifold. However, for outliers, an encoder trained with a small load is likely to give arbitrary results and thus the autoencoder reconstruction will be far from the projection on the manifold. We thus expect that out-of-distribution samples will be poorly reconstructed by an autoencoder compared to a decoder.

Several methods have been proposed that increase the robustness of autoencoders regarding over-fitting. One such method is the denoising autoencoder [4] in which noise is added to the input samples and in which the autoencoder is trained to reconstruct the noiseless version. In the present perspective, this seems like an excellent approach, because this will minimize the distance between noisy points and their projections. The contractive autoencoder [9] imposes regularization on the encoder that favors similar inputs to have similar representations. This is done by adding a regularizer term with the squared derivatives of the encoder with respect to the x (the Frobenius norm of the Jacobian). Zero gradients imply orthogonality with the manifold, and this approach therefore favors an orthogonal projection of points onto the manifold.

Both denoising and contractive autoencoders have the desired effect of making the encoder more robust, but they have the less desirable side-effect of also trying to minimize variation within the manifold. Therefore, we would suggest a variant of the denoising autoencoder in which the decoder is trained as above and the encoder is trained on noisy examples with its "true" representation as the target, that is, the representation defined by the decoder. This is essentially equivalent to training the autoencoder with *fixed decoder* and using the noisy examples both as input and output. This approach will train the decoder to give the correct projection of the out-of-distribution input onto the manifold learned by the decoder on the training set. It should thus be possible to use much higher noise levels than in denoising autoencoders, and in principle, one can train on completely random data, once the decoder is fixed. Care must be taken, however, to ensure that the encoder still encodes the training data well.

3. Hypothesis Testing on Real and Simulated Data

In this section, we aimed to answer the following questions: can a decoder on its own generalize well on small data set sizes, whereas an autoencoder overfits as hypothesized? Is the resulting representation meaningful/useful for downstream tasks? Does the decoder work on larger data sets?

We demonstrated that a representation can be learned by a decoder alone using three different data sets. The first two are popularly used image data sets MNIST [13] and CIFAR10 [14], and the last one is a simulated data set as an example of regulatory data with a known and desired representation.

All networks were implemented in Python using PyTorch [15], and Jupyter notebooks are available with the code. Most runs were done on Google Colaboratory. Details about model architectures for all experiments covered in the following sections can be found in the Appendix A.

3.1. MNIST

In our first experiment, we used the standard MNIST data set [13] and trained on random subsets of varying sizes from 500 to 15,000 examples using a decoder on its own and a naive autoencoder. The encoder of the autoencoder consists of two convolutional layers followed by a fully connected representation layer. The convolution layers both have 2D kernels of size 4, stride 2, and 64 channels. The decoder is the reverse with the representation layer fully connected to two layers of transpose convolutions. The representation layer has size 20 with linear output. The output layer uses a sigmoid transfer function, and the other layers use a rectified linear unit (ReLU) [16,17] activation.

The loss function is the binary cross entropy and network weights were trained with the Adam optimizer[18] using a learning rate of 0.001 and a weight decay of $1.e^{-5}$. When training without the encoder, the representations were optimized with a stochastic gradient descent [19] with a learning rate of 0.02 and a momentum of 0.9. All networks were trained for 200 epochs. Networks were tested on the same random subset of 1000 examples from the MNIST test set. For the decoder alone, representations of test data were found using the same gradient descent as in training (but with fixed decoder weights, of course).

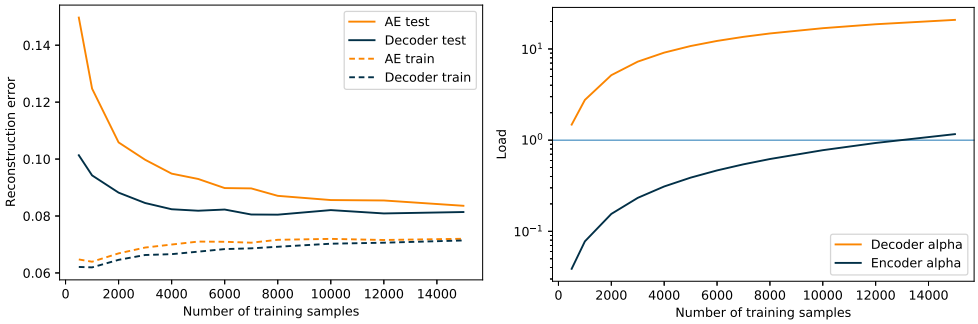


Figure 2. Left: Training and test error for a decoder and an autoencoder trained on subsamples of different sizes (x-axis) of the MNIST data. Details of the networks are given in the text. **Right**: The load of the encoder and decoder vs. the training set size.

In Figure 2, we show the training and test error for a decoder and an autoencoder trained on subsamples of the MNIST dataset. We could see that the training errors were very similar for the two models, although smallest for the decoder, as would be expected, because the autoencoder is further constrained by the encoder. For small data set sizes, both models over-fit, but the autoencoder was significantly worse than the decoder alone, which supports our hypothesis. From the other graph, we could observe that the decoder load was above 1 for almost all data set sizes, and from a load around 9 (4000 examples), the test error was stable. The load of the decoder was below 1 for all sizes except the last (15,000) and we see that the test error is still decreasing.

This example shows that although the training errors are comparable, the full autoencoder over-fits the data to a larger extend than the decoder, especially for small sample sizes. The decoder test error was almost constant from around 4000 samples.

In this experiment, we also trained an autoencoder with the trained decoder fixed. This performed much worse than the decoder and the autoencoder (results available in the supplementary notebook). Although unexpected at first, we interpreted it in the following way: the full autoencoder can find a good solution that satisfies the constraints of both the decoder and the encoder, reaching a compromise between them. When the decoder is fixed, however, the encoder is forced to learn the representations dictated by the decoder. For small data sets, the encoder can learn with a small training error but has a high test error. For larger sets, when the load of the encoder approaches 1, both the training and test error increase due to the constraints imposed by the decoder.

3.2. CIFAR10

After initial promising results from the simple task of learning a representation for MNIST, we aimed to more systematically demonstrate the efficiency and validity of our proposed approach using the more complex CIFAR10 data set [14]. Our aim was to compare the performance of a simple decoder to an autoencoder with an identical decoder and a symmetric encoder on natural image data. We trained both the single decoder and the autoencoder and compared the reconstruction capabilities of these models for different size training set sizes.

The architectures of our models are based on work from [20]. In order to find a good decoder architecture, we ran a model search whose search space we restricted based on prior knowledge derived from [20,21]. In [21], a thorough investigation of convolutional

architectures based on residual bottleneck blocks introduced by [22] was conducted, resulting in guiding principles for a limited design space of convolutional networks for image classification. The initial architecture taken from DCGAN [20] represents the generator. A supplementary notebook *DecoderTraining_CIFAR10_modelSearch.ipynb* provides a report of our model optimization and a description of how we arrived at using an altered version of the DCGAN generator as our decoder.

The decoder consisted of five 2D transposed convolutional layers with kernel size 4, stride 1, and an output layer with kernel size 1 and stride 1. The first and last transposed convolutional layers had padding 0, while all others had padding 1. For constructing channel sizes, we used a basis of 64, which we called the capacity. The representation (the input of the decoder) was a 1D vector of length 256, which was four times the capacity. This was reshaped as the decoder input to a tensor of $256 \times 1 \times 1$. The channel sizes were reduced to $capacity \times 2$ (128) in the first layer, $capacity$ (64) in the third, and 3 (output channel size) in the last layer. Batch normalization was applied between all layers, as well as ReLU activation. The last convolutional layer followed a sigmoid activation due to the normalization of the data. The corresponding autoencoder consisted of the decoder architecture and a mirrored encoder with convolutional layers instead of transposed convolutional layers. We refer to this decoder and autoencoder as decoder_1×1 and autoencoder_1×1 (or AE_1×1 in images due to limited space), respectively, to highlight the decoder's input pixel dimension.

In order to demonstrate that a decoder on its own outperforms a comparable autoencoder on small data sets, we trained decoder_1×1 and autoencoder_1×1 on three class-balanced subsets as well as the full training set (train–test split provided by the data) five times with different random seeds (see *DecoderTraining_CIFAR10.ipynb*). The resulting numbers of train images per class were 50, 100, 500, and 5000. Other training parameters included the Adam [18] optimization with weight decay $1e^{-5}$, the learning rate $1e^{-4}$, the representation learning rate $1e^{-1}$ (representations were optimized with stochastic gradient descent [19]), the mean squared error (MSE) loss, and the training time of 100 epochs. The test error was reported on a tenth of the test set (class-balanced, same set for all runs).

In Figure 3A,C we can see that the train sample size had a much lower effect on the test reconstruction error in decoder_1×1 compared to autoencoder_1×1. While both models' errors converged for the full training set size, autoencoder_1×1's test loss was approximately three times higher than that of the decoder for the smallest train sample size of 50 images per class. Figure Figure 3A shows that this difference in test loss was not a result of insufficient training time. More precisely, this figure further supports our hypothesis of the relationship between decoder and encoder load. As shown in the supplementaries (*DecoderTraining_CIFAR10.ipynb*), the decoder load was above 1 for all training subsets, while the encoder load achieved values above 1 only for the full train sets. Since mere reconstruction loss alone is not a satisfactory evaluation of a model's performance, we also trained decoder_1×1 and autoencoder_1×1 for 400 epochs each on the full train set and evaluated them on the full test set with random seed 0, saving model parameters every 50 epochs. Figure 3E shows reconstructions of the first eight test images (originals shown in Figure 3D) of both models after 50, 200, and 400 epochs. While the decoder_1×1's MSE loss for the test set was above that of autoencoder_1×1 until roughly epoch 200 and while the reconstructed images are blurry, they were much more recognizable than those of autoencoder_1×1 at any of the selected epochs. These results show that this convolutional decoder is superior to its equivalent autoencoder in reconstructing images from a low-dimensional representation for small data sets and that it can more than keep up with the autoencoder for larger data sets where the encoder load is sufficient.

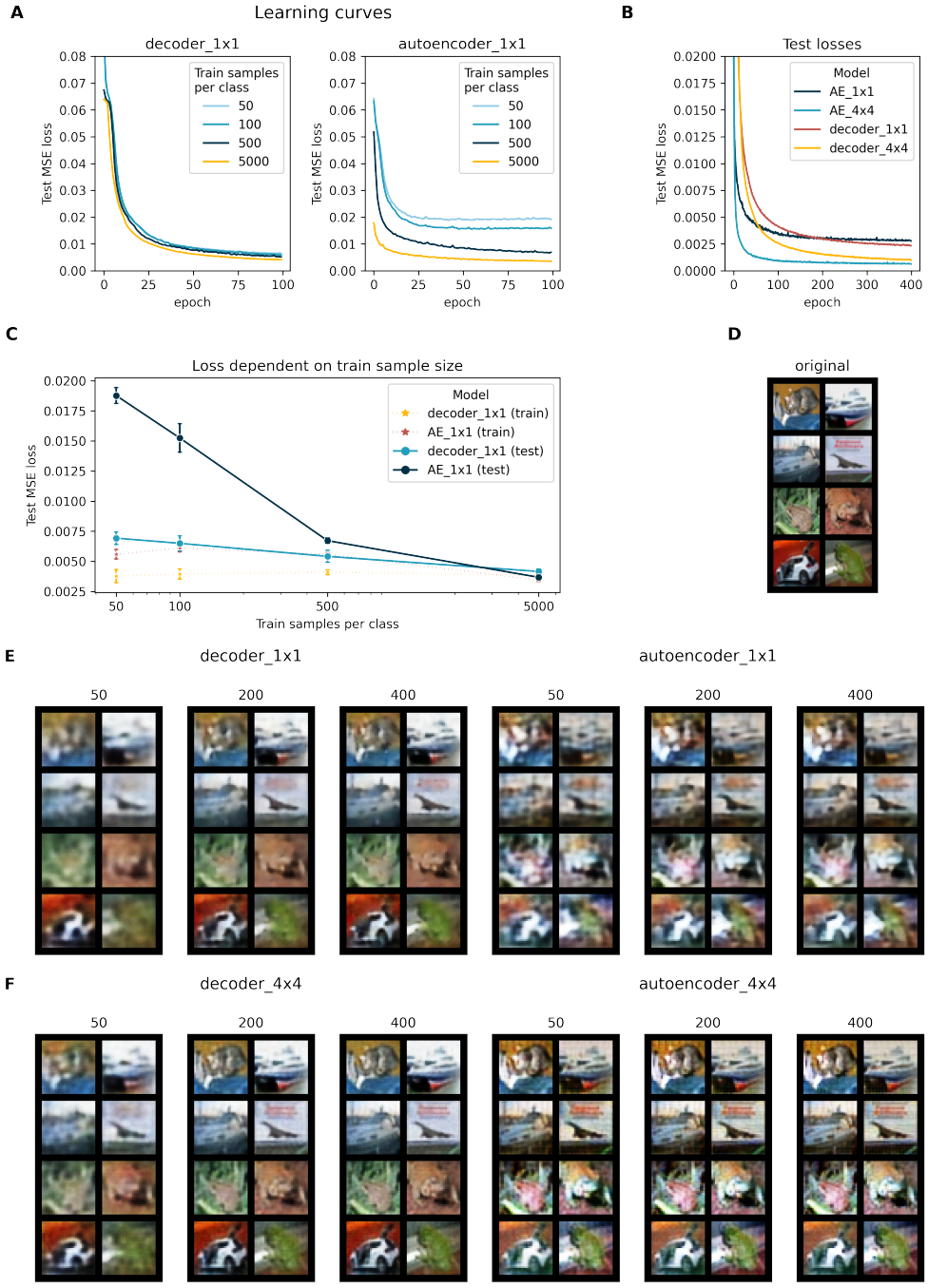


Figure 3. Results on CIFAR10. (**A**) Learning curves of decoder_1×1 and autoencoder_1×1 for different training sample sizes (indicated by color). The y axis reports test reconstruction losses as mean squared errors. All models were trained for 100 epochs and initialized with random seed 0. (**B**) Learning curves of decoder and autoencoder with different representation sizes. (**C**) Reconstruction error for decoder_1×1 and autoencoder_1×1 trained on class-balanced subsets of CIFAR10 plotted against the per-class sample size of the train set. Means and standard deviation error bars were derived from 5-fold replication with different random seeds. Dotted lines with asterisk markers refer to train loss, full lines, and round markers to the loss on the full test set. (**D**) The first 8 images of the CIFAR10 test set. (**E**) Reconstructed test images of models with representation $256 \times 1 \times 1$. Numbers above the image grids indicate the training time in epochs. The first three grids show reconstructions from decoder and representation, and the last three were those from the autoencoder. (**F**) Same as E for models with larger bottlenecks ($64 \times 4 \times 4$).

However, we are aware that designing and optimizing a decoder mapping from a low-dimensional representation and then comparing its performance to its equivalent autoencoder could be biased in favour of the decoder. We thus conducted a second experiment for which we optimized an autoencoder architecture and training parameters completely disregarding the decoder performance in the optimization step. The model development is included in supplementary notebook *DecoderTraining_CIFAR10_modelSearch.ipynb*. The architecture of the best performing autoencoder was similar to the previous architecture of autoencoder_1×1, but the last encoder and first decoder convolutional layers were removed, resulting in a representation of dimension $channel \times 4 \times 4$. We refer to these architectures as autoencoder_4×4 and decoder_4×4. The other change to the previous architecture was the number of channels in the convolutional layers. With a capacity of 16, the decoder (and symmetrically the encoder) took the representation vector of length 1024 as input of dimension $64 \times 4 \times 4$. Each layer's output channels were of size 64, 32, 16, and 3, respectively.

We trained autoencoder_4×4 and decoder_4×4 on the full train set for 400 epochs with (coincidentally) the same hyperparameters as for training 1×1 models. The test learning curves for both models are shown in Figure 3B along with those of the 1×1 models for comparison. The 4×4 learning curves show a similar but elongated behavior to those of the 1×1 models. Test losses for decoder_4×4 and autoencoder_4×4 converged to much lower levels and converged to similar losses at the end of the 400 epoch training period. This indicates an accelerated learning of autoencoder_4×4. We again reconstructed images after 50, 200, and 400 epochs, which are depicted in Figure 3F. These show an extensive improvement in the autoencoder reconstruction capabilities from autoencoder_1×1 to autoencoder_4×4. Images appeared quite sharp but showed inferior color reconstruction compared to images from decoder_4×4.

Even though the convergences for decoders trained on the full data set for 400 epochs compared to the autoencoders were slower, Figure 3D,E shows that the decoders achieved qualitatively better reconstructions, especially for lower-dimensional representations. When comparing a decoder with learned representation and an autoencoder with equivalent architecture, this clearly demonstrates that the decoder alone with learned representation poses a better solution than the autoencoder.

3.3. Simulated Data

Our last test used simple simulated, almost linear, data. They were inspired by a simplified model of a biological regulatory network that we studied in another project. The regulatory network is built of proteins Z that interact with genes X and govern their expression. These proteins are called transcription factors and present one of the main actors in gene regulation. They are sequence-specific DNA-binding proteins, which modulate expression by binding close to specific genes. We assumed for simplicity that the regulation is direct and that transcription factors operate independently. We thus assumed that the expression data are governed by the transcription factor levels, Z. Since Z are sequence-specific, they do not interact with all genes, and with those they do interact, they do not do so equally. We therefore viewed the regulatory network as a weighted bipartite graph, directed from Z to X. All this can be expressed in

$$x_i = ReLU(\sum_{j=0}^{n} a_{i,j} w_{i,j} z_j + \epsilon_i), \tag{6}$$

where $a_{i,j}$ is a sparse connectivity matrix and $w_{i,j}$ the strength (and sign) of regulation. To generate the expression vectors x of length n, regulator level vectors z of length m ($m < n$) were initialized from random samples of a gamma distribution. The weighted bipartite graph was expressed by the product of an adjacency matrix A (random binary values assigned based on a defined fraction of connections) and a weight matrix W (randomly sampled from a uniform distribution). Noise was added from a normal distribution

with mean 0 and sd 0.2. X was ensured to be $R_{(>=0)}$ through the rectified linear unit (ReLU) [16,17].

We approximated the linear relationship between the regulatory representation Z and expression $\mathbb{R}_{\geq 0}$ input space X via a single-layer decoder. Since part of the objective was to learn the input space and representation as an assignable bipartite graph, it was necessary to apply constraints enabling the hidden units in the representation to be identified as specific regulatory factors, and we therefore assumed a sparsely connected network defined by the adjacency matrix A.

Given this assignability of the representation, we were able to restrict the space of possible valid representations towards one close to the regulatory space used to generate the data. Hence, we enabled ourselves to judge whether a meaningful representation was learned that represents a unique feature of the data. As a measure, we used the Pearson correlation coefficient (PCC) between true regulatory vectors and learned representations. We investigated data reconstruction and representation correlation for different decoder loads on three models. These were the decoder, the encoder trained on the pre-trained decoder, and an autoencoder (AE) consisting of the architectures of the single models.

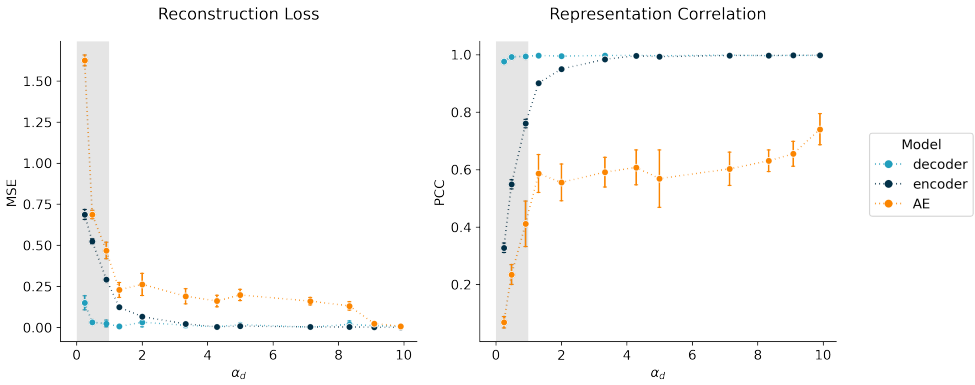


Figure 4. Performance metrics of decoder, encoder, and autoencoder for different decoder loads. Points and error bars present average losses and standard deviations over 5 replicates, respectively. Colors indicate the model type. Metrics were derived from the test data. **Left**: mean squared error (MSE) loss of the simulated data for different decoder loads. **Right**: Pearson correlation coefficients (PCCs) of true regulation dimension and model representation dimension for different decoder loads.

The decoder and the encoder both consisted of a single linear layer (sparse in case of the decoder as described above) with ReLU and leaky ReLU (slope 0.1) as activation functions, respectively. The autoencoder was comprised of a combined encoder and decoder. Different loads were achieved by varying the number of training samples N for a constant input space of dimension $n = 1000$ and hidden dimension $m = 100$. Used values for N can be found in supplementary notebook *DecoderTraining_SimulatedData.ipynb* section 2, experiment 1. The data in this experiment were simulated with zero noise and a connectivity of 0.1 (90% of values in **A** are zero). Training parameters included a mini-batch size of 32, a weight decay of $1.e^{-5}$, and a set of learning rates, which were obtained from a small-scale grid search experiment included in *DecoderTraining_SimulatedData.ipynb section 1. Learning Rate Optimization*. Decoder and representation received learning rates of 0.001 and 0.01 (with momentum 0.9), respectively. The encoder and autoencoder received a learning rate of 0.0001. Weights were optimized with the Adam optimizer [18] and representations with the stochastic gradient descent [19]. All metrics were reported on the test set of sample size 100. For each new training set, a new test set was created. Models were trained and evaluated on the same data sets. The encoder refers to an autoencoder using the pre-trained decoder whose weights were frozen, and autoencoder refers to an identical autoencoder but without pre-training the decoder. The decoder and the encoder were trained for 500 epochs each, and the autoencoder was trained for 1000 epochs.

Figure 4 shows the test reconstruction loss and the representation correlation of the decoder, the encoder (trained on pre-trained decoder), and the autoencoder (AE) for different decoder loads α_d. We observed that the load α_d must be >1 for the decoder to be consistently well-determined and thus for achieving an exact solution for the representations (a correlation of 1) and good input reconstruction. The encoder with a fixed and pre-trained decoder in this experiment took a slightly higher α_d of at least 3.3 in order to achieve correlations ≥ 0.99. This may be explained by the higher necessary load of the encoder, which was mostly mitigated here by pre-training the decoder. One reason for a successful training of an encoder on a pre-trained decoder here, unlike in the MNIST experiment, could be that the encoder in this experiment was more complex than the decoder. The naive (not pre-trained) autoencoder in this experiment was unable to find the exact solution for z, as can be seen by the generally lower and more variable representation correlations. Additionally, it took a load of roughly 9 for the autoencoder to achieve similar reconstruction losses as the decoder and the encoder, which here was equivalent to training sample sizes above 10,000. These observations support the hypothesis that the encoder is the limiting factor in learning a precise mapping from the representation z to the input space x and that the decoder on its own can learn a meaningful representation.

4. Conclusions

In this study, we introduced a new manifold learning perspective on autoencoders. We argued that it is useful to view the decoder as defining a low-dimensional manifold in input space. Training of a decoder alone amounts to optimizing the weights of the decoder and the representations of the training data so as to minimize the average distance between the training samples and their projections onto the manifold, which are the decoder reconstructions.

We showed that it is possible (and probably common) to have an over-determined decoder and an under-determined encoder. Our tests confirmed that a decoder trained alone generally performs better on the test data than an autoencoder trained from scratch. However, in one test, we saw that an autoencoder trained with a fixed optimized decoder performs almost as well as the decoder alone. We further demonstrated that this approach can lead to meaningful representations that may be useful for downstream tasks. Albeit only covering the task of reconstruction in this demonstration, we believe that this approach can easily be extended to other tasks in need of a useful representation. One such example is time series forecasting, based on building blocks such as LSTMs and transformers.

The similarity between representation learning, autoencoders, and manifold learning has often been pointed out. In most work on autoencoders, the focus has been on the encoder, and various types of regularization have been proposed to limit over-fitting of data, such as [5,9]. In our view, a better understanding can be obtained when first focusing on the decoder, which constrains the autoencoder, because it is generally much better specified by the training data than the encoder.

In this work, we focused on the reconstruction error as a measure of performance. Often, learned representations are used for classification or other tasks. We will extend these ideas to such other tasks in the future to see if the good performance of decoder training extends to these.

The results and observations presented in this study are additionally of high relevance for generative models. We are currently working on the advancement of the ideas in this context.

Supplementary Materials: The following are available at https://www.mdpi.com/article/10.3390/e23111403/s1.

Author Contributions: Conceptualization, A.K.; methodology, A.K.; software, A.K. and V.S.; validation, A.K. and V.S.; formal analysis, A.K. and V.S.; investigation, A.K. and V.S.; resources, A.K.; writing—original draft preparation, A.K.; writing—review and editing, A.K. and V.S.; visualization,

A.K. and V.S.; supervision, A.K.; project administration, A.K.; funding acquisition, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: A.K. is supported by a grant from the Novo Nordisk Foundation to the Center for Basic Machine Learning Research in Life Science, MLLS (PI: Ole Winther).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. They include MNIST [13] and CIFAR10 [14]. Other data were randomly generated and can be reproduced using the description in Section 3.3 and the supplementary code available.

Acknowledgments: We acknowledge all the learned discussions on representation learning with Wouter Boomsma, Jes Frellsen, Ole Winther, Aasa Faragen, Søren Hauberg, and other members of MLLS.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Architectural Details

Convolutional layers are described by kernel size, stride, and padding in brackets behind the layer type. If an activation function is applied right after a layer, it is included in the layer column.

Table A1. MNIST model architecture.

Layer	Output Size	Model
Input	(1,28,28)	Encoder
Conv2D (4,2,1)—ReLU	(64,14,14)	Encoder
Conv2D (4,2,1)—ReLU	(128,7,7)	Encoder
Linear—ReLU	20	Encoder
ConvTranspose2D (4,2,1)—ReLU	(128,7,7)	Decoder
ConvTranspose2D (4,2,1)—ReLU	(64,14,14)	Decoder
Linear—Sigmoid	20	Decoder

Table A2. CIFAR10 model architecture. BN refers to batch normalization.

Layer	Output Size	Model
Input	(3,32,32)	Encoder
Conv2D (1,1,0)—BN—ReLU	(64,32,32)	Encoder
Conv2D (4,2,1)— BN—ReLU	(64,16,16)	Encoder
Conv2D (4,2,1)—BN—ReLU	(128,8,8)	Encoder
Conv2D (4,2,1)—BN—ReLU	(128,4,4)	Encoder (4×4 AE latent)
Conv2D (4,2,0)—ReLU	(256,1,1)	Encoder (1×1 AE latent)
ConvTranspose2D (4,2,0)—BN—ReLU	(128,4,4)	Decoder (1×1 1st layer)
ConvTranspose2D (4,2,1)—BN—ReLU	(128,8,8)	Decoder (4×4 1st layer)
ConvTranspose2D (4,2,1)—BN—ReLU	(64,16,16)	Decoder
ConvTranspose2D (4,2,1)—BN—ReLU	(64,32,32)	Decoder
ConvTranspose2D (1,1,0)—Sigmoid	(3,32,32)	Decoder

Table A3. Model architecture for simulated data.

Layer	Output Size	Model
Input	1000	Encoder
Linear—LeakyReLU (slope = 0.01)	100	Encoder
SparseLinear—ReLU	1000	Decoder

References

1. Rumelhart, D.; Hinton, G.; Williams, R. *Learning Internal Representations by Error Propagation*; MIT Press: Cambridge, UK, 1986; Volume 1, Chapter 8, pp. 318–362.
2. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
3. Tschannen, M.; Bachem, O.; Lucic, M. Recent Advances in Autoencoder-Based Representation Learning. *arXiv* **2018**, arXiv:1812.05069.
4. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, ACM, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.
5. Jia, K.; Sun, L.; Gao, S.; Song, Z.; Shi, B.E. Laplacian Auto-Encoders: An explicit learning of nonlinear data manifold. *Neurocomputing* **2015**, *160*, 250–260.
6. Cottrell, G.; Munro, P.; Zipser, D. Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming. In Proceedings of the Ninth Annual Conference of the Cognitive Science Society, Seattle, WA, USA, 16–18 July 1987; Lawrence Erlbaum: Hillsdale, MI, USA, 1987; pp. 462–473.
7. Elman, J.; Zipser, D. Learning the Hidden Structure of Speech. *J. Acoust. Soc. Am.* **1988**, *83*, 1615–1626. [CrossRef] [PubMed]
8. Bourlard, H.; Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.* **1988**, *59*, 291–294. [CrossRef] [PubMed]
9. Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*; Getoor, L., Scheffer, T., Eds.; Omnipress: Madison, WI, USA, 2011; pp. 833–840.
10. Kavukcuoglu, K.; Ranzato, M.; LeCun, Y. Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition. *arXiv* **2010**, arXiv:1010.3467.
11. Han, T.; Lu, Y.; Zhu, S.C.; Wu, Y.N. Alternating Back-Propagation for Generator Network. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
12. Krogh, A.; Thorbergsson, G.; Hertz, J. A Cost Function for Internal Representations. In *Advances in Neural Information Processing Systems*; Touretzky, D., Ed.; Morgan Kaufmann: San Mateo, CA, USA; Denver, CO, USA, 1989; Volume 2, pp. 733–740.
13. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
14. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Technical Report; 2009. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdfhere (accessed on 23 October 2021).
15. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2019; pp. 8024–8035.
16. Fukushima, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybern.* **1980**, *36*, 193–202. [CrossRef] [PubMed]
17. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*; Fürnkranz, J., Joachims, T., Eds.; Omnipress: Madison, WI, USA, 2010; pp. 807–814.
18. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980; Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
19. Robbins, H.; Monro, S. A Stochastic Approximation Method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
20. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, arXiv:1511.06434.
21. Radosavovic, I.; Kosaraju, R.P.; Girshick, R.; He, K.; Dollár, P. Designing Network Design Spaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020.
22. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.

9 | Paper II

The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data

Viktoria Schuster[1] **and Anders Krogh**[1,2,*]

[1] Center for Health Data Science, University of Copenhagen, Blegdamsvej 3B, 2200 Copenhagen, Denmark
[2] Department of Computer Science, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark

* Corresponding author: Anders Krogh (akrogh@di.ku.dk)

Publication details

This paper was published in *Oxford Bioinformatics* on 12 August 2023.

Supplementary data are available at *Bioinformatics* online under DOI https://doi.org/10.1093/bioinformatics/btad497.

Bioinformatics, 2023, **39(9)**, btad497
https://doi.org/10.1093/bioinformatics/btad497
Advance access publication 12 August 2023
Original Paper

OXFORD

Gene expression

The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data

Viktoria Schuster [1] **and Anders Krogh** [1,2,*]

[1]Center for Health Data Science, University of Copenhagen, 2200 Copenhagen, Denmark
[2]Department of Computer Science, University of Copenhagen, 2100 Copenhagen, Denmark

*Corresponding author. Department of Computer Science, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark.
E-mail: akrogh@di.ku.dk (A.K.)

Associate Editor: Valentina Boeva

Abstract

Motivation: Learning low-dimensional representations of single-cell transcriptomics has become instrumental to its downstream analysis. The state of the art is currently represented by neural network models, such as variational autoencoders, which use a variational approximation of the likelihood for inference.

Results: We here present the Deep Generative Decoder (DGD), a simple generative model that computes model parameters and representations directly via maximum a posteriori estimation. The DGD handles complex parameterized latent distributions naturally unlike variational autoencoders, which typically use a fixed Gaussian distribution, because of the complexity of adding other types. We first show its general functionality on a commonly used benchmark set, Fashion-MNIST. Secondly, we apply the model to multiple single-cell datasets. Here, the DGD learns low-dimensional, meaningful, and well-structured latent representations with sub-clustering beyond the provided labels. The advantages of this approach are its simplicity and its capability to provide representations of much smaller dimensionality than a comparable variational autoencoder.

Availability and implementation: scDGD is available as a python package at https://github.com/Center-for-Health-Data-Science/scDGD. The remaining code is made available here: https://github.com/Center-for-Health-Data-Science/dgd.

1 Introduction

High-throughput methods in biology and medicine produce vast amounts of high-dimensional noisy data that we seek to extract knowledge from. The first step is often to obtain a low-dimensional representation of the data through clustering, PCA analysis or other similar techniques. A good example is gene expression analysis, where we need to compare counts for each of the 10–20 thousand genes between samples or cells. Here, the first analysis is often to visualize the data in two dimensions with UMAP (McInnes *et al.* 2018) or t-SNE (van der Maaten and Hinton 2008). Although these tools are indispensable, they do not model the uncertainty of the count data or reveal any of the underlying structure in the data. Therefore, it has become popular to make use of generative models that give low-dimensional representations mapping to a probability distribution over the data. Some of the most common are variational autoencoders (VAEs) (Kingma and Welling 2013), generative adversarial networks (Goodfellow *et al.* 2014), normalizing flows (Rezende and Mohamed 2015), and energy-based models (Lecun *et al.* 2006). Even though these approaches have their shortcomings (Bond-Taylor *et al.* 2021), they have shown great success in a multitude of applications, such as single-cell RNA-sequencing (Lopez *et al.* 2018, Seninge *et al.* 2021), image and surface generation (Abukmeil *et al.* 2021, Kammoun *et al.* 2022), and natural language processing (Abukmeil *et al.* 2021, Wali *et al.* 2022). In addition, diffusion models (Ho *et al.* 2020) and Transformers (Vaswani *et al.* 2017) are among the best performing and most popular approaches in generative modelling of image and text data. However, these methodologies do not necessarily include probabilistic latent spaces.

The VAE is the most common type of generative model applied to biological data. In general, VAEs consist of an encoder that maps data to a lower-dimensional probabilistic representation in latent space and a decoder that maps back to the data. The model is trained using maximum likelihood estimation (MLE), which seeks to find a set of parameters that maximize the likelihood of the data. However, the likelihood calculation is intractable due to an integral over the latent space. The solution in the VAE is to approximate the likelihood and maximize a variational lower bound (the 'ELBO'). This approximation is not always accurate (Cremer *et al.* 2018) and favours larger latent dimensionalities than necessary (Yacoby *et al.* 2020).

The decoder alone specifies the generative model and the encoder of the VAE is just a convenient tool for finding representations, but it adds a whole new set of modelling choices to be made, which influence the inference gap and expressiveness (Cremer *et al.* 2018). As a result, encoder-less representation learning has been suggested many times in different ways.

Received: November 29, 2022. **Revised:** July 12, 2023. **Editorial Decision:** July 27, 2023. **Accepted:** August 10, 2023

There are simple, deterministic approaches that maximize the likelihood over representations of the training data and the decoder parameters at the same time (Han *et al.* 2017, Bojanowski *et al.* 2018, Zadeh *et al.* 2021). Here and in our previous work, the representations are treated like the other model parameters and since gradients can be derived exactly, the model can be trained using standard gradient descent. There are also other encoder-less generative models, such as Gaussian Process Latent Variable Models (GP-LVMs) (Lawrence 2003) or PCA-based models (Collins *et al.* 2001, Mohamed *et al.* 2008, Townes *et al.* 2019). In their vanilla form, they have issues with scaling and limited model complexity and have not quite gained traction in the application to biological data compared to VAEs. However, scalable versions of GP-LVMs have been presented (Ahmed *et al.* 2019, Verma and Engelhardt 2020, Lalchand *et al.* 2022) and PCA-based methods have been applied for the joint modelling of multiple single-cell modalities (Mourragui *et al.* 2023). Scalability in these GP-LVM applications is achieved by applying sparse Gaussian processes and amortized inference (Ahmed *et al.* 2019, Verma and Engelhardt 2020, Lalchand *et al.* 2022). As a result, they are still limited to the same problems of variational inference (VI) (Cremer *et al.* 2018) and sensitivity of latent initialization (Lawrence 2003). The standard VAE and many other models use a fixed Gaussian distribution over latent space. This may result in under-expressive representations, and it has been shown that learning the parameters of a Gaussian mixture leads to improved generalization and clustering (Dilokthanakul *et al.* 2017; Guo *et al.* 2020). However, this makes the VI even more complex, as it has to be combined with score matching (Vahdat *et al.* 2021), or other approaches. Another alternative approach worth mentioning is the VQ-VAE (van den Oord *et al.* 2017), which learns discrete representations with more powerful priors than a standard Gaussian, but still requires an encoder.

In this work, we present a probabilistic formulation of a generative decoder using maximum a posteriori (MAP) estimation of all parameters. MAP estimation is the Bayesian analogue of MLE and seeks to find the set of parameters that maximize the posterior, which is the probability of the representations and model parameters given the observed data. There is no intractable step in this estimation, so both model parameters and representations can be estimated directly. The model we are introducing consists of a decoder (or generator), such as a feed-forward neural network, and a distribution over representations with learnable parameters, which are optional. Estimating the representations of the training samples as well as the parameters is straight-forward and can be done by gradient descent as in a non-probabilistic formulation (Schuster and Krogh 2021a). One of the main advantages of this approach over VI is the ease and flexibility with which distributions over representations can be learned. The advantage over GP-LVMs is that the approach is scalable. Our emphasis here is on a model with a parameterized distribution over representations. This is motivated by the intuition that representations are meaningful and are likely to group samples with different properties. We think of this approach as a marriage of generative models and manifold learning, such as UMAP. The beauty of the approach lies in its simplicity and utility. The decoder parameters, latent representations and latent distribution are estimated in the same way. The simplicity makes it easy to extend the model to more complex distributions and losses. Unlike in Gaussian Mixture VAEs (Dilokthanakul *et al.* 2017b, Bai *et al.* 2022), no additional encoder for the Gaussian components is needed. It also makes it much simpler to understand for non-specialists. This is especially important, as we see great potential for this approach in fields like biology and medicine.

In the following sections, we present the theory of the model and experiments on two very different types of data. We first demonstrate the general functionality of the proposed model on the Fashion-MNIST (Xiao *et al.* 2017) benchmark dataset. This is a typical choice in deep learning and an important step, because it enables us to investigate and understand the model's generative capabilities and potential caveats. We further explore the complexity of the distribution over representations and compare latent space and generative capabilities to models using VI. We also showcase the extension of the approach to supervised learning in which the mixtures of the latent distribution represent specific data classes. After establishing the model's functionality, we show an application to single-cell gene expression count modelling. We chose a dataset of peripheral blood mononuclear cells (PBMC) (Zheng *et al.* 2017), for which we compare our approach in terms of data reconstruction and clustering performance to scVI (Gayoso *et al.* 2022) and scVAE (Grønbech *et al.* 2020). The latter is a VAE also using an adaptive mixture of Gaussians rather than a single fixed Gaussian to model the latent representation. We show that our model learns a representation and Gaussian mixture model which cluster well according to cell type with additional, previously unobserved sub-clustering. Finally, we compare the DGD and scVI on 11 different single-cell gene expression datasets using default settings of scVI and our model.

2 Materials and methods

2.1 Models

2.1.1 The DGD

Consider a model with observed variable x and continuous latent variable z, which we also refer to as a representation. Usually, the x-space (or sample space) is of a higher dimension than the z-space, so the model gives a low-dimensional representation of data. A neural network, the decoder, with parameters θ takes z as input and outputs $f_\theta(z)$, which are the parameters for the distribution of x. These could be the means and standard deviations of independent normal distributions. Thus, the neural network defines a conditional distribution in sample space, $P(x|z,\theta) = P(x|f_\theta(z))$. We assume a distribution over representation space, $P(z|\phi)$, with parameters ϕ. The parameters may be fixed as in a standard VAE with a fixed Gaussian distribution over representations. It can also have adjustable parameters, such as a mixture of Gaussians with trainable means, covariances, and mixture coefficients, but essentially any (differentiable) parameterized distribution can be used. When introducing priors over parameters, the joint probability of everything becomes

$$\begin{aligned} P(x,z,\phi,\theta) &= P(x,z|\phi,\theta)P(\phi)P(\theta) \\ &= P(x|z,\theta)P(z|\phi)P(\theta)P(\phi). \end{aligned} \tag{1}$$

Again, $P(x|z,\theta)$ represents the decoder neural network, $P(z|\phi)$ the distribution over representations, $P(\theta)$ the prior over the decoder parameters (neural network weights), and

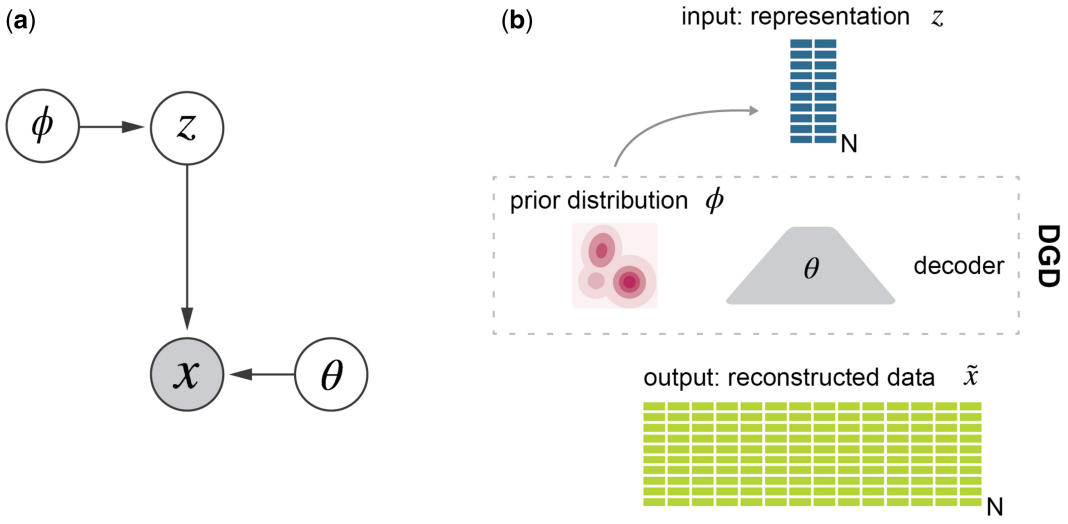


Figure 1. The model. (a) Graphical model and (b) schematic of the deep generative decoder. The DGD consists of a decoder of any desired architecture with parameters θ mapping the latent representation Z to the data space X. The representation is modelled by a probability distribution with parameters ϕ. N is the number of samples.

$P(\phi)$ the prior over the parameters in the representation space distribution. The model schematic is shown in Figure 1. We use MAP estimation to find optimal parameters and representations. For a dataset $X = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$ with N samples we thus want to find representations $Z = \{\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^N\}$ and parameters ϕ, θ that maximize $P(Z, \phi, \theta \,|X)$:

$$\begin{aligned} \underset{Z,\phi,\theta}{\operatorname{argmax}} P(Z, \phi, \theta \,|X) &= \underset{Z,\phi,\theta}{\operatorname{argmax}} P(X, Z, \phi, \theta)/P(X) \\ &= \underset{Z,\phi,\theta}{\operatorname{argmax}} P(X, Z, \phi, \theta). \end{aligned} \tag{2}$$

If we assume independence among training samples, the log of this joint probability can be obtained from (3), with i indexing the samples

$$\begin{aligned} \log P(X, Z, \phi, \theta) = \textstyle\sum_i \Big(\log P(\mathbf{x}^i|\mathbf{z}^i, \theta) + \log P(\mathbf{z}^i|\phi)\Big) \\ + \log P(\phi) + \log P(\theta). \end{aligned} \tag{3}$$

This is the quantity we want to maximize with respect to both the representations (z) for all samples and the model parameters (ϕ, θ). The training process can be imagined like this: a representation z for each training sample is initialized. This can be a random sample from a chosen distribution or a zero-valued vector. Representations are then passed through the decoder to give $P(x|z, \theta)$ and losses are computed from this and $P(z|\phi)$. The decoder can be any type of Neural Network that fulfils this constraint. The choice of reconstruction loss is equally flexible but should ideally represent $P(\mathbf{x} \mid \mathbf{z})$. All parameters θ, ϕ, and z are then updated via back-propagation. The implementation is efficient and straightforward and compatible with standard modules and loss functions in deep learning frameworks, such as PyTorch. Algorithm 1 summarises the process.

Since representations are updated once per epoch and decoder and GMM parameters every batch pass, we use different instances of the optimizer for the different parameter sets. We have also observed that the GMM requires much larger steps than neural network parameters, so we choose to have an optimizer per parameter set with individually selected learning rates.

Algorithm 1. Training

Initialize parameters for representations $\mathbf{z}^i$, decoder and GMM
for epoch in n_epochs **do**
  for $\mathbf{x}^i$, i in training data **do**
    $\mathbf{z}^i = Z_i$.
    $\mathbf{y}^i = \text{model}(\mathbf{z}^i)$
    loss $= L_{reconstruction}(\mathbf{y}^i, \mathbf{x}^i) + L_{GMM}(\mathbf{z}^i)$
    Backpropagation
    Optimizer step for model and GMM
  end for
  Optimizer step for Representation
end for

Once the model is estimated, prediction on new data points is done by first finding an optimal representation by maximizing $P(x|z, \theta)P(z|\phi)$ as above while keeping all other model parameters fixed, i.e. do gradient descent in z alone. For each new data point, one or more new representations are initialized. The type of initialization can be chosen. It can be from zero as done for the training data or from component means. For the latter initialization technique, the best starting point for each new sample is derived from the minimum reconstruction loss from all K component means. This presents our default setting and essentially assigns the optimal component to each new sample. From there on, representations are typically optimized for a very short time (10 epochs) with a batch size of 32. Inference of new representations is of course done with frozen decoder and GMM parameters.

In this work, we use a parameterized Gaussian mixture for the representations ($P(z|\phi)$) with a mollified Uniform ('soft-ball prior') as prior over component means. Hereafter, we will refer to this Gaussian Mixture Model as GMM. The mixture model consists of K mixture components each with a mean vector $\boldsymbol{\mu}$ with same dimensionality as the

representations (m), diagonal covariance $\boldsymbol{\Sigma}$ and a mixture coefficient $\boldsymbol{c}$ for each component. The mixture coefficients are transformed into component weights $\boldsymbol{w}$ through softmax activation. Since the diagonal covariance matrix can only take positive values, the according parameter is learned as the negative log-covariance.

We use a weight decay in the training of the decoder, which corresponds to a Gaussian prior on the weights. The priors over the GMM parameters ϕ are as follows. The prior on the mixture weights is an m-dimensional Dirichlet distribution with uniform parameters, α. The prior on mixture means is a mollified uniform which we call the 'softball' distribution with hyperparameters scale as the radius of the m-ball and sharpness determining the slope of the boundary. Details on initialization and formulation of the log-probability can be found in Supplementary section 'The softball prior'. For the negative log-variance, we use a Gaussian prior $\mathcal{N}(-2\log(\sigma), 1)$ with the same mean and standard deviation for all dimensions. Mixture coefficients and negative log-variances are initialized with default values 1 (so that components weights are uniformly K^{-1}) and $-2\log(\sigma)$ with $\sigma = 0.2 \times \text{scale} \times K^{-1}$, respectively. The default values for α, scale, and sharpness are all 1. Component means are initialized by sampling from the softball prior. The distributional component of the loss is given as (4)

$$\begin{aligned} -\log P(\mathbf{z}) &= -\log\left(\sum_k \exp\left(\mathbf{w}^k \sum_i \log P(\mathbf{z}^i|\phi^k)\right)\right) \\ &\quad -\log P(\phi) \\ \text{with } P(\mathbf{z}^i|\phi^k)) &= P(\mathbf{z}^i|\boldsymbol{\mu}^k, \mathbf{c}^k, -\log\boldsymbol{\Sigma}^k)) \\ \text{and } P(\phi) &= P(\boldsymbol{\mu}, \mathbf{c}, -\log\boldsymbol{\Sigma}), \end{aligned} \tag{4}$$

with $P(\mathbf{z}^i|\boldsymbol{\mu}^k, \mathbf{c}^k, -\log\boldsymbol{\Sigma}^k)$ as the density of the kth multivariate Gaussian component for representation $\boldsymbol{z}^i$.

2.1.2 Fashion-MNIST DGD

For the image-generating DGD, we choose a network architecture based on convolutions and tricks from other image generation models and image segmentation techniques (He *et al.* 2016, van den Oord *et al.* 2016, Vahdat and Kautz 2020). The decoder starts with two fully connected hidden layers fed with the latent representations. The first hidden layer has 100 units, the second *capacity* $\times 3 \times 3$ units. The capacity represents the minimum number of channels of the convolutional layers except for the grey-scale output channel and is set to 64. The output of the fully connected hidden layers is reshaped into $(batch, 3, 3, capacity)$ and fed into a series of NN blocks. These blocks consist of a transposed convolutional layer, Swish activation and a Squeeze and Excitation layer as applied by Vahdat and Kautz (2020). Input- and output channels as well as kernel size, strike, and padding depend on the position of the block in the scheme and can be seen in Supplementary Fig. S5. There are four main blocks and two skip connection blocks. The outputs of combined blocks go through the activation function after summation. The series of these NN blocks is followed by a PixelCNN with five layers and mask size five (van den Oord *et al.* 2016) and a last simple transposed convolutional layer reducing the number of channels to one. The output is scaled using Sigmoid activation.

Latent dimensionality and convolutional capacity vary depending on the experiment. For the models with 10 and 20 Gaussian components, we use a latent dimension of 20. The standard deviation of the GMM components are calculated as $\frac{scale}{components}$. This ensures that the components are sufficiently separated and alleviates the burden of another hyperparameter to optimize. The scale and hardness of the softball mean prior are three and five, respectively, and the Dirichlet alpha is set to two since we are dealing with balanced classes. For the models involving a standard Gaussian (including VAD and VAE), we test latent dimensionalities of 20, 50, and 100. Since the prior is not learned, softball and Dirichlet prior are not relevant. The Supplementary material contains results of a model with latent dimension 2, capacity 32, and varying number of Gaussian components (1, 10, and 20).

For training, we use the binary cross-entropy (BCE) loss and Adam optimization with betas 0.5 and 0.7. For decoder, representation, and GMM, we choose learning rates $1e-3$, $1e-2$, and $1e-1$, respectively. This relationship has evolved as a rule of thumb for the DGD. Learning rates are best chosen with respect to a desired decoder learning rate (which is to be optimized for each task independently). From there, the representation learning rate should be 10 times the decoder learning rate, and the GMM learning rate should be between 10 and 20 times that of the decoder.

Hyperparameters, such as the number of hidden dimensions, capacity, dropout, and the softball prior scale were found through optimization with respect to the validation reconstruction loss. The summary is available as a parallel coordinate plot from weights and biases runs in Supplementary Fig. S1.

2.1.3 VAD and VAE

The VAD and VAE decoder implementations tested here are architecturally identical to the DGDs with a standard Gaussian. The VAE encoder was for simplicity chosen to mirror the decoder, with normal convolutional layers instead of the transposed convolutions. The capacity for all models is 32 and latent dimensions tested are 20, 50, and 100. Instead of the negative log-density of the GMM, the loss for the distribution over latent space is given as the Kullback–Leibler divergence as typical for VI (Kingma and Welling 2013).

2.1.4 Supervised learning

In the unsupervised model, the log-likelihood $\log P(\mathbf{z}^i|\phi)$ of an individual representation $\mathbf{z}^i$ being drawn from the parameterized distribution $P(\phi)$ is given by the log sum of the probability densities of $\mathbf{z}^i$ per component times the corresponding component probabilities, which are given by the softmax of the weights. In the supervised setting, we can ignore all components except the one that has been assigned to the sample's class. We thus only have to calculate the probability density of $\mathbf{z}^i$ for the given component multiplied with the component probability. The losses for all other components will be zero and they will thus not receive gradients.

2.1.5 Single-cell DGD

The architecture of scDGD is much simpler than that of the Fashion-MNIST DGD. The decoder consists of the input layer with units defined by the latent dimensionality, and three hidden layers of each 100 units, connected to the output layer of (in the case of the PBMC dataset) 32 728 units, representing all transcripts in the data. The number of output units is given by the number of genes with non-zero expression counts in the data. The latent dimensionality was empirically found

best to be 20 in terms of validation reconstruction loss and clustering accuracy of the cell types. The depth of the network was also empirically determined. The parallel coordinate summary of the hyperparameter optimization can be found in Supplementary Fig. S6.

The normalized expression counts are modelled with a Negative Binomial distribution. Normalization is achieved by dividing the true expression count with the sample's largest count. We can therefore use Sigmoid activation on the output layer. The reconstruction loss is given as the probability density of the re-scaled expression count and a gene-specific, learned dispersion parameter. Since this parameter is positive definite, we learn its logarithmic counterpart. For training, we use Adam optimization with betas 0.5 and 0.7. For decoder, representation, and GMM, we choose learning rates $1e-3$, $1e-2$, and $1e-2$, respectively. Two models are trained with 9 and 18 Gaussian components, respectively, for the PBMC dataset.

The component means of the GMM are distributed according to the mollified Uniform, for which we set scale and sharpness to one each. This applies to both 9-component and 18-component model. The standard deviation of the components is initialized with a mean of 0.02 for the 9-component model and 0.01 for the 18-component model, which roughly corresponds to the formula we have introduced in the previous section, but is modified to a fifth of that in order to improve the component separation. Dirichlet alphas are set to two and one for models with 9 and 18 components, respectively. This is motivated by a Dirichlet alpha of one allowing for non-uniform component weights, which is desired in this case of imbalanced cell type classes. In the case of nine Gaussian components, a Dirichlet alpha of one resulted in even more components representing T cells, so we decided to distribute the components more uniformly by increasing alpha.

All hyperparameters described above, except for the number of GMM components and of course the output dimensionality, present the empirically determined default parameters of scDGD. We applied this default model to 11 more datasets, in which the output dimensionality was set as the number of transcript found in the data, and the number of Gaussian components was determined by the number of unique cell types identified by CellTypist. Models for all datasets except PBMC (20k) were trained for 800 epochs. PBMC (20k) was trained for 1000 epochs. In all scDGD models, the learning rate of the decoder was reduced to $1e-4$ after 500 epochs.

2.1.6 scVI

An scVI model was trained on our PBMC train set. The model is implemented in Python version 3.8 with the scvi-tools (Gayoso *et al.* 2022) package version 0.17.4. as described in Lopez *et al.* (2018) with one hidden layer of 128 hidden units in both encoder and decoder. We chose a latent dimension of 20 for comparability with our model. The dropout rate is set to 0.1. The model was trained for 1000 epochs. For the analysis and comparison to scDGD, the latent space is clustered with K-means clustering ($k=9$) and then evaluated by the ARI. The lower bound is computed on our held out test set. Latent representations and ELBO were computed using scVI's implemented functions. NLLs and RMSEs were computed based on the models returned mean and dispersion parameters for the test set.

For the remaining datasets, the scVI models were again initialized with default parameters and trained for up to 400 epochs, which represents the upper bound of the default number of epochs (for large datasets, the automatically calculated number of epochs can be very low).

2.2 Datasets

In this work, we made use of in total 13 publicly available datasets, out of which 12 represent single-cell transcriptomics sets. The dataset referred to as PBMC was used to develop the application of the DGD to single-cell expression data. The remaining single-cell datasets were used to evaluate the performance of scDGD with default parameters in comparison to the popular and successful scVI model with default parameters.

2.2.1 Fashion-MNIST

The dataset used in our proof of concept is the natural image Fashion-MNIST (Xiao *et al.* 2017) set. We used the dataset's implemented train-test split.

2.2.2 PBMC

The dataset used to develop the single-cell application of our model is a single-cell gene expression count dataset of PBMC presented by Zheng *et al.* (2017). The data are provided by 10× Genomics under 'Single Cell 3' Paper: Zheng *et al.* 2017 (v1 Chemistry)' and consists of data from the following nine cell types: CD4+/CD45RA+/CD25- naïve T cells, CD4+ helper T cells, CD4+/CD25+ regulatory T cells, CD4+/CD45RO+ memory T cells, CD8+/CD45RA+ naïve cytotoxic T cells, CD8+ cytotoxic T cells, CD56+ natural killer cells, CD34+ cells, and CD19+ B cells. As Grønbech *et al.* (2020), we used the filtered gene–cell matrices. These data are extremely sparse with 98% of the data being zero (Grønbech *et al.* 2020). Of the 32 738 genes covered by this assay, only 21 812 are expressed in the whole dataset at least once. The 92 043 samples are randomly split into train, validation, and test set using percentages 81%–9%–10%. The validation set is used to finding hyperparameters, and the test set is used for final evaluation.

2.2.3 Single-cell evaluation sets

The following datasets were taken from 10× Genomics or chosen because of their use in the scVI paper (Gayoso *et al.* 2022). We used provided raw counts of cells that passed quality control filtering and did not apply any feature selection, modelling all transcripts available. Datasets were annotated using the CellTypist (Domínguez Conde *et al.* 2022, Xu *et al.* 2023) python package. The resulting cell type labels were used in the DGD for automatic selection of the number of Gaussian components and in the clustering performance evaluation as an approximated ground truth. All datasets were split into 80% training, 10% validation, and 10% test data. The raw counts with cell type and data split assignment as observables were exported as Anndata (Virshup *et al.* 2021) objects, which were used to train and evaluate both scDGD and scVI.

PBMC (500)

The smallest of the datasets used for the elaborate analysis of our model's performance and applicability is the 'PBMCs from human (3'LT v3.1, Chromium X), Single-Cell Gene Expression Dataset by Cell Ranger 6.1.0 (2019)' dataset retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/500-human-pbm-cs-3-lt-v-3-1-chromium-x-3-

1-low-6-1-0. It contains 587 samples and 36 601 features. For cell type annotation, we used the 'Immune_All_Low' model as reference and majority voting in CellTypist. This resulted in the presence of 11 distinct cell types.

BMMC (2k)

The next dataset, 'Frozen BMMCs (Healthy Control 1), Single-Cell Gene Expression Dataset by Cell Ranger 1.1.0 (2016)', contains 1985 bone marrow mononuclear cells with 32 738 features and was retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/frozen-bmm-cs-healthy-control-1-1-standard-1-1-0. Cell type annotations were approximated using CellTpyist with the 'Immune_All_Low' reference model and majority voting. This resulted in 15 distinct cell types.

Cortex (3k)

A dataset of 3005 cells from mouse cortex and hippocampus with 19 972 features was published in Zeisel *et al.* (2015). Cell type annotations were approximated using CellTpyist with the 'Developing_Mouse_Brain' reference model and majority voting. This resulted in 14 distinct cell types.

Jejunum (5k)

The dataset of human jejunum '5k Human Jejunum Nuclei Isolated with Chromium Nuclei Isolation Kit, Single-Cell Gene Expression Dataset by Cell Ranger 7.0.0 (2022)' retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/5k-human-jejunum-nuclei-isolated-with-chromium-nuclei-isolation-kit-3-1-standard contained 4392 cells with 36 601 features. Cell type annotations were approximated using CellTpyist with the 'Cells_Intestinal_Tract' reference model and majority voting. This resulted in 24 distinct cell types.

Mouse brain (5k)

This dataset '5k Adult Mouse Brain Nuclei Isolated with Chromium Nuclei Isolation Kit, Single-Cell Gene Expression Dataset by Cell Ranger 7.0.0 (2022)' comprised 7377 cells from the adult mouse brain with 32 285 features and retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/5k-adult-mouse-brain-nuclei-isolated-with-chromium-nuclei-isolation-kit-3-1-standard. Cell type annotations were approximated using CellTpyist with the 'Developing_Mouse_Brain' reference model and majority voting. This resulted in seven distinct cell types.

Whole blood (8k)

From the blood of healthy females, a total of 8000 PBMCs, Neutrophils and Granulocytes were extracted into the dataset 'Whole Blood RBC Lysis for PBMCs and Neutrophils, Granulocytes (3'), Single-Cell Gene Expression Dataset by Cell Ranger 6.1.0 (2021)', containing 36 601 features. It was retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/whole-blood-rbc-lysis-for-pbmcs-neutrophils-granulocytes-3-3-1-standard. Cell type annotations were approximated using CellTpyist with the 'Immune_All_Low' reference model and majority voting. This resulted in 10 distinct cell types.

Heart (10k)

The 7713 cells with 31 053 features in '10k Heart Cells from an E18 mouse (v3 chemistry), Single-Cell Gene Expression Dataset by Cell Ranger 3.0.0 (2018)' were extracted from an E18 mouse. The data were retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/10-k-heart-cells-from-an-e-18-mouse-v-3-chemistry-3-standard-3-0-0. Cell type annotations were approximated using CellTpyist with the 'Immune_All_Low' reference model. This resulted in three distinct cell types.

PBMC (10k)

'10k Human PBMCs [(3'LT v3.1, Chromium X (with intronic reads)], Single-Cell Gene Expression Dataset by Cell Ranger 6.1.2 (2022)' retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/10k-human-pbmcs-3-v3-1-chromium-x-with-intronic-reads-3-1-high contains 11 984 PBMCs from healthy female donors between the ages of 25 and 30, with 36 601 transcripts covered. Cell type annotations were approximated using CellTpyist with the 'Immune_All_Low' reference model and majority voting. This resulted in 19 distinct cell types.

PBMC (20k)

Another dataset from female donors aged 25–30 provided was used, containing a total of 23 837 cells with 36 601 features. The '20k Human PBMCs (3' HT v3.1, Chromium X), Single-Cell Gene Expression Dataset by Cell Ranger 6.1.0 (2021)' data were retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/20-k-human-pbmcs-3-ht-v-3-1-chromium-x-3-1-high-6-1-0. Cell type annotations were approximated using CellTpyist with the 'Immune_All_Low' reference model and majority voting. This resulted in again 19 distinct cell types.

Hemato (25k)

After excluding the 'basal-bm1' library due to poor quality [as presented in Gayoso *et al.* (2022) by recommendation of the authors], the dataset of haematopoietic progenitor cells from mice (Tusi *et al.* 2018) comprised 25 050 cells and 28 205 features. Cell type annotations were approximated using CellTpyist with the 'Cells_Intestinal_Tract' reference model and majority voting if cell types represented less than 100 cells. This resulted in 18 distinct cell types.

Mouse brain (1M)

The extremely large dataset of 1 306 127 brain cells from two E18 mice ['1.3 Million Brain Cells from E18 Mice, Single-Cell Gene Expression Dataset by Cell Ranger 1.3.0 (2017)'] was retrieved from 10× Genomics, https://www.10xgenomics.com/resources/datasets/1-3-million-brain-cells-from-e-18-mice-2-standard-1-3-0. It contains samples from cortex, hippocampus, and the subventricular zone and features 27 998 transcripts. Cell type annotations were approximated using CellTpyist with the 'Developing_Mouse_Brain' reference model and majority voting. This resulted in 27 distinct cell types.

2.3 Performance metrics

2.3.1 Clustering metric

We use the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) for evaluating and comparing clustering performance of our models. The value of the metric ranges from zero to one, representing random and identical clustering, respectively. This metric is relatively robust to different clustering approaches and diverging numbers of clusters, which makes it a good metric for comparing our method to others.

When analysing models without GMM priors, the latent spaces are clustered using k-means (Lloyd 1982) clustering implemented in Python's scikit-learn package.

2.3.2 Reconstruction metrics

For Fashion-MNIST, we use the BCE loss in order to evaluate the reconstruction of image pixels.

In scDGDs, we calculate the NLL as the summed log-density of parameterized Negative Binomials over all genes. For each gene, the log-density is calculated based on the re-scaled model output and a gene-specific, learned, dispersion factor.

2.3.3 Image generation evaluation metric

For a quantitative evaluation of generated images, we employ the Fréchet Inception Distance (FID) (Heusel *et al.* 2017). The FID gives the squared Wasserstein distance between two multivariate Gaussian distributions. We used the PyTorch implementation (Seitzer 2020) based on the original publication Heusel *et al.* (2017). We compute the FID score three times for a given model, with random seeds 0, 21, and 87 243. In each run, we randomly generate 10 000 samples and compute the FID score with respect to the Fashion-MNIST test set.

2.4 Software, statistics, and visualization methods

Python 3.8 is used as the programming basis for all methods. Neural Networks are implemented in Pytorch 1.10 (Paszke *et al.* 2019) and training progress was monitored and logged using wandb (Biewald 2020). Dimensionality reduction is either done with our described model or common methods, such as PCA or UMAP (McInnes *et al.* 2018). If not stated otherwise, default parameters of 15 neighbours and a minimum distance of 0.1 are used in UMAP fits. Graph visualizations are achieved using the NetworkX package (Hagberg *et al* 2008). Requirements can be found in the corresponding repository.

2.5 Hardware

Models were trained on NVIDIA TITAN Xp (12 GB), except for scVI on the mouse brain (1M) dataset, which was trained on NVIDIA TITAN RTX (24 GB).

3 Results

3.1 Demonstration on image data

It is customary to test generative models on image benchmarks, so we begin our experimental analysis by applying our model to the Fashion-MNIST data. It allows us to understand and evaluate model behaviour, latent space and most importantly sampling quality much better than more complex, biological data. The decoder used for the Fashion-MNIST models is based on the DCGAN (Radford *et al.* 2016) generator architecture with modifications using common methods in image generation and is described in detail in Section 2 and Supplementary Fig. S5. We limited the design space (Radosavovic *et al.* 2020) to a latent dimension of 20, as interpolation can only occur in low-dimensional representations (Balestriero *et al.* 2021). The remaining parameters and how they were chosen are described in Section 2. The model was trained for 500 epochs.

In Fig. 2b, we see that a GMM with 20 Gaussian components distributes well over the complex latent space. The resulting distribution over latent space (the parameterized GMM) models the latent representation much better than for GMMs with less components which fail to pick up on more subtle structures (Supplementary Fig. S2). This supports the hypothesis stated in many works that overly simple distributions, such as standard Gaussians may not be sufficient for describing the probabilistic representations of many data. Another advantage of the GMM is the increase in control over sampling. Figure 2a shows generated samples from individual components of the GMM. Having more components leads to each of them covering a much smaller area of the latent space, and thus offering less varied but more specific samples. In this case of a 20D latent space with 20 mixture components, each component mean can be attributed to a (sub-)class of Fashion-MNIST. A relatively high number of components can thus enable more deliberate, more qualitative sampling, and a better model of the latent space. While the latent representations in Fig. 2b show clear separation for super-classes, such as shoes (Ankle boot, Sneaker, and Sandal) and some distinct classes like of trousers and bags, other classes are much more mixed. We evaluate the clustering with the adjusted Rand index (ARI) (Hubert and Arabie 1985). This metric achieves values between 0 and 1 with 1 representing a perfect clustering and is corrected for chance. The ARI of this clustering based on predicted labels derived from the GMM's component probabilities is only 0.295. As an extension of the DGD, it is also possible to perform supervised training, where each GMM component is assigned a data class a priori. In the optimization, each component is only committed to covering the latent space of its assigned classes' representations. When trained with these assigned 10 components, we arrive at the distinct latent clusters seen in Fig. 2c with an ARI of 0.995 (which is merely a sanity check, since the ARI is expected to be high). While the quality of random samples from this model is reduced slightly (Supplementary Fig. S3) in comparison to those from an equivalent unsupervised model, we achieve perfect control over the sample identity.

We next investigated how the MAP approach compares to VI and amortization. For this purpose, we limit the prior of the DGD to a fixed standard Gaussian. We found that once the encoder of a VAE is removed [using the variational auto-Decoder (VAD) (Zadeh *et al.* 2021)], reconstruction performance improves significantly and is equal to that of the DGD, as seen in Fig. 2d, Table 1, and Supplementary Table S1. The latter also shows that latent space normality is improved by removing the encoder. This supports Schuster and Krogh (2021a), Bojanowski *et al.* (2018), and Bond-Taylor and Willcocks (2021) in their results stating that the encoder can be a hindrance to achieving good representations. Given the standard Gaussian priors, the structure of the latent spaces is not practically changed between the three models (Supplementary Fig. S4), even though ARIs vary a lot (Supplementary Table S1).

Our demonstration of the DGD on image data comes in very handy when next evaluating the generative modelling capabilities of the different approaches. The benefit of image data is that it comes naturally to us to evaluate whether generated data is sensible or not from a qualitative standpoint. In addition, there are many quantitative measures available. We make use of the Fréchet Inception Distance (FID) (Heusel *et al.* 2017), a popular metric of the similarity between image distributions. It is, however, a biased metric depending on the number of samples and the generator itself. A qualitative

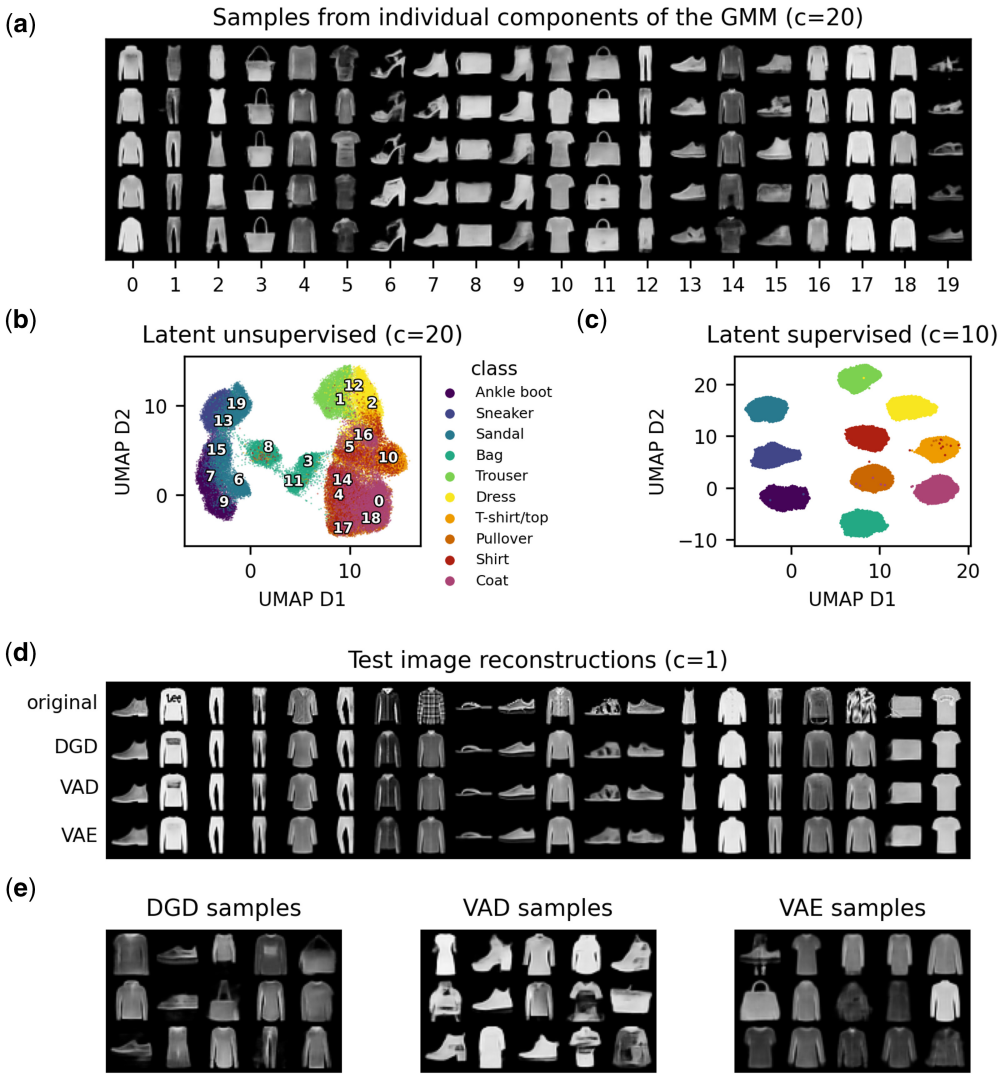


Figure 2. Fashion-MNIST latent representations and samples. The dimension of all latent spaces included in this figure is 20. The number of components of the GMM are denoted as 'c'. (a) Generated images of samples drawn from each component of the DGD with 20 Gaussian components. The number of the component drawn from is depicted on the *x* axis. (b and c) Latent representations visualized as UMAP dimensions 1 and 2 coloured by data classes. The UMAP was computed with 50 neighbours and a minimum distance of 0.7. (b) Latent representation from the model with 20 GMM components. Numbers in the plot show the positions of the corresponding component means, from which samples were drawn in (a). (c) Latent representation from the supervised DGD with 10 components. (d) Test image reconstruction from DGD, VAD, and VAE. The top row shows the first 20 original test images. The remaining three rows show test image reconstructions. Names of the corresponding models are depicted on the left. The indication of one component refers to a standard Gaussian. (e) Randomly sampled images from DGD, VAD, and VAE. Corresponding models are indicated by plot titles.

Table 1. Comparison of performance metrics for DGD, VAD, and VAE with latent dimension 20 and one fixed standard Gaussian as latent prior.[a]

Model	BCE loss	FID	Run time (s/epoch)
DGD	**0.2629 ± 0.001**	37.6 ± 0.2	**25.2 ± 2.7**
VAD	**0.2615 ± 0.001**	44.3 ± 0.2	35.7 ± 1.6
VAE	0.2798 ± 0.001	**33.9 ± 0.1**	36.5 ± 10.3

[a] Model names are indicated on the left. Performance metrics computed are indicated as the remaining columns. The BCE loss is the average binary cross-entropy loss of the reconstructed images, indicated with its standard error. The FID score is the FID, a metric of the distance between generated and original image distributions. Both metrics are computed with respect to the test images. Reported means and errors stem from repeats with three different random seeds. We additionally report run times based on the same model architectures as an average over runs for three different latent dimensions (20, 50, and 100) and 1000 epochs per run along with the standard error of the mean. The same hardware was used for all runs. Bold numbers highlight best performing models for each metric.

analysis of the samples from all models (Fig. 2e) ranks the DGD samples the highest in terms of realisticness and detail. However, the FID score of the VAE samples are slightly better (Table 1) despite the generated images being very generic and blurry.

The FID score of the DGD improves drastically with learning a more complex parameterized distribution, which lends itself naturally to our approach. When using 20 Gaussian components, the FID decreases to 27.25 ± 0.11 (from three computations using different random seeds, see Section 2), positioning itself between the probabilistic autoencoder (28.0) (Böhm and Seljak 2022) and PeerGAN (21.73) (Wei *et al.* 2022).

3.1.1 Application to single-cell data

In order to assess the model performance on scRNA data, we chose to apply it to the PBMC data from Zheng *et al.* (2017).

The single-cell expression counts are modelled by a Negative Binomial distribution. We achieved best clustering results with a latent dimension of 20 and 3 fully connected hidden layers. The model, which we will refer to as single-cell DGD (scDGD), was trained for 700 epochs after which it achieved its overall best performance in terms of reconstruction and clustering. The number of GMM components was set to nine, which represents the number of cell types in the data. Details about our implementation and how we arrived at these hyperparameters can be found in Section 2 and in the Supplementary Material.

The resulting latent space is depicted as a UMAP projection in Fig. 3. From a visual perspective, the latent space and GMM components cluster well with the cell types. For clustering purposes, a cell is assigned to the component that gives the largest probability to its representation. The ARI of this clustering on the train set is 0.618. In the work of Grønbech *et al.* (2020), scVAE was reported to have a higher clustering

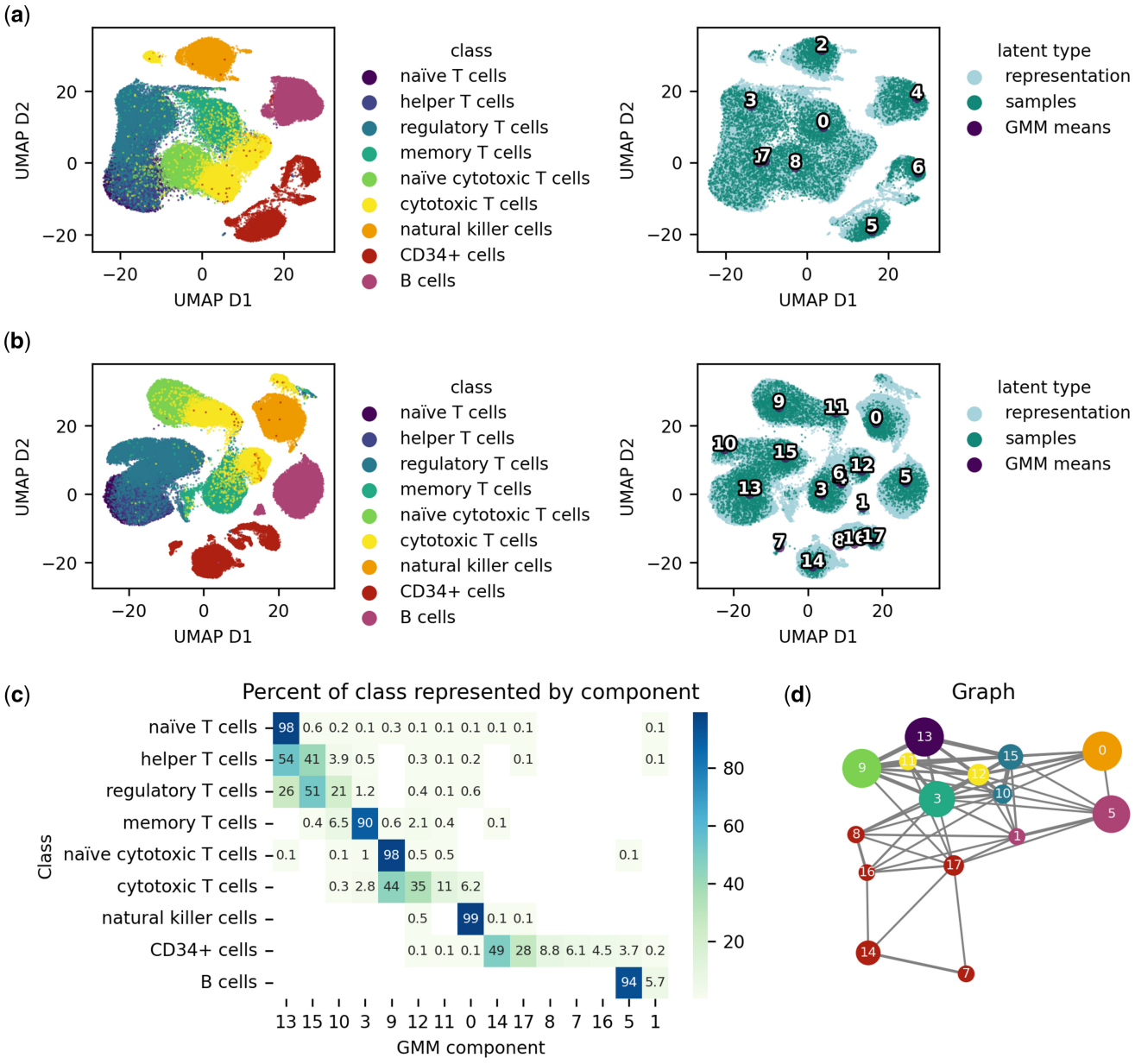


Figure 3. Latent spaces of scDGD. The latent spaces are shown coloured by cell type (left) and type of latent point (right). Visualizations are achieved using UMAP with a spread of 5 and a minimum distance of 1. 'Representation' refers to learned representation of training data, 'samples' correspond to random samples drawn from the GMM and 'gmm means' represents the component means. The IDs of the component means are shown in text on their corresponding coordinates. (a) Latent space of scDGD with nine Gaussian components. This model was trained for 700 epochs. (b) Latent space of scDGD with 18 components, trained for 600 epochs. (c–e) Spatial relationships between GMM components of the 18-component scDGD. (c) Heatmap of the GMM's component assignment to samples given as the percentage of each class with highest probability of a component. (d) Graph visualization of the GMM component means with edge lengths correlated with Euclidean distances from (a) and edge widths negatively correlated with Euclidean distances. Only the 95th percentile of distances were accepted as graph edges, resulting in a threshold of 0.14. Edge colours show components belonging to the same cell type [same colours as in (b)], except blue, referring to all CD4 cells minus memory T cells. Edge widths are correlated with proximity of clusters.

performance than scVI of around 0.66 compared to 0.53 for scVI. However, scVI clearly outperformed the Gaussian Mixture VAE in terms of reconstructions. Since the scVAE tool could not be retrained, we as well compare our model to scVI and include the Leiden algorithm (Traag *et al.* 2019) as a baseline for clustering performance. The clustering performance of scVI is evaluated based on *K*-means clustering of the latent space with nine components.

As a baseline for clustering performance, we apply the Leiden algorithm commonly used in the field of single-cell data. At nine components, this achieves an ARI of 0.51. A default scVI model trained on our train set resulted in a clustering performance of 0.566, which is derived from *K*-means clustering with nine components. The reconstruction performance of scVI is on par with our model trained with nine components in terms of goodness of fit, but under-performs in reconstruction accuracy (Table 2). The reconstruction performance is evaluated based on two metrics. Firstly, we compute the negative log-likelihood (NLL) of the true held out test counts being drawn from the gene-specific Negative Binomial distributions. These distributions are parameterized by the models' predicted means and learned dispersion parameters. Secondly, we also compute the root mean squared errors (RMSE) as a standard metric for comparability. We include both metrics for a more comprehensive understanding of the reconstruction performance. While the NLL measures the goodness of fit of the probabilistic modelling of counts and leaves more flexibility for genes that are highly variable, the RMSE emphasises the reconstruction accuracy and highlights sensitivities to outliers in the modelling. The clustering of scVI derived from the *K*-means algorithm with nine components, however, is still lower than that of scDGD (Table 2). This shows that with scDGD, there is no need for a compromise with respect to built-in, high-performance clustering and count modelling.

We also trained a model with 18 Gaussian components, since we have learned in our explorations on image data that increasing the complexity of the distribution over representation can be beneficial to modelling the substructures in the representation if the problem is complex enough. Increasing the number of components resulted in equal reconstruction performance as seen in Table 2 and highly improved clustering performance with an ARI of 0.684. As we can see in Fig. 3c, several cell types have one or more components specifically assigned to them. Among these are CD4+ naïve and memory T cells, NK, CD34+, and B cells. The cytotoxic T cell sub-types cannot easily be differentiated by component assignment, but are distinct from all other cell types in the data. The same goes for all CD4+ T cells except memory cells. Since UMAP distorts the latent space, we also show a graph of the component means in Fig. 3d based on the Euclidean distances between them (Supplementary Fig. S7).

To go one step further in the analysis of our model, we next aimed to get an understanding of the structuring of the learned latent space. In both models with 9 and 18 components, we see that the CD34-positive cells form sub-clusters that are modelled by different Gaussian components. We therefore investigated the representations for CD34-positive cells learned by the 18-component scDGD in terms of some more fine-grained differentiation markers. We were able to determine a sub-cluster best modelled by component 17 (Fig. 4a), which is primarily occupied by cells expressing markers for the lymphoid lineage of haematopoietic stem cells (HSCs). This cluster further shows a distinction between bone marrow common lymphoid progenitors (CLPs) and cord blood CLPs, indicated by markers CD10 and CD7, respectively. Another sub-cluster, represented by components 8 and 16, is made up of several cells expressing CD18, which is a marker for myeloid cells regulating neutrophil production and a general marker for granulocytes. There is also a strong presence of CD14-positive cells in these clusters, suggesting an aggregation of monocytes and neutrophils. The biggest sub-cluster, modelled by component 14, suggests the presence of different sub-populations of the myeloid lineage of HSCs. We find gradual levels of CD41 to CD62L, which are markers for megakaryoblasts and myeloblasts, respectively. These are also correlated with component 7. We additionally found a concentrated site showing expression of different dendritic cell markers (CD1c, CD141, and CD303), best represented by components 8 and 16.

Lastly, we investigated the merits of scDGD in terms of user experience and applied both default scDGD and default scVI to 10 datasets that had no part in the development of scDGD. Details about these datasets from mouse and human and their preprocessing can be found in Section 2. Figure 5 shows that scDGD outperforms scVI in both data reconstruction (accuracy measured by RMSE) and latent clustering (in 9 out of 10 and 7 out of 10 cases, respectively). We also trained scVI and scDGD on the extremely large mouse brain dataset with roughly 1.3 million cells from 10× Genomics (see Section 2). On this large dataset, scVI outperforms scDGD in terms of RMSE, but achieves again lower clustering performance and requires more computational resources (Supplementary Table S2).

4 Discussion

In this work, we have presented the DGD, a Bayesian formulation of a deep generative neural network using MAP estimation for model parameters and representations. This is a fully tractable and simple approach, for which no encoder is needed. This gives the DGD the advantage of having fewer parameters and fewer modelling choices, such as encoder architecture. We have shown that the DGD achieves good reconstruction,

Table 2. Comparison of performance metrics for scDGD and scVI.[a]

Model	ARI	NLL	RMSE	Run time (min)
scDGD ($c = 9$)	0.618	2053.76 ± 8.58	**0.2246 ± 0.0007**	201.30
scDGD ($c = 18$)	**0.684**	2052.20 ± 8.51	**0.2249 ± 0.0006**	170.77
scVI ($c = 1$)	0.566	2053.99 ± 8.52	0.2305 ± 0.0007	200.05
Leiden	0.512	n/a	n/a	n/a

[a] Model names are indicated on the left, with *c* describing the number of Gaussians in the GMM (1 indicates a standard Gaussian). In the row for Leiden clustering, n/a denotes that the metric is not applicable for the given method. Performance metrics computed are indicated as the remaining columns. The NLL refers to the negative log-likelihood of the negative binomial distribution, which models the counts in all methods. The best values for each metric are highlighted in bold.

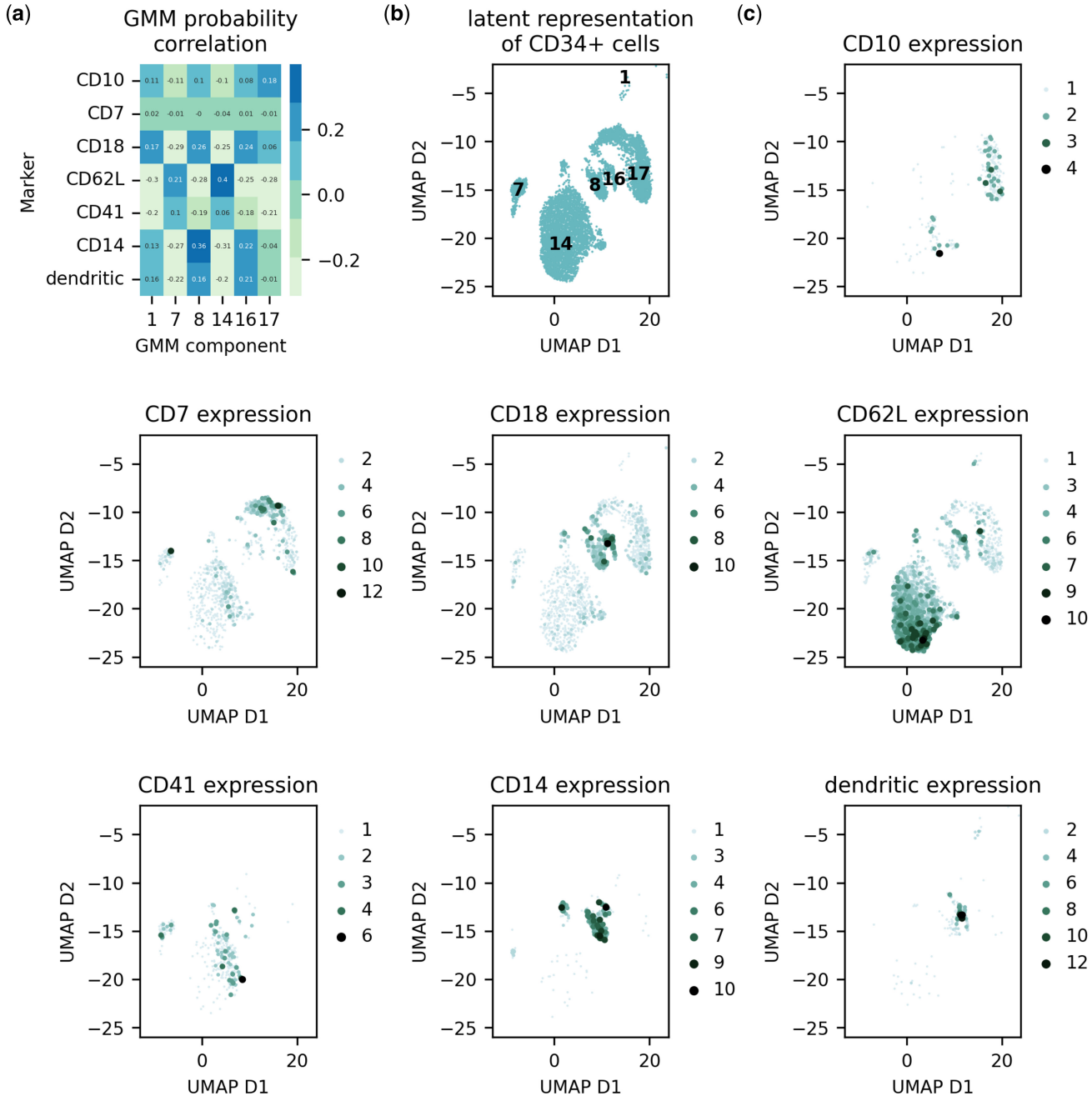


Figure 4. UMAP projection of CD34-positive cell latent representations learned by scDGD with 18 Gaussian components coloured by expression levels of different genes. (a) Heatmap of Spearman correlation coefficients between GMM component probabilities and marker expression counts for all CD34-positive samples. (b) Representations and component means (numbers) in UMAP projection of all CD34-positive cells on black background. (c) UMAP representation projection coloured by expression counts for gene markers indicated by the plot titles.

clustering, and generative performance on much smaller latent spaces than some of the other generative approaches, such as VAEs. In this sense, it is comparable to flow-based models (Xiao *et al.* 2019), but with less architectural restrictions. Additionally, we see clear advantages of this approach due to the simplicity of incorporating more complex distributions for modelling the latent representation. Analogous to our results on Fashion-MNIST, others have reported much better clustering and generative performances with a mixture of Gaussians compared to classic VAEs with standard Gaussians as priors over latent space (Grønbech *et al.* 2020).

In our application to single-cell expression counts, we showed that we can achieve reconstruction and clustering performances superior to those of comparable models, but with much smaller and better structured latent spaces, fewer parameters, and more detailed distributions. Our choice for comparison are scVI (Gayoso *et al.* 2022) and scVAE (Grønbech *et al.* 2020), two VAE-based methods modelling the representation of single-cell expression data. Overall, we compare scVI and our model on a total of 12 datasets, ranging from 500 to over 1 million cells. In these experiments, scDGD largely outperforms scVI on both reconstruction and clustering

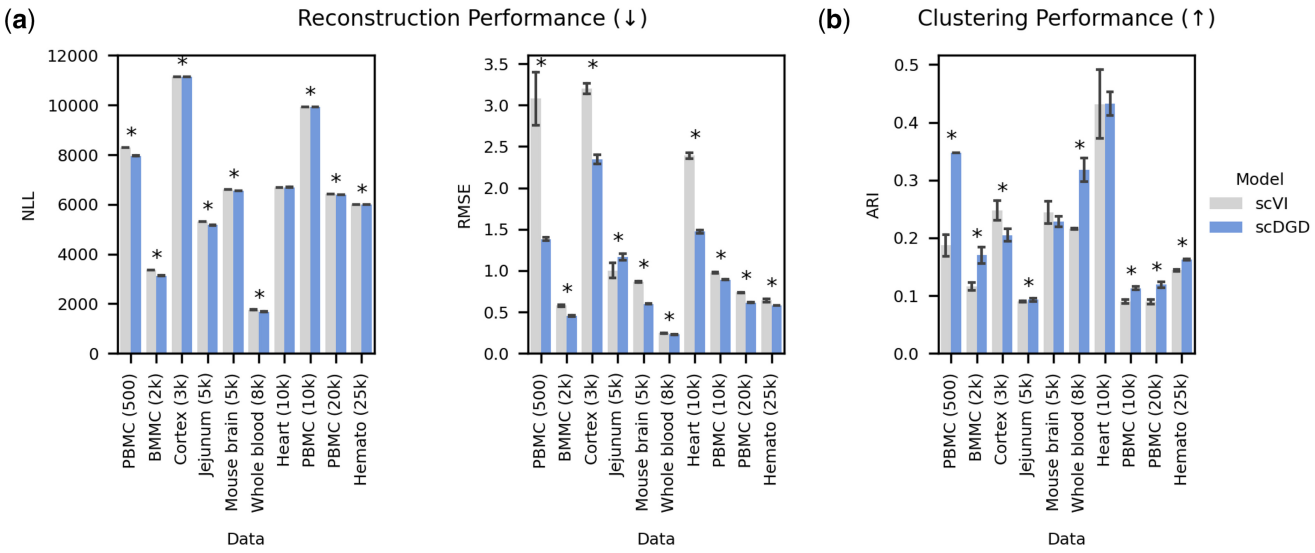


Figure 5. Performance comparison of scVI and scDGD on independent datasets. Error bars indicate the standard error of the mean from three models trained with different random seeds. Asterisks indicate whether differences are significant based on an alpha value of 0.05. (a) Reconstruction performance measured as the NLL of the negative binomial distribution and the RMSE. (b) Clustering performance measured by the ARI

performance. Investigating one of the learned representations closer, we find that the structure shows meaningful relationships and sub-clustering by Gaussian components, which can be linked to different sub-populations in some cell types.

A caveat of this model is the inference time of new data points. The lack of an encoder means that the representations of new data points need to be found by back-propagation (with fixed model parameters). This process is more time-consuming and computationally expensive than passing new data points through an encoder as in the VAE-based model. However, in our experiments this only took 0.93 s to 36.24 min for 59–130 613 test cells, respectively. In the case where inference is done repeatedly, one could also train an encoder afterwards as described in Schuster and Krogh (2021b), which still results in a more expressive representation than the traditional AE setup.

Altogether, we find that the DGD is a simple and efficient approach to learning low-dimensional representations. Unlike some other generative models, it is fully tractable and bears no architectural restrictions. Although the model can be supervised as we showed on Fashion-MNIST, our scRNA model is completely unsupervised and is suitable for data without existing labels, while achieving a meaningful clustering of the representation on its own. We also observed that a more complex distribution relating to more GMM components can be beneficial for discovering more substructures in the data than provided by class labels. This makes the DGD an ideal candidate for modelling a multitude of biological and medical data. For these types of data, interpretability of the latent space is crucial. Several downstream applications, such as perturbations and pseudo-time, require a low-dimensional and continuous representation. We plan to expand the model to a variety of biological tasks and to include semi-supervised learning and other features that are of interest to generative models for biological and medical data. We further aim to apply this approach to even more extensive single-cell expression datasets and multi-omics data that allow a more thorough investigation of the latent space and can provide meaningful tools for data integration and downstream analysis.

Acknowledgements

We acknowledge all the great discussions on representation learning and generative models with Wouter Boomsma, Jes Frellsen, Søren Hauberg, Aasa Faragen, Ole Winther, and other members of Center for Basic Machine Learning Research in Life Science (MLLS), as well as support from our Center of Health Data Science on discussions about the methods and single-cell data (especially Iñigo Prada Luengo) and for reviewing our manuscript (Jennifer Anne Bartell, Iñigo Prada Luengo, and Thilde Terkelsen). We especially thank Sarah Teichmann, her lab and Emma Dann for the insight we have gained into single-cell sequencing data, which was invaluable in the reviewing process.

Author contributions

A.K. contributed to the theoretical foundation of the approach. V.S. contributed to the experiments and figures. A.K. and V.S. contributed to the implementation of the approach and the writing of the manuscript.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

This work was supported by grants from the Novo Nordisk Foundation [NNF20OC0062606, NNF20OC0059939, NNF 20OC0063268 to A.K.].

Data availability

No datasets were generated during this study. All data used in this work are publicly available and referenced inMaterials and methods.

Code availability

scDGD is available as a python package at https://github.com/Center-for-Health-Data-Science/scDGD. The remaining code is made available here: https://github.com/Center-for-Health-Data-Science/dgd.

References

Abukmeil M, Ferrari S, Genovese A *et al.* A survey of unsupervised generative models for exploratory data analysis and representation learning. *ACM Comput Surv* 2021;**54**:1–40. https://doi.org/10.1145/3450963.

Ahmed S, Rattray M, Boukouvalas A. GrandPrix: scaling up the Bayesian GPLVM for single-cell data. *Bioinformatics* 2019;**35**:47–54. https://doi.org/10.1093/bioinformatics/bty533.

Bai J, Kong S, Gomes CP. Gaussian mixture variational autoencoder with contrastive learning for multi-label classification. *Proceedings of the 39th International Conference on Machine Learning*. Baltimore, USA: PMLR, 2022, **162**:1383–1398.

Balestriero R, Pesenti J, LeCun Y. Learning in high dimension always amounts to extrapolation. arXiv, arXiv:2110.09485 [cs], 2021, preprint: not peer reviewed.

Biewald L. Experiment tracking with weights and biases. Software available from wandb.com. 2020.

Bojanowski P, Joulin A, Lopez-Pas D *et al.* Optimizing the latent space of generative networks. In: Dy J, Krause A, *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden*, PMLR, 2018, **80**:600–9.

Böhm V, Seljak U. Probabilistic autoencoder., TMLR, 2022.

Bond-Taylor S, Leach A, Long Y *et al.* Deep generative modelling: a comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022;**44**:7327–7347.

Bond-Taylor S, Willcocks CG. Gradient origin networks. In: *Ninth International Conference on Learning Representations, Vienna, Austria*, 2021.

Collins M, Dasgupta S, Schapire RE. A generalization of principal components analysis to the exponential family. *Advances in Neural Information Processing Systems*, Cambridge, USA: MIT Press, 2001, **14**:617–24.

Cremer C, Li X, Duvenaud D. Inference suboptimality in variational autoencoders. In: Dy J, Krause A, *Proceedings of the 35th International Conference on Machine Learning*, Vienna, Austria, PMLR, 2018;**80**:1078–86.

Dilokthanakul N, Mediano PAM, Garnelo M *et al.* Deep unsupervised clustering with Gaussian mixture variational autoencoders. arXiv, arXiv:1611.02648 [cs, stat], 2017, preprint: not peer reviewed.

Domínguez Conde C, Xu C, Jarvis LB *et al.* Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science* 2022;**376**:eabl5197. https://doi.org/10.1126/science.abl5197.

Gayoso A, Lopez R, Xing G *et al.* A Python library for probabilistic analysis of single-cell omics data. *Nat Biotechnol* 2022;**40**:163–6. https://doi.org/10.1038/s41587-021-01206-w.

Goodfellow I, Pouget-Abadie J, Mirza M *et al.* Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C *et al.* (eds), *Advances in Neural Information Processing Systems*, Red Hook, USA: Curran Associates, Inc., 2014, **27**:2672–80.

Grønbech CH, Vording MF, Timshel PN *et al.* scVAE: variational auto-encoders for single-cell gene expression data. *Bioinformatics* 2020;**36**:4415–22. https://doi.org/10.1093/bioinformatics/btaa293.

Guo C, Zhou J, Chen H *et al.* Variational autoencoder with optimizing Gaussian mixture model priors. *IEEE Access* 2020;**8**:43992–4005. https://doi.org/10.1109/ACCESS.2020.2977671.

Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux G, Vaught T, Millman J (eds), *Proceedings of the 7th Python in Science Conference*. 11–5. Pasadena, CA, USA, 2008.

Han T, Lu Y, Zhu S *et al.* Alternating back-propagation for generator network. In: Singh SP, Markovitch S (eds), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 1976–84. AAAI Press, 2017.

He K, Zhang X, Ren S *et al.* Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*. 770–8. IEEE, 2016. https://doi.org/10.1109/CVPR.2016.90.

Heusel M, Ramsauer H, Unterthiner T *et al.* GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon I, Luxburg UV, Bengio S *et al.* (eds), *Advances in Neural Information Processing Systems*, Red Hook, USA: Curran Associates, Inc., 2017, **30**:6629–40.

Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, Vancouver, Canada, Curran Associates, Inc., 2020, **33**:6840–51.

Hubert L, Arabie P. Comparing partitions. *J Classif* 1985;**2**:193–218. https://doi.org/10.1007/BF01908075.

Kammoun A, Slama R, Tabia H *et al.* Generative adversarial networks for face generation: a survey. *ACM Comput Surv* 2022;**55**:1–37. https://doi.org/10.1145/1122445.1122456.

Kingma DP, Welling M. Auto-encoding variational Bayes. *arXiv*, 1312.6114 [stat.ML], 2013. preprint: not peer reviewed.

Lalchand V, Ravuri A, Dann E *et al.* Modelling technical and biological effects in scRNA-seq data with scalable GPLVMs. *Proceedings of the 17th Machine Learning in Computational Biology meeting, Cambridge, USA*, PMLR, 2022, preprint: **200**:46–60.

Lawrence N. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, Cambridge, USA: MIT Press, 2003, **16**:329–36.

Lecun Y, Chopra S, Hadsell R *et al. A Tutorial on Energy-Based Learning*. Cambridge, USA: MIT Press, 2006.

Lloyd S. Least squares quantization in PCM. *IEEE Trans Inform Theory* 1982;**28**:129–37. https://doi.org/10.1109/TIT.1982.1056489.

Lopez R, Regier J, Cole MB *et al.* Deep generative modeling for single-cell transcriptomics. *Nat Methods* 2018;**15**:1053–8. https://doi.org/10.1038/s41592-018-0229-2.

McInnes L, Healy J, Saul N *et al.* UMAP: uniform manifold approximation and projection. *JOSS* 2018;**3**:861.

Mohamed S, Ghahramani Z, Heller KA. Bayesian exponential family PCA. *Advances in Neural Information Processing Systems*, ,Vancouver, Canada, Curran Associates, Inc., 2008, **21**:1089–96.

Mourragui SMC, Loog M, van Nee M *et al. Percolate: An Exponential Family JIVE Model to Design DNA-Based Predictors of Drug Response*. In: Tang H (ed.), Research in Computational Molecular Biology, Lecture Notes in Computer Science. Cham, Switzerland: Springer Nature, 2023, 120–38. https://doi.org/10.1007/978-3-031-29119-7-8.

Paszke A, Gross S, Massa F *et al.* Pytorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A *et al.* (eds), *Advances in Neural Information Processing Systems*. Red Hook, USA: Curran Associates, Inc., 2019, **32**:8024–35.

Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.

Radosavovic I, Kosaraju RP, Girshick R *et al.* Designing network design spaces. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, USA*, IEEE 2020.

Rezende DJ, Mohamed S. Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, JMLR, 2015, **37**:1530–38.

Schuster V, Krogh A. A manifold learning perspective on representation learning: learning decoder and representations without an encoder. In: *Entropy*, Basel, Switzerland: MDPI, 2021a;23(11):1403.

Schuster V, Krogh A. The deep generative decoder: using map estimates of representations (v1). arXiv, 2110.06672, 2021b, preprint: not peer reviewed.

Seitzer M. PyTorch-FID: FID Score for PyTorch. Version 0.3.0. 2020.

Seninge L, Anastopoulos I, Ding H *et al.* VEGA is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. *Nat Commun* 2021;**12**:5684. https://doi.org/10.1038/s41467-021-26017-0.

Townes FW, Hicks SC, Aryee MJ *et al.* Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biol* 2019;**20**:295. https://doi.org/10.1186/s13059-019-1861-6.

Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* 2019;**9**:5233. https://doi.org/10.1038/s41598-019-41695-z.

Tusi BK, Wolock SL, Weinreb C *et al.* Population snapshots predict early haematopoietic and erythroid hierarchies. *Nature* 2018;**555**:54–60. https://doi.org/10.1038/nature25741.

Vahdat A, Kautz J. NVAE: a deep hierarchical variational autoencoder. In: *Advances in Neural Information Processing Systems*, Red Hook, USA: Curran Associates, Inc., 2020, **1650**:19667–79.

Vahdat A, Kreis K, Kautz J. Score-based generative modeling in latent space. In: Ranzato M, Beygelzimer A, Dauphin Y *et al.* (eds), *Advances in Neural Information Processing Systems*, Red Hook, USA: Curran Associates, Inc., 2021, **34**:11287–302.

van den Oord A, Kalchbrenner N, Kavukcuoglu K. Pixel recurrent neural networks. In: Balcan MF, Weinberger KQ (eds). *Proceedings of the 33rd International Conference on Machine Learning, Vol. 48 of Proceedings of Machine Learning Research*. 1747–56. New York, NY, USA: PMLR, 2016.

van den Oord A, Vinyals O, Kavukcuoglu K. Neural discrete representation learning. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, USA: Curran Associates, Inc., 2017, **30**:6309–18.

van der Maaten L, Hinton G. Visualizing high-dimensional data using t-SNE. *J Mach Learn Res* 2008;**9**:2579–605.

Vaswani A, Shazeer N, Parmar N *et al.* Attention is all you need. In: *Advances in Neural Information Processing Systems, Red Hook, USA*, Curran Associates, Inc., 2017.

Verma A, Engelhardt BE. A robust nonlinear low-dimensional manifold for single cell RNA-seq data. *BMC Bioinformatics* 2020;**21**:324. https://doi.org/10.1186/s12859-020-03625-z.

Virshup I, Rybakov S, Theis FJ *et al.* anndata: annotated data. bioRxiv, 10.1101/2021.12.16.473007, 2021.

Wali A, Alamgir Z, Karim S *et al.* Generative adversarial networks for speech processing: a review. *Comput Speech Lang* 2022;**72**:101308. https://doi.org/10.1016/j.csl.2021.101308.

Wei J, Liu M, Luo J *et al.* DuelGAN: a duel between two discriminators stabilizes the GAN training. In: *Computer Vision - ECCV: 17th European Conference, Tel Aviv, Israel*, 2022:290–317.

Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv, 1708.07747, 2017.

Xiao Z, Yan Q, Amit Y. Generative latent flow. arXiv, 1905.10485, 2019.

Xu C, Prete M, Webb S *et al.* Automatic cell type harmonization and integration across Human Cell Atlas datasets. bioRxiv, 10.1101/2023.05.01.538994, 2023.

Yacoby Y, Pan W, Doshi-Velez F. Failures of variational autoencoders and their effects on downstream tasks. In: *ICML Workshop on Uncertainty in Deep Learning*, 2020;**1**:1–39.

Zadeh A, Lim Y-C, Liang PP *et al.* Variational Auto-Decoder: a method for neural generative modeling from incomplete data. arXiv, arXiv:1903.00840 [cs, stat], 2021, preprint: not peer reviewed.

Zeisel A, Muñoz-Manchado AB, Codeluppi S *et al.* Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* 2015;**347**:1138–42. https://doi.org/10.1126/science.aaa1934.

Zheng GXY, Terry JM, Belgrader P *et al.* Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 2017;**8**:14049. doi: 10.1038/ncomms14049.

10 | Paper III

multiDGD: A versatile deep generative model for multi-omics data

Viktoria Schuster[1], **Emma Dann**[2], **Anders Krogh**[1,3,*], **and Sarah A. Teichmann**[2,4,*]

[1] Center for Health Data Science, University of Copenhagen, Blegdamsvej 3B, 2200 Copenhagen, Denmark
[2] Wellcome Sanger Institute, Wellcome Genome Campus, Hinxton, CB10 1SA Cambridge, United Kingdom
[3] Department of Computer Science, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark
[4] Theory of Condensed Matter Group, The Cavendish Laboratory, University of Cambridge, JJ Thomson Avenue, CB3 0HE Cambridge, United Kingdom

* Corresponding authors: Anders Krogh (akrogh@di.ku.dk) and Sarah A. Teichmann (st9@sanger.ac.uk)

Publication details

This manuscript was under review at the time of submission.

Supplementary data are available at *bioRxiv* online under DOI https://doi.org/10.1101/2023.08.23.554420.

multiDGD: A versatile deep generative model for multi-omics data

Viktoria Schuster[1], Emma Dann[2], Anders Krogh[1,3*] and Sarah A. Teichmann[2,4*]

[1]Center for Health Data Science, University of Copenhagen, Blegdamsvej 3B, Copenhagen, 2200, Denmark.

[2]Wellcome Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB10 1SA, United Kingdom.

[3]Department of Computer Science, University of Copenhagen, Universitetsparken 5, Copenhagen, 2100, Denmark.

[4]Theory of Condensed Matter Group,The Cavendish Laboratory, University of Cambridge J J Thomson Avenue, Cambridge, CB3 0HE, United Kingdom.

*Corresponding author(s). E-mail(s): akrogh@di.ku.dk; st9@sanger.ac.uk;
Contributing authors: viktoria.schuster@sund.ku.dk; ed6@sanger.ac.uk;

Abstract

Recent technological advancements in single-cell genomics have enabled joint profiling of gene expression and alternative modalities at unprecedented scale. Consequently, the complexity of multi-omics data sets is increasing massively. Existing models for multi-modal data are typically limited in functionality or scalability, making data integration and downstream analysis cumbersome. We present multiDGD, a scalable deep generative model providing a probabilistic framework to learn shared representations of transcriptome and chromatin accessibility. It shows outstanding performance on data reconstruction without feature selection. We demonstrate on several data sets from human and mouse that multiDGD learns well-clustered joint representations. We further find that probabilistic modelling of sample covatiates enables post-hoc data integration without the need for fine-tuning. Additionally, we show that multiDGD can detect statistical associations between genes and regulatory regions conditioned on the learned representations. multiDGD is available as an scverse-compatible package (https://github.com/Center-for-Health-Data-Science/multiDGD).

1 Introduction

Single-cell genomics methods have become the main technology to study cellular heterogeneity and dynamics within tissues. They also enable the measurement of multiple molecular features within individual cells, pairing measurements of the transcriptome with epigenome, proteome or genome profiling. These paired multi-modal measurements can be used for deeper characterization of cell states, differentiation processes or genotype-to-phenotype relationships [1].

Analysis of paired single-cell multi-omics data typically requires joint dimensionality reduction on multiple molecular measurements to identify cell-cell similarities, cell states and patterns of co-variation between genomic features (also known as vertical integration [2]). Several statistical models have been proposed for this task, mostly based on factor analysis [3–5] or cell-cell similarity embeddings [6, 7]. Recently, approaches have been proposed to additionally integrate paired data from measurements of individual modalities (i.e. mosaic integration) [8–11]. However, existing methods have primarily been applied to relatively small data sets, while increasing availability of multi-modal data now requires models that can handle tens of thousands of cells from multiple experiments, with the ability to account for technical differences between samples [12]. Additionally, methods for vertical integration struggle with imbalance in the dimensionality of feature spaces, especially in the joint analysis of gene expression and chromatin accessibility over hundreds of thousands of genomic regions [2]. Importantly, as the field and its methodologies are still developing, existing analytical approaches predominantly focus on dimensionality reduction for the purpose of cell clustering, with notably little emphasis on identifying relationships between molecular features [1]. This is especially relevant for joint analysis of epigenomic and transcriptomic profiles to associate regulatory regions to changes in gene expression.

In order to alleviate the problems encountered with large data sets, generative models have been applied to both uni-modal [13–18] and multi-modal [8, 11, 19–21] data. Deep generative models are powerful machine learning techniques that aim to learn the underlying function of how data is generated. This is of special interest for unsupervised analysis of single-cell data, where the goal is to interpret patterns of variation in high-dimensional and noisy data [22]. The predominant type of generative model applied in this field is the Variational Autoencoder (VAE) [23]: models tailored for scRNA-seq data [13–16] enable integration of large and complex data sets at lower computational cost [24] and have been successfully applied to the analysis of cells across human tissues and in large cohorts [25–27]. These models do however come with some limitations [18], which are continuously being addressed by a large community. With the current state of model design, it is for example not trivial to integrate new samples from different batches after training as covariates are modelled via one-hot encoding. scArches [28] is a tool introduced to solve this problem post-hoc, but the proposed fine-tuning is rather a band-aid than a solution to the underlying problem.

While the number of generative models available is vast for scRNA-seq single-cell data [13–18, 21], the application to multi-modal single-cell data has just begun. Existing models often employ simplistic architectures, priors for the generative distribution and encoding of confounding covariates such as batch effects [8, 11, 19, 20]. The results are under-performing models which hold the noise in the data accountable[29]. Generative modelling can provide much more than a joint integration. As emergent properties, deep generative models can capture underlying relationships between variables and dynamics in high-dimensional data. Straightforward examples of this would be feature interactions and cell state transitions. These properties can be learned without explicit modelling. However, these promising applications of generative models are still under-explored, as many models focus only on a fraction of the actual feature space.

In this work, we propose a new generative model, multiDGD, which aims to provide a basis for improved data integration and analysis of feature interactions. The model is an extension of the Deep Generative Decoder (DGD) [18] for single-cell multi-omics data. Unlike VAE-based models, it uses no encoder to infer latent representations but rather learns them directly as trainable parameters, and employs a Gaussian Mixture Model (GMM) as a more complex and powerful distribution over latent space. This introduces several advantages. Firstly, an encoder limits the flexibility and quality of representations. A decoder alone can better recover representations close to the optimum and reduces the number of parameters in the model [30]. Secondly, our GMM as a distribution over latent space increases the ability of the latent distribution to capture clusters in comparison to the standard Gaussian used in applied VAEs. Another strength of the DGD is its data efficiency. As presented in [30], the encoder requires more data to be well defined than the decoder. Removing the encoder makes the model applicable to not only large but also small data sets. This also translates to the number of features that can be modelled, and makes the DGD amenable to model genome-wide chromatin accessibility data where feature selection is problematic and may not be desirable.

We demonstrate on real world applications that the DGD can learn meaningful representations of complex multi-modal data, with improved performance for dimensionality reduction, cross-modality prediction, and modelling of unseen batches without the need for fine tuning. Furthermore, we provide a proof-of-concept that multiDGD can be used to predict regulatory associations between genes and peaks based on *in silico* perturbation.

2 Results

2.1 The model

multiDGD is a generative model of transcriptomics and chromatin accessibility data. It consists of a decoder mapping shared representations of both modalities to data space, and learned distributions defining latent space. Fig. 1 shows a schematic of multiDGD with its training and inference processes.

The inputs to the decoder are the low-dimensional representations Z of data X. Instead of providing them through an encoder (as in the Variational Autoencoder [23]), they are learned directly as trainable parameters [30]. One of the extensions to the model is the inclusion of disentangled representations. As a result, we can model the 'molecular' representation of cells Z^{basal} separately from technical batch effects and sample covariates (Z^{cov}). Representations Z are concatenations of the two. Parameters ϕ and ϕ^{cov} represent the parameterized distributions over latent space. They could be represented by any differentiable distribution, but we chose Gaussian Mixture Models (GMMs), as we see them as a natural choice for data containing sub-populations and can provide unsupervised clustering.

Data is generated by feeding latent representations Z to the decoder. For every ith sample of N data samples (cells), there exists a corresponding representation z_i. The decoder consists of three blocks: the shared neural network (NN) θ^h, and the two modality-specific NNs θ^{RNA} and θ^{ATAC}. The modality-specific networks predict fractions of the total counts per cell and modality, y_{ij}. These are then converted into predicted means of Negative Binomial distributions modelling counts by multiplying with the total count s_i. The training objective is given by the joint probability $p(X, Z, \theta, \phi)$ [18], which is maximized using Maximum a Posteriori estimation [18]. Both the model and the inference process are explained in more detail in the Methods section and Supplementary Figure 6.

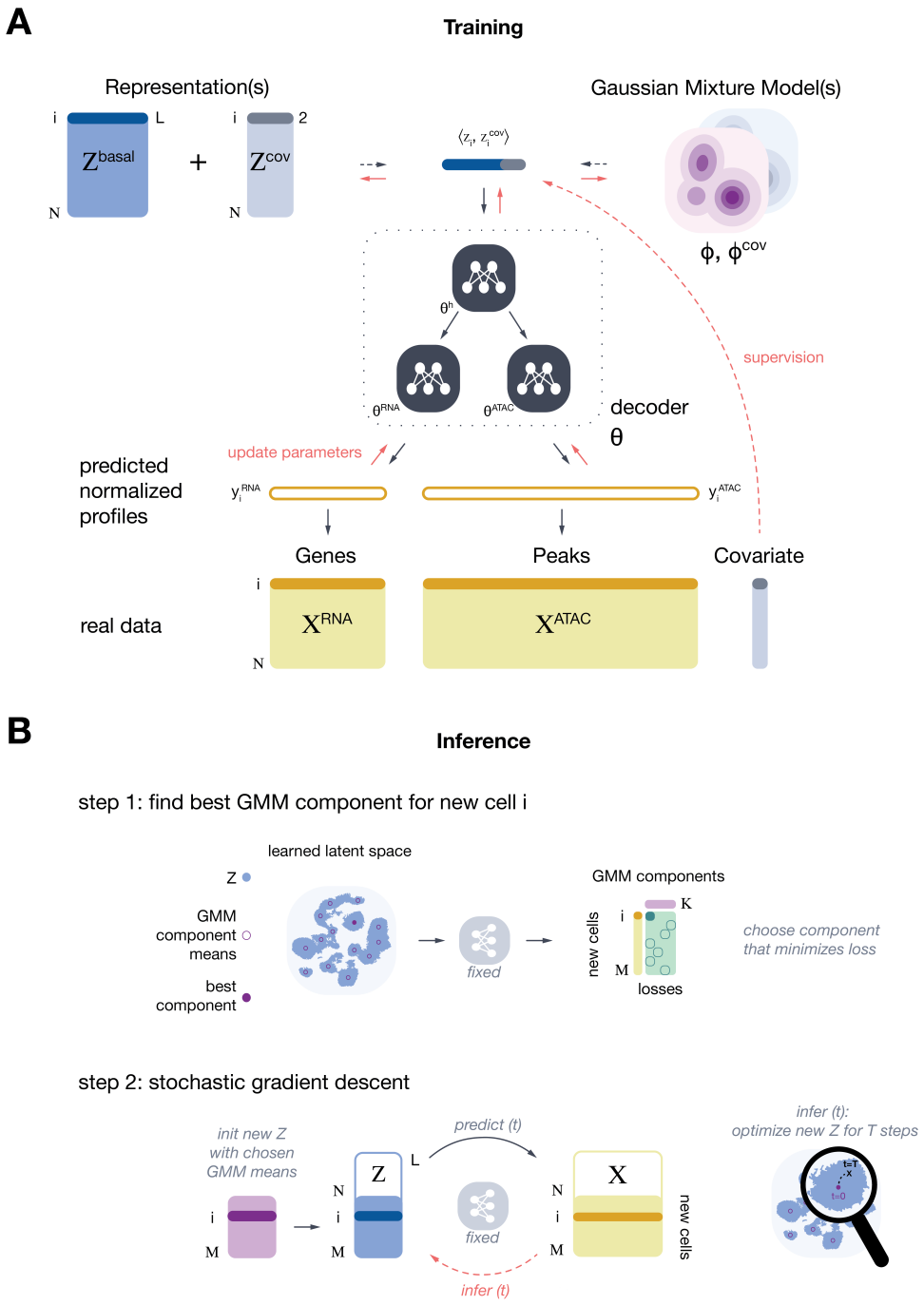


Fig. 1 Schematic and graphical model of multiDGD. A) Schematic of model architecture and generative process. Representations Z present the input to the decoder. They are distributed in latent space according to a Gaussian mixture distribution parameterized by ϕ. While Z^{basal} and ϕ are required, Z^{cov} and ϕ^{cov} are optional and depend on the presence of covariates to be modelled. For each data point (cell) $i \in N$, there exists a latent vector of length L, plus 2 dimensions for each covariate modelled. The input is transformed into modality-specific predicted normalized mean counts y through the hierarchical decoder θ. These outputs are then scaled with sample-wise count depths to predict the density of both RNA and ATAC data. Orange arrows depict the backpropagation and updating of parameters during training. **B)** Schematic of the two-step process of inferring new representations. In step 1, the best mode of ϕ is selected for each new sample $i \in M$ as the argmax of the probability densities of each component $k \in K$. In step 2, each initial representation is optimized for a specified number of steps T with fixed model parameters.

2.2 Improved performance on data reconstruction and clustering

We first evaluate this model's performance compared to VAE-based alternatives. Since MultiVI [8], a VAE-based multi-omics generative model, presents the only one which can integrate data from different batches and impute missing data, it is the main focus of our benchmark. Where applicable, we included performances of Cobolt [11] and scMM [20]. We compare performances for three different data sets of paired scRNA-seq and scATAC-seq, derived from human bone marrow [31], human brain [32] and mouse gastrulation [33] multi-omics data (see Methods). Comparing the data reconstruction performances on a held out test set, we found that multiDGD consistently outperforms MultiVI on all tested data sets (as well as scMM on the brain data) (Figure 2A-B). The improvement is especially large for the reconstruction of ATAC features.

We next evaluated the latent spaces learned by the models in terms of clustering of cell types and batch effect removal. MultiVI's shared embeddings of transcriptional and chromatin features are commonly used as input for the Leiden [34] clustering algorithm. The DGD intrinsically performs clustering with the Gaussian Mixture Model as the latent distribution (details in Methods sections 4.2.3,4.2.11). Measuring clustering performances with the Adjusted Rand Index (ARI) (Figure 2C), we see a notable variance in performance with random seeds for model initialization, more so for multiDGD than for MultiVI. However, the GMM components of multiDGD still learn latent representations whose clustering generally aligns better with the annotated cell type labels (compared to MultiVI, Cobolt and scMM). Figures 2E-F visualize the intrinsic clustering on the example of the human bone marrow benchmark data (remaining data sets are shown in Supplementary Figures 8 and 13). For some lineages, individual cell types are harder to separate than others. One of the difficult cell types in general are T cells [35]. Unlike many of the other cell types, the four subtypes do not cluster according to the four T-cell-associated GMM components. This might be due to limitations of RNA expression as a discriminant of T cell subtypes. Nevertheless, multiDGD clusters associated with T cells contain less spillover than Leiden clusters derived from MultiVI latent representations (Supplementary Figures 9 and 10). We see a clear distinction between naive CD8 T cells in component 8 and activated CD8 T cells in components 0 and 2. We also observe that very small cell type clusters, such as progenitors, can be aggregated in a single component. Other components do not seem correlated with cell type identity and may rather model outliers (components 9, 13).

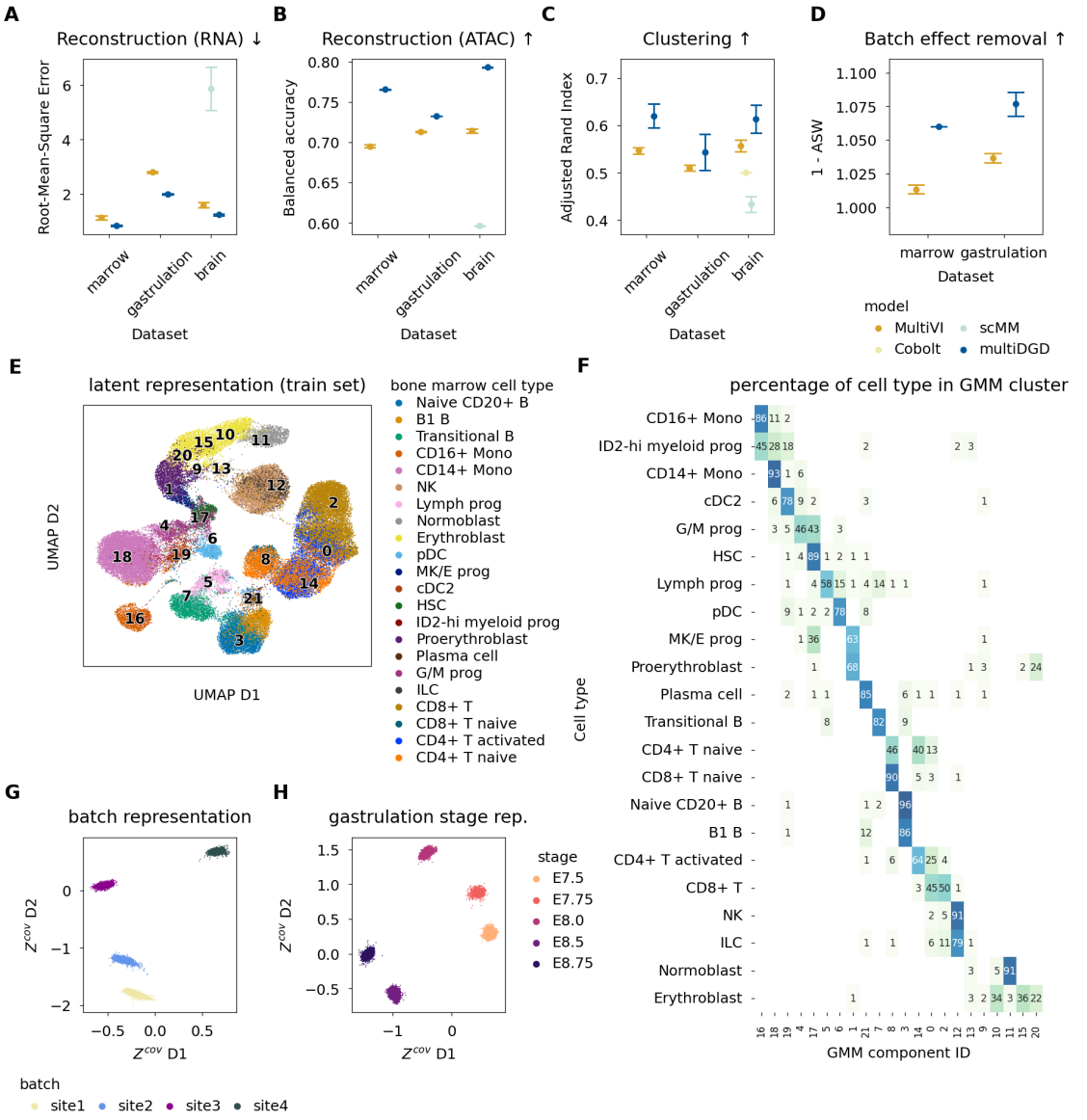


Fig. 2 High performance on reconstruction, clustering and the addition of probabilistic models for covariates. A-D) Performance evaluations were done on the three data sets marrow, gastrulation and brain, and three different random seeds for all models. Cobolt [11] and scMM [20] were included where applicable. Cobolt does not provide data reconstruction/imputation, and both Cobolt and scMM do not include batch correction. MultiVI [8] was used as the main comparison. **A)** Reconstruction performance on the test RNA data measured by RMSE. Lower is better. **B)** Comparison of the reconstruction performance on the test set ATAC data as the balanced accuracy of binarized predictions. Higher is better. **C)** Clustering performance of the train latent spaces. The metric is the ARI based on clustering derived from the GMM for multiDGD and Leiden clustering for MultiVI. The Leiden algorithm is adjusted so that the number of clusters in both models' latent spaces are comparable. More in Methods. Higher is better. **D)** Batch effect removal of marrow and gastrulation data was calculated as $1 - ASW$. Brain was not included as the data annotation contained no batch information. **E-G)** Latent space visualizations and clustering evaluation of the bone marrow latent spaces. **E)** UMAP visualization of the multiDGD latent representation. It is colored by annotated cell types. GMM component means are indicated by black numbers projected onto their transformed coordinates. **F)** Heatmap of the clustering of representation samples grouped by annotated cell types. Numbers inside the heatmap indicate the percentage of samples in a cell type assigned to a given component. **G)** Covariate representations colored by Site. The legend is the same as in H. **H)** Covariate representations of the gastrulation stage from mouse gastrulation data.

2.3 Probabilistic modelling of batch effects

Another important feature of generative models for single-cell data is the capability to alleviate batch effects. multiDGD leads to improved mixing between batches compared to MultiVI (Figure 2D and Supplementary Figure 8), although this is to be taken with a grain of salt as the average silhouette width may be skewed due to the different latent distributions. On our benchmark set, we see that the disentangled latent space results in clear separation of most cell types (Figure 2E and Supplementary Figure 8) and good mixture of the sites at which samples were processed (Supplementary Figure 11). The two-dimensional, separate representation for the batches derived from supervised training (Figure 2I) mirrors trends found in the general data distribution (see Supplementary Figure 12). These include site4 showing much more zero RNA counts than all other sites, which can explain why its cluster is distant from the others. In addition, we find that the covariate representation can capture biologically interpretable differences between samples. For example, when modelling the differences between embryos in the mouse gastrulation data set we see time-related structuring of the Gaussian components (Figure 2J). The early-to-late gastrulation phases from stage E7.5 to E8.0 [36] appear in chronological order, with stages E8.5 and E8.75 clearly separated. This distance makes sense as the differentiation of early organ progenitors is seen in stages E8.25 to E8.75 [36].

2.4 High performance on small data sets and many features

VAEs have clearly shown their usability and advantage when it comes to the speed at which they can model large data sets due to amortization, although this can come at the cost of posterior approximation [37]. Due to the missing encoder, the DGD is naturally suited for data sets with few samples and many features. Here, autoencoder-based models tend to overfit [30]. In this work, we briefly revisit this hypothesis by investigating the test performances of MultiVI and multiDGD trained on subsets of the human bone marrow set (Methods section 4.4.4). In order to put these into perspective, we compute average test loss ratios as the average test loss from the model trained on a subset over the test loss from the original model trained on the full set. Even though the variance in test loss ratios is much higher for multiDGD than for MultiVI (Figure 3), the average loss ratio of multiDGD stays stable for subsets larger than 1%, which corresponds to only 567 cells. MultiVI, on the other hand, performs worse with decreasing number of cells in the training data.

This advantage of the DGD also carries over to data with many features. Typically, in single-cell analysis feature selection is applied before performing dimensionality reduction [38], both for scalability and to increase clustering performance [31]. While robust methods to select highly variable genes exist for scRNA-seq, there are no robust statistical methods for feature selection in scATAC-seq data sets. Here, accessibility is usually measured over hundreds

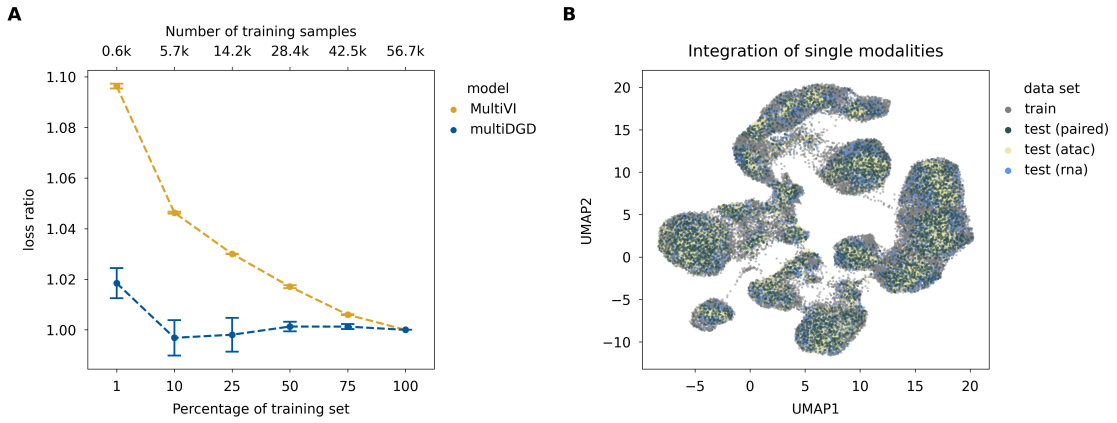


Fig. 3 multiDGD is highly data efficient and can impute missing modalities. **A)** Performances on data efficiency are reported on both multiDGD and MultiVI. Data efficiency was evaluated by training the models designed for human bone marrow data on a variety of subsets. This was repeated for three different random seeds, resulting in the depicted error bars as the standard error of the mean. **B)** UMAP visualization of the full bone marrow latent space colored by sample type. We included the train and test splits, and the artificial uni-modal test representations 'test (rna)' and 'test (atac)'.

of thousands of peaks, and several vertical integration methods suffer from this feature imbalance [2]. We compared multiDGD and MultiVI performance on data reconstruction in two scenarios on the mouse gastrulation data. The first one presents the previously presented models trained on data with feature selection (11792 genes, 69862 peaks), the second scenario presents models trained on all measured features (32285 genes, 192251 peaks). We compared performances on the shared set of features. While MultiVI lost performance for both modalities, multiDGD achieved nearly the same performance as before on ATAC data and even increased its performance on RNA data (Table 1).

Table 1 Feature efficiency on the mouse gastrulation set. Feature efficiency was investigated by training both models on two versions of the mouse gastrulation data. The first was the previously used set, where features had been selected for counts to be above zero for at least five percent of cells. The second version was the full data set. Performances were evaluated on the smaller set for comparability. We report the average performance metric with the standard error of the mean, RMSE for RNA data and balanced accuracy (BA) for ATAC data.

Model	RMSE (RNA)	BA (ATAC)	Feature set
multiDGD	$\mathbf{1.7146 \pm 0.0128}$	$\mathbf{0.7382 \pm 0.0007}$	5%
MultiVI	2.3429 ± 0.0212	0.7121 ± 0.0003	5%
multiDGD	$\mathbf{1.6939 \pm 0.0127}$	$\mathbf{0.0.7397 \pm 0.0007}$	all
MultiVI	2.4637 ± 0.0217	0.5197 ± 0.0001	all

2.5 Predicting and integrating unseen data

2.5.1 Predicting missing modalities

We next evaluated the performance of multiDGD for prediction/imputation of missing data of one of the two modalities (RNA or ATAC). Predicting data

modalities is a natural application of generative models for the case where existing uni-modal data is to be integrated with multi-modal data. In order to assess multiDGD's predictive capability, we test its performance on the held-out test set given only one modality. Imputations are achieved by optimizing the partial likelihood of the available data described in Methods section 4.2.10. Fig. 3B shows that representations inferred from either the original paired samples or the artificial uni-modal samples are well integrated into latent space. In order to assess the imputation performance of both models, we measured the relative errors of prediction (unseen modality) with respect to reconstruction (seen modality in the paired test set) (Methods section 4.4.6). This relative performance was similar for both multiDGD and MultiVI, although multiDGD shows a greater variance (Supplementary Table 4). However, the absolute prediction and reconstruction performances of multiDGD for ATAC data are still superior to those of MultiVI (Supplementary Table 5).

2.5.2 Integrating unseen batches without architectural surgery

A novel feature of the DGD is its capability to find representations for previously unseen data. This can simply be unobserved cells from the seen covariates, but also completely new data from unobserved covariates. The latter is possible thanks to the probabilistic modelling of both the desired 'molecular' and covariate components of the representation. We explore the quality of representations and predictions for unseen data by applying the leave-one-out method to train the model. For each batch in the human bone marrow data (defined as the site the data was processed at), we train a multiDGD instance on the training samples of all other batches, providing us with four models. We evaluate these models on their test performances in terms of prediction errors relative to the model trained on all batches. In Fig. 4A, we see a marginal increase in the prediction loss of unseen batches as expected, but overall prediction performance is on par with the model trained on all batches (Fig. 4B) and the unseen batch samples are well integrated into the latent space (Fig. 4D). So far, unseen covariates have been integrated with approaches such as architectural surgery (scArches [28]). We include a comparison to scArches applied to MultiVI in the same scheme. However, due to the need for a fine tuning set, we run scArches on the training portion of the held-out batch, in order to keep the test set independent. This, of course, gives MultiVI+scArches an advantange of additional data. For MultiVI+scArches, overall reconstruction error decreases compared to the MultiVI model trained on all batches, highlighting the nature of fine-tuning in scArches. Absolute performance metrics, however, are still inferior to multiDGD (Fig. 4C) and integration into latent space is equivalent (Figure 4D and Supplementary Figure 14), making post-hoc fine tuning obsolete.

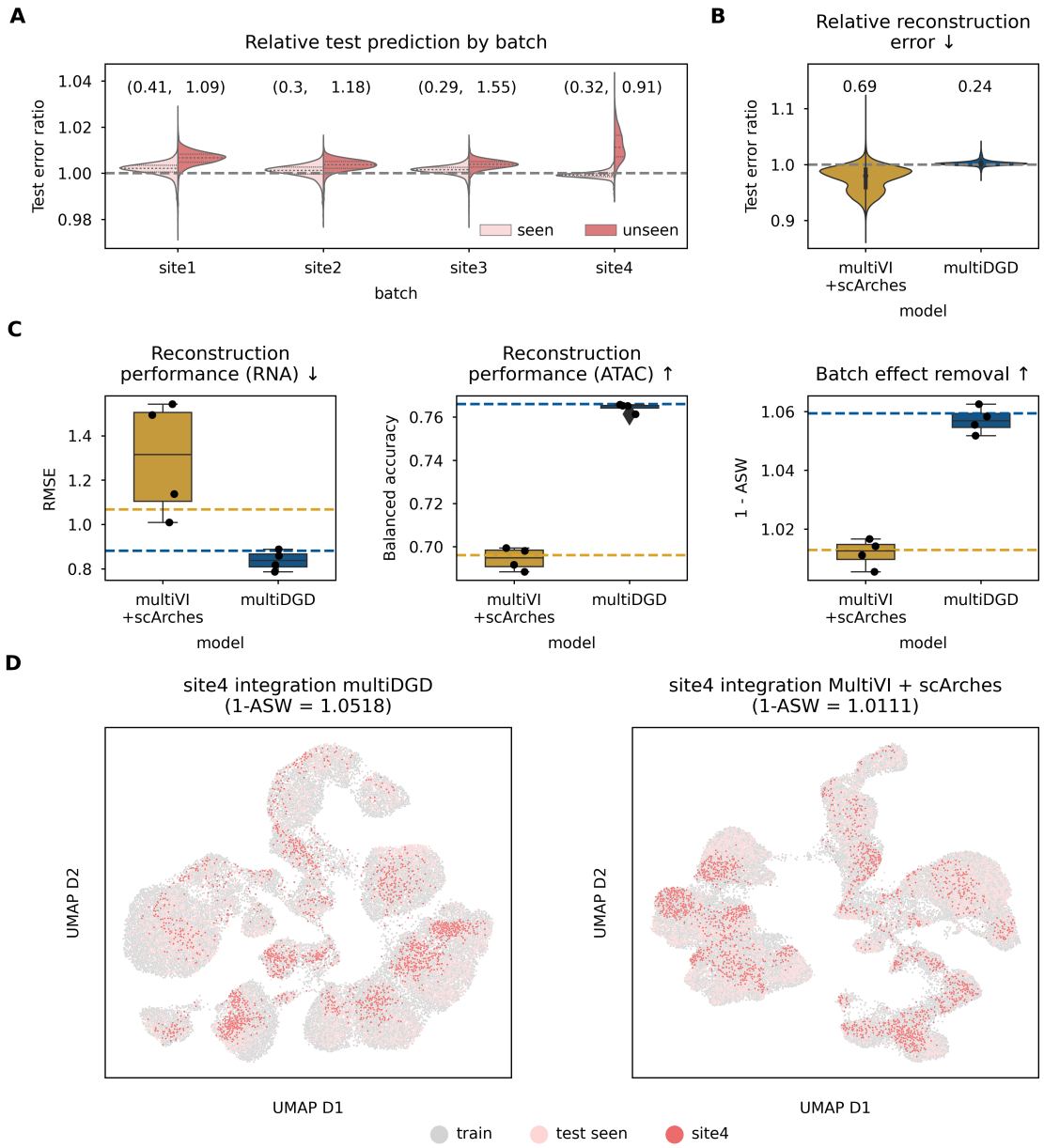


Fig. 4 Model fine-tuning is no longer needed to predict unseen covariates. All related experiments were performed on the bone marrow data set. For each site (presenting the batches), one model was trained that excluded the site from training. This resulted in four models. Comparisons are done on test predictions with respect to the model trained on all sites ('full'). scArches was applied with the left-out site from the train set to leave the test set independent. This results in a fine-tuned MultiVI model using all training data. **A)** Split violin plot of relative test errors for each multiDGD model wrt. the 'full' model. The relative errors are colored by whether the site had been included in training (seen) or not (unseen). **B)** Comparison of relative test errors for multiDGD and MultiVI fine-tuned with scArches. **A-B)** text above violins and boxes present the Kullback-Leibler divergences between the original test errors and the ones derived from the models trained on the incomplete data. **C)** Absolute performance comparison of multiDGD and MultiVI. From left to right: Reconstruction performances of multiDGD and MultiVI+scArches for RNA (I) and ATAC (II) data. Batch effect removal (III). Dashed lines present the original model performances for training on the full set. Box plots present the range of the performances of the four models from the leave-one-out scheme, which are shown as black dots. **D)** UMAP visualizations of latent spaces for multiDGD and MultiVI colored by the sample type. Samples were either taken from the train set or from the test set, which is distinguished by whether the batch was seen during training (seen) or not (unseen).

2.6 Gene-to-peak association with *in silico* perturbation

An emergent property of multiDGD is the learned connectivity between gene expression and chromatin accessibility data. We can use this to perform *in silico* predictions of where chromatin accessibility is associated with the expression of a given gene or set of genes. As depicted in Fig. 5A, we can compute gradients in latent space in the direction of perturbing a given gene X_i^{RNA} in data space. For every cell used, we have the original representation and a representation after one step of perturbation (Z^{KO}). From these representations, we predict the perturbed sample and calculate the differences in prediction $\Delta\hat{X} = \hat{X}^{KO} - \hat{X}$. $\hat{X}$ refers to model predictions.

First, we evaluated the ability of this *in silico* perturbation method to recover the association between gene expression and accessibility around the transcription start sites (TSS). When silencing a set of highly variable genes in the bone marrow data set, we observe significantly higher mean $\Delta\hat{X}$ at peaks in the proximity of the TSS as expected. $\Delta\hat{X}$ then gradually flattens for peaks over 2000 base pairs away (Figure 5B).

Next, we investigated whether we could recover associations between genes and peaks overlapping distal enhancers. As ground-truth data for gene-enhancer interaction, we used H3K27ac HiChIP data measuring physical contacts between active chromatin and promoters in primary CD4+T cells [39]. We predicted the effect on chromatin openness from silencing three genes with CRISPR-activation-validated T-cell-specific enhancers captured by HiChIP (CLEC16A, CD69, ID2) [39]. We observed high perturbation effect in naive CD4+T cells in several regions with HiChIP evidence (Figure 5C), which were not captured by HiChIP in a muscle cell line (negative control for cell-type-specific interactions). In addition, the enhancer prediction accuracy was significantly lower when considering perturbation effect in a different cell type (CD14+ monocytes) (Figure 5D). These results suggest that multiDGD is capturing cell-type-specific enhancer-gene links. In all cases, multiDGD prediction outperformed gene-peak associations derived from Spearmann correlation of gene expression and peak accessibility, where performance was close to random (Figure 5D, Supplementary Figure 15). We also observed instances of high $\Delta\hat{X}$ in absence of HiChIP signal (Figure 5). While these might be false positives driven by noise in the scATAC data, it is possible that multiDGD could be capturing indirect effects of gene expression on accessibility.

Finally, we sought to leverage *in silico* predictions with multiDGD to investigate the correspondence between the expression of transcription factors (TFs) and their effect on accessibility of peaks containing their DNA binding motifs. We tested the effect of silencing 41 TFs on chromatin accessibility, using a categorization of "activators" or "repressors" from Gene Ontology (GO) terms [40]. When measuring the perturbation effects at peaks containing TF binding motifs, we found that silencing activator TFs tends to lead to significantly higher fractions of closing peaks compared to silencing of annotated repressors (Figure 5E). For 36 out of 41 TF perturbations, we found a significant difference in mean $\Delta\hat{X}$ between peaks containing TF binding motifs and

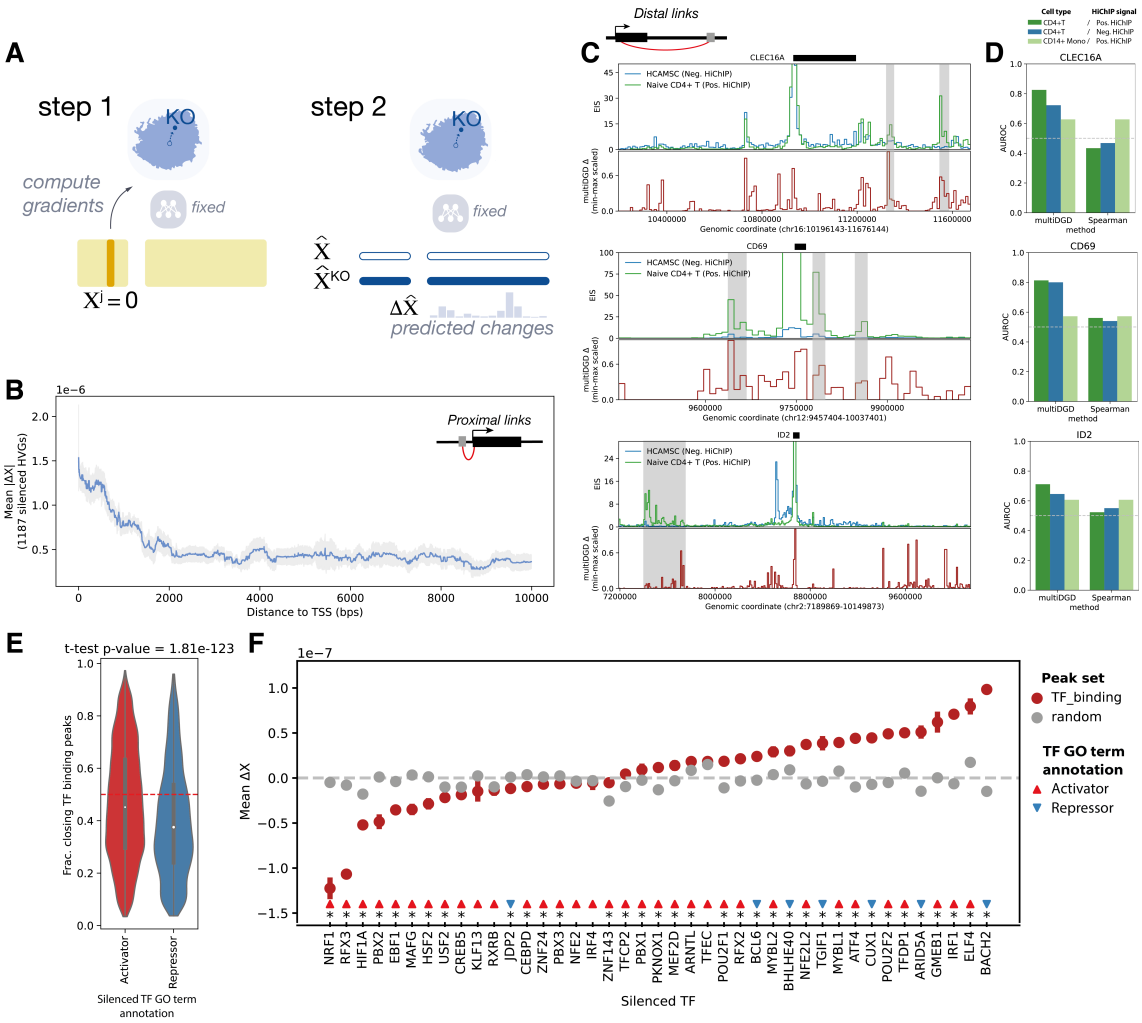


Fig. 5 Prediction of association between gene expression and peak accessibility.. A) Schematic of gene-peak association prediction with *in silico* perturbation. **B)** Mean absolute effect of perturbation on peak accessibility (ΔX, y-axis) within 10 kb window around transcription start site (TSS) of silenced gene. The rolling average and 95% confidence interval of perturbation effect over the distance to TSS is shown for silencing of 862 highly variable genes (HVGs), with window size of 100 bps. **C)** Comparison of HiChIP signal around 3 genes (CLEC16A, CD69, ID2) with ΔX from silencing of the gene in naive CD4+T cells. For each gene, the top track shows the Enhancer Interaction Score (EIS) calculated from HiChIP data using the gene promoter as viewpoint (see methods), for HiChIP on primary naive CD4+ T cells (Positive HiChIP, green) and on the muscle cell line HCAMSC (Negative control HiChIP, blue). The bottom track (red line) shows the scaled ΔX for the prediction of peaks associated with expression of the gene of interest. The location of the transcript for the gene of interest is shown on top of each plot. T-cell-specific enhancer regions are highlighted in grey. **D)** Barplots of Area under the Receiver Operating Characteristic Curve (AUROC, y-axis) for prediction of HiChIP enhancer regions from ΔX or Spearmann correlation between gene expression and peak accessibility in the selected cell type (x-axis). We show AUROC for predicted associations in bone marrow CD4+ T cells of CD4+ T cell HiChIP signal (dark green) or HCAMSC HiChIP signal (blue), and for predicted associations in CD14+ monocytes of CD4+ T cell HiChIP signal (light green). **E)** Violin plots of fractions of closing peaks ($\Delta X < 0$) for *in silico* silencing of transcription factors (TFs) annotated as activators (red) or repressors (blue). The black box and white point denote respectively the interquartile range and median of the distribution. The p-value for a 2-sample t-test comparing the mean of the distributions is shown. **F)** Mean ΔX in response to TF silencing for 34 predicted activators and 7 annotated repressors. We show the mean effect for all affected cells over a set of 10,000 peaks containing TF binding motifs (red points) or over a set of 10,000 peaks sampled at random amongst the peaks containing at least one TF binding motif (grey points) with the standard error of the mean. Asterisks denote TF perturbations for which the difference in mean effect between TF binding peaks and random peaks is significant (2-sample T-test p-value < 0.01).

matched random peaks (T-test p-value < 0.01). However, we observe broad variation in perturbation effects for different TFs. We measure the strongest chromatin closing effects in response to silencing (indicating a positive correlation between accessibility and expression) for TFs with reported activation effects on a broad set of metabolic genes involved in stress response, including NRF1 [41] and HIF1A [42]. For about one third of the TFs annotated as activators based on GO terms, we detected perturbation effects at TF binding peaks consistent with repressive activity (i.e. chromatin opening upon TF silencing). In several cases these discrepancies could be explained by conflicting GO terms, where the same TF is reported to have both activator and repressor function in different cellular contexts (e.g. ELF4 [43], IRF1 [44], POU2F1-2 [45]). We frequently observe perturbation changes in both directions for this class of TFs, with mean $\Delta\hat{X}$ varying between cell types (Suppl. Figure 16B). PBX1, for example, is generally regarded as a transcriptional activator in the context of cancer, but can play a dual role in hematopoiesis [46]. In our data this factor is specifically expressed in HSCs and erythroid progenitors (Suppl. Figure 16A). While its *in silico* silencing leads to chromatin closing in most cell types, we predict chromatin activation in HSCs and granulocytic-myeloid progenitors (G/M prog, GMP). This is in line with evidence from mouse studies suggesting that PBX1 KO leads to premature derepression of GMP transcripts in myeloid progenitors [46]. These results further support the idea that multiDGD is learning cell-type-specific patterns of regulation and that coupling the generative model with *in silico* perturbation can be used to interpret the interplay between molecular features in cells.

3 Discussion

We present multiDGD, a novel generative model for single-cell multi-omics data. We demonstrate its use as a tool for dimensionality reduction and cross-modality prediction, but also new functionalities. Firstly, it enables the integrated modeling of unseen batches without the need for fine tuning methods such as scArches [28]. Secondly, it contains a built-in analysis of gene-peak associations. This is possible due to the emergent properties of generative models, which enable us to combine integration and analysis of data in a single framework. We have thus focused on comparing to existing generative models for multi-omics data [8, 11, 20] in this work. Among these, however, only MultiVI provides functionalities such as batch effect removal, which is critical for our analysis. We show that multiDGD presents a strong improvement compared to MultiVI [8], which is based on VAEs and presents a popular generative model architecture used in single-cell analysis. Our model outperforms MultiVI in terms of data reconstruction, cross-modality prediction and cell type clustering. Part of the performance increase on modelling ATAC data may be due to the use of raw counts rather than binarised data, preserving more information [47]. However, we attribute much of the general performance increase in comparison to MultiVI to the more complex latent distribution and

the removal of the encoder. The performance increase in modelling RNA and ATAC counts can also be seen in the prediction of unseen cells and of missing modalities, although the variance in prediction performance is higher for multiDGD. One potential reason for this could be the over-denoising in MultiVI, which would also contribute to generally lower performance.

Even though multiDGD also outperforms the Leiden algorithm on MultiVI latent dimensions in terms of cell type clustering and learns meaningful embeddings, sampling from the prior over component means in the initialization leads to a high variance in clustering performance. We are eager to investigate how to stabilise this behaviour. Potential directions are to initialise the component means from origin as well or to re-sample representations during training to avoid local minima.

As single-cell multi-omics profiling becomes more robust and accessible, models for analysis will need to efficiently handle multi-sample data sets. They will further have to be able to disentangle technical batch effects from biological differences between cell types. We have introduced probabilistic modelling of covariates for the DGD. Probabilistic modelling of covariates improves basal representations, successfully captures interpretable sample-specific differences, and enables the integration of unseen data from different categories without architectural surgery. We have demonstrated this application and show that multiDGD can easily predict an unseen covariate with nearly the same performance as if it had been trained on it, but without any fine tuning. We believe this feature will facilitate the construction and re-use of large multi-omics atlases.

Multi-omics data sets, however, are often still small compared to scRNA-seq data sets [1] . The number of genomic peaks frequently outnumbers the amount of sequenced cells. Here we show that multiDGD shows clear advantages in modelling small data sets with high-dimensional spaces compared to data-hungry VAEs. We envision these capabilities will be of great value in allowing us to consider genome-wide epigenetic profiles for targeted analyses of data subsets of interest, such as specific lineages.

The goals of multi-omics analysis of course go way beyond efficient and high-quality embedding of cells. What is really desired is to further our understanding of gene regulation. Since we can incorporate larger decoders in multiDGD compared to VAE-based methods, explaining non-linear relationships may become easier. The resulting reconstruction performance increase certainly enables more reliable analysis at feature level. We made use of this in the prediction of gene-peak associations based on *in silico* perturbations. We demonstrated meaningful associations in both proximal and distal interactions, and showed that the model can capture the effect of activating and repressing transcription factors at DNA binding sites. We recognize that at the current state, this gene-peak linkage prediction is a proof of concept, with several open questions to be investigated. For example, the extent by which multiDGD captures direct or secondary interactions remains to be determined,

and whether different types of association can be distinguished. Further investigation is needed to determine whether the magnitude of perturbation changes is meaningful, or whether it is the most sensible to model counts as fractions of the count depth, as it is commonly done, for this application. Nevertheless, our results emphasize the potential of generative models as tools to capture interactions between molecular layers.

Altogether, multiDGD provides a strong performance increase on data reconstruction and clustering, incorporates modelling of covariates and provides a unified framework for integration and analysis of genomic features. We see these features as a significant next step in the evolution of single-cell multi-omics modelling.

4 Materials and Methods

4.1 Data

This work makes use of single-cell multiome data sets from human bone marrow, human brain tissue and mouse gastrulation stages.

4.1.1 Acquisition

The human bone marrow multiome data set from [31] was downloaded from NCBI GEO [48] under accession GSE194122 on September 12 2022. For human brain data, we used the annotated data from [32] The annotated mouse gastrulation set used was taken from [33].

4.1.2 Preprocessing

Human bone marrow data was used directly without any preprocessing. It comprises counts for 13431 gene transcripts and 116490 chromatin accessibility peaks. The 69249 cells represent 22 different cell types and were sequenced at four different sites. The sites are here interpreted as different batches.

The raw human brain counts were collected into an AnnData object according to 10X HDF5 Feature Barcode Matrix Format. This data set contains 3534 cells with a total of 274892 features. We performed feature selection by excluding features that were not present (meaning counts of zero) in at least one percent of all cells. The result were 15172 transcripts and 95677 peaks. For cell type annotation, we chose the ATAC cell type annotation with 16 different types. From this data, we used no batch annotation.

For the mouse gastrulation data, we again performed feature selection based on the percentage of cells. The original number of features were 32285 for transcripts and 192251 for peaks. We excluded features that were only present in five percent of the cells and arrived at 11792 gene expression features and 69862 chromatin accessibility features. The total number of cells in this data is 56861 with 37 different cell types. This data contains a temporal component and thus makes the definition of batches more difficult. Nevertheless, we chose the gastrulation stage as the batch and expect this variable to only be

Z	representation
X	data
$\hat{X}$	predicted/ reconstructed data
mod	modality
θ	decoder parameters
ϕ	GMM parameters
S	cell-specific scaling factor
$i \in N$	single sample i among N total samples
$k \in K$	component k among K components
l	latent dimension
$c \in C$	class c in C covariate classes
μ	GMM mean
Σ	GMM covariance
w, π	component coefficient, component weight
α	Dirichlet alpha

Table 2 Summary of relevant symbols used to describe the DGD.

partially removed from the latent representation as cell type appearance is not independent of the stage.

4.1.3 Data splits

In order to adequately compare model performances across methods and random seeds, we created data splits for training, validation and testing. All three data sets were randomly split into a train set comprising 80 % of the samples and validation and held-out test sets with 10 % of the samples each.

4.2 The model

multiDGD is an extension of the Deep Generative Decoder (DGD) [18] for single-cell multiomics data. The core model consists of a decoder and a parameterized distribution over latent space. This is presented by a Gaussian Mixture Model (GMM) here. Since there is no encoder, inference of latent representations is achieved by learning representations as trainable parameters. This process is described in detail in [18]. multiDGD additionally offers the option of disentangled covariate representations. For this purpose, multiDGD learns not only a set of representations Z and distribution parameters ϕ, but also Z^{cov_v} and ϕ^{cov_v} for every vth covariate. The corresponding graphical model is depicted in Supplementary Fig. 6. The following sections will describe the model and associated processes in more detail.

4.2.1 Relevant notation

4.2.2 Probabilistic formulation

The training objective is given by the joint probability $p(X, Z, \theta, \phi)$ [18], which is maximized using Maximum a Posteriori estimation [18]. This can be decomposed into:

$$p(X, Z, \theta, \phi) = p(X \mid Z, \theta)\, p(Z \mid \phi) \tag{1}$$

$p(X \mid Z, \theta)$ in this model is presented as the Negative Binomial distribution's mass of the observed count x_i given the predicted mean count and a learned dispersion parameter r_j for each feature j

$$p(x_i \mid z_i, \theta, s_i) = \prod_{j=1}^{D} p(x_{ij} \mid z_i, \theta, s_i) \tag{2}$$

and

$$p(x_{ij} \mid z_i, \theta, s_i) = \mathcal{NB}(x_{ij} \mid s_i y_{ij}, r_j), \tag{3}$$

where $\mathcal{NB}(x \mid y, r)$ is the negative binomial distribution. These equations are valid for each modality (RNA and ATAC) separately, as we have a total count s per modality.

The joint probability $p(X, Z, \theta, \phi)$ that is maximized in the DGD [18] further contains the objective for the representation to follow the latent distribution, $p(Z \mid \phi)$. Since ϕ is a GMM, this results in the weighted multivariate Gaussian probability density

$$p(z_i \mid \phi) = \sum_{k=1}^{K} \pi_k \mathcal{N}_L(z_i \mid \mu_k, \Sigma_k) \tag{4}$$

with K as the number of GMM components and $\mathcal{N}_L(z_i \mid \mu, \Sigma)$ is a multivariate Gaussian distribution with dimension L (the latent dimension), mean vector μ and covariance matrix Σ.

For new data points the representation is found by maximizing $p(x_i \mid z_i, \theta, s)p(z_i \mid \phi)$ only with respect to z_i, as all other model parameters are fixed. More of this in section 4.2.8.

4.2.3 Architecture

Decoder

The decoder in multiDGD is of hierarchical nature and will here be described in two sections: the shared network θ^h from latent space Z to the hidden state H, and the modality-specific networks θ^{mod} from H to their respective data spaces X^{mod}. All layers in the networks consist of a linear layer followed by Rectified Linear Unit (ReLU) activation, except for the last layer in θ^{mod}. The widths and depths of the networks are defined by hyperparameters described in Section 4.2.5. The activation of the last layer in θ^{mod} depends on the type of count scaling applied. Per default, the predicted normalized count means y^{mod} are scaled with the count depth s^{mod}. The count depth presents the sum of all counts per modality. In this case, the predicted count means are achieved through softmax [49] activation of y^{mod}. The probabilistic modelling of the counts and the corresponding objective function are described in the following section.

Count modelling

In multiDGD, counts of both gene expression and chromatin accessibility are modeled with Negative Binomial distributions (see Equation 3). For probabilistic modelling of outputs, we include 'output modules' which entail additional learned parameters and loss functions matching the probability distribution used. For the Negative Binomial output module, the necessary additional parameters are the feature-specific dispersion factors. For each feature in a given modality, we learn a dispersion factor to describe the shape of this individual feature's distribution. The loss function in this module is given by the negative log probability mass function of the Negative binomial given an observed count. This provides us with the reconstruction loss of the given modality.

$$Loss_{recon_i}^{mod} = -\sum_{j=1}^{M} \log \mathcal{NB}(x_i^j \mid \hat{x}_i^j, r^j) \tag{5}$$

GMM (basal)

The Gaussian Mixture Model (GMM) presents the complex distribution over latent space in this model. It is a parameterized distribution which determines the shape of the latent space and is optimized in parallel to decoder and representation during training. The GMM consists of a set of K multivariate Gaussians with the same dimensionality as the corresponding representation. For the purpose of simplicity, we let multiDGD choose K based on the number of unique annotated cell types. This parameter is of course flexible and allows for tailored latent spaces depending on the desired clustering resolution. The objective for the representations is given as

$$Loss_{rep_i}^{basal} = -\log p(z_i \mid \phi) = -\log \sum_{k=1}^{K} \pi_k \mathcal{N}_l(z_i \mid \mu_k, \Sigma_k) \tag{6}$$

Trainable parameters include the means μ and covariances Σ of the components and the mixture coefficients w, which are transformed into mixture weights π through the softmax function. These parameters are in turn also learned with respective priors. The composition of the prior loss is given as follows.

$$\begin{aligned} Loss_{prior}^{basal} &= -\log p(\phi) = -\log p(\mu, w, -\log \Sigma) \\ &= -\log \left(p(\mu)\, p(w)\, p(-\log \Sigma)\right) \end{aligned} \tag{7}$$

$$
\begin{aligned}
p(\mu) &= \prod_k p_{Softball}(\mu_k \mid \text{scale}, \text{sharpness}) \\
p(w) &= \prod_k Dir(\mu_k \mid \alpha) \\
p(-\log \Sigma) &= \prod_k \prod_l \mathcal{N}(-\log \Sigma_{k,l} \mid -\log 0.2 \times \frac{scale}{K}, 1)
\end{aligned} \tag{8}
$$

Altogether, these losses form the objective for both the representations and the GMM and will be referenced as the latent loss

$$
Loss^{basal}_{latent} = \sum_i^N Loss^{basal}_{rep_i} + Loss^{basal}_{prior} \tag{9}
$$

GMM (covariate)

The difference between the GMM for the basal latent space and the covariate space is merely the training scheme. As mentioned above, training for covariate representation and GMM is supervised. This results in a change in the objective as only probability densities of components assigned to a sample's label are taken into account.

$$
Loss^{cov}_{rep_i} = -\log p(z_i \mid \phi, c_i) = -\log \mathcal{N}_l(z_i \mid \mu_{k=c_i}, \Sigma_{k=c_i}) \tag{10}
$$

This means that the conditional probability $p(z_i \mid \phi)$ is solely dependent on the component with index identical to the numerical label $c_i \in 0, ..., C$ with C as the number of unique covariate labels.

4.2.4 Representations

Representations are treated as trainable parameters. However, they formally do not belong to the model architecture since they represent the low-dimensional embedding of data.

Representation (basal)

For each sample x_i with $i \in N$, there exists one representation z_i. The basal representations Z^{basal} represent the main embedding of data X, which aims to model the desired biological attributes of the data in low-dimensional space. As this structuring is unknown, Z^{basal} is inferred in an unsupervised setting. As described in [18], the representations are updated once per epoch with the gradients derived from reconstruction and distribution losses. Default initialization is given in Section 4.2.5.

Representation (covariate)

Covariates represent experimental variables we wish not to influence Z^{basal}. In order to separate these influences, we model these attributes in distinct two-dimensional spaces Z^{cov}. Here, it is necessary to follow a supervised training approach for successful disentanglement. This process is described in the corresponding section for the covariate GMM.

4.2.5 Initialization and default parameters

The decoder contains two layers in the shared network and two in the modality-specific ones. All layers except the last one in modality-specific networks have 100 hidden units ($layer\_width$). The last layer in a modality-specific network receives $max(layer\_width, \sqrt{M^{mod}})$ hidden units. Depth and width hyperparameters can of course be altered and should be considered depending on the number of samples and features available. Weights and biases are initialized per default using PyTorch's Kaiming Uniform [50] method. In the Negative Binomial output module, dispersion parameters are initialized with a default value of 2.

Representations are generally initialized at origin, meaning they all start from zero vectors. One could also initialize from a pre-defined matrix, for example an l-dimensional Principal Component Analysis (PCA) or sampling from the prior. However, linear mappings are not always representative of the true underlying structure. In the default settings, the latent dimensionality l is set to 20, and covariate representations receive two dimensions.

The GMM is generally initialized with Softball prior scale 2 and hardness 5 and a Dirichlet α of 1. The prior over the covariance matrix Σ is defined by the number of mixture components as in [18] with $0.2 \times \frac{scale}{K}$. The GMM per default contains a single Gaussian. This setting is used if no observables are provided for the basal latent space. However, we do not recommend the single component as it will decrease the flexibility and complexity of the basal representation and will not provide the intrinsic clustering of the model. If an observable is given, the number of unique classes will be used as the number of components in the basal GMM ϕ. This observable is commonly the cell type label. For the covariate GMM ϕ^{cov}, the number of components is equal to the number of unique categories in the covariate.

4.2.6 Training

The general training algorithm remains as presented in [18], with an extension due to the covariate latent model and presence of multiple modalities.

Algorithm 1 Training

Initialize parameters for representations Z, decoder θ and GMM ϕ
for epoch in n_epochs:
  for x_i, i, s_i in training data:
    $z_i = \langle z_i^{basal}, z_i^{cov} \rangle$
    $\hat{x}_i = \text{model}(z_i)$
    Computation of loss
    Backpropagation
    Optimizer step for model and GMMs
  Optimizer step for Representations

The training data is iterated over in mini batches with a default batch size of 128. Each set of parameters receives their on Adam [51] optimizer with betas $(0.5, 0.7)$ and learning rates of $1e-4$ for the decoder and $1e-2$ for representations and GMMs. As a proxy for the prior over θ, a weight decay of $1e-4$ is applied. The default maximum number of epochs is set to 1000, with early stopping applied at the earliest in epoch 50, taking into account the last 10 epochs.

The loss is presented as

$$-\log p(x, z, \phi, \theta) = \sum_{mod} Loss_{recon}^{mod} + Loss_{rep}^{basal} + \sum_{cov} Loss_{rep}^{cov} + Loss_{prior} \quad (11)$$

Positive definite parameters such as the Negative Binomial dispersion factors and the GMM covariances are learned as their logarithmic counterparts for numerical stability and enforcing the positive constraint.

4.2.7 Validation

Validation is performed in parallel to training. Representations for the validation set are equally initialized at origin and optimized every epoch. In the validation loop, only the representation parameters of the validation set are updated, and covariate representations are inferred in an unsupervised manner.

4.2.8 Testing and prediction of new data

With testing and predicting, we refer to the inference stages after the model parameters θ and ϕ have been trained and are regarded as frozen. The inference of representations for unseen data points is depicted in Fig. 1B. Firstly, the best mode (i.e. GMM component) is found for each sample. The best mode is given by the maximization of $p(x \mid z, \theta, \phi) \prod_k p(z \mid \phi_k)$ with respect to k. This step is the memory-critical process as for each new data point X_m with $m \in M$, K losses have to be computed. In the case of present covariate models, this problem becomes combinatorial. In total, $M \times K \times \prod_{q=1}^{Q} C_q$ losses have to be computed, with C_q representing the number of covariate classes for covariate q.

After the best modes have been determined, the representations are optimized for a set number of steps, per default 10. This process is very fast and negligible compared to the total run time as long as the number of cells is in the thousands [18].

4.2.9 Integrating unseen covariates

Because the covariate models are probabilistic, the method for integrating unseen covariate classes presented here works just like predicting new data. The unobserved covariate label might not have received its own distribution, but the model is capable to find the best covariate representation given its prior knowledge. One could see this as the unseen class being determined as a linear combination of the observed ones.

There is also the option of inferring an additional GMM component alongside the new covariate label. However, this is closer to fine-tuning as in scArches [28] and thus not discussed here, although we do plan to include this functionality in the future.

4.2.10 Missing modality prediction

We again start from a trained model with all internal parameters (decoder and GMM) fixed. For the new samples, representations are initialized as described above. The only change to the simple data inference is that only the loss of the observed modality is used to infer representations. After inference, the predictions for all features are generated, so we get a completed picture of the sample.

4.2.11 Internal clustering

The GMM is naturally equipped to cluster sample representations. Part of the objective calculation is to get a K-length vector for every representation containing the probability densities of said representation under each component $k \in K$. The argmax of this vector returns the index of the component with highest probability.

4.2.12 Gene-to-peak association with *in silico* perturbation

Figure 1C depicts the mechanism of the gene2peak feature of multiDGD. Intuitively, we associate features to each other across modalities by predicting the effect of silencing a given gene or set of genes $X^{j \in RNA}$ on the reconstructed peak accessibility profiles $\hat{X}^{i \in ATAC}$. To simulate the effect of silencing gene j on chromatin accessibility, we consider cells in which $X^{j \in RNA} > 0$. Then for each cell i we generate a pseudo-expression profile X_{KO} where $X_{KO}^{j \in RNA} = 0$ and compute the loss as usual. The loss is then backpropagated to get a gradient on Z^{basal}. As a result, we have the original representation Z^{basal} and the perturbed representation Z_{KO}^{basal}. From both, $\hat{X}$ and $\hat{X}_{KO}$ are predicted, and the perturbation changes $\Delta\hat{X}$ are computed as $\Delta\hat{X} = \hat{X} - \hat{X}_{KO}$.

4.3 Model search

4.3.1 Hyperparameter optimization

Architectures and training parameters can vary strongly in both VAE and DGD. For a comparison of tools rather than Machine Learning methods, we chose the default settings for both MultiVI and multiDGD. The default settings for multiDGD are derived from design knowledge gained in [18]. Additional parameters such as model depth of the hierarchical DGD were found experimentally. Final default parameters are described in section 4.2.5 above. MultiVI models were trained with one, two and three layers in the encoder and decoder each. The best architecture was chosen for each individual data set. multiDGD does not have an encoder, but shared and modality-specific

networks. We tested different depths of one to three layers for the shared and modality-specific feed-forward networks. A summary of our parameter search can be found in Supplementary Figure 7.

4.3.2 Training

For each of the data sets included in this work, we train three instances of each method with random seeds 0, 37 and 8790.

4.3.3 Model selection

For each data set, the best model from hyperparameter optimization was chosen based on the validation loss. This refers to the negative log density for multiDGD and the ELBO for MultiVI.

4.4 Performance evaluation

4.4.1 Reconstruction

Reconstruction performance metrics were chosen based on their compatibility for both MultiVI and multiDGD. Expression count reconstructions could be compared directly as both methods model the counts with negative binomial distributions. We thus report RMSE and MAE of the test predictions.

$$RMSE = \sqrt{\frac{1}{D \times N} \sum_{j=1}^{D} \sum_{i=1}^{N} (x_{ji} - \hat{x}_{ji})^2} \tag{12}$$

Chromatin accessibility is modelled differently in the two approaches. While multiDGD uses negative binomials, MultiVI models the peak counts as probabilities of being open. We thus binarized all predictions. As multiDGD does not use probabilities with a range from zero to one but models the actual counts, we calculated a prediction threshold of 0.2 for the binarization. After transforming the predictions, we compute the balanced accuracy as the average of sensitivity and specificity.

$$\begin{aligned} BA &= \frac{sensitivity \times specificity}{2} \\ \text{with } sensitivity &= \frac{TP}{TP \times FN} \\ \text{with } specificity &= \frac{TN}{TN \times FP} \end{aligned} \tag{13}$$

TP, FN, TN, FP present true positives, false negatives, true negatives and false positives, respectively. Positives are defined as 'open' regions (value = 1) and negatives as 'closed' (value = 0).

4.4.2 Clustering

multiDGD can naturally cluster samples based on the probabilities of the GMM components for a given representation. This is not possible with MultiVI's standard Gaussian prior, so it is common practice to perform Leiden [34] clustering on the latent space. We compare GMM and Leiden clustering through the Adjusted Rand Index (ARI) with respect to the cell type annotations. For a fair comparison, we adjust the Leiden algorithm such that it results in a similar number of clusters as the DGD, which is based on the number of unique cell types in the data. For bone marrow and brain data, the default scanpy Leiden parameters were used. For the gastrulation set, the resolution was set to 2. The ARI is the adjusted-for-chance version of the Rand index, which is related to clustering accuracy. This metric takes values between zero and one, with one representing perfect clustering according to the reference.

4.4.3 Batch effect removal

The batch effect is measured as the Average Silhouette Width (ASW). It is given as

$$\begin{aligned} ASW &= \frac{1}{N}\sum_{i}^{N}\frac{b(i)-a(i)}{max\{a(i),b(i)\}} \\ \text{with } a(i) &= \frac{1}{\mid C_I \mid -1}\sum_{j\in C_I,\, j\neq i} d(i,j) \\ \text{and } b(i) &= \min_{J\neq I}\frac{1}{\mid C_I \mid}\sum_{j\in C_I} d(i,j) \end{aligned} \tag{14}$$

$a(i)$ presents the mean distance between point i and all other points belonging to the same cluster. $d(i,j)$ is the distance between points i and j. $b(i)$ is the smallest mean distance of i to all points belonging to different clusters. This metric ranges between minus one and one. A value of one indicates a perfect clustering and a value of minus one indicates that samples would better fit into other clusters. For interpretability, we report $1-ASW$ as the batch effect removal metric, where larger values indicate better performance.

4.4.4 Data efficiency

In order to test model data efficiency, we created subsets of the largest data set in our study, with 1, 10, 25, 50 and 75 % of the training data. We trained both multiDGD and MultiVI instances on all these subsets with the hyperparameters determined for the full set and for the same three random seeds 0, 37 and 8790 as before. The performance of models on the subset is evaluated by the relative test losses, which we refer to as test loss ratios $\frac{1}{N}\sum_{i}^{N}\frac{Loss_i^{\text{trained on subset}}}{Loss_i^{\text{trained on full set}}}$ for every random seed.

4.4.5 Feature efficiency

For this experiment we chose the mouse gastrulation data as it had previously been used with stringent feature selection (section 4.1.2) and offered the most additional features of all three data sets. The data with feature selection (features that were present in at least five percent of cells) contained 11792 genes and 69862 peaks. The full data set with all measured features is comprised of 32285 genes and 192251 peaks. We trained instances for both multiDGD and MultiVI on the full data set with random seed 0.

In order to assess in what way training on all features affected the models' performances, we evaluated the reconstruction performances of RNA and ATAC data on the features previously selected for training (' 5%').

4.4.6 Modality prediction/imputation

The predictive performance is measured as the mean ratio of prediction error over reconstruction error $\frac{1}{N}\sum_i^N \frac{Loss_i^{\text{mod, pred}}}{Loss_i^{\text{mod, recon}}}$. Prediction refers to the data generation of the missing modality in the unpaired samples. Data generation of the original, paired samples is described as reconstruction.

4.5 Gene-peak association

All gene-peak association predictions were performed on the test set of the bone marrow data (6925 cells).

For prediction of perturbation effects around transcription start sites (Figure 5B), we selected a sample of highly variable genes in the RNA data for the test set, using the method implemented in scanpy. We then ran the *in silico* silencing for all the sampled genes (as described in 4.2.12) and measured the mean perturbation effect on chromatin ($\Delta\hat{X}$) across all perturbed cells on peaks located within 10kb of the TSS of the silenced gene (using gene annotations from Ensembl v108). Of note, the mean perturbation effect across cells in TF binding sites is partially dependent on the total number of cells expressing the silenced gene in the test set ($R^2 = 0.45$, p-value = 0.0029), suggesting that the estimates for perturbation effects might be more reliable with more support data.

For validation of gene-peak association predictions in distal enhancers (Figure 5C-D), we downloaded H3K27ac HiChIP data from primary T cells [39] from the Gene Expression Omnibus (GSE101498). Raw .hic files were converted to matrices of interaction signal between any two genomic bins of size 10kb using Juicebox tools [52], replicating the workflow and parameters described in [53]. We then computed the mean enhancer interaction signal (EIS) between 2 replicate samples for naive CD4+T cells and HCAMSC cells using as viewpoint the bins containing the promoter of the gene of interest. We calculated cell-type-specific gene-peak associations the absolute predicted change $|\sum_i^{|idx(ct)|} \Delta\hat{X}_{idx(ct)}|$ where ct stands for cell type and $idx(ct)$ is the subset of the indices for this cell type. We evaluate the ability to recover

enhancer-gene interactions from HiChIP data with ROC curve analysis, where we consider a genomic bin to be an enhancer region if the EIS is higher than the 75% quantile computed over the whole locus.

For the TF perturbation analysis (Figure 5E-F), we considered a list of transcription factors annotated as activators or repressors based on mining of GO terms by [40]. We identified peaks containing TF binding motifs using the JASPAR database (release 2022). We then restricted our analysis to TFs which had matches in less than 80% of all peaks and that were expressed in at least 250 cells in the test set. We computed $\Delta\hat{X}$ for silencing of each TF and computed the mean $\Delta\hat{X}$ across 10k peaks sampled amongst the peaks containing TF binding motifs, and across 10k peaks sampled amongst all peaks containing at least one TF binding motif. This strategy to select the null 'random' set was used to exclude peaks with extremely sparse counts at distal intergenic locations, which might represent an unfair comparison for this analysis.

4.6 Visualization

We applied UMAP[54] to dimensionality reductions used in visualizations.

4.7 Software and Hardware

All code is written in Python (version $\geq$ 3.9.12) and executed on a cluster with x86_64 architecture and NVIDIA TITAN RTX and NVIDIA TITAN X GPUs. The machine learning framework used was PyTorch [55] version 1.10. Training progress was monitored and logged using weights and biases (wandb) [56]. MultiVI was used as part of the scvi-tools [57] package with version 0.19.0. For Cobolt and scMM, we applied version v1.0.1 and release 1, respectively. The scanpy [58] package was used for parts of the analysis.

Funding

S.A.T. and E.D. acknowledge Wellcome Sanger core funding (WT206194). A.K. is supported by grants NNF20OC0062606, NNF20OC0063268, and NNF20OC0059939 from the Novo Nordisk Foundation.

Acknowledgments

This publication is part of the Human Cell Atlas (www.humancellatlas.org/publications/). We acknowledge all the great discussions at the Sanger Institute regarding data generation, processing and analysis and the wonderful support from the Center for Health Data Science.

Author contributions

A.K. contributed the theoretical foundation of the approach. A.K. contributed to the implementation of the approach. A.K. contributed to the writing of the manuscript.

E.D. contributed to the acquisition and preprocessing of data. E.D. contributed to the experimental designs in the work. E.D. contributed to the data analysis experiments. E.D. contributed to the writing of the manuscript. E.D. contributed to figures.

S.A.T. contributed to the experimental designs in the work. S.A.T. contributed to the writing of the manuscript.

V.S. contributed the theoretical foundation of the approach. V.S. contributed to the implementation of the approach. V.S. contributed to the acquisition and preprocessing of data. V.S. contributed to the experimental designs in the work. V.S. contributed to the writing of the manuscript. V.S. contributed the main experiments. V.S. contributed to figures.

Competing interests

S.A.T. is a Scientific Advisory Board member of ForeSite Labs, Qiagen and Element Biosciences, and is a co-founder and equity holder of TransitionBio and EnsoCell Therapeutics.

Code availability

The multiDGD code and package is made available here https://github.com/Center-for-Health-Data-Science/multiDGD.
The code to reproduce the presented results is available here https://github.com/Center-for-Health-Data-Science/multiDGD_paper.

Data availability

In this work we made use of three publicly available data sets. These are a human bone marrow set [31] with accession number GSE194122, a mouse gastrulation set [33] with accession number GSE205117 and a

human brain data set from [32] with accession number GSE162170. All processed data and trained model parameters have been deposited on Figshare (https://doi.org/10.6084/m9.figshare.23796198.v1).

References

[1] Baysoy A, Bai Z, Satija R, Fan R. The technological landscape and applications of single-cell multi-omics;p. 1–19. Publisher: Nature Publishing Group. https://doi.org/10.1038/s41580-023-00615-w.

[2] Argelaguet R, Cuomo ASE, Stegle O, Marioni JC. Computational principles and challenges in single-cell data integration;p. 1–14. Bandiera_abtest: a Cg_type: Nature Research Journals Primary_atype: Reviews Publisher: Nature Publishing Group Subject_term: Computational biology and bioinformatics;Systems biology Subject_term_id: computational-biology-and-bioinformatics;systems-biology. https://doi.org/10.1038/s41587-021-00895-7.

[3] Argelaguet R, Arnol D, Bredikhin D, Deloro Y, Velten B, Marioni JC, et al. MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data;21(1):111. https://doi.org/10.1186/s13059-020-02015-1.

[4] Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, et al. Comprehensive Integration of Single-Cell Data;177(7):1888–1902.e21. Publisher: Elsevier. https://doi.org/10.1016/j.cell.2019.05.031.

[5] Welch JD, Kozareva V, Ferreira A, Vanderburg C, Martin C, Macosko EZ. Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity;177(7):1873–1887.e17. https://doi.org/10.1016/j.cell.2019.05.006.

[6] Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data;184(13):3573–3587.e29. Publisher: Elsevier. https://doi.org/10.1016/j.cell.2021.04.048.

[7] Singh R, Hie BL, Narayan A, Berger B. Schema: metric learning enables interpretable synthesis of heterogeneous single-cell modalities;22(1):131. https://doi.org/10.1186/s13059-021-02313-2.

[8] Ashuach T, Gabitto MI, Koodli RV, Saldi GA, Jordan MI, Yosef N. MultiVI: deep generative model for the integration of multimodal data. Nature Methods. 2023 Jun;p. 1–10. Publisher: Nature Publishing Group. https://doi.org/10.1038/s41592-023-01909-9.

[9] Hao Y, Stuart T, Kowalski MH, Choudhary S, Hoffman P, Hartman A, et al. Dictionary learning for integrative, multimodal and scalable single-cell analysis;p. 1–12. Publisher: Nature Publishing Group. https://doi.org/10.1038/s41587-023-01767-y.

[10] Ghazanfar S, Guibentif C, Marioni JC. Stabilized mosaic single-cell data integration using unshared features;p. 1–9. Publisher: Nature Publishing Group. https://doi.org/10.1038/s41587-023-01766-z.

[11] Gong B, Zhou Y, Purdom E. Cobolt: integrative analysis of multimodal single-cell sequencing data;22(1):351. https://doi.org/10.1186/s13059-021-02556-z.

[12] Luecken MD, Burkhardt DB, Cannoodt R, Lance C, Agrawal A, Aliee H, et al. A sandbox for prediction and integration of DNA, RNA, and protein data in single cells;.

[13] Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder;10(1):390. Number: 1 Publisher: Nature Publishing Group. https://doi.org/10.1038/s41467-018-07931-2.

[14] Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics;15(12):1053–1058. Number: 12 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computational biology and bioinformatics;Computational models Subject_term_id: computational-biology-and-bioinformatics;computational-models. https://doi.org/10.1038/s41592-018-0229-2.

[15] Xu C, Lopez R, Mehlman E, Regier J, Jordan MI, Yosef N. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models;17(1):e9620. Publisher: John Wiley & Sons, Ltd. https://doi.org/10.15252/msb.20209620.

[16] Lotfollahi M, Wolf FA, Theis FJ. scGen predicts single-cell perturbation responses;16(8):715. https://doi.org/10.1038/s41592-019-0494-8.

[17] Grønbech CH, Vording MF, Timshel PN, Sønderby CK, Pers TH, Winther O. scVAE: variational auto-encoders for single-cell gene expression data. Bioinformatics. 2020 Aug;36(16):4415–4422. https://doi.org/10.1093/bioinformatics/btaa293.

[18] Schuster V, Krogh A.: The Deep Generative Decoder: MAP estimation of representations improves modeling of single-cell RNA data.

[19] Lotfollahi M, Litinetskaya A, Theis FJ.: Multigrate: single-cell multi-omic data integration. bioRxiv. Pages: 2022.03.16.484643 Section: New Results. Available from: https://www.biorxiv.org/content/10.1101/2022.03.16.484643v1.

[20] Minoura K, Abe K, Nam H, Nishikawa H, Shimamura T. A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data. Cell Reports Methods. 2021;1(5):100071. https://doi.org/https://doi.org/10.1016/j.crmeth.2021.100071.

[21] Cui H, Wang C, Maan H, Wang B.: scGPT: Towards Building a Foundation Model for Single-Cell Multi-omics Using Generative AI. bioRxiv. Pages: 2023.04.30.538439 Section: New Results. Available from: https://www.biorxiv.org/content/10.1101/2023.04.30.538439v1.

[22] Lopez R, Gayoso A, Yosef N. Enhancing scientific discoveries in molecular biology with deep generative models;16(9):e9198. Publisher: John Wiley & Sons, Ltd. https://doi.org/10.15252/msb.20199198.

[23] Kingma DP, Welling M.: Auto-Encoding Variational Bayes. arXiv. ArXiv:1312.6114 [cs, stat]. Available from: http://arxiv.org/abs/1312.6114.

[24] Luecken MD, Büttner M, Chaichoompu K, Danese A, Interlandi M, Mueller MF, et al. Benchmarking atlas-level data integration in single-cell genomics;p. 2020.05.22.111161. Publisher: Cold Spring Harbor Laboratory Section: New Results. https://doi.org/10.1101/2020.05.22.111161.

[25] Suo C, Dann E, Goh I, Jardine L, Kleshchevnikov V, Park JE, et al. Mapping the developing human immune system across organs;376(6597):eabo0510. Publisher: American Association for the Advancement of Science. https://doi.org/10.1126/science.abo0510.

[26] Eraslan G, Drokhlyansky E, Anand S, Fiskin E, Subramanian A, Slyper M, et al. Single-nucleus cross-tissue molecular reference maps toward understanding disease gene function;376(6594):eabl4290. Publisher: American Association for the Advancement of Science. https://doi.org/10.1126/science.abl4290.

[27] Sikkema L, Ramírez-Suástegui C, Strobl DC, Gillett TE, Zappia L, Madissoon E, et al. An integrated cell atlas of the lung in health and disease;29(6):1563–1577. Number: 6 Publisher: Nature Publishing Group. https://doi.org/10.1038/s41591-023-02327-2.

[28] Lotfollahi M, Naghipourfar M, Luecken MD, Khajavi M, Büttner M, Wagenstetter M, et al. Mapping single-cell data to reference atlases by transfer learning. Nature Biotechnology. 2022 Jan;40(1):121–130.

Number: 1 Publisher: Nature Publishing Group. https://doi.org/10.1038/s41587-021-01001-7.

[29] Lance C, Luecken MD, Burkhardt DB, Cannoodt R, Rautenstrauch P, Laddach A, et al.: Multimodal single cell data integration challenge: results and lessons learned [preprint]. Available from: http://biorxiv.org/lookup/doi/10.1101/2022.04.11.487796.

[30] Schuster V, Krogh A. A Manifold Learning Perspective on Representation Learning: Learning Decoder and Representations without an Encoder. Entropy. 2021;23(11). https://doi.org/10.3390/e23111403.

[31] Luecken M, Burkhardt D, Cannoodt R, Lance C, Agrawal A, Aliee H, et al. A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. In: Vanschoren J, Yeung S, editors. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. vol. 1; 2021. Available from: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/158f3069a435b314a80bdcb024f8e422-Paper-round2.pdf.

[32] Trevino AE, Müller F, Andersen J, Sundaram L, Kathiria A, Shcherbina A, et al. Chromatin and gene-regulatory dynamics of the developing human cerebral cortex at single-cell resolution. Cell. 2021 Sep;184(19):5053–5069.e23. https://doi.org/10.1016/j.cell.2021.07.039.

[33] Argelaguet R, Lohoff T, Li JG, Nakhuda A, Drage D, Krueger F, et al.: Decoding gene regulation in the mouse embryo using single-cell multi-omics. bioRxiv. Pages: 2022.06.15.496239 Section: New Results. Available from: https://www.biorxiv.org/content/10.1101/2022.06.15.496239v2.

[34] Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports. 2019 Mar;9(1):5233. Number: 1 Publisher: Nature Publishing Group. https://doi.org/10.1038/s41598-019-41695-z.

[35] Buettner F, Natarajan KN, Casale FP, Proserpio V, Scialdone A, Theis FJ, et al. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nature Biotechnology. 2015 Feb;33(2):155–160. Number: 2 Publisher: Nature Publishing Group. https://doi.org/10.1038/nbt.3102.

[36] Bardot ES, Hadjantonakis AK. Mouse gastrulation: Coordination of tissue patterning, specification and diversification of cell fate. Mechanisms of Development. 2020;163:103617. https://doi.org/https://doi.org/10.1016/j.mod.2020.103617.

[37] Cremer C, Li X, Duvenaud D. Inference Suboptimality in Variational Autoencoders. arXiv:180103558 [cs, stat]. 2018 May;ArXiv: 1801.03558.

[38] Heumos L, Schaar AC, Lance C, Litinetskaya A, Drost F, Zappia L, et al. Best practices for single-cell analysis across modalities;p. 1–23. Publisher: Nature Publishing Group. https://doi.org/10.1038/s41576-023-00586-w.

[39] Mumbach MR, Satpathy AT, Boyle EA, Dai C, Gowen BG, Cho SW, et al. Enhancer connectome in primary human cells identifies target genes of disease-associated DNA elements;49(11):1602–1612. Number: 11 Publisher: Nature Publishing Group. https://doi.org/10.1038/ng.3963.

[40] Domcke S, Hill AJ, Daza RM, Cao J, O'Day DR, Pliner HA, et al. A human cell atlas of fetal chromatin accessibility;370(6518). Publisher: American Association for the Advancement of Science Section: Research Article. https://doi.org/10.1126/science.aba7612.

[41] Ruvkun G, Lehrbach N. Regulation and Functions of the ER-Associated Nrf1 Transcription Factor;15(1):a041266. Company: Cold Spring Harbor Laboratory Press Distributor: Cold Spring Harbor Laboratory Press Institution: Cold Spring Harbor Laboratory Press Label: Cold Spring Harbor Laboratory Press Publisher: Cold Spring Harbor Lab. https://doi.org/10.1101/cshperspect.a041266.

[42] Corcoran SE, O'Neill LAJ. HIF1 and metabolic reprogramming in inflammation;126(10):3699–3707. Publisher: American Society for Clinical Investigation. https://doi.org/10.1172/JCI84431.

[43] Suico MA, Shuto T, Kai H. Roles and regulations of the ETS transcription factor ELF4/MEF;9(3):168–177. https://doi.org/10.1093/jmcb/mjw051.

[44] Fragale A, Gabriele L, Stellacci E, Borghi P, Perrotti E, Ilari R, et al. IFN regulatory factor-1 negatively regulates CD4+ CD25+ regulatory T cell differentiation by repressing Foxp3 expression;181(3):1673–1682. https://doi.org/10.4049/jimmunol.181.3.1673.

[45] Hwang SS, Kim LK, Lee GR, Flavell RA. Role of OCT-1 and partner proteins in T cell differentiation;1859(6):825–831. https://doi.org/10.1016/j.bbagrm.2016.04.006.

[46] Ficara F, Crisafulli L, Lin C, Iwasaki M, Smith KS, Zammataro L, et al. Pbx1 restrains myeloid maturation while preserving lymphoid potential in hematopoietic progenitors;126(14):3181–3191. https://doi.org/10.1242/jcs.125435.

[47] Martens LD, Fischer DS, Theis FJ, Gagneur J.: Modeling fragment counts improves single-cell ATAC-seq analysis. bioRxiv. Pages:

2022.05.04.490536 Section: New Results. Available from: https://www.biorxiv.org/content/10.1101/2022.05.04.490536v1.

[48] Edgar R, Domrachev M, Lash AE. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Research. 2002 01;30(1):207–210. https://doi.org/10.1093/nar/30.1.207. https://academic.oup.com/nar/article-pdf/30/1/207/9901036/300207.pdf.

[49] Boltzmann L, Hasenöhrl F. Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten; 2012. .

[50] He K, Zhang X, Ren S, Sun J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv. ArXiv:1502.01852 [cs] version: 1. Available from: http://arxiv.org/abs/1502.01852.

[51] Kingma DP, Ba J.: Adam: A Method for Stochastic Optimization. Cite arxiv:1412.6980. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. Available from: http://arxiv.org/abs/1412.6980.

[52] Durand NC, Shamim MS, Machol I, Rao SSP, Huntley MH, Lander ES, et al. Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments;3(1):95–98. Publisher: Elsevier. https://doi.org/10.1016/j.cels.2016.07.002.

[53] Granja JM, Klemm S, McGinnis LM, Kathiria AS, Mezger A, Corces MR, et al. Single-cell multiomic analysis identifies regulatory programs in mixed-phenotype acute leukemia;p. 1–8. https://doi.org/10.1038/s41587-019-0332-7.

[54] McInnes L, Healy J, Saul N, Grossberger L. UMAP: Uniform Manifold Approximation and Projection. The Journal of Open Source Software. 2018;3(29):861.

[55] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024–8035. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[56] Biewald L.: Experiment Tracking with Weights and Biases. Software available from wandb.com. Available from: https://www.wandb.com/.

[57] Gayoso A, Lopez R, Xing G, Boyeau P, Valiollah Pour Amiri V, Hong J, et al. A Python library for probabilistic analysis of single-cell omics data. Nature Biotechnology. 2022 Feb;40(2):163–166. Number: 2 Publisher: Nature Publishing Group. https://doi.org/10.1038/s41587-021-01206-w.

[58] Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biology. 2018 Feb;19(1):15. https://doi.org/10.1186/s13059-017-1382-0.