Deep Learning for Embedding and Integrating Multimodal Biomedical Data

Matthew Amodio
Yale University Graduate School of Arts and Sciences, matthew.amodio@yale.edu

Abstract

Deep Learning for Embedding and Integrating Multimodal Biomedical Data

Matthew Amodio

2022

Biomedical data is being generated in extremely high throughput and high dimension by technologies in areas ranging from single-cell genomics, proteomics, and transcriptomics (cytometry, single-cell RNA and ATAC sequencing) to neuroscience and cognition (fMRI and PET) to pharmaceuticals (drug perturbations and interactions). These new and emerging technologies and the datasets they create give an unprecedented view into the workings of their respective biological entities. However, there is a large gap between the information contained in these datasets and the insights that current machine learning methods can extract from them.

This is especially the case when multiple technologies can measure the same underlying biological entity or system. By separately analyzing the same system but from different views gathered by different data modalities, patterns are left unobserved if they only emerge from the multi-dimensional joint representation of all of the modalities together. Through an interdisciplinary approach that emphasizes active collaboration with data domain experts, my research has developed models for **data integration**, extracting important insights through the joint analysis of varied data sources.

In this thesis, I discuss models that address this task of multi-modal data integration, especially generative adversarial networks (GANs) and autoencoders (AEs). My research has been focused on using both of these models in a generative way for concrete problems in cutting-edge scientific applications rather than the exclusive focus on the generation of high-resolution natural images. The research in this thesis is united around ideas of building models that can extract new knowledge from scientific data inaccessible to currently existing methods.

Deep Learning for Embedding and Integrating Multimodal Biomedical Data

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Matthew Amodio

Dissertation Director: Smita Krishnaswamy

May 2022

Contents

List of Figures

List of Tables

Acknowledgements

I wish to express sincere gratitude for all of the support I have received during my PhD. While my name will be the only one on the diploma, my accomplishments represent a group effort, with support in various forms provided to me by a number of kind and helpful people.

Most significantly, I want to thank my advisor Dr. Smita Krishnaswamy, who has transformed me from a naive researcher with limited perspective into the mature researcher I am now. She has taught me much both in terms of specific technical knowledge and more qualitative advice for navigating the world of academia. She has also given me the latitude to explore my own thoughts and ideas, which has given me tremendous confidence as I progress further into my academic career.

I also want to thank Dr. Guy Wolf, who in his time at Yale and since, has been a strong mentor both personally and professionally. He has lent his technical expertise (not least of which being LaTex wizardry the likes of which I have never seen) and a grounded perspective on how to guide a particular research project from its initial stage over the finish line to publication.

The lab environment in the Krishnaswamy Lab has been a fantastic learning experience, both in terms of productivity and my enjoyment of spending time with the other members of the lab. To fellow PhD students Alex Tong, Dan Burkhardt, Scott Gigante, Manik Kuchroo, Jay Stanley, Egbert Castro, and Aarthi Venkat, senior researchers David van Dijk, Dennis Shung, Ofir Lindenbaum, Jessie Huang, and Feng Gao, and the many undergrads who have spent a brief sojourn in the lab (and all of those I am sure I am unintentionally omitting that I owe a debt of gratitude to): thank you all, you have been an integral part of this experience.

While Yale is the setting for my highest academic achievement, it is not the beginning of my academic journey. I wish to thank my teachers, from my childhood days in the public school system of Medina, Ohio, to my teachers at the Ohio State University, to those at the University of Wisconsin-Madison. I have learned so much at each level of my journey, and that is only with all of your help.

Last but in no way least, I want to thank my family. My parents, James and Bonnie, and my brothers, Steve, Dan, and Mike, as well as my grandparents, aunts and uncles, cousins, and all of my extended family have been my rocks through the ups and downs of my years workings towards this degree. Thank you for all of your help.

Chapter 1

Introduction

Generative modeling is one of the fastest growing fields of deep learning currently. Just in the time between when I started my graduate research and the time when I'm finishing it, they have gone from generating text that at least made sense at the sentence level or basic images that look realistic if you do not look too closely, to generating whole articles that would fool a reader into thinking it was written by a human and high-resolution images with high fidelity to a real image.

One use of these advances in generative models has been that of data alignment, or the combination of multiple different datasets into a single unified representation, by learning to map between them. Existing work along these lines has focused extensively on mapping between image domains, like pictures of human faces with brown hair and pictures of human faces with blond hair.

This dissertation instead targets applications in biological and medicinal fields, as I believe these to be more promising areas for both broader societal benefits and computational development. The broader societal benefits are likely evident, as increased biological understanding has the opportunity to improve life outcomes rather directly by guiding diagnoses and treatments. The promise in terms of computational development may be less easily seen. The problem lies in the fact that obsessive focus on images when designing models runs the risk of not seeing assumptions and biases built into their design simply because canonical image datasets satisfy the assumptions. By working on new biological datasets, we expose these models to a more varied set of tests and thus can come out with stronger frameworks as a result.

Data alignment in biological applications has grown in importance especially because of the invention of new technologies that can view a system from different perspectives, but that have trade-offs motivating alignments that can unite all of the different datasets generated from these technologies. In this dissertation I discuss various methods for alignment of these datasets, and I highlight how these are useful for discovering underlying aspects about them.

In Chapter 3, I introduce the first method for data alignment that I developed based on GANs, called the Manifold Alignment GAN (MAGAN) [4]. Broadly speaking, one theme recurring throughout this dissertation will be that of taking under-regularized alignment methods that are only used in specific cases where their assumptions are met and adding constraints that render them capable of producing meaningful alignments in a wider array of applications. In the case of MAGAN, it builds upon the framework of a cycle-consistent GAN by adding a correspondence loss to the total objective, which directly enforces a notion of alignment between all of the representations of a point in each distribution (from each domain). The previously existing cycle-consistent networks assume that any invertible mapping (one that does not lose information) produces a good alignment. With MAGAN, I demonstrate that this is evidently not true, and that the added correspondence loss is necessary in the biological applications considered to produce meaningful alignment.

In Chapter 4, I further generalize the notion of paired GANs by introducing the Transformation Vector Learning GAN (TraVeLGAN) [5]. In this chapter, I discuss the drawbacks to using a cycle-consistency loss at all in this context. The first of these is the dependency on a mean squared error distance metric to measure information retention (which in some application domains, such as images, bias information retention towards low frequency patterns that exist over large numbers of pixels but are less important to the content of the image, at the expense of high frequency patterns that cover fewer pixels but drive the content of the image). The second of which is that invertibility is not necessarily a good indicator of whether a pair of mappings between two domains is meaningful, and it may restrict the types of domains that can be considered unnecessarily. The TraVeLGAN sidesteps the need for cycle-consistency entirely by utilizing a third network that is introduced to learn a latent space in which the two representations of each point (in each domain) are aligned via data geometry.

In Chapter 5, I move to a new framework of relating different data domains for alignment with the use of a conditional GAN called the Feature Mapping GAN (FMGAN) [9]. In this application, heavily focused on use cases in drug discovery, one domain contains pharmaceutical drugs represented by metadata such as their chemical structures, and the second domain contains transcriptomic measurements of cell populations perturbed by these drugs. This situation is especially challenging because the domains are of drastically different cardinality: each drug is a single point in drug space, but each one corresponds to an entire distribution of many cells in the cell space. This situation requires a different solution than the cycle-consistent GANs of the previous chapters. The FMGAN approaches this with a conditional GAN where the conditions are the drugs and the data conditioned upon the drug is the cell transcriptomic distributions. With conditions as complex as images of drug structures, the FMGAN relies upon the addition of a condition embedding network that processes the conditions prior to them being given to the generator, as in the case of a traditional conditional GAN.

In Chapter 6, I change paradigms and propose an autoencoder based framework for aligning samples rather than the previously discussed GAN based frameworks for the task, beginning with the Sparse Autoencoder for Unsupervised Clustering, Imputation, and Embedding (SAUCIE) [7]. Autoencoders benefit from the stability and ease of training and the inherent automatic information retention that GANs lack. Thus, using manipulations of the latent layers of autoencoders to perform generation offers significant advantages over alternative methods. SAUCIE uses regularizations of internal layers motivated by information theoretic principles to perform various tasks important to the applications of single-cell analysis. One of these tasks is batch correction, which corrects technical differences in distributions of cells from separate independent measurements (such as different runs of the sequencer or from different subjects). SAUCIE utilizes a novel maximum mean discrepancy regularization in a latent layer to align distributions from each batch so that its output can be interpreted in one unified analysis.

In Chapter 7, I contribute another autoencoder-based framework for data alignment called neuron editing [8]. While the previously discussed autoencoder uses regularizations during training of the network to align samples, this method is based on the observation that this constrains alignment to only areas of the data space that are in the training sample, because autoencoders draw points towards the data manifold. The goal of alignment sometimes requires producing points that are different from those in the training set, though. In biological applications, there may be a rare cell type that exists in one subject in the healthy population but none in the diseased population. We would be forced to generate a prediction that this cell type disappears when a subject becomes diseased rather than predicting the disease does to it. Neuron editing avoids this problem by training an autoencoder on the full dataset and then performing a defined transformation over the learned feature space. Due to the transformation being performed offline (after training), it is capable of accurate out-of-sample extrapolation in a way that traditional autoencoder methods cannot perform.

In this dissertation, the chapters are based on research published in conferences and journals important to the community. The full publications are available for access in their respective original sources. The correspondence between chapter and publication is as follows (* † Denote equal contribution):

- Chapter 3: Amodio, M. and Krishnaswamy, S. *MAGAN: Aligning biological manifolds.* International Conference on Machine Learning, 2018.

- Chapter 4: Amodio, M. and Krishnaswamy, S. *TraVeLGAN: Image-to-image translation by transformation vector learning.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

- Chapter 5: Amodio, M, Shung, D, Burkhardt, D B, Wong, P, Simonov, M, Yamamoto, Y, van Dijk, D, Wilson, F P, Iwasaki, A, Krishnaswamy, S. *Generating hard-to-obtain information from easy-to-obtain information: applications in drug discovery and clinical inference.* Patterns (Cell Press), 2021.

- Chapter 6: Amodio, M*, van Dijk, D*, Srinivasan, K*, Chen, W, Mohsen, H, Moon, K, Campbell, A, Zhao, Y, Wang, X, Venkataswamy, M, Desai, A, Ravi, V, Kumar, P, Montgomery, R, Wolf†, G, Krishnaswamy, S†. *Exploring single-cell data with deep multitasking neural networks.* Nature Methods, 2019.

- Chapter 7: Amodio, M, van Dijk, D, Wolf, G, Krishnaswamy, S. *Learning general transformations of Data for out-of-sample extensions.* Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2020.

Chapter 2

Background

In this chapter, I discuss important fundamental topics that my research, discussed in detail later in the dissertation, will build upon.

2.1 Generative Adversarial Networks (GANs)

The GAN framework trains a generator network G along with a discriminator network D such that G learns to produce samples from the unknown data distribution implied by the training data set. Consider $X \sim P_X, x_i \in \mathbb{R}^d, i = 1...N_x$, a N_x-point sample from P_X, a d-dimensional distribution. The goal is for the generator G to learn to approximately sample from P_X by mapping input from a noise distribution $Z \sim P_Z$. G is guided toward this goal by gradient signals from the discriminator D that is trained with G in an adversarial framework, alternatively optimizing the objective function:

$$\begin{aligned}\min_G \max_D L_{GAN} =& \mathbb{E}_{x\sim P_x}[log(D(x))] \quad + \\ & \mathbb{E}_{z\sim P_z}[log(1 - D(G(z)))]\end{aligned}$$

where this objective is attempted to be minimized by G and maximized by D[52, 53, 137].

GANs have shown tremendous ability to generate hyper-realistic samples in even extremely complex data domains, such as images and text [23, 40]. They are able to do so because they benefit from several advantages. Formulating the generator's loss as a function of another network D allows the GAN to benefit from advances in neural network designs. As a result, the loss function for an image dataset can easily incorporate spatial invariance by using a convolutional neural network, or the loss function for a text dataset can be given sequential regularity by parameterizing the discriminator as a recurrent neural network [114, 174].

One cost of these advantages that have facilitated great generative performances, though, is a marked instability of training. Attempts to study the dynamics of GAN training have struggled to find a foundational footing in theory that convincingly explains observed behaviors in real data [113, 84]. This has motivated many variations on the original GAN paradigm that are intended to stabilize training in practice [173, 186]. While these have achieved varying levels of success, many of the most recent significant advances have come from stabilizing training techniques [23].

The GAN framework is built on foundational work in many related sub-topics, which we discuss here.

2.1.1 The Manifold Assumption

A commonly used notion in mathematical data analysis is that of the *manifold assumption*. The manifold assumption states that data points that exist natively in a high-dimensional, arbitrarily complex ambient space in fact lie on or close to a low-dimensional, locally Euclidean manifold. Since in practice points never lie exactly on such a manifold, it is necessary to allow the use of the lax condition of merely being near the manifold (and the magnitude of deviation can actually be key for analysis techniques such as anomaly detection).

This assumption is important in data analysis in general because low-dimensional, Euclidean spaces are easier to work with for several reasons. Often, analysis techniques rely on the ability to measure distances accurately. In the ambient space of these data points, often an accurate distance metric is unavailable because the coordinates of the space are encoding complex, non-monotonic latent variables. Traditional distance metrics like mean squared error or cross entropy are meaningless when dealing with text data or image data, especially with reference to important transformations like translations or shifts that we are interested in studying. An accurate distance metric in the ambient space of transformations like these is analytically intractable. Another property that makes locally Euclidean manifolds attractive for data analysis methods is the ability to move around in the space meaningfully by taking small steps (for example, following the gradient of some target function). That is, if we take one point in the ambient space x_i and its location in the manifold space m_i, where

$$x_i \sim X \in \mathbb{R}^d$$
$$m_i \sim M \in \mathbb{R}^n$$

such that $n \leq d$, then $m_i + \epsilon$ also is on or near the support of X for small ϵ. This is essential as exploring the space by moving around in small steps without straying from the data distribution is impossible for any non-trivial ambient data space under analyis.

The manifold assumption plays a crucial role in the study of GANs. As mentioned previously, GANs train a generator network G that maps from a noise sample $z \sim Z$ to a point that cannot be distinguished from a point $x \sim X$ that is drawn from the data distribution. The generator is guided throughout training by minimizing its objective function informed by the direction of the gradient with respect to z. This is possible because as we have discussed, we can take small steps around the manifold and stay on or near the manifold and as training proceeds, the sample manifold Z approaches the true data manifold M. Training a GAN on a reduced, low-dimensional space leverages the manifold assumption in a way other generative models like diffusion models [138, 67, 86] or normalizing flows [105, 70, 85] lack.

2.1.2 Conditional GANs

One early augmentation of the original GAN framework was that of the conditional GAN (cGAN). The cGAN modifies the original GAN by making the generator network mapping from the noise sample z conditioned upon a condition c, aka $G(z|c)$, with an analogous conditioning of the discriminator network $D(x|c)$. Coupled with the sample from the noise distribution, the generator then gives a full distribution of samples for each condition.

Applications of conditional GANs include conditioning images on natural language text, generating as a function of an auxiliary variable like time, image colorization and super-resolution, and data augmentation [42, 20, 128, 63, 125]. Conditional GANs benefit from the sharing of information, weights, and gradients across the classes by virtue of using the same network for different classes, while adding flexibility through the input of different conditions and gaining controllability, i.e. a desired output can be generated with greater ease by inputting a particular condition as compared to generating a desired output with a traditional GAN.

2.1.3 Cycle-consistent GANs

A later development in the GAN paradigm is related to the conditional GAN in general construct. The formal setting considers a pair of domains for data, X and Y. Each domain is sampled from, yielding a finite dataset of $\{x_i\}_{i=1}^{N_x} \in X$ and $\{y_i\}_{i=1}^{N_y} \in Y$. Crucially, these points are unpaired, in that while there is a correspondence between the domains at a distribution level, we do not have pairs of points (x_i, y_i) where $x_i \in X$ corresponds to the same underlying latent identity as $y_i \in Y$. The goal in this task, termed unsupervised domain mapping, is to learn functions $G_{XY} : X \to Y$ and $G_{YX} : Y \to X$ that map between the two domains. These generators are guided by discriminators D_X and D_Y for domains X and Y respectively, which try to distinguish between real samples from its domain and the generated samples from the generator in that direction. Note that the generators receive a point from one domain as input, while being trained to produce a point in the other domain as output. In this way, these generators resemble the generators in the conditional GAN framework discussed previously.

While the naive solution would be to just train these as completely distinct and separate GANs, drastic improvements in mapping quality, both in terms of the realism of the generated samples and the quality of matching of the generated sample and the real sample at a distribution level, were available through a slightly more complicated framework. This framework leveraged the information theoretic principle of information retention. The cycle-consistency loss did this, first introduced in the DiscoGAN model and then later popularized by the CycleGAN model [80, 189]. The cycle-consistency loss enforces the principle that the mapping in one direction must not lose any information contained in the input point by requiring that the other generator be able to invert the mapping and reconstruct the original point. That is, for example if the mean squared error loss is used to measure reconstruction, the generators are trained to minimize $||G_{YX}(G_{XY}(X)) - X||^2$ and $||G_{XY}(G_{YX}(Y)) - Y||^2$. In addition to producing meaningful mappings, this produces stable training, as only the space of invertible functions is explored. Since their advent, cycle-consistent GANs have dominated the space of unsupervised domain mapping [3, 78, 65, 176].

2.2 Autoencoders

Autoencoders are models originally used for representation learning, by pairing a dimensionality-reducing encoder E and a dimensionality-increasing decoder that when paired can reconstruct the original point, with an objective minimizing the loss over a data set X often using the mean squared error metric $||X - D(E(X))||^2$. As originally formulated, they are not productive as generative models because they can only be used to reproduce a point that is already available and given as input. However, they can also be used for generation in a variety of ways:

Variational autoencoders A modification of the autoencoder framework that places it in a probabilistic generative framework is the variational autoencoder (VAE) [82]. In the VAE, the latent layer that is output from the encoder $l = E(x)$ and fed to the decoder such that $x \approx D(l)$ is viewed probabilistically. The decoder D is a prior distribution over l giving the likelihood of the data $D(x|l)$. The encoder E meanwhile is a posterior distribution $E(l|x)$. The loss function is

$$L = -\mathbb{E}[log D(x|l)] + KL(E(l|x)||p(z))$$

where KL is the Kullback-Leibler divergence and $p(z)$ is a tractable, easy-to-sample-from distribution usually chosen to be an isotropic Gaussian. The two components of the loss are: maximizing the likelihood of the data under the decoder conditioned upon the latent variable plus the divergence between the output of the encoder conditioned upon the data as input and the chosen form of random distribution to represent the latent variables. This loss encourages the encoder to output points that approximate the latent variable distribution when fed a training data point, and the decoder to output a training data point when fed a sample from the latent variable distribution. A key to training this function, which relies on being able to backpropagate gradients from the last layer of the decoder all the way back to the first layer of the encoder, is the so-called “re-parameterization trick”. This approximates a sample from the latent variable distribution (which must be isotropic Gaussian for this trick to work) by taking samples from $l \sim N(0, 1)$ and transforming them with the μ_l, σ_l that is output by the encoder by simply taking $l' = \sigma_l \cdot (l + \mu_l)$. This facilitates backpropagation through μ_l, σ_l while channeling the non-differentiability through the randomness of the sample l. VAEs have proven to be popular in both theoretical and practical studies due to their analytic tractability and relatively ease of understanding and interpretability [41, 24, 104, 136].

Adversarial autoencoders Adversarial autoencoders mix ideas from the autoencoder and GAN frameworks [109, 77]. They replace the Kullback-Leibler divergence enforcing that the output of the encoder resembles the chosen noise distribution with a discriminator (“adversary”) that is trained with the standard GAN loss discussed above. The adversarial autoencoder benefits from the stability of training of the standard autoencoder because the loss function tied to the data is the reconstruction mean squared error loss. The benefits over VAEs include the flexibility of choosing non-standard latent distributions because they do not need to rely on the analytic usage of the re-parameterization trick. However, they also suffer from some of the same disadvantages: due to the use of the reconstruction loss, they suffer from limited ability to extrapolate outside of the training set, like autoencoders.

2.3 Deep Learning

Some topics are important to virtually every type of deep learning method, agnostic of whether the particular framework is a GAN or autoencoder or any other type of model.

2.3.1 Optimizers

Virtually all deep learning models rely on a standard off-the-shelf optimizer that is model agnostic to train the weights of all models involved. Perhaps surprisingly, one optimizer has been demonstrated to be the most effective in cases ranging across drastically different types of layers, architectures, and loss functions: the adam optimizer [185, 157]. This optimizer builds off of the standard stochastic gradient descent by adding a momentum-based adaptive per-parameter learning rate update to get the new updated weight w_{new} from the old weight w_{old} as follows:

$$w_{new} = w_{old} - \eta \cdot \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon}$$

where after initializing $m_0 = v_0 = 0$, then for training step t and gradient g:

$$\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t}
\end{aligned}$$

β_1, β_2 are hyperparameters usually chosen to be $0.9, 0.999$, respectively, η is the learning rate, and ϵ is a small normalizing constant. As the value of t increases, $\hat{m}$ and $\hat{v}$ become better approximations of the first and second moment of g. Scaling the universal learning rate η, which is only a single scalar chosen to be used in every weight in the network, with the per-parameter moments allows each weight to get its own learning rate based on its own gradient history. This facilitates hyperparameter choice by making the range of reasonable values for the learning rate much smaller and consistent across architectures and training regimes.

2.3.2 Invariance-inducing layers

Another topic used extensively in a wide variety of deep learning applications is designing types of layers that induce desired invariances with respect to particular transformations. Notably, those include **convolutional neural networks (CNNs)** and **recurrent neural networks (RNNs)**.

In CNNs, primarily used in the processing of images, induces translational invariance with respect to shifts between certain dimensions. Unlike tabular data, where columns are independent and our networks should treat information contained in each completely separately, in images we usually want the network to be insensitive to whether an object is centered at pixel (r_i, c_j) or three pixels over. CNNs induce this kind of translational invariance by applying a number of filters with learned weights, which is performed over a defined set of dimensions interpreting the pixels in their true underlying grid formation.

RNNs, on the other hand, induce not translational invariance but temporal invariance over discrete time steps. RNNs are frequently used in text data to be able to process long sequences of words or characters by conditioning its output on a state vector from the previous time step and then altering the state before passing it to the next step. The RNN function can be trained stably because it is learning a single function for all time steps, but gets different output at different times through the changing state vector that is passed through the RNN.

RNNs have become increasingly complicated to deal with the some of the problems they have in practice. For example, while the state vector being passed from time t to $t + 1$, theoretically information can be passed forward across an arbitrarily large number of time steps, as is necessary for a particular learning task. However, in practice, the chain rule dictates the gradient signal gets smaller and smaller because the recurrent weight multiplication is performed sequentially. Thus, RNNs suffer from this decay of the gradient signal, what is called the vanishing gradient problem. To rectify this, complex formulations like the long short-term memory network (LSTM) and gated recurrent unit (GRU) units have been built off of the foundation of the RNN unit. While they differ in their details, the LSTM and GRU both further decompose the state vector that is being passed forward to the next time step of the recurrent unit. By making the change to the state vector an additive change, and passed through so-called “gates” which are $[0, 1]$ scalars that can either let new information in or perform the identity function to keep the old information.

Part I

Multi-sample alignment with Generative Adversarial Networks

Chapter 3

Manifold Aligning GAN (MAGAN)

3.1 Introduction

It is increasingly common in many types of natural and physical systems (especially biological systems) to have different types of measurements performed on the same underlying system. In such settings, it is important to align the manifolds arising from each measurement in order to integrate such data and gain an improved picture of the system; we tackle this problem using generative adversarial networks (GANs). Recent attempts to use GANs to find correspondences between sets of samples do not explicitly perform proper alignment of manifolds. We present the new Manifold Aligning GAN (MAGAN) that aligns two manifolds such that related points in each measurement space are aligned. We demonstrate applications of MAGAN in single-cell biology in integrating two different measurement types together: cells from the same tissue are measured with both genomic (single-cell RNA-sequencing) and proteomic (mass cytometry) technologies. We show that MAGAN successfully aligns manifolds such that known correlations between measured markers are improved compared to other recently proposed models.

We commonly have samples from a pair of related domains and want to ask the natural question of how samples from one relate to samples from the other. Our motivational system for this is two types of measurements on cells sampled from the same population in a biological system. It is important for the discovery of new biology to integrate these datasets, which are often generated at great cost and expense. However, a fundamental challenge is

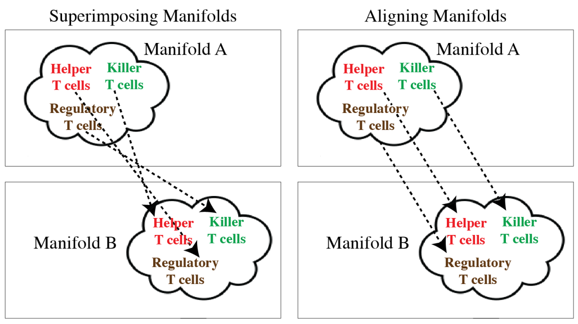


Figure 3.1: There are exponentially many mappings that superimpose the two manifolds, fooling a GAN's discriminator. By aligning the manifolds, we maintain pointwise correspondences.

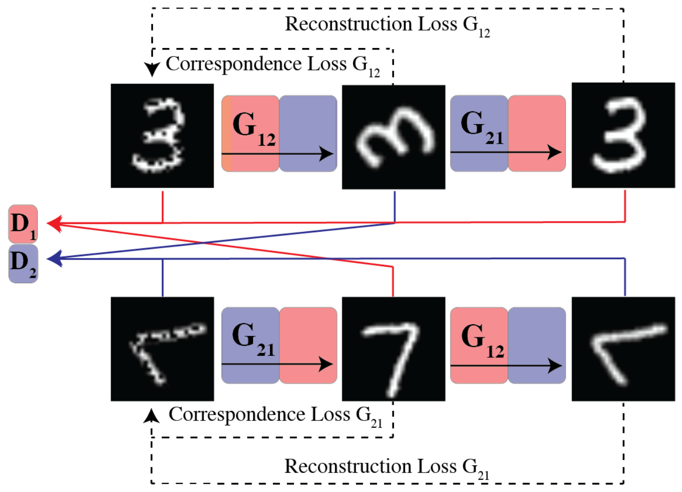

Reconstruction Loss G_{12}
Correspondence Loss G_{12}
G_{12}
G_{21}
D_1
D_2
Correspondence Loss G_{21}
Reconstruction Loss G_{21}

Figure 3.2: MAGAN's architecture with two generators, two discriminators, reconstruction loss, and correspondence loss. Domain 1 comprises upright images of 3's and 7's, Domain 2 comprises rotated images of 3's and 7's.

that there are exponentially many possible relationships that could exist between the two domains of measurement and the system must learn a logical way to map between them.

One way to approach this task to use dual GANs mapping between each domain. The first of these approaches required supervised paired examples from each domain, an impractical demand for many applications [73]. Recently, there have been attempts at performing the same task without the supervision of paired data [189, 181, 80, 94]. Like these previous models, MAGAN learns to map between distinct domains from unsupervised, unpaired data without pretraining. However, unlike them, MAGAN aligns rather than superimposes the manifolds of the two domains.

We draw the connection between unsupervised domain mapping with dual GANs and the alignment of the domain manifolds [44]. Much work has framed the generation problem of GANs as sampling points from this manifold [124, 188]. In the dual GAN framework, each domain's GAN learns a mapping that fools a discriminator by generating a point on the other domain's manifold when given a point on its manifold. There are exponentially many mappings that produce the same distribution of outputs from the same distribution of inputs but do so by mapping different individual input points to different individual output points. To the GAN's discriminator, these mappings are identical. But when we interpret the generator as finding correspondences between domains, we instead have preferences amongst them.

We motivate our preference for some mappings between domain manifolds by differentiating between *aligning* manifolds and *superimposing* manifolds3.1. A mapping that superimposes two manifolds would make the two indistinguishable by making their supports and densities identical. However, in our setting, we have produced each domain by measuring cells from the same underlying tissue twice. The two manifolds could be superimposed without aligning the two observations of each latent cell. To consider the two manifolds aligned rather than just superimposed, we require that a cell's representation in one manifold be aligned with that same cell's representation on the other manifold.

In this paper we propose the novel concept of using adversarial neural networks for alignment of manifolds arising from different biological experimental data measurement types. Single-cell biological experiments create many situations where manifold alignment problems are of interest. New technologies allow for measurements to be made at the granularity of each cell, rather than older technologies which could only acquire aggregate summary statistics for whole populations of cells. While these instruments allow us to discover biological phenomena that were not apparent before, it is a challenge to integrate and analyze this information in a unified fashion for biological discovery. Further, even for the same technology, experiments run on different days or in different batches can show variations even on the same populations, possibly due to calibration differences. In such cases even replicate experiments need alignment before comparison. Two such technologies that we examine are single-cell RNA sequencing which measures cells in thousands of gene (mRNA) dimensions and mass cytometry which measures protein abundances in several dozen dimensions [19, 83].

In all of these examples we have two data manifolds with a latent physical cell being measured analogously in each manifold. In some applications it might be adequate to simply superimpose these manifolds in any way. In many applications though, including the ones demonstrated here, we would like to be able to align them such that the two representations of each latent cell are aligned. MAGAN improves upon neural models for manifold alignment by finding the mapping between the manifolds (*correspondence*) that models these latent points by penalizing differences in each point's representation in the two manifolds.

We summarize the contributions of this paper as follows:

1. The introduction of a novel GAN architecture that aligns rather than superimposes manifolds to find relationships between points in two distinct domains

2. The demonstration of novel applications made possible by the new architecture in the analysis of single-cell biological data

The rest of this paper is organized as follows. First, there is a detailed description of the MAGAN architecture. Next, there is a validation of its performance on artificial data and the standard MNIST dataset. Then, there are demonstrations on three real-world biological applications: mapping between two replicate cytometry domains, mapping between two different cytometry domains, and mapping between one cytometry domain and a single-cell RNA sequencing domain.

3.2 Model

3.2.1 Architecture

MAGAN (Figure 3.2) is composed of two GANs, each with a generator network G that takes as input X and outputs a target dataset X'. We refer to each generator as a *mapping* from the input domain to the output domain. Each generator attempts to make its output $G(X)$ indistinguishable by D from X'. Denote the two datasets X_1 and X_2. Let the generator mapping from X_1 to X_2 be G_{12} and the generator mapping from X_2 to X_1 be G_{21}. The discriminator that tries to separate true points from mapped ones for the first domain is D_1 and the discriminator doing so for the second domain is D_2.

The loss for G_1 on minibatches x_1 and x_2 is:

$$x_{12} = \boldsymbol{G_{12}}(x_1)$$
$$x_{121} = \boldsymbol{G_{21}}(x_{12})$$
$$L_r = L_{reconstruction} = L(x_1, x_{121})$$
$$L_d = L_{discriminator} = -\mathbb{E}_{x_1 \sim P_{X_1}}\left[log\boldsymbol{D_2}(x_{12})\right]$$
$$L_c = L_{correspondence} = L(x_1, x_{12})$$
$$\boldsymbol{L_{G_1}} = L_r + L_d + L_c$$

where L is any loss function, here mean-squared error (MSE).

Similarly, the loss for G_2 is:

$$x_{21} = \boldsymbol{G_{21}}(x_2)$$
$$x_{212} = \boldsymbol{G_{12}}(x_{21})$$
$$L_r = L(x_2, x_{212})$$
$$L_d = -\mathbb{E}_{x_2 \sim P_{X_2}}\left[log\boldsymbol{D_1}(x_{21})\right]$$
$$L_c = L(x_2, x_{21})$$
$$\boldsymbol{L_{G_2}} = L_r + L_d + L_c$$

The losses for D_1 and D_2 are:

$$\begin{aligned}\boldsymbol{L_{D_1}} &= -\mathbb{E}_{x_1 \sim P_{X_1}}\left[log\boldsymbol{D_1}(x_1) + log\boldsymbol{D_1}(x_{121})\right] \\ &\quad -\mathbb{E}_{x_2 \sim P_{X_2}}\left[log(1 - \boldsymbol{D_1}(x_{21}))\right] \\ \boldsymbol{L_{D_2}} &= -\mathbb{E}_{x_2 \sim P_{X_2}}\left[log\boldsymbol{D_2}(x_2) + log\boldsymbol{D_2}(x_{212})\right] \\ &\quad -\mathbb{E}_{x_1 \sim P_{X_1}}\left[log(1 - \boldsymbol{D_2}(x_{12}))\right]\end{aligned}$$

3.2.2 Correspondence Loss

Previous models included only two restrictions: (1) that the two generators be able to reconstruct a point after it moves to the other domain and back, and (2) that the discriminators not be able to distinguish batches of true and mapped points. To do this, the generators could learn arbitrarily complex mappings as long as they superimpose the two manifolds.

To instead enforce the manifolds be fully aligned, MAGAN includes a correspondence loss between a point in its original domain and that point's representation after being mapped to the other domain. This correspondence loss needs to be chosen appropriately for the manifolds in any particular problem. We propose two such formulations: one unsupervised and one supervised.

Unsupervised Correspondence

In the biological domains considered here, we measure the same physical system in two different experiments where a subset of the dimensions in each experiment are shared. For example, in Section 3.3.4, we measure the amount of 35 proteins in a physical tissue in the first experiment and then the amount of 31 proteins in the same physical tissue in a second experiment. Sixteen of the proteins (CD4 for example) are measured in both experiments. Thus, the pre-mapping amount of CD4 in the first domain should equal the post-mapping amount of CD4 in the second domain. This leverages the information from the domains partially overlapping while generating a point in the full space (i.e. the generator maps a first domain point to the full 31-dimensional second domain by preserving the values of the 16 shared dimensions and then filling in the values of the 15 unique dimensions to make a plausible second domain point).

Formally, for each shared dimension pair (i, j), the correspondence loss is:

$$L_c = MSE(\boldsymbol{G_{12}}(x_1)_j, (x_1)_i) + MSE(\boldsymbol{G_{21}}(x_2)_i, (x_2)_j)$$

We note that there are many types of relationships between a dimension in the first domain and a dimension in the second domain that could define the unsupervised correspondence loss in other experimental settings. For example, rather than knowing that two dimensions

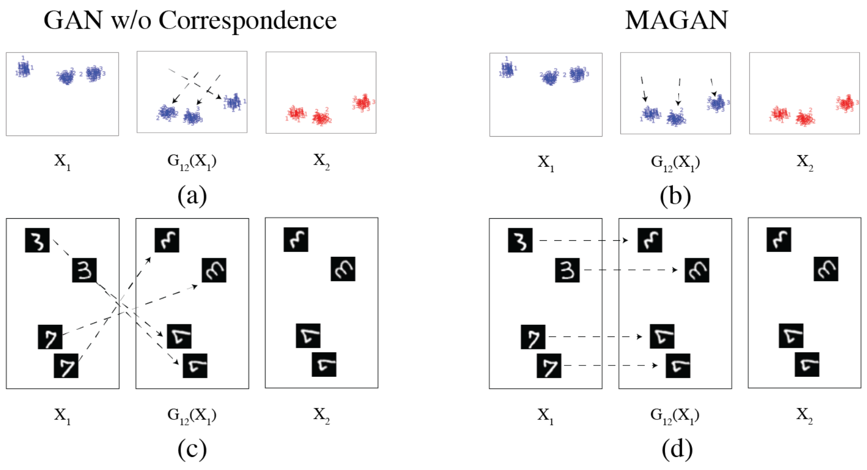


Figure 3.3: Both models superimpose the manifolds, meaning the first domain (X_1) is mapped to the second domain (X_2) such that the dataset of the first domain after mapping ($G_{12}(X_1)$) matches the second domain. Without the correspondence loss, though, this mapping is arbitrary and thus the relationships found vary. With the correspondence loss, the relationships found are coherent. This is confirmed with (a) a GAN without correspondence loss on artificial data (b) MAGAN on artificial data (c) a GAN without correspondence loss on MNIST and (d) MAGAN on MNIST.

of the experiment are identical, we might know that two markers are negatively correlated. A cell in one domain high in a T cell identifier like CD8 should not be mapped to a cell in the other domain high in a B cell identifier like CD19. The unsupervised correspondence loss can be formulated to enforce this by penalizing deviations from this known relationship.

Semi-supervised Correspondence

In the case where no known relationship between the domains is known, the correspondence loss could alternatively be formulated as a semi-supervised learning setting. Of course, if each point in X_1 already had a known correspondence with a point in X_2, no framework of dual GANs would be necessary to discover relationships. In some domains, though, it is easy to acquire a very small number of labeled pairs. We would like a model that learns from unsupervised data but can improve with any small number of labels that can be acquired. In those situations, we want to leverage both (1) the information that the unsupervised model has learned on all of the data and (2) incorporate the information the labels provide where they exist.

Here we choose the loss function to be nonzero only at the paired points in each domain.

Its value is then the sum of the losses on each labeled pair, where the loss for a particular labeled pair $(x_{1i}, x_{2j}), x_{1i} \in X_1, x_{2j} \in X_2$ is:

$$L_c = MSE(\boldsymbol{G_{12}}(x_{1i}), x_{2j}) + MSE(\boldsymbol{G_{21}}(x_{2j}), x_{1i})$$

3.2.3 Manifold Data Augmentation

MAGAN also utilizes a novel technique for data augmentation, leveraging the imperfect reconstructions each generator produces within its domain. It has been well established that autoencoders model and reconstruct from the data manifold [66, 166]. We note that the dual GANs within each domain function as an autoencoder, meaning their reconstruction x_i' of a sample x_i is another point near the underlying manifold, but importantly $x_i' \neq x_i$. By letting each discriminator see the reconstructions as true samples from the real domain, we both (1) augment the original data with new samples from the manifold and (2) prevent the discriminators from learning to separate real from generated examples by modeling the noise around the manifold, which differs between X_1 and $G_{21}(X_2)$ and between X_2 and $G_{12}(X_1)$. This is especially important in biological settings, where the number of measurements per cell dwarfs the number of cells measured and dropout in the measuring process produces sparsity.

3.3 Experiments

All experiments were performed with the MAGAN framework with discriminators of five layers each and generators of three layers each. Layer sizes depended on the dataset, while Leaky ReLU activations were used on all layers except the output layers of the discriminators (which were sigmoid) and the generators (which were linear). Dropout of 0.9 was applied during training and for images convolutional layers were used. Optimization was performed on 100,000 iterations of batches of size 256 by the ADAM optimizer with learning rate 0.001.

As with other GANs, the generators and discriminators are trained alternatively, so they each must get progressively better as their adversaries make their tasks harder and harder. One known difficulty in the adversarial training process is preventing a collapse of the generator into mapping all inputs to one point, chasing the minimum probability region of the discriminator as it moves. To combat this, MAGAN includes the approach outlined in [137]. This involves giving the discriminator access to minibatch information by having a subset of the network process a rotation of the original data matrix.

3.3.1 Artificial Data

We first test MAGAN on a generated example of points sampled from Gaussian distributions with varying means. Figure 3.3a shows the three subpopulations in the first domain X_1 in blue and the three in the second domain X_2 in red with an example mapping where, without the correspondence loss, each subpopulation in X_1 is mapped to a subpopulation in X_2, but not to the closest one. Even though the distribution of $G_{12}(X_1)$ matches the distribution of X_2, for an individual point $x_{1i} \in X_1$, $G_{12}(x_{1i})$ is not the member of X_2 that is most closely analogous to it. MAGAN finds a mapping that fools the discriminator, too: the one that least alters the original input (Figure 3.3b).

Without the correspondence loss, not only is a less-preferred manifold superimposition chosen, but the one chosen varies from run to run of the model. We compare the variability of the learned mappings across multiple runs of each model with 100 independent trials. In each trial we evaluate the relationships by calculating $G_{12}(x_{1i})$ for each $x_{1i} \in X_1$ and calculating its nearest neighbor x_{2j} in the real X_2. Then, this is repeated for the other domain. Figure 3.4a confirms that for the GAN without the correspondence loss, the learned manifold superimposition (and thus the correspondences) varies with repeated training the model. Figure 3.4b confirms MAGAN instead aligns the manifolds and finds the same correspondence every time.

3.3.2 MNIST

Next we test a subset of the MNIST handwritten digit data by taking only 3's and 7's as the first domain X_1, and a 120 degree rotation of each image as the second domain X_2. Without the correspondence loss (Figure 3.3c), each subpopulation in X_1 maps to one of the subpopulations in X_2, but the original 3's go to the rotated 7's and vice versa. There is no term in the objective function to create a preference for the mapping that sends original

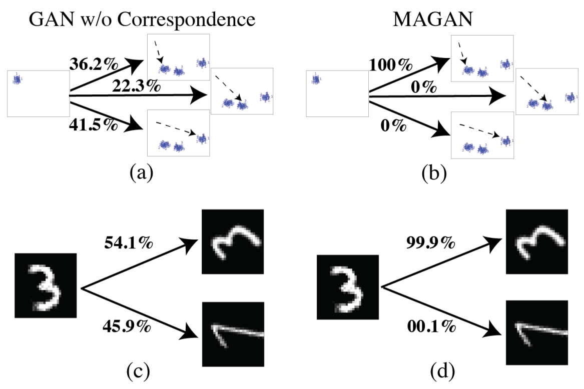


Figure 3.4: In simulations of 100 complete training runs of each model, without correspondence loss the resulting relationships learned varied randomly in both the (a) toy and (c) MNIST datasets. With correspondence loss, the most coherent relationship was found repeatedly for both (b) toy and (d) MNIST datasets.

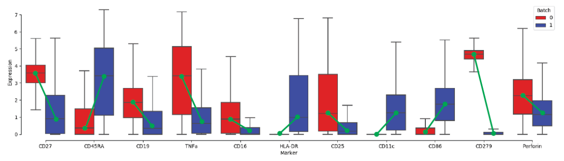


Figure 3.5: Selected markers illustrating large batch effects that separate the two data manifolds.

3's to rotated 3's. It would be difficult to define a distance measure that captures the notion of alignment with these manifolds, but it is a natural place where a small number of labeled pairs could be easily acquired. The semi-supervised correspondence loss with just a single labeled pair of points finds the desired manifold alignment and gets the correct correspondences for all of the other points that are unlabeled (Figure 3.3d).

Using the same simulation design as in the previous section, we can test the robustness of the models in finding these particular mappings. The GAN without the correspondence loss discovers either relationship with roughly even probability (Figure 3.4c). Remarkably, MAGAN is able to use the single labeled example to learn that (except for a few sloppily written 3's that in fact look more like 7's) the original 3's correspond to the rotated 3's and that the original 7's correspond to the rotated 7's every time (Figure 3.4d).

3.3.3 Correspondence: CyTOF Replicates

We now test MAGAN on real biological data from single-cell time-of-flight mass cytometry (CyTOF) measurements of protein abundance. Each protein, also referred to as a *marker*, is measured individually for each cell, allowing for more granular analysis than processes that only measure population totals for the cells in a given sample. Here the same sample was run twice in different batches (*replicates*), but due to machine calibration and other experimental details that are impossible to reproduce precisely each time, there are distortions between the batches. Thus, even though the same physical blood sample is being measured, the data manifold of each batch is different. The type of noise introduced by these distortions is not known *a priori*, need not fit any parametric assumption, and is likely to be highly nonlinear.

To analyze these two batches together, we need to know which cells in the first batch correspond to which cells in the second batch. To do this, we learn a mapping with MAGAN between the batches, each of which contains 75,000 cells with 34 individual markers measured. Figure 3.5 shows that the two batches indeed contain distinct differences in both the values of each marker and their distribution. For example, the mean value of HLA-DR in the second batch is higher than the *maximum* value in the first batch.

We demonstrate that MAGAN with its correspondence loss preserves crucial information that is lost with the mapping from the GAN without the correspondence loss. Often, analysis starts by identifying subpopulations of interest. For example, naive T-cells and central memory T-cells serve distinct functions and can be identified by looking at two isoforms of the CD45 marker, CD45RA and CD45RO [28]. In naive T-cells, CD45RA is present while CD45RO is not (CD45RA+CD45RO-), and in central memory T-cells CD45RA is not present while CD45RO is (CD45RA-CD45RO+). Figure 3.6a shows that very few cells had any CD45RA readings in the first batch, a typical case of instrument-induced dropout. Figure 3.6b shows proper readings for CD45RA in the second batch, where the two distinct subpopulations are clearly seen.

Both models learn a mapping for the first batch of cells x_1 such that $G_{12}(x_1)$ fools their discriminators by looking like the second batch of cells x_2. However, in the GAN without the correspondence loss (Figure 3.7a), naive T-cells in the first batch are mapped to central

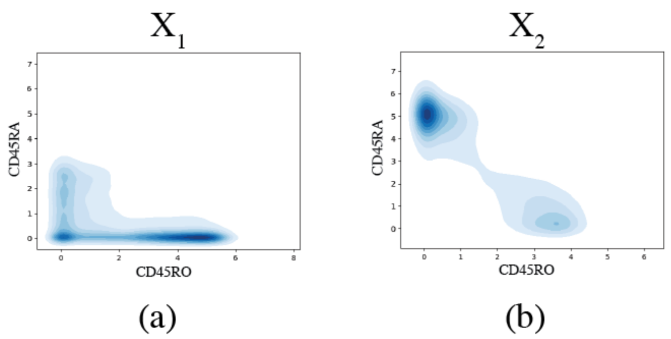


Figure 3.6: Two distinct populations of T-cells (CD45RA+CD45RO- and CD45RA-CD45RO+) with severe dropout in the CD45RA marker that causes a difference between that between the (a) first batch and (b) second batch.

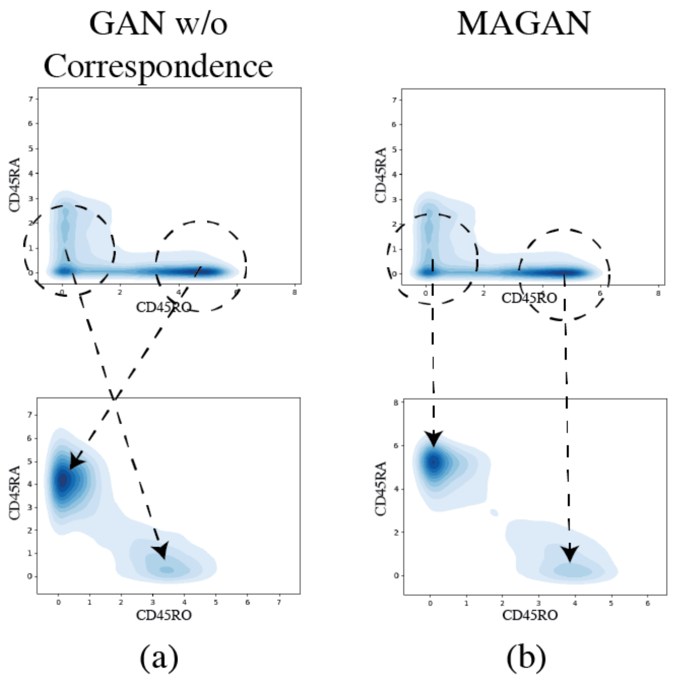


Figure 3.7: (a) Without correspondence loss, the GAN corrects the batch effect but subpopulations are reversed. (b) MAGAN still corrects the batch effect and subpopulations are preserved.

memory T-cells in the second batch and vice versa. If we went through the manual process of gating (selecting cells by manually looking at relative marker expression) central memory T-cells in the first batch and wanted to know whether their expression was similar in the second batch, we would be led to believe incorrectly that either there are none of these cells in the second batch or their expression profile is radically different.

MAGAN learns a different mapping (Figure 3.6b), the one in which subpopulation correspondences are preserved. Notably, the resulting mapped dataset $G_{12}(x_1)$ is not negatively affected by the correspondence loss. Instead, out of the two mappings that have similar results at the aggregate level, the one that maintains pointwise correspondences is learned. With the cell correspondences from other manifold superimpositions, the wrong biological conclusions could be made. This application necessitates MAGAN's manifold alignment.

3.3.4 Correspondence: Different CyTOF Panels

Next we demonstrate MAGAN's ability to align two manifolds in domains whose dimensionality only partly overlap. Despite the other advantages of CyTOF instruments, one disadvantage is that CyTOF experiments can only measure the expression of 30-40 markers per cell. Each experiment chooses which 30-40 markers to measure and refers to this set as the *panel*. Even though each panel has a limited capacity, different panels can be run on different samples from the same physical blood or tissue. MAGAN provides the opportunity to combine the results from these multiple panels and effectively increase the number of expression measurements acquired for each cell.

To test this, we use the datasets from two experiments published in [140] where each experiment had a different panel that was run on samples from the same population of cells. The first panel measured 35 markers, the second panel measured 31 markers, and 16 of those were measured in both. Without any advanced methods, all we would be able to do across experiments is compare population summary statistics — and lose all of the information at a single-cell resolution that motivated these experiments being done in the first place.

If we can identify points in each panel that measure the same cell, we can combine the measurements and have an augmented 50-dimensional dataset. To accomplish this, we take the first experiment's panel as one domain and the second experiment's panel as the other domain and use MAGAN to learn a mapping between the two. We then combine the original 35 dimensions of a cell in the first experiment x_{1i} with the 15 dimensions unique to the second experiment from that cell after mapping $G_{12}(x_{1i})$.

For combining the measurements from each experiment to be meaningful, the mapped point $G_{12}(x_{1i})$ must correspond accurately to the true point x_{2i}. This notion can be captured by taking the correspondence loss function to be the MSE across the 16 dimensions that are shared between the experiments. In other words, MAGAN should use the shared measurements to match cells between experiments, and then learn the required mapping for all of the measurements that are not shared. Without incorporating this correspondence measure into the model, x_{1i} need not be analogous to $G_{12}(x_{1i})$ in any way, and their information could not be combined.

We evaluate the accuracy of each model's learned correspondence by removing one of the markers measured in both experiments, CD3, from the first experiment. Then, we map points from the first experiment to the second experiment and evaluate how well the discovered CD3 values correspond with the true, held-out CD3 values for each cell from the first experiment.

Figure 3.8a shows that the GAN without the correspondence loss finds a manifold superimposition that does not preserve the values of CD3 for each cell accurately. Quantitatively, we can evaluate this with the correlation coefficient between the real, held-out CD3 values and the CD3 values predicted after mapping each point to the other domain. For the GAN without the correspondence loss (Figure 3.8a), the correlation is -.275, while for MAGAN (Figure 3.8b) it is .801. The negative correlation means that without the correspondence loss, the GAN will systematically map cells in one panel to different cells in the other panel.

We perform cross-validation by repeating this test with each of the 16 shared markers in turn for the GAN without correspondence loss (Figure 3.8c) and MAGAN (Figure 3.8d). While some of the markers have more shared information than others and are recovered more accurately, in all cases the correlation is better with the MAGAN.

If we had not measured one of these in the first experiment, we would have been able to use the learned value from the mapping in its place with remarkable accuracy. MAGAN can

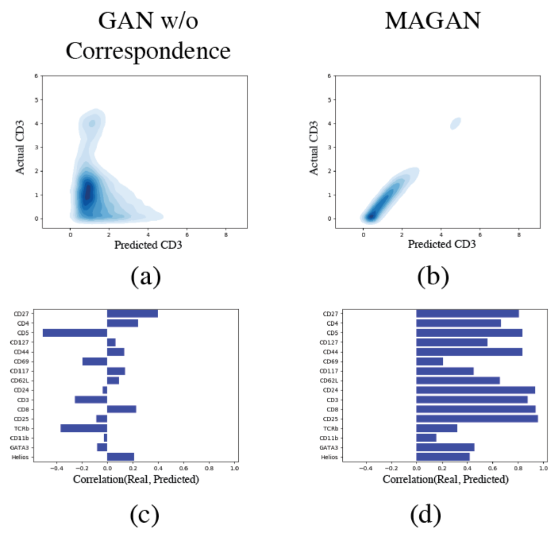


Figure 3.8: Using MAGAN's correspondence loss, measurements from each experiment can be combined. Their true values are known because they are measured in both experiments. Performing cross-validation by holding each out from the first experiment, we can measure the correlation between the predicted value and the real, correct value.

powerfully increase the impact of CyTOF experiments by expanding their limited capacity of markers that can be measured at any one time.

3.3.5 Correspondence: Cytometry and scRNA-seq

To demonstrate MAGAN aligning manifolds of domains with radically different dimensionality and underlying structure, we use it to find correspondences between flow cytometry (FACS-sorted) and scRNA-seq measurements made on the same set of cells. These two types of measurements have advantages and disadvantages, including the throughput, quality, and amount of information acquired from each. Being able to combine their information offers the possibility of getting the best from each and finding insights that might not otherwise be obtainable. In order to do this, though, it is crucial for pointwise correspondences to be accurate, or else features of a data point in the scRNA-seq domain will be ascribed to the incorrect point in the cytometry domain and the relationships will be meaningless.

To test MAGAN in this setting, we use a dataset consisting of 2830 measurements, where the dimensionality of each domain is 12 and 12496 for cytometry and scRNA-seq, respectively [165]. The scRNA-seq data was normalized with the inverse hyperbolic sine transform and preprocessed with MAGIC [163]. Here we know the true correspondences of

Table 3.1: With MAGAN's correspondence loss, the accuracy of the learned mapping is dramatically improved, as measured by the MSE between the known real point x and the predicted point $G(x)$ after mapping.

Paired Cytometry & scRNA-seq	GAN	MAGAN	Modified LLE
MSE(x_1, $G_{21}(x_2)$)	99.3	22.0	10.5
MSE(x_2, $G_{12}(x_1)$)	33.7	7.1	2.4

which points in the two domains are the same cell. In this setting we use the semi-supervised correspondence loss and show the impact of providing the pairing of just 10 cells, which can easily be acquired via inspection.

We evaluate the quality of the correspondences learned by calculating the *correspondence error*, or MSE between the true known value $x_{1i} \in X_1$ and the predicted correspondence $G_{21}(x_{2i})$. In this semi-supervised setting where we have some known labels, we compare MAGAN to both the GAN without correspondence loss and the method of [61] which uses a modification of locally linear embeddings for manifold alignment. We first find equal-sized manifolds for each domain separately. Then, for two observations we know to be corresponding, we constrain the point on each manifold to be identical. This aligns the two manifolds as anchored by the ground truth corresponding points. Then, in order to obtain correspondences in the original space rather than the manifold space, we interpolate between points in the original space based on a point's nearest neighbors in the latent manifold space.

We note that not only is this method only defined in the semi-supervised setting, it also requires finding the eigenvectors of a square matrix of size $n = 2830$, so cannot scale to the dataset sizes that neural network approaches like MAGAN can. Table 3.1 shows the correspondence error for the correspondences mapping to and from each domain. The LLE approach achieves the best error, showing that for small datasets with some ground truth labels, this method is preferable. We see that the correspondence loss is necessary for the GAN framework to perform manifold alignment, as MAGAN comes much closer to the performance of the baseline while the GAN is off by an order of magnitude.

3.4 Discussion

The use of GANs for manifold alignment is well motivated. By virtue of being parallelizable deep learning models trained with minibatch gradient descent, GANs can work with massive datasets. In contrast, other methods are graph-based and involve eigen decomposition of matrices that scale with the number of data points. Additionally, manifold alignment methods themselves are not inherently generative. They align points in the latent space and need to be augmented with original space interpolation (meaningful interpolation may not be achievable in all domains), while GANs explicitly generate points in the original space.

Furthermore, GANs learn a general non-linear mapping between manifolds that is not limited to any specific assumptions such as a linear latent dimension of correspondence [26] or that correlation distance is preserved across manifolds [58].

Outside of GANs, much previous work has been devoted to the field of manifold alignment. In [61, 62], semi-supervised approaches are used, but the learned alignments are only defined on the training points, unlike MAGAN which learns a universal mapping guided by its semi-supervised loss. In [167], Procrustes alignment is used to provide extension beyond the training points. This is still a two-step alignment process, though, which first finds a manifold for each domain and then separately aligns them. As such, information from the original space lost in the reduction to the estimated manifold is lost and not used in alignment. MAGAN jointly learns the manifolds and aligns them, allowing each to inform the other. In [168], they learn the manifolds and perform alignment jointly and do so without semi-supervised correspondences. They do this by matching the local geometry (thus requiring a meaningful distance metric in the original space), as opposed to MAGAN's unsupervised correspondence loss which can be an arbitrary function defined over the original spaces.

3.5 Conclusion

MAGAN discovers relationships between domains by aligning their manifolds rather than just superimposing them. Crucially, this can be used when one system is measured in two different ways and thus forms two different manifolds. In this case, the point in each manifold for one object in the underlying system are linked. This preserves information at a pointwise (rather than just population aggregate) level.

MAGAN facilitates integration of datasets from multiple biological modalities. As each type of experiment captures different information with different strengths and weaknesses, combining them makes possible discoveries that could not be found otherwise.

Chapter 4

Transformation Vector Learning GAN (TraVeLGAN)

4.1 Introduction

Interest in image-to-image translation has grown substantially in recent years with the success of unsupervised models based on the cycle-consistency assumption. The achievements of these models have been limited to a particular subset of domains where this assumption yields good results, namely homogeneous domains that are characterized by style or texture differences. We tackle the challenging problem of image-to-image translation where the domains are defined by high-level shapes and contexts, as well as including significant clutter and heterogeneity. For this purpose, we introduce a novel GAN based on preserving intra-domain vector transformations in a latent space learned by a siamese network. The traditional GAN system introduced a discriminator network to guide the generator into generating images in the target domain. To this two-network system we add a third: a siamese network that guides the generator so that each original image shares semantics with its generated version. With this new three-network system, we no longer need to constrain the generators with the ubiquitous cycle-consistency restraint. As a result, the generators can learn mappings between more complex domains that differ from each other by large differences - not just style or texture.

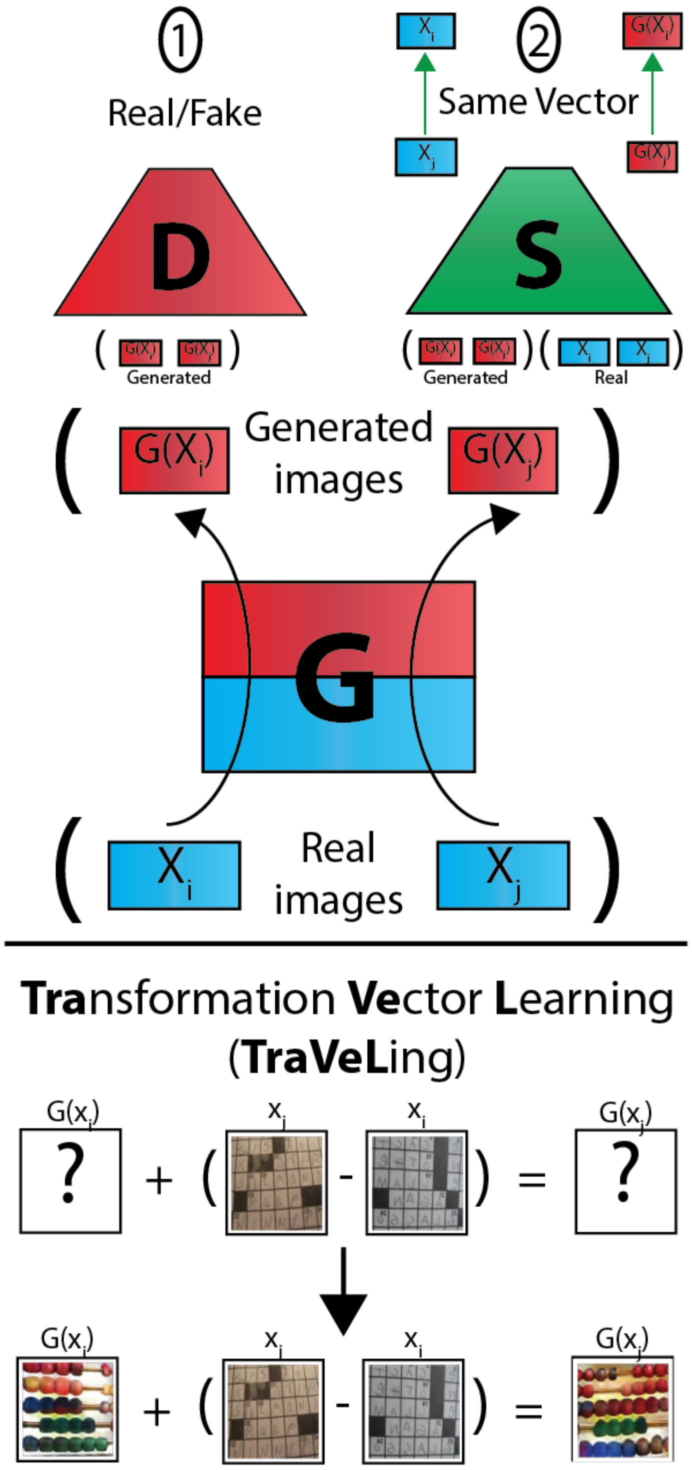


Figure 4.1: The TraVeLGAN architecture, which adds a siamese network S to the traditional generator G and discriminator D and trains to preserve vector arithmetic between points in the latent space of S.

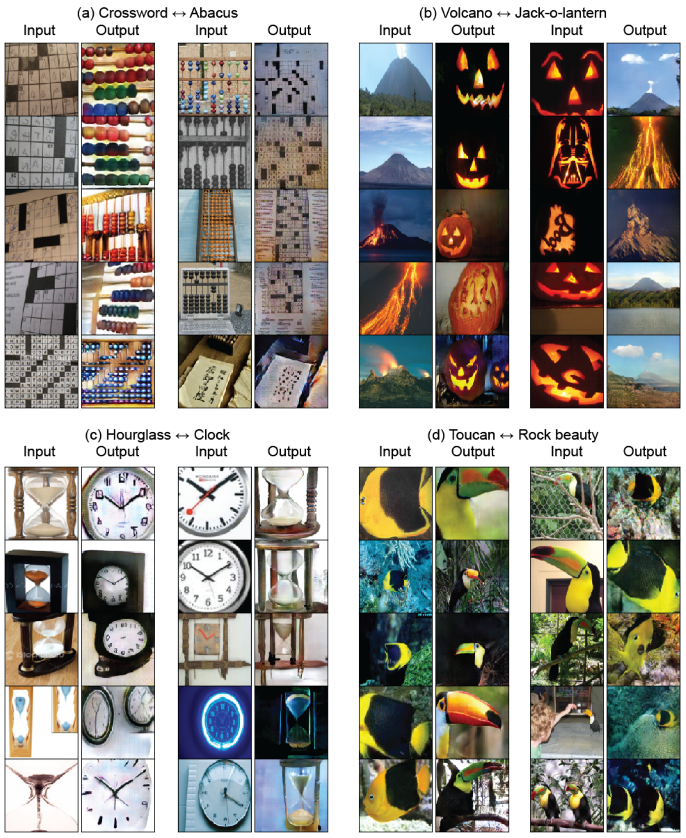


Figure 4.2: Examples of TraVeLGAN generated output on Imagenet domains that are too different and diverse for cycle-consistent GANs to map between. The TraVeLGAN successfully generates images that are both fully realistic in the output domain (shape of object, color, background) and have preserved semantics learned by the siamese network.

Learning to translate an image from one domain to another has been a much studied task in recent years [178, 72, 68, 184, 50]. The task is intuitively defined when we have paired examples of an image in each domain, but unfortunately these are not available in many interesting cases. Enthusiasm has grown as the field has moved towards unsupervised methods that match the distributions of the two domains with generative adversarial networks (GANs) [74, 45, 135, 151, 97]. However, there are infinitely many mappings between the two domains [96], and there is no guarantee that an individual image in one domain will share any characteristics with its representation in the other domain after mapping.

Other methods have addressed this non-identifiability problem by regularizing the family of generators in various ways, including employing cross-domain weight-coupling in some layers [97] and decoding from a shared embedding space [98]. By far the most common regularization, first introduced by the CycleGAN and the DiscoGAN, has been forcing the generators to be each other's inverse, known as the cycle-consistency property [69, 189, 80, 134, 102, 3, 35, 12, 181]. Recent findings have shown that being able to invert a mapping at the entire dataset level does not necessarily lead to the generation of related real-generated image pairs [94, 4, 45].

Not only do these dataset-level regularizations on the generator not provide individual image-level matching, but also by restricting the generator, they prevent us from learning mappings that may be necessary for some domains. Previous work continues to pile up regularization after regularization, adding restrictions on top of the generators needing to be inverses of each other. These include forcing the generator to be close to the identity function [189], matching population statistics of discriminator activations [80], weight sharing [97], penalizing distances in the latent space [134], perceptual loss on a previously trained model [98], or more commonly, multiple of these.

Instead of searching for yet another regularization on the generator itself, we introduce an entirely novel approach to the task of unsupervised domain mapping: the **Tra**nsformation **Ve**ctor **L**earning GAN (TraVeLGAN).

The TraVeLGAN uses a third network, a siamese network, in addition to the generator and discriminator to produce a latent space of the data to capture high-level semantics characterizing the domains. This space guides the generator during training, by forcing the generator to preserve vector arithmetic between points in this space. The vector that transforms one image to another in the original domain must be the same vector that transforms the generated version of that image into the generated version of the other image. Inspired by word2vec embeddings [51] in the natural language space, if we need to transform one original image into another original image by moving a foreground object from the top-left corner to the bottom-right corner, then the generator must generate two points in the target domain separated by the same transformation vector.

In word2vec, semantic vector transformations are a *property* of learning a latent space from known word contexts. In TraVeLGAN, we *train* to produce these vectors while learning the space.

Domain mapping consists of two aspects: (a) transfer the given image to the other domain and (b) make the translated image similar to the original image in some way. Previous work has achieved (a) with a separate adversarial discriminator network, but attempted (b) by just restricting the class of generator functions. We propose the natural extension to instead achieve (b) with a separate network, too.

The TraVeLGAN differs from previous work in several substantial ways.

1. It completely eliminates the need for training on cycle-consistency or coupling generator weights or otherwise restricting the generator architecture in any way.
2. It introduces a separate network whose *output* space is used to score similarity between original and generated images. Other work has used a shared latent *embedding* space, but differs in two essential ways: (a) their representations are forced to overlap (instead of preserving vector arithmetic) and (b) the decoder must be able to decode out of the embedding space in an autoencoder fashion [98, 134] ([98] shows this is in fact equivalent to the cycle consistency constraint.
3. It is entirely parameterized by neural networks: nowhere are Euclidean distances between images assumed to be meaningful by using mean-squared error.
4. It adds interpetability to the unsupervised domain transfer task through its latent space, which explains what aspects of any particular image were used to generate its paired image.

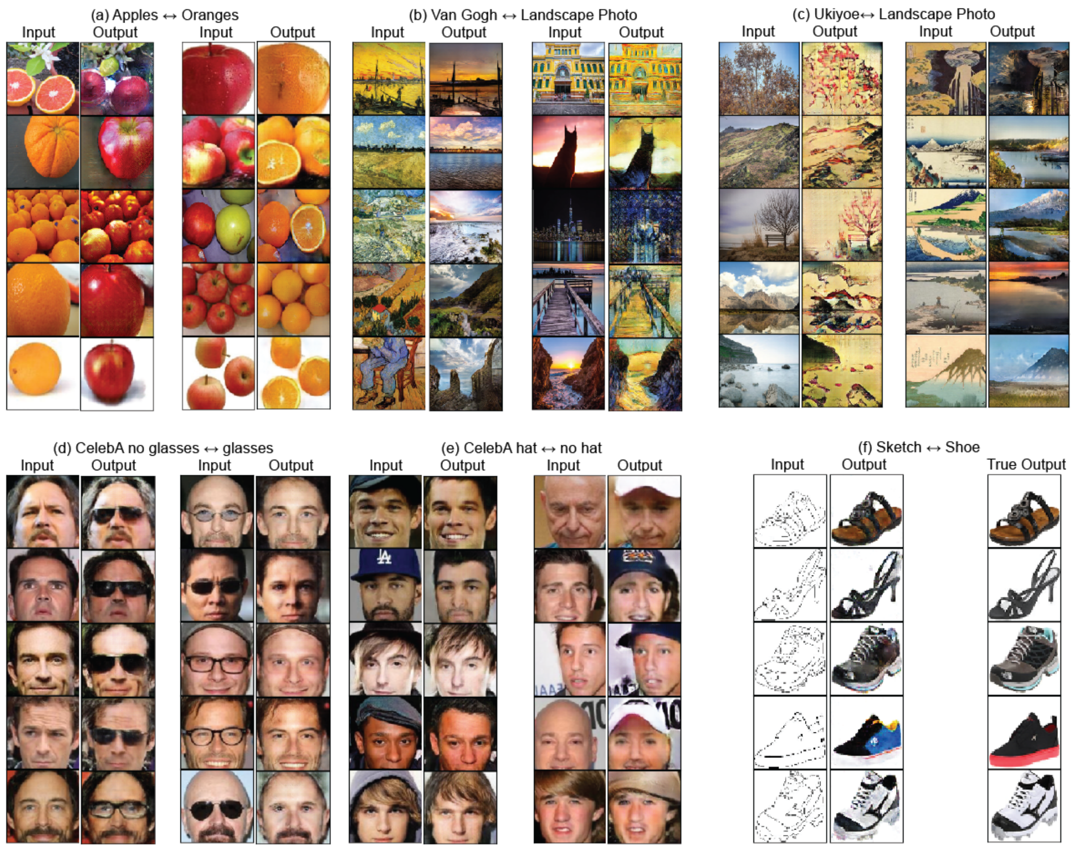


Figure 4.3: Examples of TraVeLGAN generated output on traditional datasets for unsupervised domain transfer with cycle-consistent GANs. Little change to the original image is necessary in these problems, and TraVeLGAN generates the expected, minimally changed image in the other domain.

As a consequence of these differences, the TraVeLGAN is better able to handle mappings between complex, heterogeneous domains that require significant and diverse shape changing.

By avoiding direct regularization of the generators, the TraVeLGAN also avoids problems that these regularizations cause. For example, cycle-consistency can unnecessarily prefer an easily invertible function to a possibly more coherent one that is slightly harder to invert (or preventing us from mapping to a domain if the inverse is hard to learn). Not only must each generator learn invertible mappings, but it further requires that the two invertible mappings be each other's inverses. Furthermore, cycle-consistency is enforced with a pixel-wise MSE between the original and reconstructed image: other work has identified the problems caused by using pixelwise MSE, such as the tendency to bias towards the mean images [22].

Our approach bears a resemblance to that of the DistanceGAN [18], which preserves pairwise distances between images after mapping. However, they calculate distance directly on the pixel space, while also not preserving any notion of directionality in the space between images. In this paper, we demonstrate the importance of not performing this arithmetic in the pixel space.

Many of these previous attempts have been developed specifically for the task of style transfer, explicitly assuming the domains are characterized by low-level pixel differences (color, resolution, lines) as opposed to high-level semantic differences (shapes and types of specific objects, composition) [22, 181, 50]. We demonstrate that these models do not perform as well at the latter case, while the TraVeLGAN does.

4.2 Model

We denote two data domains X and Y, consisting of finite (unpaired) training points $\{x_i\}_{i=1}^{N_x} \in X$ and $\{y_i\}_{i=1}^{N_y} \in Y$, respectively. We seek to learn two mappings, $G_{XY} : X \to Y$ and $G_{YX} : Y \to X$, that map between the domains. Moreover, we want the generators to do more than just mimic the domains at an aggregate level. We want there to be a meaningful and identifiable relationship between the two representations of each point. We claim that this task of unsupervised domain mapping consists of two components: **domain membership** and **individuality**. Without loss of generality, we define these terms with respect to G_{XY} here, with G_{YX} being the same everywhere but with opposite domains.

Domain membership The generator should output points in the target domain, i.e. $G_{XY}(X) \in Y$. To enforce this, we use the standard GAN framework of having a discriminator D_Y that tries to distinguish the generator's synthetic output from real samples in Y. This yields the typical adversarial loss term L_{adv}:

$$L_{adv} = E_X\left[D_Y(G_{XY}(X))\right]$$

Individuality In addition, our task has a further requirement than just two different points in X each looking like they belong to Y. Given $x_i, x_j \in X, i \neq j$, we want there to be some relationship between x_i and $G_{XY}(x_i)$ that justifies why $G_{XY}(x_i)$ is the representation in domain Y for x_i and not for x_j. Without this requirement, the generator could satisfy its objective by ignoring anything substantive about its input and producing arbitrary members of the other domain.

While other methods try to address this by regularizing G_{XY} (by forcing it to be close to the identity or to be inverted by G_{YX}), this limits the ability to map between domains that are too different. So instead of enforcing similarity between the point x_i and the point $G_{XY}(x_i)$ directly in this way, we do so implicitly by matching the relationship between the x_i's and the relationship between the corresponding $G_{XY}(x_i)$'s.

We introduce the notion of a *transformation vector* between two points. In previous natural language processing applications [51], there is a space where the vector that would transform the word *man* to the word *woman* is similar to the vector that would transform *king* to *queen*. In our applications, rather than changing the gender of the word, the transformation vector could change the background color, size, or shape of an image. The crucial idea, though, is that whatever transformation is necessary to turn one original image into another original image, an analogous transformation must separate the two generated versions of these images.

Formally, given $x_i, x_j \in X$, define the transformation vector between them $\nu(x_i, x_j) = x_j - x_i$. The generator must learn a mapping such that $\nu(x_i, x_j) = \nu(G_{XY}(x_i), G_{XY}(x_j))$. This is a more powerful property than even preserving distances between points, as it requires the space to be organized such that the directions of the vectors as well as the magnitudes be preserved. This property requires that the vector that takes x_i to x_j, be the same vector that takes $G_{XY}(x_i)$ to $G_{XY}(x_j)$.

As stated so far, this framework would only be able to define simple transformations, as it is looking directly at the input space. By analogy, the word-gender-changing vector transformation does not hold over the original one-hot encodings of the words, but instead holds in some reduced semantic latent space. So we instead redefine the transformation vector to be $\nu(x_i, x_j) = S(x_j) - S(x_i)$, where S is a function that gives a representation of each point in some latent space. Given an S that learns high-level semantic representations of each image, we can use our notion of preserving the transformation vectors to guide generation. We propose to learn such a space with an analogue to the adversarial discriminator D from the traditional GAN framework: a cooperative siamese network S.

The goal of S is to map images to some space where the relationship between original images is the same as the relationship between their generated versions in the target domain:

$$
\begin{aligned}
L_{TraVeL} &= \Sigma\Sigma_{i \neq j} Dist(\nu_{ij}, \nu'_{ij}) \\
\nu_{ij} &= S(x_i) - S(x_j) \\
\nu'_{ij} &= S(G_{XY}(x_i)) - S(G_{XY}(x_j))
\end{aligned}
$$

where $Dist$ is a distance metric, such as cosine similarity. Note this term involves the parameters of G, but G needs this space to learn its generative function in the first place. Thus, these two networks depend on each other to achieve their goals. However, unlike in the case of G and D, the goals of G and S are not opposed, but cooperative. They both want L_{TraVeL} to be minimized, but G will not learn a trivial function to satisfy this goal, because it also is trying to fool the discriminator. S could still learn a trivial function (such as always outputting zero), so to avoid this we add one further requirement and make its objective multi-task. It must satisfy the standard siamese margin-based contrastive objective [112, 122] L_{S_c}, that every point is at least δ away from every other point in the latent space:

$$
L_{S_c} = \Sigma\Sigma_{i \neq j} max(0, (\delta - ||\nu_{ij}||_2))
$$

This term incentivizes S to learn a latent space that identifies some differences between images, while L_{TraVeL} incentivizes S to organize it. Thus, the final objective terms of S and G are:

$$
\begin{aligned}
L_S &= L_{S_c} + L_{TraVeL} \\
L_G &= L_{adv} + L_{TraVeL}
\end{aligned}
$$

G and S are cooperative in the sense that each is trying to minimize L_{TraVeL}, but each has an additional goal specific to its task as well. We jointly train these networks such that together G learns to generate images that S can look at and map to some space where the relationships between original and generated images are preserved.

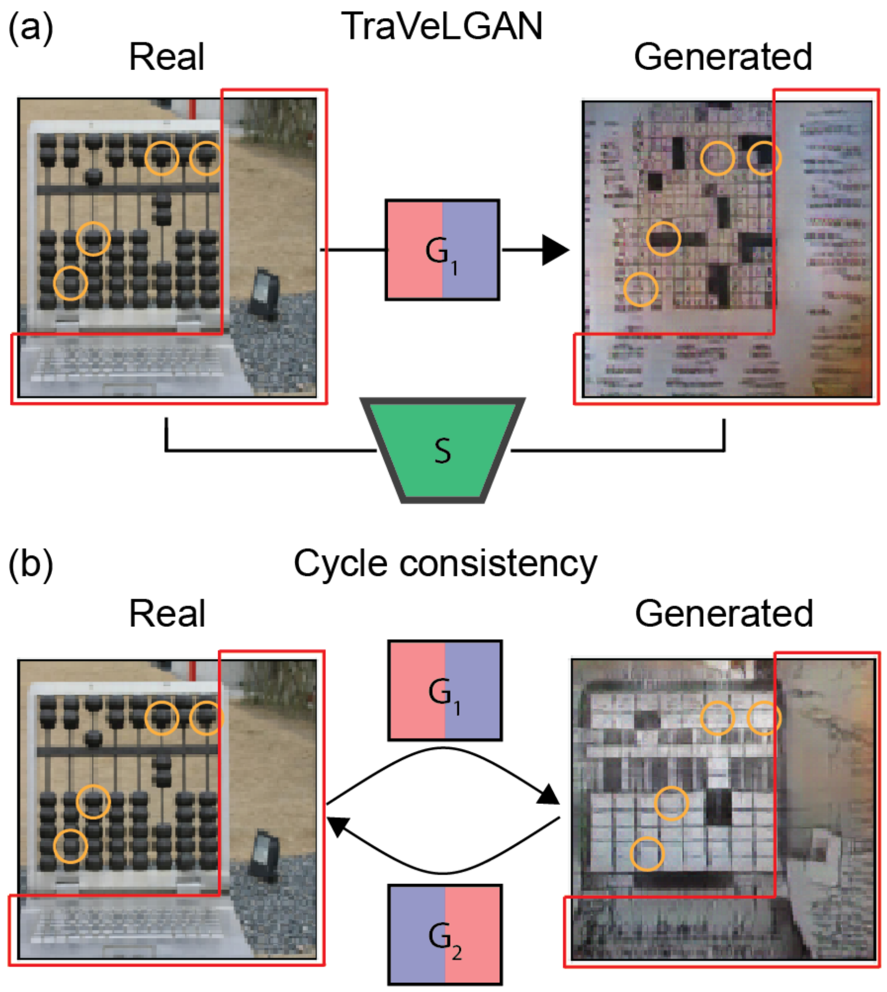


Figure 4.4: It is hard to learn mappings between domains that are each other's inverse when the domains are asymmetric (e.g. crossword configurations are more complex than abacus configurations). (a) G_1 can change the background (red selection) or black beads (orange circles) in hard-to-invert ways. (b) The cycle-consistency assumption forced every black bead to a white crossword square and every blank space to a black crossword square, even though the result is not a realistic crossword pattern. The background is also not fully changed because it could not learn that more complicated inverse function.

4.3 Experiments

Our experiments are designed around intentionally difficult image-to-image translation tasks. These translations are much harder than style or texture transfer problems, where the domain transformation can be reduced to repeating a common patch transformation on every local patch of every image without higher-level semantic information (e.g. turning a picture into a cartoon) [134, 75]. Instead, we choose domains where the differences are higher-level and semantic. For example, when mapping from horses to birds, any given picture of a horse might solely consist of style, texture, and patches that appear in other pictures of real birds (like blue sky, green grass, sharp black outlines, and a brown exterior). Only the higher-level shape and context of the image eventually reveal which domain it belongs to. Additionally, because we use datasets that are designed for classification tasks, the domains contain significant heterogeneity that makes finding commonality within a domain very difficult.

We compare the TraVeLGAN to several previous methods that first regularize the generators by enforcing cycle-consistency and then augment this with further regularizations [189, 80, 4, 134, 102, 3, 35, 12]. Namely, we compare to a GAN with just the cycle-consistency loss (cycle GAN) [189], with cycle-consistency loss plus the identity regularization (cycle+identity GAN) [189], with cycle-consistency loss plus a correspondence loss (cycle+corr GAN) [4], with cycle-consistency loss plus a feature matching regularization (cycle+featmatch GAN) [80], and with cycle-consistency loss plus a shared latent space regularization (cycle+latent GAN) [98]. The TraVeLGAN utilizes a U-net architecture with skip connections [132] for the generator. The discriminator network is a standard stride-2 convolutional classifier network that doubles the number of filters at each layer until the layer is 4x4 and outputs a single sigmoidal probability. The siamese network is identical except rather than outputting one node like the discriminator it outputs the number of nodes that is the size of the latent space, without any nonlinear activation. For the cycle-consistent GANs we compare to, we optimized the hyperparameters to get the best achievement we could, since our focus is on testing our different loss formulation. This involved trying both Resnet and U-Net architectures for the models from [189]: the U-Net performed much better than the Resnet at these tasks, so we use that here. We also had to choose a value of the cycle-consistent coefficient that largely de-emphasized it in order to get them to change the input image at all (0.1). Even so, we were not able to achieve nearly as convincing results with any of the baseline models as with the TraVeLGAN.

4.3.1 Similar domains

The datasets we first consider are traditional cases for unsupervised domain mapping with cycle-consistent networks, where little change is necessary. These are:

Apples to oranges The photos of apples and oranges from [189] (Figure 4.3a). The TraVeLGAN successfully changes not only the color of the fruit, but also the shape and texture. The stem is removed from apples, for example, and the insides of oranges aren't just colored red but fully made into apples. In the last row, the TraVeLGAN changes the shape of the orange to become an apple and correspondingly moves its shadow down in the frame to correspond.

Van Gogh to landscape photo The portraits by Van Gogh and photos of landscapes, also from [189] (Figure 4.3b). Here the prototypical Van Gogh brush strokes and colors are successfully applied or removed. Notably, in the last row, the portrait of the man is changed to be a photo of a rocky outcrop with the blue clothes of the man changing to blue sky and the chair becoming rocks, rather than becoming a photo-realistic version of that man, which would not belong in the target domain of landscapes.

Ukiyoe to landscape photo Another dataset from [189], paintings by Ukiyoe and photos of landscapes (Figure 4.3c). It is interesting to note that in the generated Ukiyoe images, the TraVeLGAN correctly matches reflections of mountains in the water, adding color to the top of the mountain and the corresponding bottom of the reflection.

CelebA glasses The CelebA dataset filtered for men with and without glasses [29] (Figure 4.3d). As expected, the TraVeLGAN produces the minimal change necessary to transfer an image to the other domain, i.e. adding or removing glasses while preserving the other aspects of the image. Since the TraVeLGAN learns a semantic, rather than pixel-wise, information preserving penalty, in some cases aspects not related to the domain are also changed (like hair color or background). In each case, the resulting image is still a convincingly realistic image in the target domain with a strong similarity to the original, though.

CelebA hats The CelebA dataset filtered for men with and without hats [29] (Figure 4.3e). As before, the TraVeLGAN adds or removes a hat while preserving the other semantics in the image.

Sketch to shoe Images of shoes along with their sketch outlines, from [144] (Figure 4.3f). Because this dataset is paired (though it is still trained unsupervised as always), we are able to quantify the performance of the TraVeLGAN with a heuristic: the pixel-wise mean-squared error (MSE) between the TraVeLGAN's generated output and the true image in the other domain. This can be seen to be a heuristic in the fourth row of Figure 4.3c, where the blue and black shoe matches the outline of the sketch perfectly, but is not the red and black color that the actual shoe happened to be. However, even as an approximation it provides information. Table 4.2 shows the full results, and while the vanilla cycle-consistent network performs the best, the TraVeLGAN is not far off and is better than the others. Given that the TraVeLGAN does not have the strict pixel-wise losses of the other models and that the two domains of this dataset are so similar, it is not surprising that the more flexible TraVeLGAN only performs similarly to the cycle-consistent frameworks. These scores provide an opportunity to gauge the effect of changing the size of the latent space learned by the siamese network. We see that our empirically chosen default value of 1000 slightly outperforms a smaller and larger value. This parameter controls the expressive capability of the model, and the scores suggest providing it too small of a space can limit the complexity of the learned transformation and too large of a space can inhibit the training. The scores are all very similar, though, suggesting it is fairly robust to this choice.

Quantitative results Since the two domains in these datasets are so similar, it is reasonble to evaluate each model using structural similarity (SSIM) between the real and generated images in each case. These results are presented in Table 4.1. There we can see that the TraVeLGAN performs comparably to the cycle-consistent models. It is expected that the baselines perform well in these cases, as these are the standard applications they were designed to succeed on in the first place; namely, domains that require little change to the

SSIM	Apple	Van Gogh	Ukiyoe	Glasses	Hats
TraVeLGAN	0.302	0.183	0.222	0.499	0.420
Cycle	**0.424**	0.216	0.252	0.463	**0.437**
Cycle+ident	0.305	**0.327**	**0.260**	**0.608**	0.358
Cycle+corr	0.251	0.079	0.072	0.230	0.204
Cycle+featmatch	0.114	0.117	0.125	0.086	0.209
Cycle+latent	0.245	0.260	0.144	0.442	0.382

Table 4.1: Real/generated SSIM on the similar-domains datasets.

Pixel MSE	Sketches	Shoes
TraVeLGAN	0.060	0.267
TraVeLGAN (D_{latent}=100)	0.069	0.370
TraVeLGAN (D_{latent}=2000)	0.064	0.274
Cycle	**0.047**	**0.148**
Cycle+corr	0.427	0.603
Cycle+featmatch	0.077	0.394
Cycle+latent	0.072	0.434

Table 4.2: Per-pixel MSE on the shoes-to-sketch dataset.

original images. Furthermore, it is expected that the TraVeLGAN changes the images slightly more than the models that enforce pixel-wise cycle-consistency. That the TraVeLGAN performs so similarly demonstrates quantitatively that the TraVeLGAN can preserve the main qualities of the image when the domains are similar.

4.3.2 Imagenet: diverse domains

The previous datasets considered domains that were very similar to each other. Next, we map between two domains that are not only very different from each other, but from classification datasets where the object characterizing the domain is sometimes only partially in the frame, has many different possible appearances, or have substantial clutter around it. In this most difficult task, we present arbitrarily chosen classes from the Imagenet [43] dataset. These images are much higher-resolution (all images are rescaled to 128x128), making it easier to learn a transfer that only needs local image patches (like style/texture transfer) than entire-image solutions like TraVeLGAN's high-level semantic mappings.

We chose classes arbitrarily because we seek a framework that is flexible enough to make translations between any domains, even when those classes are very different and arbitrarily picked (as opposed to specific domains contrived to satisfy particular assumptions). The pairs are: 1. abacus and crossword (Figures 4.2a and 4.8) 2. volcano and jack-o-lantern (Figures 4.2b and 4.11) 3. clock and hourglass (Figures 4.2c and 4.10) 4. toucan and rock

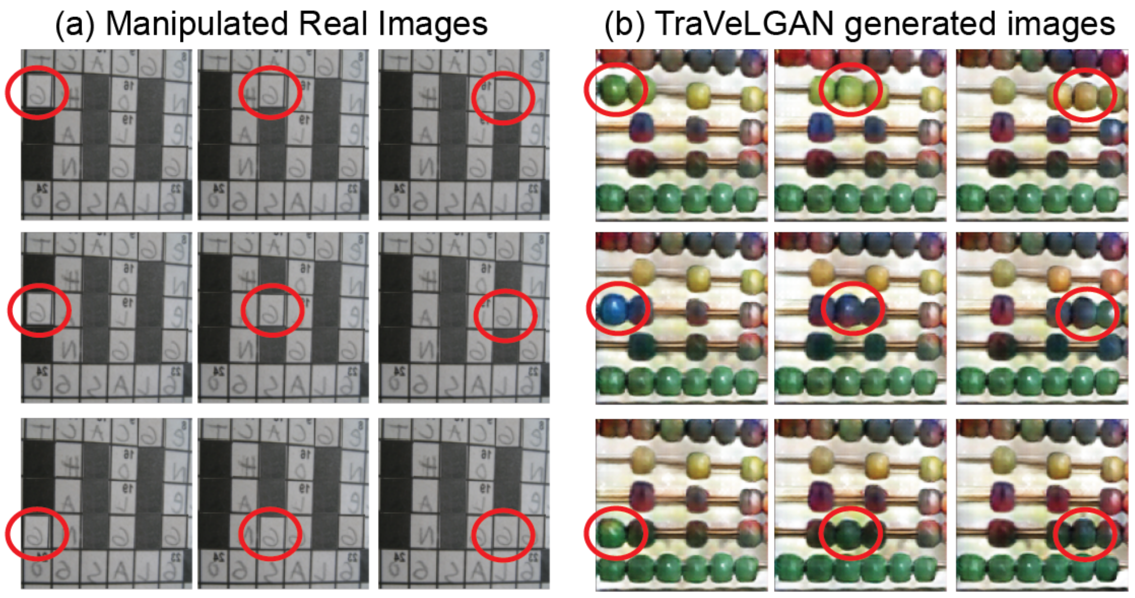


Figure 4.5: (a) A real crossword image artificially manipulated to move a white square around the frame. (b) The TraVeLGAN, which has not seen any of these images during training, has learned a semantic mapping between the domains that moves an abacus bead appropriately with the crossword square.

beauty (Figures 4.2d and 4.9).

Asymmetric domains Learning to map between the domains of abacus and crossword showcases a standard property of arbitrary domain mapping: the amount and nature of variability in one domain is larger than in the other. In Figure 4.4, we see that the TraVeLGAN learned a semantic mapping from an abacus to a crossword by turning the beads of an abacus into the white squares in a crossword and turning the string in the abacus to the black squares. However, in an abacus, the beads can be aligned in any shape, while in crosswords only specific grids are feasible. To turn the abacus in Figure 4.4 (which has huge blocks of beads that would make for a very difficult crossword indeed!) into a realistic crossword, the TraVeLGAN must make some beads into black squares and others into white squares. The cycle-consistency loss fights this one-to-many mapping because it would be hard for the other generator, which is forced to also be the inverse of this generator, to learn the inverse many-to-one function. So instead, it learns a precise, rigid bead-to-white-square and string-to-black-square mapping at the expense of making a realistic crossword (Figure 4.4b). Even though the background is an unimportant part of the image semantically, it must recover all of the exact pixel-wise values after cycling. We note that the TraVeLGAN

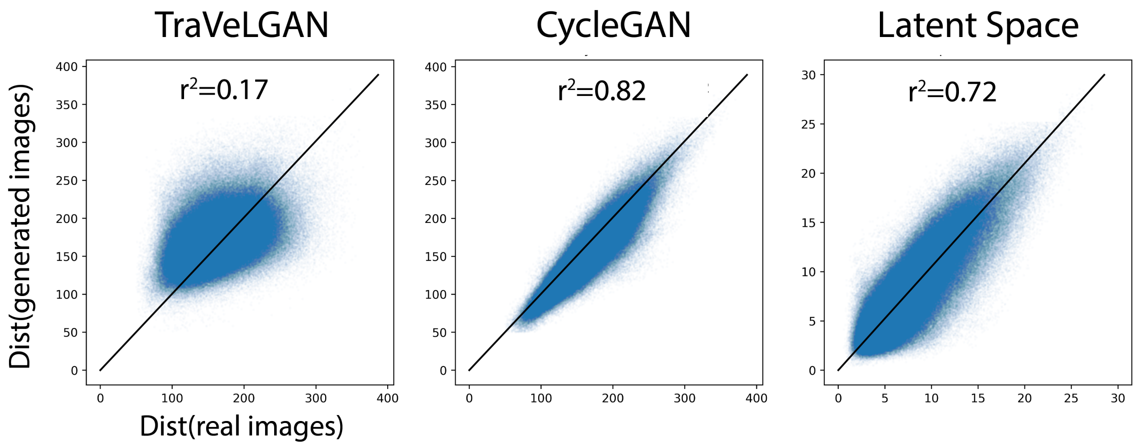


Figure 4.6: The CycleGAN generates images such that pairwise L2-distances in ***pixel space*** are strongly preserved. The TraVeLGAN generated images are virtually uncorrelated in pixel space, but the siamese network learns a ***latent space*** where pairwise distances are preserved.

automatically relaxed the one-to-one relationship of beads to crossword squares to create realistic crosswords. On the other hand, any real crossword configuration is a plausible abacus configuration. In the next section, we show that the TraVeLGAN also automatically discovered this mapping can be one-to-one in white-squares-to-beads, and preserves this systematically.

Manipulated images study Next we examine the degree to which the TraVeLGAN has learned a meaningful semantic mapping between domains. Since the Imagenet classes are so cluttered and heterogeneous and lack repetition in the form of two very similar images, we create similar images with a manipulation study. We have taken one of the real images in the crossword domain, and using standard photo-editing software, we have created systematically related images. With these systematically related images, we can test to see whether the TraVeLGAN's mapping preserves the semantics in the abacus domain in a systematic way, too.

In Figure 4.5, we started with a crossword and created a regular three-by-three grid of black squares by editing an image from Figure 4.8. Then, systematically, we move a white square around the grid through each of the nine positions. In each case, the TraVeLGAN generates an abacus with a bead moving around the grid appropriately. Remarkably, it even colors the bead to fit with the neighboring beads, which differ throughout the grid. Given that none of the specific nine images in Figure 4.5 were seen in training, the TraVeLGAN

FID score	(a)	(b)	(c)	(d)
TraVeLGAN	**1.026**	**0.032**	**0.698**	**0.206**
Cycle	1.350	1.281	1.018	0.381
Cycle+identity	1.535	0.917	1.297	1.067
Cycle+corr	1.519	0.527	0.727	0.638
Cycle+featmatch	1.357	1.331	1.084	0.869
Cycle+latent	1.221	0.485	1.104	0.543

Table 4.3: FID scores for each of the models on each of the Imagenet datasets. Column labels correspond to Figure 4.2.

Discriminator score	(a)	(b)	(c)	(d)
TraVeLGAN	**0.035**	**0.206**	**0.074**	**0.145**
Cycle	0.014	0.008	0.033	0.008
Cycle+identity	0.011	0.044	0.040	0.064
Cycle+corr	0.009	0.191	0.026	0.001
Cycle+featmatch	0.002	0.029	0.066	0.014
Cycle+latent	0.009	0.069	0.047	0.039

Table 4.4: Discriminator scores for each of the models on each of the Imagenet datasets. Column labels correspond to Figure 4.2.

has clearly learned the semantics of the mapping rather than memorizing a specific point.

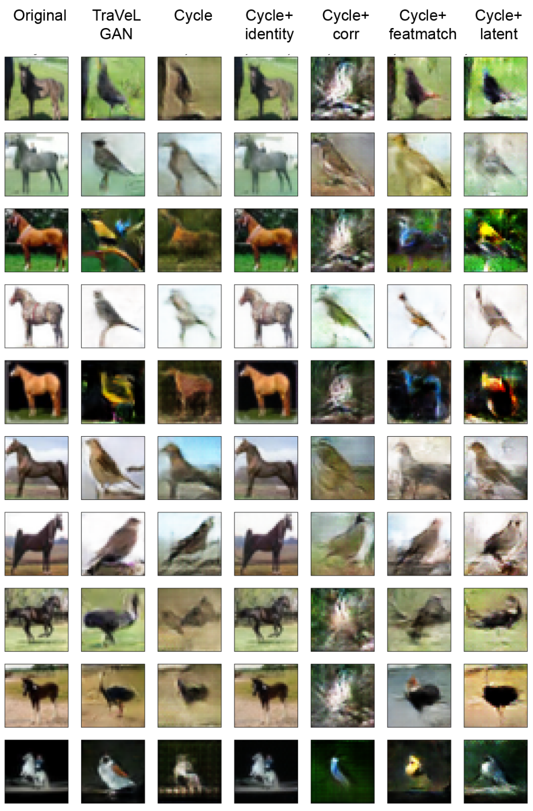
Pairwise distance preservation The DistanceGAN [18] has shown that approximately maintaining pairwise distances between images in ***pixel space*** achieves similar success to the cycle-consistent GANs. In fact, they show that cycle-consistent GANs produce images that preserve the pixel pairwise distance between images with extremely highly correlation. On the toucan to rock beauty dataset, we observe the same phenomenon ($r^2 = 0.82$ in Figure 4.6). While this produced plausible images in some cases, maintaining pixel-wise distance between images could not generate realistic toucans or rock beauties. The TraVeLGAN pairwise distances are virtually uncorrelated in pixel space ($r^2 = 0.17$). However, we understand the role of the siamese network when we look at the pairwise distances between real images in ***latent space*** and the corresponding pairwise distances between generated images in ***latent space***. There we see a similar correlation ($r^2 = 0.72$). In other words, the TraVeLGAN simultaneously learns a mapping with a neural network to a space where distances can be meaningfully preserved while using that mapping to guide it in generating realistic images.

Quantitative results Lastly, we add quantitative evidence to the qualitative evidence already presented that the TraVeLGAN outperforms existing models when the domains are very different. While we used the SSIM and pixel-wise MSE in the previous sections to evaluate success, neither heuristic is appropriate for these datasets. The goal in these mappings is *not* to leave the image unchanged and as similar to the original as possible, it is to fully change the image into the other domain. Thus, we apply two different metrics to evaluate the models quantitatively on these Imagenet datasets.

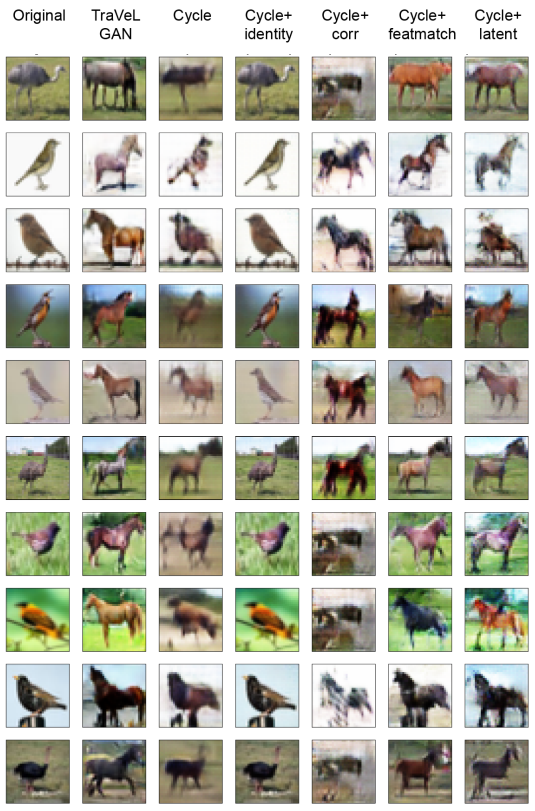
In general, quantifying GAN quality is a hard task [17]. Moreover, here we are specifically interested in how well a generated image is paired or corresponding to the original image, point-by-point. To the best of our knowledge, there is no current way to measure this quantitatively for arbitrary domains, so we have pursued the qualitative evaluations in the previous sections. However, in addition to those qualitative evaluations of the correspondence aspect, we at least quantify how well the generated images resemble the target domain, at a population level, with heuristic scores designed to measure this under certain assumptions. The first, the Fréchet Inception Distance (FID score) [47] is an improved version of the Inception Score (whose flaws were well articulated in [17]) which compares the real and generated images in a layer of a pre-trained Inception network (Table 4.3). The second, the discriminator score, trains a discriminator from scratch, independent of the one used during training, to try to distinguish between real and generated examples (Table 4.4). The TraVeLGAN achieved better scores than any of the baseline models with both metrics and across all datasets.

4.3.3 Discussion

In recent years, unsupervised domain mapping has been dominated by approaches building off of the cycle-consistency assumption and framework. We have identified that some cluttered, heterogeneous, asymmetric domains cannot be successfully mapped between by generators trained on this cycle-consistency approach. Further improving the flexibility of domain mapping models may need to proceed without the cycle-consistent assumption, as we have done here.

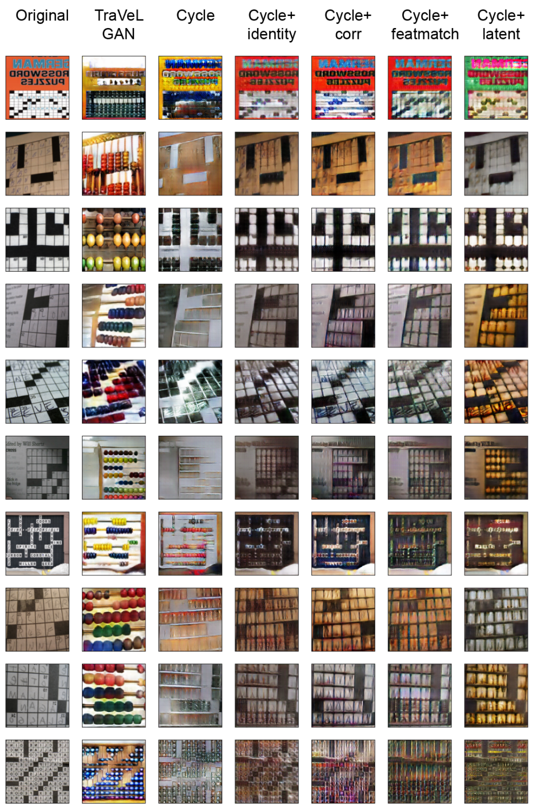


(a)

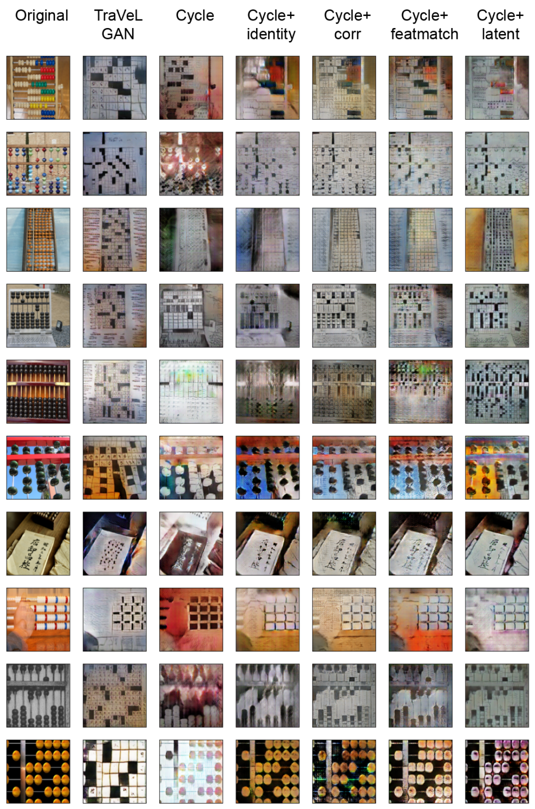


(b)

Figure 4.7: Learning to map between two CIFAR domains: (a) horse to bird (b) bird to horse.

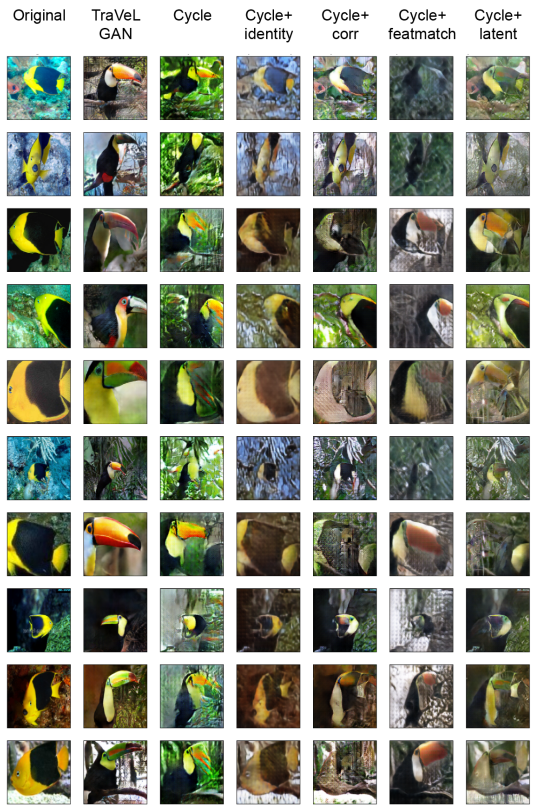


(a)

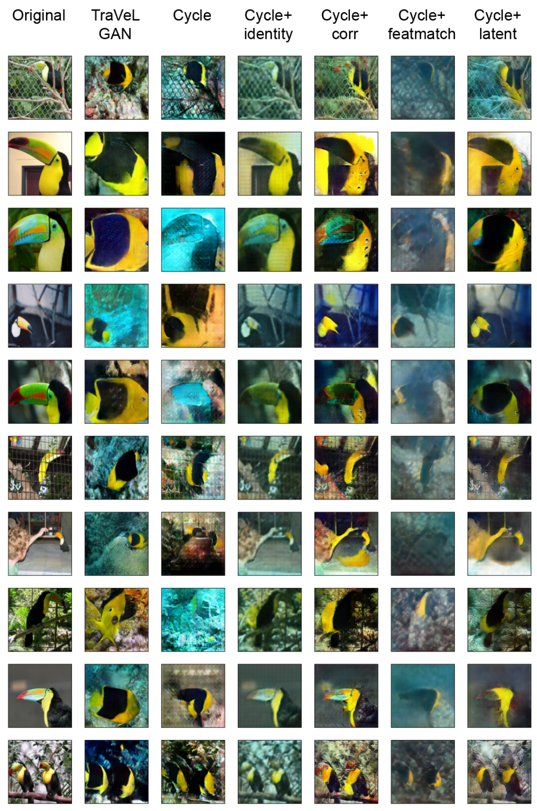


(b)

Figure 4.8: Learning to map between two imagenet domains: (a) crossword to abacus (b) abacus to crossword.

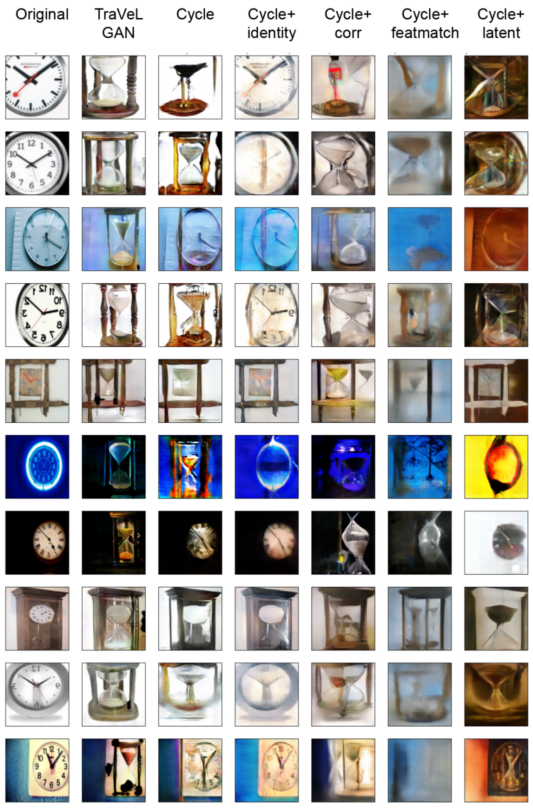


(a)

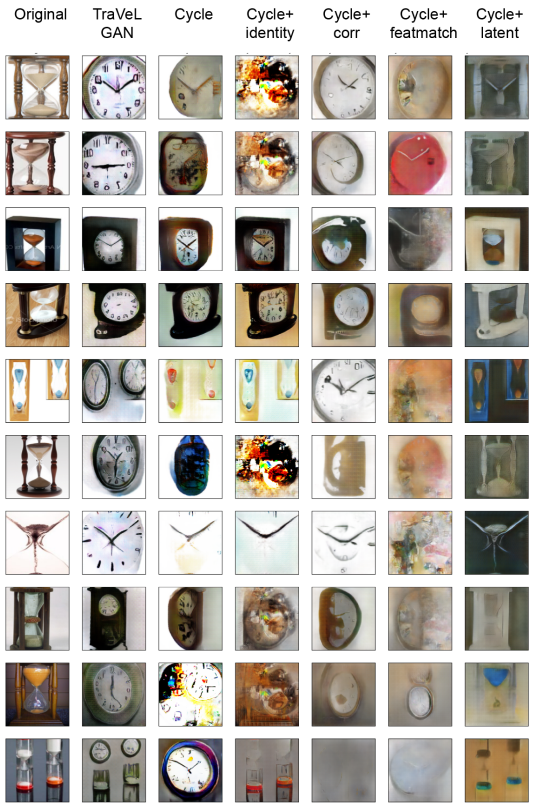


(b)

Figure 4.9: Learning to map between two imagenet domains: (a) rock beauty to toucan (b) toucan to rock beauty.

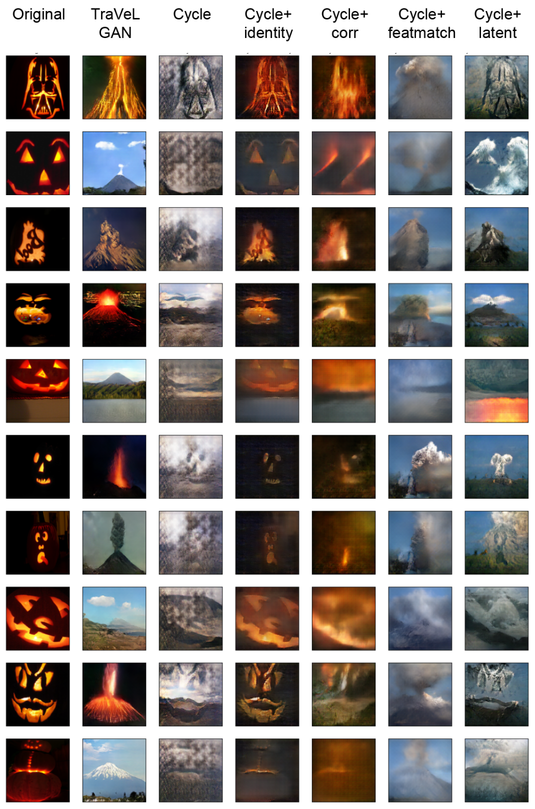


(a)

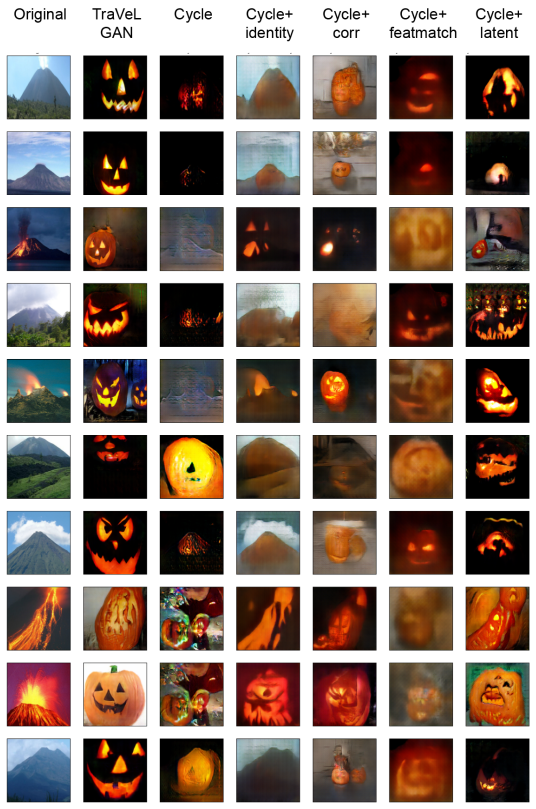


(b)

Figure 4.10: Learning to map between two imagenet domains: (a) clock to hourglass (b) hourglass to clock.

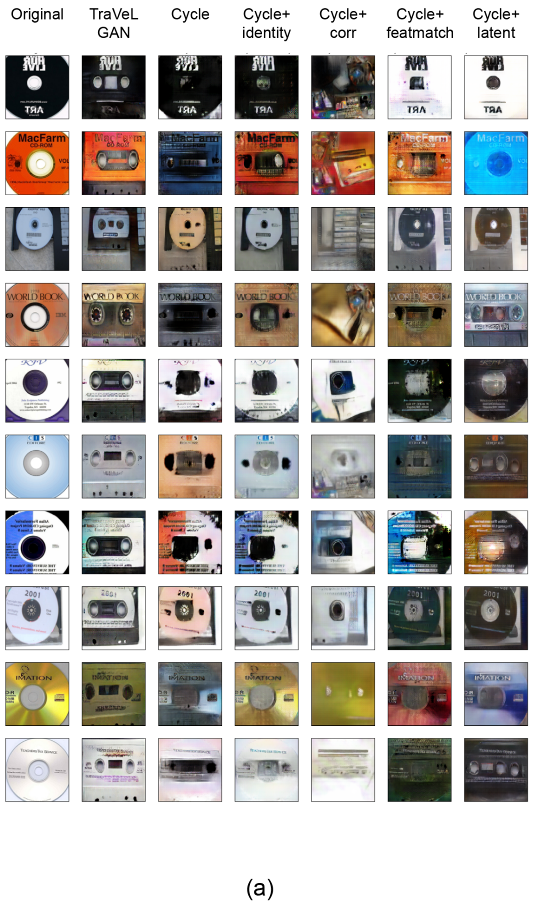


(a)



(b)

Figure 4.11: Learning to map between two imagenet domains: (a) jack-o-lantern to volcano (b) volcano to jack-o-lantern.



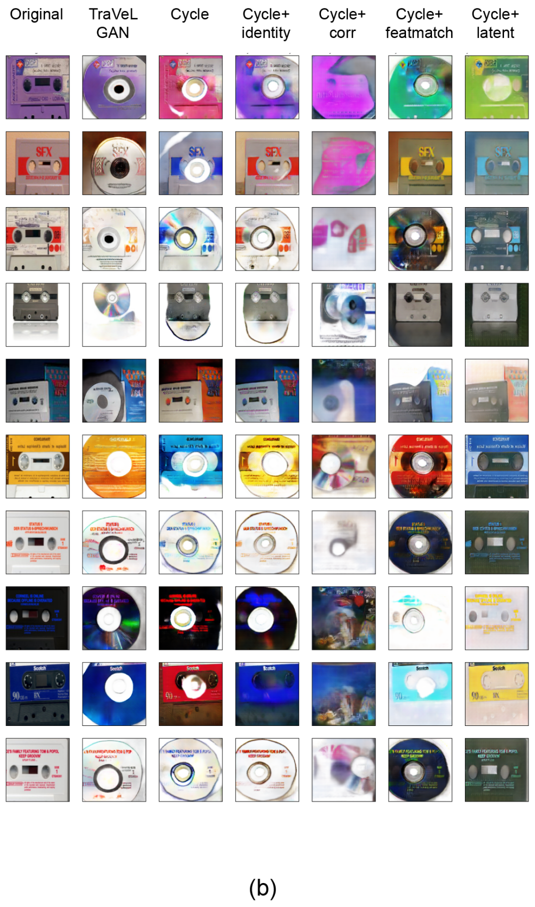


Figure 4.12: Learning to map between two imagenet domains: (a) cd to cassette (b) cassette to cd.

4.4 Final notes

Quantitative results Quantitative results are summarized by the FID score (Table 4.3) and the discriminator score (Table 4.4). We note that these scores were both designed to evaluate models that attempt to generate the full diversity of the Imagenet dataset, while in our case we only map to a single class.

The Fréchet Inception Distance (FID score) [47] calculates the Fréchet distance between Gaussian models of the output of a the pre-trained Inception network [149] on real and generated images, respectively. Lower distances indicate better performance. The results are the mean of the scores from each direction.

The discriminator score is calculated by training a new discriminator, distinct from the one used during training, to distinguish between real and generated images in a domain. A score of zero means the discriminator was certain every generated image was fake, while higher scores indicate the generated images looked more like real images. As in the FID, the results are the mean of the scores from each direction.

Optimization and training parameters Optimization was performed with the adam [81] optimizer with a learning rate of 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.9$. Gradient descent was alternated between generator and discriminator, with the discriminator receiving real and generated images in distinct batches.

Architecture The TraVeLGAN architecture is as follows. Let d denote the size of the image. Let c_n be a standard stride-two convolutional layer with n filters, t_n be a stride-two convolutional transpose layer with kernel size four and n filters, and f_n be a fully connected layer outputting n neurons. The discriminator D has layers until the size of the input is four-by-four, increasing the number of filters by a factor of two each time, up to a maximum of eight times the original number (three layers for CIFAR and five layers for Imagenet). This last layer is then flattened and passed through a fully connected layer. The overall architecture is thus $c_n - c_{2n} - c_{4n} - c_{8n} - c_{8n} - f_1$. The siamese network has the same structure as the discriminator except its latent space has size 1000, yielding the architecture $c_n - c_{2n} - c_{4n} - c_{8n} - c_{8n} - f_{1000}$. The generator uses the U-Net architecture [132] that has skip connections that concatenate the input in the symmetric encoder with the decoder, yielding layers of $c_n - c_{2n} - c_{4n} - c_{4n} - c_{4n} - t_{8n} - t_{8n} - t_{8n} - t_{4n} - t_{2n} - t_3$. For the cycle-consistency networks, the architectures of the original implementations were used, with code from [189], [189], [4], [80], for the cycle, cycle+identity, cycle+corr, and cycle+featmatch, respectively. All activations are leaky rectified linear units with leak of 0.2, except for the output layers, which use sigmoid for the discriminator, hyperbolic tangent for the generator, and linear for the siamese network. Batch normalization is used for every layer except the first layer of the discriminator. All code was implemented in Tensorflow [2] on a single NVIDIA Titan X GPU.

CIFAR While the CIFAR images [88] are relatively simple and low-dimensional, it is a deceptively complex task compared to standard domain mapping datasets like CelebA, where they are all centered close-ups of human faces (i.e. their shoulders or hair are in the same pixel locations). The cycle-consistent GANs struggle to identify the characteristic shapes of each domain, instead either only make small changes to the images or focusing on the color tone. The TraVeLGAN, on the other hand, fully transfers images to the target domain. Furthermore, the TraVeLGAN preserves semantics like orientation, background color, body color, and composition in the pair of image (complete comparison results in Figure 4.7)

Interpretability As the siamese latent space is learned to preserve vector transformations between images, we can look at how that space is organized to tell us what transformation the network learned at a dataset-wide resolution. Figure 4.13 shows a PCA visualization of the siamese space of the CIFAR dataset with all of the original domain one (bird) and domain two (horse) images. There we can see that S learned a logical space with obvious structure, where mostly grassy images are in the bottom left, mostly sky images in the top right, and so forth. Furthermore, the layout is analogous between the two domains, verifying that the network automatically learned a notion of similarity between the two domains. We also show every generated image across the whole dataset in this space, where we see that the transformation vectors are not just interpretable for some individual images and not others, but are interpretable across the entire distribution of generated images.

Salience We next perform a salience analysis of the TraVeL loss by calculating the magnitude of the gradient at each pixel in the generated image with respect to each pixel in the original image (Figure 4.14). Since the TraVeL loss, which enforces the similarity aspect of the domain mapping problem, is parameterized by another neural network S, the original image contributes to the generated image in a complex, high-level way, and as such the gradients are spread richly over the entire foreground of the image. This allows the generator to make realistic abacus beads, which need to be round and shaded, out of square and uniform pixels in the crossword. By contrast, the cycle-consistency loss requires numerical precision in the pixels, and as such the salience map largely looks like a grayscale version of the real image, with rigid lines and large blocks of homogeneous pixels still visible. This is further evidence that the cycle-consistency loss is preventing the generator from making round beads with colors that vary over the numerical RGB values.

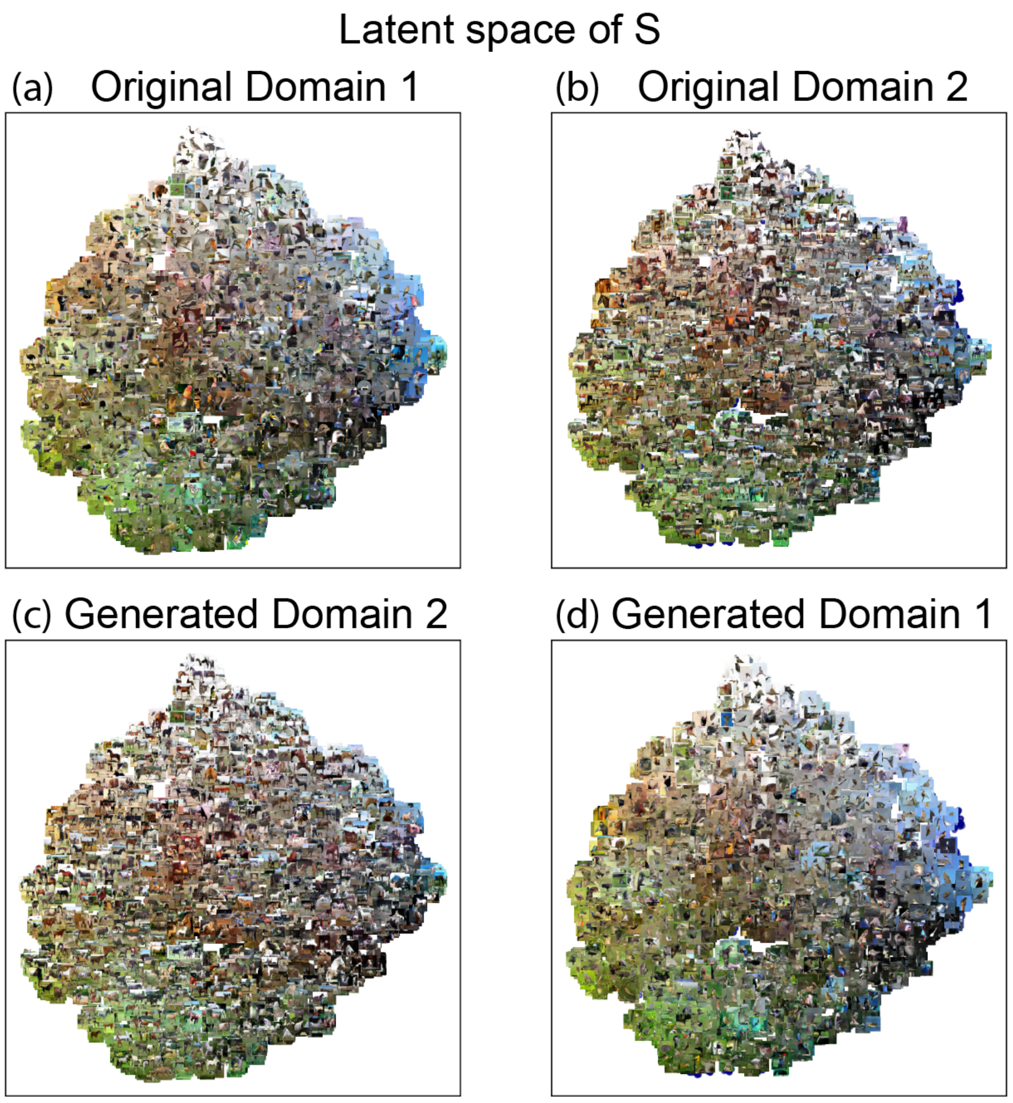


Figure 4.13: Having access to the siamese space output by S provides an interpretability of the TraVeLGAN's domain mapping that other networks lack. PCA visualizations on the CIFAR example indicate S has indeed learned a meaningfully organized space for G to preserve transformation vectors within.

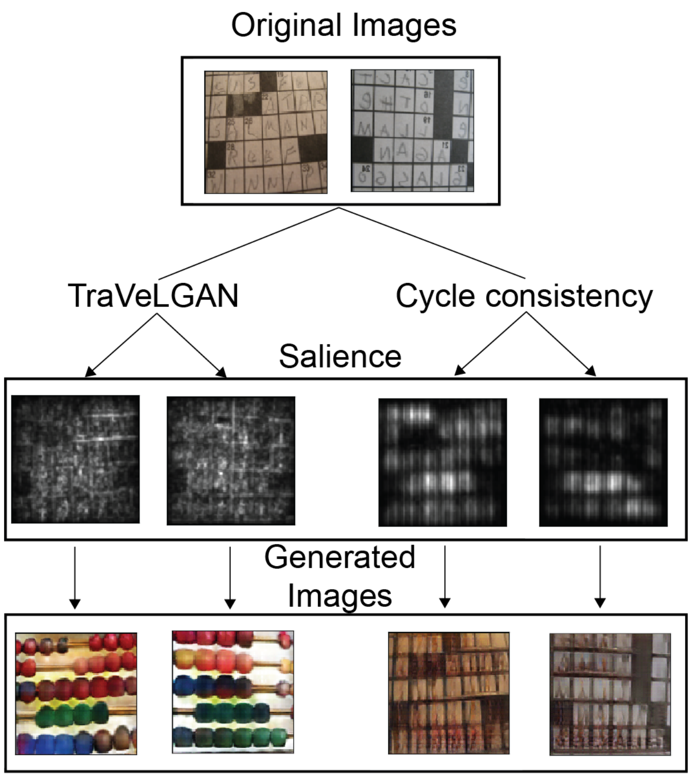


Figure 4.14: A salience analysis exploring how the TraVeLGAN's objective loosens the restriction of cycle-consistency and allows it more flexibility in changing the image during domain transfer. The TraVeL loss requires significantly less memorization of the input pixels, and as a result, more complex transformations can be learned.

Chapter 5

Feature Mapping GAN (FMGAN)

5.1 Introduction

Often when biological entities are measured in multiple ways, there are distinct categories of information: some information is easy-to-obtain information (EI) and can be gathered on virtually every subject of interest, while other information is hard-to-obtain information (HI) and can only be gathered on some. We propose building a model to make probabilistic predictions of HI using EI. Our Feature Mapping GAN (FMGAN), based on the conditional GAN framework, uses an embedding network to process conditions as part of the conditional GAN training to create manifold structure when it is not readily present in the conditions. We experiment on generating RNA sequencing of cell lines perturbed with a drug conditioned on the drug's chemical structure and generating FACS data from clinical monitoring variables on a cohort of COVID-19 patients, effectively describing their immune response in great detail.

When collecting information on biological entities, for example hospital patients, cells, or drugs, we are often faced with the choice of collecting easy-to-obtain information (EI) on many entities or collecting hard-to-obtain information (HI) on a few entities. For example, in a drug library of millions of drugs, it is easy to obtain chemical structure information but hard to obtain RNA sequencing information of cells treated with drugs. On patients, it may be easy to obtain information such as heart rate and lab values, but hard to obtain blood flow cytometry information. Here, we present a neural network-based method that can bridge the gap between these sources of information on entities like drugs or patients.

We introduce a framework based on a conditional Generative Adversarial Network (cGAN) that we call Feature Mapping GAN (FMGAN), which learns a mapping from EI to a distribution of HI. The FMGAN takes in *noise* as input, the *EI information* as the condition and the *distribution of HI for that EI* as the output. Unique to the FMGAN is an auxiliary network called the *condition-embedding* network. This network processes the EI information in order to discover its latent manifold dimensions, from which the mapping can be smoother or more regular to learn. To illustrate the utility of this, consider a simple linear mapping between an EI variable and an HI variable. The linearity guarantees that small changes in the EI will result in a small change in the HI, i.e. a mapping mapping is *smooth*. However, such a mapping is only possible if the EI is linearly related to the HI. With chemical structure, for example, this is known not to be true: a small change in chemical structure can lead to vastly different properties of a drug. Therefore, our condition-embedding network embeds or finds alternative coordinates for our conditions that lie in low dimensional spaces (like lines or planes) thereby discovering latent structure that can render the mapping smooth. In the case of chemical structure, we use a convolutional neural network which performs a complex mapping into manifold dimensions. This is key as the addition of convolutional architectures allows the FMGAN to integrate conditions that have image structure with tabular, non-image transcriptomic measurements.

After the EI condition is appropriately processed to discover latent dimensions, it is passed to the generator to allow for a stochastic mapping. Since the HI information is a *distribution* of cells in a RNA sequencing or FACS sample, a model needs to generate data at the resolution of single cells given a condition, but also generate a variety of different cells for that individual condition. Moreover, in other cases it is unlikely that the EI has complete information about the drug or patient in question, and thus it is important for each EI condition to be able to map to a range or a distribution of HI conditions. For example, replicates of a drug perturbation experiment result in different gene expression results even when applied on the same cell line [148]. This stochastic response can only be captured by a generative model that can produce stochastic output.

We showcase FMGAN on two main application areas, drug discovery and clinical

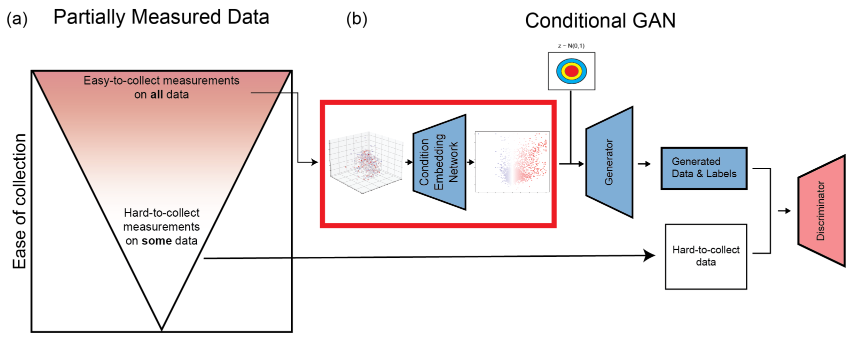


Figure 5.1: (a) The measurements on data are separated into "easy-to-collect information" (EI) and "hard-to-collect information" (HI). The easy-to-collect measurements are available on all data, while the hard-to-collect measurements are only available on some data. (b) With a Conditional GAN, we can learn to model the relationship between these two categories of measurements. The conditions go through the condition-embedding network to find manifold structure for the generator to utilize even if it is not initially present in the conditions.

prediction. Here we specifically consider measurements involving drug perturbations, a commonly used technique for measuring the effect of a drug [59, 89, 87]. We utilize drug perturbation data from the L1000 Connectivity Map dataset [148]. Because perturbing a cell line with a drug involves physically performing an experiment, including obtaining the cells, applying the drug, and getting the sequencing results, this process can be expensive and time-consuming. We use the FMGAN to generate the RNA-sequencing results from the drug structure to speed this process up by not having to perform all of the experiments exhaustively. If only a subset of the drugs have *a priori* RNA-sequencing measurements, the rest can be generated with the FMGAN, obviating the need for additional experimentation on a large number of candidates.

We also use these experiments to compare different ways of representing the chemical structure of drugs which we use as EI. We compare string and image based representations, and by measuring how well the FMGAN models the HI data in each case, we can draw conclusions about the different representations. For example, we show that pictorial diagrams of chemical structure are more effective than string sequences when all other things are held equal.

Another motivating setting is that of **clinical data**. In the clinical setting, some measurements are readily available EI, either because they are already measured as part of the standard patient monitoring, or because they are non-invasive and do not pose any risk.

The clinical dataset we work with uses as EI these clinical measurements from COVID-19 patients from Yale New Haven Hospital. In this case, the HI information are future single-cell flow cytometry measurements from samples gathered on some of the patients. In practice, these types of single-cell measurements cannot be performed exhaustively on every patient in the clinic, for reasons of cost as well as time sensitivity. Thus, we use the FMGAN to be able to generate future flow cytometry data which depicts compartments of the immune system from readily available clinical data. With the FMGAN, we are then able to generate flow cytometry data for any number of patients who only have clinical measurements available. This can be valuable as immune responses have been shown to be predictive of mortality in COVID-19 [103].

5.2 Related Work

5.2.1 Conditional GANs

The FMGAN builds off of the conditional GAN (cGAN) framework, which differs from a regular GAN by learning to model data distributions that are conditioned upon a label that is supplied along with noise as input to the generator. In our FMGAN, we use the terminology “easy-to-obtain information” for the conditional label and “hard-to-obtain information” for the data distribution; we do this to emphasize the broad applicability of the FMGAN. In our applications, the EI is not a “label” for the HI in any classical sense: for example, it would be odd to call a patient’s present monitoring data a label for that patient’s future FACS data. Thus, our use of EI and HI are intended to help move beyond the scope of traditional cGAN applications.

Those traditional cGAN applications are usually similar to its first application: image generation. In image generation contexts the condition referred to what type of image should be generated (e.g. a dog). The cGAN is able to generate a *distribution* that is conditioned on the label, for example generating a wide variety of images of dogs when given the conditional label for dogs. Canonical use cases for cGANs use simple integer conditions of non-overlapping classes, like the CIFAR dataset where there are 10 distinct classes with 6000 images per class, all known *a priori* [88]. In cases like this, a human has already separated all images of one group from those of the others and the labels have no relation to each other: class 1 is not more similar to class 2 than it is class 9 or 10. More importantly, this setting offers no ability to extrapolate. When trained on 10 conditional labels, no network is able to extrapolate to an unseen 11th condition because it doesn't have any information about how it relates to the conditions it trained on. This is because previous cGAN models do not model the *condition space*, instead only modeling each individual condition label separately. In the FMGAN, we model the condition space itself via the condition-embedding network. This allows for extrapolation to never-before-seen conditions, using the functional model of the condition space it learned from the training data.

5.2.2 Biological applications of conditional GANs

Previous work has used cGANs for biological applications in ways that are related to, but in crucial ways differ from our method. Some approaches have modeled single-cell transcriptomic data, as we do, but have predicted the expression of some target genes using the value of other landmark genes, not using any meta-data separate from the expression matrix at all [170]. Several other works used cell-type conditional labels, but have used *integer* conditions. These integers represent manually labeled cell types in some cases [175], or clusters identified from the transcriptomic data itself via an off-the-shelf clustering model [110]. To the best of our knowledge, no previous work has learned conditional generation based on processing sequential or image representations of chemicals.

5.2.3 Learning embeddings of biological data manifolds

In this work, we build off of the observation that biological data in a high-dimensional ambient space often exhibits manifold structure in a low-dimensional latent space. We assume that the conditions in our framework form such a manifold, and that learning to generate off of their manifold coordinates leads to an easier mapping problem than doing so off of their ambient coordinates. In the FMGAN, we assume that the conditions form such a space. Much previous work has shown biological datasets can be meaningful represented as low-dimensional manifolds [117, 54, 7, 145, 111].

5.3 Results

5.3.1 FMGAN

The FMGAN consists of a conditional GAN where the condition input is given by an auxiliary network called the *condition-embedding* network. This network that processes conditions is trained adversarially during the GAN training to best optimize the conditions for mapping between EI and HI.

A standard GAN learns to map from random stochastic input $z \sim N(0,1)$ (or a similarly simple distribution) to the data distribution by training G and D in alternating gradient descent with the following objective:

$$\min_G \max_D \mathbb{E}_{x \sim P(x)} log(D(x)) + \mathbb{E}_{z \sim P_z} log(1 - D(G(z)))$$

The generator in a cGAN receives both the random stochastic input z and a conditional label l and thus has the following objective:

$$\min_G \max_D \mathbb{E}_{x_l \sim P(x|l)} log(D(x|l)) + \mathbb{E}_{z \sim P_z} log(1 - D(G(z|l)))$$

5.3.2 Condition-Embedding Network

Condition-embedding networks on each side of the generator/discriminator adversarial process, a feature of the FMGAN that distinguishes it from previous models, allow for conditions to be used in a cGAN even when the information in the conditions is not in an easy-to-access form. This is because most biological data is known to have manifold structure [117].

Often, this manifold structure is not present in the original space the data exists in, but instead exists in a latent lower-dimensional space. In the original data space for the EI, transformations from the EI to the HI may be highly non-smooth and irregular. For example, consider shifting a particular element in a chemical formula. Shifting this element's location by several spaces may result in no change or minor change to the molecule's effect on a cell system. Then, a critical point is reached, when one more shift fundamentally alters the structure as observable by the perturbed cell system.

The condition-embedding network transforms the representations of these chemical structures into latent manifold representations (as seen in Figure 5.1). Small, smooth movements in these manifold coordinates results in small, smooth observable effects in the perturbed cell system. Thus, while in ambient data coordinates all shifts to the chemical structure are equal, in the manifold coordinates, only the ones that change the molecule's function result in movement on the manifold.

Specifically, the condition-embedding network learns a mapping that produces a d_2-dimensional embedding m from a d_1-dimensional condition l:

$$\{l_1, l_2, \dots l_{d_1}\} \rightarrow \{m_1, m_2, \dots m_{d_2}\}$$

, where $M_i = \{m_1, m_2, \dots m_{d_2}\}$ for condition i. The function learned is such that $M_1 - M_2$ implies an empirical bound on the difference in generated distributions P_{M_1} and P_{M_2}. We demonstrate this experimentally on our drug structure data in a later section.

The condition-embedding network, parameterized as either a fully-connected neural network or as a convolutional neural network when the ambient data has structure that convolutions can process, uncovers the latent manifold and transforms points into these coordinates, subsequently passing them to the generator.

This rendering of the EI into manifold structure allows for generalization, i.e., it learns a landscape of the EI, for instance a landscape of drugs of which some examples of the drug effects on cells can be used to infer the effects of neighboring drugs (for more discussion of manifold structure, please see the supplementary information). After training the condition-embedding network, the FMGAN can generate from never-before-seen conditions, as it has a functional model of the *condition space* as well as the generated data space.

The framework of the FMGAN is summarized in Figure 5.1. The information on each data point is separated into easy-to-collect information (EI) and hard-to-collect information (HI). In the notation of the GAN, we use the EI as the conditional label l and the HI as the data x. For data points that have both, we process the condition and train the FMGAN with the generator receiving a label l and a noise point z, while the discriminator receives the label l and both real points x and the generated points $G(z|l)$. Then, after training, the generator can generate points for conditions l without known data x. This allows us to impute HI where we only have EI.

5.3.3 Modeling drug perturbation experiments

We first demonstrate the results of our FMGAN model on data from the L1000 Connectivity Map (CMap) dataset [148]. The CMap dataset contains a matrix of genes by count values on various cell lines under different drug perturbations. We examine the A375 cell line, a cell line from a human diagnosed with malignant melanoma. In this densely measured dataset, we have all gene expression measurements for each drug. Each drug also has various numbers of replicates of the same experiment. These replicates produce variable effects, motivating the need for a framework that is capable of modeling such stochasticity.

We design four separate experiments with this dataset:

1. A proof-of-concept that the cGAN framework can effectively model and predict gene expression values when the conditions are known to be meaningful because they are selected holdout genes from the expression matrix itself.

2. An experiment where the conditions are taken from a non-linear dimensionality reduction method applied to the expressions, and thus do not need significant processing to make them a usable data manifold.

3. A test of the full FMGAN pipeline where conditions represent chemical structure in the form of SMILES strings, and thus do not provide information about the drug in a readily available numeric form, and meaningful embeddings for conditions must be learned.

4. A variation of the chemical structure conditions where they are represented as images of the chemical structure diagram as opposed to SMILES strings.

In each dataset, the measurement we choose for evaluation is maximum mean discrepancy (MMD) [46]. We choose this because we require a metric that is a distance between distributions, not a distance merely between points. Taking the mean of a distance between points would not capture the accuracy of any moments in the desired distribution beyond the first one. For the experiments based on drug metadata (the SMILES strings and the chemical structure images experiments), we consider the drug's distribution to be all of the gene profiles from that drug. For the experiments with conditions derived from each gene profile (the heldout genes and dimensionality-reduction experiments), we take a neighborhood of drugs around each condition and compare the predicted distribution of gene profiles for those drugs with the true distribution.

We make several comparisons to our FMGAN with each dataset. We first compare to a simpler model that takes in the condition and stochastic noise and minimizes mean-squared-error (MSE) between the output of a linear transformation and the real gene profile for that condition. As it is given noise input as well as a condition, it is still able to generate whole distributions as predictions for each condition, rather than deterministic single points. Secondly, we compare to a variational autoencoder (VAE) model that also receives the condition and must produce the real gene profile for that condition. The VAE then stochastically generates from its latent layer when needing to generate an entire distribution of points from a single condition. Finally, we compare to two models just like the FMGAN but without one aspect of the full model (an ablation test). We compare to an FMGAN that does not use our novel condition-embedding network, and then we compare to an FMGAN that has the condition-embedding network but uses a MSE loss rather than an adversarial GAN loss. These ablation tests showcase the crucial role both the condition-embedding network and the adversarial training play in the overall FMGAN framework. We note in developing baselines, since generating conditional distributions (especially based off of oddly structured conditions like images or strings) is relatively understudied in the computational biology field, we find no directly comparably published methods that can be applied to this problem.

Predicting gene expression under drug perturbation

To show our FMGAN can learn informative mappings from the EI space to the gene expression space, as distinct from the rest of the process, we first choose a means of obtaining EI that are known to be meaningfully connected to the gene expression space. Specifically,

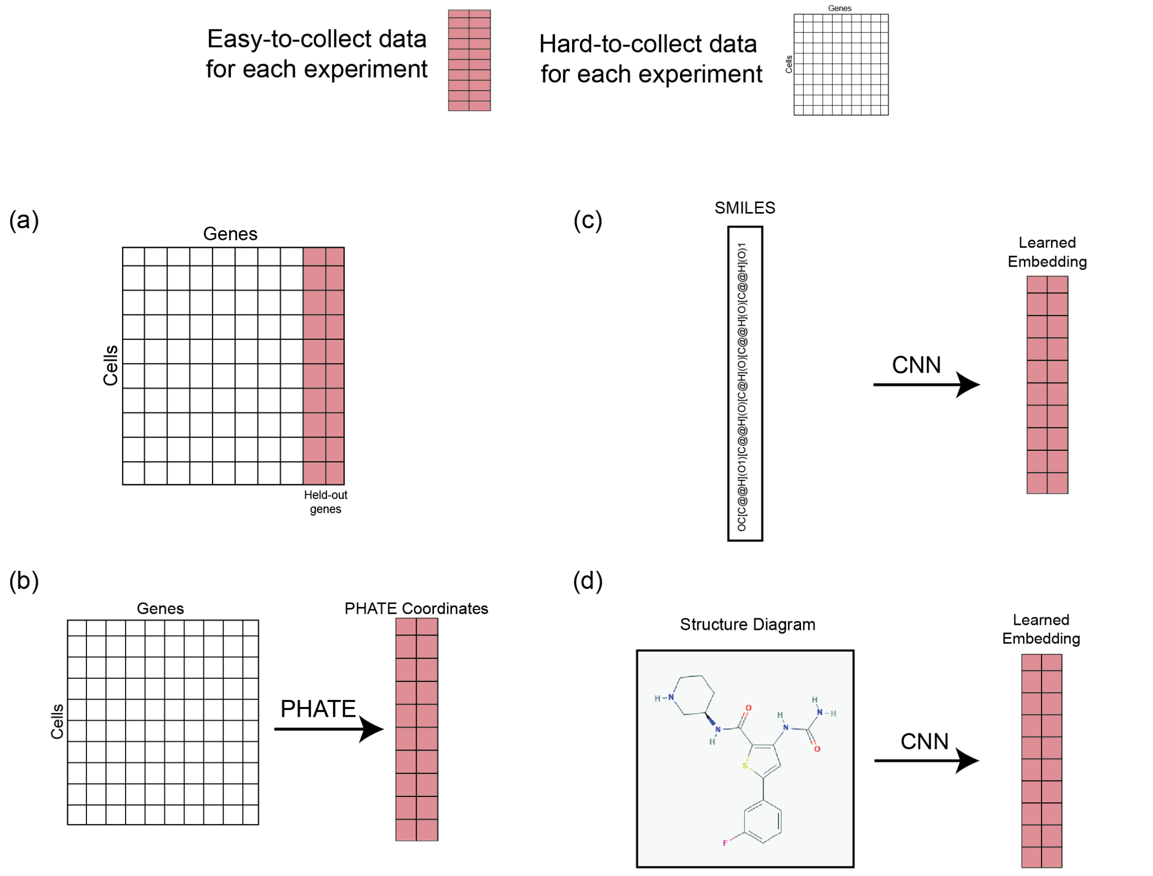


Figure 5.2: The formation of easy-to-collect (red columns) and hard-to-collect (white columns) data for each experiment with drug perturbation data. (a) in the held-out genes experiment, the easy-to-collect measurements are taken from held-out genes (b) in the PHATE coordinate experiment, they are the result of running on the genes matrix (c) in the SMILES string experiment, the easy-to-collect data is an embedding from processing this representation with a CNN (d) in the structure diagram experiment, it is the same as in the SMILES string experiment except run on the structure diagrams.

we artificially hold out ten genes and use their values as EI, with the FMGAN tasked with generating the values for all other genes.

This experimental design is summarized in Figure 5.2a. We choose the ten genes algorithmically by selecting one randomly and then greedily adding to the set the one with the least shared correlation with the others, to ensure the information in their values have as little redundancy as possible: PHGDH, PRCP, CIAPIN1, GNAI1, PLSCR1, SOX4, MAP2K5, BAD, SPP1, and TIAM1. In addition to dividing up the gene space to use these ten genes to predict all of the rest, we also divide up the cell space and train on 80% of the cell data, with the last 20% heldout for testing.

MMD Scores	Heldout Genes	PHATE Coordinates	SMILES	Chemical Structure Image
FMGAN (full model)	**2.841 +/- 0.006**	**0.213 +/- 0.008**	**1.232 +/- 0.005**	**1.219 +/- 0.007**
FMGAN (no condition-embedding)	2.883 +/- 0.005	0.220 +/- 0.007	1.307 +/- 0.006	1.621 +/- 0.010
FMGAN (no GAN)	2.956 +/- 0.003	0.424 +/- 0.005	1.482 +/- 0.009	1.511 +/- 0.011
Linear	2.912 +/- 0.006	0.533 +/- 0.001	1.565 +/- 0.005	1.772 +/- 0.010
VAE	2.962 +/- 0.003	0.497 +/- 0.002	1.886 +/- 0.0035	2.012 +/- 0.003

Table 5.1: MMD scores (lower is better) across all datasets for the drug data for all models with mean and standard deviation reported across five independent runs. The full FMGAN with all of its components most accurately predicts the distribution from each condition for all methods of forming the condition space, although the datasets that require more advanced convolutional processing benefit the most.

We find our FMGAN is able to successfully leverage information in the EI space to accurately model the data. We designed our proof of concept deliberately so that the true values are known for each gene expression and drug we ask our network to predict. These real heldout values can be compared to the predictions with MMD for a measure of accuracy.

As shown in Table 5.1, our FMGAN is able to generate predictions with the lowest MMD between them and the never-before-seen evaluation set (drugs it has never previously seen), showing it very effectively learned to model the dependency structure between the EI space and the HI space. The table reports the average and standard deviation of the MMD scores across five independent iterations. The FMGAN's performance is significantly better in comparison to the other models, which have higher (worse) MMDs. It is noteworthy that the FMGAN outperforms the baselines even in this case, where we don't know *a priori* of any significant processing of the EI that needs to take place, as they are numerically meaningful values to begin with.

We also can visualize the embedding spaces learned by the generator to investigate the model. Shown in Figure 5.3a are the generator's embeddings colored by each of the heldout genes. As we can see, the generator found some of these more informative in learning an EI embedding than others. We can quantify this by building a regression model to try to predict the value of each gene given the embedding to determine the most valuable of the heldout genes. By this measure, PHGDH, PRCP, and GNAI1 are the most important genes. Analyzing the embeddings in this way is useful for determining which part of the EI space was most informative for generating the HI space, and we will continue to do this with more complex EI in later experiments.

PHATE coordinates as conditions for manifold-structured EI

Our next experiment uses an EI space that consists of a dimensionality-reduced manifold representation of the data. While the whole FMGAN will take raw ambient data and learn a manifold simultaneously with learning to generate from the manifold, here we first experiment with generating from an already-learned manifold. We do this to test whether the FMGAN has the ability to accurately generate from a condition manifold as distinct from its ability to learn the manifold of the conditions, as well.

We theorize that this approach would be beneficial over the previous held-out-genes experiment if the gene space exhibits manifold structure, which previous work has shown is often the case [7, 4, 119]. If so, and if the FMGAN is able to leverage the manifold structure, this processing will have made a geometric representation of the EI that corresponds to the HI, and thus the mapping is computationally simpler.

We run the embedding tool PHATE on the gene profiles to calculate two coordinates, which we then use as EI in our FMGAN [119]. Doing so preserves the manifold structure of the data, allowing for a meaningful transformation to the HI space. This process is depicted in Figure 5.2b. As usual, we separate cells into an 80%/20% training/testing split for evaluation purposes, after being subsampled to ten thousand points for computational feasibility with the dimensionality reduction method, and we report scores on the evaluation points.

As shown in Table 5.1, once again the FMGAN better models the target distribution, as measured by MMD between its predictions in the neighborhood of each point and the true values. The FMGAN's predictions once again have a lower MMD than any of the alternative models. This is notable as PHATE has already processed the conditions in thee sequencing domain to produce lower-dimensional, more compact representations. The fact that the full FMGAN still performs the best indicates that optimal conditions for generating may be different from optimal conditions for visualizing or some other task, implying it is still beneficial to utilize the FMGAN's condition-embedding network.

We observe the PHATE coordinates are much more effective than heldout genes as

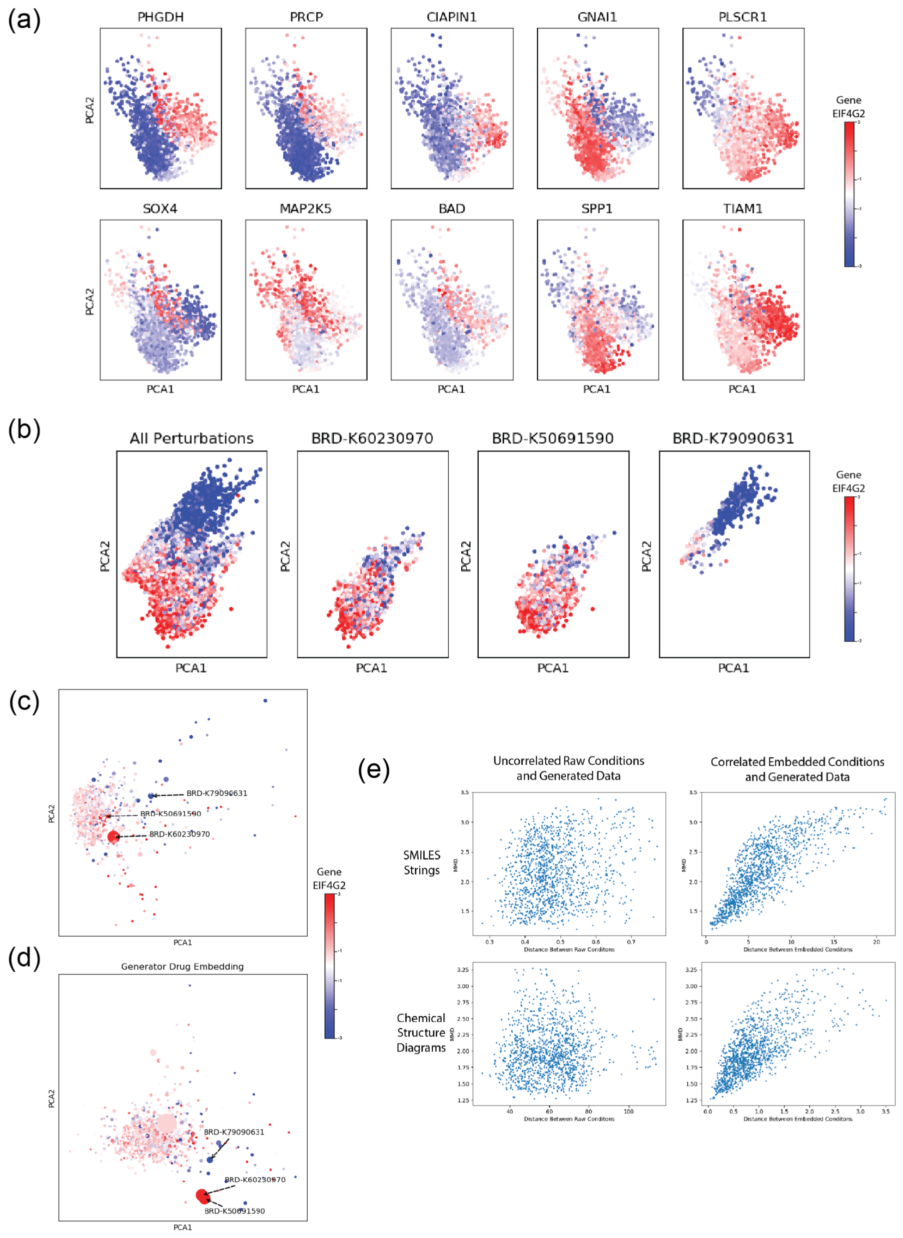


Figure 5.3: (a) Visualization of the embedding of *cells* in the held-out genes experiment, colored by each held-out gene. The network has inferred the structure of the space from these genes. (b) The raw data, colored by the expression of gene EIF4G2, separated into the three most abundant drugs: BRD-K60230970, BRD-K50691590, and BRD-K79090631. (c) The generator's embedding space of *drugs* from the SMILES strings experiment, with the same three drugs highlighted. The embedding shows that the drugs with similar distributions have been embedded into similar locations in the learned embedding space. (d) The same as in (c) but with the structure diagram experiment. (e) The conditions are more correlated to the generated data after they have been embedded.

conditions. This is in line with our hypothesis that manifold structure that is related to the data space is beneficial for generating. Despite containing loosely correlated information, the heldout genes are too few and noisy, while the PHATE coordinates actually calculate a full data manifold from more genes and with more shared information with the ambient data that needs to be modeled.

Predicting gene expression from drug chemical structure represented as SMILES

Next, we test the full pipeline of FMGAN by using SMILES strings as the EI (summarized in Figure 5.2c). This is a much more challenging test case than the previous ones, because in the previous cases each point in HI space had a distinct condition, and in the case of the PHATE coordinates, that condition was derived from the data it had to predict.

Most importantly, the conditions are in a raw data structure (one-hot vectors representing the SMILES strings token-by-token), rather than *a priori* existing in their final numerical form like heldout genes or scalar PHATE coordinates. This structure is not trivial to extract information from, as simple changes like one insertion shift the dimensions of every subsequent token, or the necessity of identifying recurring patterns that occur in different locations. These motivate the need for a convolutional condition-embedding network that looks for these kinds of structural forms and processes them prior to being given to the standard fully-connected network that generates the RNA sequencing data.

As in the previous experiment, we separate the data into an 80%/20% training/testing split for evaluation purposes, but this time split along the drugs since each condition gives rise to many points in the HI space. Table 5.1 indicates that the FMGAN has the lowest MMD of any of the models in this application. Perhaps unsurprisingly, there is also a larger gap between the full FMGAN and the FMGAN without an condition-embedding network than there was in the previous experiments. This provides verification that there is information about the RNA sequencing that can be leveraged in the drug metadata, but special architectures are necessary to process them and access it.

EI Space Analysis In addition to superior generative performance, we show the usefulness of having a generative model that learns embeddings by analyzing the learned EI SMILES strings embedding space.

In this learned EI space, there is one condition coordinate for each drug (while the HI consists of many perturbations from each drug). Shown in Figure 5.3b is the raw data colored by the value of gene EIF4G2. Then, all of the perturbations from each of three drugs are shown separately: BRD-K60230970, BRD-K50691590, and BRD-K79090631. As we can see, the first two are characterized by high expression of this gene and are quite similar to each other. The third, however, is quite distinct, in a separate space of the embedding, and is characterized by much lower expression of this gene.

We compare this to the embedding learned by the generator, which we show in Figure 5.3c. In this plot, each drug is one point, colored by the mean gene value of all perturbations for that drug and with a point whose size is scaled by the number of perturbations for that drug. We see that the first two drugs are in the central part of the space, and closer to each other than they are to BRD-K79090631. The drug BRD-K79090631 is off in a different part of the space, along with other drugs low in EIF4G2. This shows that the learned conditions from the generator have indeed identified information about the drugs and taken complex sequential representations and mapped them into a much simpler space.

Condition-to-generated data correlation The importance of the condition-embedding network can be seen by analyzing the distances between conditions and the distances between their corresponding generated data distributions. Since the generator must map from a condition to its generated distribution, the function it learns needs to be more complex (and will generalize poorly) if nearby conditions induce very different distributions and faraway conditions induce very similar distributions. The condition-embedding network is able to take conditions from their original space which may not be conveniently structured into a manifold space which is.

To test this, we perform the following experiment. For each pair of held-out drugs i and j, we calculate the distance between condition l_i and l_j, the distance between embedded conditions $E(l_i)$ and $E(l_j)$, and the MMD between the generated data from each condition $G(E(l_i))$ and $G(E(l_j))$.

As Table 5.2 shows, the correlation of distances between conditions and their MMDs

Correlation with Generated Data	Raw Conditions	Embedded Conditions
SMILES	0.071 +/- 0.002	0.709 +/- 0.016
Chemical Structure Images	0.220 +/- 0.008	0.800 +/- 0.012

Table 5.2: Pairwise correlation of the distance between two conditions and the MMD between their generated distribution. For distances in raw condition space, they are substantially uncorrelated. After being processed by the condition-embedding network, the distances in the embedded space are highly correlated with the difference in their generated data.

between the generated data distributions is just 0.071. This is unsurprising, as the SMILES strings have structure that violates notions of Euclidean distances with respect to the underlying similarity of the drugs. Drugs that only differ by one element can be very far from each other by Euclidean distance if the rest of the formula is shifted, but may produce almost identical data distributions if that inserted element has little effect on the overall structure. In the space that the condition-embedding network produces, the conditions form a manifold that is smooth with respect to the generated distributions. The correlation soared to 0.709, with distances in the embedded space meaningfully relating to how different their corresponding distributions are. These correlations can also be seen visually in the plotted data, as shown in Figure 5.3e. For further details and figures about this experiment, please refer to the supplement.

Predicting gene expression from drug structure diagrams

The final experiment we consider for the drug perturbation data is the formation of the condition space from an image representation of the chemical structure (Figure 5.3d). These images are downloaded from the PubChem PUG REST API. An example image for the drug BRD-U86686840 is shown in Figure 5.2d. They are given as input to a two-dimensional CNN designed for image processing, as points in the original h x w x c pixel space, with $h = w = 64$ and $c = 3$. While a CNN is used in both the SMILES string case and this one, the underlying data is in a fundamentally different structure.

Table 5.1 shows that the FMGAN also outperformed the baseline models in this case, as before. However, something more deeply revealing is also apparent from the scores in this experiment as compared to the previous experiments.

The FMGAN scored slightly *better* with these chemical structure diagrams as compared to the SMILES strings. The baseline models, on the other hand, all scored significantly worse with the drug structure diagram images. For processing these images, clearly the convolutional condition-embedding network is necessary to achieve good generation performance. The full FMGAN is able to leverage the information in image form just as well as in a long one-dimensional sequence form, while other models (including the FMGAN without the condition-embedding network) are not.

This illustrates the FMGAN's flexibility, as it performs comparably with such drastically different structures. That the chemical structure images perform slightly better is perhaps a sign that two-dimensional image convolutional networks are currently more effective at distilling this information than one-dimensional sequence convolutional networks, but the FMGAN's flexible framework allows it to keep improving with advances in deep learning architectures. Another possibility is that the structure diagrams have relevant information more easily separable from irrelevant information, making them an easier statistical task.

EI Space Analysis In Figure 5.3d, we show the learned embedding from the generator. We color the embedding by the same gene and highlight the same three drugs as in the previous experiment: BRD-K60230970, BRD-K50691590, and BRD-K79090631. As before the learned conditions have taken a space where it is hard to characterize the information it contains (raw images in pixel space) and mapped them to a simpler space with numerically meaningful points. This can be seen by noting that the two drugs with similar distributions in the raw data (BRD-K60230970 and BRD-K50691590) have been mapped to nearly identical conditions, while they are separate from the drug with a very different distribution (BRD-K79090631). In fact, this goes towards an explanation of the improvement in performance over the SMILES string model, as the condition-embedding network has placed the drugs with similar distributions closer to each other in conditions, making the generator's job easier.

Condition-to-generated data correlation As in the SMILES string experiment, we evaluate the correlation of the distance between conditions and the distance between their data distributions. And just as in the previous case, the condition space is originally structured in such a way that distances are not meaningfully related to the effect that condition produces (images can be far away simply because they have the same diagram but it is rotated or shifted). The condition-embedding network produces a manifold that increases the correlation of these distances from 0.220 to 0.800 (Table 5.2).

We note that we can evaluate the two different representations of the chemical structure and compare them by looking at the scores in these experiments. The embedding with chemical structure images increases the correlation with MMD between ambient data distributions by 12% over the embedding with SMILES representations. In this latter case, the embedding network was slightly better able to organize the embedding space to align with the ambient data distributions. We caution careful interpretation of this, though: this improvement could stem from the images being a better representation of information, image-based embedding architectures being more powerful than sequence-based embedding architectures, or some combination of both. But in either case, the learned embedding space is vastly better organized than the original representation of the conditions.

5.3.4 Predicting flow cytometry data on COVID-19 patients

We demonstrate the versatility of our proposed method by experimenting on data in a very different context from the drug perturbations of the previous section. Here we work on clinical data that is derived from measurements taken early in the clinical stay and predict measurements that are taken later in the stay. Specifically, in this section we present an experiment that learns a mapping between clinical measurements and FACS measurements from COVID-19 patients [60]. The clinical measurements are taken from the first 24 hours in the ICU, with a patient's record being the most extreme value taken during that period when more than one record is taken. To test the ability of FMGAN to make practical, and actionable predictions we learn to generate the first flow cytometry measurement, taken from anywhere from the first week to the eleventh week of the stay. Thus, we model *future* flow cytometry with *present* clinical data.

The conditions we use for the FMGAN are PHATE coordinates of the present available clinical variables. In the PHATE embedding each patient is represented by a vector of

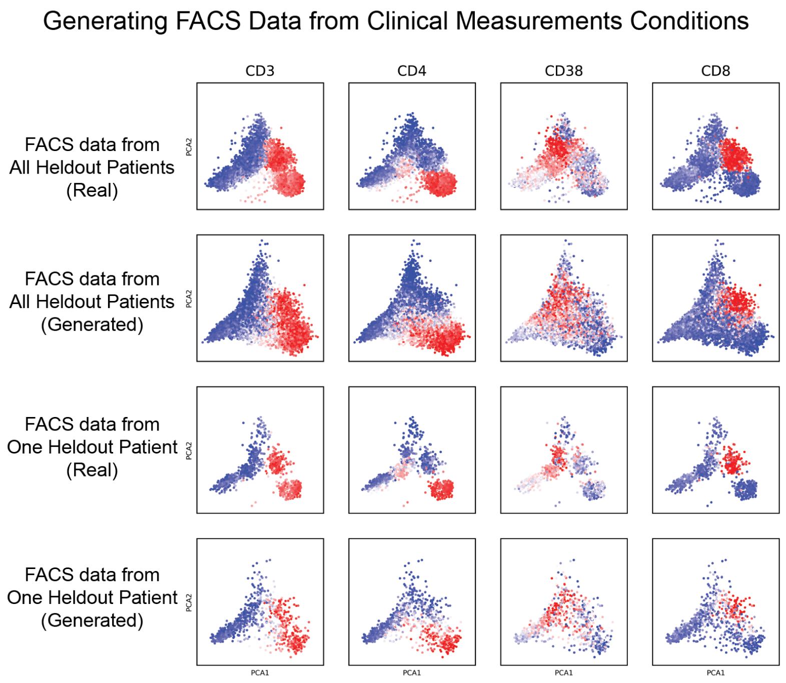


Figure 5.4: FACS data generated from clinical measurements in the COVID-19 data. Top row: for all 26 held-out patients in the first fold, the real FACS measurements. Second row: for all 26 held-out patients, generated FACS measurements from the FMGAN. Third row: a single patient's real FACS measurements. Bottom row: a single patient's generated FACS measurements.

variables, listed in the supplement. For each of 129 patients, we also have matched FACS measurements on 14 proteins obtained from each patient, which are also listed in the supplement. While the clinical measurements are relatively easy and inexpensive to obtain, FACS samples are comparatively expensive and time-consuming to obtain. Thus, we wish to learn a model that can accurately generate FACS data from a patient's clinical measurements alone. We note that while these conditions are in tabular form and processed with PHATE, we still use the full FMGAN to learn embeddings, as this achieved the best generative performance even for conditions of this form.

To evaluate the ability of the FMGAN to perform this generation, and to handle the relatively few number of distinct patients, we perform K-fold cross-evaluation, training each time on 80% of the patients (103) and withholding 20% of the patients (26) for evaluation. We train to generate a distribution of FACS measurements from each single condition corresponding to a patient's clinical measurements. In Figure 5.4, we see the resulting data from all heldout patients in the top row from the first fold. In the second row, we see the corresponding FMGAN generated data. Remarkably, the FMGAN learned to accurately model the true distribution of FACS data even for the never-before-seen patients. Distinct populations of cells are visible: CD3+ T cell populations including both CD4+ (T helper cells) and CD8+ (Cytotoxic T cells), as well as a CD38+ population. With each protein marker, the FMGAN accurately models the underlying data distribution.

In the bottom two rows of Figure 5.4, we see the FMGAN model the distribution from a single patient accurately, as well. This per-patient generation forms the basis for our quantification of the model's accuracy. We utilize the same baselines as in the previous section. For each fold (reported separately), and for each patient within that fold, we measure the distribution distance between the predicted distribution and the true distribution of FACS data (scored by MMD, as before). These numbers are reported in Table 5.3. That evaluation shows the FMGAN is able to produce distributions very close to the true underlying distribution for each patient, while the baseline models do not. As each distribution is complex with many different cell populations with varying proportions, it is not surprising that the more richly expressive FMGAN is best able to model the true data.

We note that with the FMGAN, we are able to predict the FACS measurements on never-before-seen patients, based on their clinical measurement alone. However, this relied upon the patients in the training set being representative of the patients in the held-out set. In practical applications, this means that the population of patients would need to be chosen carefully and diversely for the predictions to be meaningful for future patients.

5.4 Experimental Procedures

5.4.1 Conditional Generative Adversarial Networks

In a Generative Adversarial Network (GAN), samples from the generator G can be obtained by taking samples from $z \sim Z$ and then performing the forward pass with the learned weights of the network. But while the values of z control which points G generates, we do not know how to ask for specific types of points from G (more discussion of the original, unconditional GAN is in the Supplementary Information).

The lack of this functionality motivated the need for the conditional GAN (cGAN) framework [146, 101]. The cGAN augments the standard GAN by introducing label information for each point. These labels stratify the total population of points into different groups. The generator is provided a given label in addition to the random noise as input, and the discriminator is provided with not only real and generated points, but also the labels for each point. As a result, the generator not only learns to generate realistic data, but it also learns to generate realistic data *for a given label*.

After training, the labels, whose meaning is known to us, can be provided to the generator to generate points of a particular type on demand. Because G is provided both a label and a random sample from Z, the cGAN is able to model not just a mapping from a label to a single point, but instead a mapping from a label to an entire distribution.

Expressing the cGAN formula mathematically yields a similar equation as to the original GAN, except with the modeled data distributions being marginal distributions conditioned on the label l of each point:

$$\min_G \max_D \mathbb{E}_{x_l \sim P(x|l)} log(D(x|l)) + \mathbb{E}_{z \sim P_z} log(1 - D(G(z|l)))$$

Learning a generative model conditioned on the labels allows information sharing across labels, another advantage of the cGAN framework. Since the generator G must share weights across labels, the signal for any particular label l_i is blended with the signal from all other labels $l_j, j \neq i$, allowing for learning without massive amounts of data for each label.

5.4.2 Chemical Structure and SMILES Strings

Conditional GANs are a powerful construction for guided generation, but require some known label space to be used. While the label space must be relevant to the measured data space for an informative model to be learned, the relationship need not be simple and can be noisy. When the data space is gene expression after a drug perturbation, as in our application here, one relevant source of labels is metadata about the structure of the drug used for the perturbation. We consider two ways of representing this structure for our label space: a one-dimensional sequence of letters called a Simplified Molecular-Input Line-Entry System (SMILES) string, and a two-dimensional image called a structure diagram.

SMILES strings A SMILES string encodes the chemical structure of a drug in a variable-length set of standard letters and symbols. Each character in the string represents an element of the chemical's physical formation, for example an atom, a bond, or a ring. For example, the common molecule glucose has the following structure:

OC[C@@H](O1)[C@@H](O)[C@H](O)[C@@H](O)[C@@H](O)1

The letters indicate elements oxygen, carbon, and hydrogen, with @ denoting steriochemical configuration, and brackets and parentheses representing bonds and branches, respectively. Clearly, while providing rich information about the drug, this representation does not immediately lend itself to use as a condition. In order to distill these variable-length sequences into a fixed-size representation where similar structures have similar representations, we use a sequence-encoding neural network to embed each structure into a latent space.

Structure diagram An alternate way of representing chemical structure, more intelligible for a human observer than SMILES strings, is a structure diagram. These have letters representing elements as in the SMILES strings, but also are distinguished by colors, while different types of bonds are indicated with simple lines. These images are downloaded from the PubChem PUG REST API. While specifying how to get information about the structure out of this image explicitly would be impossible (in terms of RGB pixels), a neural network can learn how to process these images itself in order to accomplish its training objective, all through a completely differentiable optimization with stochastic gradient descent.

5.4.3 FMGAN Architecture

We describe the architecture for the FMGAN in this section. In the SMILES strings experiment, to obtain a fixed-length D_E-dimensional vector for each string, we represent each input as a sequence of length N_{seq} vectors, with N_{seq} being the longest SMILES string in the database. Each element in the sequence is a vector representing the character in that position of the sequence (with a null token padding the end of any sequence shorter than N_{seq}). As is standard in language processing, we learn character-level embeddings simultaneously with the sequence-level processing. Let V be the vocabulary, or set of all characters. The character-level embeddings are rows of a $|V| \times D_{char}$ matrix W, where $|V|$ is the number of characters in the vocabulary and D_{char} is a hyperparameter, the size of the character embedding. Each input is then represented as a sequence where the i^{th} element is the row of W corresponding to the i^{th} character in the SMILES string.

The size of the vocabulary (number of characters including start, end, and null tokens) is 43. We chose the size of the character-level embedding to be 100. The condition-embedding network E consists of two convolutional layers with 64 and 32 filters, respectively, each with a kernel-size of 40 and stride-length 2 with batch normalization and a leaky ReLU activation applied to the output. These convolutional layers are followed by four fully-connected layers which gradually reduce the dimensionality of the data with $400, 200, 100$, and 50 filters, respectively. All layers except the last one have batch normalization and leaky ReLU activations. The generator and discriminator have the same architecture as the previous experiment.

This input representation is then passed through E, a convolutional neural network (CNN), which produces the sequence embeddings. E performs one-dimensional convolutions over each sequence followed by fully-connected layers, eventually outputting a single D_E-dimensional vector for each SMILES string. We let these embeddings form the condition space for the next stage in FMGAN, the conditional GAN.

For the structure diagram experiment, we start with images that are points in h x w x c space, with $h = w = 64$ and $c = 3$. They are then processed with a CNN. The CNN consists of four convolutional layers with stride 2, kernel size 3, and filters of 32, 64, 128, and 256, respectively. Batch normalization and a ReLU activation was used for each layer. Finally, after the convolutions, one fully connected layer maps the flattened output to a 100-dimensional point, representing the embedding learned for the particular diagram.

For both experiments, the generator structure, after the drugs are processed into conditions, is the same. Let c_i be the condition for drug i formed by the condition-embedding network. Let x_i be the D_x-dimensional corresponding gene expression profile from a perturbation experiment performed with drug i. We build a GAN that trains a generator G to model the underlying data distribution conditioned upon the structure $p_{data}(x|c)$. G

takes as input both a sample from a noise distribution (we choose an isotropic Gaussian) $z \sim Z$, and a condition c_i. G maps these inputs to a D_x-dimensional point. Then, the discriminator D takes both a D_x-dimensional point and a condition c and outputs a single scalar representing whether it thinks the point was generated by G or was a sample from p_{data}. These networks then train in the standard alternating gradient descent paradigm of GANs previously detailed.

For specific hyperparameter choices and data dimensionality details, we refer to the Supplementary Information.

We note a few additional points about the FMGAN framework. First, since everything in the network including the character-level embeddings, the condition-embedding network E, and the GAN are all expressed differentiably, the whole pipeline can be trained at once in an end-to-end manner. Thus, the character-level embeddings and the convolutional weights can be optimized for producing SMILES strings embeddings *useful for this specific task and context.* This is a powerful consequence, as defining what makes a good static embedding of a high-dimensional sequence may be ambiguous without reference to a particular task.

5.5 Discussion

The FMGAN model allows us to predict hard-to-obtain information for samples where we only directly measure easy-to-obtain information. We demonstrate that the FMGAN can accurately model never-before-seen samples in these contexts. In the drug discovery context, this allows the potential impact of saving on expense and time by not performing as many physical experiments and instead modeling their results. In the clinical context, this allows for the modeling of patient data sooner, with more time to take positive interventions.

Furthermore, the flexible framework of the cGAN we develop for the FMGAN allows for EI that requires advanced processing to be used as the conditional input. We demonstrate this on images and long one-dimensional sequences, but this can extended to other difficult-to-represent data. For example, in the clinical setting, the advances in natural language processing achieved by deep neural networks could be utilized to process doctor's notes as raw text and then incorporated into the model.

COVID-19 FACS K-fold validation	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
FMGAN	**0.039** ± 0.02	**0.024** ± 0.01	**0.022** ± 0.01	**0.041** ± 0.02	**0.031** ± 0.01
Linear	0.851 ± 0.03	0.915 ± 0.01	0.701 ± 0.02	0.758 ± 0.01	0.881 ± 0.01
VAE	0.623 ± 0.02	0.499 ± 0.01	0.682 ± 0.01	0.521 ± 0.01	0.588 ± 0.02

Table 5.3: MMD distance between real and generated data (lower is better) on the COVID-19 data, with mean and standard deviation across the 26 held-out patients in each fold in the cross-evaluation. The FMGAN outperforms the baselines significantly in all cases.

We demonstrate that the FMGAN is able to leverage structure in the condition space in both manifold form (from the PHATE coordinates) and discrete form (from chemical structure strings). While seemingly similar, these are very different from an information theoretical point of view. In the manifold setting, differences in input can create differences in output in a smooth way, but in the discrete setting, one small change in an individual feature may have a large effect on the output while another small change in a different feature has no effect on the output at all. For example, in a chemical structure string, modifications to some locations will not change the function at all, while in other locations a single change will determine function.

While we demonstrate that the FMGAN can be usefully applied to generative problems in a wide variety of modalities, and, as we show, even in the presence of high amounts of stochasticity.

5.6 Supplemental Experimental Procedures

5.6.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a deep learning framework for learning a generative model of a data distribution. In recent years, they have gained significant popularity by achieving state-of-the-art performance on applications to images, language, sequences, and many other data modalities [23, 177, 5, 56, 4]. GANs differ from other types of models by not using explicit likelihood measures nor relying on having a meaningful distance measure between points. Instead, they teach a generator neural network G with a second discriminator network D using the following equation:

$$\min_G \max_D \mathbb{E}_{x\sim P_x}[log(D(x))] + \mathbb{E}_{z\sim P_z}[log(1 - D(G(z)))]$$

where x is the training data, z is a noise distribution that provides stochasticity to the generator and is chosen to be easy to sample from (typically an isotropic Gaussian).

5.6.2 Conditional Generative Adversarial Networks

Conditional Generative Adversarial Networks (cGANs) originated from the desire for having greater control over generation from GANs. In the case where external information, such as class labels, are available, we would like to be able to generate a random point from a specific class. The methods devised to achieve this involve providing a random label to the generator during training and then providing this label and the generated image to the discriminator. The discriminator also receives real images and their labels, allowing it to learn their joint distribution.

Once the model has been trained in this way, control over generation can be used to generate a point from a particular class by feeding the desired class into the generator. This process especially benefits from having fine-grained, continuous conditions like we have, as this gives even more precise control over generation.

5.6.3 Optimization

The networks G and D take turns optimizing their objectives through alternating gradient descent. Throughout training, the discriminator provides gradient information to the generator guiding it to better quality generation. This powerful framework provides the ability to model arbitrarily complex distributions without making any explicit parametric or limiting assumptions about their shape.

Theoretical analysis of GANs have shown their ability to converge to an optimal point where the generated distribution is indistinguishable from the true distribution [84, 64, 113]. The ability to converge to this optimal generative model without specifying a distribution distance is especially helpful in our applications, where the points lie in high dimensions and the curse of dimensionality makes distances problematic [71].

Manifold learning A useful assumption in representation learning is that high biomedical dimensional data originates from an intrinsic low dimensional manifold that is mapped via nonlinear functions to observable high dimensional measurements; this is commonly referred to as the manifold assumption. In particular, we believe that since biological entities like patients, cells lie in lower dimensional spaces because of informational redundancy and coordination between measured features (coordinating genes, or coordinated combinations of residues on molecules). Further, we believe that these low dimensional spaces form smoothly varying patches because of natural heterogeneity between entities. The fact that the manifold model is successful in modeling biological entities has been shown in literature numerous times [117] and has lead to successful methods data denoising [164], clustering [91], visualization [118], and progression analysis [57].

Formally, let $\mathcal{M}^d$ be a hidden d dimensional manifold that is only observable via a collection of $n \gg d$ nonlinear functions $f_1, \ldots, f_n : \mathcal{M}^d \to \mathbb{R}$ that enable its immersion in a high dimensional ambient space as $F(\mathcal{M}^d) = \{\mathbf{f}(z) = (f_1(z), \ldots, f_n(z))^T : z \in \mathcal{M}^d\} \subseteq \mathbb{R}^n$ from which data is collected. Conversely, given a dataset $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ of high dimensional observations, manifold learning methods assume data points originate from a sampling $Z = \{z_i\}_{i=1}^N \in \mathcal{M}^d$ of the underlying manifold via $x_i = \mathbf{f}(z_i)$, $i = 1, \ldots, n$, and aim to learn a low dimensional intrinsic representation that approximates the manifold geometry of $\mathcal{M}^d$.

To learn a manifold geometry from collected data, we use the popular diffusion maps construction of [37] that uses diffusion coordinates to provide a natural global coordinate system derived from eigenfunctions of the heat kernel, or equivalently the Laplace-Beltrami operator, over manifold geometries. This construction starts by considering local similarities defined via a kernel $\mathcal{K}(x, y)$, $x, y \in F(\mathcal{M}^d)$, that captures local neighborhoods in the data. We note that a popular choice for $\mathcal{K}$ is the Gaussian kernel $\exp(-\|x - y\|^2/\sigma)$, where $\sigma > 0$ is interpreted as a user-configurable neighborhood size. However, such neighborhoods encode sampling density information together with local geometric information. To construct a diffusion geometry that is robust to sampling density variations we use an anisotropic kernel

$$\mathcal{K}(x,y) = \frac{\mathcal{G}(x,y)}{\|\mathcal{G}(x,\cdot)\|_1^\alpha \|\mathcal{G}(y,\cdot)\|_1^\alpha}, \quad \mathcal{G}(x,y) = e^{-\frac{\|x-y\|^2}{\sigma}}$$

as proposed in [37], where $0 \leq \alpha \leq 1$ controls the separation of geometry from density, with $\alpha = 0$ yielding the classic Gaussian kernel, and $\alpha = 1$ completely removing density and providing a geometric equivalent to uniform sampling of the underlying manifold. Next, the similarities encoded by $\mathcal{K}$ are normalized to define transition probabilities $p(x,y) = \frac{\mathcal{K}(x,y)}{\|\mathcal{K}(x,\cdot)\|_1}$ that are organized in an $N \times N$ row stochastic matrix

$$\mathbf{P}_{ij} = p(x_i, x_j) \tag{5.1}$$

that describes a Markovian diffusion process over the intrinsic geometry of the data. Finally, a diffusion map [37] is defined by taking the eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ and (corresponding) eigenvectors $\{\phi_j\}_{j=1}^N$ of $\mathbf{P}$, and mapping each data point $x_i \in X$ to an N dimensional vector $\Phi_t(x_i) = [\lambda_1^t \phi_1(x_i), \ldots, \lambda_N^t \phi_N(x_i)]^T$, where t represents a diffusion-time (i.e., number of transitions considered in the diffusion process). In general, as t increases, most of the eigenvalues λ_j^t, $j = 1, \ldots, N$, become negligible, and thus truncated diffusion map coordinates can be used for dimensionality reduction [37].

PHATE for structure-preserving visualization of Data Several dimensionality reduction methods that render data into 2-D visuals like PCA and tSNE. [162] and UMAP [111] exist. However, they often cannot handle the degree of noise in biomedical data. More importantly, most of these methods are not constructed to preserve the global manifold structure of the data. PCA cannot denoise in non-linear dimensions, tSNE/UMAP effectively only constrains for near neighbor preservation—losing global structure. This motivated us to develop a method of dimensionality reduction that retains both local and global structure, and denoises data [118].

PHATE also builds upon the diffusion-based manifold learning framework described above, and involves the creation of a diffused Markov transition matrix from cellular data, as in MAGIC, $\mathbf{P}^t$ (Equation 5.1). PHATE collects all of the information in the diffusion operator into two dimensions such that global and local distances are retained. To achieve this, PHATE considers the ith row of $\mathbf{P}$ as the representation of the ith datapoint in terms of its t-step diffusion probabilities to *all* other datapoints. PHATE then preserves a novel distance between two datapoints, based on this representation called *potential distance (pdist)*. Potential distance is an M-divergence between the distribution in row i, $\mathbf{P}^t_{i,.}$ and the distribution in row j, $\mathbf{P}^t_{j,.}$. These are indeed distributions as $\mathbf{P}^t$ is Markovian: $pdist(i,j) = \sqrt{\sum_k (\log(P^t(i,k) - P^t(j,k))^2}$.

The log scaling inherent in potential distance effectively acts as a damping factor which makes faraway points similarly equal to nearby points in terms of diffusion probability. This gives PHATE the ability to maintain global context. These potential distances are embedded with metric MDS as a final step to derive a data visualization. We have shown that PHATE outperforms tSNE [162], UMAP [111], force directed layout and 12 other methods on preservation of manifold affinity, and adjusted rand index on clustered datasets, in a total of 1200 comparisons on synthetic and real datasets. In [118] we also showcased the ability of PHATE to reason about differentiation systems and differentiation trajectories in human embryonic cell development.

5.6.4 Maximum Mean Discrepancy

To evaluate the accuracy of the predicted distribution with respect to the true distribution for a given condition, we utilize maximum mean discrepancy (MMD) [93]. The MMD is a distribution distance based on a kernel applied to pairwise distances of each distribution. Specifically, MMD is calculated as:

$$MMD(X,Y) = \frac{1}{n} \sum_{i \neq i'} k(x_i, x_{i'}) + \frac{1}{m} \sum_{i \neq i'} k(y_i, y_{i'}) - \frac{2}{mn} \sum_{i \neq j} k(x_i, y_j)$$

for finite samples from distributions $X = \{x_1, ...x_m\}$ and $Y = \{y_1, ...y_n\}$ and kernel function k. Two distributions have zero MMD if and only if they are equal. MMD has been used successfully in biological systems in the past, particularly in detecting whether two systems were different in distribution [21].

5.6.5 Raw and Embedded Condition Space Correlation Experiment

The correlation experiment was conducted as follows. For the drugs held-out to be used as evaluation conditions, samples from the generator were calculated, along with the condition's embedding coordinates that went into the generator. A fixed noise sample was used throughout. Then, for each pair of conditions i and j, we calculate distance between condition l_i and l_j, distance between embedded conditions $E(l_i)$ and $E(l_j)$, and the MMD between the generated data from each condition $G(E(l_i))$ and $G(E(l_j))$. We plot each distance against the MMD in Figure 5.3e for both the SMILES string experiment (top row) and the chemical structure diagram experiment (bottom row).

5.6.6 COVID-19 Clinical data

The cohort of patients included only those who were hospitalized at any of 6 hospitals in the Yale-New Haven Health System (Bridgeport, Greenwich, St. Raphael's Campus, Westerley, Lawrence and Memorial, York Street Campus) during the period between March 1st, 2020 and June 1st, 2020 with a positive COVID test (nasopharyngeal source) between admission and discharge. Only the first encounter was included in the dataset for patients with multiple encounters during the time period of observation. Patients with a positive test prior to hospital admission but not tested during admission or tested negative during admission were not included in the cohort. Data for these patients was then extracted from the electronic health record (Epic, Verona, WI) and included data domains of demographics (e.g. age and sex), medical history (e.g. history of diabetes), laboratory samples (e.g. white blood cell count), as well as vital signs (e.g. blood pressure measurement). Pre-defined outcomes included in-hospital mortality, transfer to the intensive care unit (ICU), as well as requirement for invasive ventilation. In-hospital mortality was measured as patients being discharged from the hospital with a deceased status. ICU care was measured through location data for patients and was manually validated through chart review. Ventilation status was measured through procedure orders placed during the patient's hospitalization and were validated through chart review.

Time-varying data, specifically vital signs as well as laboratory studies, were extracted at all timepoints of measurement during a patient's admission.

Features were selected from a predictive model developed to predict early hospital respiratory decompensation among patients with Covid-19 and augmented with treatment received. There were a total of 19 clinical, laboratory, and treatment variables extracted: systolic blood pressure, respiratory rate, oxygen saturation, blood urea nitrogen, creatinine,chloride, glucose, white blood cell count, alanine aminotransferase, aspartate aminotransferase, high-sensitivity C-reactive protein, ferritin, procalcitonin, age, gender, and treatment with hydroxychloroquine, steroid, antibiotic, or tocilizumab. Only complete cases, or patients with recorded values for all 19 variables in the first 24 hours, were included in the final dataset. The FACS markers used were: CCR7, CD3, CD4, CD8, CD25, CD38, CD45RA, CD127, CXCR5, FSC, HLA-DR, PD1, SSC, TIM3.

As preprocessing, the most abnormal value in the first 24 hours was selected for the clinical and laboratory variables according to the methodology described in a previous electronic health record-based study. The categorical variables for treatment were coded as binary (1 for received, 0 for not recorded).

Part II

Multi-sample alignment with Autoencoders

Chapter 6

SAUCIE

6.1 Introduction

Biomedical researchers are generating high-throughput, high-dimensional single-cell data at a staggering rate. As costs of data generation decrease, experimental design is moving towards measurement of many different single-cell samples in the same dataset. These samples can correspond to different patients, conditions, or treatments. While scalability of methods to datasets of these sizes is a challenge on its own, dealing with large-scale experimental design presents a whole new set of problems, including batch effects and sample comparison issues. Currently, there are no computational tools that can both handle large amounts of data in a scalable manner (many cells) and at the same time deal with many samples (many patients or conditions). Moreover, data analysis currently involves the use of different tools that each operate on their own data representation, not guaranteeing a synchronized analysis pipeline. For instance, data visualization methods can be disjoint and mismatched with the clustering method. For this purpose, we present SAUCIE, a deep neural network that leverages the high degree of parallelization and scalability offered by neural networks, as well as the deep representation of data that can be learned by them to perform many single-cell data analysis tasks, all on a unified representation.

A well-known limitation of neural networks is their interpretability. Our key contributions here are newly formulated regularizations (penalties) that render features learned in hidden layers of the neural network interpretable. When large multi-patient datasets are fed into SAUCIE, the various hidden layers contain denoised and batch-corrected data, a low dimensional visualization, unsupervised clustering, as well as other information that can be used to explore the data. We show this capability by analyzing a newly generated 180-sample dataset consisting of T cells from dengue patients in India, measured with mass cytometry. We show that SAUCIE, for the first time, can batch correct and process this 11-million cell data to identify cluster-based signatures of acute dengue infection and create a patient manifold, stratifying immune response to dengue on the basis of single-cell measurements.

Processing data of high dimensionality and scale is inherently difficult, especially considering the degree of noise, batch effects, artifacts, sparsity and heterogeneity in the data [154, 171]. However, this effect becomes exacerbated as one tries to compare between samples, which themselves contain noisy heterogeneous compositions of cellular populations. Deep learning offers promise as a technique for handling the size and dimensionality of modern biological datasets. However, deep learning has been underutilized for unsupervised exploratory tasks. In this paper, we use a regularized autoencoder, which is a neural network that learns to recreate its own input via a low-dimensional bottleneck layer that learns representations of the data and enables a denoised reconstruction of the input from them [169, 152, 153, 30, 31], as a basis for deep learning framework with layers constrained (via architectural choices and regularization), so they can be used to extract task-oriented features of the data. Since autoencoders learn their own features, they can reveal structure in the data without defining or explicitly learning a similarity or distance metric in the original data space as other dimensionality reduction methods do (for instance, PCA uses covariance and diffusion maps [37] utilize affinities based on a kernel choice). We use this approach to construct SAUCIE, a Sparse Autoencoder for Unsupervised Clustering, Imputation, and Embedding, which is aimed to enable exploratory tasks via its design choices. SAUCIE is a multilayered deep neural network, whose input layer is fed single-cell measurements, such as mass cytometry or single-cell RNA sequencing, of an individual cell. We see that the output or reconstruction layer of SAUCIE gives similarly denoised and imputed data as the manifold denoising method MAGIC [163], effectively learning the manifold of the data in a similar way to data diffusion [166] methods. Manifold learning methods are traditionally difficult to scale due to the computational complexity of kernel computation and eigendecomposition operations. Deep learning comes to the rescue here by being amenable to GPU speedup and parallelization of matrix operations. SAUCIE's regularizations, balanced against reconstruction accuracy, reveal different representations of the data: for visualization, batch correction, clustering, and denoising.

We apply SAUCIE to a twenty-million cell mass cytometry dataset with 180 samples from forty subjects in a study of the dengue flavivirus [187]. SAUCIE is able to batch correct 180 samples and cluster them in such a way that subpopulation proportions become comparable prima facia. This obviates the need for approaches such as first clustering samples separately and then performing "meta-clustering", or other methods that cannot operate uniformly on combined data of this size (the problems of which are illustrated in Figure S10). SAUCIE results show that acute subjects are characterized by enrichment in distinct subpopulations of CD4-CD8-$\gamma\delta$T cells and cells involved in Type I interferon signaling. When subjects are measured in convalescence, there is an increase in CD4+Foxp3+ T reg cells. Thus, SAUCIE provides a unified representation of data where different aspects or features are emphasized in different layers, forming a one-step data analysis pipeline. This unified analysis uncovers a cell-space manifold as well as a sample-space manifold, thus enabling a multilevel analysis of complex experimental design where the samples are stratified on the basis of their cell-level features. We additionally evaluate SAUCIE extensively on all of its designed tasks using ten public single-cell datasets.

6.2 Results

6.2.1 The SAUCIE Architecture and Layer Regularizations

To enable unsupervised learning in a scalable manner, we base our method on the autoencoder. Autoencoders learn to recreate their input at the output layer, but via a low-dimensional informational bottleneck layers which are forced to learn meaningful structure-preserving representations of the data. However, a key challenge is to extract meaning from this representation. Specifically, we seek representations in hidden layers that are useful for performing the various analysis tasks associated with single cell data. Here, we introduce several design decisions and novel regularizations to our autoencoder architecture (Figure 6.1)

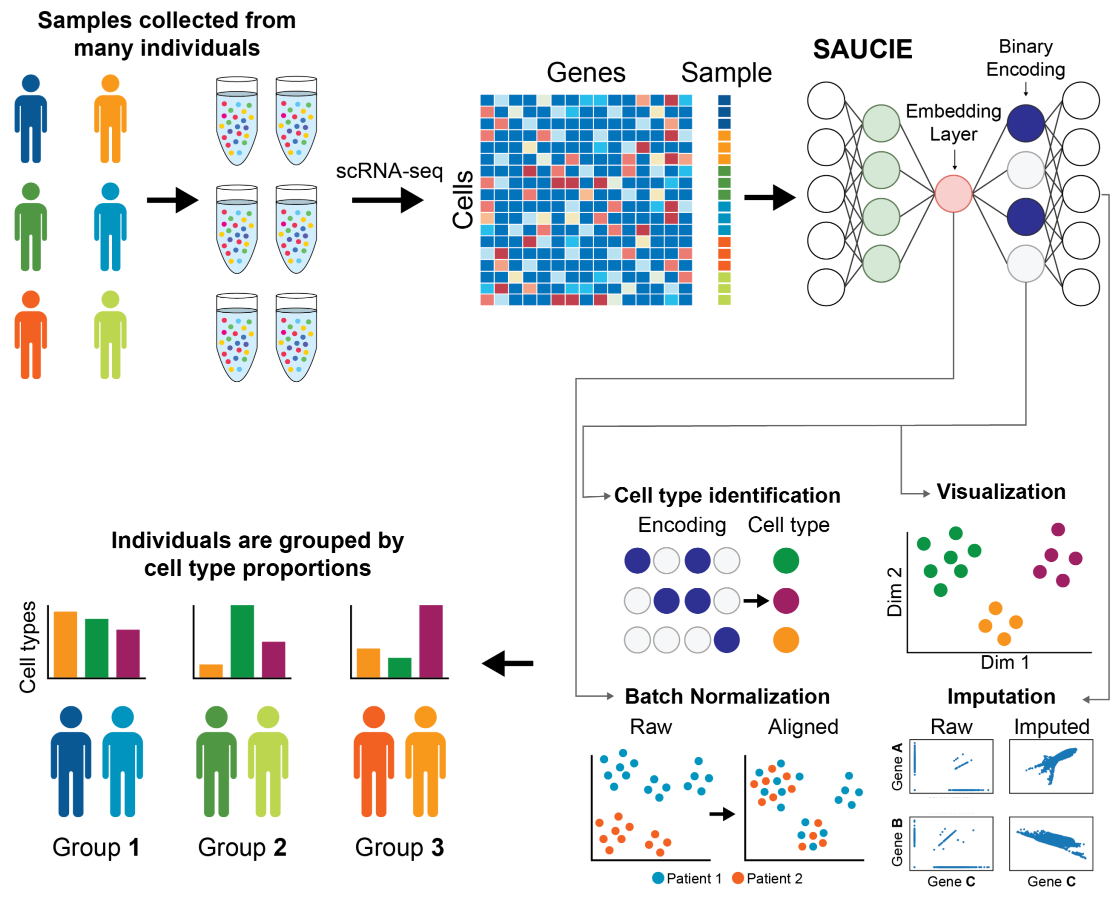


Figure 6.1: **The pipeline for analyzing single-cell data in large cohorts with SAUCIE.** Many individual patients are separately measured with a single-cell technology such as CyTOF or scRNA-seq, producing distinct datasets for each patient. SAUCIE performs imputation and denoising, batch effect removal, clustering, and visualization on the entire cohort with a unified model and is able to provide interpretable, quantifiable metrics on each subject or group of subjects.

in order to constrain the learned representations for four key tasks:

1. visualization and dimensionality reduction,
2. batch correction,
3. clustering, and
4. denoising and imputation.

For each task, dedicated design decisions are used to produce the desirable result.

Clustering: First, to cluster the data, we introduce the *information dimension* regularization that encourages activations of the neurons in a hidden layer of the network to be binarizable. The idea is that if we can obtain a "digital" binary encoding, then we can easily turn these codes into clusters. As Figure 6.2A shows, the network without regularizations tends to store its information in a distributed, or "analog" way. With the ID regularization the activations are all near 0 or 1, i.e., binary or "digital", and thus amenable to clustering by simple thresholding-based binarization. As seen in Figure 6.3A, this leads to a clustering of the cells that effectively represents the data space. Thus, the ID regularization achieves an analog-to-digital conversion that enables interpretation of the representation as data groups or clusters corresponding to each binary code. A previous work in the same vein, Binary Connect, has shown the promise in encouraging networks to learn in ways that are easy to binarize. That work differs from SAUCIE though, in that they learn binary weights rather than binary activations, along with the goal being to improve computational efficiency rather than achieve a clustering of the data [38]. Further work has considered binarizing activations, as well, but do so with exact binarization (as opposed to our activations which are still continuous but are encouraged to be near binary) and do so with the aim of compressing the network into a smaller memory footprint (rather than our clustering) [155, 39].

Batch Correction: Batch effects are generally systematic differences found in biological data measured under different experimental runs, largely due to ambient conditions such as temperature, machine calibration or day-to-day variation in measurement efficiency. Thus, measurements even from very similar systems, such as blood cells of the same patient, appear to have a shift or difference between two different experimental runs. To solve this problem, we introduce a maximal mean discrepancy (MMD) correction that penalizes differences between the probability distributions of internal activations of samples. Previous work has attempted batch correction by minimizing MMD. However, those models assume that batch effects are minor and simple shifts close to the identity function, which is often not the case [141]. Moreover, minimizing MMD alone only removes any and all differences between batches. In contrast, the additional autoencoder reconstruction penalty in SAUCIE forces it to preserve the original structure in each batch, balancing the goals of, on one hand, making the two batches alike while on the other hand not changing them. We note that this notion of a biological batch (data measured or run together) is distinct from the mini-batches used in stochastic gradient descent to train neural networks and the two should not be confused. The term batch is exclusively used to describe biological batches and when training with stochastic gradient descent the term mini-batches is used.

Figure 6.5 shows that analyzing data before batch correction can lead to misleading results, as artificial variation from batch effects can drown out the relevant variation within the biology that we are interested in. Penalizing MMD directly on the input space would be a flawed way of addressing batch effects because it would require making the assumption of (and thus being sensitive to the choice of) meaningful distance and similarity measures on the input points. Since the data is noisy and possibly sparse, by instead penalizing MMD on an internal layer of the network, we can correct complex, highly nonlinear batch effects by aligning points on a data manifold represented in these layers.

Imputation and denoising Next, we leverage the fact that an autoencoder does not reconstruct its input exactly, but instead must learn a lower dimensional representation of the data, and decode this representation for data reconstruction. This means the reconstructions are denoised versions of the input and are thus naturally solutions to the dropout and other noise afflicting much real-world data, especially single cell RNA-sequencing data. The gene-gene relationships plotted in Figure 6.3C illustrate the ability of SAUCIE to recover the meaningful relationship between genes despite the noise in the data. Thus, downstream activities like differential gene expression are now enhanced by these improved expression profiles.

Visualization Finally, we design the informational bottleneck layer of the autoencoder to be two dimensional, which lets it serve as a visualization and nonlinear embedding of the data. Because the network must reconstruct the input accurately from this internal representation, it must compress all the information about a cell into just these two dimensions, unlike methods like PCA or Diffusion Maps, which explicitly leave some variation unmodeled. Consequently, the information stored is also global, meaning points close together in the SAUCIE visualization are more similar than points that are farther apart, which is not true beyond small neighborhoods in a local method like tSNE. The ability to flexibly learn and accurately reflect the structure in the data with SAUCIE is demonstrated in Figure 6.3B.

Considered together, these customized regularizations and architectural choices make SAUCIE ideally suited for the exploratory data analysis when presented with single-cell biological data. Further, SAUCIE is entirely self-contained and not require any external algorithms that may not be able to process the scale of multisample single-cell data.

6.2.2 Comparison to other methods

We begin by offering an extensive comparison between SAUCIE and other (generally specialized) methods at each of these tasks in turn. We find that SAUCIE performs as well as, or even better than, specialized algorithms, which are much less scalable, for each individual task. Moreover, SAUCIE performs all tasks on a unified representation leading to visualizations that are coherent with clusters and cluster expression.

Throughout the comparisons on each of the tasks, we use an artificial dataset (simulation from mixtures of Gaussians), along with ten different single-cell datasets. Five datasets are CyTOF: the dengue dataset we extensively evaluate later in the manuscript, T cell development data from [140], renal cell carcinoma data from [33], breast tumor data from [14], and iPSC data from [190]. Five datasets are scRNA-seq: mouse cortex data, retinal bipolar cells from [143], hematopoiesis data from [126], mouse brain data from [182], and the 10x mouse megacell demonstration from [1].

fig:regularizations

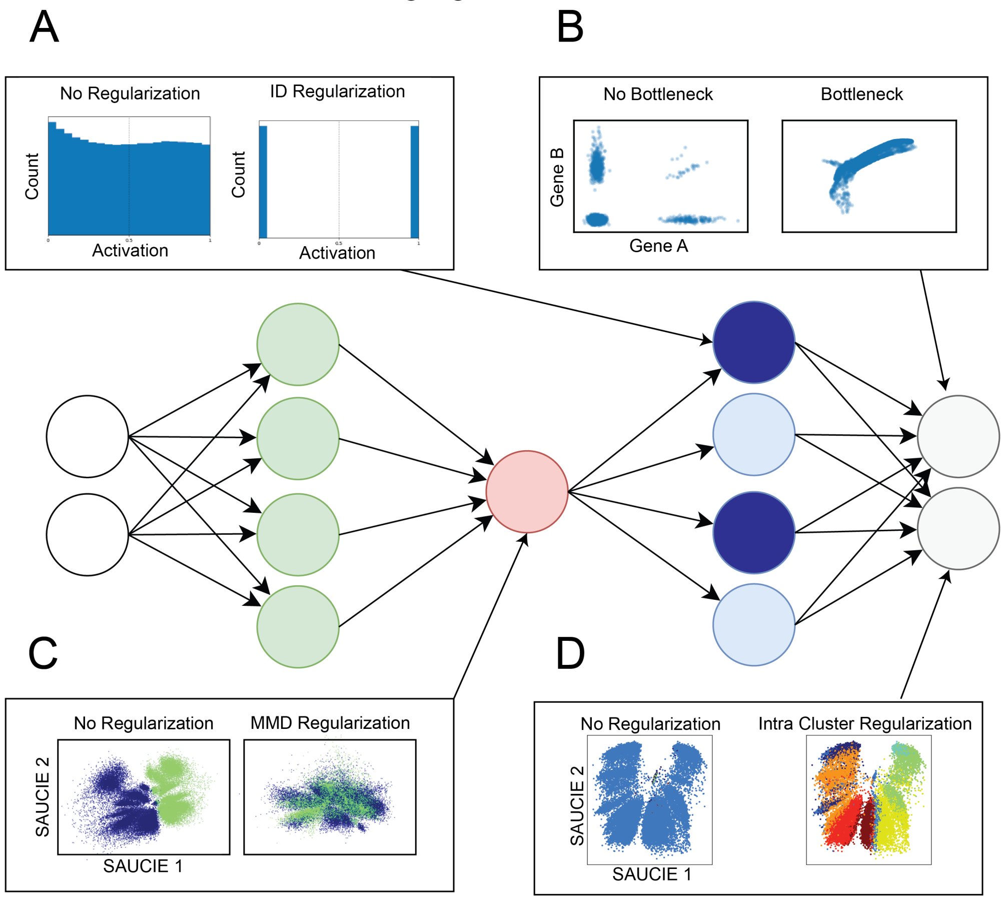


Figure 6.2: **Regularizations and architecture choices in SAUCIE.** A) the ID regularization applied on the sparse encoding layer produces digital codes for clustering B) the informational bottleneck, i.e. a smaller embedding layer, uses dimensionality reduction to produce denoised data at the output C) the MMD regularization removes batch artifacts D) the within cluster distance regularization applied to the denoised data provides coherent clusters.

fig:smallcomp

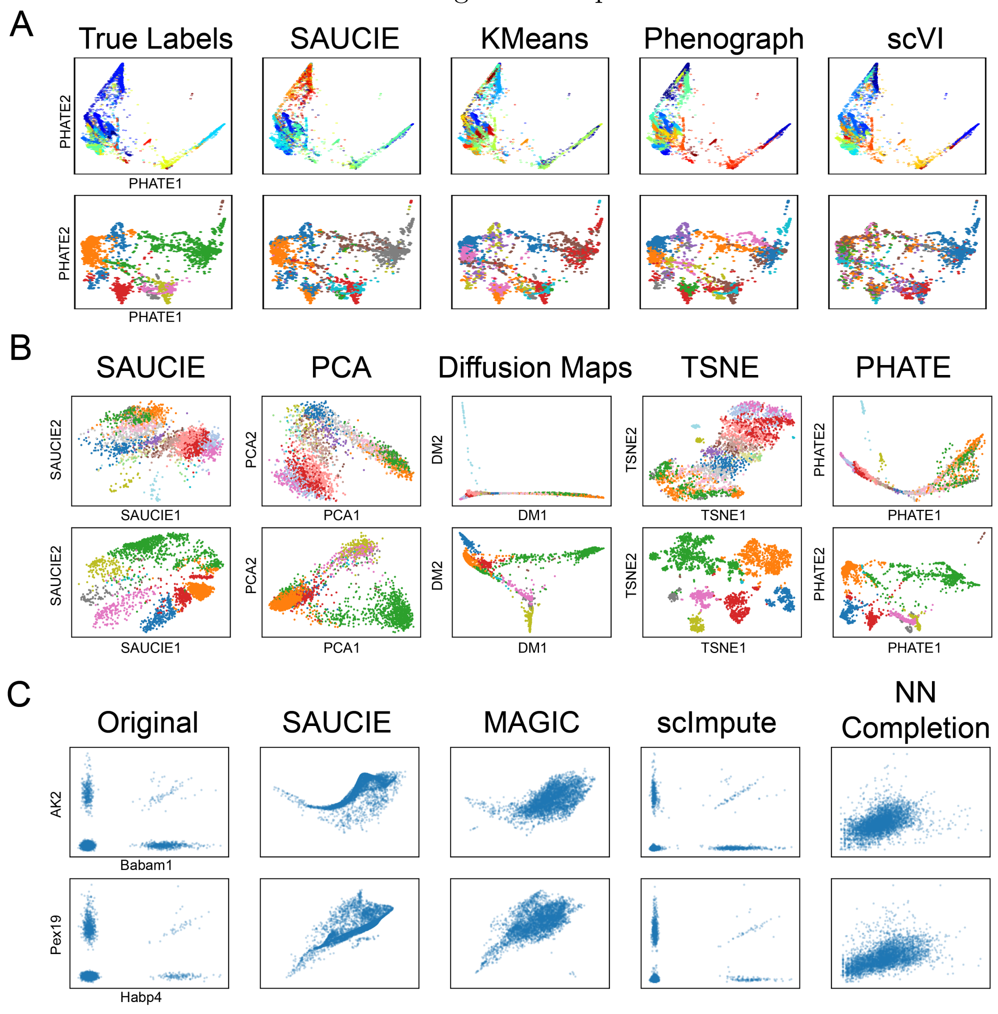


Figure 6.3: **A comparison of the different analysis tasks performed by SAUCIE against other methods.** A) A comparison of clustering performance on the data from Shekhar et al (top) and Zeisel et al (bottom) with samples of size 27499 and 3005, respectively. B) A comparison of SAUCIE's visualization on the same datasets as part (A). C) A comparison of imputation on the 10x mouse dataset subset of size 4142.

suppfig:clustcomp

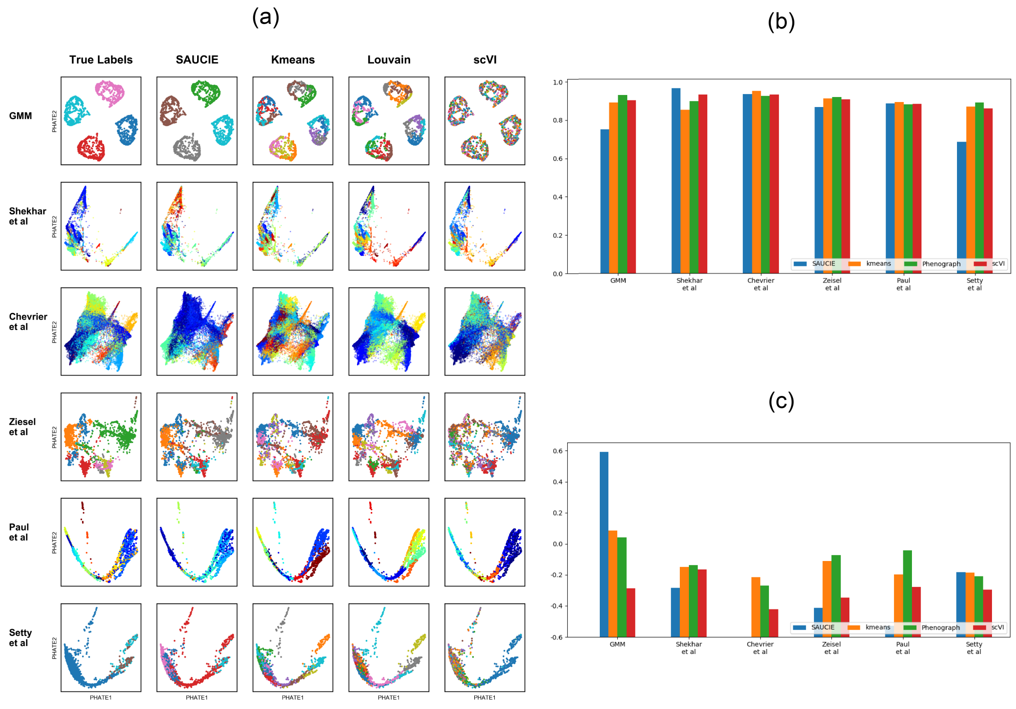


Figure 6.4: A comparison of the SAUCIE clustering to other clustering methods on artificial and real data. Rows show the different datasets. Along with the first artificial dataset, there are two CyTOF datasets and three scRNA-seq datasets. Columns show the different clustering methods. From left to right: True “ground truth” labels, SAUCIE, kmeans, Phenograph, scVI. In (b) and (c), we add the scores for the modularity and silhouette heuristics from Table 6.1, respectively.

Clustering

To evaluate the ability of SAUCIE to find meaningful clusters in single-cell data, we compare it to several alternative methods: minibatch kmeans [127], Phenograph [90], and another neural network approach called Single-cell Variational Inference (scVI) [100]. While we compare to scVI as it and SAUCIE are both neural networks, we emphasize a fundamental difference between the two: scVI only returns a latent space, which must then be visualized or clustered by another outside method, while SAUCIE explicitly performs these tasks. Since kmeans needs to be told how many clusters there are ahead of time (k), we use the number of clusters identified by Phenograph as k. We look at the following datasets: artificially generated Gaussians rotated into high dimensions, and public single-cell datasets for which we have curated cell clusters as presented by the authors: [143], [33], [182], [126], and [140].

tab:scoresclustering

	SAUCIE	kmeans	Phenograph	scVI
GMM	0.7512/0.8162	0.8917/0.3097	0.9302/0.2662	0.9030/-0.0626
Shekhar et al	0.9662/-0.0602	0.8530/0.0753	0.8981/0.0868	0.93139/0.0593
Chevrier et al	0.9347/-0.3761	0.9517/0.0085	0.9258/-0.0452	0.9330/-0.1967
Zeisel et al	0.8663/-0.1881	0.9138/0.1135	0.9209/0.1529	0.9085/-0.1238
Paul et al	0.8854/-0.3060	0.8930/0.0249	0.8819/0.1802	0.8839/-0.0540
Setty et al	0.6860/0.0425	0.8704/0.0377	0.8912/0.0147	0.8591/-0.0718

Table 6.1: A comparison of modularity (left) and silhouette (right) scores of each of the clustering algorithms on each dataset.

In addition to analyzing the clusters visually (Figure 6.4), we also quantitatively assess cluster performance of the methods by computing modularity and silhouette scores [127] on the generated clusters and ground truth labels (Table 6.1). First, we look at an artificially generated dataset of four two-dimensional Gaussian point clouds with different means rotated into 100 dimensions. We find that SAUCIE is the only method that automatically identifies exactly four clusters, which was the underlying number of clusters in the generation model. This illustrates why optimizing modularity, like Phenograph does, is not necessarily the best heuristic to follow, as it adds additional complexity to the clustering in order to increase the modularity score, resulting in too many clusters. Likewise, scVI did not identify the four clusters, which is unsurprising as the data did not fit its parametric model appropriate for gene counts.

We also examine clustering performance on five public single-cell datasets to evaluate the ability of SAUCIE to cluster real biological data: from [143], [33], [182], [126], and [140]. Visual inspection reveals that SAUCIE produces clusters that are qualitatively coherent on the embedding. Quantitatively, the modularity scores of its clusters corroborate this evaluation. As shown in Table 6.1 the average modularity score across datasets is 0.8531. In a wide variety of data from both CyTOF and scRNA-seq measurements, SAUCIE is able to produce clusters that reasonably represent the data qualitatively, quantitatively, and by comparison to other methods.

fig:batchcorrectioncartoon

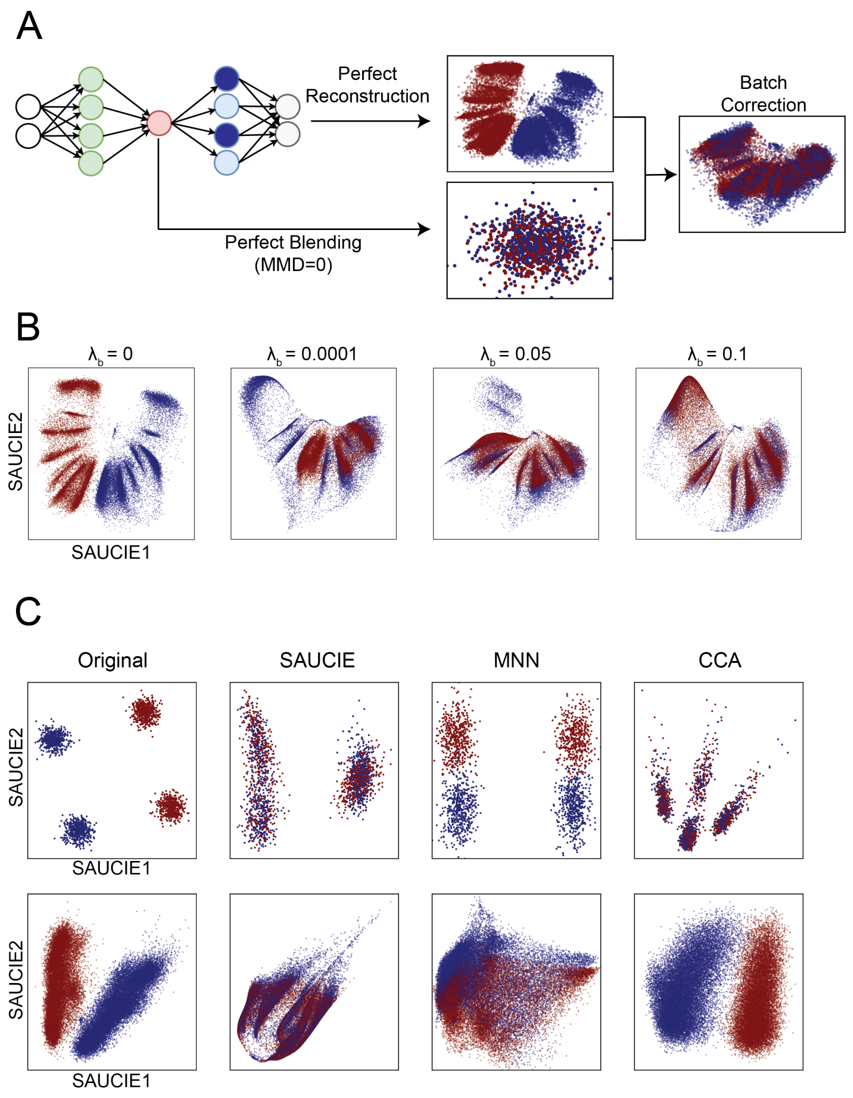


Figure 6.5: **Demonstration of SAUCIE's batch correction abilities.** A) SAUCIE batch correction balances perfect reconstruction (which would leave the batches uncorrected) with perfect blending (which would remove all of the original structure in the data) to remove the technical variation while preserving the biological variation. B) The effect of increasing the magnitude of the MMD regularization on the dengue data of size 41721. Sufficient MMD regularization is capable of fully removing batch effect. C) Results of batch correction on the synthetic GMM data (of size 2000) (top) and the dengue data (bottom) shows that SAUCIE better removes batch effects than MNN and better preserves the structure of the data than CCA.

suppfig:batchcorrectioncomp

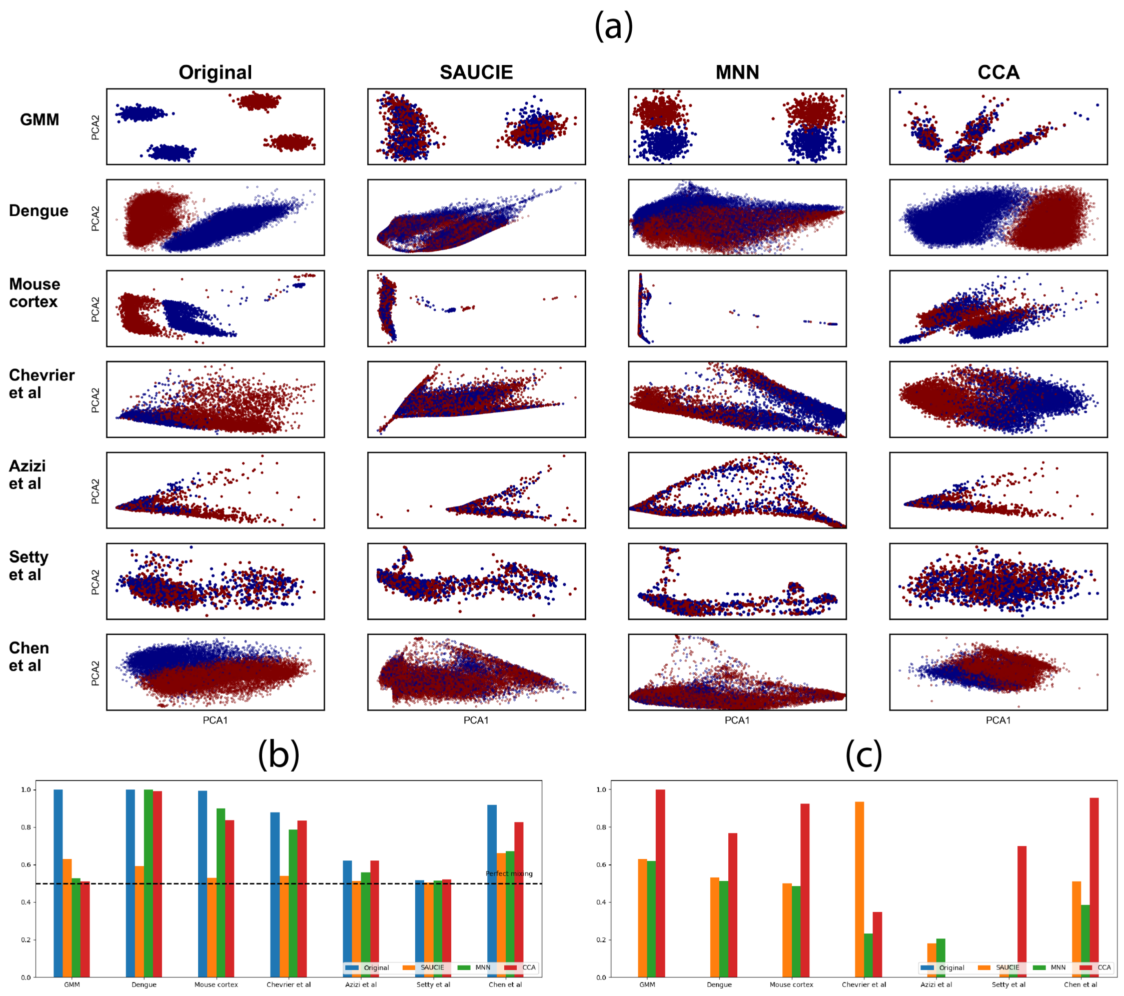


Figure 6.6: A comparison of batch correction with SAUCIE to other methods on an artificial dataset, two technical replicates from the dengue CyTOF data, non-technical replicates on scRNA-seq batches from mouse cortex, and then public data from Chevrier et al, Azizi et al, and Setty et al. Rows show the different datasets. Columns show the different batch correction methods. From left to right: The original data prior to batch correction, SAUCIE, mutual nearest neighbors (MNN), canonical correlation analysis (CCA). In (b) and (c), we add graphs of the mixing score and shape preserving score results from Table 6.2 for quantitative evaluation, respectively.

tab:scoresbatchcorrection

	Original	SAUCIE	MNN	CCA
GMM	0.999/—	0.630/0.629	0.526/0.620	0.510/0.998
Dengue	0.999/—	0.593/0.532	0.998/0.512	0.992/0.765
Mouse cortex	0.994/—	0.530/0.498	0.898/0.485	0.836/0.923
Chevrier et al	0.880/—	0.540/0.934	0.787/0.232	0.835/0.346
Azizi et al	0.621/—	0.512/0.180	0.560/0.205	0.621/0.000
Setty et al	0.518/—	0.504/0.064	0.514/0.067	0.523/0.698
Chen et al	0.919/—	0.661/0.508	0.672/0.383	0.826/0.955

Table 6.2: A comparison of mixing (left) and Procrustes (right) scores of each of the batch correction algorithms on each dataset.

Batch correction

We assess our ability to remove batch-related artifacts with SAUCIE by comparison to two published batch correction methods that have been specifically designed to remove batch effects in single-cell data. The first, Mutual Nearest Neighbors (MNN) [58], uses mutual nearest neighbors on a k-nearest neighbors graph to align two datasets, and the second, Canonical Correlation Analysis [25], finds a latent space in which the two batches are aligned. To evaluate the performance of these methods and SAUCIE, we use several different datasets with varying degrees of batch artifacts. We note that SAUCIE is the only method capable of scaling batch correction to hundreds of samples as we do in the next section. Nonetheless, here we compare performance on datasets small enough for the alternative methods to handle.

To quantitatively assess the quality of batch correction, we apply a test we term the *mixing score* (similar to that of [27]):

$$mixing\ score = \frac{N_{b_1}}{N_{b_2}} * \Sigma_{x_j \in KNN(x_i)} \left(\mathbb{1}_{batch(x_i)=batch(x_j)} \right) \tag{6.1}$$

where N_{b_1} and N_{b_2} are the number of points in the first and second batch respectively. This score calculates for each point the number of nearest neighbors that are in the same batch as that point, accounting for the difference in batch sizes. In perfectly mixed batches, this score is 0.5, while in perfectly separated batches it is 1.0. As batch correction should not only mix the batches but also preserve their shape as best as possible, we quantify the

	SAUCIE	PCA	Monocle2	DM	UMAP	TSNE	PHATE
Artificial Tree 3	0.993	0.956	0.935	0.963	0.967	0.968	0.947
Artificial Tree 7	0.994	0.951	0.921	0.980	0.986	0.990	0.971
Artificial Tree 20	0.948	0.896	0.854	0.940	0.938	0.940	0.939
DLA Tree	0.865	0.817	0.725	0.819	0.836	0.845	0.847
Half Circles	0.975	0.970	0.940	0.937	0.958	0.946	0.925
GMM	0.999	0.972	0.953	0.500	0.992	0.992	0.969
Paul et al	0.944	0.948	0.896	0.807	0.842	0.865	0.856
Setty et al	0.882	0.870	0.839	0.508	0.501	0.501	0.491
Zunder et al	0.939	0.903	0.884	0.505	0.522	0.513	0.510
Shekhar et al	0.942	0.908	0.918	0.506	0.863	0.508	0.496
Ziesel et al	0.952	0.914	0.903	0.943	0.909	0.881	0.905

Table 6.3: A comparison of precision-recall area-under-the-curves (AUCs) for each of the visualization algorithms on each dataset.

distortion between the original and batch corrected data using Procrustes, which finds the error between the optimal alignments of the two batches by linear transformation [107]. These numbers are reported in Table 6.2. While the other methods each have some datasets that violate their assumptions and thus they perform poorly, SAUCIE performs as well or better at each of the wide variety of datasets.

We compare SAUCIE to the alternative methods on artificial GMM data, spike-in CyTOF samples from the dengue dataset, non-technical scRNA-seq replicates from developing mouse cortex, and four public datasets from [33], [14], [140], and [32]. In Table 6.2, we see SAUCIE has the best average mixing score across the wide range of data types, without distorting the data more than is necessary.

Visualization

To evaluate the SAUCIE visualization and its ability to provide a faithful low-dimensional data representation, we provide comparisons of this visualization to other frequently used methods. We make use of artificial datasets where the underlying structure is known, as well as real biological datasets that have been extensively characterized previously, so we have prior understanding of the structure we expect to see in the visualization (Figure 6.7).

We measure the quality of the visualizations with a quantitative metric taken from [106]. In line with their method's precision and recall metrics, we compute a neighborhood around suppfig:vizcomp

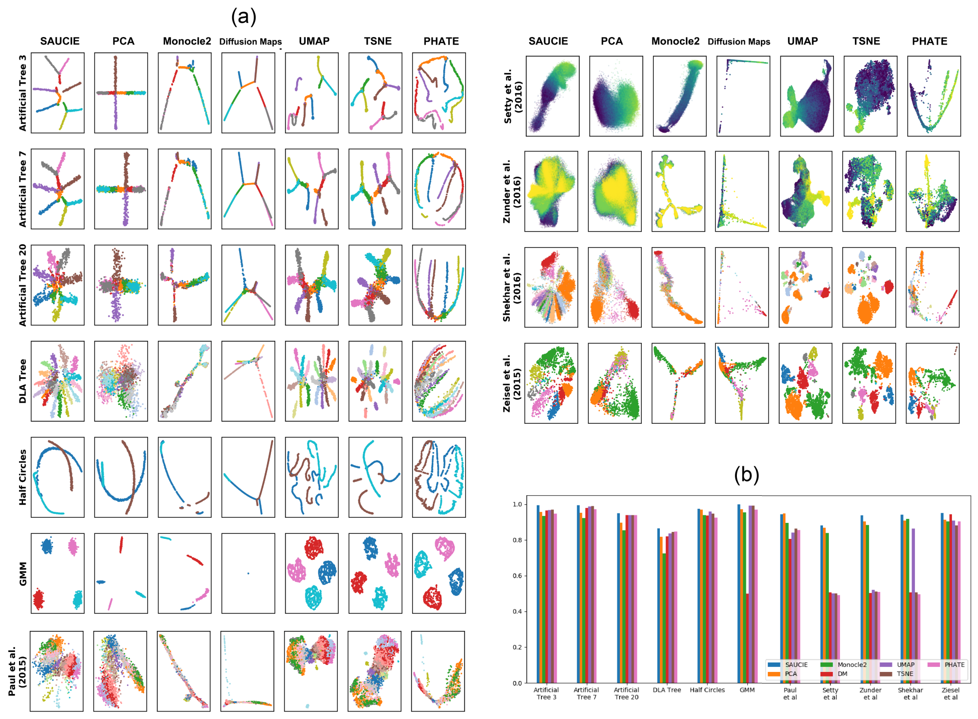


Figure 6.7: A comparison of the SAUCIE visualization to other methods on a number of artificial and real datasets. The columns show the different methods. From left to right: SAUCIE, PCA, Monocle2, Diffusion Maps, UMAP, tSNE, PHATE. The rows show the different datasets. From top to bottom: Artificially generated trees with varying amounts of noise, random tree generated with diffusion limited aggregation (DLA), intersecting half circles, Gaussian mixture model, scRNA-seq hematopoiesis from Paul et al [126], CyTOF T cell development from Setty et al [140], CyTOF ipsc from Zunder at al [190], scRNA-seq retinal bipolar cells from Shekhar et al [143], scRNA-seq mouse cortex from Zeisel et al [182]. In (b), we add a graph of the precision-recall metric results from Table 6.3 for quantitative evaluation.

each point in both the original data space and the embedding space, and compare the neighbors of each. An embedding with high recall has most of a point's original-space neighbors in its embedding-space neighborhood. Similarly, an embedding with high precision has most of the point's embedding-space neighbors in its original-space neighborhood. As directed by the authors' algorithm, we gradually increase the size of the neighborhood and report the area-under-the-curve (AUC) for the precision-recall curve. These results are in Table 6.3, where SAUCIE has the highest average score of 0.9342, averaged across all datasets. Despite this, we note that this is only a heuristic and can give undesirable results at times, as it only looks at fixed neighborhoods in the original input space. For example, PCA fails to produce any visual separation in the data from Zunder et al, yet scores well by this metric. Likewise, tSNE artificially shatters the trajectories in the DLA data, yet produces a high score. Nonetheless, this metric offers some corroboration at the quality of SAUCIE's visualization, a hard task to measure quantitatively.

In Figure 6.7, we some methods preserve global information, but frequently at the expense of not preserving local and more fine-grained variation, like PCA. Other methods, like Diffusion Maps, provide visualizations that look like connected trajectories on every dataset, no matter the underlying distribution. tSNE preserves local information, but at the expense of not preserving global information, on the other hand. SAUCIE, meanwhile, balances global and local information preservation and provides varied visualizations, depending on the structure of the underlying data.

Imputation

We analyze the SAUCIE imputation and its ability to recover missing values by implicitly interpolating on a data manifold in several ways. First, Figure 6.8 shows several relationships from the scRNA-seq data of the 10x mouse megacell dataset affected by severe dropout. This dataset consists of 1.3 million cells, and SAUCIE was the only method in the comparison to be able process the full dataset. Moreover, it was able to do this in just 44 minutes. Additionally, because training a neural network only requires small minibatches in memory at one time, we were able to do this without ever loading the entire large dataset into memory all at once. Thus, to enable this comparison, we subsampled the data by taking suppfig:imputationcomp

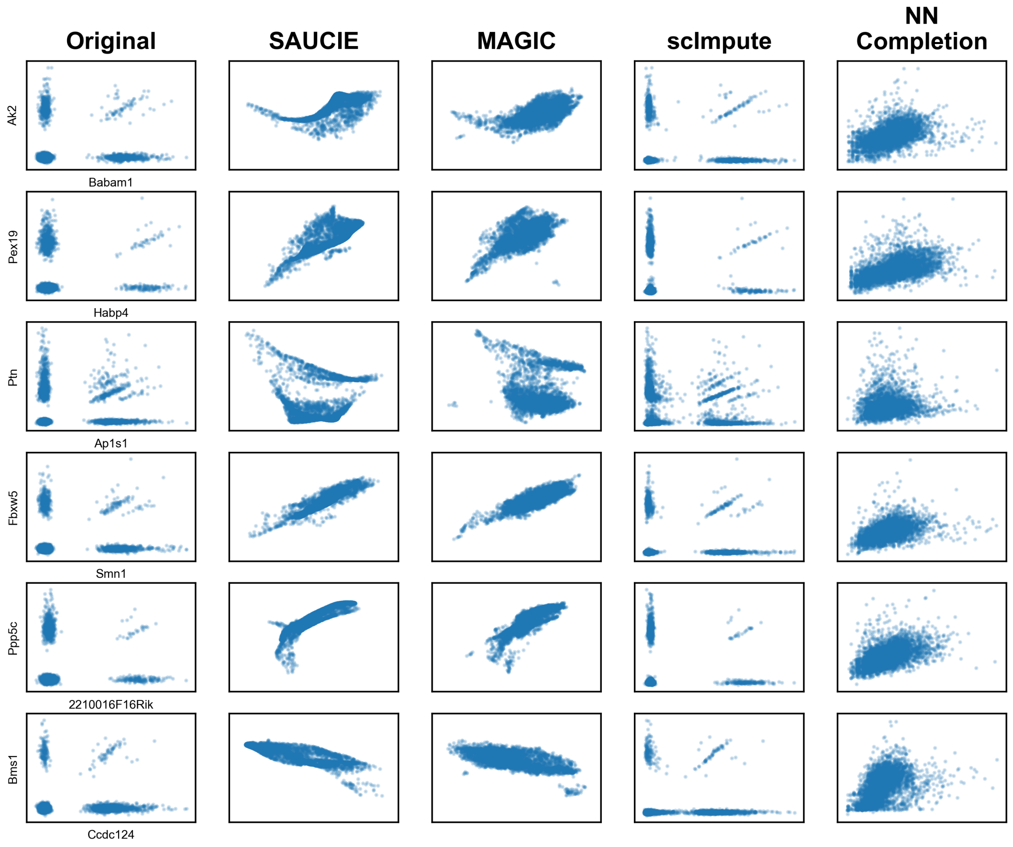


Figure 6.8: A comparison of imputation methods including SAUCIE. Several gene-gene associations are shown from the 10x mouse cortex dataset. From left to right: The original (sparse) data, data after imputation with SAUCIE, MAGIC, scImpute, and nearest neighbor completion.

one of the SAUCIE clusters consisting of 4172 cells.

For this comparison, we measure against several popular imputation methods for scRNA-seq data: MAGIC, which is a data diffusion based approach, scImpute, which is a parametric statistical method for imputing dropouts in scRNA-seq data, and Nearest Neighbors Completion (NN Completion), which is an established method for filling in missing values in a general application of high-dimensional data processing.

In Figure 6.8, we show six relationships of the mouse megacell dataset for the original data and the different imputation methods. We observe that the original raw data is highly sparse, which can be seen by the large number of values on the axes where one of the variables is exactly zero. Note that most cells have one or both genes missing. This is a problem because this prevents us from identifying trends that exist between the genes. After imputation with SAUCIE, we can observe that the sparse character of the data has been removed, with values filled in that reveal underlying associations between the gene pairs. These associations are corroborated by MAGIC, which imputes similar values to SAUCIE in each case. MAGIC is a dedicated imputation tool that is widely used, so SAUCIE matching the relationships it found gives confidence in the ability of SAUCIE to impute dropout effectively. The resulting imputation in scImpute does not look significantly less sparse from the original and we do not see continuous trends emerge. NN Completion appears to desparsify the data, but the resulting trends all look similar to each other (i.e., positively correlated). This suggests that it does not correctly identify the underyling trends, as we would expect different genes to have different relationships. While scRNA-seq is highly sparse, the undersampling affects all entries in the matrix, including the nonzero values. As such, manifold-based methods like SAUCIE and MAGIC are more suited for finding these true relationships because they denoise the full dataset as opposed to just filling in zeros.

Due to the fact that ground truth values for the missing counts in this single-cell data are not known, we further test the accuracy of the imputation abilities of SAUCIE with an artificially constructed experiment. We first leverage the bulk RNA sequencing data of 1076 cells from [159], because it accurately captures the relationships between genes due to it not being sparse (as opposed to generating our own synthetic data from a parametric generating function that we have the ability to choose, where we can create the relationships). We then

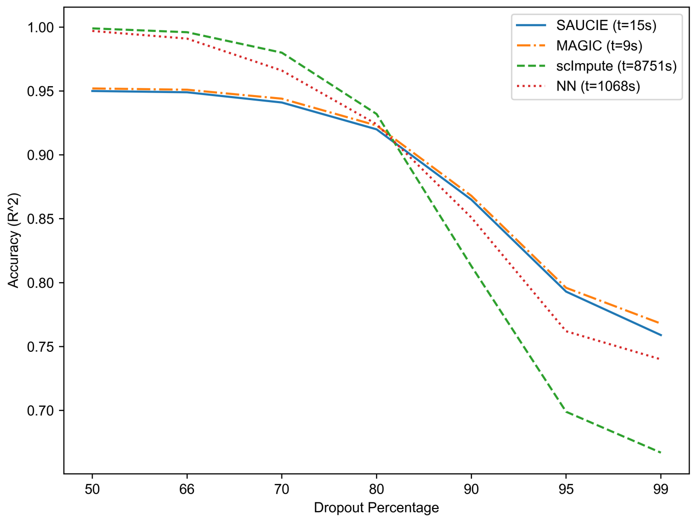


Figure 6.9: A comparison of imputation with SAUCIE to other methods on the simulated dropout experiment. Increasing amounts of dropout are along the horizontal axis from left to right, and the accuracy of each method as measured by R^2 is along the vertical axis. The time each method took to complete is in the legend in seconds.

simulate increasing amounts of dropout and compare the imputed values returned by each method to the true values we started with. To simulate dropout in a manner that reflects the underlying mechanisms of inefficient mRNA capture, we remove *molecules* instead of just setting values for genes to zero. As a result, the level of dropout is conditional upon expression level, reflecting the dropout structure of single-cell RNA sequencing data. The results are reported in Figure 6.9, where SAUCIE compares favorably to other methods, recovering the true values accurately even after as much as 99% dropout. The dataset for this experiment consisted of just 1076 cells, which allowed us to compare to the methods that cannot process larger datasets, but even on a dataset of this size SAUCIE gave a more than 100-times speedup over NN Completion and 600-times speedup over scImpute.

Runtime Comparison

In order to showcase the scalability of SAUCIE, we compare to a host of other methods on a subset of our newly generated CyTOF dataset consisting of over 11 million cells existing in 35 dimensions. We display the runtimes of each method on a random sample of N points,

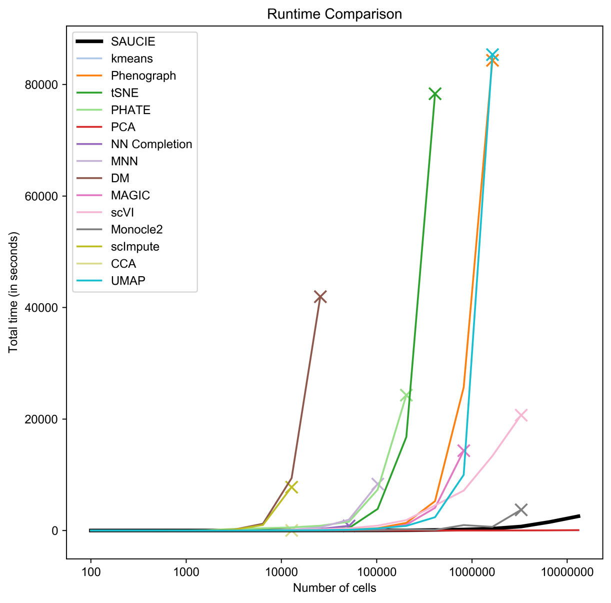


Figure 6.10: Comparison of runtimes on an increasing number of points. The number of points is represented on the horizontal axis and the time in seconds the method took to complete is on the vertical axis. If a method ran out of resources and could not complete a run for a certain number of points, that is demarcated with an 'x' and no further time points were attempted for that method. SAUCIE is the fastest method besides PCA and kmeans.

with $N = 100, 200, 400, 800, \ldots, 11000000$ in Figure 6.10. For each step, the method was given a timeout after 24 hours. Points where a method stopped scaling in Figure 6.10 are marked with an 'x'.

SAUCIE performs visualization, batch correction, imputation, and clustering in its run, while each of the other methods only performs one of these tasks. Moreover, SAUCIE does not just compute simple linear functions on the data, but instead performs complex non-linear transformations in the process. Despite its complexity, it also scales very well with the extremely large dataset sizes, which can be further improved by simply adding more independent GPUs for calculations. Each additional (relatively inexpensive) GPU can offer a near linear increase in computation time, as opposed to more CPUs which offer diminishing returns in parallelizability. All experiments were run on a single machine with just one GPU, meaning these results could still benefit even more from this potential for scalability. For further details on how the runtime experiment was performed, see the Methods section.

Among the batch correction methods, there are no other methods that correct multiple batches simultaneously. However even when we restrict to pairwise comparisons, SAUCIE is the only method that comes close to handling this amount of data. CCA and MNN both stop scaling in the tens of thousands of cells. In the group of imputation methods, scImpute and NN completion also stop scaling in the tens of thousands, while MAGIC stops scaling in the hundreds of thousands. For visualization, PCA was the only method faster than SAUCIE, which is unsurprising because calculating it using fast randomized SVD is quick, but it gives a simple, strictly linear blurry views of the data, in contrast to SAUCIE's nonlinear dimensionality reduction. The other more complex visualization methods do not scale to these dataset sizes: Diffusion Maps, PHATE, tSNE, and Monocle2 all stop scaling before even reaching the full eleven million cells. For clustering, kmeans is the only one faster than SAUCIE, due to using its minibatched version. However, it still assumes circular clusters in the Euclidean space and comes with the intrinsic flaw that the number of clusters must be known ahead of time, which is not possible in any realistic setting like ours where we are performing exploratory data analysis on a large new dataset. Phenograph and scVI do not scale to the full dataset, either. Despite being another neural network method, scVI cannot scale to these larger sizes because it only produces a latent space that then must be clustered with another method. This requirement then becomes its bottleneck, emphasizing the importance of SAUCIE performing all tasks directly instead of acting as a pre-processing step for other methods.

SAUCIE is the only method that can efficiently batch correct, impute and denoise, visualize, and cluster datasets of this size, while using a nonlinear manifold representation of the data.

6.2.3 Analysis of immune response to dengue infection with SAUCIE

Next, we demonstrate an application of SAUCIE as an important tool enabling exploratory analysis of a new “big” dataset that consists of single-cell CyTOF measurements of T cells from 45 subjects including a group acutely infected with the dengue virus and healthy controls from the same endemic area [187]. While dengue is estimated to affect sixty million people yearly and cause ten thousand deaths, like other tropical diseases, it remains understudied. Moreover, dengue is especially challenging since there are several different serotypes with complex interactions between them. Specifically, there are four strains that have very different characteristics. While infection with a particular strain may provide some immunity towards reinfection with that same strain, an antibody dependent enhancement results in faster uptake of another strain upon reinfection [79]. Drugs have proven difficult to develop for dengue. Further, vaccine development has also been challenging in the case of dengue. Recently, the WHO has ruled that the dengue vaccine of Senofi Pasteur only be administered to patients who are infected for the second (or subsequent) time [139]. This is because the vaccine itself is thought to leave patients vulnerable to very severe reinfections. So unlike other viruses, the dengue virus apparently leaves patients more vulnerable the second time. These types of complex effects require deep and detailed analysis of both infected and convalescent patients at the single cell level to understand the immune response.

We applied SAUCIE to the single-cell CyTOF data of T cells collected in an area endemic for dengue virus infection [187] to study general T cell compartment composition, variability and changes in the variability after convalescence. We believe that the dengue data is an ideal test case for SAUCIE, because the samples are shipped from India and samples were collected over a period of months and were assesed over different experiment days

[187]. Thus, there is a pressing need for batch correction and data cleaning as well as uniform processing, clustering and meta-analysis of patient stratification. As part of the study, cells from additional patient groups beyond the acutely infected were also measured: healthy people unrelated to the subjects as a control and the same acute subjects at a later convalescent time point. Primary research questions include understanding profile of the acute subjects and how they differ from the other groups. Across all groups, there are 180 samples resulting in over twenty million cells with results analyzed on 35 different protein markers, a massive amount of data that would cause difficulties in most standard analytic frameworks.

Differential cluster proportions between subjects

We first batch correct and denoise the data using SAUCIE's MMD regularization and then obtain the clusters characteristic of each group and then further analyze them for marker enrichments as single cell versions of blood biomarkers [130]. For the clustering considered here, we use a coarse-grained clustering obtained with a coefficient for ID regularization of 0.1. This was chosen by scanning across values of 0.01, 0.1, 0.2, 0.3, 0.4, and 0.5, and choosing the clustering that yielded the best modularity. If other granularities are desired, lower coefficients could be used and the impact of this parameter on the number of clusters is shown in Figure 6.12. The two regularizations λ_d and λ_c affect the number of clusters that result. For a given value of λ_d, as λ_c increases, the number of clusters decreases (coarser granularity). Higher values of λ_d yield more clusters (finer granularity). Notably, these results are robust and yield reasonable results for varying values of the two regularizations. These two together act as knobs that can be tuned to get the desired granularity of clustering. The methods section further discusses how these regularizations affect the number of clusters.

For the SAUCIE clustering, we focus on T cells as particularly relevant to the immune process and an abundant subset of the data (eleven million total cells), looking for clusters that are over- or under-represented in the cells of each group. We look for clusters that behave differently in the acute compared to the convalescent time points. These would then represent a population of cells that might have an important role in the process, which could be further investigated. To understand what cell population this is, we examine the marker fig:zikaclustering

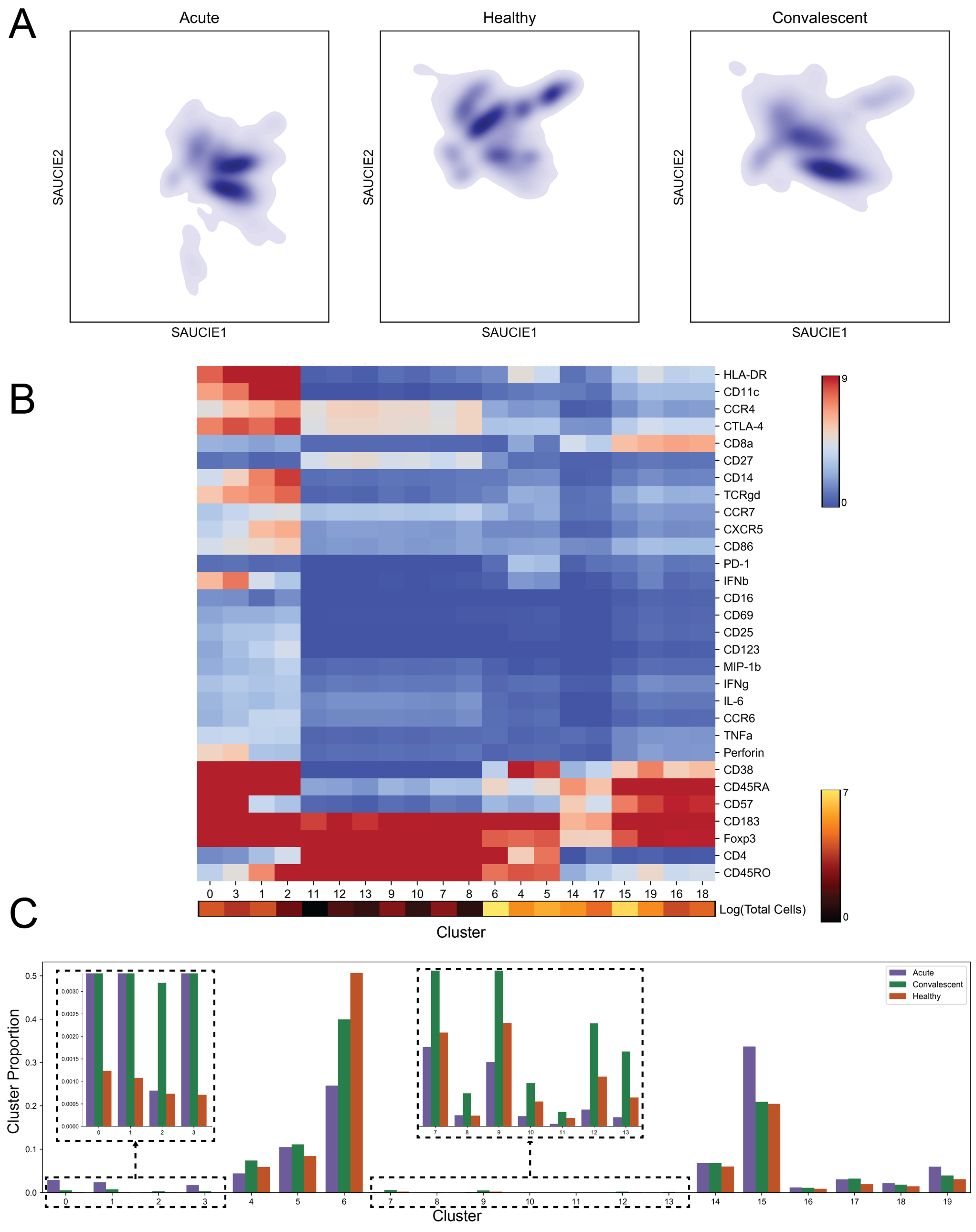
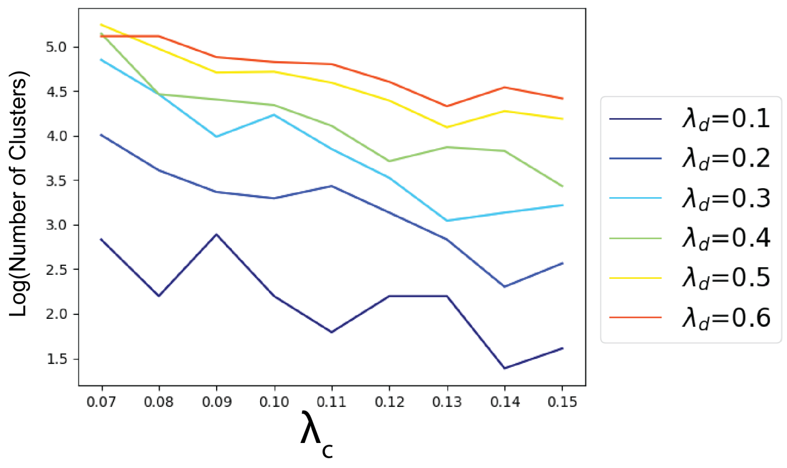


Figure 6.11: **SAUCIE identifies and characterizes cellular clusters, whose proportions can be used to compare patients.** SAUCIE on the entire dengue dataset of 11228838 cells. A) The cell manifolds identified by the two-dimensional SAUCIE embedding layer for the T lymphocyte subsets from acute, healthy, and convalescent subjects. B) A heatmap showing clusters along the horizontal axis and markers along the vertical axis. Cluster sizes are represented as a color bar beneath the heatmap. C) Cluster proportions for acute, convalescent, and healthy patients.



suppfig:clustering

Figure 6.12: The granularity of the clustering, as measured by the total number of clusters found. Each line represents a fixed value of λ_d as λ_c increases from left to right.

abundance profile for the cluster. The mean for each cluster and marker is shown in the heatmap in Figure 6.11B.

We find twenty total clusters within the T cell populations, five of which are CD8 T cells and thirteen of which are CD4 T cells. In addition, interestingly, there are six clusters of CD4-CD8- T cells, where four are $\gamma\delta$ T cells. These have been noted as a characteristic of reaction to viral infections [123, 158, 49, 34, 36]. There are twelve clusters representing effector memory cells and nine regulatory T cells that are CD4+Foxp3+. Two of the clusters are naive T cells.

Several of these populations are indicative of differences between acute, convalescent, and healthy subjects, and can be used for characterizing the nature of the reaction of each of these groups, as we do below.

1. $\gamma\delta$ T cells are a relatively rare type of T cells, but SAUCIE is still able to identify them. Despite their rarity, they appear to have significance in identifying different populations, which emphasizes the importance of this attribute of SAUCIE. These cells signal especially strong early in immune response, particularly skin and mucosal immunity. They have less variable TCR sequences than $\alpha\beta$ T cells [131]. These cells are a bridge between T cells and myeloid cells, as they have some innate immune activity, where they express CD11c and CD86. They can bind to lipid antigens. Clusters 0 and 3 (consisting of 7% of the total cells) show upregulation of CD57. This is an

indication of terminal differentiation. CTLA-4 and CD38 are also high, so these are highly activated cells and potentially dysfunctional. We see that these clusters are highest in the acute subjects and lowest in the healthy subjects. Out of the fifteen subjects that were measured both as acute subjects and later in convalescence, thirteen had more of these cells during their acute infection.

2. We find another group of $\gamma\delta$ T cells that are CD45RO and CD45RA positive (cluster 2, consisting of 1% of the total cells), but not yet fully terminally differentiated, so these could be transitional between naïve and effector memory. The effector memory cells express less IFNb. As this cluster is more expressed in the healthy subjects, it indicates that even these subjects may have had some exposure to dengue. There is a lack of an inflammatory state, i.e., low in IFNb and Perforin, so we expect that these are actually memory cells instead of effector cells. It makes more sense then that these populations are more expressed in convalescent and healthy subjects.

3. We also find another population of CD4+ T cells (clusters 3-15, consisting of 45% of the total population) that are not expressing any inflammatory markers or activation markers, and these are higher in the convalescent and healthy subjects, while being very low in the acute subjects. These look to be other memory cells that may characterize these convalescent subjects. In fact, out of the fifteen subjects with acute-convalescent paired measurements, eleven had more of these cells during convalescent measurement. These have signs of recent activation as they do not have CD69, which is an early activation marker, nor any of the cytokines like IFNg, IFNb, or IL-6.

4. Additionally, we find a population of CD8+ effector cells (cluster 15, which consists of 3% of the total cells) that are highly expressed in the acute subjects. These cells also express CD57 and CD38, but are not $\gamma\delta$ as the previous populations were. These appear to be more differentiated and are likely not transitional, as the previous ones were, either.

We can also visualize the cell-level cluster proportions on a patient manifold (Figure 6.13B). There, we see that cluster proportions arranged on this manifold reveal clusters that are changing across the space. This analysis indicates clearly that cluster 1 is representative of acute subjects and cluster 5 is representative of the healthy subjects. Furthermore, we can evaluate the same individual when measured after acute infection, and then later at a convalescent time point (Figure 6.13C). Viewed in this way, we see that cluster 11 is also more present in most subjects when they came in with an acute infection than at the convalescent time point.

Visualization

SAUCIE can process all cells from all subjects to construct a cellular manifold and extract its features. First, we visualize this manifold using the 2-D visualization layer. Figure 6.11A is divided into two embeddings that show the cell manifolds for acute and healthy subjects separately. As can be seen, there is a characteristic change in the manifold that becomes apparent when comparing the embeddings side-by-side. The acute subjects have cell populations distinctly missing that are present in the healthy subjects.

After characterizing the nature of the cellular space in the aggregate, we can additionally analyze manifolds formed by the distributions of T lymphocytes within each patient separately. As each patient has a heterogeneous population of cells, including with different total numbers of cells, it becomes a challenge to define a meaningful measure of similarity between the individuals. Here we are able to leverage the manifold constructed by the SAUCIE embedding and calculate MMD (a distribution distance) between the distribution of cells in the latent space for each pair of subjects. With a measure of similarity between each pair of patients, we can now construct a manifold not of the cells but also of the *subjects* (Figure 6.13A).

6.3 Discussion

We presented SAUCIE, a neural network framework that streamlines exploratory analysis of datasets that contain a multitude of samples and a large volume of single cells measured in each sample. The key advantage in SAUCIE is its ability to perform a variety of crucial tasks on single-cell datasets in a highly scalable fashion (utilizing the parallelizability of deep learning with GPUs) without needing to call external algorithms or processing methods. As a fig:mdsonmmd

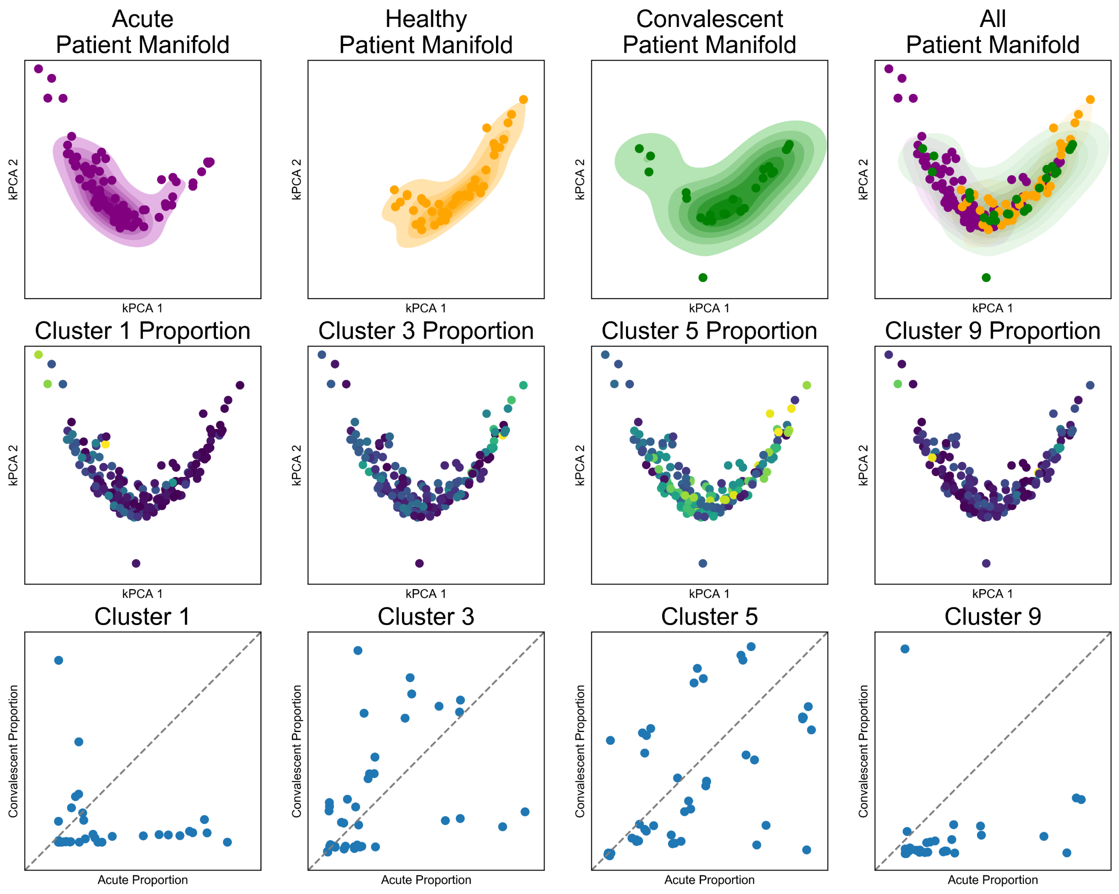


Figure 6.13: **SAUCIE produces patient manifolds from single-cell cluster signatures.** SAUCIE on the entire dengue dataset of 11228838 cells. Top row) The patient manifold identified by SAUCIE cluster proportions, visualized by kernel PCA with acute, healthy, convalescent, and all subjects combined from left to right. The healthy manifold overlaps with the convalescent manifold to a much higher degree than the acute manifold. Middle row) The same patient manifold shown colored by each patient's cluster proportion. Cluster 1 is more prevalent in acute, cluster 3 in healthy, cluster 5 is ubiquitous, and cluster 9 is rare and in acute patients. Bottom row) A comparison of the cluster proportion for acute (X-axis) versus convalescent (Y-axis) for patients that have matched samples.

result, SAUCIE is able to process multisample data in a unified way using a single underlying representation learned by a deep autoencoder. Thus, different samples can be visualized in the same coordinates without batch effects via the embedding layer of the neural network, and cluster proportions can be directly compared, since the whole dataset is decomposed into a single set of clusters without requiring cluster matching or metaclustering. These unified representations can be readily used for inter-sample comparisons and stratification, on the basis of their underlying cell-to-cell heterogeneity.

Mathematically, SAUCIE presents a new way of utilizing deep learning in the analysis of biological and biomedical data by directly reading and interpreting hidden layers that are regularized in novel ways to understand and correct different aspects of data. Thus far, deep learning has primarily been used in biology and medicine as a black-box model designed to train classifiers that often mimic human classifications of disease or pathology. However, the network internal layers themselves are typically not examined for mechanistic understanding. SAUCIE provides a way of obtaining information from internal layers of a deep network. Deep autoencoding neural networks essentially perform nonlinear dimensionality reduction on the data. As such they could be used "off-the-shelf" for obtaining new coordinates for data in a reduced-dimension space, to which other algorithms can be applied. However, in SAUCIE we aim to go further to structure the reduced dimensions in specifically interpretable ways using novel regularizations. Our information-theoretic regularization encourages near-binary activations of an internal layer, thus making the layer amenable to directly output encoded cluster identifications. We believe that such regularizations add interpretability to layers in neural networks, thus turning these "black boxes" into "glass boxes."

The ability to stratify patients on the basis of their single-cell subpopulations, which can emerge as features in deep neural networks, can be key to a new generation of biomarkers that can be used in diagnosis and treatment. Traditionally, biomarkers are proteins or antibodies that are circulating in blood, which signals the presence of infection or other conditions. However, immune cells are highly plastic and can evolve or activate in specific ways in response to disease conditions in different patients. Here, we showcase the heterogeneity of immune cells in response to acute dengue infection in a large patient cohort. We see that specific subpopulations are enriched in the acute conditions, as opposed to convalescent or healthy controls. We showed with our dengue dataset it is possible discover cell populations, even rare ones that are indicative of patient and experimental conditions. Other datasets comprising of large patient cohorts measured at single-cell resolution are underway already in many hospitals and clinical trials. In the future, we are confident that this capability will be useful in many studies including immunotherapy, autoimmunity, and cancer, where there are immune subsets that emerge in response.

6.4 Methods

6.4.1 Computational Methods

In this section we explain the SAUCIE framework in greater detail including the philosophy behind using autoencoders for learning the cellular manifold, details of the regularizations used in different layers of SAUCIE to achieve particular data analysis tasks as well as training and implementation details. Finally, we discuss the emergent higher level organization of the patient manifold as a result of the cellular manifold of the subjects learned by SAUCIE.

Multitask manifold learning

A popular and effective approach for processing big high-dimensional data in genomics, as well as other fields, is to intuitively model the intrinsic geometry of the data as being sampled from a low dimensional manifold – this is commonly referred to as the manifold assumption [1]. This assumption essentially means that local regions in the data can be linearly mapped to low dimensional coordinates, while the nonlinearity and high dimensionality in the data comes from the curvature of the manifold. Typically, a notion of locality is derived from the data with nearest-neighbor search or adaptive kernels to define local neighborhoods that can approximate tangent spaces of the manifold. Then, these neighborhoods are either used directly for optimizing low dimensional embeddings (e.g., in TSNE [108] and LLE [133]), or they are used to infer a global data manifold by considering relations between them (e.g., using diffusion geometry [37, 120, 116, 163]).

The characterization of the intrinsic data geometry as a data manifold is also closely related to the underlying approach in SAUCIE. Indeed, neural networks can be considered as piecewise linear approximations of target functions [115]. In our case, we essentially approximate the data manifold coordinate charts and their inverse with the autoencoder architecture of SAUCIE. The encoder training identifies local patches and maps them to low dimensional coordinates, while sewing these patches together in this embedding to provide a unified visualization. The decoder learns the linear relation between these intrinsic coordinates and the tangent spaces of the manifold, positioned in the high dimension. This also results in a projection of data points on the manifold (via its tangent spaces), which creates a denoising effect similar to the diffusion-based one used recently in MAGIC [163]. Finally, the clustering layer in SAUCIE is trained to recognize and aggregate similar data regions to ensure an appropriate granularity (or resolution) of the identified neighborhoods and prevent excessive fragmentation of the manifold.

While tools using the scaffold of manifold learning have emerged for various tasks in single cell data analysis, there is currently no unified manifold model that provides all of the necessary tasks in a scalable fashion. For example, MAGIC [163] uses manifold learning to impute the data, but does not address embedding, visualization, or clustering. Diffusion pseudotime [57] provides an organization of the data to infer latent temporal structure and identifies trajectories, but it does not deal with imputation, clustering, or visualization. Furthermore, manifold learning methods do not work well across batches and typically just focus on single batches. Thus, their construction may suffer from batch effects and be dominated by the geometry between batches rather than their biology, as demonstrated by the example of Phenograph in Figure 6.16.

To address these shortcomings, SAUCIE performs all operations on a unified manifold geometry, which is learned implicitly by a deep multitasking neural network. It utilizes the scalability of deep learning to process high throughput data and construct a manifold that is jointly optimized for multiple tasks; namely, clustering, visualization, imputation, and batch correction. Therefore, the tasks themselves respect the manifold assumption and have the associated advantages, such as robustness to noise, while also agreeing with each other on a coherent underlying structure of the data.

SAUCIE architecture

SAUCIE consists of three encoding layers, an embedding layer, and then three decoding layers. The default number of neurons per hidden layer in the encoder used were 512, 256, and 128 with a symmetric decoder. The GMM dataset, being simpler, was clustered with layers of 50, 30, and 10. For batch correction, the best results were achieved with layer sizes of 1024, 512, and 256. The ID regularization was applied to the final decoder layer, which uses a ReLU. The two-dimensional embedding layer uses a linear activation, while all other layers use a leaky rectified linear activation with 0.2 leak. The coefficients λ_d and λ_c were chosen depending on the dataset, with the best values generally being λ_d twice λ_c. Their magnitude was guided by the effect of these two knobs on the granularity (shown in Figure 6.12). Training was performed with minibatches of 256, mean-squared-error for the reconstruction error function, and the optimizer chosen is ADAM with learning rate 0.001.

Batch correction and MMD Regularization

A major challenge in the analysis of single-cell data is dealing with so-called batch effects that result from technical variability between replicates of an experiment. Combining replicates often results in technical and experimental artifacts being the dominant source of variability in the data, even though this variability is entirely artificial. This experimental noise can come in the form of dropout, changes of scale, changes of location, or even more complicated differences in the distributions of each batch. It is infeasible to parametrically address all of the potential differences explicitly, for example, by assuming measurements are drawn from a Gaussian distribution. Instead of addressing specific explicit models of noise, SAUCIE minimizes a distance metric between distributions. The batch correction term L_b calculates the Maximal Mean Discrepancy (MMD) between batches, as

$$L_b = \Sigma_{i \neq ref} MMD(V_{ref}, V_i),$$

where V_{ref} is the visualization layer of one of the replicates, arbitrarily chosen to be considered as a reference batch. MMD compares the average distance from each point to any other point in its own batch, with the distance to points in the other batch. MMD is zero only when two distributions are equal. Thus minimizing this metric encourages SAUCIE to align the batches. MMD has been used effectively to remedy batch effects in residual networks, but here SAUCIE uses it in a feedforward autoencoder and combines it with other tasks of interest in biological exploratory data analysis [141].

The choice of reference does not affect the degree to which two distributions can be aligned, but a reference batch is necessary because the encoding layers of a standard network will be encouraged to embed different batches in different places in the visualization layer. It does this because the decoder is required to make its reconstruction $\hat{X}$ match the original data in X, which includes the batch effects. To remedy this, the decoder in SAUCIE is required to reconstruct the reference batch exactly as usual, but other batches must only be reconstructed to preserve the points normalized by mean and variance. Consequently, the MMD regularization term will be minimized when batches are aligned, and the decoder need only be able to reconstruct the exact values of the reference batch and the *relative values* of the non-reference batches. The non-reference batches will be aligned to the reference batch in a way that preserves their internal structure as best as possible.

Regularizations and Post-processing for clustering

Information Dimension Regularization We consider the task of clustering data points by interpreting the sparse layer B in the network as encoding cluster assignments. We note that a common activation function used to introduce nonlinearities in neural networks (including SAUCIE) is the Rectified Linear Unit (ReLU), and it provides a natural threshold for binarizing neuron activation to be either zero or one. These units are either “off” at or below zero or “on” for any positive value, so a small positive value ϵ can be used a threshold to binarize the activations in B. This results in an interpretable clustering layer that creates ‘digital’ cluster codes out of an ‘analog’ hidden layer, thus providing a binary code for each input point of the network. These binary codes are in turn used as cluster identifiers in order to group data points with the same code into a single cluster.

In order to automatically learn an appropriate granularity of clusters, we developed a novel regularization that encourages near-binary activations and minimizes the information (i.e., number of clusters) in the clustering layer. Our regularization is inspired by the von

Neumann (or spectral) entropy of a linear operator [10], which is computed as the Shannon entropy of their normalized eigenvalues. This entropy serves as a proxy for the numerical rank of the operator, and thus provides an estimation of the essential dimensionality of its range. In our case, we extend this notion to the nonlinear transformation of the neural network by treating neurons as our equivalent of eigenvalues, and computing the entropy of their total activation over a batch. We call this entropy 'information dimension' (ID) and the corresponding ID regularization aims to minimize this entropy while still encoding sufficient information to allow reconstruction of the input data points.

The ID regularization is computed from the clustering layer activations in B by first computing the activation of each neuron j as $a_j = \sum_{i=1}^{n} B_{ij}$, then normalizing these activations to form an activation distribution $\vec{p} = \vec{a}/\|\vec{a}\|_1$, and finally computing the entropy of this activation distribution as

$$L_c(B) = -\sum_{j=1}^{k} p_j \log p_j.$$

By penalizing the entropy of neuron activations, this regularization encourages a sparse and binary encoding. This counters the natural tendency of neural networks to maximize the amount of captured (i.e., encoded) information by spreading activations out across a layer evenly. By forcing the activations to be concentrated in just a few distinct neurons, different inputs end up being represented with rather similar activation patterns, and thus naturally clustered. When combined with the reconstruction loss, the network will retain enough information in the sparse layer for the decoder to reconstruct the input, keeping similar points in the same cluster.

Intracluster distance regularization The digital codes learned by SAUCIE create an opportunity to interpret them as clusters, but these clusters would not necessarily be comprised of only similar points. To emphasize that inputs only be represented by the same digital code if they are similar to each other, SAUCIE also penalizes intracluster pairwise distances. Beyond suffering reconstruction loss, using the same code for points that are far away from each other will now incur an even greater loss.

This loss is calculated as the euclidean distance between points with the same binary code:

$$L_d(B, \hat{X}) = \sum_{i,j:b_i=b_j} \|\hat{x}_i - \hat{x}_j\|^2$$

where $\hat{x}_i, \hat{x}_j$ and b_i, b_j are the i-th and j-th rows of $\hat{X}$ and B, respectively.

Since ID regularization is minimized by using the same code to represent all inputs, this term acts as an opposing balance. Intracluster distances are minimized when all points are in a cluster by themselves. Together with the reconstruction penalty, these terms encourage SAUCIE to learn clusters that are composed of as many points as possible that are near to each other.

An additional benefit of clustering via regularization is that not only is the number of clusters not needed to be set *a priori*, but by changing the value of λ_c the level of granularity of the clustering can be controlled, so both coarse clustering and fine clustering can be obtained to further add insight into the underlying structure of the data.

Cluster merging As the binarized neural network may not converge to the ideal level of granularity due to the many possible local optima in the loss landscape, we process the SAUCIE clustering with a cluster merge step to fix the ideal level of granularity everywhere. The cluster merging is performed by calculating MMD between clusters in the SAUCIE latent space and merging all clusters $i, j \in C$, where C is the set of all clusters, such that both of the following equations hold

$$\underset{\xi \in C}{\text{argmin}} MMD(i, \xi) = j \tag{6.2}$$

$$\underset{\xi \in C}{\text{argmin}} MMD(j, \xi) = i \tag{6.3}$$

This merging finds clusters that would be a single cluster in another granularity and fixes them to a single cluster.

Patient Manifold Visualization

In addition to the cell-level manifold constructed by SAUCIE, we also consider the geometry between samples to provide a coarser patient-level manifold. We construct and embed this manifold in low dimensions by applying kernel-PCA (kPCA) with an RBF kernel to the metric space defined by MMD distances between subjects. This augments the analysis SAUCIE provides of the biological variations identified in the cell space with an analysis of the variation in the patient space. Normally, without batch correction, the two sources of variation would be confounded, and batch effects would prevent clear analysis at either level (patient or cell) across batches. With our approach here we are able to separate them to provide on one hand, a stable (batch-invariant) cell-level geometry by the SAUCIE embedding, and on the other hand, a robust patient geometry provided by kPCA embedding. The patient geometry then allows us to recover patient-level differences and utilize them further for data exploration, in conjunction with the cell-level information. For example, as Figure 6.13A shows, we have a notable stratification between the acute and non-acute subjects. There is also a noticeable difference between the convalescent subjects and the acute, albeit a less drastic one than the difference between acute subjects and the others.

Training

To perform multiple tasks, SAUCIE uses a single architecture as described above, but is run and optimized sequentially. The first run imputes noisy values and corrects batch effects in the original data. This preprocessed data is then run through SAUCIE again to obtain a visualization and to pick out clusters. The different runs are done by optimizing different objective functions. In the following, we describe the optimization of each run over a single batch of n data points. However, the full optimization of each run independently utilizes multiple (mini-)batches in order to converge and minimize the described loss functions.

For the first run, formally let X be an $n \times d$ input batch, where each row is a single data point, and d is the number of features in the data. It is passed through a cascade of encoding linear and nonlinear transformations. Then, a cascade of decoding transformations reconstruct the denoised batch $\hat{X}$, which has the same dimensions as the input X and is optimized to reconstruct it.

For the next run, the cleaned batch $\hat{X}$ is passed through encoding transformations and a visualization layer denoted by $V \in \mathbb{R}^{n \times 2}$. We also consider a clustering layer in another run where the decoder outputs near-binary activations $B \in \mathbb{R}^{n \times d_B}$, where d_B is the number of hidden nodes in the layer, which will be used to encode cluster assignments, as described below. The activations in B are then passed to the reconstruction $\tilde{X}$ that has the same dimensions as $\hat{X}$ (and X) and is optimized to reconstruct the cleaned batch.

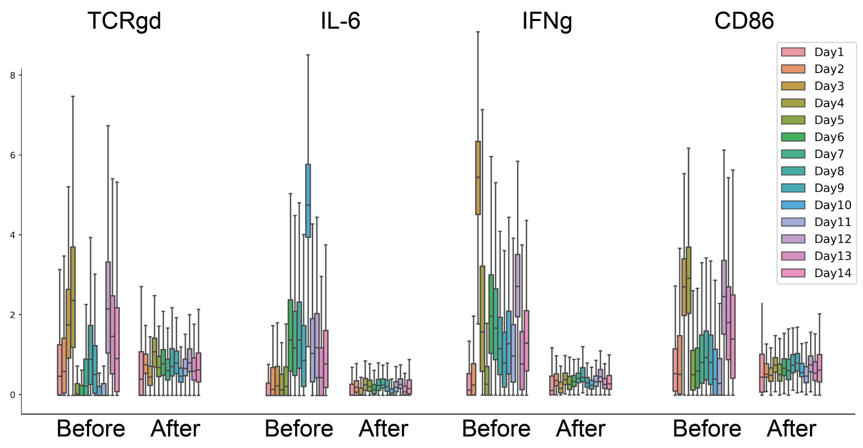
The loss function of all runs starts with a reconstruction loss L_r forcing the autoencoder to learn to reconstruct its input at the end. SAUCIE uses the standard mean-squared error loss (i.e., $L_r(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$, where x_i and $\hat{x}_i$ are the i-th row of X and $\hat{X}$ correspondingly). We note that while MSE is a standard and effective choice in general, other loss functions can also be used here as application-specific substitutes that may be more appropriate for particular types of data. For the first run, we add to this loss a regularization term L_b that enables SAUCIE to perform batch correction. This regularization is computed from the visualization layer to ensure consistency across subsampled batches. The resulting total loss is then

$$L = L_r(X, \hat{X}) + \lambda_b \cdot L_b(V).$$

The loss function of the clustering run then optimizes L_r along with two regularization terms L_c and L_d that together enable SAUCIE to learn clusters:

$$L = L_r(\hat{X}, \tilde{X}) + \lambda_c \cdot L_c(B) + \lambda_d \cdot L_d(B, \hat{X}).$$

The first term L_c guides SAUCIE to learn binary representations via the activations in B using a novel information dimensionality penalty that we introduce in this paper. The second term L_d encourages interpretable clusters that contain similar points by penalizing intra-cluster distances in the cleaned batch $\hat{X}$, which is fixed for this run.



suppfig:batcheffect3

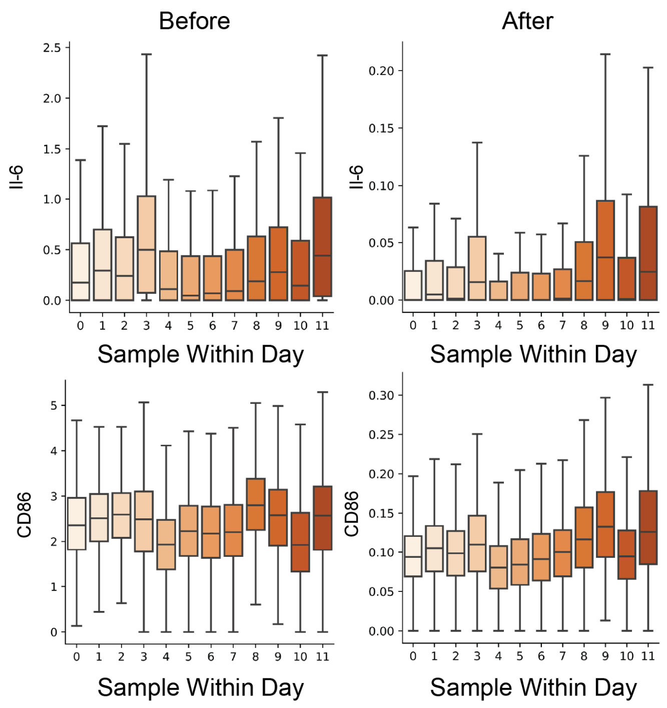
Figure 6.14: Four select marker abundances with samples grouped by day they were run on the cytometry instrument, with each day having fourteen distinct samples in the group. For each marker, the fourteen samples before batch correction are shown to the left of the same fourteen samples after batch correction.

Dengue dataset batch correction

Beyond the sheer size of the total dataset, due to the large number of distinct samples in the experiment there are significant batch related artifacts effects, stemming from day-to-day differences, instruments, handling and shipping of the samples. While there are true biological differences between the individual samples, to identify those true differences in the samples we have to remove differences that are caused by these technical variables.

Differences that are highly associated with the day the samples were run on the cytometry instrument can be seen by grouping all of the samples together by run day and examining their marker-by-marker abundances. Each run day has twelve samples chosen such that each day has samples from each experimental condition, so any differences between the samples from each day are batch effects. As shown in Figure 6.6, these difference exist in the spike-in controls as well as the samples, confirming their identity as batch effect and not true variation.

Figure 6.14 shows four markers with extreme batch effects: TCRgd, IL-6, IFNg, and CD86. These batch effects would normally mean only samples within each run day could be compared to each other, as comparisons between samples from different run days would be dominated by the differences in the run days. Instead, the SAUCIE batch correction removes these undesirable effects by combining the samples from each day and aligning them



suppfig:batcheffect1

Figure 6.15: Histograms of marker expression (top: IL-6, bottom: CD86) of samples run together on the cytometry instrument on day two, separated by sample. The values for each sample and marker are shown before SAUCIE batch correction (left) and after SAUCIE batch correction (right).

to a reference batch, here chosen to be Day 1. Figure 6.14 shows that after SAUCIE the differences between run days disappear so that now what it means to be low or high in a marker is the same for each day. Before, the cells with the lowest IFNg in samples from Day 3 would still be considered IFNg+ while the cells with the highest IFNg in samples from Day 1 would still be IFNg-. After batch correction with SAUCIE, these can be directly compared.

The challenge of batch correction is to remove differences due to artifacts while preserving biological differences. We reason that to prevent removing true biological variation, the 'shape' of the data (but not its position and scale) within each day must be preserved. We define the shape of the data as any moment beyond the first two - mean and variance. We examine this in detail by considering a run day with the most significant batch effects, Day 2. In Figure 6.6C, the SAUCIE visualization shows that the reference and nonreference batches are completely separated. When MMD regularization is added in SAUCIE, though, these two batches are fully overlapped. In Figure 6.15, we examine the twelve individual samples that were run on Day 2. Initially, we see that this confirms our idea that the differences between days are batch effects, because each sample measures high in IL-6 and CD86. So the differences between samples run on Day 1 and Day 2 in CD86 abundance is not dominated by having more of a certain sample type in Day 2. Instead, all samples in Day 2 have been shifted higher. As desired, after batch correction, the mean of each marker is reduced to the level of the reference-batch mean. Crucially, the relationship of samples in Day 2 relative to each other is preserved. The samples with the highest IL-6 in Day 2 are still Samples 3, 9, and 11 while the samples with the lowest are still Samples 4, 5, and 6. SAUCIE has just changed what it means to be high or low for samples in this day such that it reconciles what it means to be high or low for samples in the reference day.

Comparison to Phenograph

We next compare the SAUCIE pipeline of batch correcting, clustering, and visualizing single-cell data from a cohort of subjects to an alternative approach called metaclustering [92]. We first cluster each sample individually with Phenograph. Then, we represent each cluster as its centroid and use Phenograph again on the clusters to obtain metaclusters. We examine the pipelines on ten of the 180 samples here, where the metaclustering approach took forty minutes. We note that the SAUCIE pipeline took *45 minutes* to process all 180 samples, while the metaclustering approach would take *12 hours* to process all of them. Figure 6.16 shows tSNE embeddings of the cluster centroids where the size of the cluster is proportional to the size of the point. Coloring by sample, we see that the metaclusters have identified batch effects. Metacluster 0 is only composed of samples 1, 3, 4, and 5. These samples have no clusters in any other metacluster, and none of the other samples have any cluster in this metacluster. Examining the gene expression heatmap, we see that metacluster 0 has separated cells with high CD86 values, which were shown earlier to be batch effects. Moreover, the metaclusters are very heterogeneous internally with respect to gene expression. This is a results of metaclustering the cluster centroids, as the metaclusters then have no information about the individual cells comprising that centroid.

In contrast, Figure 6.17 shows the SAUCIE pipeline on these ten samples. The cluster

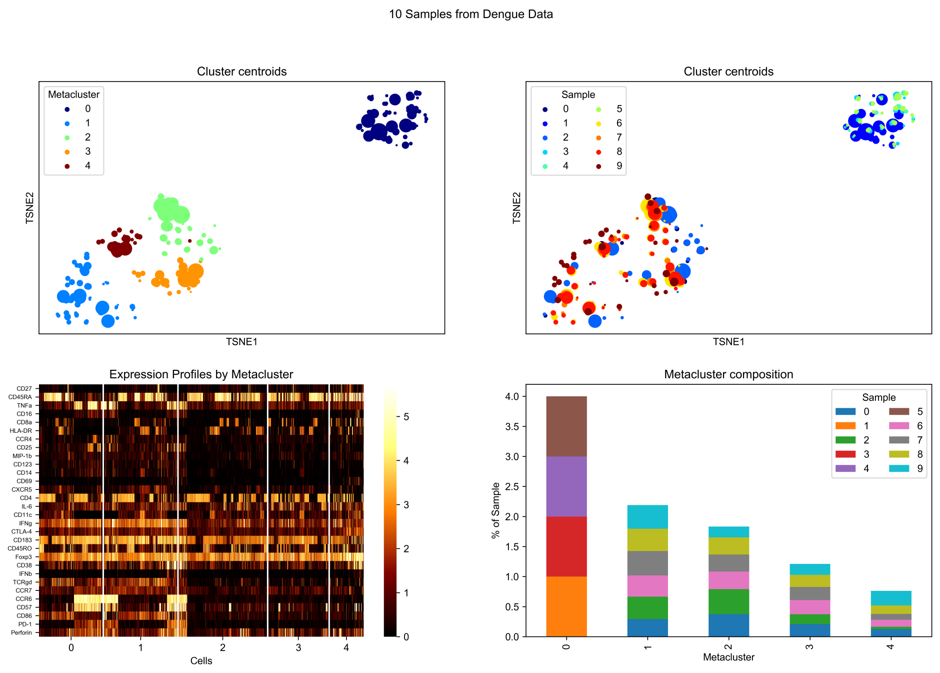


Figure 6.16: An illustration of the metaclustering process on the dengue dataset. Top left: cluster centroids embedded by tSNE and colored by metacluster, sized according to the number of cells in each cluster. Top right: cluster centroids colored by sample, also sized according to the number of cells in each cluster. Bottom left: a cell-level heatmap of expression grouped by metacluster. Bottom right: the composition of each metacluster by sample.

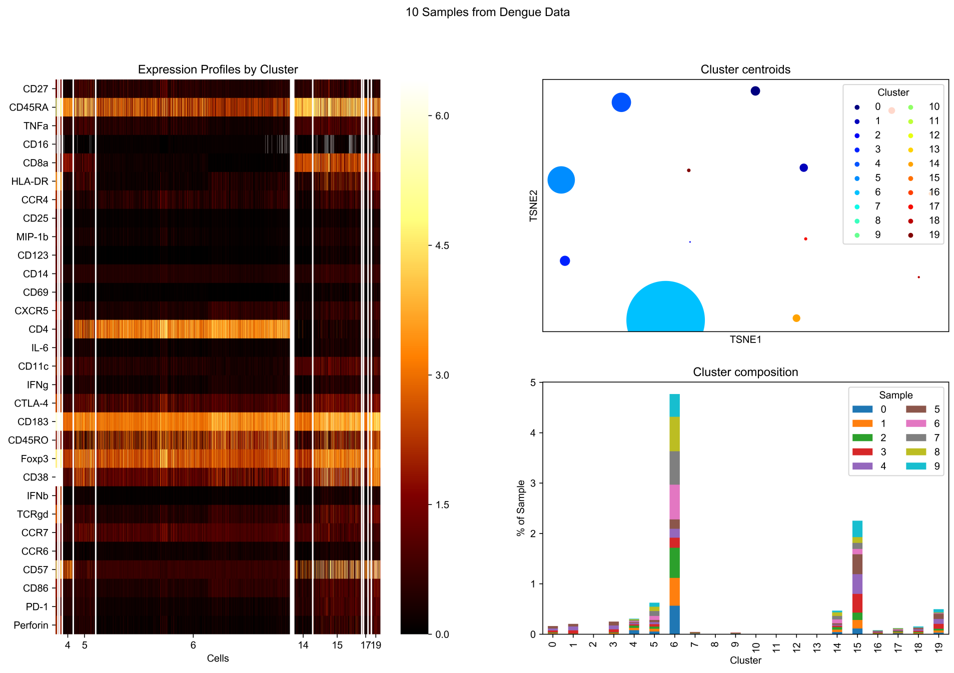


Figure 6.17: An illustration of the SAUCIE pipeline on the dengue dataset. Left: cell-level heatmap of expression grouped by cluster. Top right: cluster centroids embedded by tSNE, sized according to the number of cells in each cluster. Bottom right: the composition of each cluster by sample.

proportions show that each cluster is fully mixed with respect to the samples, as opposed to the sample-segregated metaclusters of the previous approach. Similarly, the clusters are more homogeneous internally, meaning they actually keep similar cells together, as opposed to the metaclusters, which lost this information when each cluster was represented by only its centroid. Finally, we find that SAUCIE effectively compares cells across subjects, while the metaclustering approach still fails at patient-to-patient comparisons, instead only identifying batch effect variation. This emphasizes the importance of multitask learning using a unified representation in SAUCIE.

Runtime Comparison Methodology

For each visualization, clustering, and imputation method, the dataset of size N was given to the method as input and returned the appropriate output. For batch correction, the dataset of size N was divided into two equal-sized batches that were corrected. For the methods that operated on minibatches, minibatches of size 128 were used. For the methods that train by stochastic gradient descent, the number of steps was determined by taking the total number of points and dividing by the size of the minibatch, so that a complete pass through the entire dataset was performed. In order to return clusters, the latent space of scVI must be clustered by another method, and since the number of clusters is not known ahead of time, the fastest method that does not require this to be known (Phenograph) was used. For SAUCIE, batch correction, imputation, clustering, and visualization were all produced in the timed run. All computations were performed on a single machine with 16 CPU cores and a GeForce GTX 1080 GPU.

Number of Clusters

As discussed earlier, the number of clusters resulting from SAUCIE is not specified in advance, but dictated by the structure of the data that the model discovers, and by the choice of regularization coefficients λ_d and λ_c. For a given value of λ_d, as λ_c increases, the number of clusters decreases. Increasing λ_d, on the other hand, increases the number of clusters (Figure 6.12). This is because λ_c penalizes entropy in the activations of the n neurons in the clustering layer of the network. While entropy can be initially decreased by making all n neurons either 0 or 1, it can be further decreased by making all n neurons 0. Thus, as this term is considered more influential in the total loss, in the extreme, all points can be mapped to the same binary code. In contrast, λ_d penalizing intra-cluster distances, so this value can be decreased by making clusters smaller and smaller (and thus getting more of them). In the extreme for this term, every point can be made its own cluster and intra-cluster distances would decrease to 0. By balancing these two, the desired granularity of clustering can be obtained from SAUCIE. In our experiments, we find making λ_d to be between two and three times larger than λ_c, with values around 0.2 generally results in medium coarse-grained clustering. Another consideration that affects the number of clusters is the number of neurons in the clustering layer. We found varying this number does not improve performance and for all experiments here we use a fixed size of 256 neurons.

6.4.2 Experimental methods

Study Subjects

Dengue patients and healthy volunteers were enrolled with written informed consent under the guidelines of the Human Investigations Committees of the NIMHANS and Apollo Hospital, and Yale University [187]. The Human Investigations Committee of each institution approved this study. Patients with dengue virus infection were defined as dengue fever using WHO-defined clinical criteria, and/or laboratory testing of viral load or serotyping at the time of infection. Healthy volunteers included household contacts of dengue patients present in the same endemic area. Participants were of both genders (26.7% female) and were all of Indian heritage. Subjects from the symptomatic and healthy groups were not statistically different for age, gender, or race in this study.

Sample Collection and Cell Isolation

Heparinized blood was collected from patients and healthy volunteers and employed a 42 marker panel of metal conjugated antibodies following methods previously described [180, 179]. Purification of peripheral blood mononuclear cells (PBMCs) was performed by density-gradient centrifugation using Ficoll-Paque (GE Healthcare) according to the manufacturer's instructions following isolation and cryopreservation guidelines established by the Human Immunology Phenotyping Consortium. PBMCs for CyTOF were frozen in 90% FBS containing 10% DMSO and stored in liquid N2 for shipping following the guidelines of the DBT. Samples for this study were received in three shipments and viability was average 85% (range $50 - 98$) across the dates.

Mass Cytometry Acquisition

For mass cytometry at Yale University, PBMCs (5 x 106 cells/vial) were thawed incubated in Benzonase (50U/ml) in RPMI/10% human serum, and seeded in 96-well culture plate (6 x 103-1.2 x 106 cells/well. Monensin (2mM, eBioscience) and Brefeldin A (3mg/ml, eBioScience) added for the final 4 h of incubation for all groups. Groups of samples (8-13/day) were infected in vitro per day on 5 separate days and included a CD45-labeled spike-in reference sample in every sample. Surface markers were labeled prior to fixation and detailed staining protocols have been described. Briefly, cells were transferred to 96-well deep well plates (Sigma), resuspended in 25 mM cisplatin (Enzo Life Sciences) for one minute, and quenched with 100% FBS. Cells were surface labeled for 30 min on ice, fixed (BD FACS Lyse), and frozen at -80°C. Intracellular labeling was conducted on batches of cells (12/day). Fixed PBMCs were permeabilized (BD FACS Perm II) for labeling with intracellular antibodies for 45 min on ice. Cells were suspended overnight in iridium interchelator (125 nM; Fluidigm) in 2% paraformaldehyde in PBS and washed 1X in PBS and 2X in H2O immediately before acquisition. A single batch of metal-conjugated antibodies was used throughout for labeling panels. Metal-conjugated antibodies were purchased from Fluidigm, Longwood CyTOF Resource Core (Cambridge, MA), or carrier-free antibodies were conjugated in house using MaxPar X8 labeling kits according to manufacturer's instructions (Fluidigm). A total of 180 samples were assessed by the Helios (Fluidigm) on 15 independent experiment dates using a flow rate of 0.03 ml/min in the presence of EQ Calibration beads (Fluidigm) for normalization. An average of $112,537 \pm 71,444$ cells (mean $\pm$ s.d.) from each sample were acquired and analyzed by CyTOF. Data was preprocessed with the hyperbolic sine transformation. Additional experimental details will be given in [187].

Chapter 7

Neuron Editing

7.1 Introduction

While generative neural networks can learn to transform a specific input dataset into a specific target dataset, they require having just such a paired set of input/output datasets. For instance, to fool the discriminator, a generative adversarial network (GAN) exclusively trained to transform images of black-haired *men* to blond-haired *men* would need to change gender-related characteristics as well as hair color when given images of black-haired *women* as input. This is problematic, as often it is possible to obtain *a* pair of (source, target) distributions but then have a second source distribution where the target distribution is unknown. The computational challenge is that generative models are good at generation within the manifold of the data that they are trained on. However, generating new samples outside of the manifold or extrapolating "out-of-sample" is a much harder problem that has been less well studied. To address this, we introduce a technique called *neuron editing* that learns how neurons encode an edit for a particular transformation in a latent space. We use an autoencoder to decompose the variation within the dataset into activations of different neurons and generate transformed data by defining an editing transformation on those neurons. By performing the transformation in a latent trained space, we encode fairly complex and non-linear transformations to the data with much simpler distribution shifts to the neuron's activations. Our technique has the advantage of being generally applicable to a wide variety of data domains, modalities, and applications. We first demonstrate it on image transformations and then move to our two main applications in biology: removal of batch artifacts representing unwanted noise and modeling the effect of drug treatments to predict synergy between drugs.

Many experiments in biology are conducted to study the effect of a treatment or a condition on a set of samples. For example, the samples can be groups of cells and the treatment can be the administration of a drug. However, experiments and clinical trials are often performed on only a small subset of samples from the entire population. Usually, it is assumed that the effects generalize to all of the samples in a context-independent manner, potentially conflating sample-specific variation with treatment-induced variation. Mathematically modeling both sources of variation and isolating the treatment-induced parts provides an opportunity to improve generalization beyond the samples measured.

We propose a neural network-based method for learning a general edit function corresponding to treatment in the biological setting. Popular neural network architectures like GANs pose the problem as one of learning to *output* data in the region of the space occupied by the target distribution, no matter where the input data is coming from. To fool the discriminator, the generator's output must end up in the same part of the space as the target distribution. The discriminator does not take into account the input points into the generator in any way.

Instead, we reframe the problem as learning a transformation *towards* the target distribution that is more sensitive to where the input data starts. Thus, we could learn an edit between pre- and post- treatment samples on one patient and apply it to pre-treatment samples of another patient without also modeling patient-specific characteristics that were present both pre- and post-treatment.

We propose to learn such an edit, which we term *neuron editing*, in the latent space of an autoencoder neural network with non-linear activations. First we train an autoencoder on the entire population of data which we are interested in transforming. This includes all of the pre-treatment samples and the post-treatment samples from the subset of the data on which we have post-treatment measurements. Neuron editing then involves extracting differences between the observed pre-and post-treatment activation distributions for neurons in this layer and then applying them to pre-treatment data from the rest of the population to synthetically generate post-treatment data. Thus performing the edit node-by-node in this space actually encodes complex multivariate edits in the ambient space, performed on denoised and meaningful features, owing to the fact that these features themselves are complex non-linear combinations of the input features.

Neuron editing is a general technique that could be applied to the latent space of any neural network, even GANs themselves. We focus exclusively on the autoencoder in this work, however, to leverage its denoising ability, robustness to mode dropping, and superior training stability as compared to GANs. We demonstrate that neuron editing can work on a variety of architectures, while offering the advantages of introducing no new hyperparameters to tune and being stable across multiple runs.

While latent space manipulation has been explored in previous work, ours differs in several ways. For example, [129] represents a transformation between two distributions as a single constant shift in latent space. In addition to assuming the latent transformation is the same for all points in the distribution, [161] also uses an off-the-shelf pre-trained Imagenet classifier network. Our work, on the other hand, does not require a richly supervised pre-trained model; also, we model the shift between two distributions as a complex, non-constant function that learns different shifts for different parts of the space. We compare to this ”constant-shift” approach and demonstrate empirically why it is necessary to model the transformation more complexly.

By performing the edit to the neural network internal layer, we allow for the modeling of some context dependence. Some neurons are less heavily edited but still influence the output jointly with edited neurons due to their integration in the decoding layers, propagating their effect into the output space.

We note that neuron editing makes the assumption that the internal neurons have semantic consistency across the data, i.e., the same neurons encode the same types of features for every data manifold. We demonstrate that this holds in our setting because we choose to let the autoencoder learn a joint manifold of all of the given data, including pre- and post-treatment samples of the experimental subpopulation and pre-treatment samples from the rest of the population. Recent results show that neural networks prefer to learn patterns over memorizing inputs even when they have the capacity to do so [183].

We demonstrate that neuron editing extrapolates better than generative models on two important criteria. First, as to the original goal, the predicted change on extrapolated data more closely resembles the predicted change on interpolated data. Second, the editing process produces more complex variation, since it simply preserves the existing variation in the data rather than needing a generator to learn to create it. We compare to standard GAN approaches, dedicated parametric statistical methods used by computational biologists, and alternative autoencoder frameworks. In each case, we see that they stumble on one or more of several hurdles: out-of-sample input, desired output that differs from the target of the training data, and data with complex variation.

In the following section, we detail the neuron editing method. Then, we motivate the extrapolation problem by trying to perform natural image domain transfer on the canonical CelebA dataset [99]. We then move to two biological applications where extrapolation is essential: correcting the artificial variability introduced by measuring instruments (batch effects), and predicting the combined effects of multiple drug treatments (combinatorial drug effects) [11].

7.2 Model

Let $S \in \mathbb{R}^{n_S \times d}, T \in \mathbb{R}^{n_T \times d, X \in \mathbb{R}^{n_X \times d}}$ represent d-dimensional source, target, and second source distributions with n_S, n_T, and n_X observations, respectively. We seek a transformation such that: 1. when applied to S it produces a distribution equivalent to T 2. when applied to T it is the identity function and 3. when applied to X it does not necessarily produce T if S is different from X. While GANs learn a transformation with the first two properties, they fail at the third property due to the fact that T is the only target data we have for training, and thus the generator only learns to output data like T. Therefore, instead of learning such a transformation parameterized by a neural network, we learn a simpler transformation on a *space learned by a neural network* (summarized in Figure 7.1).

We first train an encoder/decoder pair E/D to map the data into an abstract neuron space decomposed into high-level features such that it can also decode from that space, i.e.,

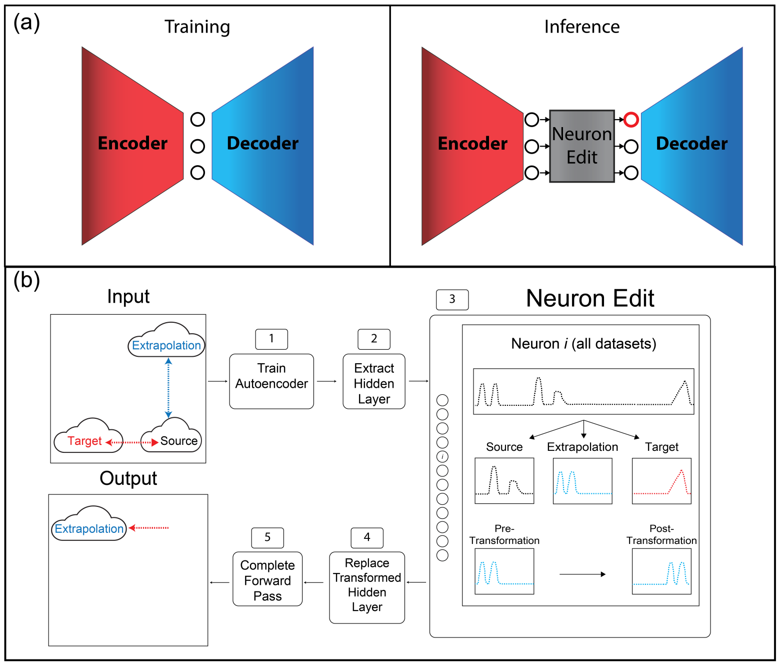


Figure 7.1: (a) Neuron editing interrupts the standard feedforward process, editing the neurons of a trained encoder/decoder to include the source-to-target variation, and letting the trained decoder cascade the resulting transformation back into the original data space. (b) The neuron editing process. The transformation is learned on the distribution of neuron activations for the source and applied to the distribution of neuron activations for the extrapolation data.

the standard autoencoder objective L:

$$L(S,T,X) = \text{MSE}\left[(S,T,X), D(E(S,T,X))\right]$$

where MSE is the mean-squared error. The autoencoder is trained on all three data distributions S, T, and X and thus learns to model their joint manifold. Then, without further training, we separately extract the activations of an n-dimensional internal layer of the network for inputs from S and from T, denoted by $a_S : S \to \mathbb{R}^n, a_T : T \to \mathbb{R}^n$. We define a piecewise linear transformation, called $NeuronEdit$, which we apply to these distributions of activations:

$$NeuronEdit(a) = \left(\frac{a - p_j^S}{p_{j+1}^S - p_j^S} \cdot (p_{j+1}^T - p_j^T) \right) + p_j^T \tag{7.1}$$

where $a \in \mathbb{R}^n$ consists of n activations for a single network input, $p_j^S, p_j^T \in \mathbb{R}^n$ consist of the j^{th} percentiles of activations (i.e., for each of the n neurons) over the distributions of a_S, a_T correspondingly, and all operations are taken pointwise, i.e., independently on each of the n neurons in the layer. Then, we define $NeuronEdit(a_S) : S \to \mathbb{R}^n$ given by $x \mapsto NeuronEdit(a_S(x))$, and equivalently for a_T and any other distribution (or collection) of activations over a set of network inputs. Therefore, the $NeuronEdit$ function operates on distributions, represented via activations over network input samples, and transforms the input activation distribution based on the difference between the source and target distributions (considered via their percentile disctretization).

We note that the $NeuronEdit$ function has the three properties we stated above:

1. $NeuronEdit(a_S) \approx a_T$ (in terms of the represented n-dimensional distributions)

2. $NeuronEdit(a_T) = a_T$

3. $NeuronEdit(a_X) = NeuronEdit(a_S) \implies a_X = a_S$

.

This last property is crucial since learning to generate distributions like T, with a GAN for example, would produce a discriminator who encourages the output to be funneled as close to T as posssible no matter where in the support we start from.

To apply the learned transformation to X, we first extract the activations of the internal layer computed by the encoder, a_X. Then, we edit the activations with the neuron editing function $\hat{a}_X$. Finally, we cascade the transformations applied to the neuron activations through the decoder without any further training. Thus, the transformed output $\hat{X}$ is obtained by:

$$\hat{X} = D(NeuronEdit(E(X)))$$

We emphasize that at this point, since we do no further training of the encoder and decoder, and since the neuron editing transformation has no weights to learn, there is no further objective term to minimize at this point and the transformation is fully defined.

Crucially, the nomenclature of an *autoencoder* no longer strictly applies. If we allowed the encoder or decoder to train with the transformed neuron activations, the network could learn to undo these transformations and still produce the identity function. However, since we freeze training and apply these transformations exclusively on inference, we turn an autoencoder into a generative model that need not be close to the identity.

Training a GAN in this setting could exclusively utilize the data in S and T, since we have no real examples of the output for X to feed to the discriminator. Neuron editing, on the other hand, is able to model the variation intrinsic to X in an unsupervised manner despite not having real post-transformation data for X. Since we know *a priori* that X will differ substantially from S, this provides significantly more information.

Furthermore, GANs are notoriously tricky to train [137, 55, 172]. Adversarial discriminators suffer from oscillating optimization dynamics [95], uninterpretable losses [17, 13], and most debilitatingly, mode collapse [147, 80, 121]. Under mode collapse, significant diversity that should exist in the output of the generator is lost, instead producing synthetic data that is a severely degenerated version of the true target distribution.

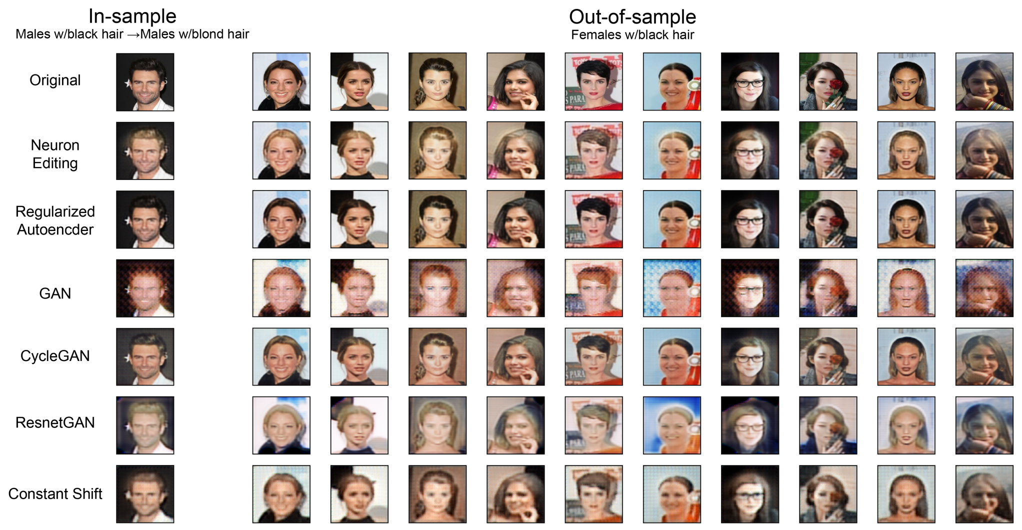
Neuron editing avoids all of these traps by learning an unsupervised model of the data space with the easier-to-train autoencoder. The essential step that facilitates generation is the isolation of the variation in the neuron activations that characterizes the difference between source and target distributions.

There is a relationship between neuron editing and the well-known word2vec embeddings in natural language processing [51]. There, words are embedded in a latent space where a meaningful transformation such as changing the gender of a word is a constant vector in this space. This vector can be learned on one example, like transforming *man* to *woman*, and then extrapolated to another example, like *king*, to predict the location in the space of *queen*. Neuron editing is an extension in complexity of word2vec's vector arithmetic, because instead of transforming a single point into another single point, it transforms an entire distribution into another distribution.

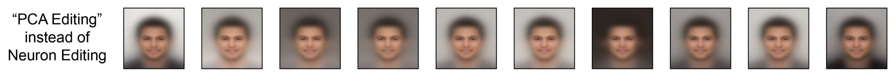
7.3 Experiments

We compare the predictions from neuron editing to those of several generation-based approaches: a traditional GAN, a GAN implemented with residual blocks (ResnetGAN) to show generating residuals is not the same as editing [150], and a CycleGAN [189]. While in other applications, like natural images, GANs have shown an impressive ability to generate plausible individual points, we illustrate that they struggle with these two criteria. We also motivate why neuron editing is performed on inference by comparing against a regularized autoencoder that performs the internal layer transformations during training, but the decoder learns to undo the transformation and reconstruct the input unchanged [6]. Lastly, we motivate why the more complex neuron editing transformation is necessary by comparing against a naive "latent vector arithmetic" approach. We find the constant vector between the mean of the source and the mean of the target in the internal layer of our pre-trained autoencoder, and apply this single shift to all neurons in the target (Constant Shift).

For the regularized autoencoder, the regularization penalized differences in the distributions of the source and target in a latent layer using maximal mean discrepancy [6, 46]. The image experiment used convolutional layers with stride-two filters of size four, with 64-128-256-128-64 filters in the layers. All other models used fully connected layers of size 500-250-50-250-500. Leaky ReLU activation was used with 0.2 leak. Training was done with minibatches of size 100, with the Adam optimizer [81], and learning rate 0.001.



(a)



(b)

Figure 7.2: Data from CelebA where the source data consists of males with black hair and the target data consists of males with blond hair. The extrapolation is then applied to females with black hair. (a) A comparison of neuron editing against other models. Only neuron editing successfully applies the blond hair transformation. (b) An illustration that neuron editing must be applied to the neurons of a deep network, as opposed to principle components.

CelebA	Neuron Editing	GAN	CycleGAN	ResnetGAN	RegAE	Constant Shift
FID	**121.63 +/- 2.12**	282.26 +/- 13.32	153.03 +/- 6.55	184.31 +/- 9.71	272.12 +/- 1.10	320.97 +/- 1.04

Table 7.1: FID scores on the CelebA extrapolation task.

7.3.1 CelebA Hair Color Transformation

We first consider a motivational experiment on the canonical image dataset of CelebA [99]. If we want to learn a transformation that turns a given image of a person with black hair to that same person except with blond hair, a natural approach would be to collect two sets of images, one with all black haired people and another with all blond haired people, and teach a generative model to map between them. The problem with this approach is that the learned model may perform worse on input images that differ from those it trained on. This has troubling consequences for the growing concern of socially unbiased neural networks, as we would want model performance to go unchanged for these different populations [156].

This is illustrated in Figure 7.2a, where we collect images that have the attribute male and the attribute black hair and try to map to the set of images with the attribute male and the attribute blond hair. Then, after training on this data, we extrapolate and apply the transformation to females with black hair, which had not been seen during training. The GAN models are less successful at modeling this transformation on out-of-sample data. In the parts of the image that should stay the same (everything but the hair color), they do not always generate a recreation of the input. In the hair color, only sometimes is the color changed. The regular GAN model especially has copious artifacts that are a result of the difficulty in training these models. This provides further evidence of the benefits of avoiding these complications when possible, for example by using the stable training of an autoencoder and editing it as we do in neuron editing.

We quantify the success of neuron editing by using the common metric of Frechet Inception Distance (FID) that measures how well the generated distribution matches the distribution targeted for extrapolation. These scores are reported in Table 7.1, where we see neuron editing achieve the best result on an average of three runs. Notably, due to the autoencoder's more stable training, the standard deviation across multiple runs is also lower than the GAN-based methods.

In Figure 7.2b, we motivate why we need to perform the $NeuronEdit$ transformation on the internal layer of a neural network, as opposed to applying it on some other latent space like PCA. Only in the neuron space has this complex and abstract transformation of

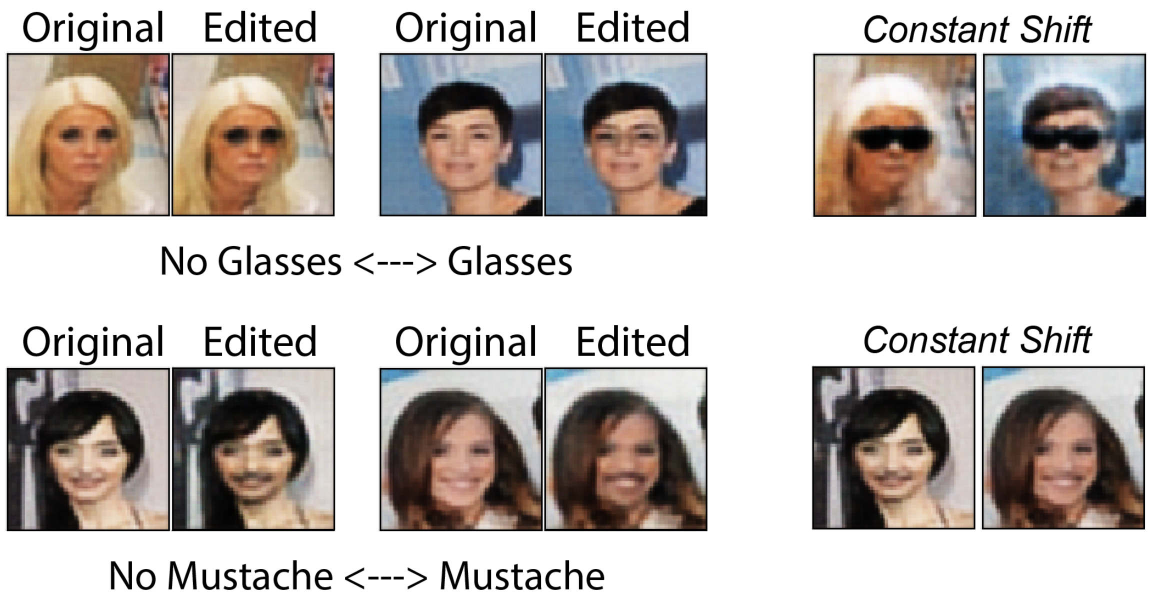


Figure 7.3: Additional CelebA transformations.

changing the hair color (and only the hair color) been decomposed into a relatively simple and piecewise linear shift.

Beyond hair color transformation, neuron editing is able to learn general transformations on CelebA males and apply them to females. In Figure 7.3, we learn to transform between having/not having the mustache attribute and having/not having the glasses attribute. The latter transformation on glasses demonstrates the importance of learning a non-constant transformation. The glasses attribute is bimodal, with both examples of sunglasses and reading glasses in the dataset. With neuron editing, we are able to learn to map to each of these different parts of the latent space, as opposed to the constant shift which adds dark sunglasses to the entire distribution.

7.3.2 Batch correction by out-of-sample extension from spike-in samples

We next demonstrate another application of neuron editing's ability to learn to transform a distribution based on a separate source/target pair: biological batch correction. Many biological experiments involve using an instrument to measure different populations of cells and then characterizing the features that distinguish between them. However, these complex instruments can be difficult to calibrate and use consistently, and thus can introduce technical artifacts into the data they are used to measure. In fact, we can even measure the same population of cells twice and get two very different datasets back. When we measure *different* populations, these technical artifacts (batch effects) get confounded with the true differences between the populations. Batch effects are a ubiquitous problem in biological experimental data can lead to incorrect conclusions in downstream analysis. Addressing batch effects is a goal of many new models [48, 160, 25, 58], including some deep learning methods [142, 6].

One method for grappling with this issue is to repeatedly measure an unvarying control (called a spike-in) set of cells with each population of interest (called a sample) [15]. Because we know any observed differences in the spike-in are technical artifacts, we can model and then remove this artifact in the population of interest. In our previous terminology, the two spike-in distributions are our known source/target pair while the actual population of interest is our second source that lacks a known target.

Existing methods of batch correction based on spike-ins work directly in the data space, operate independently on each dimension, and only do crude matching of distribution statistics. The most common approach is to simply subtract the difference in means between the spike-ins from the sample. We believe this is natural opportunity for deep learning, where the same concept can be extended to an abstract feature space, composed of combinations of features, and a more powerful transformation. Moreover, we expect neuron editing to shine as the spike-ins likely differ drastically from the sample.

The dataset we investigate in this section comes from a mass cytometry [16] experiment which measures the amount of particular proteins in each cell in two different individuals infected with dengue virus [6]. We note that these data are in a drastically different format from the images of the previous experiment, as they are in tabular form with cell i being row i and the amount of protein j in column j. We believe a key strength to neuron editing is its general applicability to a wide range of data types and modalities. In this particular experiment, there are four datasets, each consisting of measurements of 35 proteins: the two spike-ins we refer to as Control1 and Control2 are shape 18919×35 and 22802×35, respectively, while the two populations we actually want to study, called Sample1 and Sample2, are shape 94556×35 and 55594×35. To better grasp the problem of batch effects,

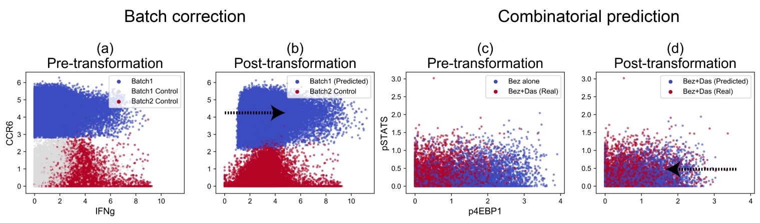


Figure 7.4: Neuron editing corrects the variation in IFNg while preserving the variation in CCR6 and correctly predicting the effect of combining two drugs.

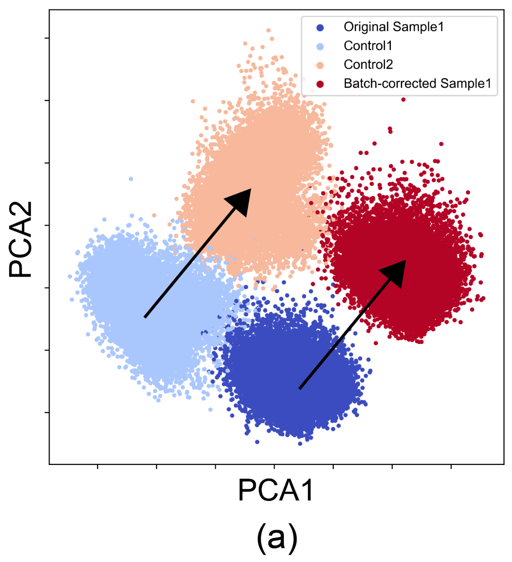
Neuron Editing	GAN	CycleGAN	ResnetGAN	RegAE	Constant Shift	CCA	MNN	ComBat	Limma
0.96275	0.5108	0.4310	0.6268	-0.0508	0.88205	0.6034	0.5339	0.5569	0.5431

Table 7.2: Correlation between observed change in spike-ins and applied change to samples. Neuron editing most accurately applies just the transformation observed as batch effect and not true biological variation.

we visualize a biaxial plot with two of the proteins where there is a batch effect in one dimension and a true underlying biological difference in the other dimension (Figure 7.4). By using the controls, we seek to correct the artificially low readings of the protein IFNg in Sample1 (along the x-axis) without removing the biologically accurate readings of higher amounts of protein CCR6 (along the y-axis).

We would like our model to identify this source of variation and compensate for the lower values of IFNg without losing other true biological variation in Sample1. For example, Sample1 also has higher values of the protein CCR6, and as the controls show, this is a true biological difference, not a batch effect (the y-axis in Figure 7.4a).

We quantify the performance of the models at this goal by measuring the correlation between the change in median marker values observed in the spike-in with the change applied to the sample. If this correlation is high, we know the transformation applied to the samples only removes the variation where we have evidence, coming from the spike-ins, that it is a technical artifact. This data is presented in Table 7.2, where we compare to not only the deep generative models we have already introduced, but also dedicated batch correction methods commonly used by practitioners [76, 58, 25]. We see that neuron editing outperforms all of the alternatives at extrapolating from the spike-ins to the samples. This is unsurprising, as the GAN methods are only trained to produce data like Control2, and thus will not preserve much of the variation in the sample. The traditional batch correction methods make specific



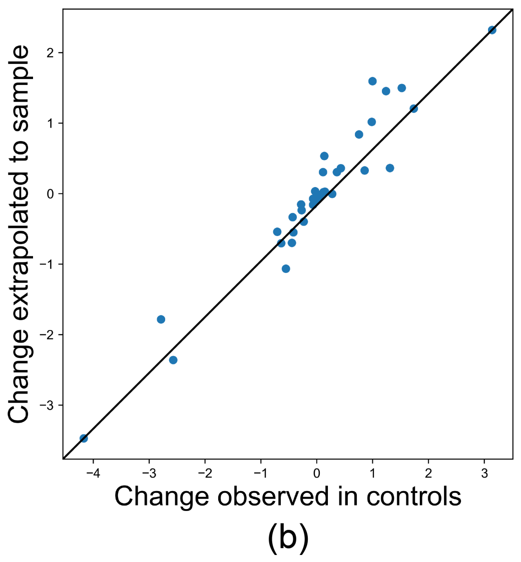


Figure 7.5: (a) The global shift in the two controls (light blue to red) is isolated and this variation is edited into the sample (dark blue to red), with all other variation preserved. (b) The median change in the sample in each dimension corresponds accurately with the evidence in each dimension in the controls.

	Neuron Editing	GAN	CycleGAN	ResnetGAN	RegAE	Constant Shift
r	**0.99661/0.97232**	0.87014/0.58130	0.92687/0.85380	0.94529/0.93032	0.91965/0.96053	0.96680/0.96925

Table 7.3: Correlation between real and predicted means/variances on the combinatorial drug prediction data. The GANs generate data that is less accurate (means are off) and less diverse (variances are smaller) than the real data, while neuron editing best models the true distribution.

parametric distributional assumptions on the data that are not held in practice, and thus also perform poorly. The regularized autoencoder, since the transformation is performed during training rather than after training like neuron editing, just reproduces its input unchanged.

In Figure 7.5a, a PCA embedding of the data space is visualized for Control1 (light blue), Control2 (light red), Sample1 (dark blue), and post-transformation Sample1 (dark red). The transformation from Control1 to Control2 mirrors the transformation applied to Sample1. Notably, the other variation (intra-sample variation) is preserved. In Figure 7.5b, we see that for every dimension, the variation between the controls corresponds accurately to the variation introduced by neuron editing into the sample. These global assessments across the full data space offer additional corroboration that the transformations produced by neuron editing reasonably reflect the transformation as evidenced by the controls.

7.3.3 Combinatorial drug treatment prediction on single-cell data

Finally, we consider biological data from a combinatorial drug experiment on cells from patients with acute lymphoblastic leukemia [11]. The dataset we analyze consists of cells under four treatments: no treatment (basal), BEZ-235 (Bez), Dasatinib (Das), and both Bez and Das (Bez+Das). These measurements also come from mass cytometry, this time on 41 dimensions, with the four datasets consisting of 19925, 20078, 19843, and 19764 observations, respectively. In this setting, we define the source to be the basal cells, the target to be the Das cells, and then extrapolate to the Bez cells. We hold out the true Bez+Das data and attempt to predict the effects of applying Das to cells that have already been treated with Bez.

Predicting the effects of drug combinations is an application which is typically approached through regression, fitting coefficients to an interaction term in a multiple linear regression model. This limitation of only fitting linear relationships and treating each protein independently, greatly restricts the model in a biological contexts where we know nonlinearity and protein regulatory networks exist and play a large role in cellular function. Using neuron editing in this context facilitates learning a much richer transformation than previous, non-deep learning methods.

We quantitatively evaluate whether neuron editing produces a meaningful transformation in Table 7.3, where we calculate the correlation between the real and generated means and variances of each dimension. Neuron editing more accurately predicts the principle direction and magnitude of transformation across all dimensions than any other model. Furthermore, neuron editing better preserves the variation in the real data. The GANs have trouble modeling the diversity in the data, as manifested by their generated data having significantly less variance than really exists.

We see an example of the learned transformation by looking at a characteristic effect of applying Das: a decrease in p4EBP1 (seen on the x-axis of Figure 7.4c). No change in another dimension, pSTATS, is associated with the treatment (the y-axis of Figure 7.4c). Neuron editing accurately models this change in p4EBP1, without introducing any change in pSTATS or losing variation within the extrapolation dataset (Figure 7.4d).

We note that since much of the variation in the target distribution already exists in the source distribution and the shift is a relatively small one, we might expect the ResnetGAN to be able to easily mimic the target. However, despite the residual connections, it still suffers from the same problems as the other models using the generating approach: namely, the GAN objective encourages all output to be like the target it trained on. This leaves it unable to produce the correct distribution if it differs from the target of the learned transformation, as we see in this case.

7.4 Discussion

In this work, we have only consider learning from a single pair of distributions and applying it to another single distribution. We consider it an interesting direction for future work to extend this to multiple distributions, either for learning from and application to. Additional future work along these lines could include training parallel encoders with the same decoder, or training to generate conditionally.

Chapter 8

Discussion and Future Work

This thesis has put together concepts from computer science, namely deep learning and artificial intelligence, but also statistics and computational biology. Generative modeling is a complex goal at the intersection of many different fields. As the need for generative models increases, especially in data integration, no single model will be best in all settings. This thesis presents a few different ways to meld a generative model with its specific task and application setting.

Despite the progress made within it, open problems abound in the topics covered in this thesis. These open problems range from further applications of the models introduced in it, to improvements in the models themselves, to new models entirely for these applications. We discuss a few of these, but by no means a comprehensive list of them, next.

8.1 Applications of models

The models in this thesis have many further applications beyond those considered here. For example, data integration is a key task for the future of neuroscience. As the goals of understanding cognition and the brain grow more and more ambitious, experimental design is becoming increasingly complex to try to capture more information. As the experimental design becomes more complex, they became harder to analyze jointly. This is where data integration comes to prominence, especially the multi-modal kind rather than multi-sample of the same modality. MAGAN and TraVeLGAN are positioned well in the landscape to perform important roles for this application. When we seek to align multiple datasets of significantly different structure, flexible models like GANs are necessary rather than simpler ones that only perform rigid transformations or cannot be used in different data dimensionality at all. Furthermore, the need to constrict transformations during alignment motivates the use of an appropriate correspondence loss in MAGAN. Likewise, the very abstracted notion of information preservation used in TraVeLGAN is crucial for settings where one wants to use a pair of GANs for alignment but the traditional invertibility requirement for information preservation is inappropriate.

The FMGAN has exciting applications, as well, in future work either in the drug discovery field or diagnostic medicine modeling. For example, this thesis has only scraped the surface of predicting perturbation results based on drug metadata. By considering the chemical structure of a drug, we are able to predict perturbation results on a population of cells from a known cell line. Future work using other drug metadata, like known performance at treatment or encoded structural information rather than just a diagram of the structure, could be used. Most promising, the rapidly growing field of combinatorial drug perturbations represents an opportunity for the use of the FMGAN. Predicting not only the effect of one drug, but multiple drugs used in combination, is a key task going forward. While performing one experiment per drug is already timely and expensive, due to the math of combinatorics, the number of experiments that would need to be performed to measure all pairwise interaction of drugs, or three-way interactions, or further, grows exponentially and very quickly becomes infeasible. To be able to do this accurately with a model that runs at the speed of a standard neural network would make analyzing this important data possible, because it will have been generated rather physically experimented.

8.2 Improvements of models

While these models have been successful and represent improvements to the previously existing models at their tasks, there are further improvements that can be made. In the MAGAN framework, the correspondence loss depends on the ability to formulate a domain-specific formula that differentiates numerically between the aspects of a good alignment and those of a bad alignment. While this is often possible, and while it is a crucial step to use such a loss when it is possible to formulate one as opposed to never using such a loss, this is a limitation because it is not always possible to express such a loss. In this case, it would be helpful to have more generic forms of correspondence loss beyond the mean squared error to use in any situation such as this. These generic forms could be based on data geometry, data symmetry, external feature generation methods like graph scattering, or other ideas that leverage mutual information that could be preserved across the domains of alignment.

The FMGAN similarly could benefit from generalization of specific architectural choices made for effective performance on the given datasets considered. The network chosen to process conditions, for example, must be tailored to the type of conditions in a given dataset. While convolutional networks (two-dimensional in the case of images or one-dimensional in the case of strings) were effective in the datasets in this thesis, there exist conditions where the assumptions and invariances built into these models would be inappropriate. A generalization of the condition embedding network, or framework changes to the FMGAN that make it less sensitive to this architectural design choice would be beneficial and extend its applicability.

8.3 New frameworks of models

At a higher level, the field of generative modeling and multi-sample embedding have many different directions they can take. While the current landscape is dominated by the very popular GAN models, these need not be the dominant models in the intermediate to long term. Adversarial modeling has its disadvantages, some of which are being overcome at a rapid pace, but others of which may prove to be insurmountable. Entirely new paradigms not based on the use of an adversarial discriminator may be the future.

For example, several of the improvements to existing models presented in this thesis are based on adding ideas of data geometry to the GAN adversarial loss, which is based on the data density along the manifold. In aligning samples which are independent draws from related but not necessarily identical distributions, data density can be misleading. Some observation populations may only exist in one sample, or the different populations may have varying densities in the two samples. Both of these situations would render the use of a GAN adversary inappropriate. While this thesis introduces data geometry regularizations on top of the GAN adversary to address this problem, a future without a GAN adversary at all might be the next step forward.

The key to having the right mindset going forward will be to remain open. Generative models are so computationally demanding that their use has propelled forward drastically in recent years. In such a fast-developing landscape, one always needs to remain open to new developments. Keeping the target on the applications, and then letting the tools be determined by that goal, will be essential for continuing to develop important and useful research.

Bibliography

[1] 10x Genomics. *10x Genomics Datasets*. `https://support.10xgenomics.com/single-cell-gene-expression/datasets`.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*, 2018.

[4] Matthew Amodio and Smita Krishnaswamy. Magan: Aligning biological manifolds. *arXiv preprint arXiv:1803.00385*, 2018.

[5] Matthew Amodio and Smita Krishnaswamy. Travelgan: Image-to-image translation by transformation vector learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8983–8992, 2019.

[6] Matthew Amodio, David van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *bioRxiv*, page 237065, 2018.

[7] Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha

Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, pages 1–7, 2019.

[8] Matthew Amodio, David van Dijk, Guy Wolf, and Smita Krishnaswamy. Learning general transformations of data for out-of-sample extensions. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

[9] Matthew Amodio, Dennis Shung, Daniel B Burkhardt, Patrick Wong, Michael Simonov, Yu Yamamoto, David van Dijk, Francis Perry Wilson, Akiko Iwasaki, and Smita Krishnaswamy. Generating hard-to-obtain information from easy-to-obtain information: applications in drug discovery and clinical inference. *Patterns*, page 100288, 2021.

[10] Kartik Anand, Ginestra Bianconi, and Simone Severini. Shannon and von neumann entropy of random networks with heterogeneous expected degree. *Physical Review E*, 83(3):036109, 2011.

[11] Benedict Anchang, Kara L Davis, Harris G Fienberg, Brian D Williamson, Sean C Bendall, Loukia G Karacosta, Robert Tibshirani, Garry P Nolan, and Sylvia K Plevritis. Drug-nem: Optimizing drug combinations using single-cell perturbation response to account for intratumoral heterogeneity. *Proceedings of the National Academy of Sciences*, 115(18):E4294–E4303, 2018.

[12] Asha Anoosheh, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Combogan: Unrestrained scalability for image domain translation. *arXiv preprint arXiv:1712.06909*, 2017.

[13] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[14] Elham Azizi, Ambrose J Carr, George Plitas, Andrew E Cornish, Catherine Konopacki, Sandhya Prabhakaran, Juozas Nainys, Kenmin Wu, Vaidotas Kiseliovas, Manu Setty, et al. Single-cell map of diverse immune phenotypes in the breast tumor microenvironment. *Cell*, 174(5):1293–1308, 2018.

[15] Rhonda Bacher and Christina Kendziorski. Design and computational analysis of single-cell rna-sequencing experiments. *Genome biology*, 17(1):63, 2016.

[16] Dmitry R Bandura, Vladimir I Baranov, Olga I Ornatsky, Alexei Antonov, Robert Kinach, Xudong Lou, Serguei Pavlov, Sergey Vorobiev, John E Dick, and Scott D Tanner. Mass cytometry: technique for real time single cell multitarget immunoassay based on inductively coupled plasma time-of-flight mass spectrometry. *Analytical chemistry*, 81(16):6813–6822, 2009.

[17] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[18] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. In *Advances in neural information processing systems*, pages 752–762, 2017.

[19] Sean C Bendall, Garry P Nolan, Mario Roederer, and Pratip K Chattopadhyay. A deep profiler's guide to cytometry. *Trends in immunology*, 33(7):323–332, 2012.

[20] Marc Gorriz Blanch, Marta Mrak, Alan F Smeaton, and Noel E O'Connor. End-to-end conditional gan-based architectures for image colourisation. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2019.

[21] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[22] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[24] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in b-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[25] Andrew Butler and Rahul Satija. Integrated analysis of single cell transcriptomic data across conditions, technologies, and species. *bioRxiv*, page 164889, 2017.

[26] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5):411, 2018.

[27] Maren Buttner, Zhichao Miao, Alexander Wolf, Sarah A Teichmann, and Fabian J Theis. Assessment of batch-correction methods for scrna-seq data with a new test metric. *bioRxiv*, page 200345, 2017.

[28] J Donald Capra, Charles A Janeway, Paul Travers, and Mark Walport. *Inmunobiology: the inmune system in health and disease*. Garland Publishing,, 1999.

[29] celeba. Large-scale celebfaces attributes (celeba) dataset. `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`. Accessed: 2018-10-20.

[30] Huaming Chen, Jun Shen, Lei Wang, and Jiangning Song. Leveraging stacked denoising autoencoder in prediction of pathogen-host protein-protein interactions. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 368–375. IEEE, 2017.

[31] Lujia Chen, Chunhui Cai, Vicky Chen, and Xinghua Lu. Learning a hierarchical representation of the yeast transcriptomic machinery using an autoencoder model. *BMC bioinformatics*, 17(1):S9, 2016.

[32] William S Chen, Nevena Zivanovic, David van Dijk, Guy Wolf, Bernd Bodenmiller, and Smita Krishnaswamy. Embedding single-cell experimental conditions to reveal manifold structure of cancer drug perturbation effects. *bioRxiv*, 2019. doi: 10.1101/455436. DOI: 10.1101/455436.

[33] Stéphane Chevrier, Jacob Harrison Levine, Vito Riccardo Tomaso Zanotelli, Karina Silina, Daniel Schulz, Marina Bacac, Carola Hermine Ries, Laurie Ailles, Michael Alexander Spencer Jewett, Holger Moch, et al. An immune atlas of clear cell renal cell carcinoma. *Cell*, 169(4):736–749, 2017.

[34] Yueh-hsiu Chien, Christina Meyer, and Marc Bonneville. γ δ t cells: first line of defense and beyond. *Annual review of immunology*, 32:121–155, 2014.

[35] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint*, 1711, 2017.

[36] Eleonora Cimini, Concetta Castilletti, Alessandra Sacchi, Rita Casetti, Veronica Bordoni, Antonella Romanelli, Federica Turchi, Federico Martini, Nicola Tumino, Emanuele Nicastri, et al. Human zika infection induces a reduction of ifn-γ producing cd4 t-cells and a parallel expansion of effector vδ2 t-cells. *Scientific reports*, 7(1):6313, 2017.

[37] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[38] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[39] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[40] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[41] Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

[42] Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979, 2017.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[44] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[45] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[46] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[47] fid. Fréchet inception distance (fid score) in pytorch. `https://github.com/mseitzer/pytorch-fid`. Accessed: 2018-10-20.

[48] Rachel Finck, Erin F Simonds, Astraea Jager, Smita Krishnaswamy, Karen Sachs, Wendy Fantl, Dana Pe'er, Garry P Nolan, and Sean C Bendall. Normalization of mass cytometry data with bead standards. *Cytometry Part A*, 83(5):483–494, 2013.

[49] Beatriz Garcillán, Ana VM Marin, Anaïs Jiménez-Reinoso, Alejandro C Briones, Miguel Muñoz-Ruiz, María J García-León, Juana Gil, Luis M Allende, Eduardo Martínez-Naves, María L Toribio, et al. gd t lymphocytes in the diagnosis of human t cell receptor immunodeficiencies. *Frontiers in immunology*, 6:20, 2015.

[50] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423. IEEE, 2016.

[51] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[54] Alexander N Gorban and Andrei Zinovyev. Fast and user-friendly non-linear principal manifold learning by method of elastic maps. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–9. IEEE, 2015.

[55] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[56] Anvita Gupta and James Zou. Feedback gan (fbgan) for dna: a novel feedback-loop architecture for optimizing protein functions. *arXiv preprint arXiv:1804.01694*, 2018.

[57] Laleh Haghverdi, Maren Buettner, F Alexander Wolf, Florian Buettner, and Fabian J Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods*, 13(10):845, 2016.

[58] Laleh Haghverdi, Aaron TL Lun, Michael D Morgan, and John C Marioni. Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 2018.

[59] Saad Haider and Ranadip Pal. Inference of tumor inhibition pathways from drug perturbation data. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 95–98. IEEE, 2013.

[60] Adrian Haimovich, Neal G Ravindra, Stoytcho Stoytchev, H Patrick Young, Francis P Wilson, David van Dijk, Wade L Schulz, and Richard Andrew Taylor. Development and

validation of the covid-19 severity index (csi): a prognostic tool for early respiratory decompensation. *medRxiv*, 2020.

[61] Ji Hun Ham, Daniel D Lee, and Lawrence K Saul. Learning high dimensional correspondences from low dimensional manifolds. 2003.

[62] Jihun Ham, Daniel D Lee, and Lawrence K Saul. Semisupervised alignment of manifolds. In *AISTATS*, pages 120–127, 2005.

[63] Changhee Han, Yoshiro Kitamura, Akira Kudo, Akimichi Ichinose, Leonardo Rundo, Yujiro Furukawa, Kazuki Umemoto, Yuanzhong Li, and Hideki Nakayama. Synthesizing diverse lung nodules wherever massively: 3d multi-conditional gan-based ct image augmentation for object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 729–737. IEEE, 2019.

[64] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[65] Yuta Hiasa, Yoshito Otake, Masaki Takao, Takumi Matsuoka, Kazuma Takashima, Aaron Carass, Jerry L Prince, Nobuhiko Sugano, and Yoshinobu Sato. Cross-modality image synthesis from unpaired data using cyclegan. In *International workshop on simulation and synthesis in medical imaging*, pages 31–41. Springer, 2018.

[66] Geoffrey E Hinton, Peter Dayan, and Michael Revow. Modeling the manifolds of images of handwritten digits. *IEEE transactions on Neural Networks*, 8(1):65–74, 1997.

[67] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[68] Judy Hoffman, Erik Rodner, Jeff Donahue, Trevor Darrell, and Kate Saenko. Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*, 2013.

[69] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[70] Christian Horvat and Jean-Pascal Pfister. Denoising normalizing flow. *Advances in Neural Information Processing Systems*, 34, 2021.

[71] Michael E Houle. Dimensionality, discriminability, density and distance distributions. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 468–473. IEEE, 2013.

[72] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1510–1519, 2017.

[73] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[74] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[75] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[76] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

[77] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics*, 14(9):3098–3104, 2017.

[78] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *ICASSP 2019-2019*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6820–6824. IEEE, 2019.

[79] Leah C Katzelnick, Lionel Gresh, M Elizabeth Halloran, Juan Carlos Mercado, Guillermina Kuan, Aubree Gordon, Angel Balmaseda, and Eva Harris. Antibody-dependent enhancement of severe dengue disease in humans. *Science*, 358(6365):929–932, 2017.

[80] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[81] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[82] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, page 121, 2014.

[83] Allon M Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A Weitz, and Marc W Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

[84] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[85] Zhifeng Kong and Kamalika Chaudhuri. The expressive power of a class of normalizing flow models. In *International Conference on Artificial Intelligence and Statistics*, pages 3599–3609. PMLR, 2020.

[86] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[87] Anil Korkut, Weiqing Wang, Emek Demir, Bülent Arman Aksoy, Xiaohong Jing, Evan J Molinelli, Özgün Babur, Debra L Bemis, Selcuk Onur Sumer, David B Solit,

et al. Perturbation biology nominates upstream–downstream drug combinations in raf inhibitor resistant melanoma cells. *Elife*, 4:e04640, 2015.

[88] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[89] Ola Larsson, Masahiro Morita, Ivan Topisirovic, Tommy Alain, Marie-Jose Blouin, Michael Pollak, and Nahum Sonenberg. Distinct perturbation of the translatome by the antidiabetic drug metformin. *Proceedings of the National Academy of Sciences*, 109(23):8977–8982, 2012.

[90] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, jul 2015. doi: 10.1016/j.cell.2015.05.047. URL `http://dx.doi.org/10.1016/j.cell.2015.05.047`.

[91] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El-ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1):184–197, 7 2015. ISSN 00928674. doi: 10.1016/j.cell.2015.05.047.

[92] Jacob H Levine, Erin F Simonds, Sean C Bendall, Kara L Davis, D Amir El-ad, Michelle D Tadmor, Oren Litvin, Harris G Fienberg, Astraea Jager, Eli R Zunder, et al. Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, 2015.

[93] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[94] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*, pages 5495–5503, 2017.

[95] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. Towards understanding the dynamics of generative adversarial networks. *arXiv preprint arXiv:1706.09884*, 2017.

[96] Torgny Lindvall. *Lectures on the coupling method.* Courier Corporation, 2002.

[97] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[98] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[99] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

[100] Romain Lopez, Jeffrey Regier, Michael Cole, Michael Jordan, and Nir Yosef. A deep generative model for single-cell rna sequencing with application to detecting differentially expressed genes. *arXiv preprint arXiv:1710.05086*, 2017.

[101] William Lotter, Gabriel Kreiman, and David Cox. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, 2015.

[102] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditional cyclegan for attribute guided face image generation. *arXiv preprint arXiv:1705.09966*, 2017.

[103] Carolina Lucas, Patrick Wong, Jon Klein, Tiago BR Castro, Julio Silva, Maria Sundaram, Mallory K Ellingson, Tianyang Mao, Ji Eun Oh, Benjamin Israelow, et al. Longitudinal analyses reveal immunological misfiring in severe covid-19. *Nature*, 2020.

[104] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32:9408–9418, 2019.

[105] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, pages 715–732. Springer, 2020.

[106] Kry Lui, Gavin Weiguang Ding, Ruitong Huang, and Robert McCann. Dimensionality reduction has quantifiable imperfections: Two geometric bounds. In *Advances in Neural Information Processing Systems*, pages 8461–8471, 2018.

[107] Bin Luo and Edwin R Hancock. Iterative procrustes alignment with the em algorithm. *Image and Vision Computing*, 20(5-6):377–396, 2002.

[108] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[109] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[110] Mohamed Marouf, Pierre Machart, Vikas Bansal, Christoph Kilian, Daniel S Magruder, Christian F Krebs, and Stefan Bonn. Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks. *Nature communications*, 11(1):1–12, 2020.

[111] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[112] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 378–383. IEEE, 2016.

[113] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

[114] Shervin Minaee and Amirali Abdolrashidi. Iris-gan: Learning to generate realistic iris images using convolutional gan. *arXiv preprint arXiv:1812.04822*, 2018.

[115] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[116] Kevin R Moon, David van Dijk, Zheng Wang, William Chen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Phate: A dimensionality reduction method for visualizing trajectory structures in high-dimensional biological data. *bioRxiv*, page 120378, 2017.

[117] Kevin R. Moon, Jay S. Stanley, Daniel Burkhardt, David van Dijk, Guy Wolf, and Smita Krishnaswamy. Manifold learning-based methods for analyzing single-cell RNA-sequencing data. *Current Opinion in Systems Biology*, 7:36–46, 2 2018. ISSN 24523100. doi: 10.1016/j.coisb.2017.12.008.

[118] Kevin R. Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia van den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, 2019.

[119] Kevin R Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B Burkhardt, William S Chen, Kristina Yim, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, et al. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, 2019.

[120] Boaz Nadler, Stephane Lafon, Ioannis Kevrekidis, and Ronald R Coifman. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Advances in neural information processing systems*, pages 955–962, 2006.

[121] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in Neural Information Processing Systems*, pages 5585–5595, 2017.

[122] Eng-Jon Ong, Sameed Husain, and Miroslaw Bober. Siamese network of deep fisher-vector descriptors for image retrieval. *arXiv preprint arXiv:1702.00338*, 2017.

[123] Alexander Panda, Feng Qian, Subhasis Mohanty, David Van Duin, Frances K Newman, Lin Zhang, Shu Chen, Virginia Towle, Robert B Belshe, Erol Fikrig, et al. Age-associated decrease in tlr function in primary human dendritic cells predicts influenza vaccine response. *The Journal of Immunology*, page ji_0901022, 2010.

[124] Noseong Park, Ankesh Anand, Joel Ruben Antony Moniz, Kookjin Lee, Tanmoy Chakraborty, Jaegul Choo, Hongkyu Park, and Youngmin Kim. Mmgan: Manifold matching generative adversarial network for generating images. *arXiv preprint arXiv:1707.08273*, 2017.

[125] Mansi Patel, Xuyu Wang, and Shiwen Mao. Data augmentation with conditional gan for automatic modulation classification. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, pages 31–36, 2020.

[126] Franziska Paul, Ya'ara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, David Lara-Astiaso, Meital Gury, Assaf Weiner, et al. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 163(7):1663–1677, 2015.

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

[128] Jiaojiao Qiao, Huihui Song, Kaihua Zhang, Xiaolu Zhang, and Qingshan Liu. Image super-resolution using conditional generative adversarial network. *IET Image Processing*, 13(14):2673–2679, 2019.

[129] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[130] Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, et al. Science forum: the human cell atlas. *Elife*, 6:e27041, 2017.

[131] Barbara L Rellahan, Jeffrey A Bluestone, Bronwyn A Houlden, Melissa M Cotterman, and Louis A Matis. Junctional sequences influence the specificity of gamma/delta t cell receptors. *Journal of Experimental Medicine*, 173(2):503–506, 1991.

[132] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[133] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[134] Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Moressi, Forrester Cole, and Kevin Murphy. Xgan: Unsupervised image-to-image translation for many-to-many mappings. *arXiv preprint arXiv:1711.05139*, 2017.

[135] Paolo Russo, Fabio M Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan. *arXiv preprint arXiv:1705.08824*, 2017.

[136] Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. Simple and effective vae training with calibrated decoders. In *International Conference on Machine Learning*, pages 9179–9189. PMLR, 2021.

[137] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[138] Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

[139] ScienceMag. *A new dengue vaccine should only be used in people who were previously infected, WHO says*, 2018 (accessed August 9, 2018).

[140] Manu Setty, Michelle D Tadmor, Shlomit Reich-Zeliger, Omer Angel, Tomer Meir Salame, Pooja Kathail, Kristy Choi, Sean Bendall, Nir Friedman, and Dana Pe'er. Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nature biotechnology*, 34(6):637, 2016.

[141] Uri Shaham, Kelly P. Stanton, Jun Zhao, Huamin Li, Khadir Raddassi, Ruth Montgomery, and Yuval Kluger. Removal of batch effects using distribution-matching residual networks. *Bioinformatics*, 33(16):2539–2546, 2017. doi: 10.1093/bioinformatics/btx196. URL `+http://dx.doi.org/10.1093/bioinformatics/btx196`.

[142] Uri Shaham, Kelly P Stanton, Jun Zhao, Huamin Li, Khadir Raddassi, Ruth Montgomery, and Yuval Kluger. Removal of batch effects using distribution-matching residual networks. *Bioinformatics*, page btx196, 2017.

[143] Karthik Shekhar, Sylvain W Lapan, Irene E Whitney, Nicholas M Tran, Evan Z Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z Levin, James Nemesh, Melissa Goldman, et al. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell*, 166(5):1308–1323, 2016.

[144] shoe. igan. `https://github.com/junyanz/iGAN/tree/master/train_dcgan`. Accessed: 2019-02-01.

[145] Davi Sidarta-Oliveira and Licio Velloso. Comprehensive visualization of high-dimensional single-cell data with diffusion-based manifold approximation and projection (dbmap). *CELL-REPORTS-D-20-01731*.

[146] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[147] Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

[148] Aravind Subramanian, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, John F Davis, Andrew A Tubelli, Jacob K Asiedu, et al.

A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *Cell*, 171(6):1437–1452, 2017.

[149] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[150] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[151] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[152] Jie Tan, Matthew Ung, Chao Cheng, and Casey S Greene. Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. In *Pacific Symposium on Biocomputing Co-Chairs*, pages 132–143. World Scientific, 2014.

[153] Jie Tan, John H Hammond, Deborah A Hogan, and Casey S Greene. Adage-based integration of publicly available pseudomonas aeruginosa gene expression data with denoising autoencoders illuminates microbe-host interactions. *MSystems*, 1(1):e00025–15, 2016.

[154] Jie Tan, Georgia Doing, Kimberley A Lewis, Courtney E Price, Kathleen M Chen, Kyle C Cady, Barret Perchuk, Michael T Laub, Deborah A Hogan, and Casey S Greene. Unsupervised extraction of stable expression signatures from public compendia with an ensemble of neural networks. *Cell systems*, 5(1):63–71, 2017.

[155] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[156] Rachael Tatman. Gender and dialect bias in youtube's automatic captions. In

Proceedings of the First ACL Workshop on Ethics in Natural Language Processing, pages 53–59, 2017.

[157] Ange Tato and Roger Nkambou. Improving adam optimizer. 2018.

[158] Chen-Yu Tsai, Ka Hang Liong, Matilda Gertrude Gunalan, Na Li, Daniel Say Liang Lim, Dale A Fisher, Paul A MacAry, Yee Sin Leo, Siew-Cheng Wong, Kia Joo Puan, et al. Type i ifns and il-18 regulate the antiviral response of primary human γ δ t cells against dendritic cells infected with dengue virus. *The Journal of Immunology*, page 1303343, 2015.

[159] Aviad Tsherniak, Francisca Vazquez, Phil G Montgomery, Barbara A Weir, Gregory Kryukov, Glenn S Cowley, Stanley Gill, William F Harrington, Sasha Pantel, John M Krill-Burger, et al. Defining a cancer dependency map. *Cell*, 170(3):564–576, 2017.

[160] Po-Yuan Tung, John D Blischak, Chiaowen Joyce Hsiao, David A Knowles, Jonathan E Burnett, Jonathan K Pritchard, and Yoav Gilad. Batch effects and the effective design of single-cell gene expression studies. *Scientific reports*, 7:39921, 2017.

[161] Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. Deep feature interpolation for image content changes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7064–7073, 2017.

[162] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[163] David van Dijk, Juozas Nainys, Roshan Sharma, Pooja Kathail, Ambrose J Carr, Kevin R Moon, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *BioRxiv*, page 111591, 2017.

[164] David van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J. Carr, Cassandra Burdziak, Kevin R. Moon, Christine L. Chaffer, Diwakar Pattabiraman, Brian Bierie, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and

Dana Pe'er. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716 – 729.e27, 2018. doi: 10.1016/j.cell.2018.05.061.

[165] Lars Velten, Simon F Haas, Simon Raffel, Sandra Blaszkiewicz, Saiful Islam, Bianca P Hennig, Christoph Hirche, Christoph Lutz, Eike C Buss, Daniel Nowak, et al. Human haematopoietic stem cell lineage commitment is a continuous process. *Nature cell biology*, 19(4):271, 2017.

[166] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[167] Chang Wang and Sridhar Mahadevan. Manifold alignment using procrustes analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1120–1127. ACM, 2008.

[168] Chang Wang and Sridhar Mahadevan. Manifold alignment without correspondence. In *IJCAI*, volume 2, page 3, 2009.

[169] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *CVPR Workshops*, 2014.

[170] Xiaoqian Wang, Kamran Ghasedi Dizaji, and Heng Huang. Conditional generative adversarial network for gene expression inference. *Bioinformatics*, 34(17):i603–i611, 2018.

[171] Gregory P Way and Casey S Greene. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *bioRxiv*, page 174474, 2017.

[172] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. *arXiv preprint arXiv:1803.01541*, 2018.

[173] Lilian Weng. From gan to wgan. *arXiv preprint arXiv:1904.08994*, 2019.

[174] Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, 2018.

[175] Yungang Xu, Zhigang Zhang, Lei You, Jiajia Liu, Zhiwei Fan, and Xiaobo Zhou. scigans: single-cell rna-seq imputation using generative adversarial networks. *Nucleic acids research*, 48(15):e85–e85, 2020.

[176] Heran Yang, Jian Sun, Aaron Carass, Can Zhao, Junghoon Lee, Zongben Xu, and Jerry Prince. Unpaired brain mr-to-ct synthesis using a structure-constrained cyclegan. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 174–182. Springer, 2018.

[177] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Unsupervised neural machine translation with weight sharing. *arXiv preprint arXiv:1804.09057*, 2018.

[178] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2142–2150, 2015.

[179] Yi Yao, Rebecca Liu, Min Sun Shin, Mark Trentalange, Heather Allore, Ala Nassar, Insoo Kang, Jordan S Pober, and Ruth R Montgomery. Cytof supports efficient detection of immune cell subsets from small samples. *Journal of immunological methods*, 415:1–5, 2014.

[180] Yi Yao, Dara M Strauss-Albee, Julian Q Zhou, Anna Malawista, Melissa N Garcia, Kristy O Murray, Catherine A Blish, and Ruth R Montgomery. The natural killer cell response to west nile virus in young and old individuals with or without a prior history of infection. *PloS one*, 12(2):e0172625, 2017.

[181] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint*, 2017.

[182] Amit Zeisel, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, Hermany Munguba, Liqun He, Christer Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

[183] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[184] Wei Zhang, Chen Cao, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. Style transfer via image component analysis. *IEEE Transactions on multimedia*, 15(7):1594–1601, 2013.

[185] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018.

[186] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature quantization improves gan training. *arXiv preprint arXiv:2004.02088*, 2020.

[187] Yujiao Zhao, Matthew Amodio, Brent Vander Wyk, David van Dijk, Kevin Moon, Xiaomei Wang, Anna Malawista, Megan E. Cahill1, Anita Desai, Purnima Parthasarathy, Manjunatha Ventataswamy, V. Ravi, Priti Kumar, Smita Krishnaswamy, and Ruth R. Montgomery. Dengue virus patients show distinct immune signatures and retain immune response to infection with zika virus, 2018.

[188] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[189] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[190] Eli R Zunder, Ernesto Lujan, Yury Goltsev, Marius Wernig, and Garry P Nolan. A

continuous molecular roadmap to ipsc reprogramming through progression analysis of single-cell mass cytometry. *Cell Stem Cell*, 16(3):323–337, 2015.