The Krishnaswamy Laboratory

Yale Genetics and Yale SEAS present

Machine Learning

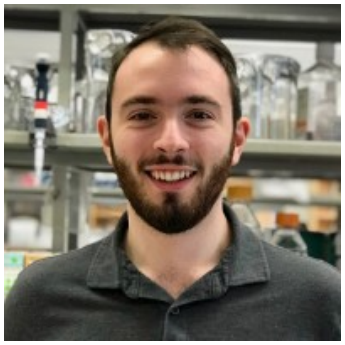
for

Single Cell Analysis

Online - May 20-29, 2020

What's something fun you're looking forward to doing or trying this summer?

Smita Krishnaswamy, PhD



Daniel Burkhardt

Scott Gigante

Nur-Taz Rahman, PhD

Sameet Mehta, PhD

Francesc Lopez, PhD

Andrew Benz

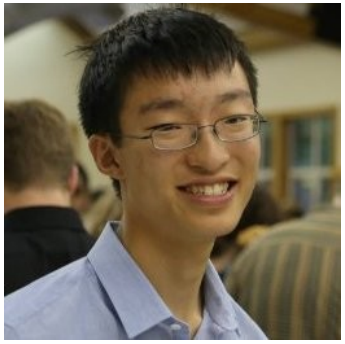
Mamie Wang

Wes Lewis

Matthew Amodio

Edel Aron

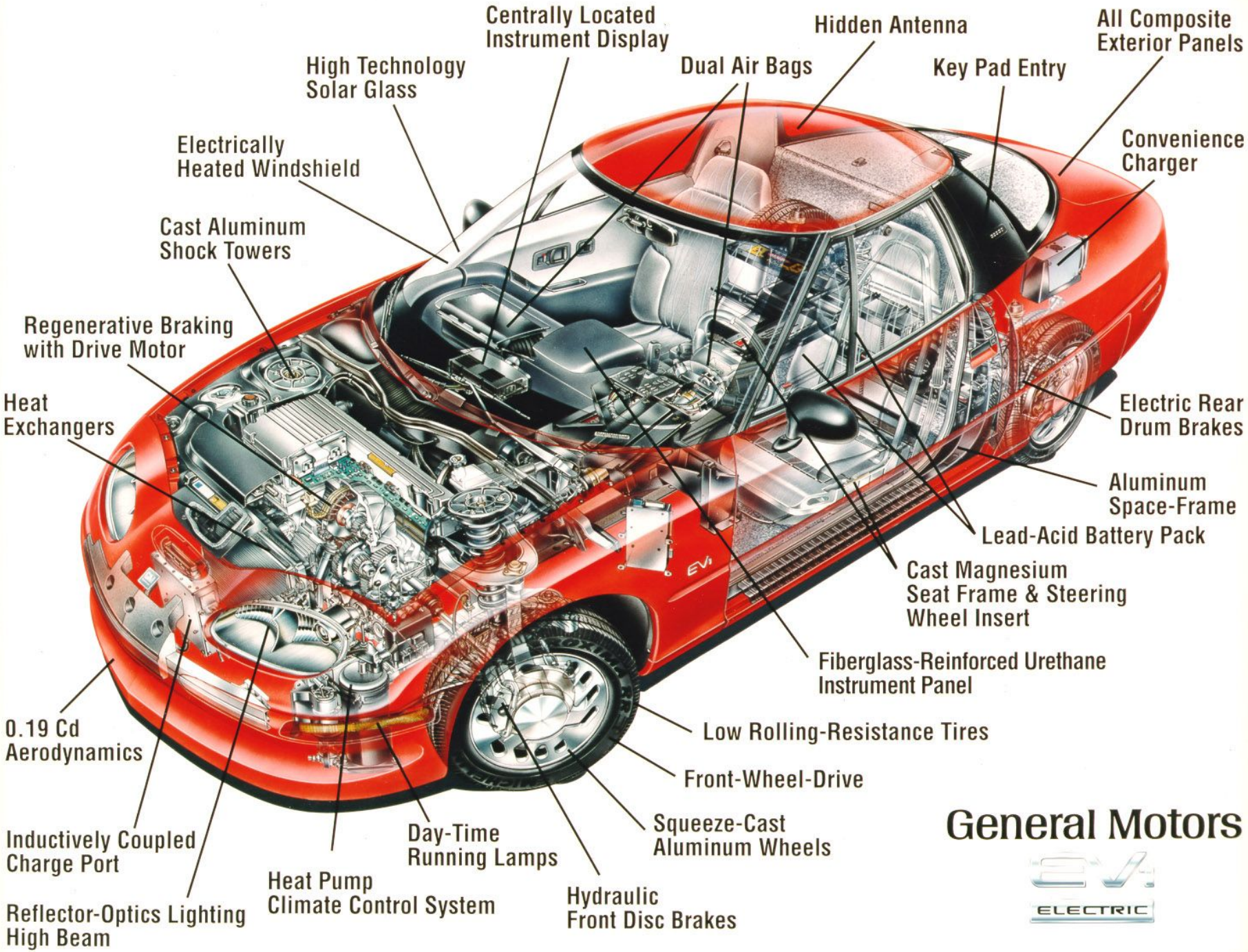
Egbert Castro



Alexander Tong

Evan Cudone

Aarthi Venkat


Centrally Located
Instrument Display
Hidden Antenna
All Composite
Exterior Panels
High Technology
Solar Glass
Dual Air Bags
Key Pad Entry
Electrically
Heated Windshield
Convenience
Charger
Cast Aluminum
Shock Towers
Regenerative Braking
with Drive Motor
Heat
Exchangers
Electric Rear
Drum Brakes
Aluminum
Space-Frame
Lead-Acid Battery Pack
Cast Magnesium
Seat Frame & Steering
Wheel Insert
Fiberglass-Reinforced Urethane
Instrument Panel
0.19 Cd
Aerodynamics
Low Rolling-Resistance Tires
Front-Wheel-Drive
Inductively Coupled
Charge Port
Day-Time
Running Lamps
Squeeze-Cast
Aluminum Wheels
Heat Pump
Climate Control System
Hydraulic
Front Disc Brakes
Reflector-Optics Lighting
High Beam
General Motors
EV1
ELECTRIC

Two kinds of machine learning

Supervised learning

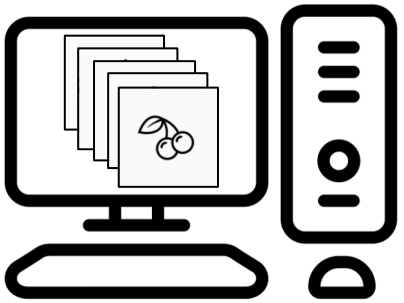
- Have a bunch of labelled data, want to label new data
- Learn a function $f(X) \rightarrow Y$ where all values of Y are known for some samples of X

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it
- Learn an embedding $f(X) \rightarrow Y, X \in \mathbb{R}^n, Y \in \mathbb{R}^m, n \gg m$
- Lower dimensional, easier to interpret (e.g. as clusters)

Supervised Learning Model

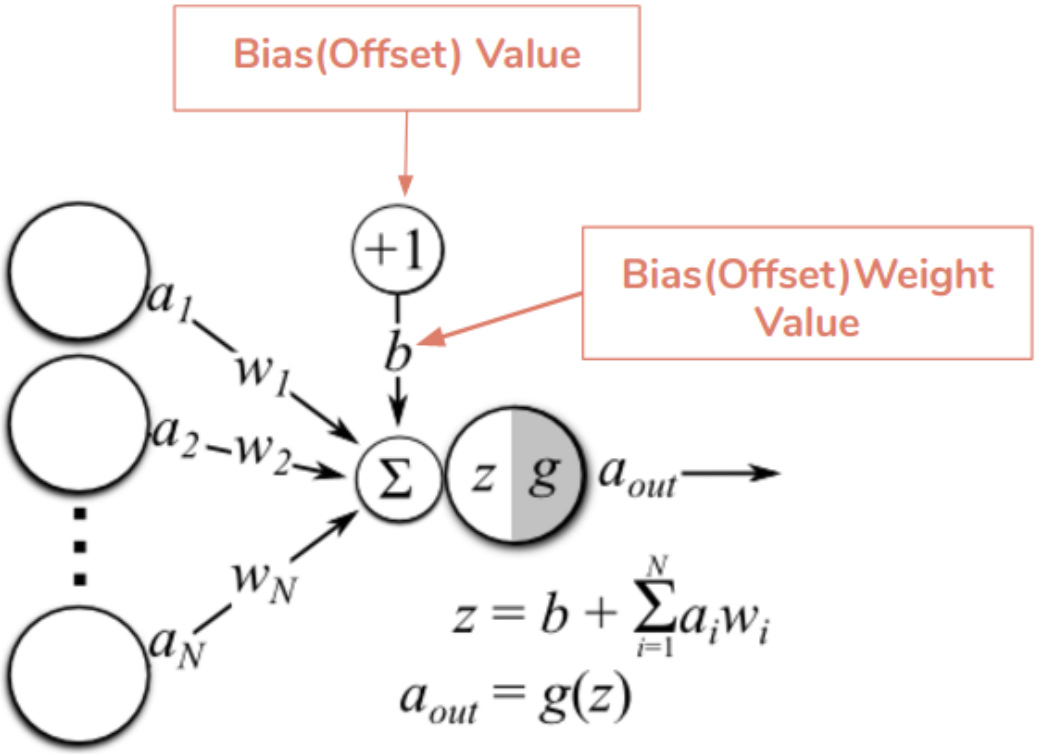
Unsupervised Learning Model



Noun project

The output of a neuron is a sum of (weights * input) + bias

$$Y = \sum (weight * input) + bias$$



https://docs.paperspace.com/machine-learning/wiki/weights-and-biases

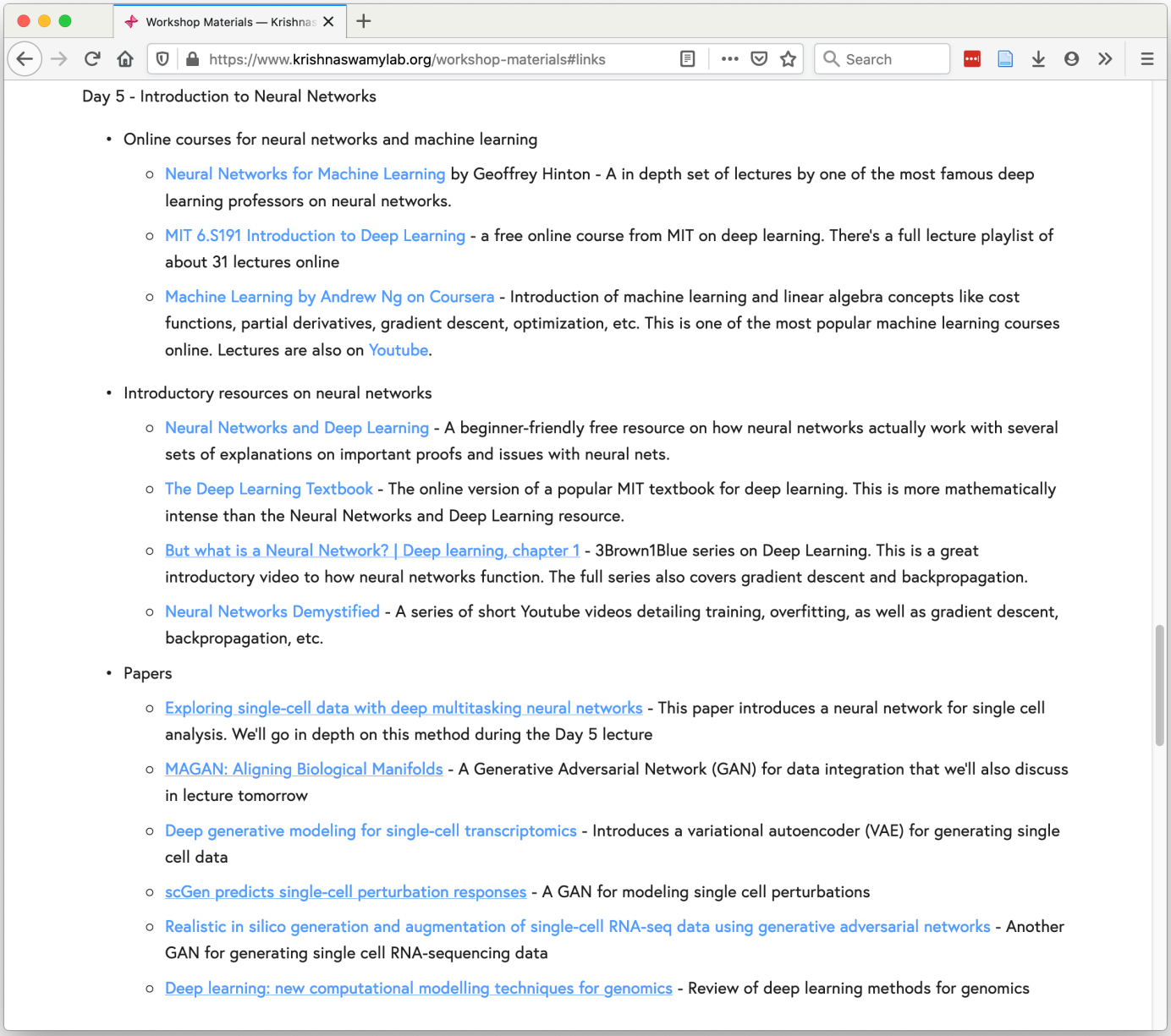
How to train your network

Training the network

1. **Randomly initialize** all **weights** and **biases**
2. Run the network on a set of **training data** (batch)
3. Compare the output of the network to the expected output for the input using the **cost function**
4. Calculate the **gradient** of the cost function
5. Update the weights and biases according to the gradient using **backpropagation**
6. Repeat steps 2-5 until **convergence**
7. Compare results to the **validation set** to evaluate the performance of the network
8. If performance on validation set is poor due to **underfitting** or **overfitting**, reconfigure the network **hyperparameters** (e.g. # nodes, # layers, activation function, etc) and repeat 1 - 8

Running the network

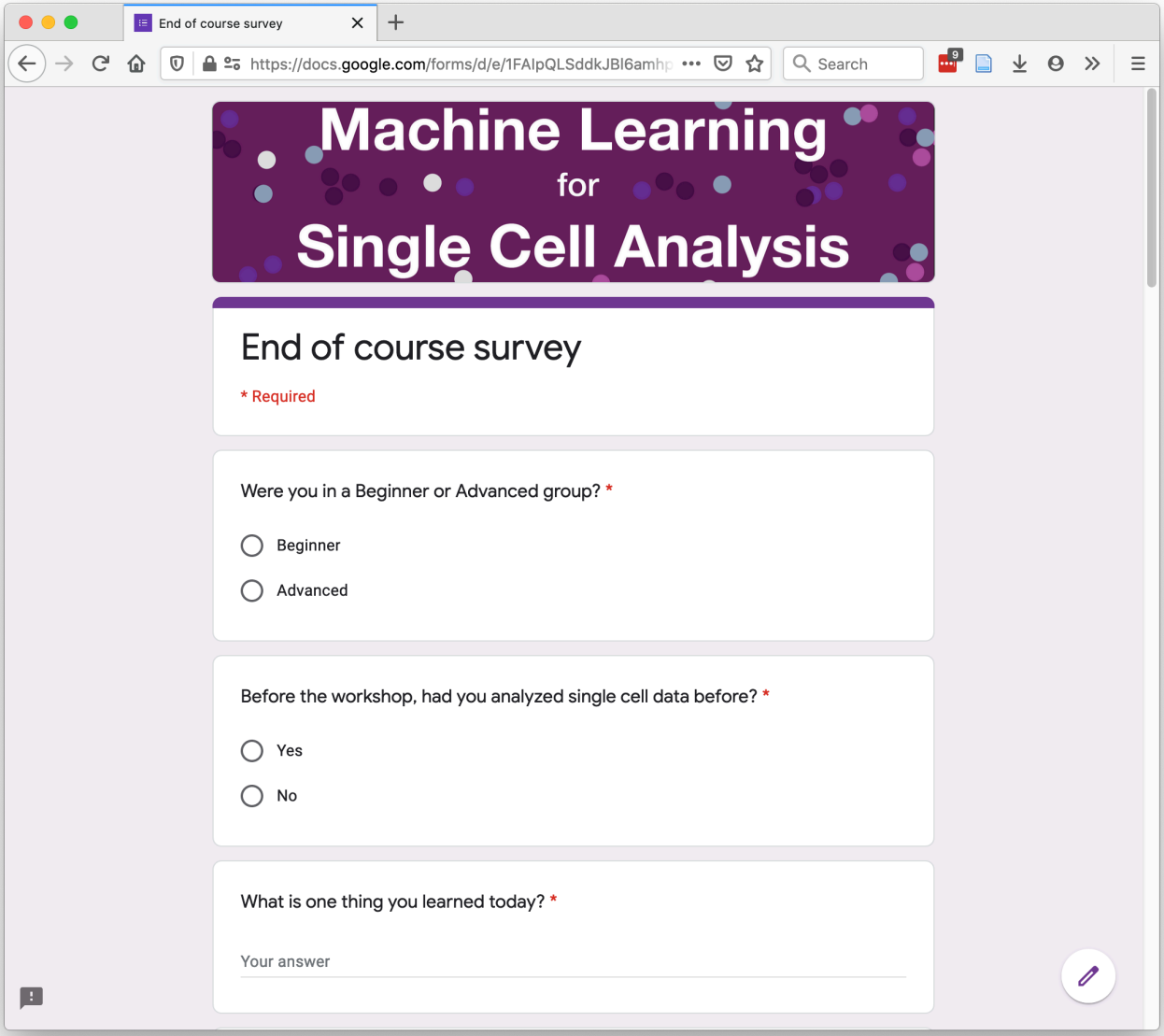
1. Run the network on your data
2. If your data is sufficiently different from the training data, you will need to **retrain** the network



Day 5 - Introduction to Neural Networks

- Online courses for neural networks and machine learning
  - Neural Networks for Machine Learning by Geoffrey Hinton - A in depth set of lectures by one of the most famous deep learning professors on neural networks.
  - MIT 6.S191 Introduction to Deep Learning - a free online course from MIT on deep learning. There's a full lecture playlist of about 31 lectures online
  - Machine Learning by Andrew Ng on Coursera - Introduction of machine learning and linear algebra concepts like cost functions, partial derivatives, gradient descent, optimization, etc. This is one of the most popular machine learning courses online. Lectures are also on Youtube.
- Introductory resources on neural networks
  - Neural Networks and Deep Learning - A beginner-friendly free resource on how neural networks actually work with several sets of explanations on important proofs and issues with neural nets.
  - The Deep Learning Textbook - The online version of a popular MIT textbook for deep learning. This is more mathematically intense than the Neural Networks and Deep Learning resource.
  - But what is a Neural Network? | Deep learning, chapter 1 - 3Brown1Blue series on Deep Learning. This is a great introductory video to how neural networks function. The full series also covers gradient descent and backpropagation.
  - Neural Networks Demystified - A series of short Youtube videos detailing training, overfitting, as well as gradient descent, backpropagation, etc.
- Papers
  - Exploring single-cell data with deep multitasking neural networks - This paper introduces a neural network for single cell analysis. We'll go in depth on this method during the Day 5 lecture
  - MAGAN: Aligning Biological Manifolds - A Generative Adversarial Network (GAN) for data integration that we'll also discuss in lecture tomorrow
  - Deep generative modeling for single-cell transcriptomics - Introduces a variational autoencoder (VAE) for generating single cell data
  - scGen predicts single-cell perturbation responses - A GAN for modeling single cell perturbations
  - Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks - Another GAN for generating single cell RNA-sequencing data
  - Deep learning: new computational modelling techniques for genomics - Review of deep learning methods for genomics

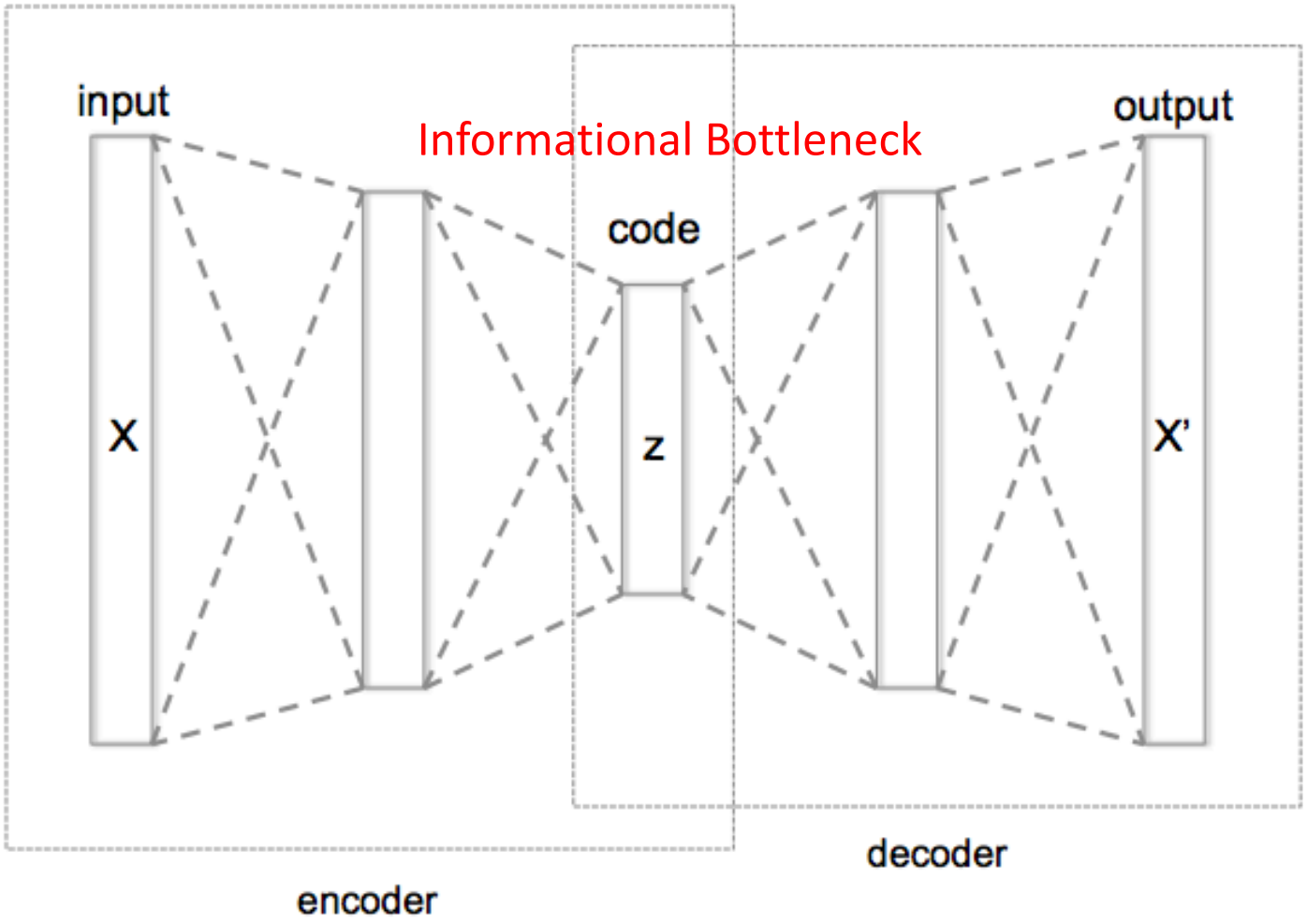
https://www.krishnaswamylab.org/workshop-materials#links



https://bit.ly/YaleML_May_PostSurvey

Autoencoders, GANs, Conditional GANs, Cycle GANs in Single Cell Analysis

Autoencoder



[Hinton, Salakhutdinov, Science 2006]

Autoencoder (AE)

- Neural network trained to copy its input x to its output
- Hidden layer z describes a ***code*** to represent the input
- Two parts
  - Encoder: $\mathrm{z} = f(x)$
  - Decoder: $x' = g(z)$
- Goal: minimize $\mathrm{C}\Big(x, g\big(f(x)\big)\Big)$
  - C penalizes $g\big(f(x)\big)$ for being dissimilar from x
  - Example: mean squared error

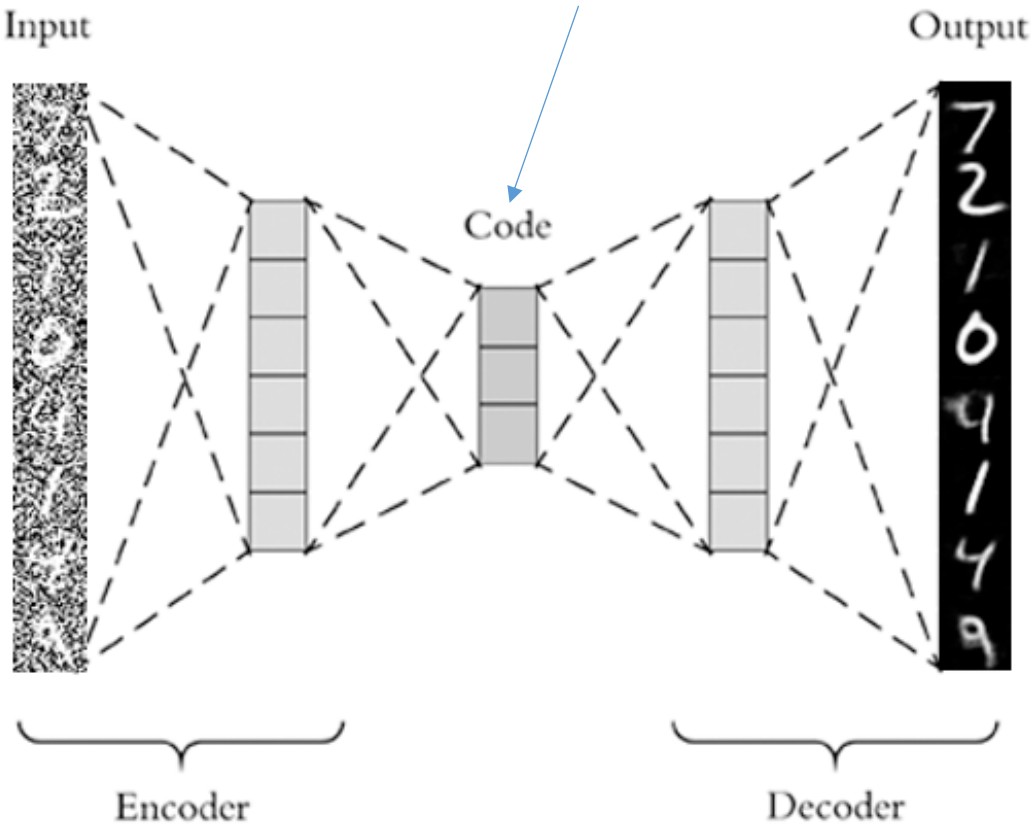
Autoencoders for Single-Cell Analysis

- Autoencoders can do most of the single cell tasks that we have talked about so far by construction!
- Dimensionality reduction
- Visualization
- Data denoising
- Batch correction
- Clustering
- (synthetic) Data generation

How?

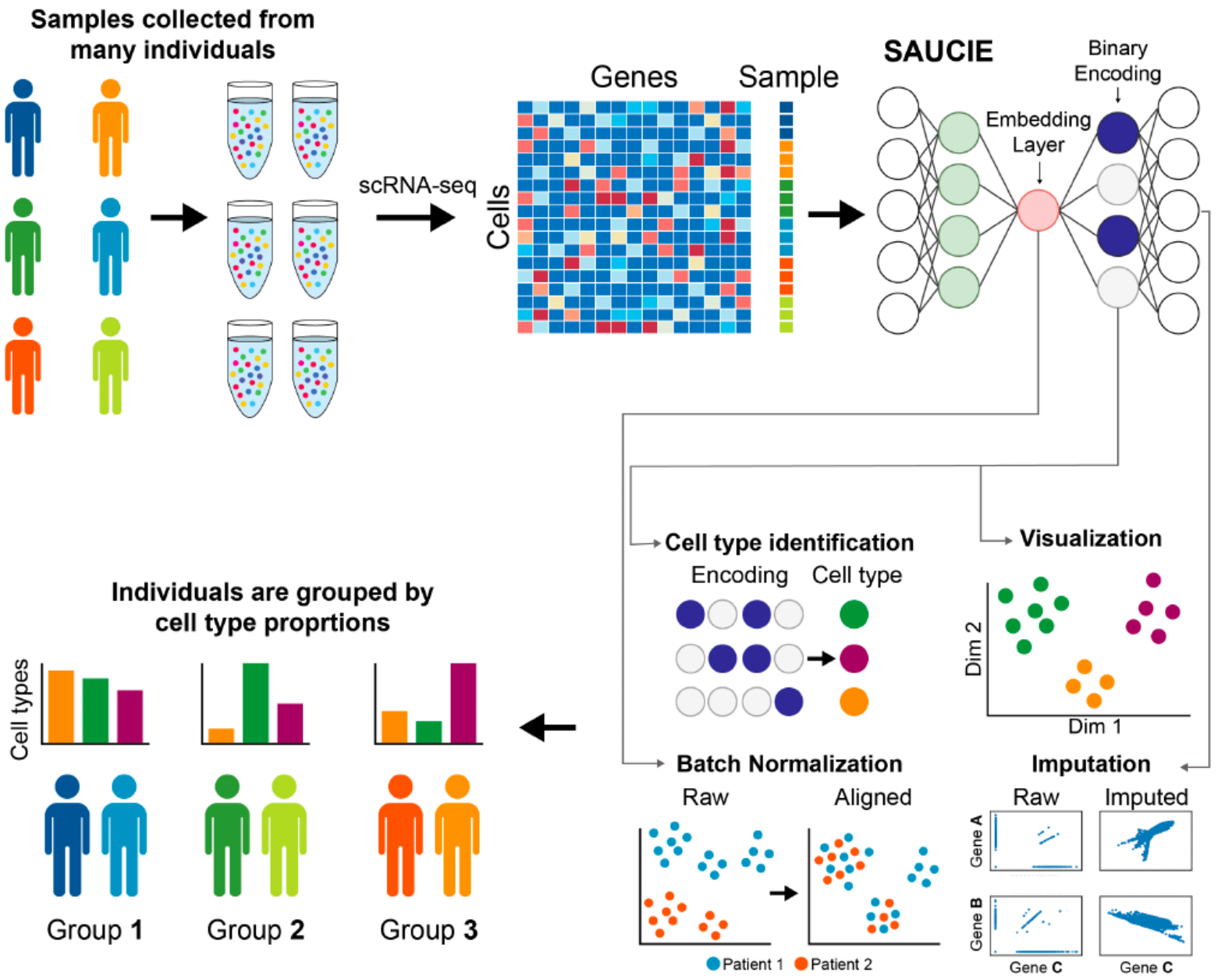
Code is a dimensionality reduced embedding

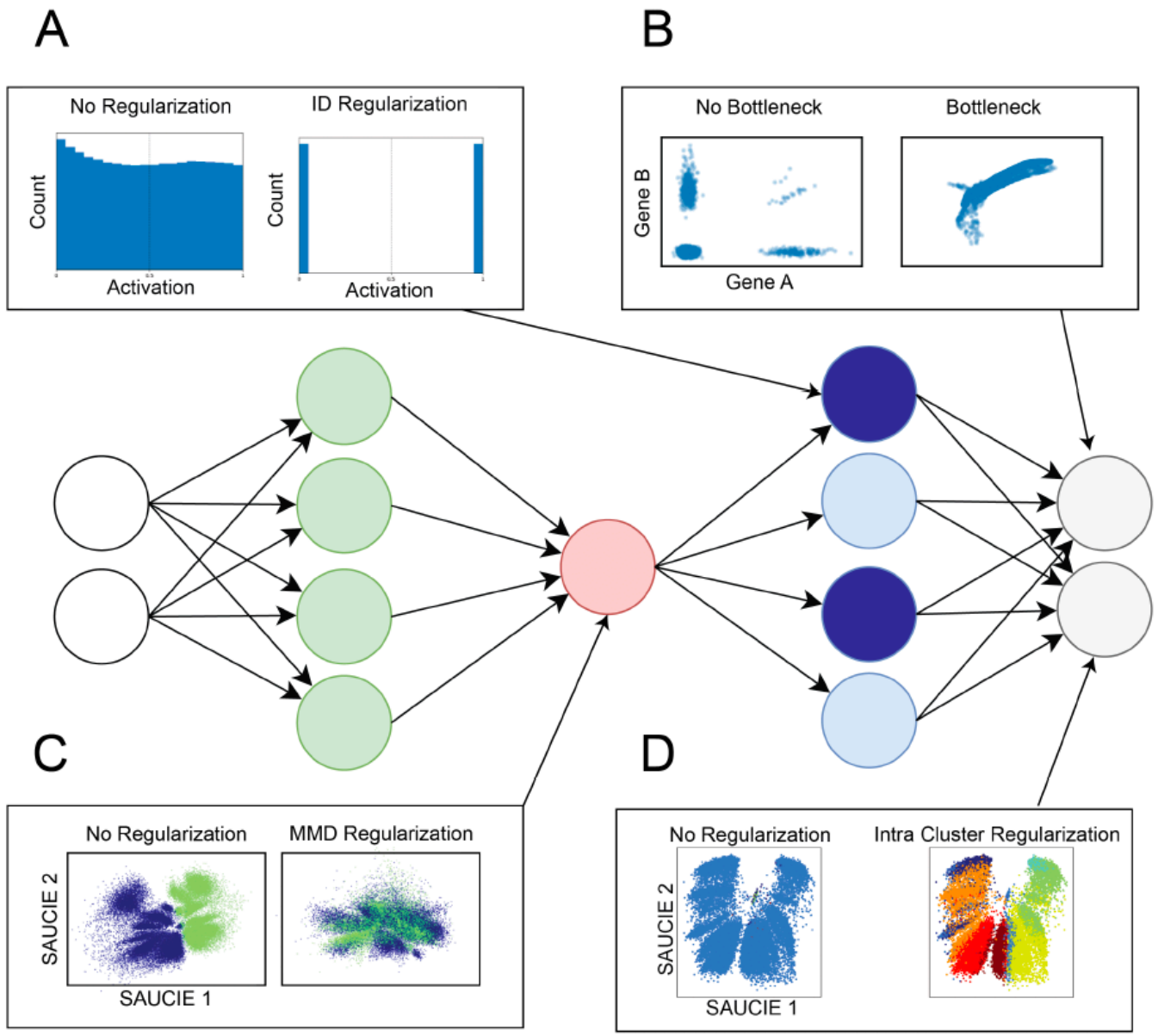
Reconstructed output is denoised (reconstructed from Reduced dimensions)



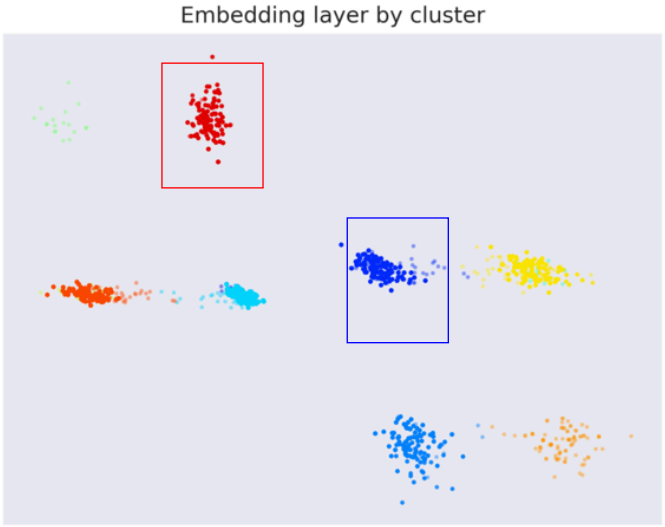
Remaining tasks need special regularizations

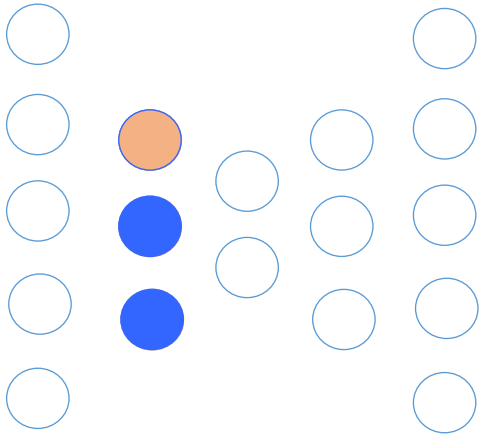
SAUCIE


Samples collected from many individuals
scRNA-seq
Genes
Sample
Cells
SAUCIE
Binary Encoding
Embedding Layer
Cell type identification
Encoding
Cell type
Visualization
Dim 2
Dim 1
Batch Normalization
Raw
Aligned
Patient 1
Patient 2
Imputation
Raw
Imputed
Gene A
Gene B
Gene C
Gene C
Individuals are grouped by cell type proprtions
Cell types
Group 1
Group 2
Group 3


A
No Regularization
ID Regularization
Count
Activation
Count
Activation
B
No Bottleneck
Bottleneck
Gene B
Gene A
C
No Regularization
MMD Regularization
SAUCIE 2
SAUCIE 1
D
No Regularization
Intra Cluster Regularization
SAUCIE 2
SAUCIE 1

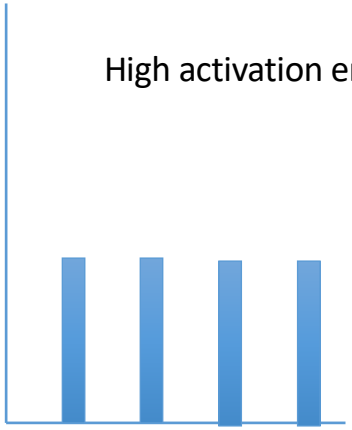
Representation for Clustering

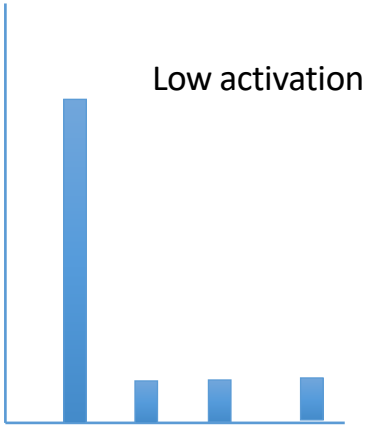




We want: codes for subspaces
Binary activation, good spacing

Clustering: Information Dimension Regularization





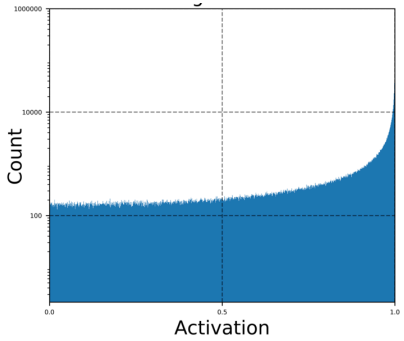
Penalty:

$$-\sum a_i \log(a_i)$$

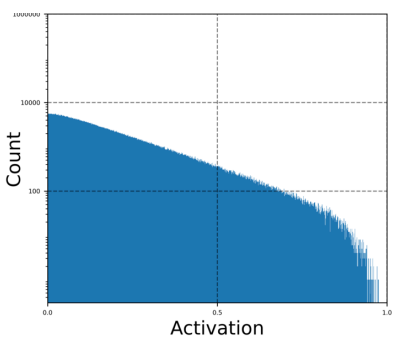
$$a_i = \frac{A_i}{\sum A_i}$$

Information Dimension (ID) Regularization

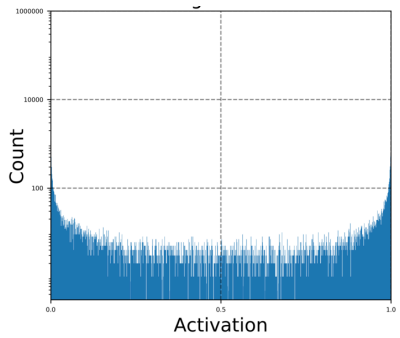
No regularization



L1 regularization

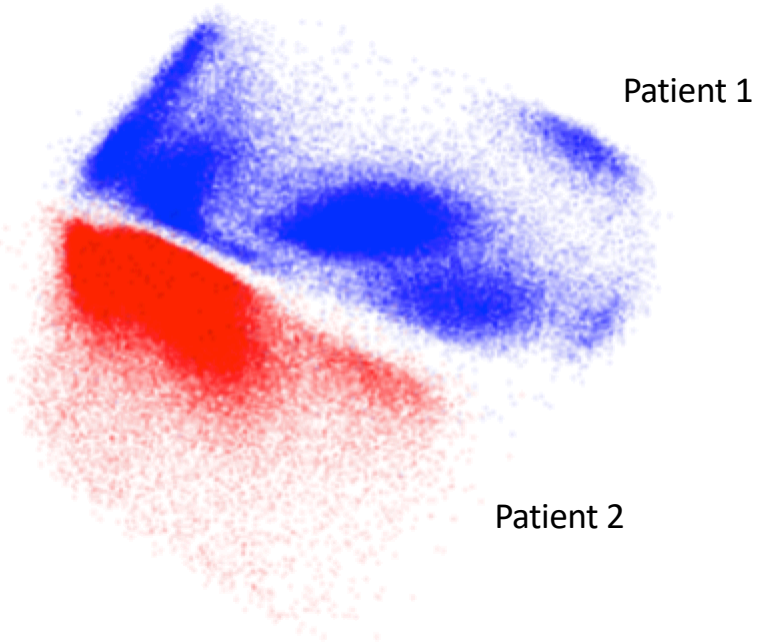


ID regularization

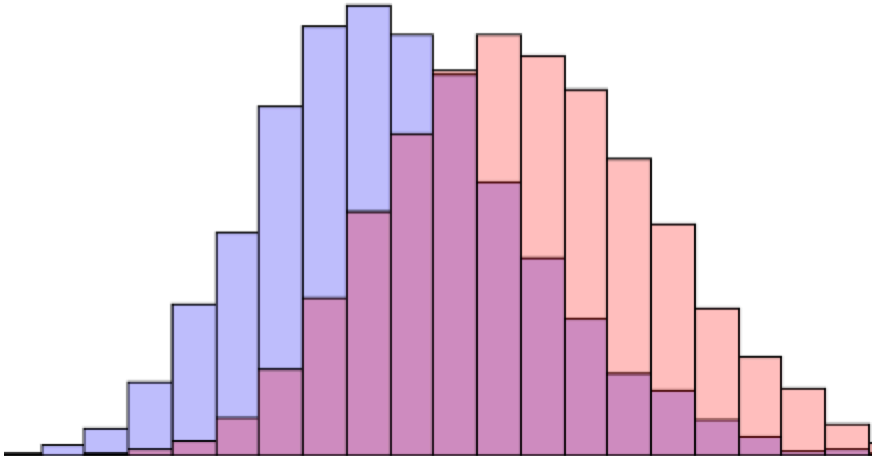


ID regularization enforces a binary-like code

Batch Effects: Manifold Alignment

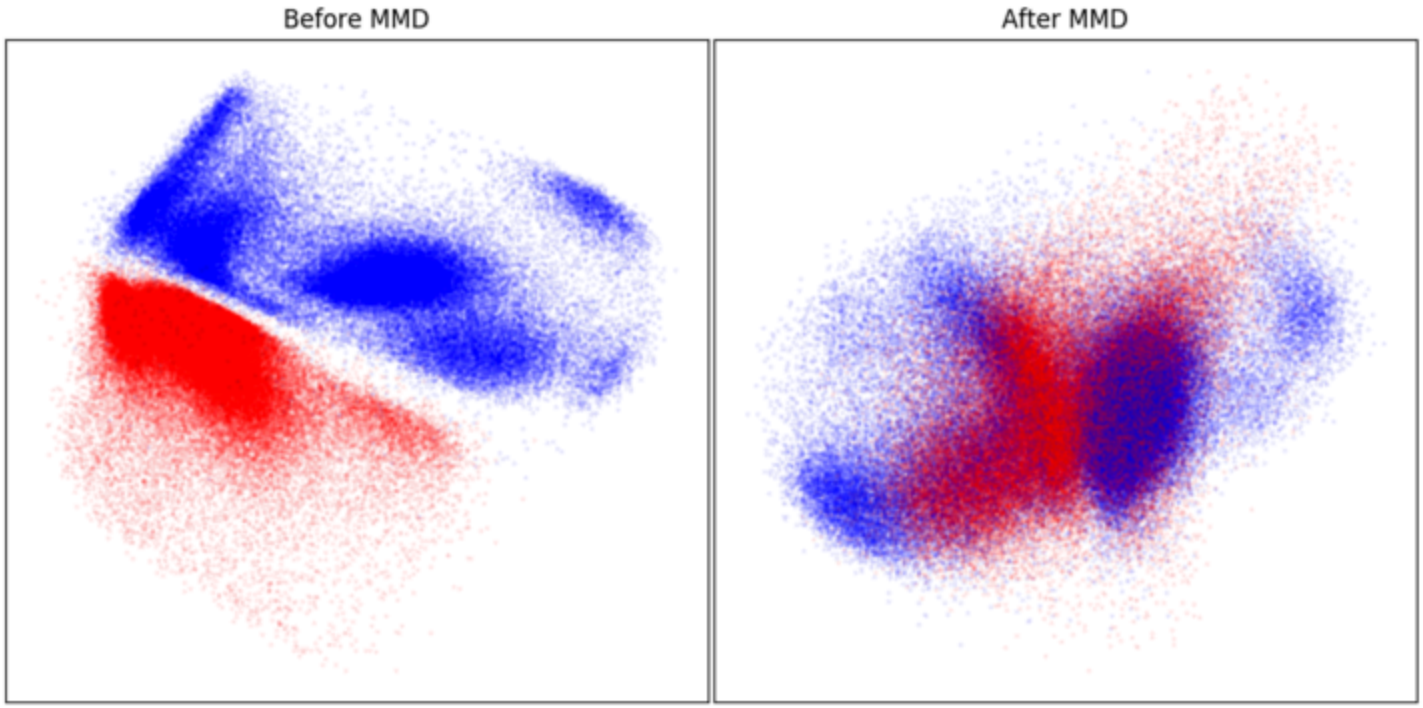


Mean Maximal Discrepancy

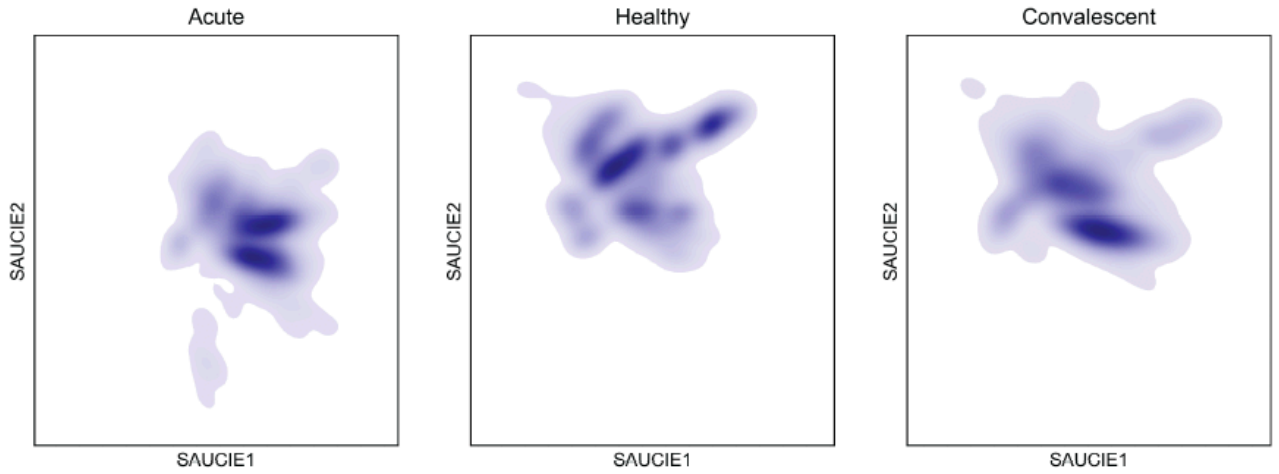


$$MMD(p,q) = \frac{1}{m^2}\Sigma_{i,j\in m}K(p_i,p_j) - \frac{2}{mn}\cdot\Sigma_{i,j}K(p_i,q_j) + \frac{1}{n^2}\Sigma_{i,j\in n}K(q_i,q_j)$$

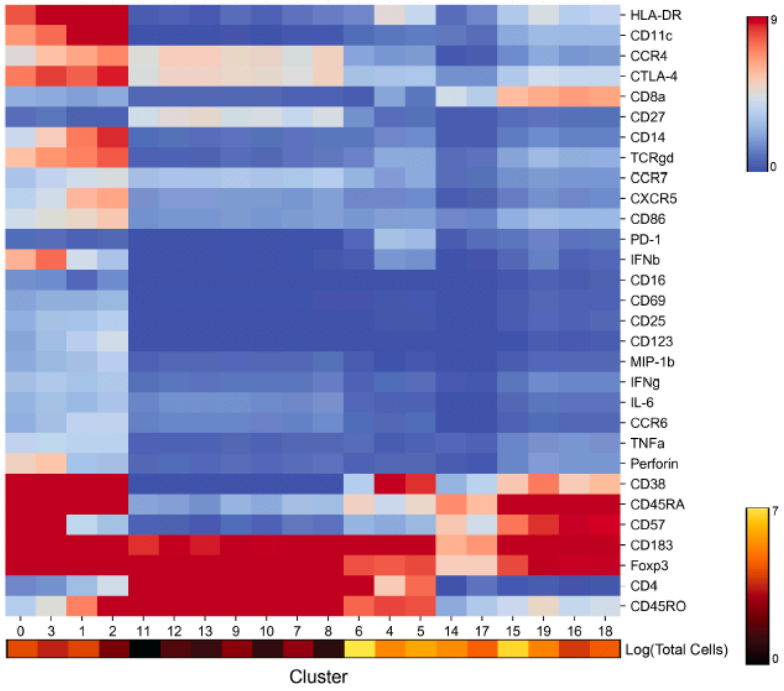
[Dziugaite, Roy, Ghahramani, UAI 2015]


Before MMD
After MMD

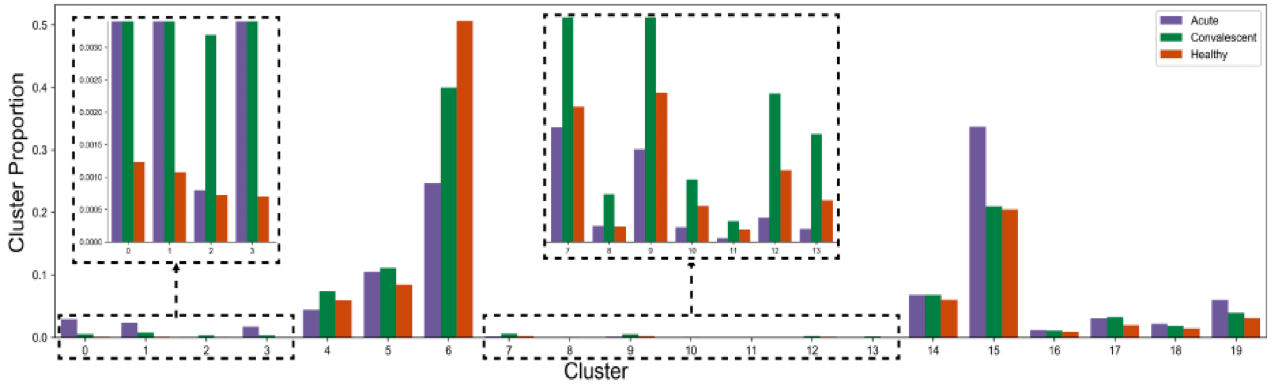
Cellular Manifolds of Dengue Patients



Cell Clusters: 180 Samples Combined

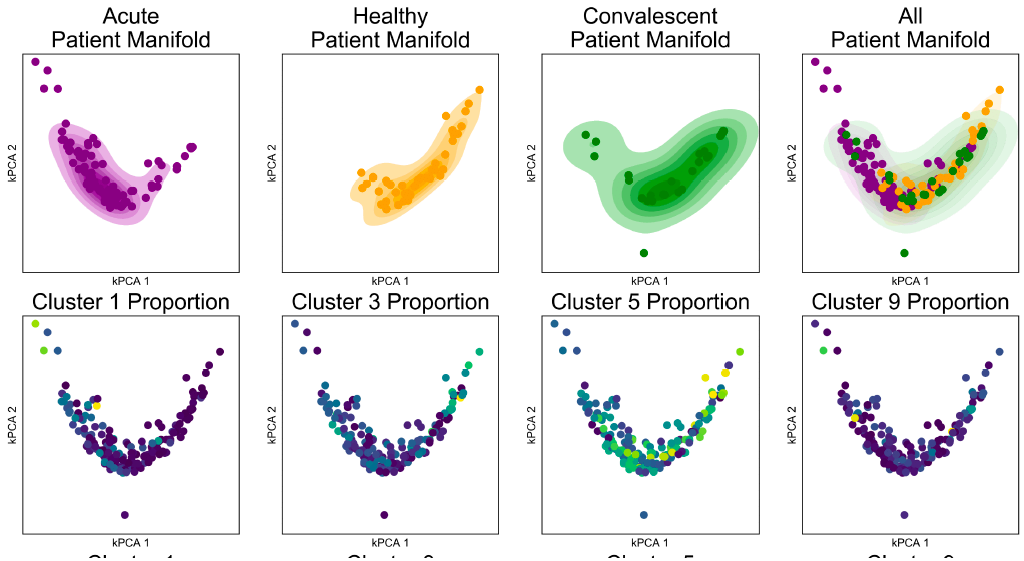


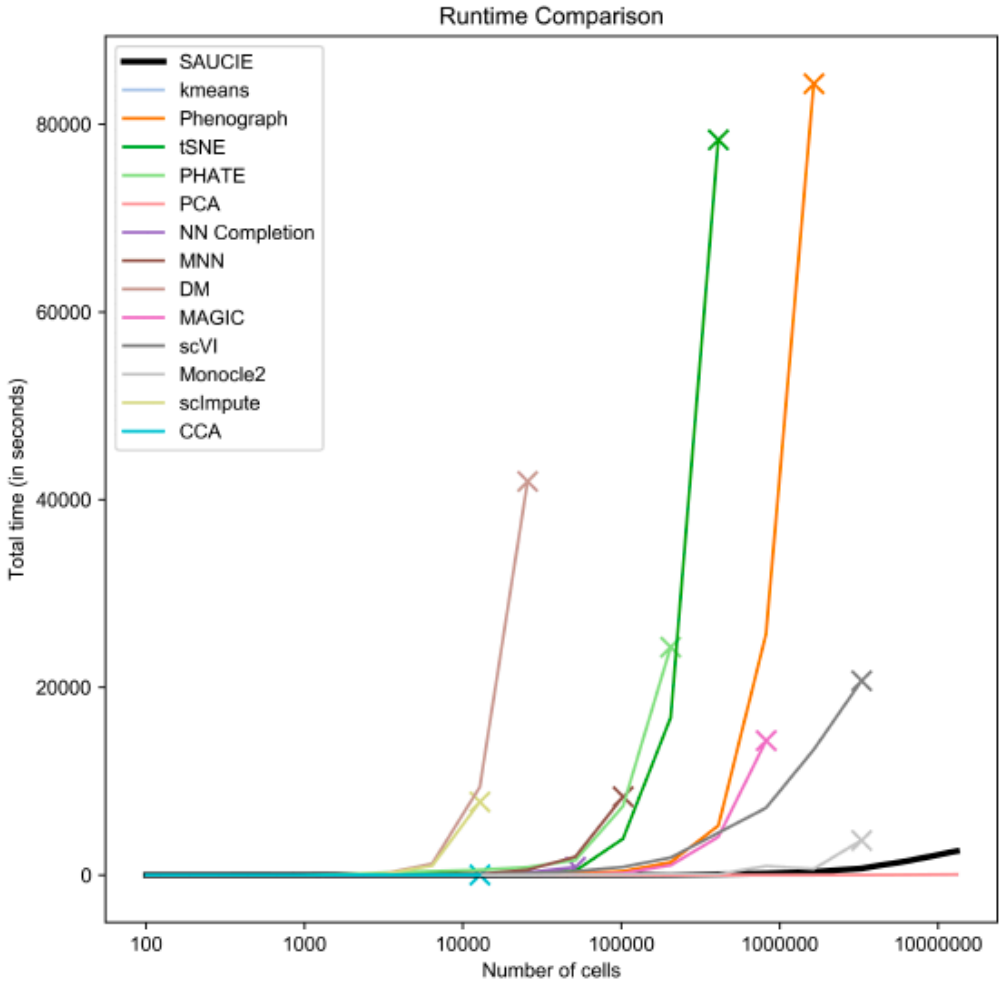
Patient Cluster Signatures



MMD distance can be derived between patient cluster proportions and embedded

Patient manifolds show coherent organization




Runtime Comparison
SAUCIE
kmeans
Phenograph
tSNE
PHATE
PCA
NN Completion
MNN
DM
MAGIC
scVI
Monocle2
scImpute
CCA
Total time (in seconds)
0
20000
40000
60000
80000
100
1000
10000
100000
1000000
10000000
Number of cells

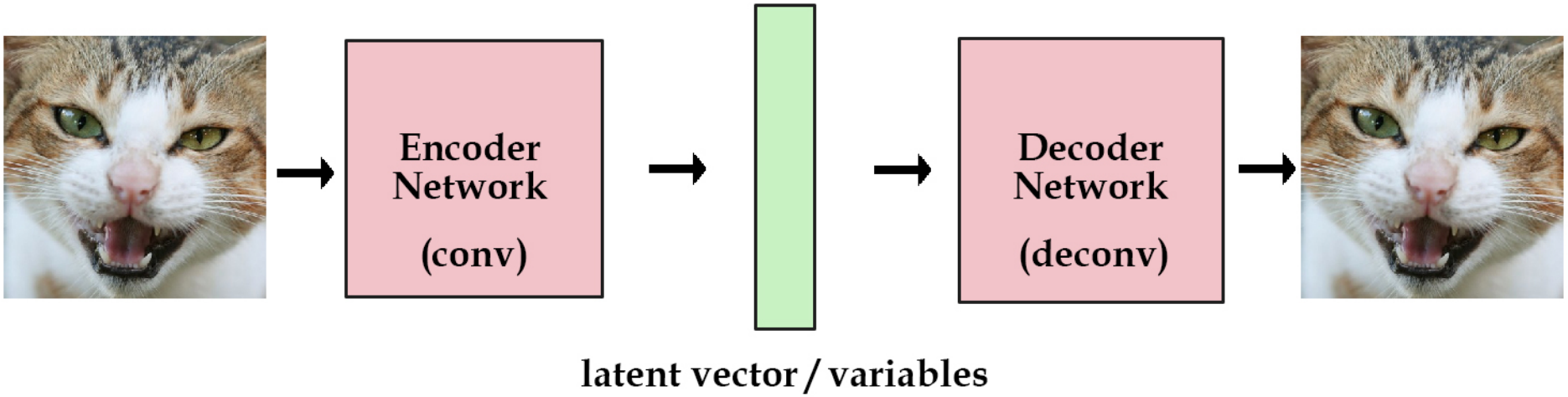
Generative Models

Generating samples from the data

- Key use of neural networks is generating samples from data
- One idea is to use the decoder without the encoder, sample from z and then have it decode to a datapoint

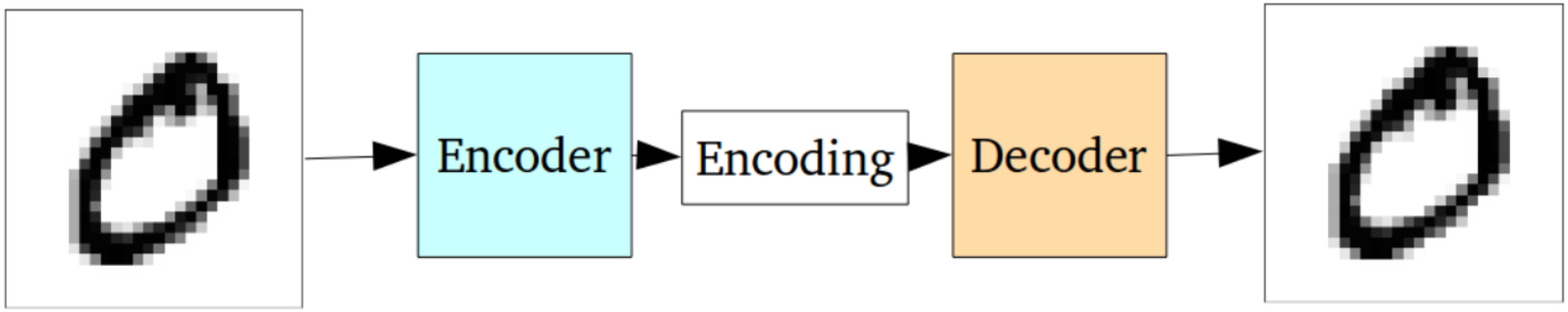
- For single cell data, this would mean generating synthetic cells from a distribution

Variational Autoencoders (VAEs)



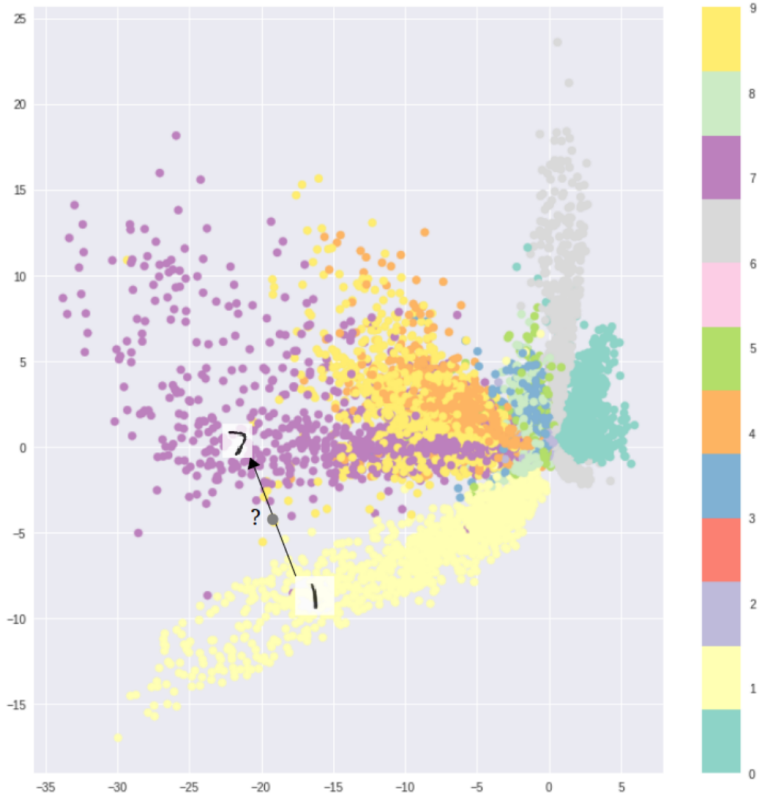
- A type of neural network whose latent space is designed for sampling

Why not autoencoders?



- Optimized with Reconstruction Loss
- Not explicitly penalized for Generative Purposes

Latent Space Not Optimal for Generation



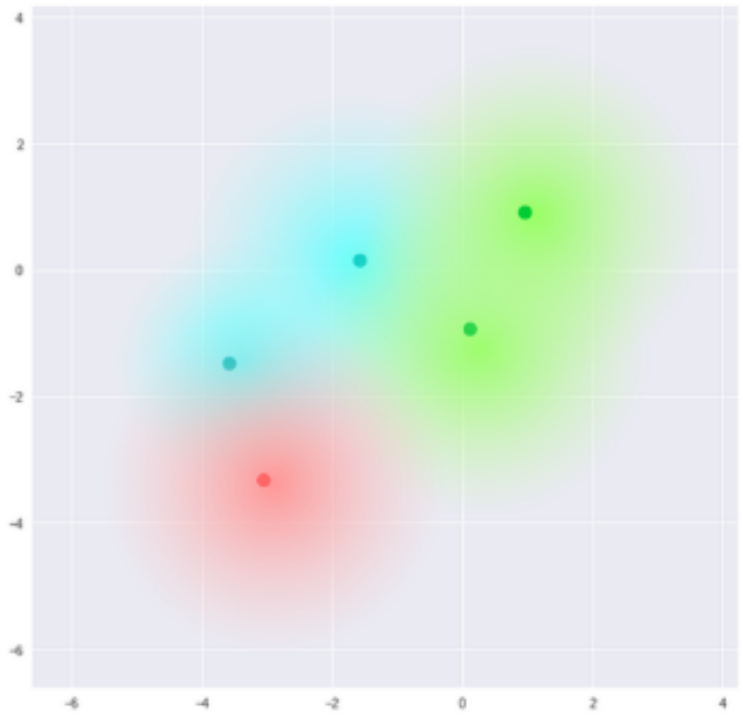
Optimizing purely for reconstruction loss

Formation of clusters helps decoding

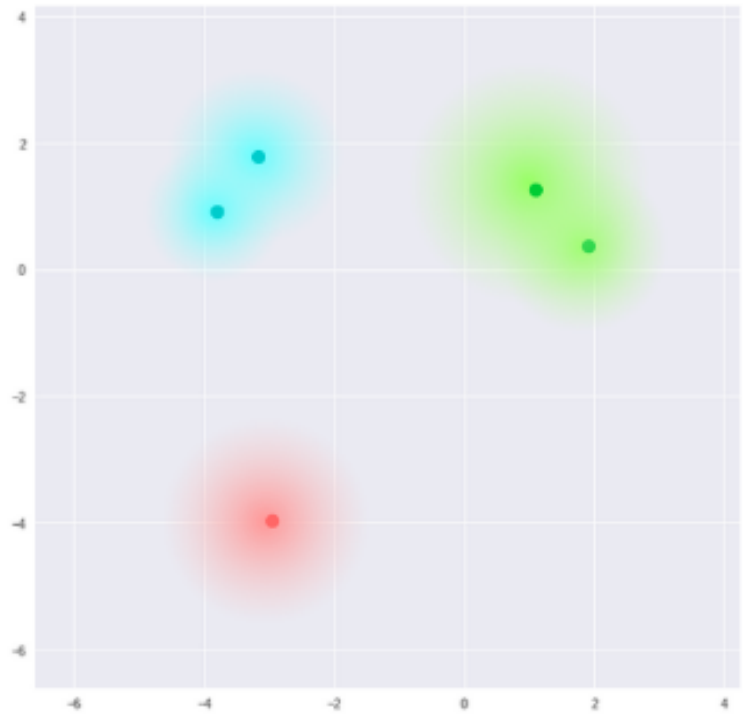
Does not help generation

Latent space cannot be interpolated

Encouraging continuous latent spaces



What we require



What we may inadvertently end up with

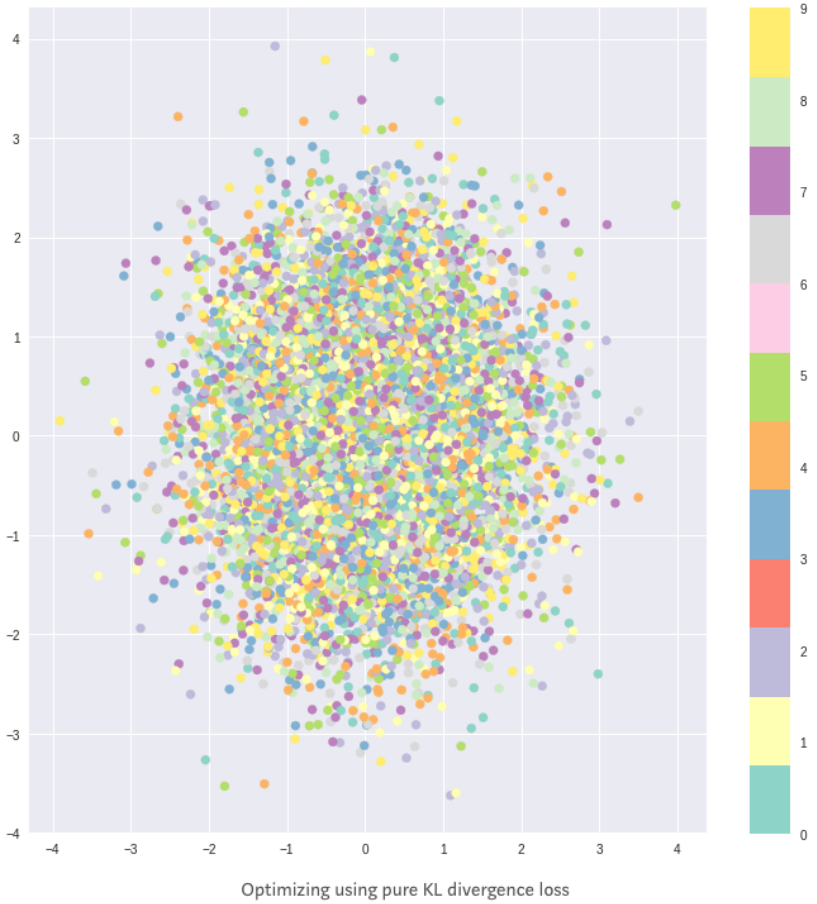
Solution: Penalize the Gaussian balls if they don't overlap!

KL Divergence Penalty

$$\mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Penalizing distributions
In the latent space from being
too far from a standard normal

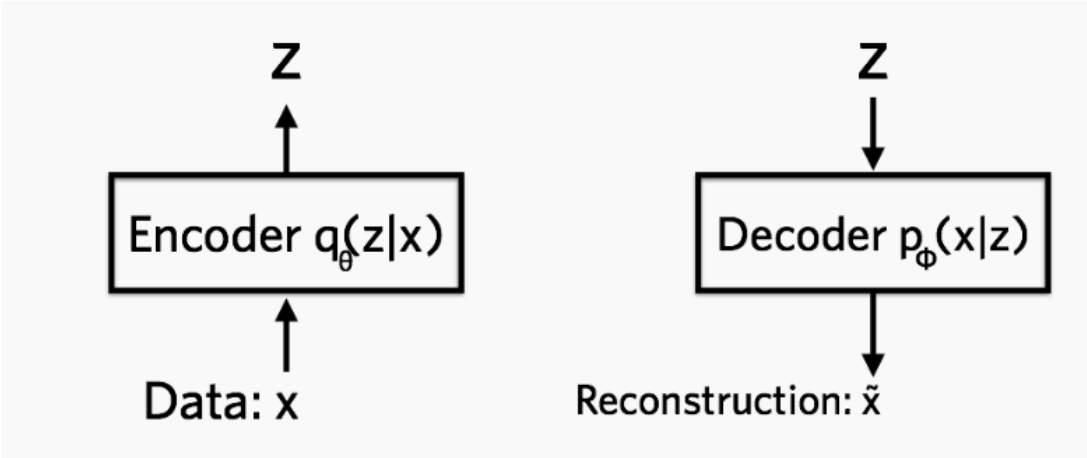
$$p(z) = Normal(0, 1).$$



Encourages every ball to be the same!

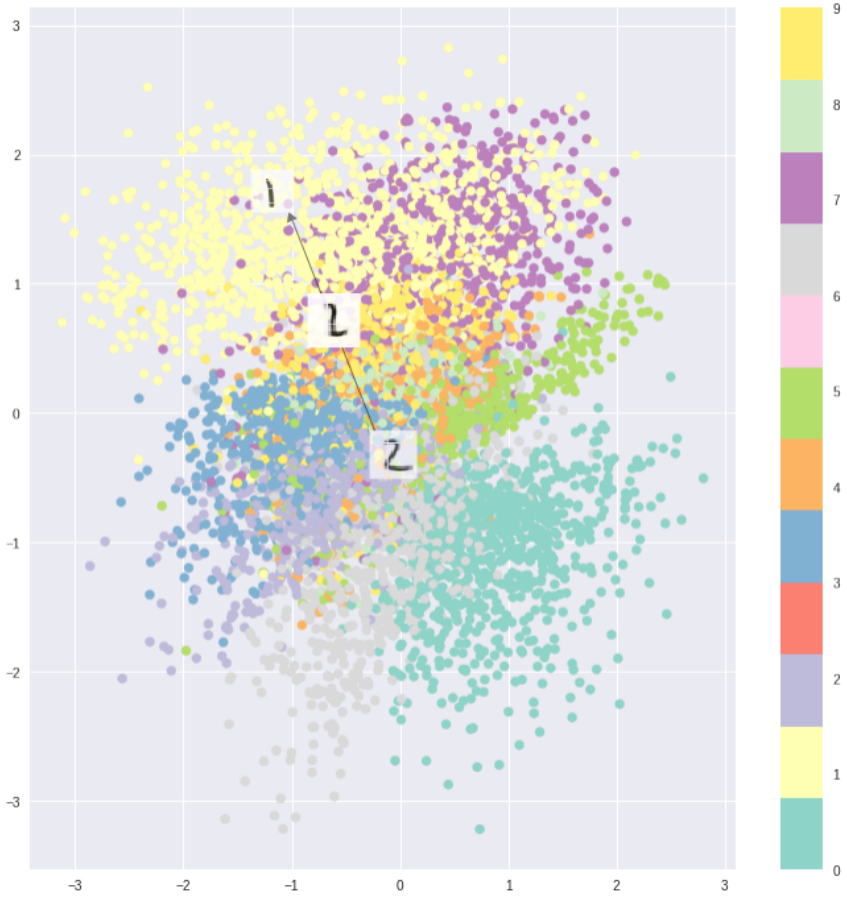
Loss Function of VAE

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$



Reconstruction penalty encourages separation $-\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]$

KL+Reconstruction Loss



Optimizing using both reconstruction loss and KL divergence loss

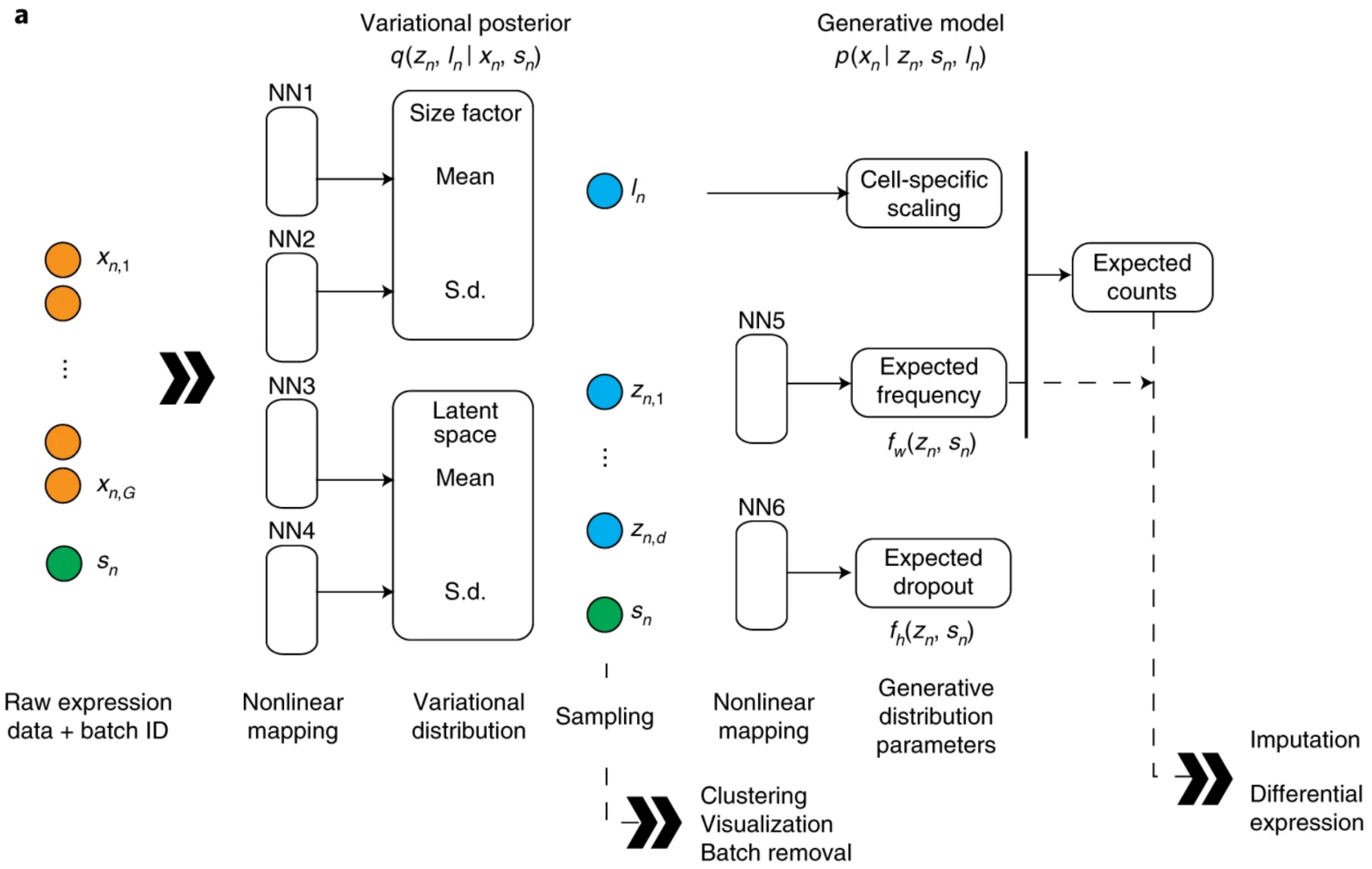
Variational Autoencoders (VAEs)

1st Epoch

9th Epoch

Original

VAEs in Single Cell Analysis

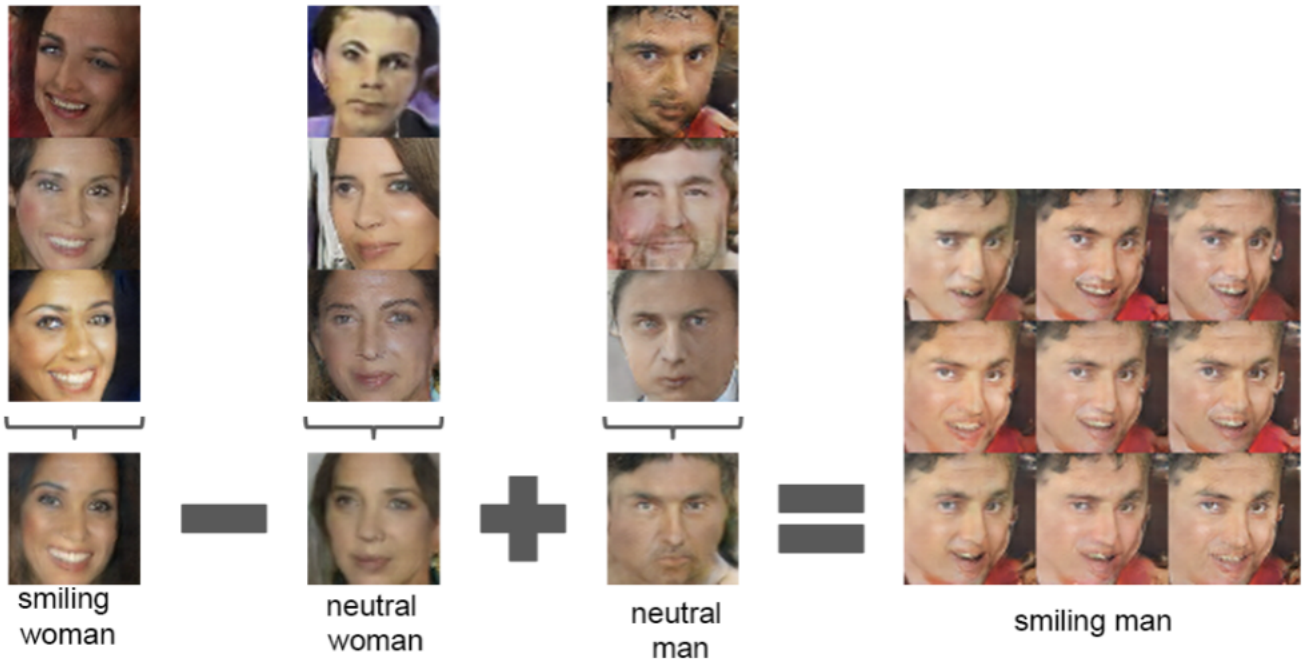


scVI uses latent space of VAE as a dimensionality reduced encoding fed to other algorithms

Probabilistic assumptions

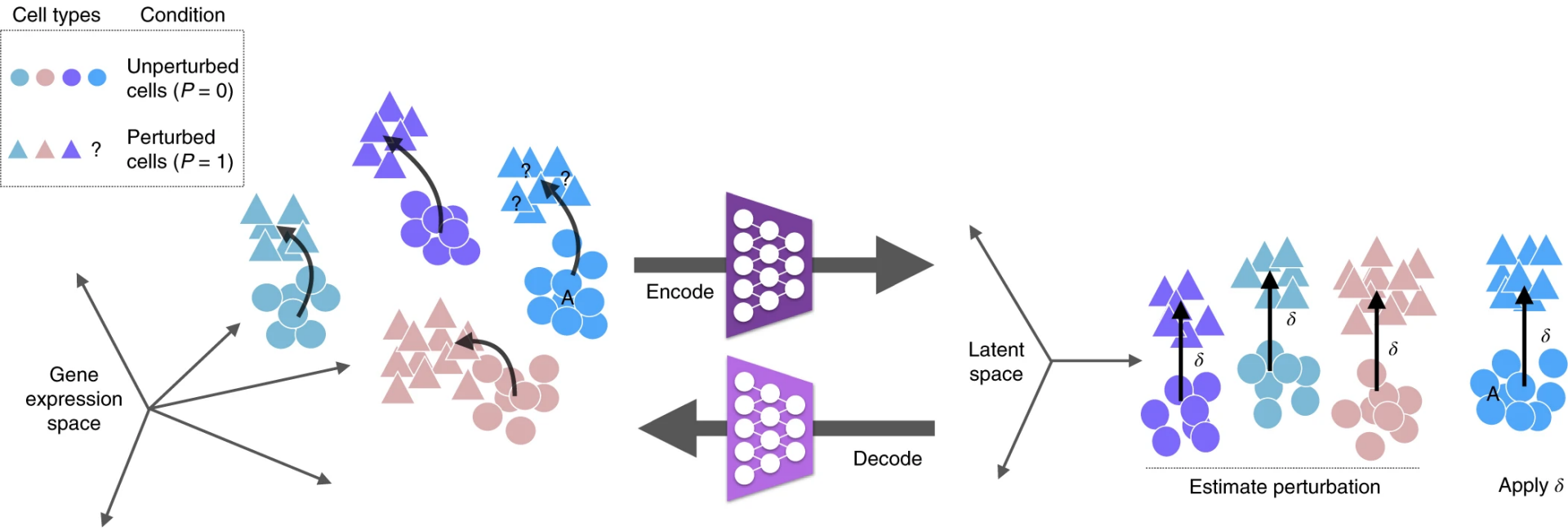
$$
\begin{aligned}
z_n &\sim \text{Normal}\,(0, I) \\
\ell_n &\sim \text{log normal}\left(\ell_\mu, \ell_\sigma^2\right) \\
\rho_n &= f_w\,(z_n, s_n) \\
w_{ng} &\sim \text{Gamma}\left(\rho_n^g, \theta\right) \\
y_{ng} &\sim \text{Poisson}\left(\ell_n w_{ng}\right) \\
h_{ng} &\sim \text{Bernoulli}\left(f_h^g\,(z_n, s_n)\right) \\
x_{ng} &= \begin{cases} y_{ng} \text{ if } h_{ng} = 0 \\ 0 \text{ otherwise} \end{cases}
\end{aligned}
$$

Latent Space Arithmetic



scGEN

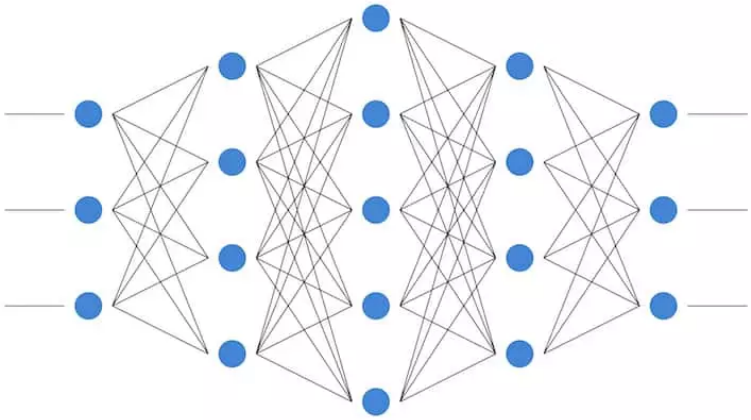
- Generates single cell response by doing latent space arithmetic



Perturbations are modelled as shifts

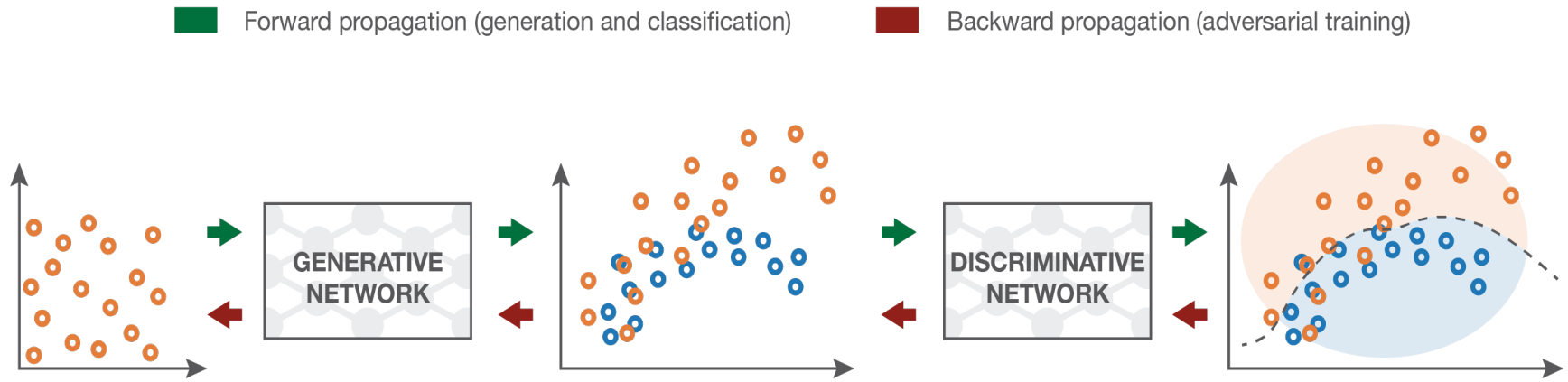
Generative Adversarial Networks

Noise



Something “like” training sample

GAN functionality



Input random variables.

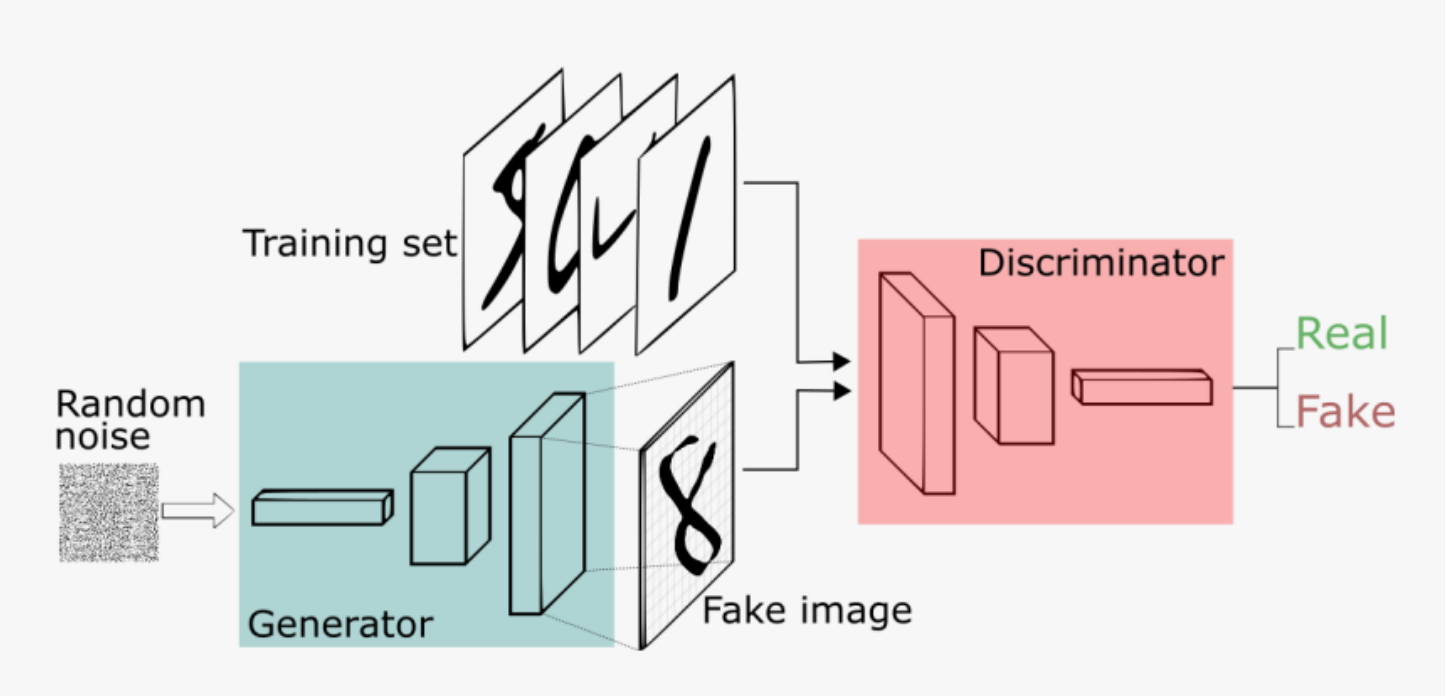
The generative network is trained to **maximise** the final classification error.

The generated distribution and the true distribution are not compared directly.

The discriminative network is trained to **minimise** the final classification error.

The classification error is the basis metric for the training of both networks.

GANs

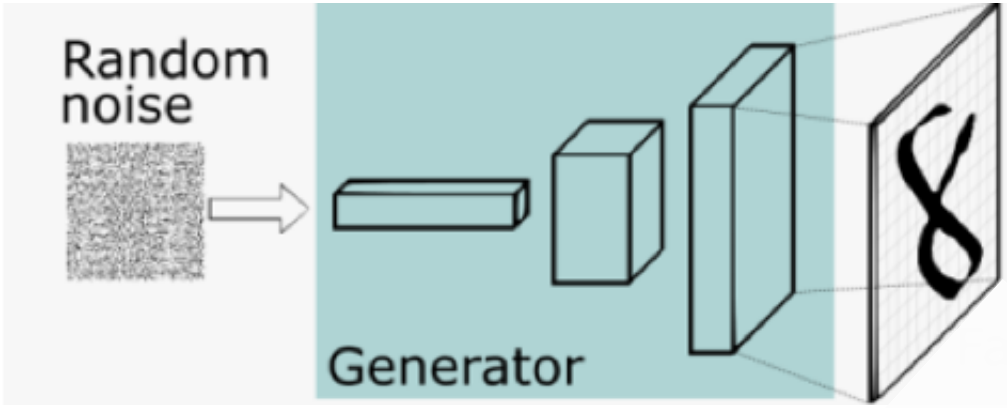


- Sample from an easy noise distribution (uniform, Gaussian)
- One network learns a mapping G(**Z**)
- Second network tries to distinguish true examples of **X** from “fake” samples G(**Z**)

2 Networks

- **Generator**: takes random noise and transforms into sample from target distribution
- **Discriminator:** discriminates between generated (“fake”) examples and real examples
- The two networks play a “minimax game”
- Indirect way of matching distributions

Only use generator after training



- Generator network takes noise and transmutes them into samples from the data distribution
- Implicitly learns high dimensional distributions

Generated Examples

GAN Loss function

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

Maximize discriminator value (1) on real samples

Maximize discriminators value (0) on generated samples

Generator wants the discriminator to do badly!

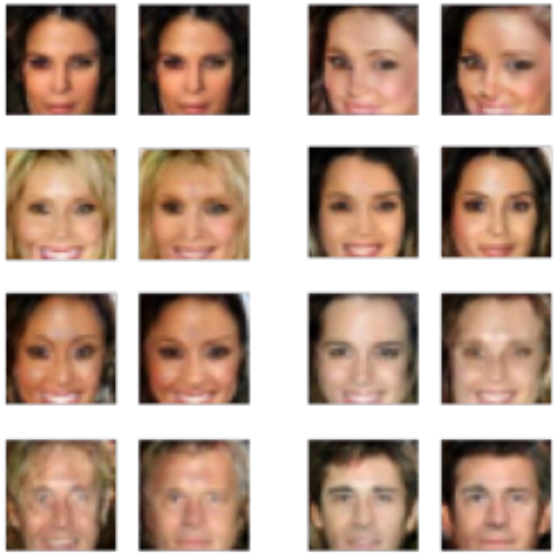
GAN Loss function

$$min_G\, max_D\ E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

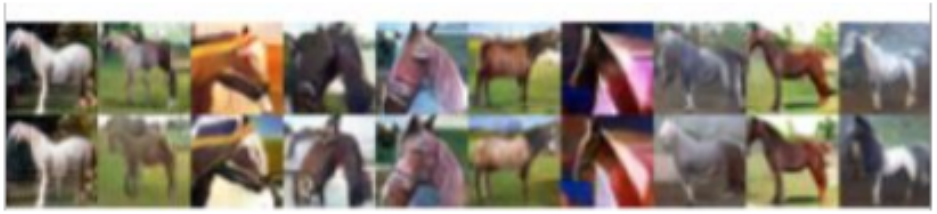
This is the minimax game that the two adversaries play

Mode Collapse

The generator can fool the discriminator by memorizing a small number of images.



evidence of lack of diversity



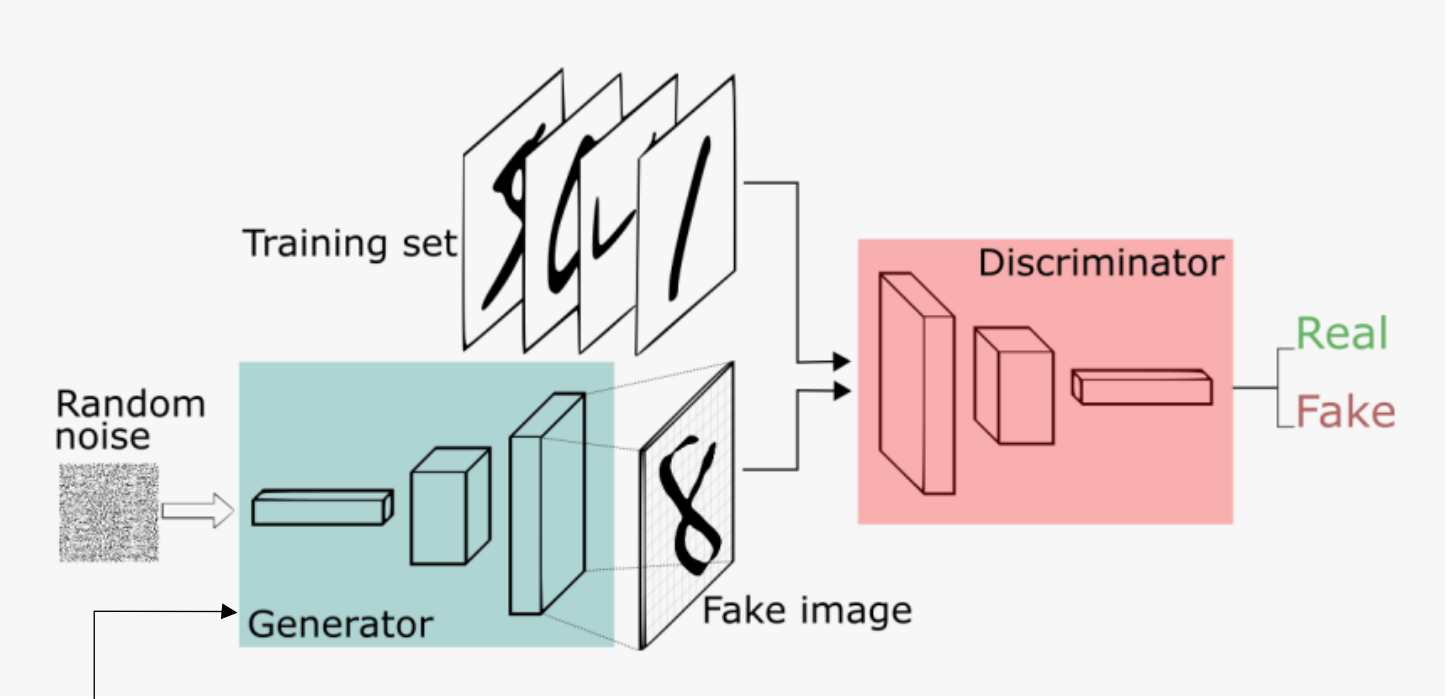
GANs can be hard to train

- Instability during training
- Mode collapse
  - Take the whole batch as a training sample
  - Penalize for a property of the distribution (mean, variance)

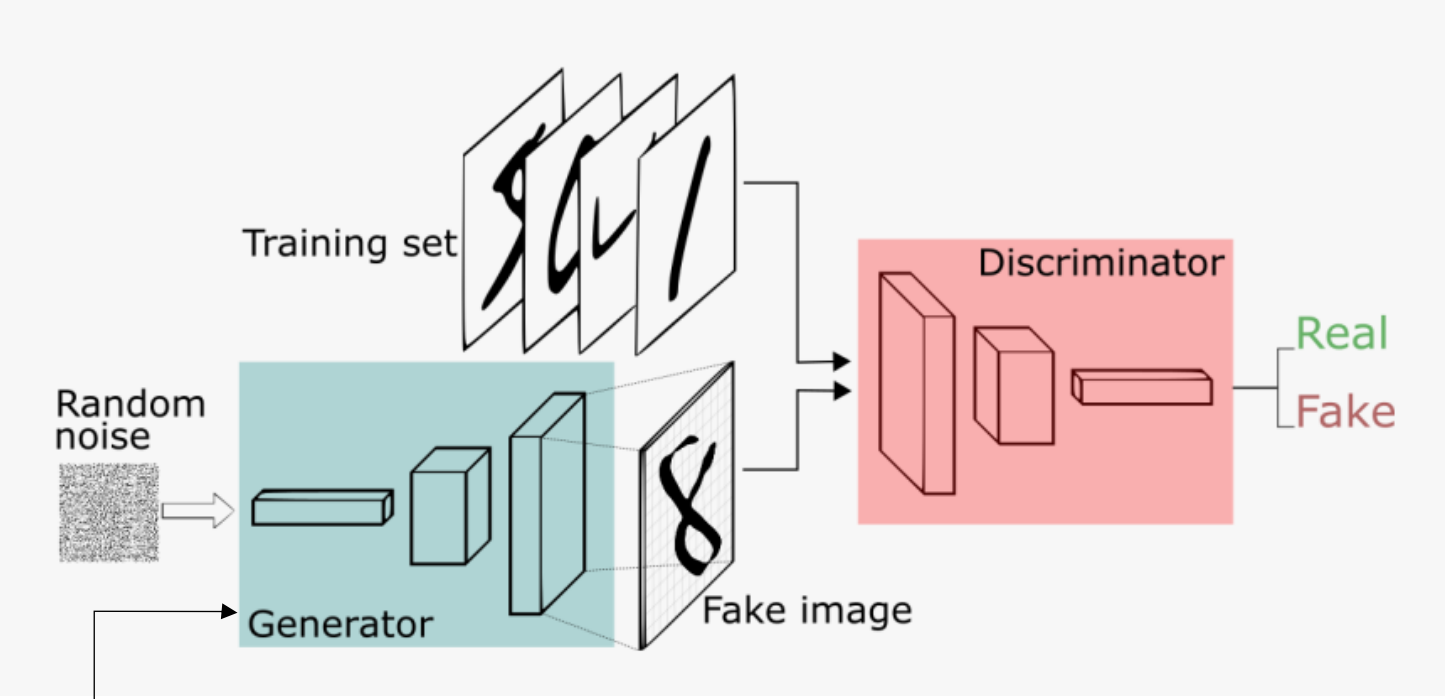
Conditional GANs

- GANs take noise z and turn it into our data x
- Great! But what if we want to control it?
  - “It generated ***another*** image of a 3? I wanted a 7 this time!”
- Conditional GANs:
  - If we have labels, we want to teach it to generate not just any x, but an x of a particular label when given some noise z

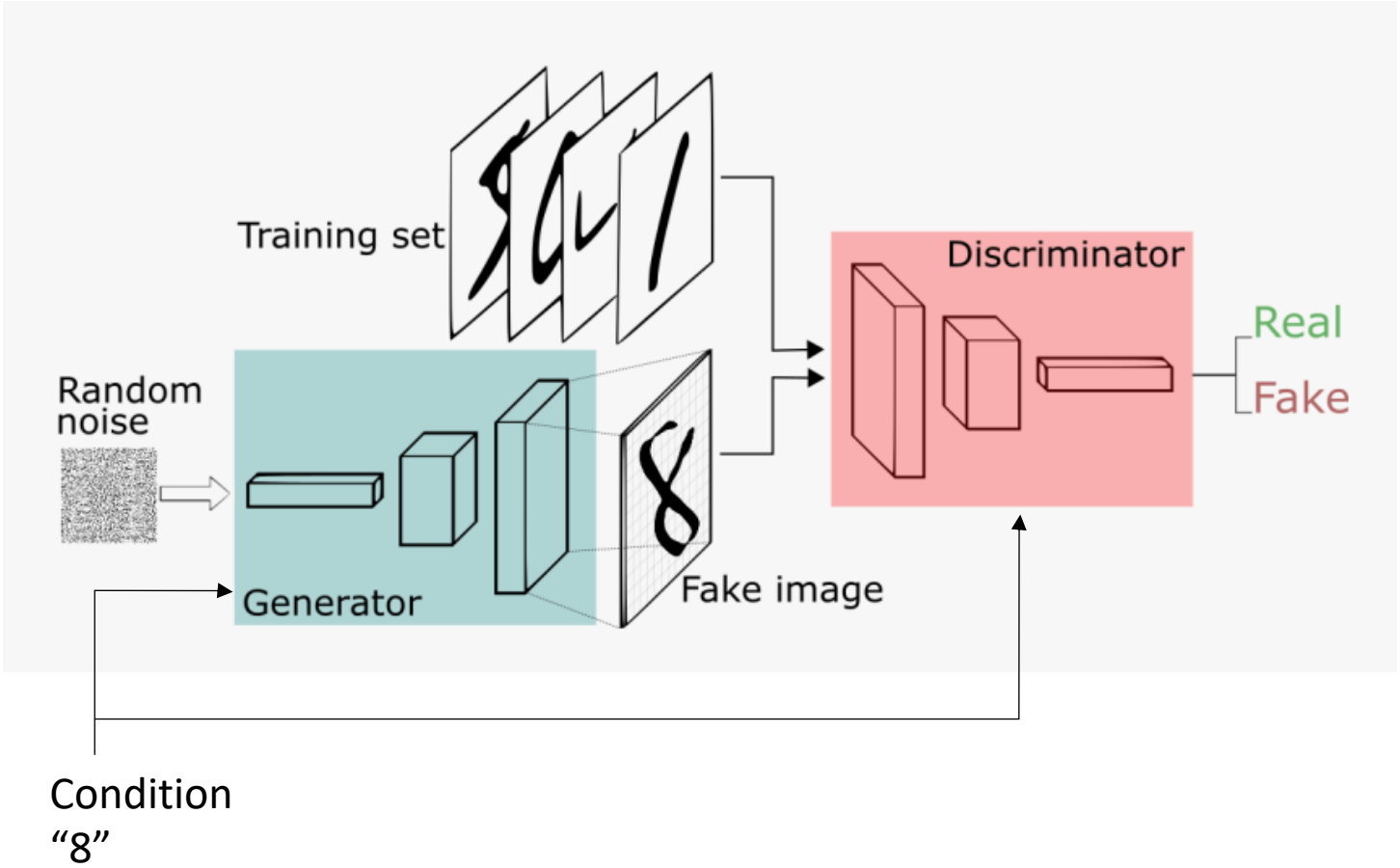
Conditional GANs



Conditional GANs



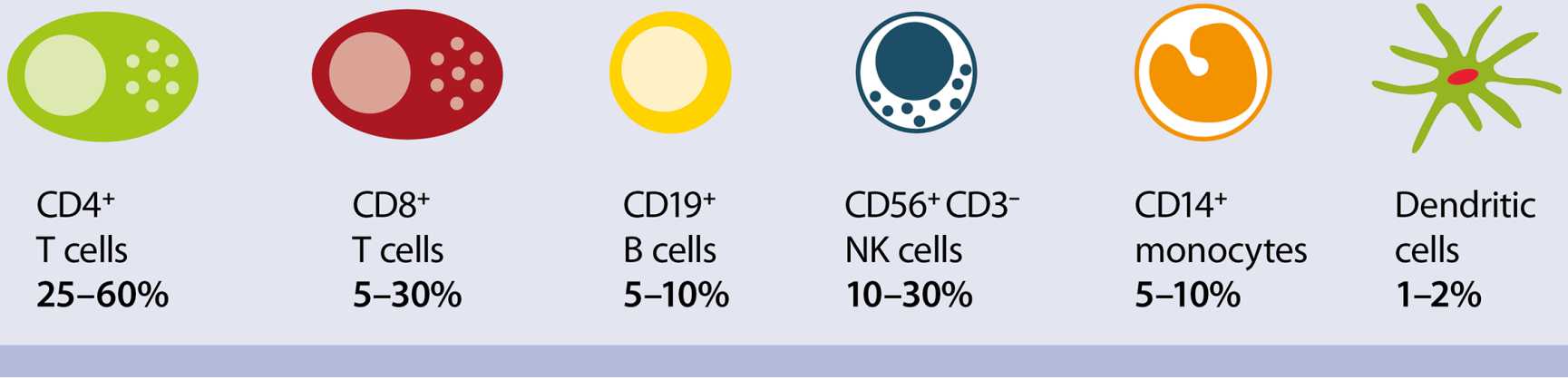
Conditional GANs



Both generator and discriminator look at the condition

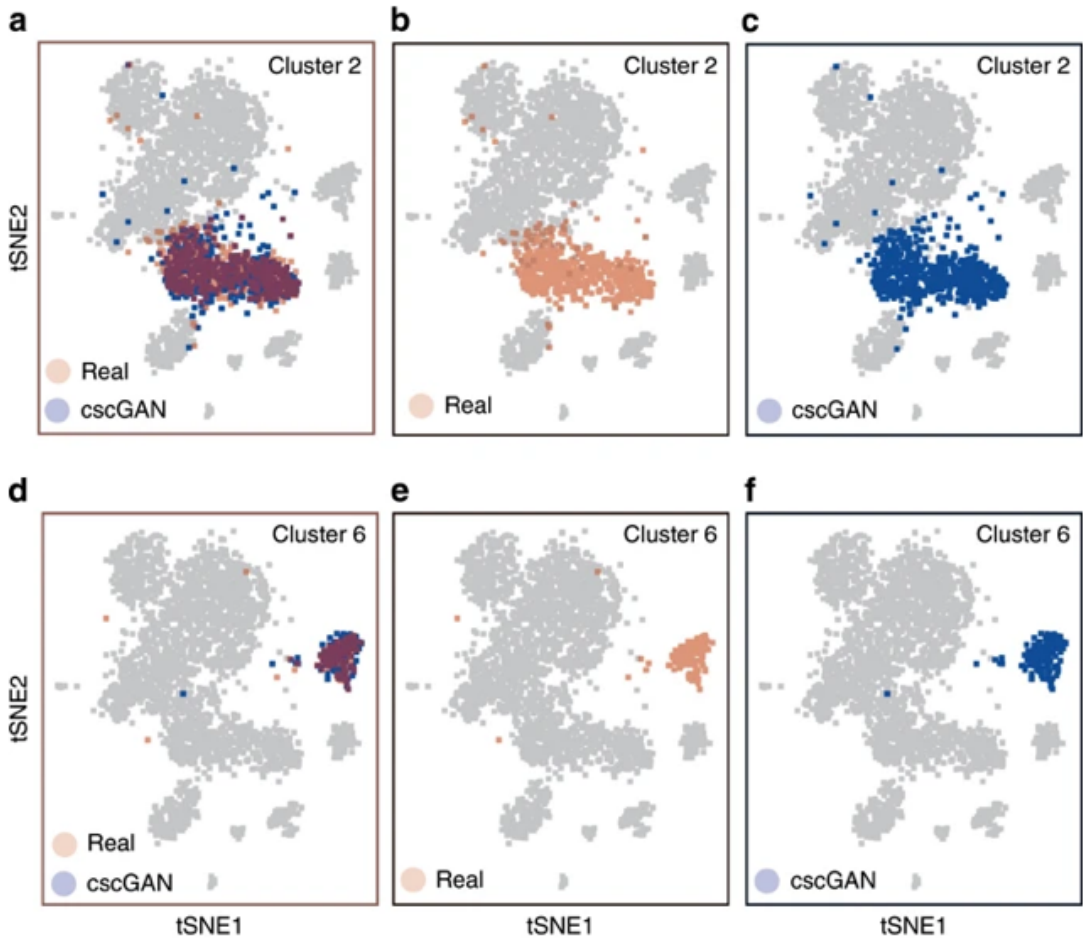
cscGAN

- Maps from noise to single cell measurments
- Conditions correspond to cell types in PBMCs



Some cell types are naturally rare, may want to augment measurements with “in silico” cell generation

cscGAN



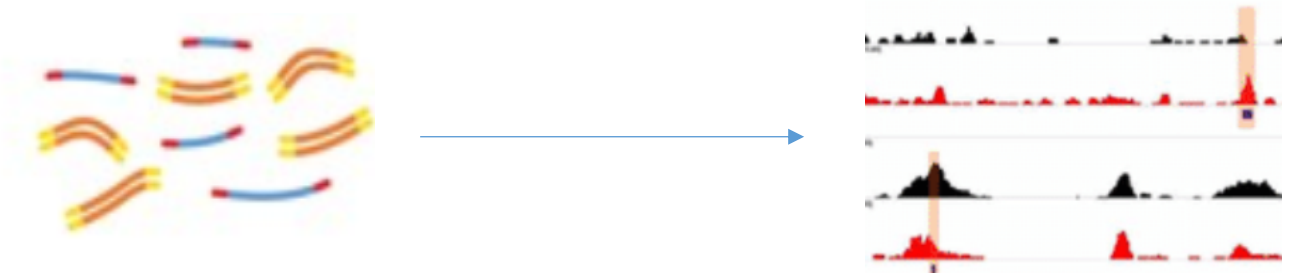
Generated cells visualized on tSNE

Domain Adaptation

- GANs and adversarial AEs sample from one domain
- What if we have two domains, A and B?
  - A=single cell RNAseq, B= single cell ATACseq
  - A=healthy, B=diseased
  - A=control, B=perturbed
  - A=batch1, B=batch 2?

Unsupervised matching

- Can we learn to turn a point in A into a point that looks like its from B?
- We don’t have the “answer” or corresponding measurement in the other modality



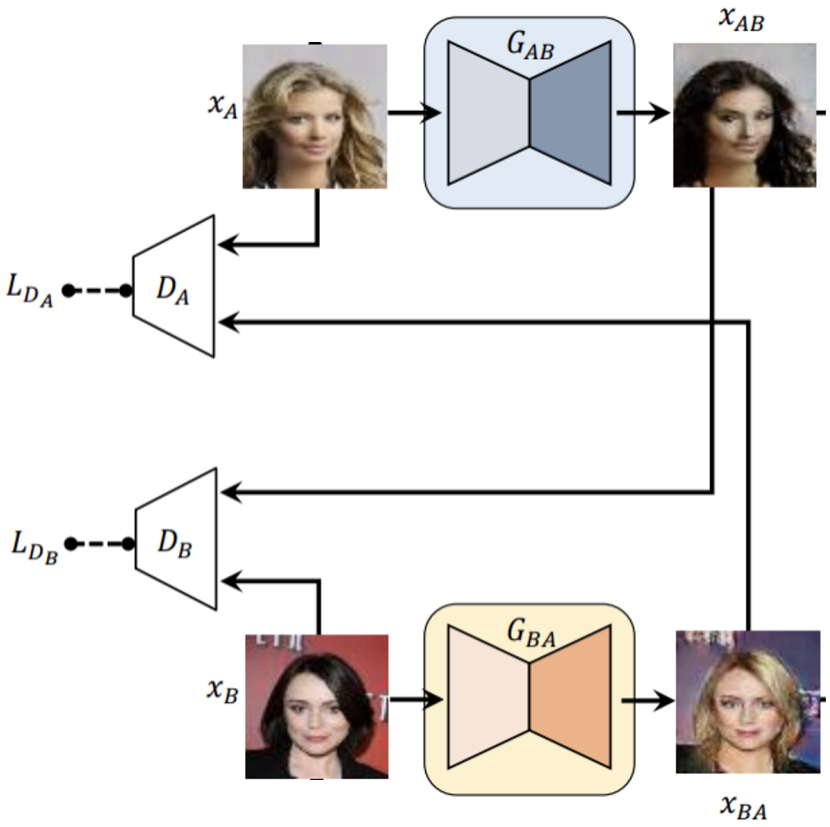
- CyceGAN frameworks were set up to do this

DiscoGAN: Motivation

- Domain A: blond hair

- Domain B: black hair

Previous approach

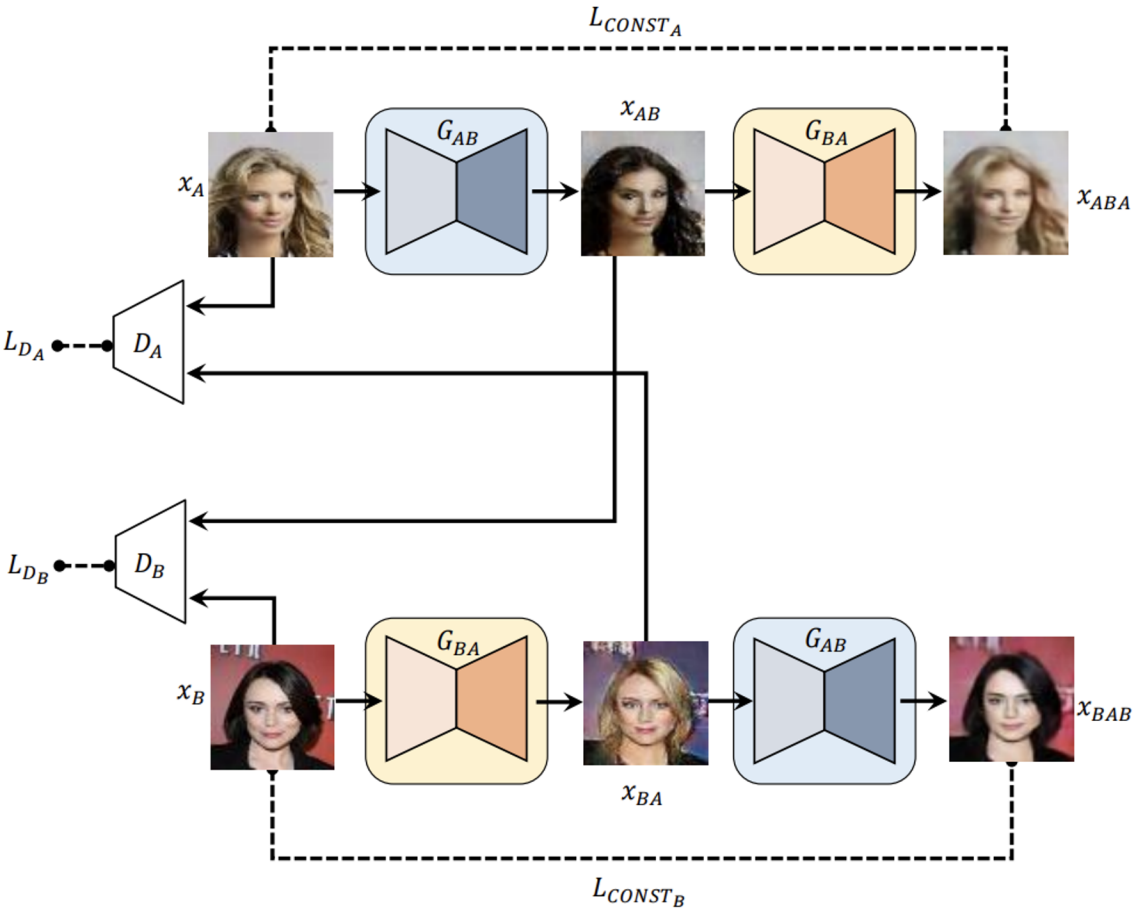


Kim, Taeksoo, et al. "Learning to discover cross-domain relations with generative adversarial networks." *arXiv preprint arXiv:1703.05192* (2017).

DiscoGAN

- Domain A: blond hair

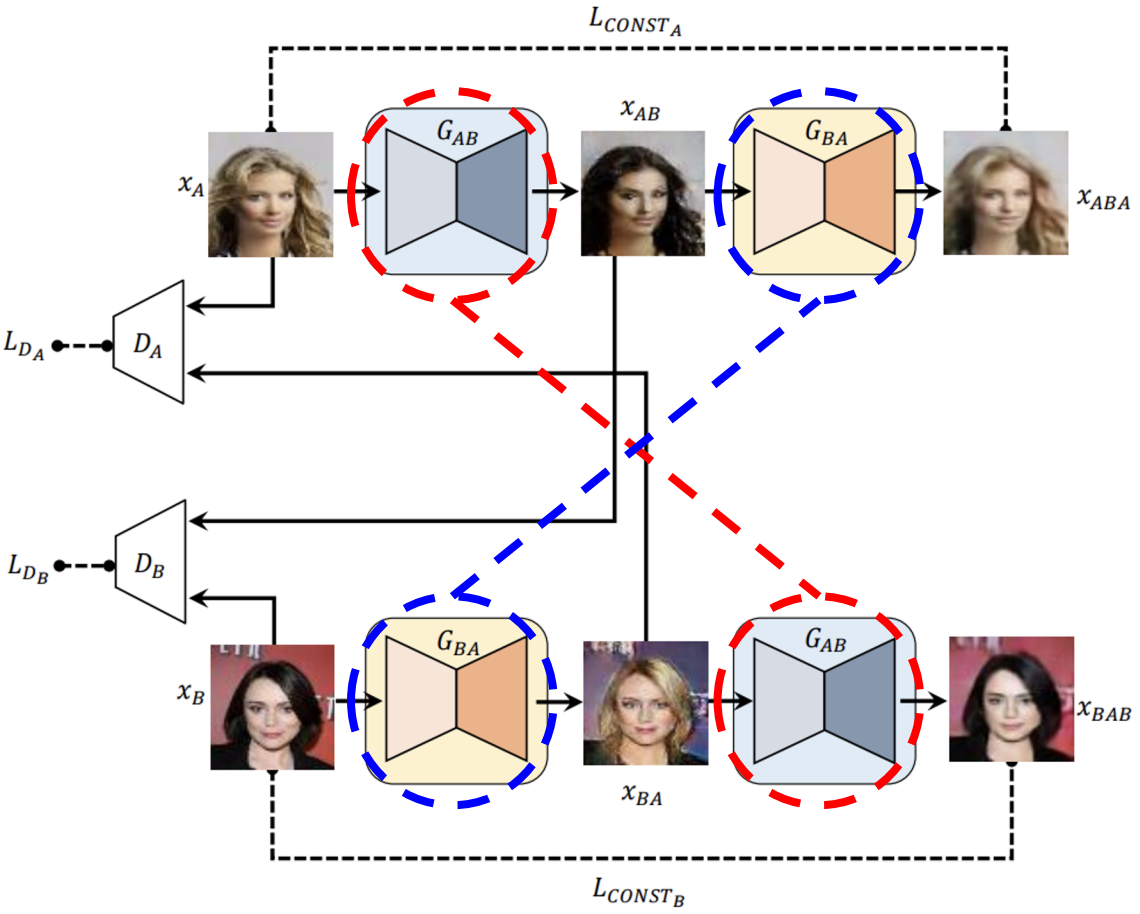
- Domain B: black hair



Kim, Taeksoo, et al. "Learning to discover cross-domain relations with generative adversarial networks." *arXiv preprint arXiv:1703.05192* (2017).

DiscoGAN

- Weight sharing/reuse is important
  - Same reason CNNs work
- Same network weights/activations must perform multiple tasks
  - Indirectly increases number of training samples and generalizability
- Reduces number of weights to train
  - Optimization is easier



Kim, Taeksoo, et al. "Learning to discover cross-domain relations with generative adversarial networks." *arXiv preprint arXiv:1703.05192* (2017).

DiscoGAN

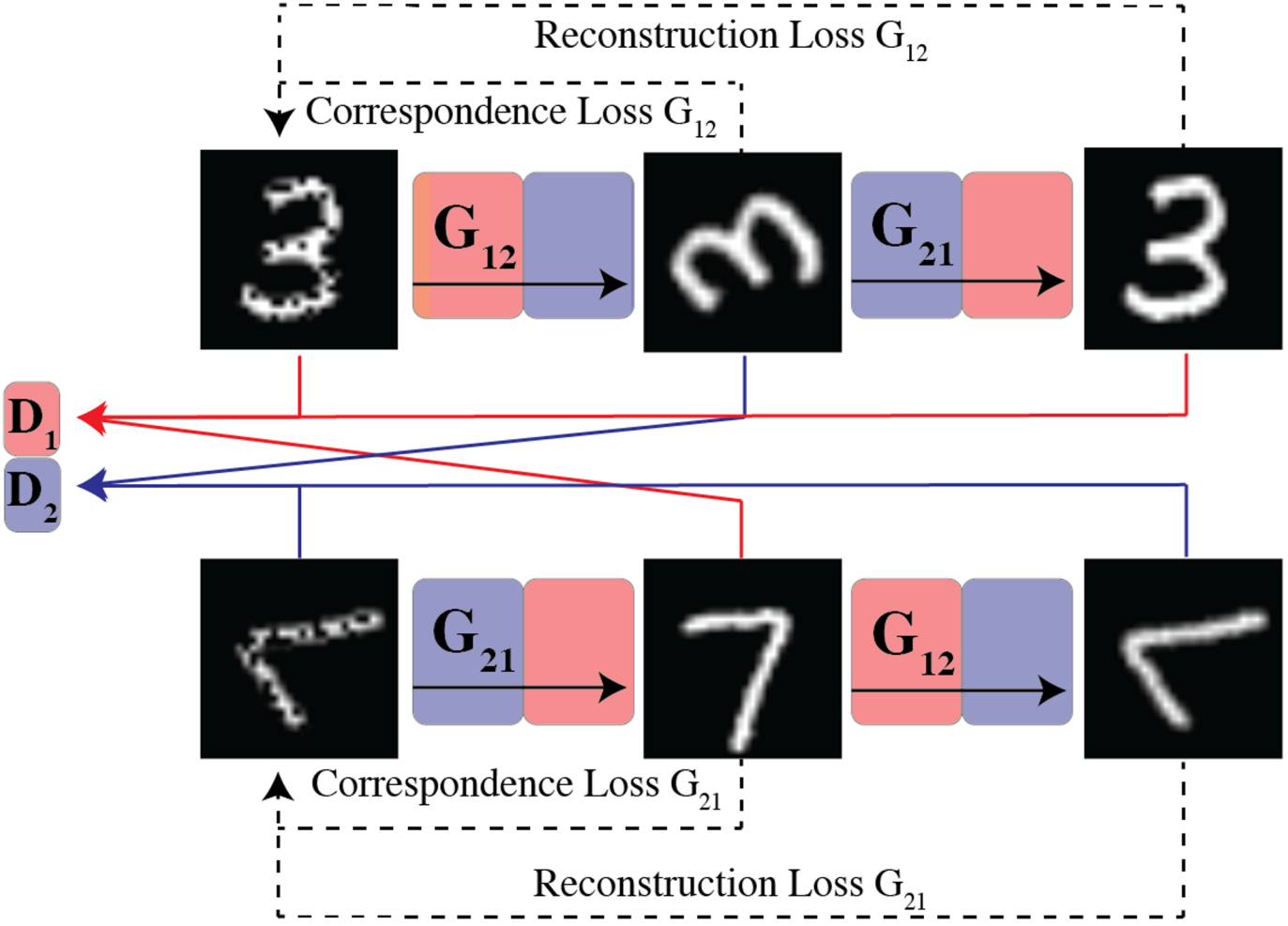
DiscoGAN

Issues

- Is this the correct shoe for this purse?
- Is there a better match?
- Original paper doesn't talk about it

- Sometimes the mapping isn't correct

MAGAN Architecture

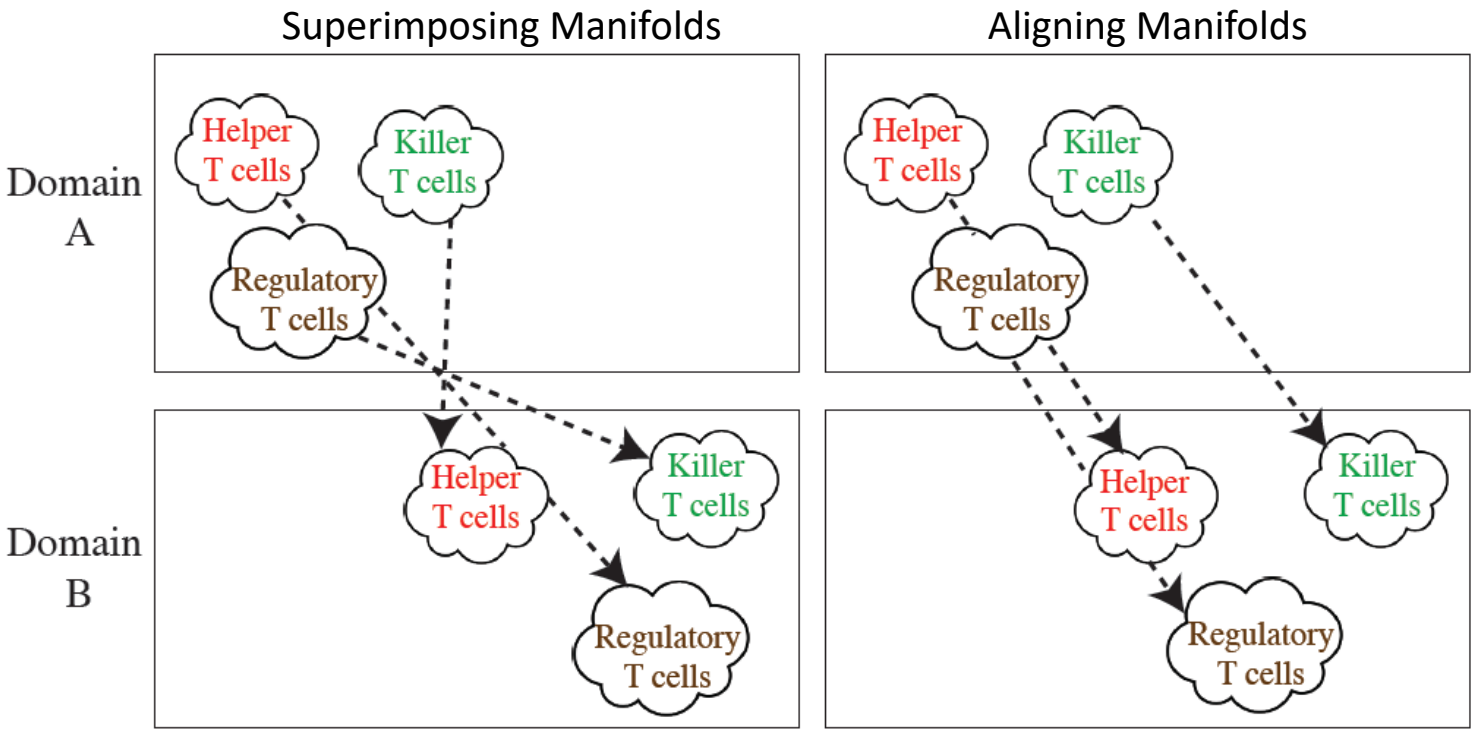


MAGAN

- Ensures that cells translated from domain A to domain B still correspond in some way
- This is enforced by a correspondence loss
- Correspondence loss formulation requires domain specific knowledge
- How can we tell if a normal cell and a stimulated cell could correspond?
  - If the cell type is the same (stimulated T cells are still T cells)

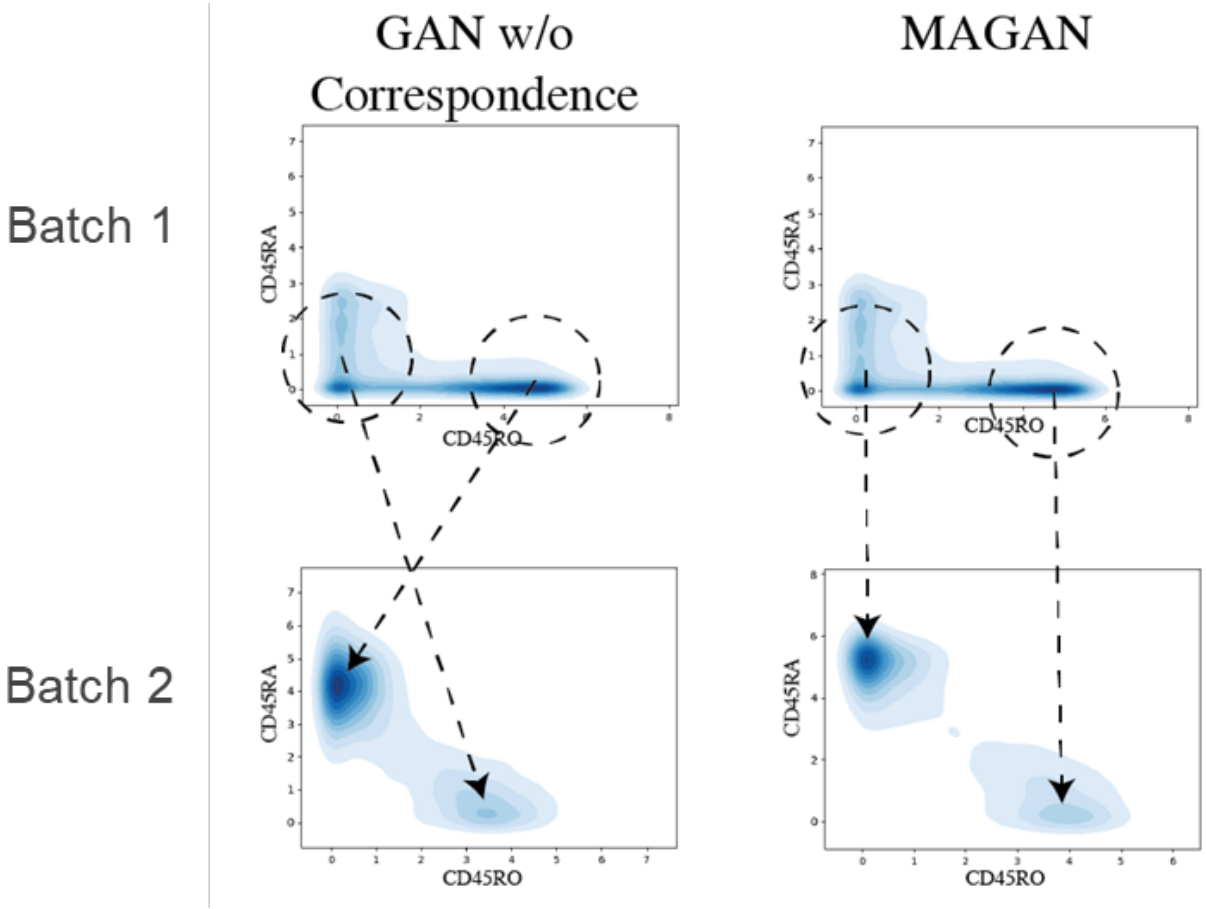
Combining 2 cytof Panels

- 2 Panels A and B
- Each panel has 40 markers, 10 are shared
- Want a dataset with 30+10+30=70 markers!
- For each cell measured with panel A, generate panel B
- Correspondence loss: difference between shared markers should be small



Aligning CyTOF Batches

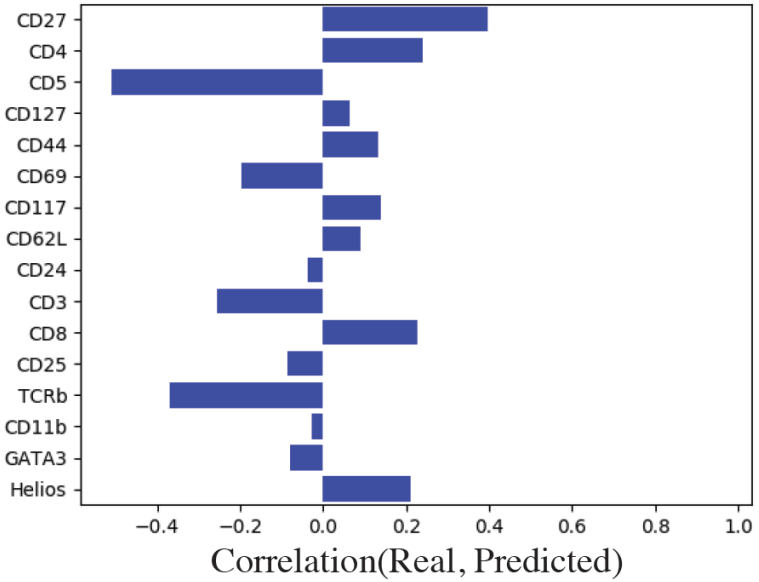
- MAGAN correctly matches the cell types in the two batches



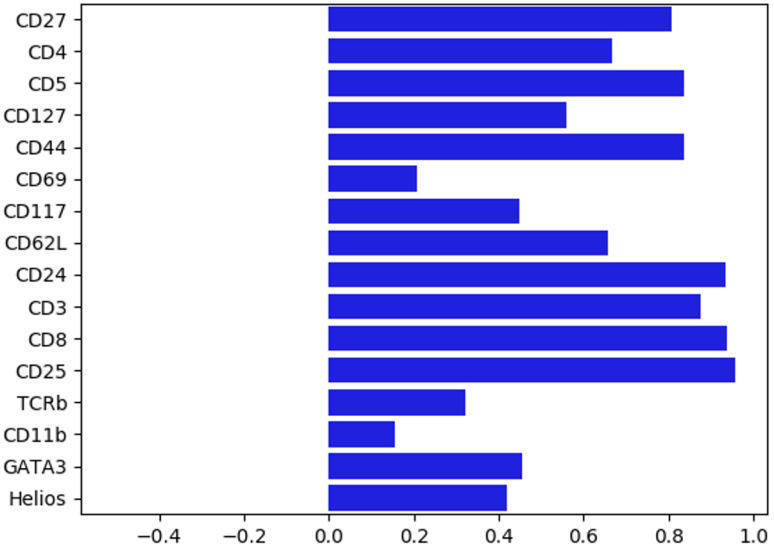
Aligning CyTOF Panels w/ MAGAN

- Validation on publicly available CyTOF data from developing mouse brain*

GAN w/o Correspondence



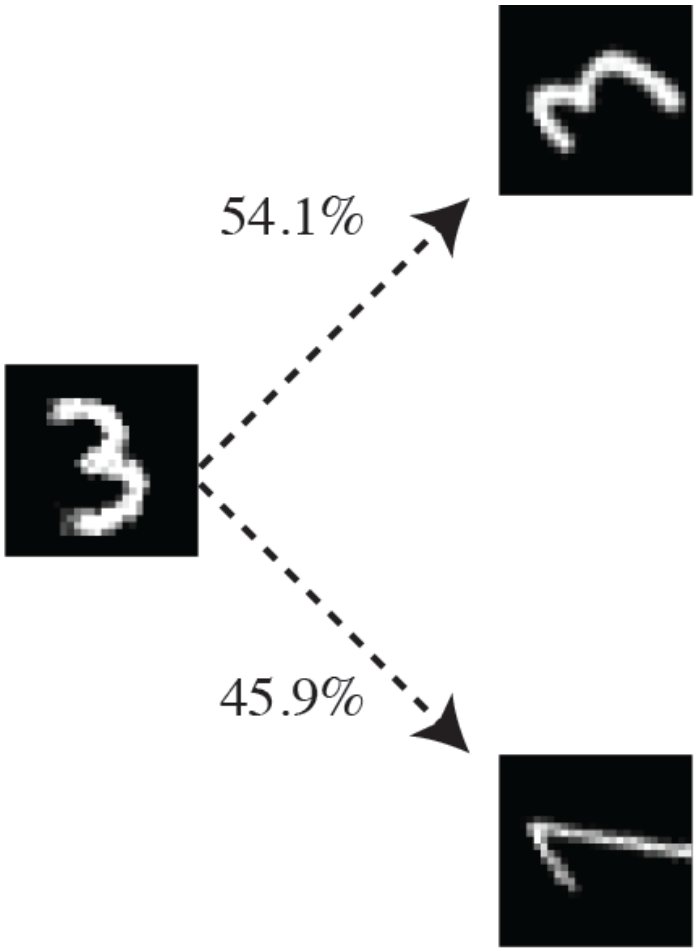
MAGAN



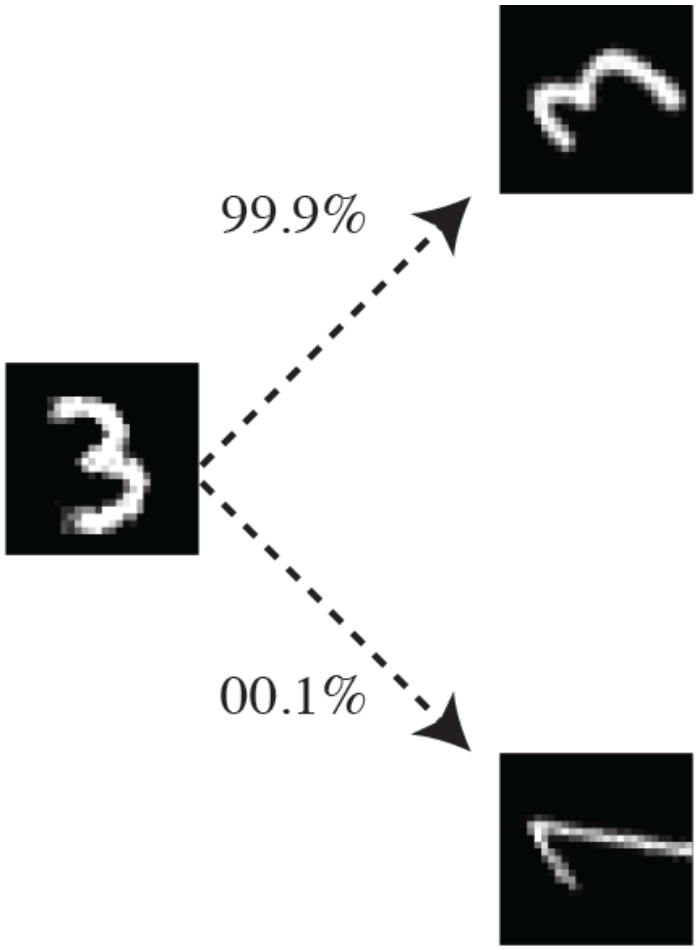
* Setty, Manu et al. "Wishbone identifies bifurcating developmental trajectories from single-cell data". *Nature Biotechnology.*

Matching rotated digits

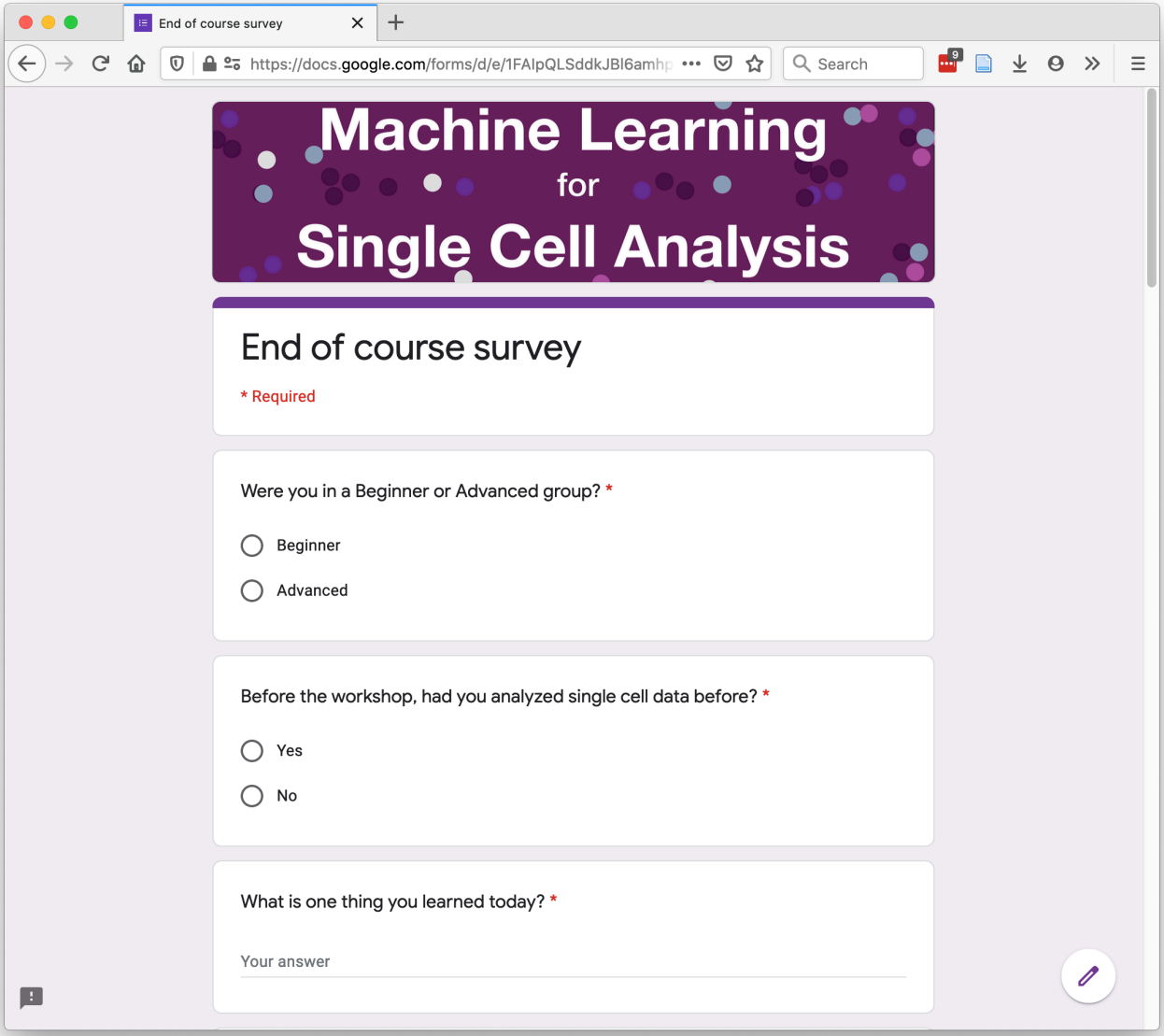
- 100 simulations of each model



GAN w/o Correspondence



MAGAN



https://bit.ly/YaleML_May_PostSurvey