# BST 261: Data Science II
# Lecture 15
## Advanced Topics:
## Variational Autoencoders (VAEs),
## Generative Adversarial Networks (GANs),
## Reinforcement Learning (RL)

Heather Mattie
Harvard T.H. Chan School of Public Health
Spring 2 2020

# Recipe of the Day!

Spiced Sweet Potato & Poblano Tostadas

# Paper Presentations

# Generative Adversarial Networks for Electronic Health Records:

## A Framework for Exploring and Evaluating Methods for Predicting Drug-Induced Laboratory Test Trajectories

Authors: Alexandre Yahi, Rami Vanguri, Noémie Elhadad, Nicholas P. Tatonetti

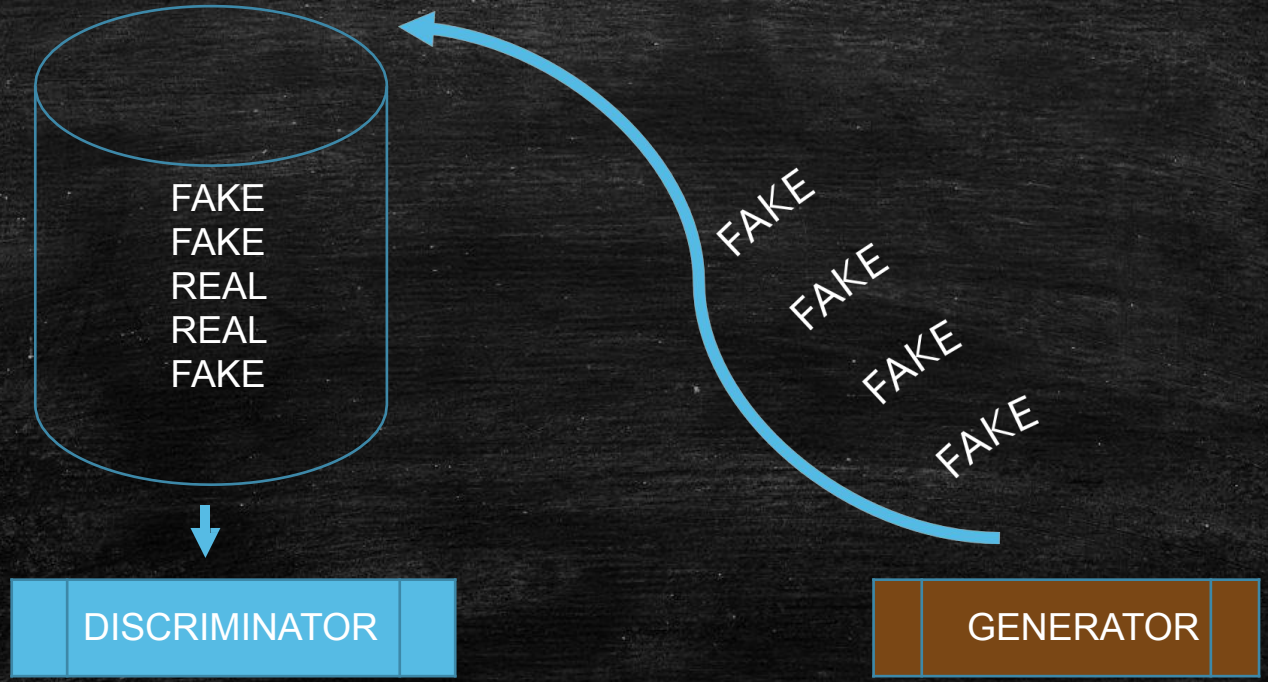
*Department of Biomedical Informatics, Columbia University, New York, USA*

BST 261
Paper Presentation
Pooja Tyagi

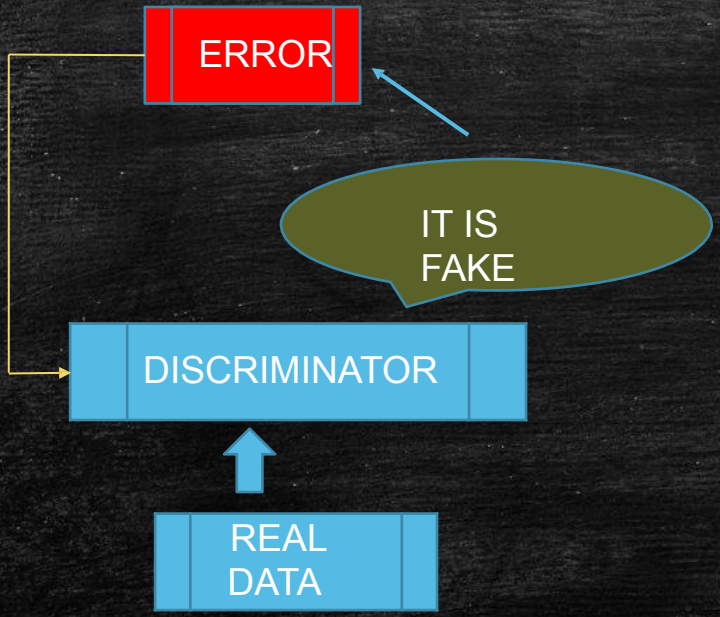
## Motivation

- Question:
  - Based on a patient's lab results history, can we predict how they
    would respond to a drug?
- Approach:
  - Train a Generative Adversarial Network (GAN) and use it to predict
    patient's lab test trajectory post drug exposure.
- Benefit:
  - Might allow detection of adverse events
  - quantify the impact of a drug on a specific lab test.

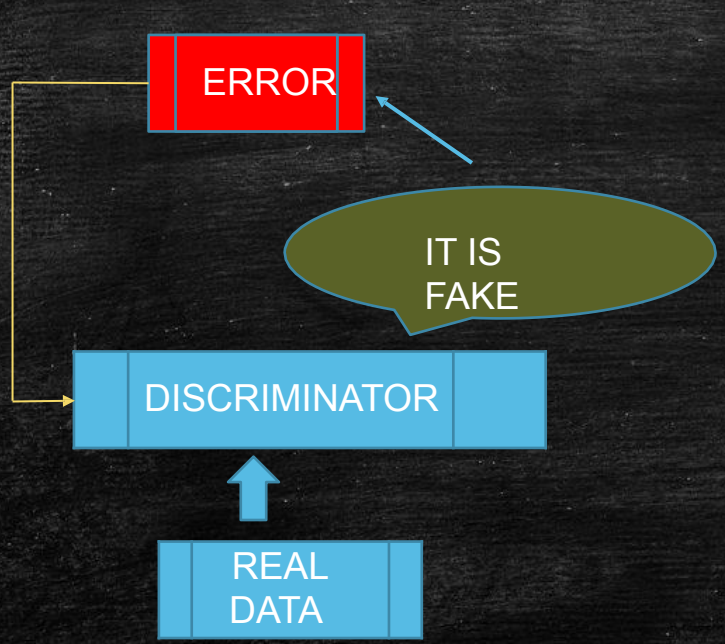
Generative Adversarial Network

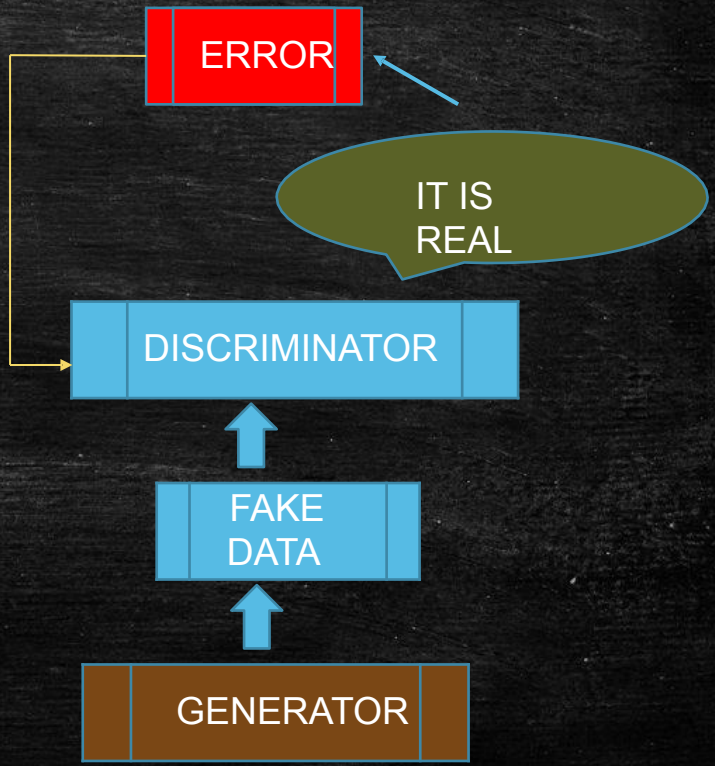


Generative Adversarial Network: Training

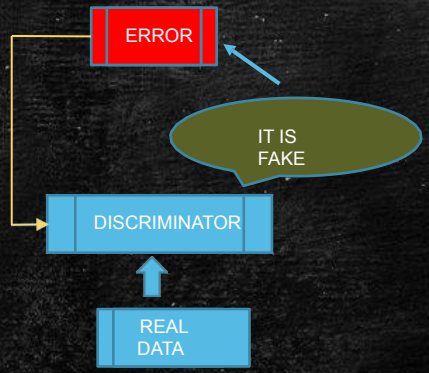


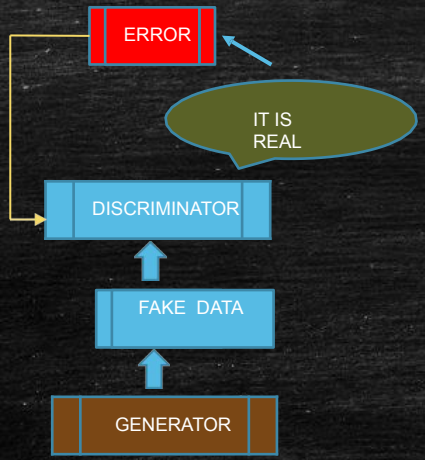
Generative Adversarial Network: Training

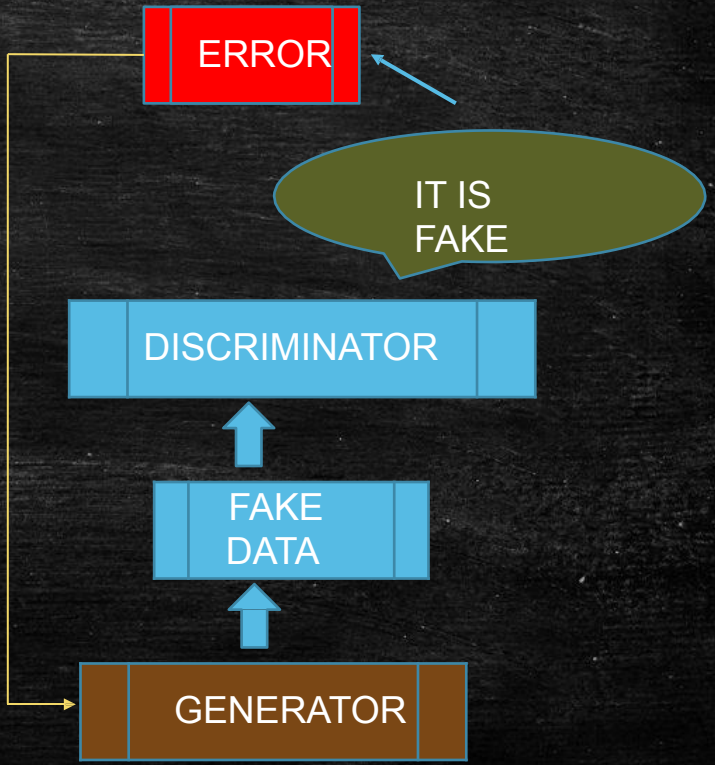


# Generative Adversarial Network: Training



# Data

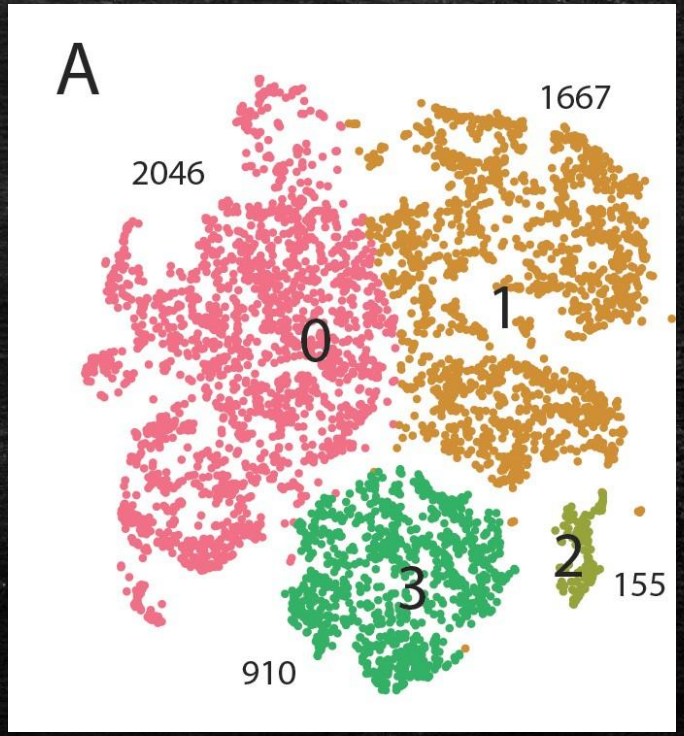
- EHR data were collected at the New York Presbyterian/Columbia  University Irving Medical Center between 2000 and 2013 with 19.6  million drug prescriptions
- 485,306 patients and 473.6 million laboratory test observations.
- Selected all the patients exposed to statins at any point in time.
- Collected all cholesterol measurements.

## Data Preprocessing

- Each patient contributes a time series of cholesterol lab tests.
- Annotate the time series as pre-exposure and during exposure to
  statins.
- Exclude patients
  - with no measurement during drug exposure, or
  - without measurements within a year before exposure
- Final dataset = 4830 patients
- 16 time points per patient: 8 before starting on statins and 8 after

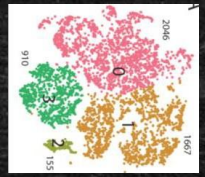
## T-SNE and spectral clustering

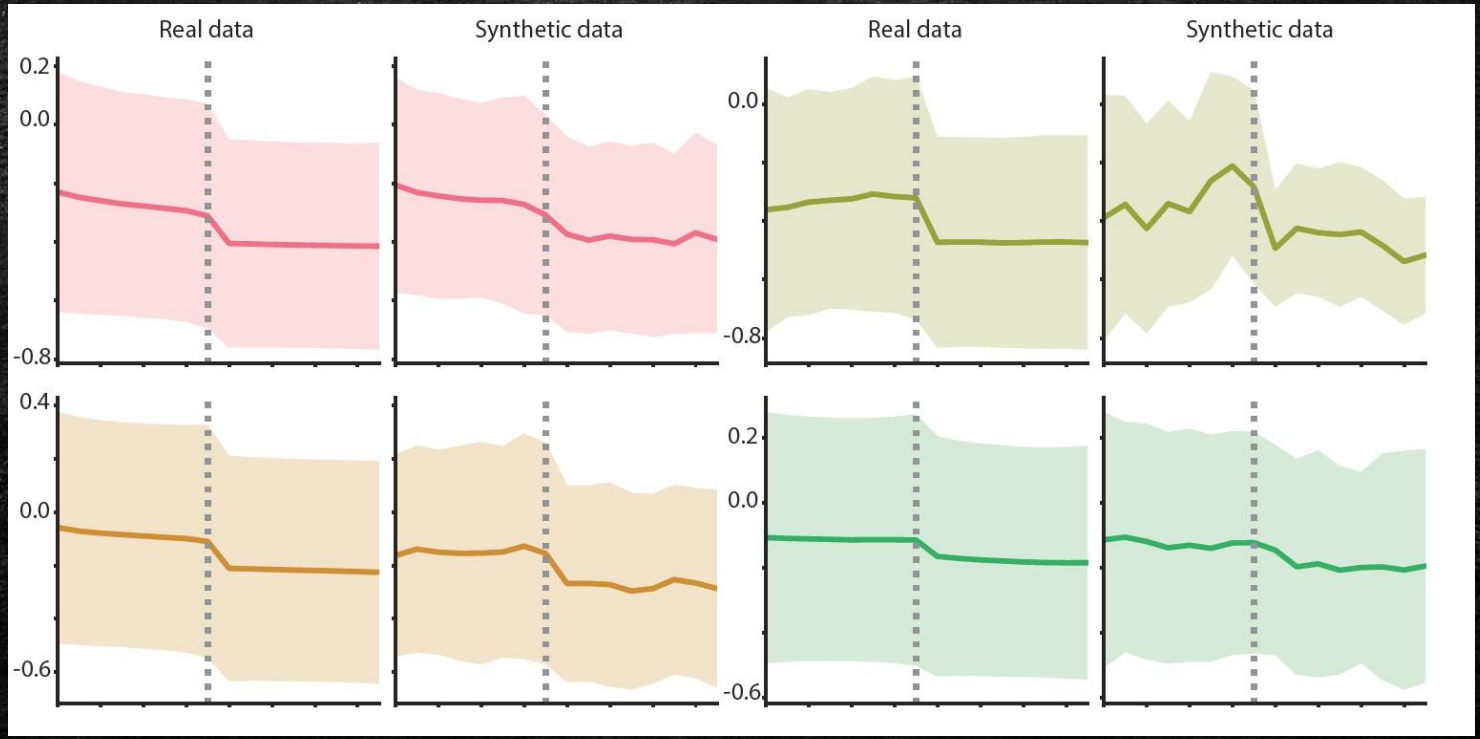
- Separate patient population into clusters based on:
  - Drug prescriptions
  - Diagnoses




* Cluster 3: consists of only diabetic patients.

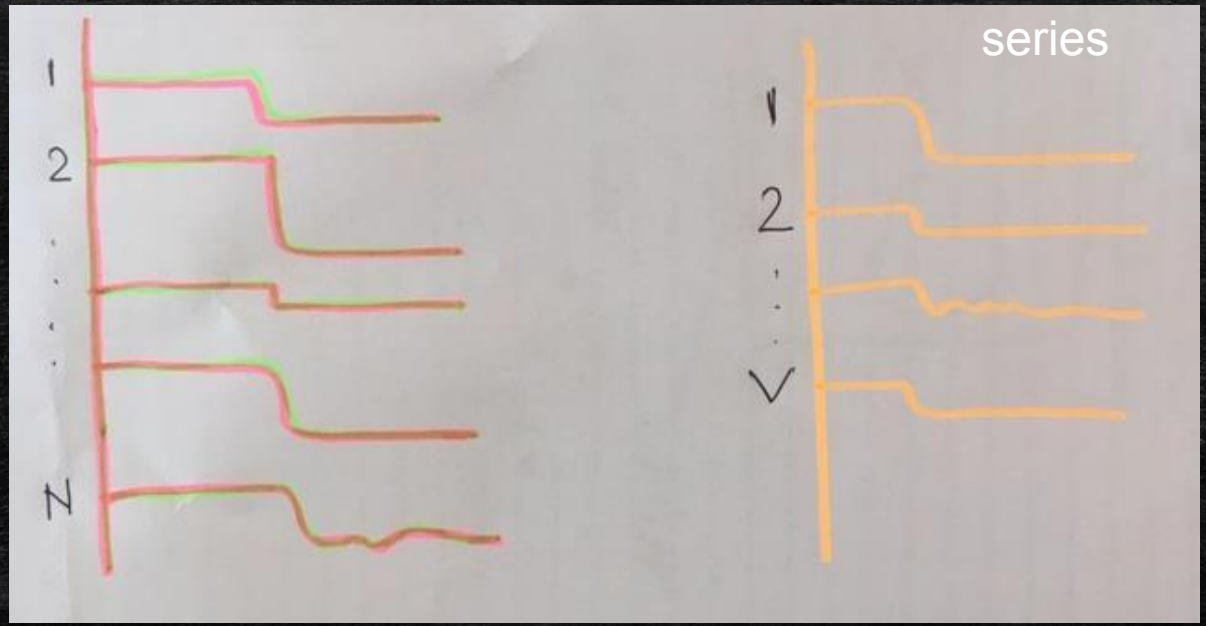
Generated time series using GANs



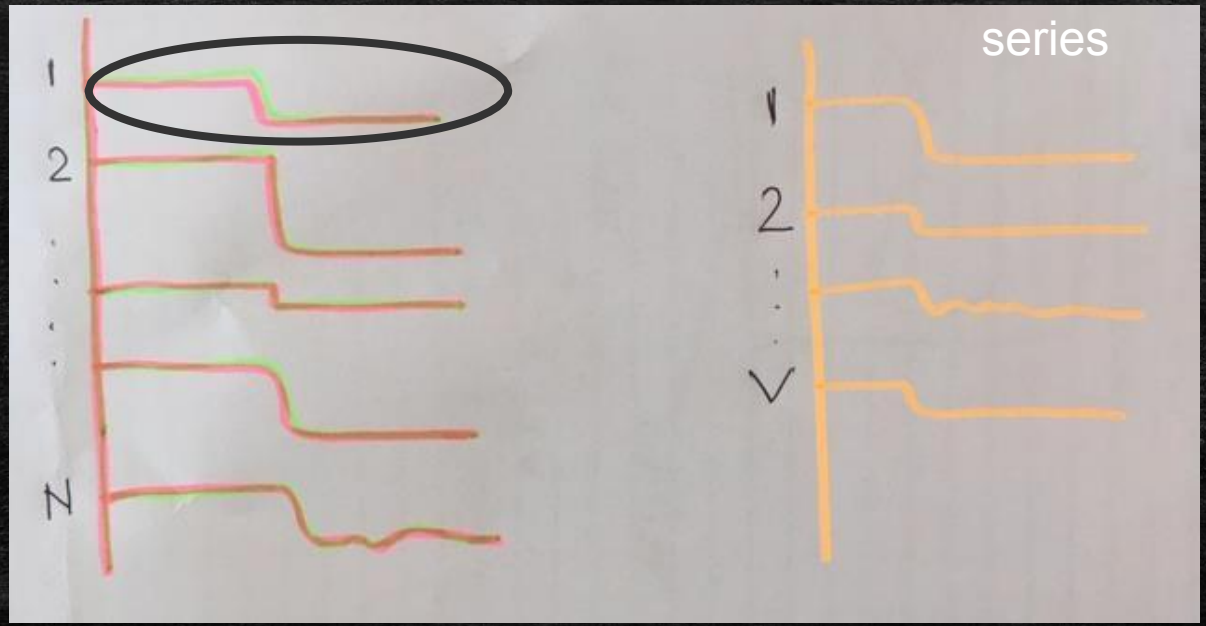
Prediction Error to Evaluate GAN Performance

Choose one of N series → *Loop over generated series* → Select the most realistic of all time series



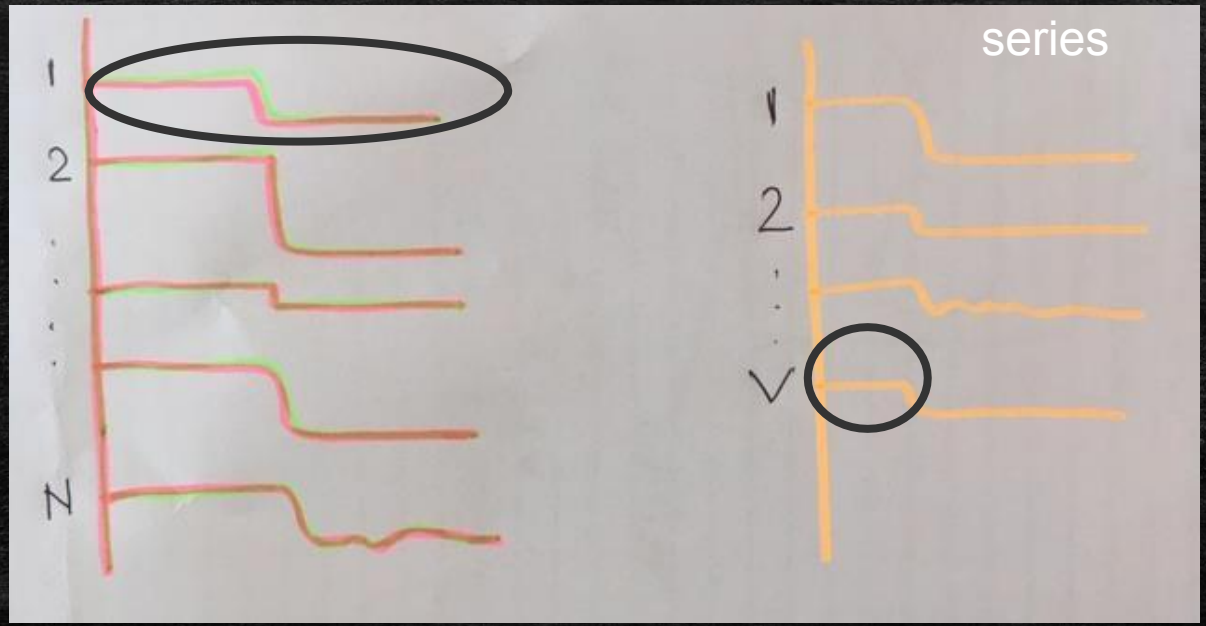
Prediction Error to Evaluate GAN Performance

Choose one of N series → *Loop over generated series* → Select the most realistic of all time series



Prediction Error to Evaluate GAN Performance

Choose one of N series → *Loop over generated series* → Select the most realistic of all time series



Prediction Error to Evaluate GAN Performance

$$P_{err}(S_N, \hat{S}_V) = \frac{1}{N} \sum_{k=1}^{N} \mathrm{MSE}_{exp}(x_k, \arg\min_{\hat{x}_j \in V}(\mathrm{MSE}_{pre}(x_k, \hat{x}_j)))$$

Which pre-exposure time series most closely approximates

## Prediction Error to Evaluate GAN Performance

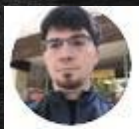
Table 1: Predictivity error $P_{err}$ ($\pm$ SD)

| | Clinical Clusters | | | Random Clusters | | |
|---|---|---|---|---|---|---|
| Cluster | subGAN | totalGAN | p-value | subGAN | totalGAN | p-value |
| Cluster 0 | 0.13 ($\pm$0.22) | 0.16 ($\pm$0.28) | 5.7e-4 | 0.27 ($\pm$0.39) | 0.16 ($\pm$0.27) | 2.0e-303 |
| Cluster 1 | 0.15 ($\pm$0.27) | 0.16 ($\pm$0.26) | 1.5e-1 | 0.30 ($\pm$0.45) | 0.16 ($\pm$0.26) | 2.2e-308 |
| Cluster 2 | 0.11 ($\pm$0.21) | 0.22 ($\pm$0.35) | 3.9e-5 | 0.24 ($\pm$0.40) | 0.15 ($\pm$0.26) | 1.1e-21 |
| Cluster 3 | 0.12 ($\pm$0.20) | 0.15 ($\pm$0.24) | 3.9e-4 | 0.28 ($\pm$0.38) | 0.16 ($\pm$0.27) | 1.5e-144 |

- subGANs predict data better for individual clusters than total GAN.
- subGANs trained on random clusters of identical size perform worse than those trained on t-SNE identified clusters (i.e. clinical features are important).

## Main takeaways

- This paper presented a method to evaluate GAN performance  for the prediction of laboratory test trajectories.
- They showed that by clustering patient population, more  accurate GANs can be produced.



**Ian Goodfellow**
@goodfellow_ian

4.5 years of GAN progress on face generation. arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434 arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196 arxiv.org/abs/1812.04948


Posted on: Jan. 14, 2019

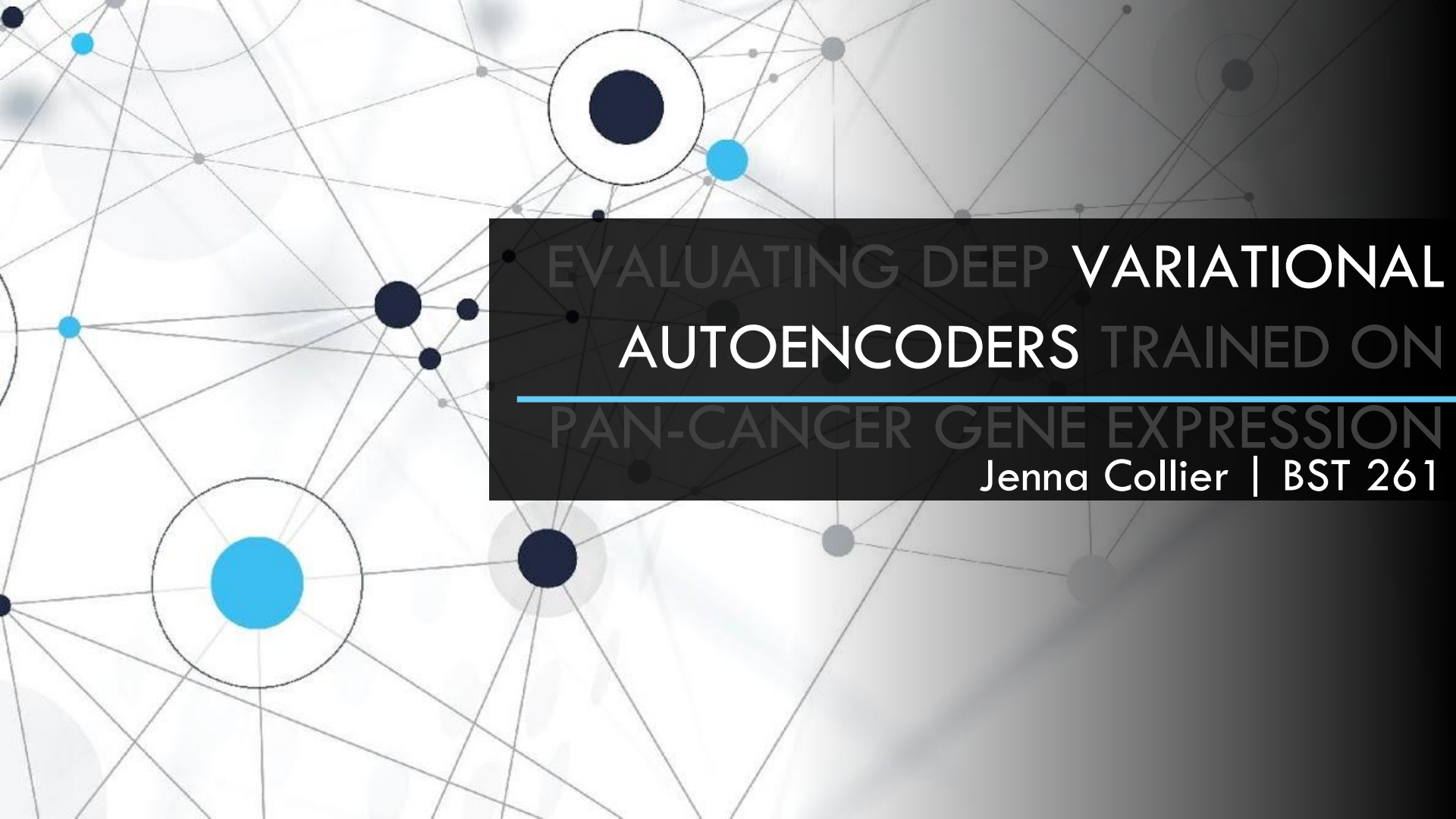
## Tech specs for GAN

- Activation (for reconstruction) : tanh
- Loss: Mean Squared Error (data is continuous)
- Generator: 16 neurons in one layer.
- Discriminator: 2 layers with 32 and 16 neurons.
- Epochs = 100
- Batch size* = 10

(* batch size = 5 for the smallest cluster)

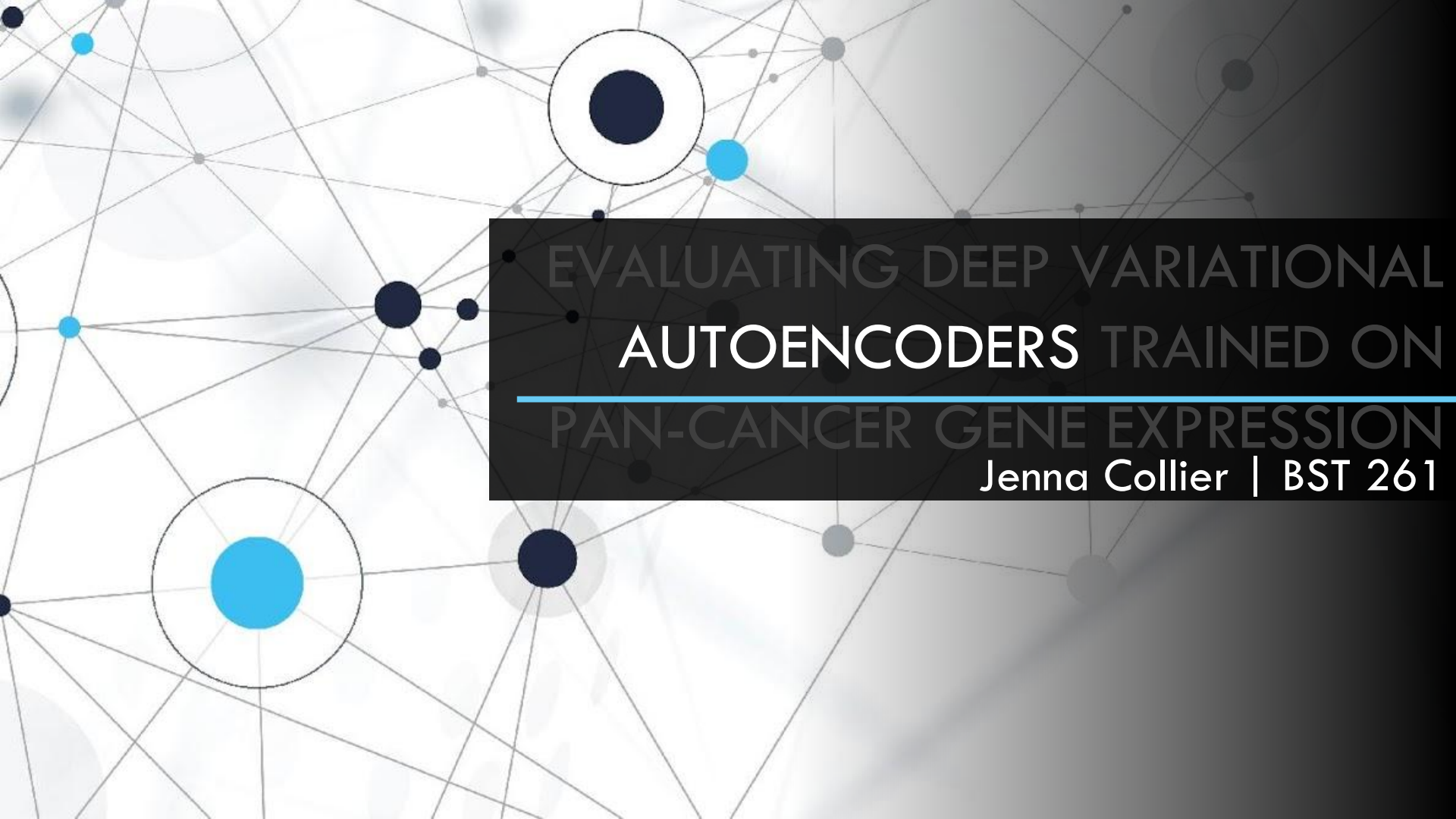
# EVALUATING DEEP VARIATIONAL AUTOENCODERS TRAINED ON PAN-CANCER GENE EXPRESSION

Jenna Collier | BST 261



# EVALUATING DEEP VARIATIONAL AUTOENCODERS TRAINED ON PAN-CANCER GENE EXPRESSION

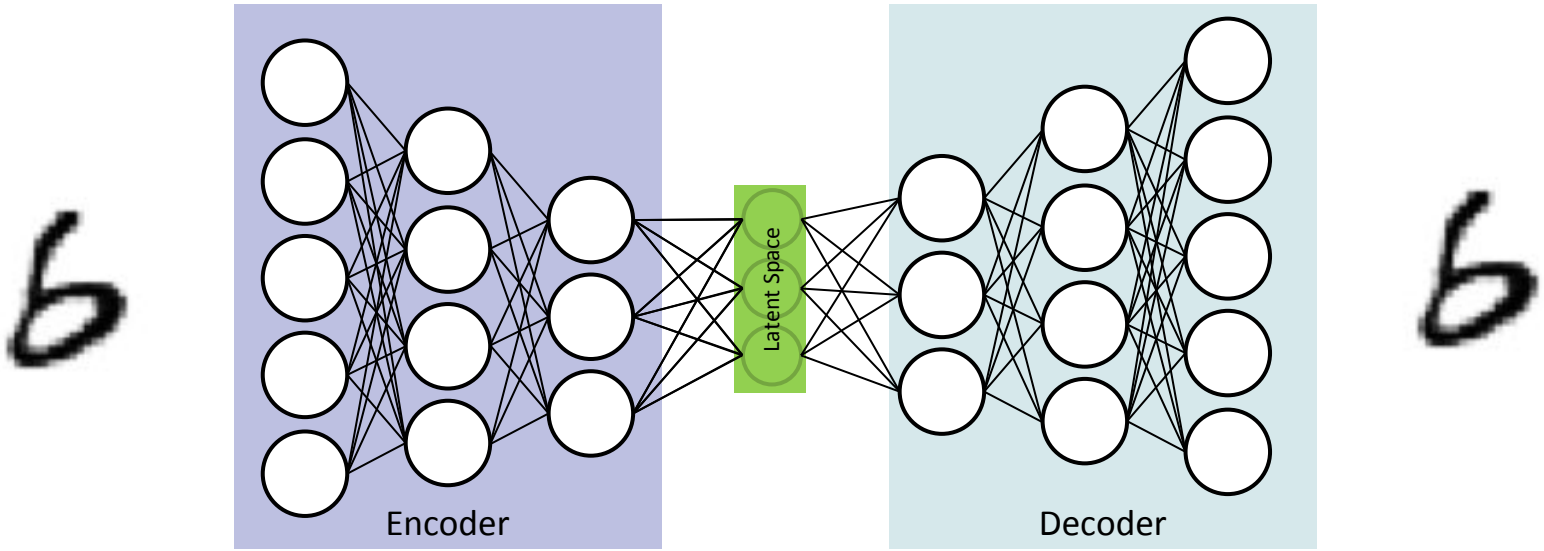
Jenna Collier | BST 261



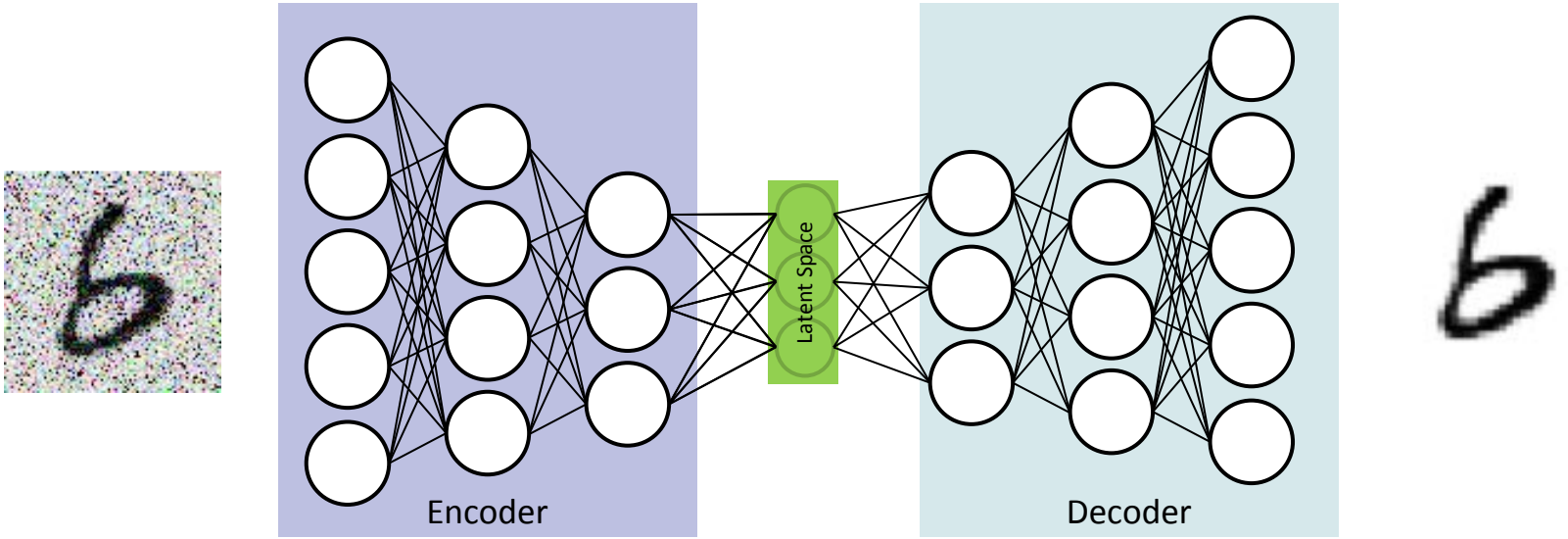
# EVALUATING DEEP VARIATIONAL AUTOENCODERS TRAINED ON PAN-CANCER GENE EXPRESSION

Jenna Collier | BST 261

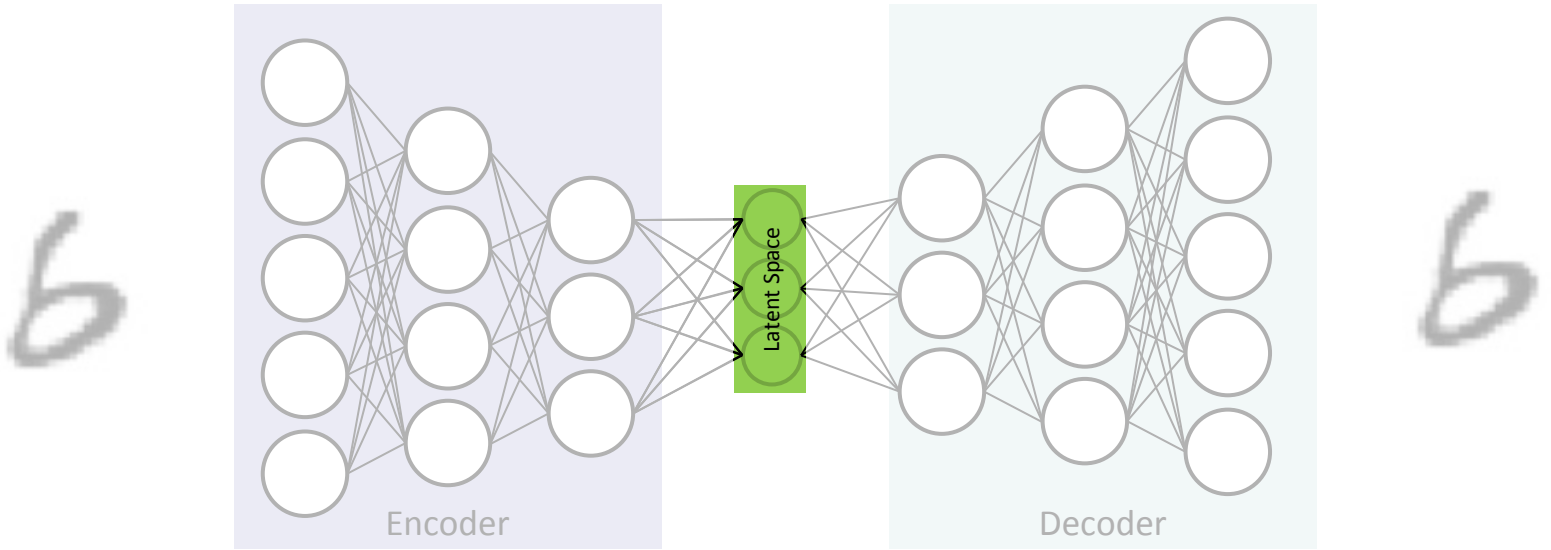
AUTOENCODERS



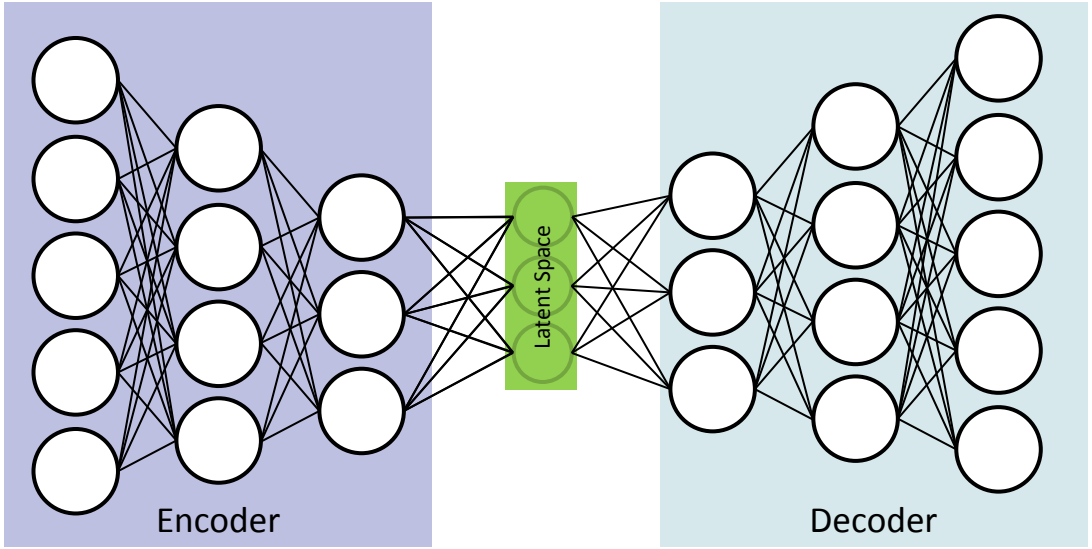
DENOISING AUTOENCODERS



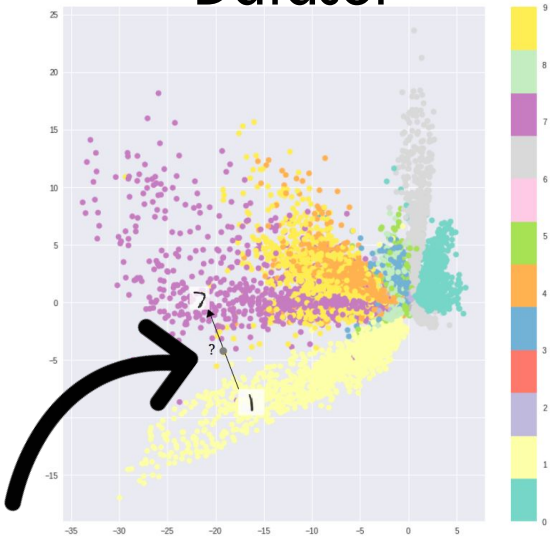
AUTOENCODERS



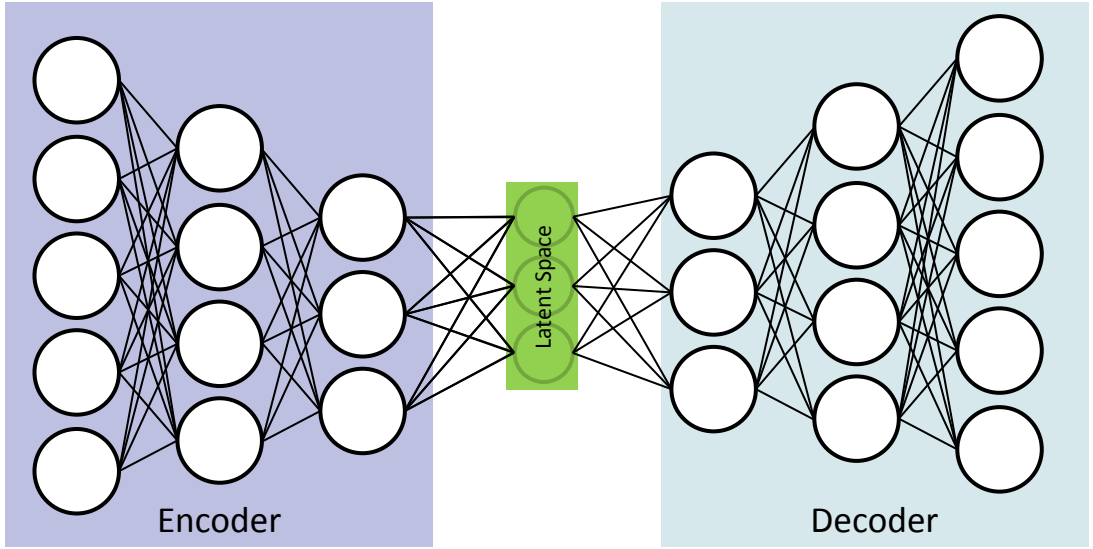
AUTOENCODERS



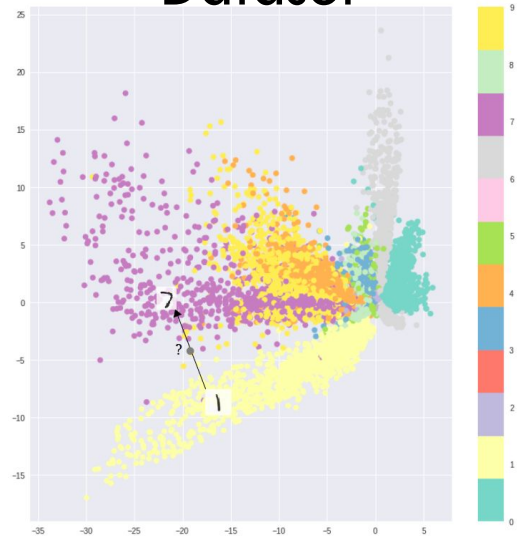

MINST Dataset



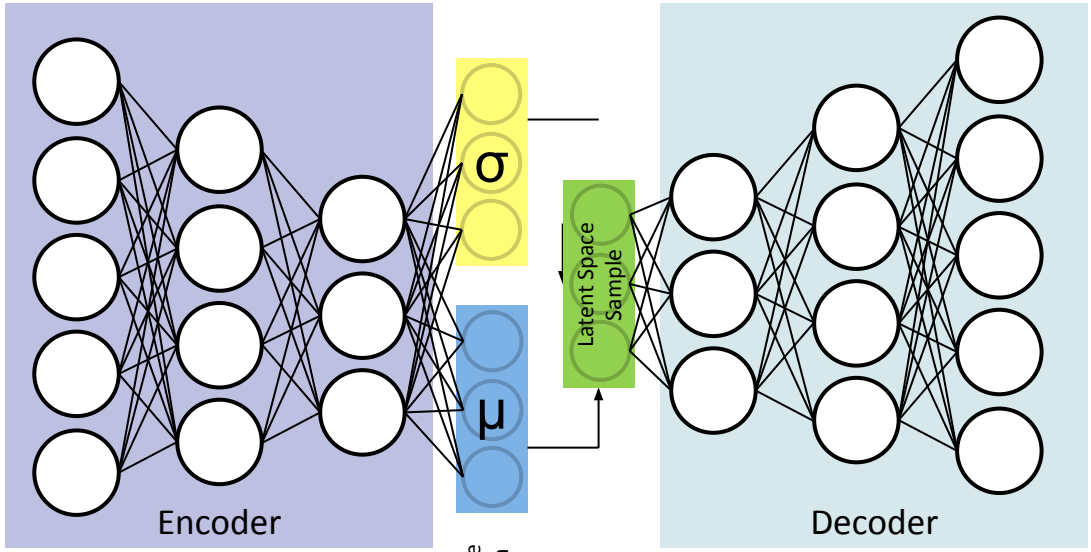
AUTOENCODERS




MINST Dataset



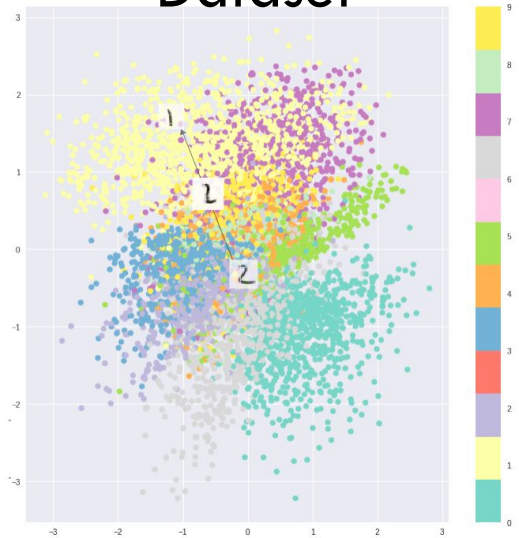
VARIATIONAL AUTOENCODERS




*Where latent space* $(\bar{z})$

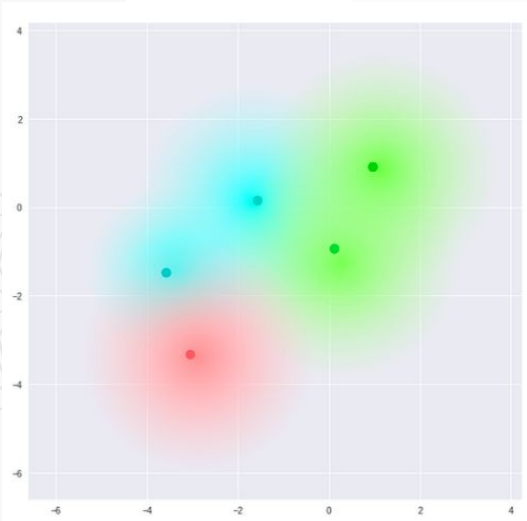
$$\bar{z} \approx N(\bar{\mu}, \bar{\sigma})$$

MINST Dataset

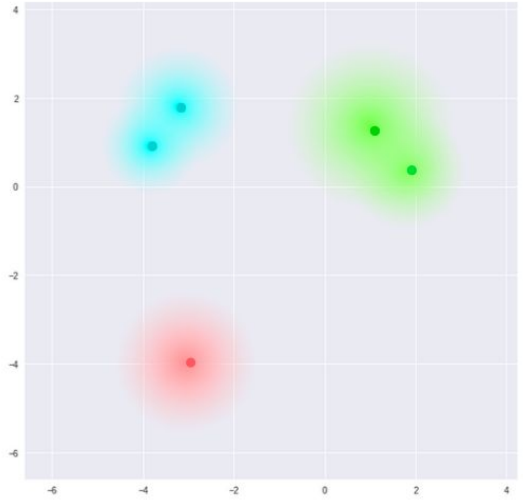


VARIATIONAL AUTOENCODERS



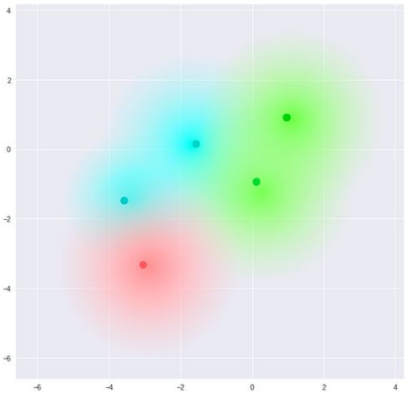



What we require

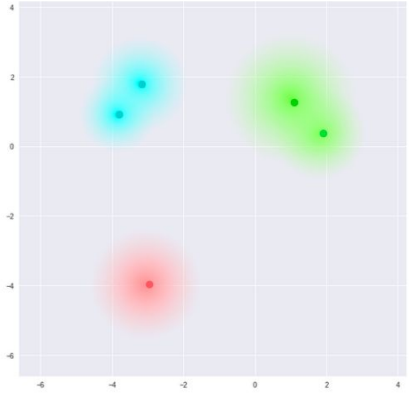


What we may inadvertently end up with

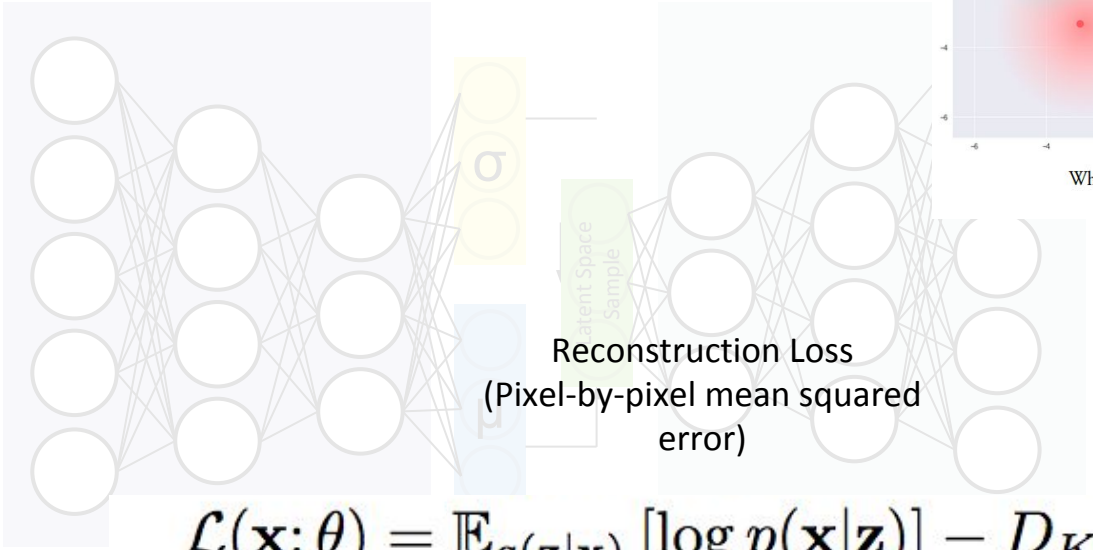
VARIATIONAL AUTOENCODERS



What we require



What we may inadvertently end up with



Reconstruction Loss
(Pixel-by-pixel mean squared error)

Kullback–Leibler divergence
(KL divergence)

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Where latent space ($\bar{z}$)

$\bar{z} \approx N(\bar{\mu}, \bar{\sigma})$

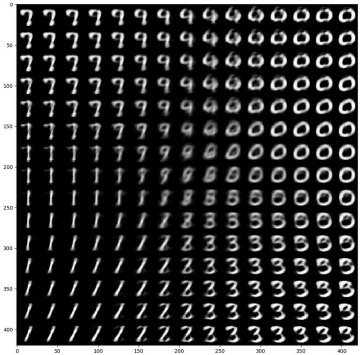
# MINST DATASET (HANDWRITTEN DIGITS)

## Generative Adversarial Networks (GANs)

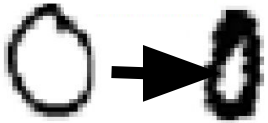
"Create new written zeros"

- ✓ Higher diversity of output
- ✓ Sharp output images
- ✕ Lower stability during training

## Variational Autoencoders (VAEs)



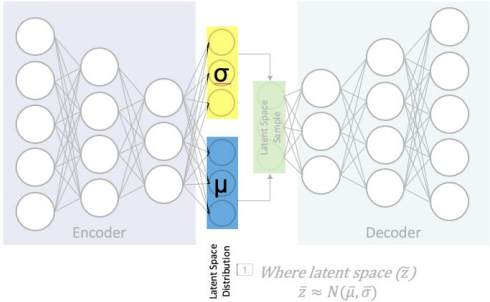
"Increase the width of this zero"



- ✕ Lower diversity of output
- ✕ Blurry output images
- ✓ Higher stability during training

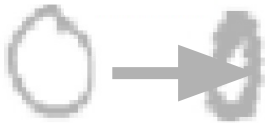
MOTIVATING QUESTIONS FROM CANCER BIOLOGY

Can we use a VAE to identify important genes (latent features) associated with different tumor types?



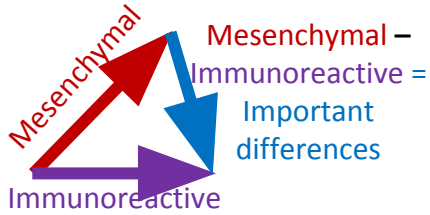

Can we subtract latent spaces defining two types of tumors to identify biologic pathways that differ between them?

"Increase the width of this zero"



Mesenchymal tumors ⊣ Mesenchymal tumors

Immunoreactive tumors ⊣ Immunoreactive tumors



TRAINING DATASET: The Cancer Genome Atlas (TCGA)

Gene expression RNAseq data

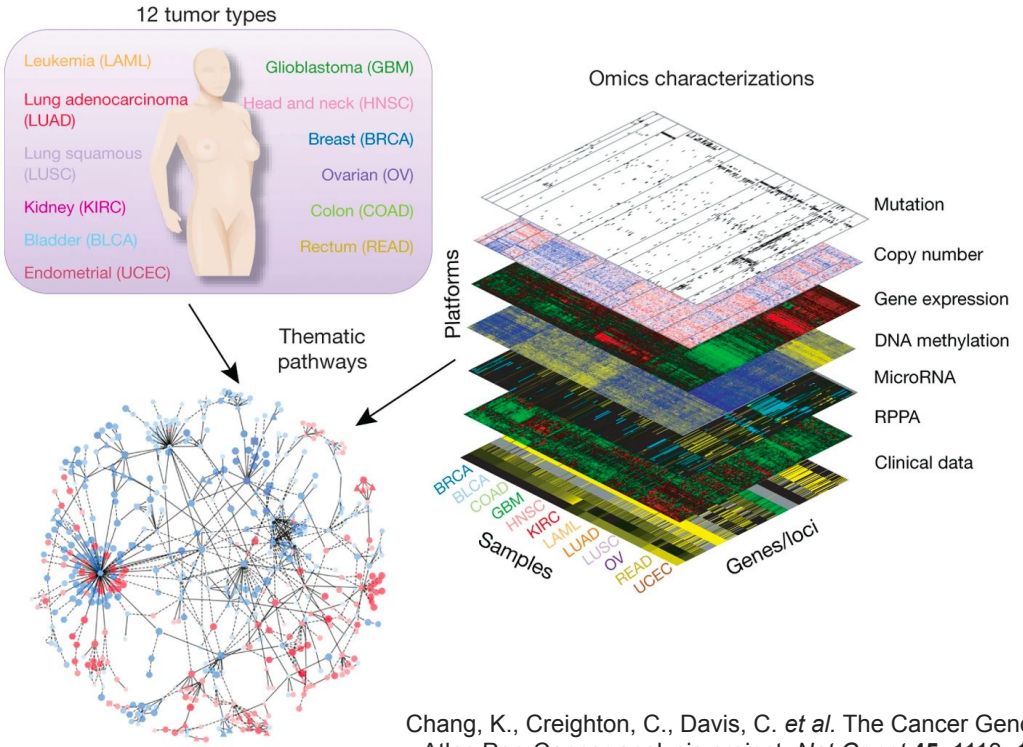
9,732 tumors
+ 727 tumor adjacent normal

= 10,459 samples

Selected 5,000 most variable genes

Batch-correction of gene abundance: Log2(FPKM+1) = RSEM

Scaled expression values from 0 to 1

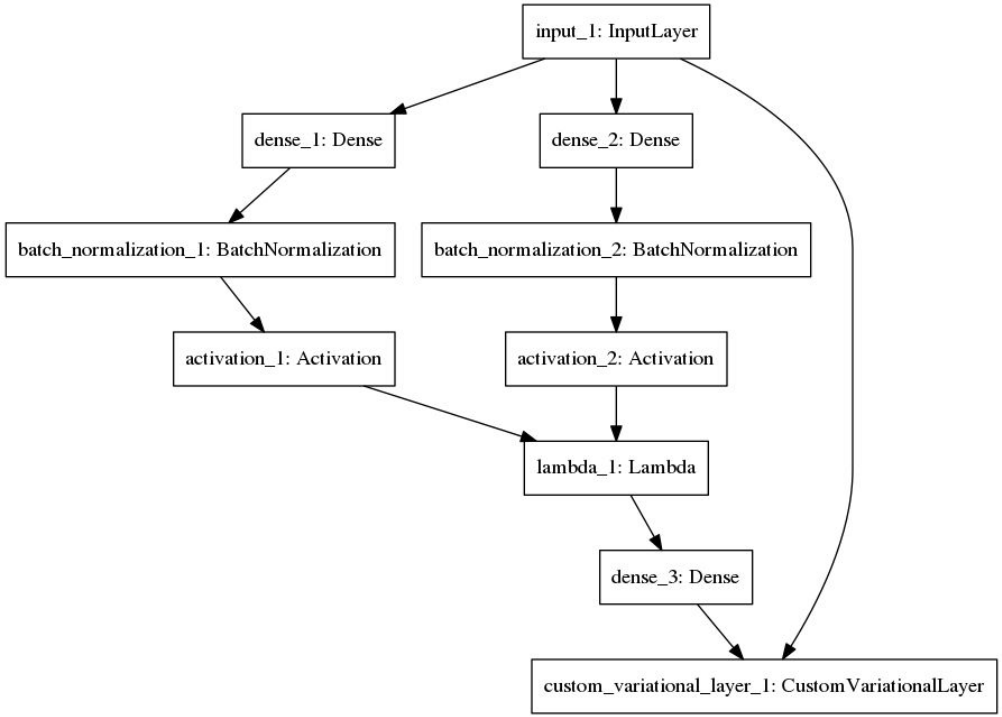



Chang, K., Creighton, C., Davis, C. *et al.* The Cancer Genome Atlas Pan-Cancer analysis project. *Nat Genet* **45,** 1113–1120 (2013).

TYBALT
ARCHITECTURE

- Batch normalization
- ReLU activation in encoder
- Sigmoid activation in decoder

- Parameter sweep over batch sizes, epochs, learning rates, warmups ($\kappa$)
  - Batch size = 50
  - Epochs = 50
  - Learning rate = 0.0005
  - $\kappa$ = 1



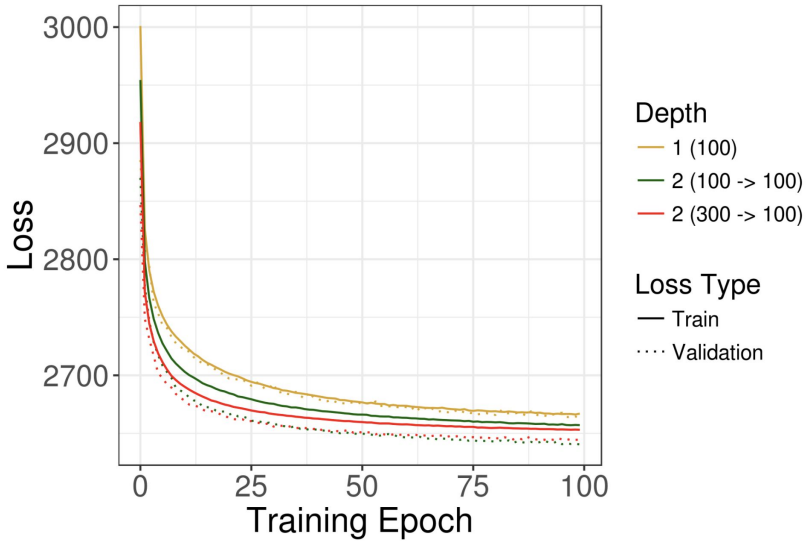

https://github.com/greenelab/tybalt

# TESTING ALTERNATIVE VAE ARCHITECTURES

Table 1: VAE Architectures

| Name | Hidden Layers | Hidden Layer Size | Latent Feature Size |
|---|---|---|---|
| Tybalt | 1 | | 100 |
| Two Hidden VAE (100) | 2 | 100 | 100 |
| Two Hidden VAE (300) | 2 | 300 | 100 |

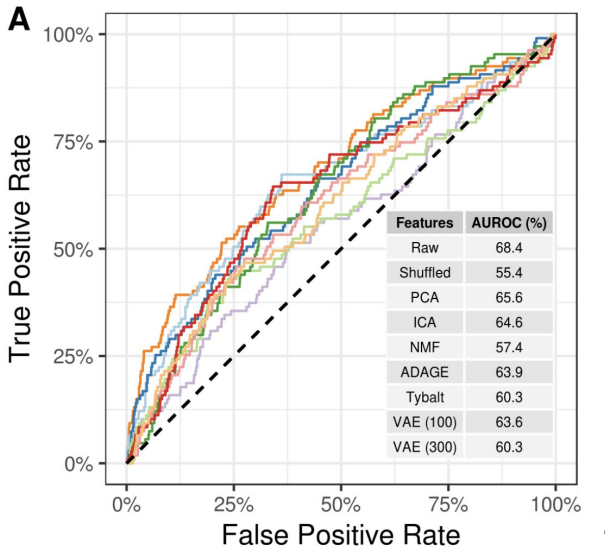
Modest improvements with modifying the hidden layer; Tybalt was good enough!



## PREDICTING NF1 INACTIVATION FROM LATENT FEATURES

- NF1 inactivated samples identified from paired copy number or mutation data
- Trained elastic net logistic regression classifiers
- Input: latent space features from Tybalt, PCA, NMF, ADAGE, etc.

- Negative control: shuffled gene expression matrices for each individual sample
- Positive control: Raw features

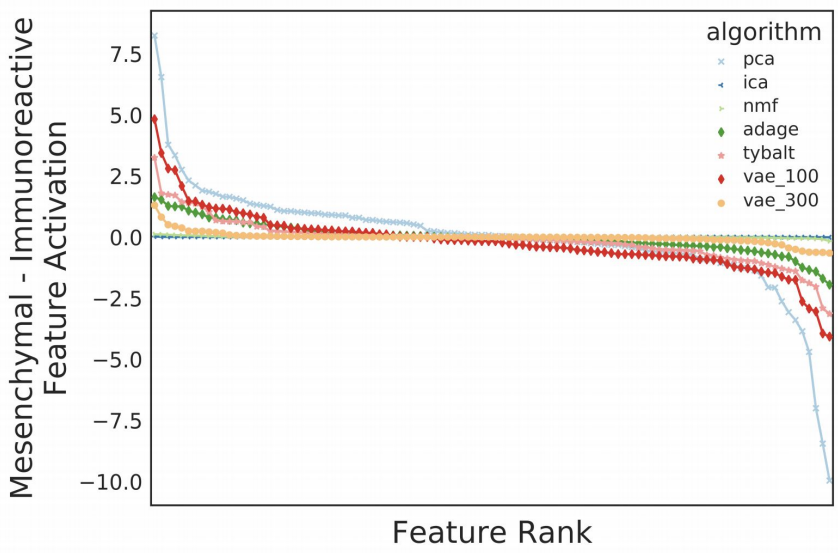



Tybalt produces predictive latent features (AUROC=60%, 12 features), but not as well as PCA (AUROC=66%, 7 features)

LATENT SPACE ARITHMETIC OF HGSC SUBTYPES

- Latent space features derived from various reduction algorithms using
  - 1) mesenchymal samples
  - 2) immunoreactive samples
- Overrepresentation (ORA) analysis of Gene Ontology (GO) terms of high weight genes to identify pathways

Tybalt and VAE (300) identified collagen catabolic process as important distinguishing pathway that has been validated in literature

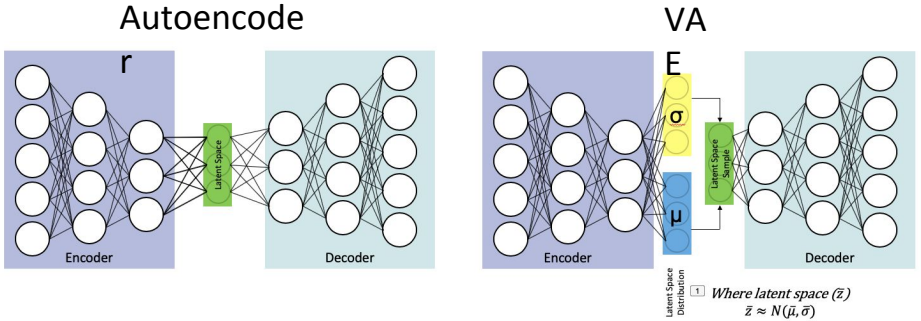


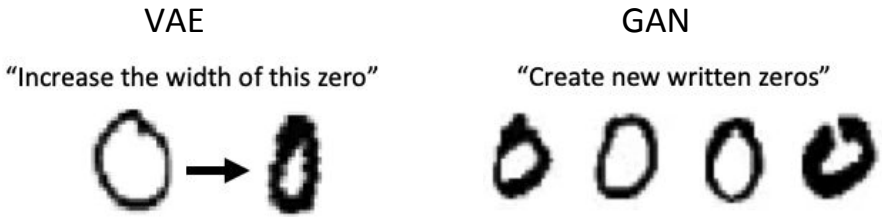

Mesenchymal – Immunoreactive = Differing Features

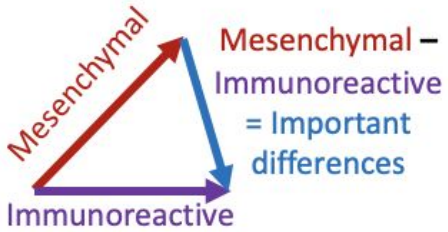
| Algorithm | Top Pathway | Adj. p value |
|---|---|---|
| PCA | *Zero high weight genes* | |
| ICA | *No significant pathways* | |
| NMF | Homophilic Cell Adhesion Via Plasma Membrane Adhesion Molecules | $3.3e^{-06}$ |
| ADAGE | *No significant pathways* | |
| Tybalt | Collagen Catabolic Process | $1.8e^{-09}$ |
| VAE (100) | Epidermis Development | $8.0e^{-04}$ |
| VAE (300) | Collagen Catabolic Process | $1.7e^{-03}$ |

SUMMARY

- Autoencoders
- Variational Autoencoders (VAEs)
- VAEs vs. GANs
- VAEs are good at distilling high dimensional data into important latent features
- **VAEs can identify biologically relevant latent spaces**
- Arithmetic can be done using these latent features
- Tybalt is a VAE trained on TCGA data



# REFERENCES


Relevant Papers:

- arXiv:1711.04828
- Gregory P. Way & Casey S. Greene. Extracting a Biologically Relevant Latent Space from Cancer Transcriptomes with Variational Autoencoders. bioRxiv 174474; doi: https://doi.org/10.1101/174474
- Chang, K., Creighton, C., Davis, C. *et al.* The Cancer Genome Atlas Pan-Cancer analysis project. *Nat Genet* **45,** 1113–1120 (2013).

Github:

- https://github.com/greenelab/tybalt

Great Overview of VAEs (with helpful images):

- Irhum Shafkat. Intuitively Understanding Variational Autoencoders. Towards Data Science – a Medium publication. <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf > Accessed May 11, 2020

# Image Generation

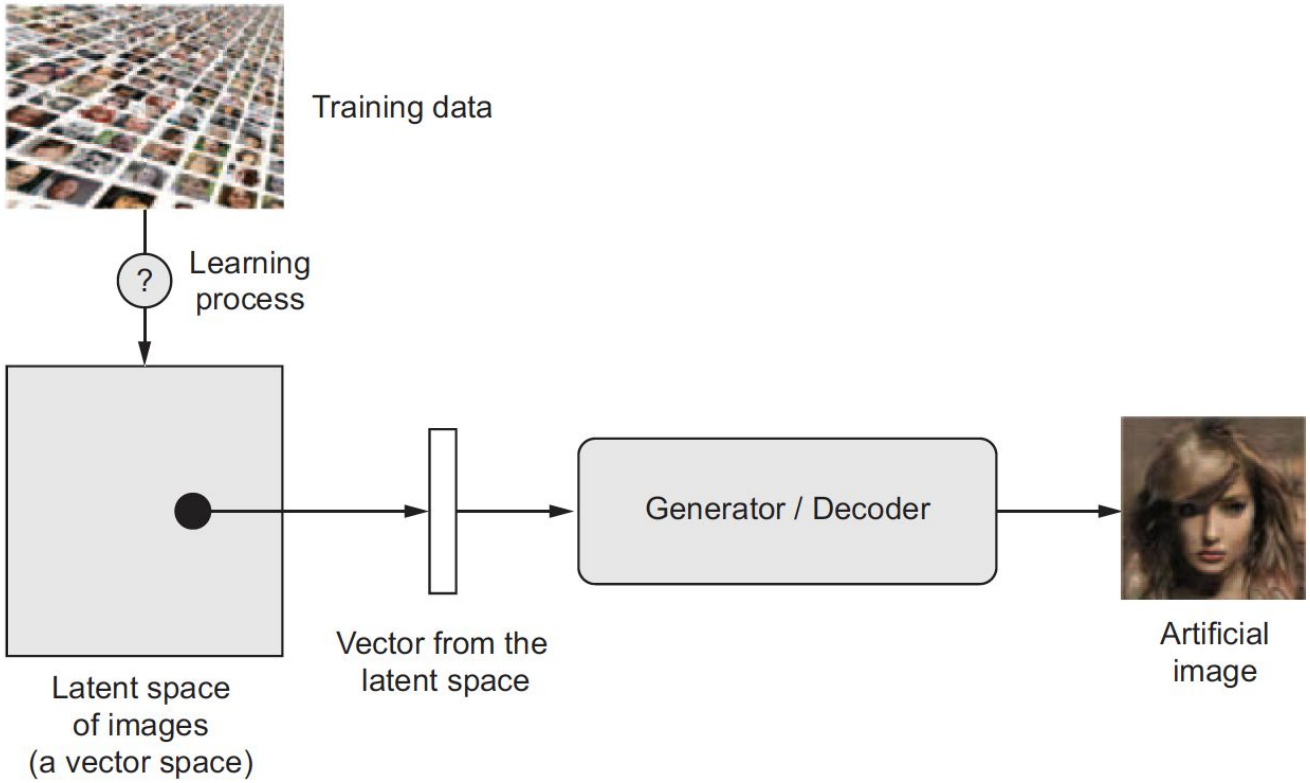
# Generating Images

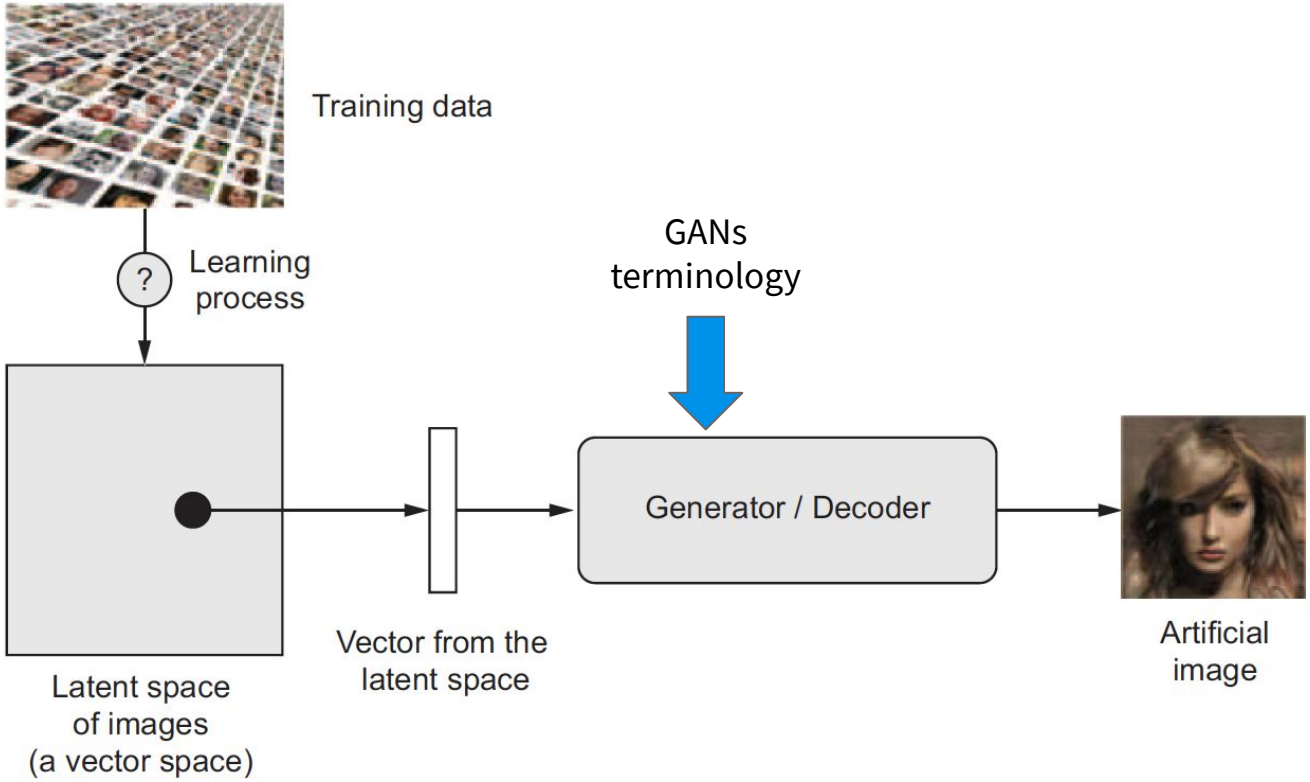
- We'll discuss two different ways of generating images
  - Variational autoencoders (VAEs)
  - Generative adversarial networks (GANs)

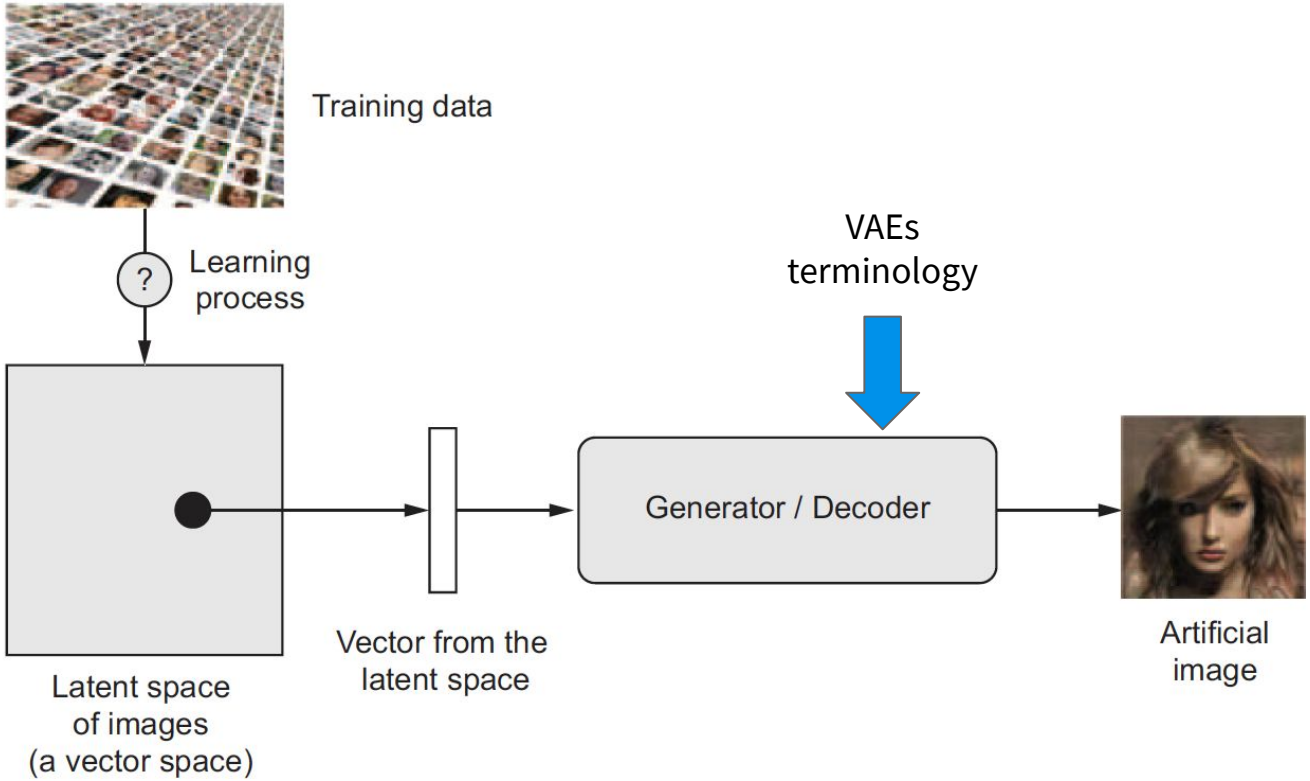
- These methods can also be used to generate sound, music or text, but we'll focus on images

# Generating Images

- **Main idea:** sample from a **latent space** of images to create entirely new images
  - Latent space: a low-dimensional representation (vector space) where any point can be mapped to a realistic-looking image
  - The module that takes in a point from the latent space and generates an image is called a **generator** (in the case of GANs) or a **decoder** (in the case of VAEs)
  - Generates images never explicitly seen before


Training data
?
Learning process
Generator / Decoder
Vector from the latent space
Artificial image
Latent space of images (a vector space)

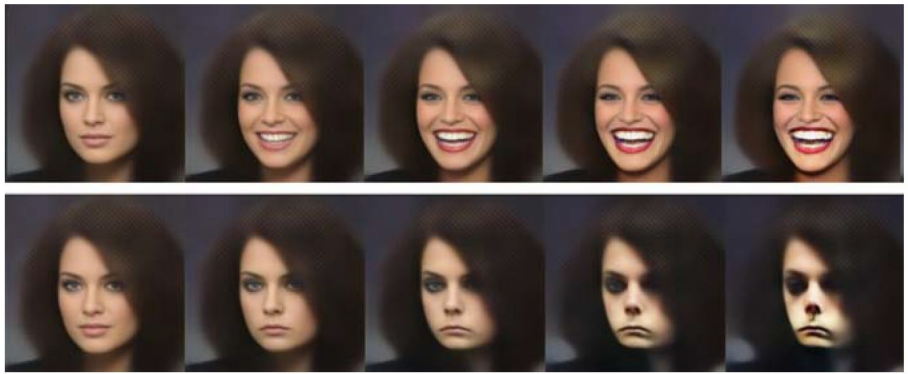

Training data
?
Learning process
GANs terminology
Generator / Decoder
Latent space of images (a vector space)
Vector from the latent space
Artificial image


Training data
?
Learning process
VAEs terminology
Generator / Decoder
Latent space of images (a vector space)
Vector from the latent space
Artificial image

Images generated from a VAE

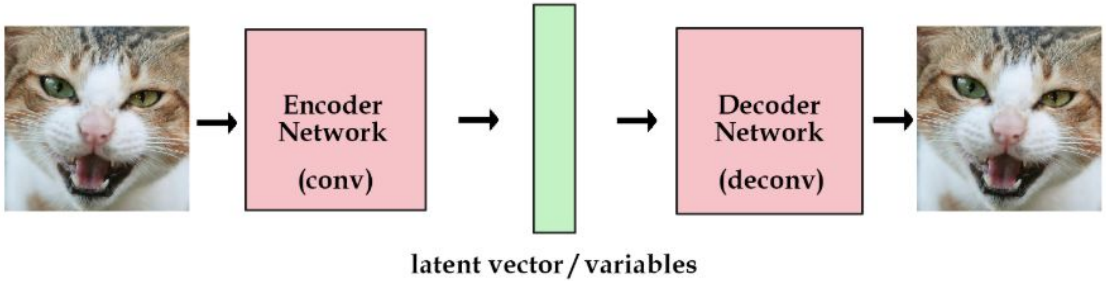
# Concept vectors

- We want to create a latent vector space
  - We can think of this as embedding - just like what we did with text and RNNs
  - Certain directions in the space may encode interesting axes of variation in the original data
  - Example: vectors that represent a smile or vectors that represent sunglasses
- Once we create these vectors we can edit images by projecting them into the latent space, moving their representation in a meaningful way, and then decoding them back to image space



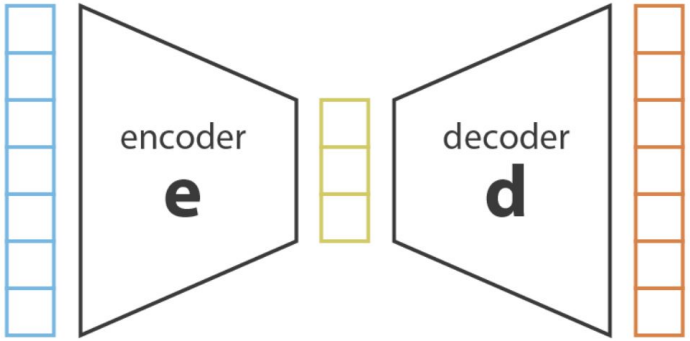
# Autoencoders

- Neural networks used to learn efficient representations (encodings) in an unsupervised way
  - A type of dimensionality reduction
- At the same time learn to decode the representations into something very close to the input
  - Reconstructs the original image as much as possible
- Essentially “memorizing” images
  - Not creating (generating) anything new



# Autoencoders

- Used for
  - Denoising, face recognition, anomaly detection, dimensionality reduction, etc.




$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

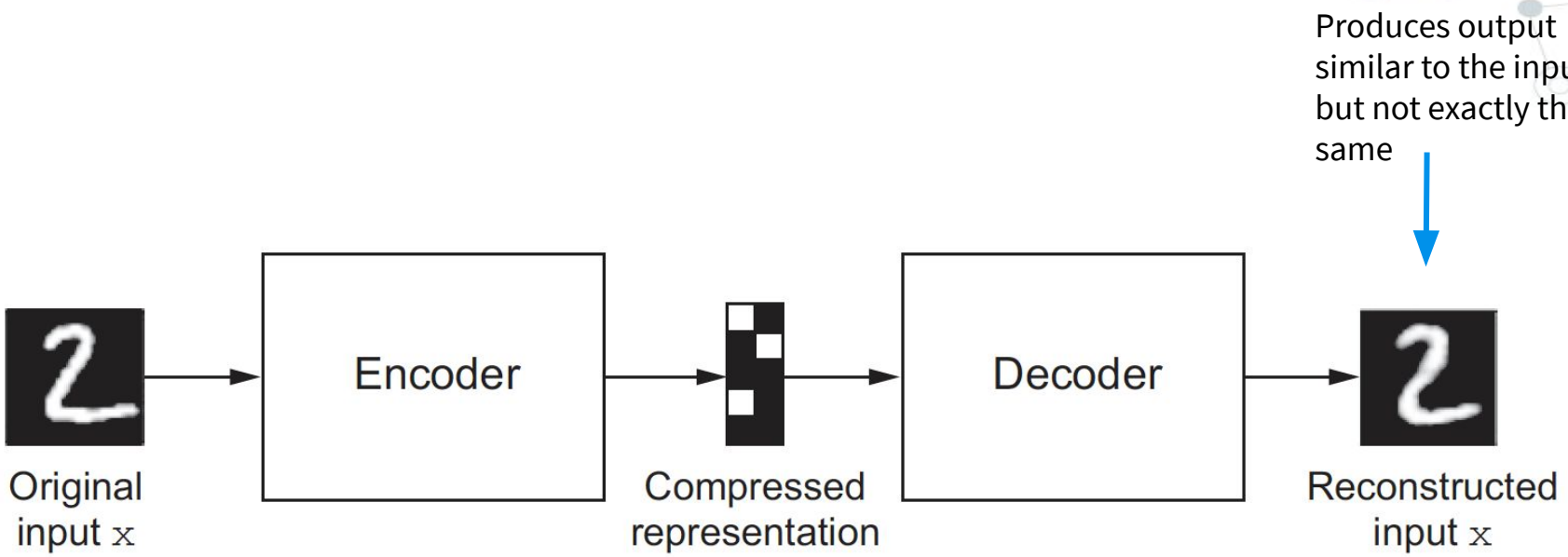
$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

# Variational Autoencoders (VAEs)

- Simultaneously discovered by Kingma and Welling in December 2013 and Rezende, Mohamed, and Wierstra in January 2014
- A generative model especially appropriate for the task of image editing using concept vectors
- Modern version of **autoencoders**
  - Autoencoders are networks that encode an input to a low-dimensional latent space and then decode it back
  - The encoder learns how to reconstruct the original inputs with some added noise to create new images
  - Learn to compress the input data into fewer bits of information

# Variational Autoencoders (VAEs)



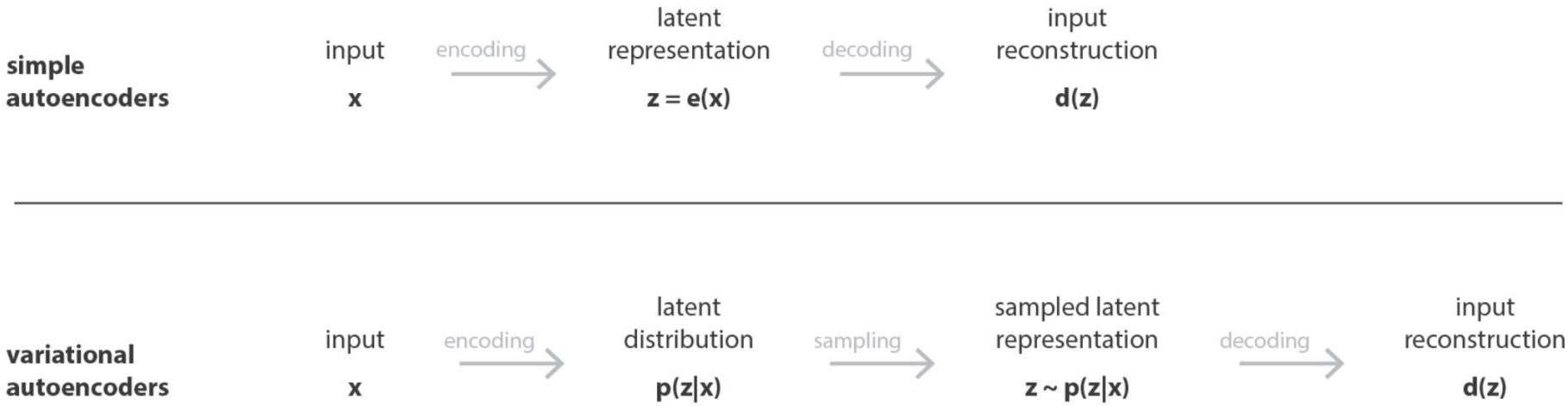
# VAEs

- Autoencoders aren’t great at creating nicely structured latent spaces and they also don’t generate anything new
- VAEs augment autoencoders to learn highly structured latent spaces and are able to generate new images
- Are regularized versions of autoencoders

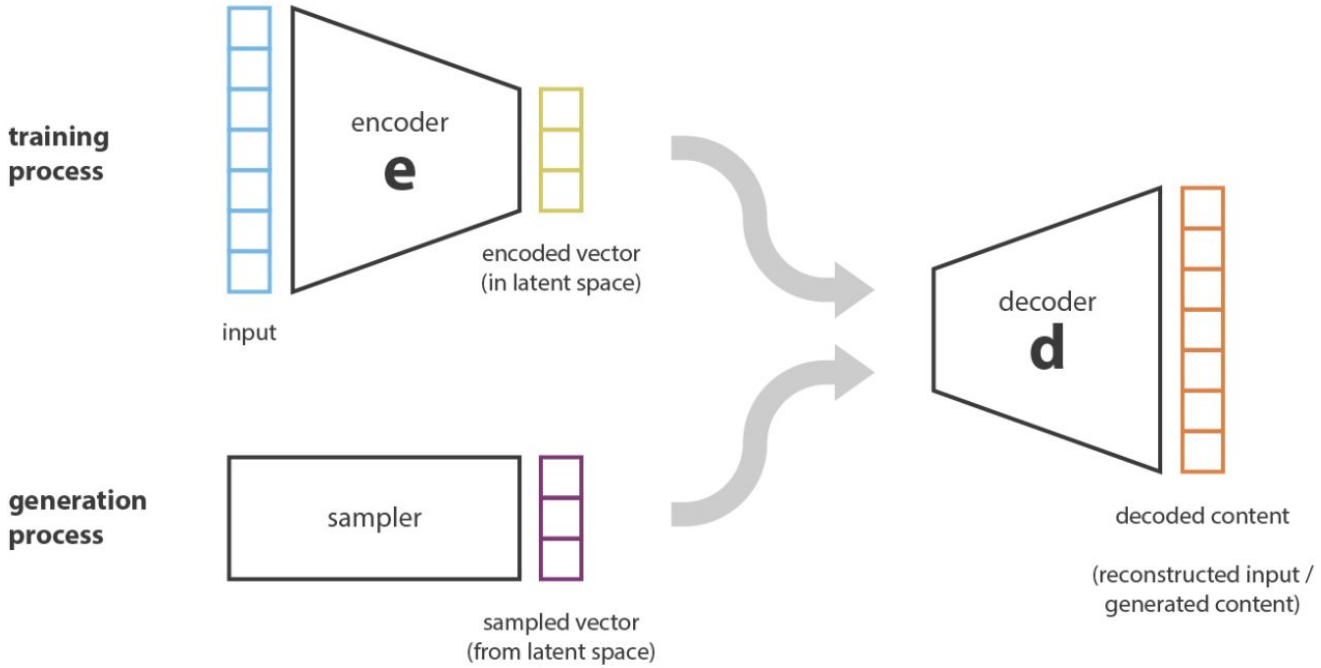
# VAEs

- Rather than compressing the image into a fixed code in the latent space, VAEs turn the image into the **parameters of a statistical distribution** (a mean and variance)
  - Assume the input image has been generated by a statistical process and that the **randomness** of this process should be taken into account when encoding and decoding
  - A VAE uses the mean and variance parameters to randomly sample one element from the distribution and decodes that element back to the original input
    - This process improves robustness and forces the latent space to encode meaningful representations everywhere: every point in the latent space is decoded to a valid output

# Autoencoders vs VAEs

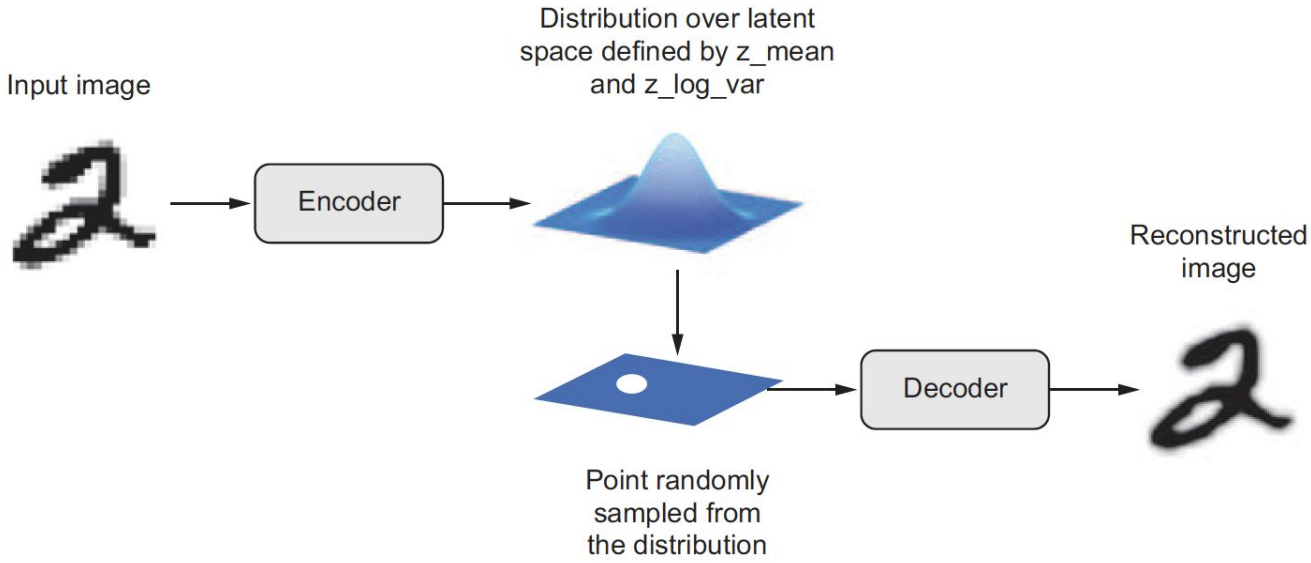


# VAEs




We can generate new data by decoding points that are randomly sampled from the latent space. The quality and relevance of generated data depend on the regularity of the latent space.

# VAEs


Input image
Encoder
Distribution over latent space defined by z_mean and z_log_var
Point randomly sampled from the distribution
Decoder
Reconstructed image

# VAEs

- Algorithm steps:
  - An encoder module turns the input samples into two parameters in a latent space of representations, **z_mean** and **z_log_variance**
  - Randomly sample a point z from the latent normal distribution that's assumed to generate the input image, via

    **z = z_mean + exp(z_log_variance) *epsilon**

    where epsilon is a random tensor of small values
  - A decoder module maps this point in the latent space back to the original input image

- Having epsilon be random ensures every point that's close to the latent location where you encoded the input image can be decoded to something similar to the input image - but not exactly the same
  - Also forces every direction in the latent space to encode a meaningful axis of variation of the data, making the latent space very structured and highly suitable to manipulation via concept vectors

# VAE Training

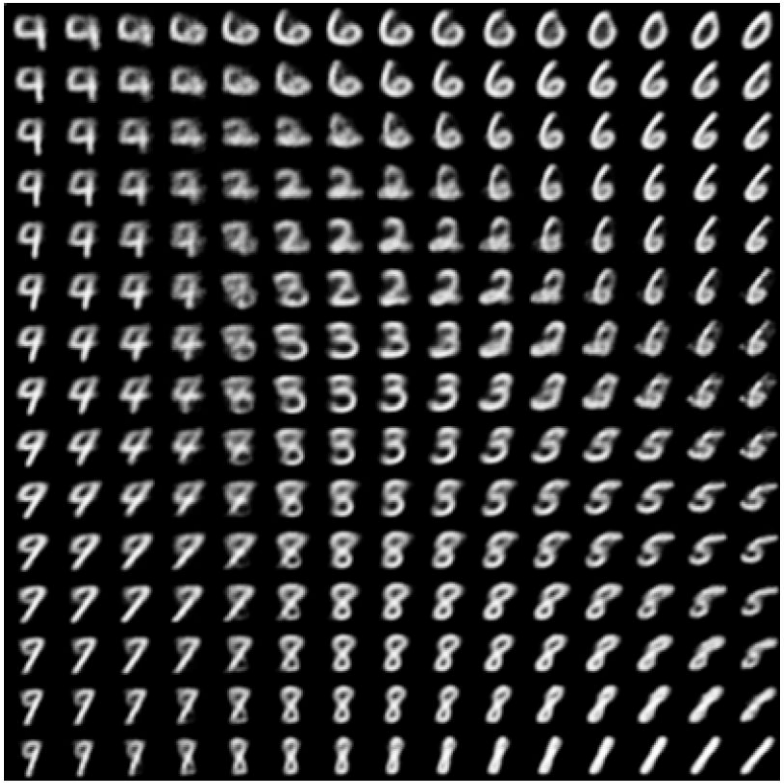
- Parameters are trained using two loss functions
  - **Reconstruction (generative) loss**: measures how accurately the network reconstructed the images; forces the decoded samples to match the initial inputs
  - **Regularization (latent) loss**: measures how closely the latent variables match a unit Gaussian distribution; helps learn well-formed latent spaces and reduce overfitting to the training data

- Because this is a different type of loss, Keras allows you to have a custom loss function and add it as a layer to your network using the **add_loss** layer

# VAE on MNIST Data

Digits decoded from the latent space

Note how one digit morphs into another and that directions in this space have specific meaning (one direction for “four-ness”, one direction for “one-ness”, etc.



# VAEs for Medicine

## Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders

**Gregory P. Way** and
Genomics and Computational Biology Graduate Program, University of Pennsylvania, Philadelphia, PA 19104, USA

**Casey S. Greene***
Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, PA 19104, USA

### Abstract

The Cancer Genome Atlas (TCGA) has profiled over 10,000 tumors across 33 different cancer-types for many genomic features, including gene expression levels. Gene expression measurements capture substantial information about the state of each tumor. Certain classes of deep neural network models are capable of learning a meaningful latent space. Such a latent space could be used to explore and generate hypothetical gene expression profiles under various types of molecular and genetic perturbation. For example, one might wish to use such a model to predict a tumor's response to specific therapies or to characterize complex gene expression activations existing in differential proportions in different tumors. Variational autoencoders (VAEs) are a deep

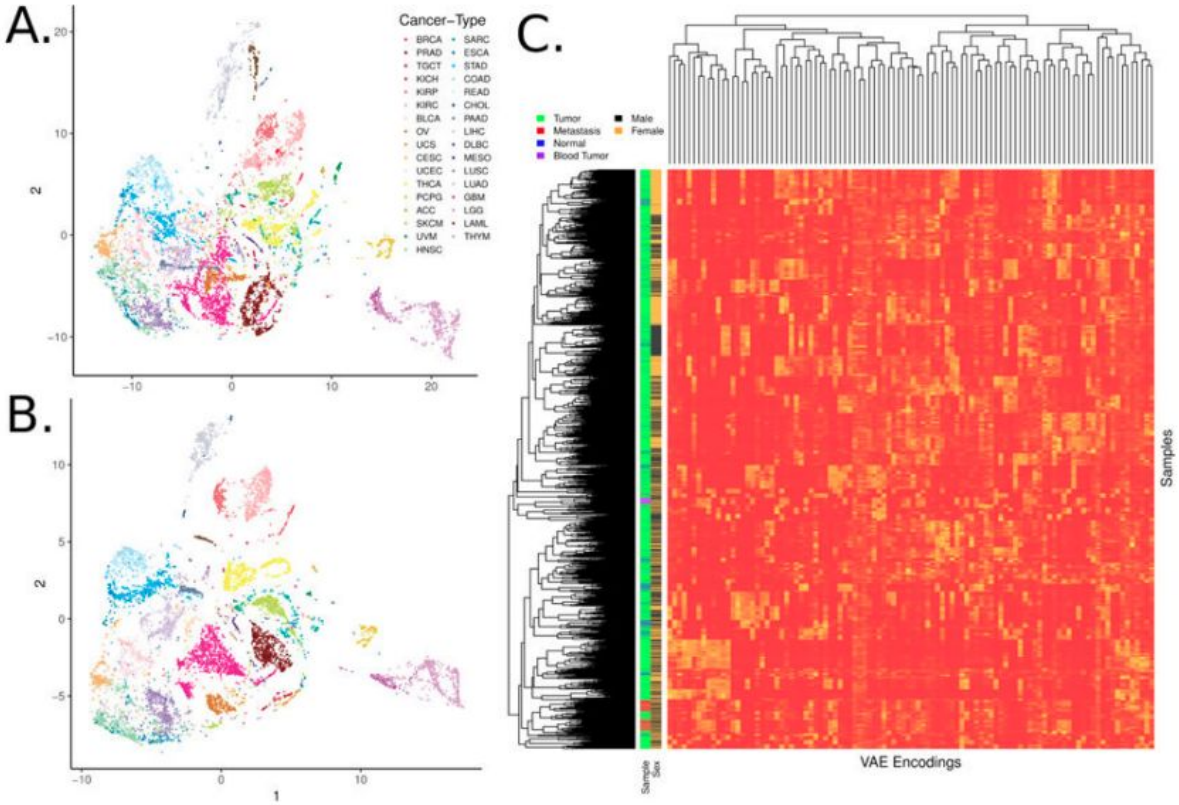



**Fig. 2. Samples encoded by a variational autoencoder retain biological signals**
(**A**) t-distributed stochastic neighbor embedding (t-SNE) of TCGA pan-cancer tumors with Tybalt encoded features. (**B**) t-SNE of 0-1 normalized gene expression features. Tybalt retains similar signals as compared to uncompressed gene expression data. (**C**) Full Tybalt encoding features by TCGA pan-cancer sample heatmap. Given on the y axis are the patients sex and type of sample.

| | | |
|---|---|---|
| setup.py | add util import to init (#133) | a year ago |
| tsne_tybalt_features.ipynb | Adding ADAGE model to tsne visualization (#114) | a year ago |
| tybalt_twohidden.ipynb | Add Two Hidden Layer Model (#81) | 2 years ago |
| tybalt_vae.ipynb | Update conda environments (#108) | a year ago |

README.md

# Tybalt 

## *A Variational Autoencoder trained on Pan-Cancer Gene Expression*

**Gregory Way and Casey Greene 2017**

DOI 10.5281/zenodo.1047069

The repository stores scripts to train, evaluate, and extract knowledge from a variational autoencoder (VAE) trained on 33 different cancer-types from The Cancer Genome Atlas (TCGA).

The specific VAE model is named *Tybalt* after an instigative, cat-like character in Shakespeare's "Romeo and Juliet". Just as the character Tybalt sets off the series of events in the play, the model Tybalt begins the foray of VAE manifold learning in transcriptomics. Also, deep unsupervised learning likes cats.

We discuss the training and evaluation of Tybalt in our PSB paper:

*Extracting a Biologically Relevant Latent Space from Cancer Transcriptomes with Variational Autoencoders*.

# VAEs for Medicine

## Dr.VAE: Drug Response Variational Autoencoder

Ladislav Rampasek*†‡ Daniel Hidru†‡ Petr Smirnov§
Benjamin Haibe-Kains§¶‖ Anna Goldenberg*‡†

**Abstract**

We present two deep generative models based on Variational Autoencoders to improve the accuracy of drug response prediction. Our models, Perturbation Variational Autoencoder and its semi-supervised extension, Drug Response Variational Autoencoder (Dr.VAE), learn latent representation of the underlying gene states before and after drug application that depend on: (i) drug-induced biological change of each gene and (ii) overall treatment response outcome. Our VAE-based models outperform the current published benchmarks in the field by anywhere from 3 to 11% AUROC and 2 to 30% AUPR. In addition, we found that better reconstruction accuracy does not necessarily lead to improvement in classification accuracy and that jointly trained models perform better than models that minimize reconstruction error independently.

# Generative Adversarial Networks (GANs)

# GANs

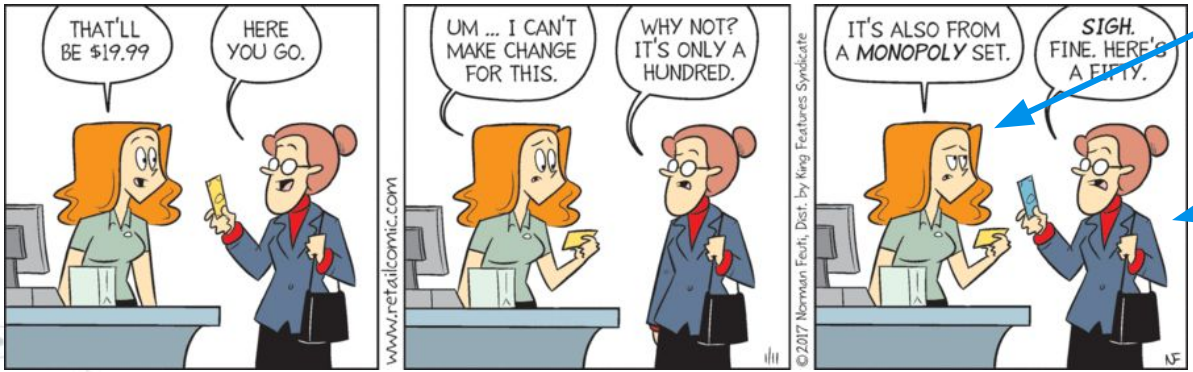
- Introduced by Goodfellow et al in 2014
- An alternative to VAEs for learning latent spaces of images
- Enable generation of fairly realistic (have been increasingly realistic over time) synthetic images by forcing the generated images to be statistically almost indistinguishable from real ones
- Made of 2 parts:
  - **Generator network**: takes as input a random vector (a random point in the latent space) and decodes it into a synthetic image - trained to fool the discriminator network
  - **Discriminator network (or adversary)**: takes as input an image (real or synthetic) and predicts whether the image came from the training set or was created by the generator network

# Intuition Examples

Think of someone trying to create counterfeit money who has a spy inside a bank. They can create the fake money and try to slip it past back employees. The bank employees will notice the fake money easily in the beginning, but as the spy relays information to the counterfeiter, they will make better fake versions of money that will be more difficult for bank employees to distinguish as fake. Meanwhile, the bank will come up with new ways of detecting the updated counterfeit money.

# Intuition Examples

Think of someone trying to create counterfeit money who has a spy inside a bank. They can create the fake money and try to slip it past back employees. The bank employees will notice the fake money easily in the beginning, but as the spy relays information to the counterfeiter, they will make better fake versions of money that will be more difficult for bank employees to distinguish as fake. Meanwhile, the bank will come up with new ways of detecting the updated counterfeit money.

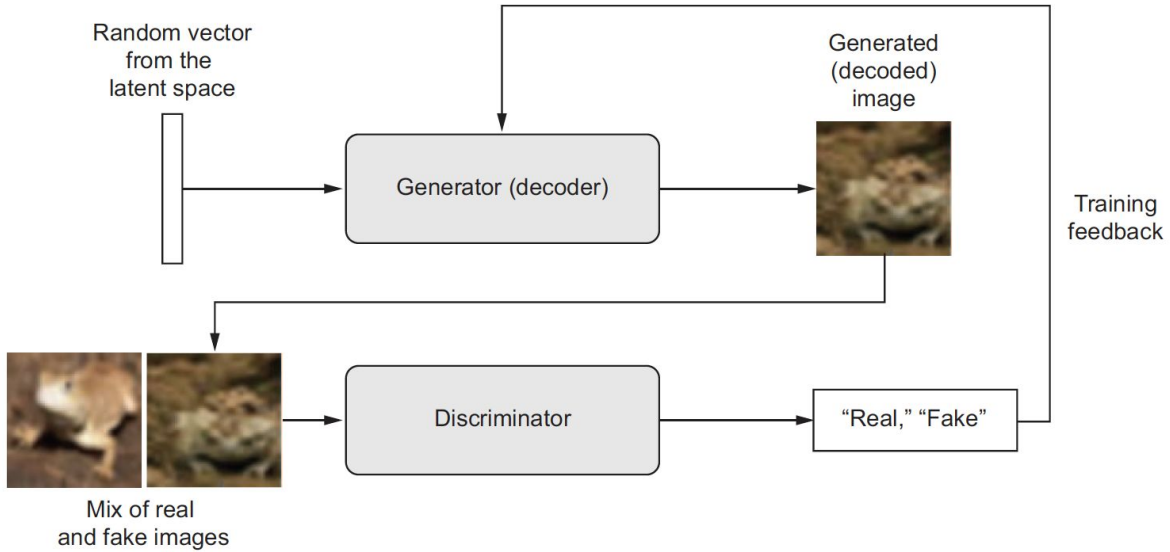


# Intuition Examples

Now think of someone trying to forge Picasso paintings. They will be bad in the beginning, but will improve over time with feedback from an expert art dealer on how they detected the fake art from the real paintings. At the same time, new, more advanced techniques will be discovered to spot forged paintings.

# GANs

- The generator learns to generate increasingly realistic images as it trains
- At the same time, the discriminator network is constantly adapting to the gradually improving capabilities of the generator
- After training, the generator is able to turn any point in its input space into a believable image
- Unlike VAEs, this latent space has fewer explicit guarantees of meaningful structure



# Training

- The optimization minimum isn't fixed for GANs
  - Every step taken during gradient descent changes the entire landscape - the optimization process is dynamic
  - Need to find an equilibrium, not a minimum
    - There are 2 opposing forces here - the generator and the discriminator
- Very difficult to train
  - Many parameters to tune
  - Complex model architecture

# Training

- Implementation of a deep convolutional GAN (DCGAN) in Keras
  - The generator and discriminator are deep CNNs
  - A **generator** network maps vectors to images
  - A **discriminator** network maps images to a binary score estimating the probability that the image is real
  - A gan network chains the generator and the discriminator together:

    **gan(x) = discriminator(generator(x))**

    Thus this gan network maps latent space vectors to the discriminator's assessment of the realism of these latent vectors as decoded by the generator

# Training

- You train the discriminator using examples of real and fake images along with “real”/“fake” labels, just as you train any regular image-classification model

- To train the generator, you use the gradients of the generator’s weights with regard to the loss of the gan model. This means, at every step, you move the weights of the generator in a direction that makes the discriminator more likely to classify as “real” the images decoded by the generator. In other words, you train the generator to fool the discriminator

# Tricks for Training

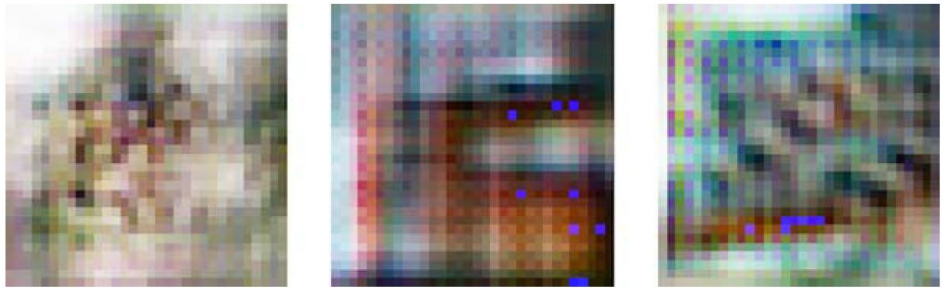
- There are a few useful tricks to help train a GAN
  - Use tanh as the last activation in the generator, instead of sigmoid, which is more commonly found in other types of models
  - Sample points from the latent space using a normal distribution (Gaussian distribution), not a uniform distribution
  - Stochasticity is good to induce robustness. Because GAN training results in a dynamic equilibrium, GANs are likely to get stuck in all sorts of ways. Introducing randomness during training helps prevent this. You can introduce randomness in two ways: by using dropout in the discriminator and by adding random noise to the labels for the discriminator

# Tricks for Training

- Sparse gradients can hinder GAN training. In deep learning, sparsity is often a desirable property, but not in GANs. Two things can induce gradient sparsity: max pooling operations and ReLU activations. Instead of max pooling, it’s recommended to use strided convolutions for downsampling, and using a LeakyReLU layer instead of a ReLU activation. It’s similar to ReLU, but it relaxes sparsity constraints by allowing small negative activation values

# Tricks for Training

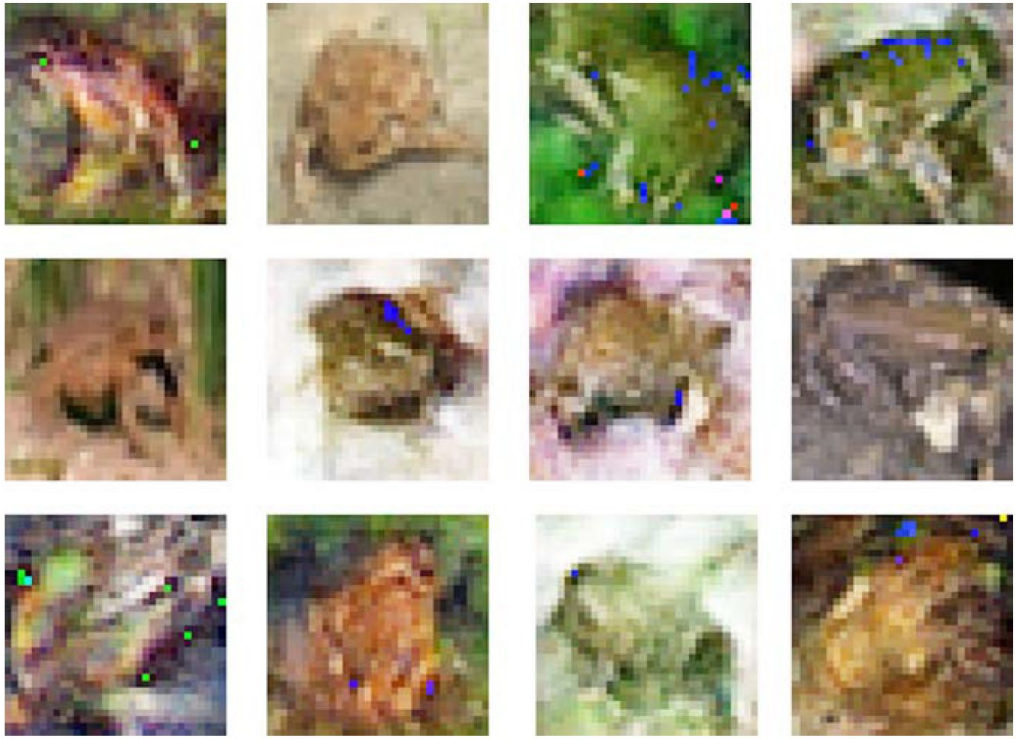
- In generated images, it's common to see checkerboard artifacts caused by unequal coverage of the pixel space in the generator. To fix this, use a kernel (filter) size that's divisible by the stride size whenever you use a strided Conv2DTranpose or Conv2D in both the generator and the discriminator



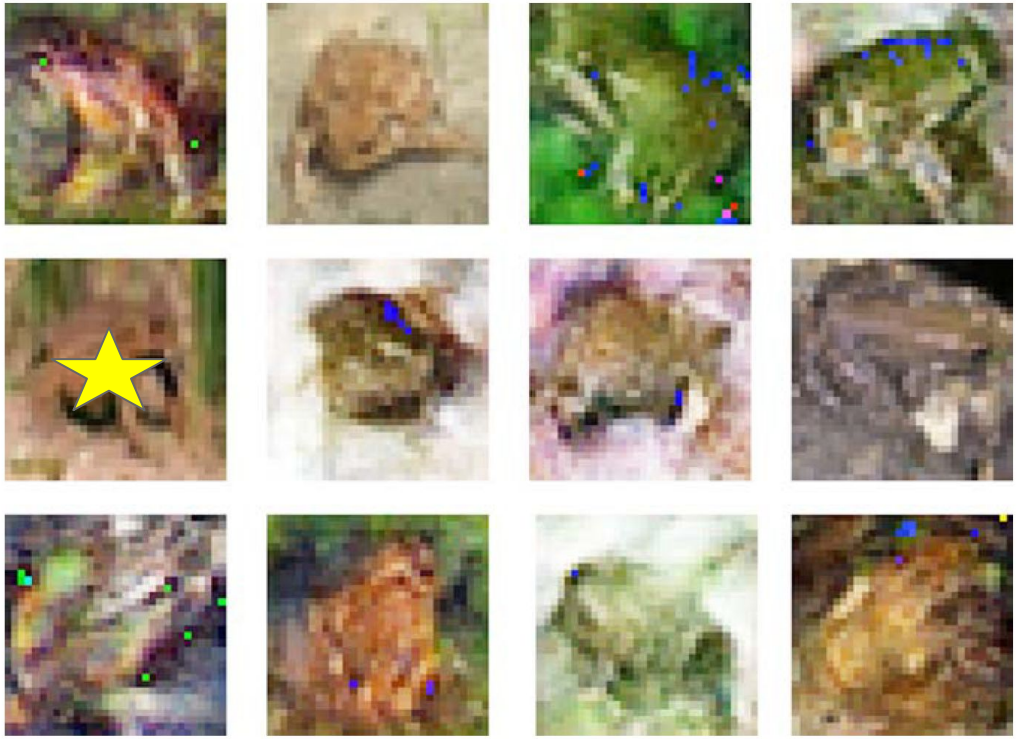
# Tricks for Training

- When training, you may see the generator loss begin to increase considerably, while the discriminative loss tends to zero—the discriminator may end up dominating the generator. If that's the case, try reducing the discriminator learning rate, and increase the dropout rate of the discriminator

- Can you guess which image is from the training set (not created by the generator) in each column?



- Can you guess which image is from the training set (not created by the generator) in each column?



- Can you guess which image is from the training set (not created by the generator) in each column?

- Can you guess which image is from the training set (not created by the generator) in each column?

- Can you guess which image is from the training set (not created by the generator) in each column?

# GANs for Medicine

https://arxiv.org/pdf/1809.06222.pdf

## GANs for Medical Image Analysis

Salome Kazeminia[a,1], Christoph Baur[b,1], Arjan Kuijper[c], Bram van Ginneken[d], Nassir Navab[b], Shadi Albarqouni[b], Anirban Mukhopadhyay[a]

[a]*Department of Computer Science, TU Darmstadt, Germany*
[b]*Computer Aided Medical Procedures (CAMP), TU Munich, Germany*
[c]*Fraunhofer IGD, Darmstadt, Germany*
[d]*Radboud University Medical Center, Nijmegen, The Netherlands*

**Abstract**

Generative Adversarial Networks (GANs) and their extensions have carved open many exciting ways to tackle well known and challenging medical image analysis problems such as medical image de-noising, reconstruction, segmentation, data simulation, detection or classification. Furthermore, their ability to synthesize images at unprecedented levels of realism also gives hope that the chronic scarcity of labeled data in the medical field can be resolved with the help of these generative models. In this review paper, a broad overview of recent literature on GANs for medical applications is given, the shortcomings and opportunities of the proposed methods are thoroughly discussed and potential future work is elaborated. We review the most relevant papers published until the submission date. For quick access, important details such as the underlying method, datasets and performance are tabulated. An interactive visualization categorizes all papers to keep the review alive[2].

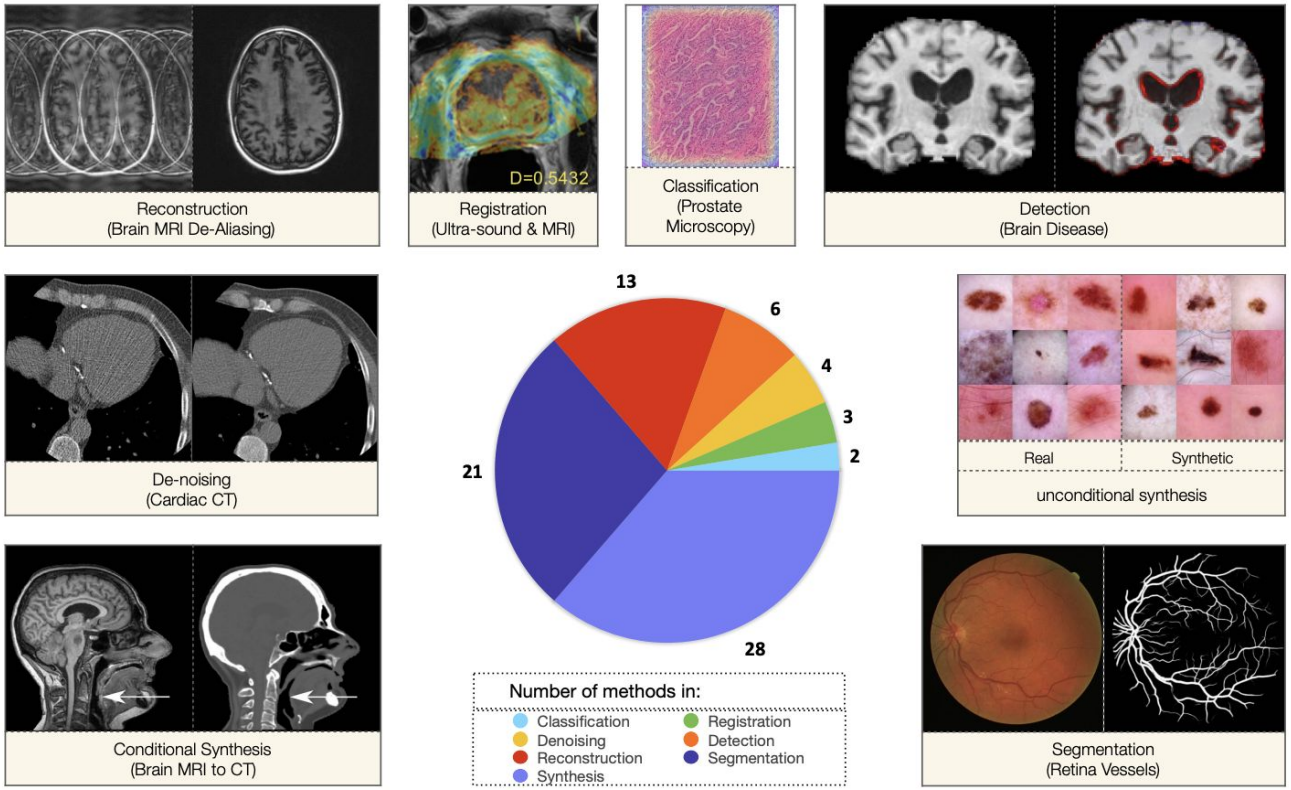



Figure 1: The pie chart of distribution of papers and visual examples of GAN functionality among the different applications. Examples are taken from papers as the following: Conditional synthesis Wolterink et al. (2017a), Denoising Wolterink et al. (2017b), Reconstruction Zhang et al. (2018), Registration Yan et al. (2018), Classification Ren et al. (2018), Detection Baumgartner et al. (2018), Unconditional synthesis Baur et al. (2018b), and Segmentation Son et al. (2017).

# *GAN

- A ton of different GAN architectures have been created
  - cGAN
  - SeGAN
  - DCGAN
  - ACGAN
  - CycleGAN
  - MGAN
  - LSGAN
  - VAE-GAN
  - WGAn
  - ...

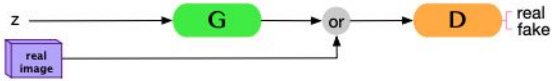



Figure 2: GAN

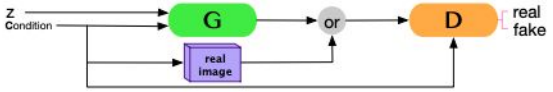



Figure 3: cGAN

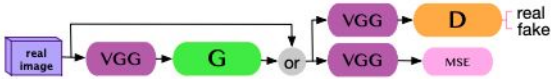



Figure 4: MGAN

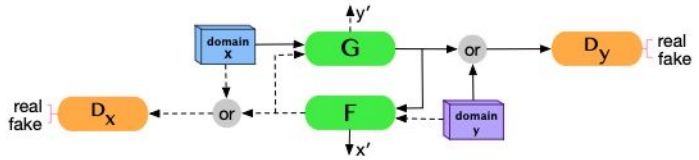



Figure 5: cycleGAN

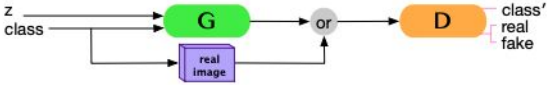
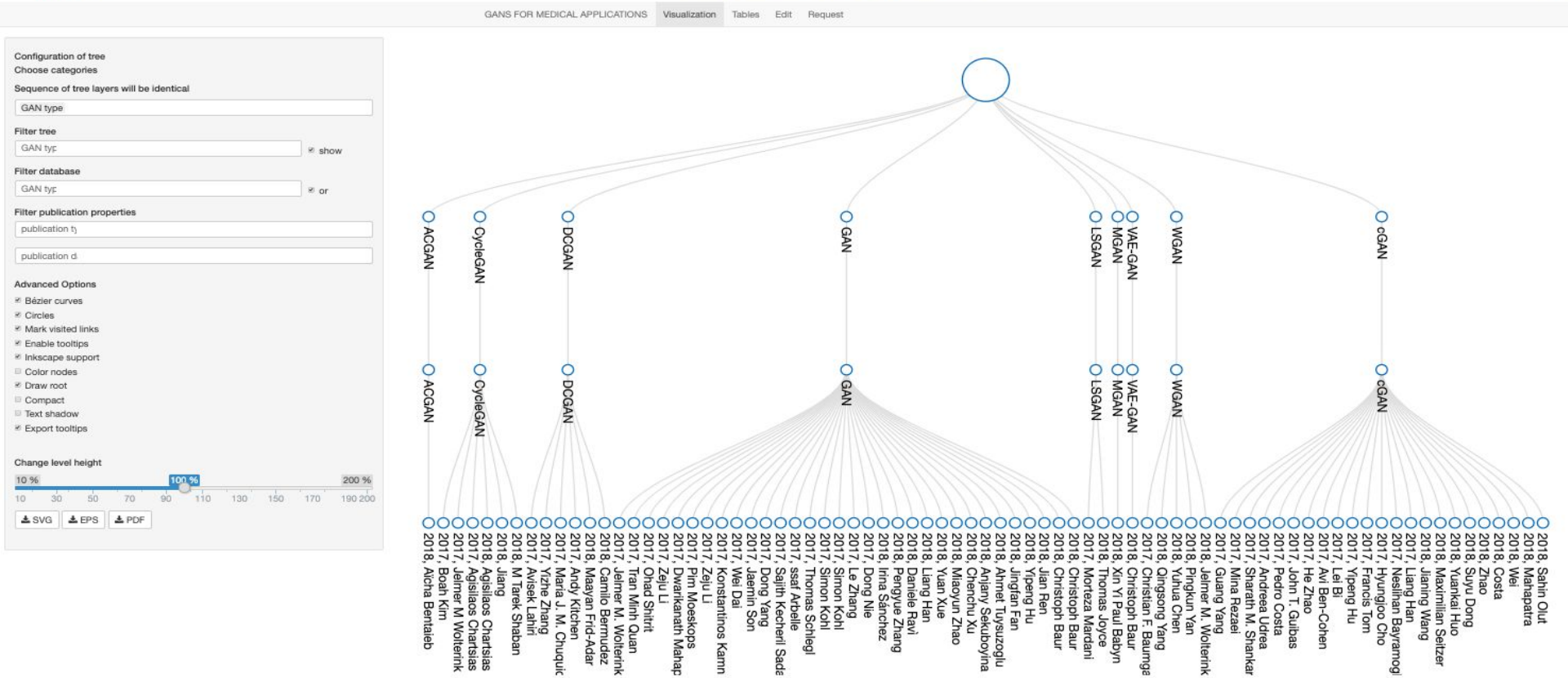



Figure 6: AC-GAN




http://livingreview.in.tum.de/GANs_for_Medical_Applications/

# GANs for Medicine

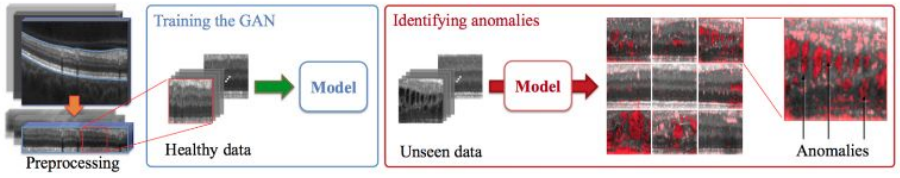
**Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery**

Thomas Schlegl[1,2] *, Philipp Seeböck[1,2], Sebastian M. Waldstein[2], Ursula Schmidt-Erfurth[2], and Georg Langs[1]

[1]Computational Imaging Research Lab, Department of Biomedical Imaging and Image-guided Therapy, Medical University Vienna, Austria
thomas.schlegl@meduniwien.ac.at
[2]Christian Doppler Laboratory for Ophthalmic Image Analysis, Department of Ophthalmology and Optometry, Medical University Vienna, Austria

**Abstract.** Obtaining models that capture imaging markers relevant for disease progression and treatment monitoring is challenging. Models are typically based on large amounts of data with annotated examples of known markers aiming at automating detection. High annotation effort and the limitation to a vocabulary of known markers limit the power of such approaches. Here, we perform unsupervised learning to identify anomalies in imaging data as candidates for markers. We propose *AnoGAN*, a deep convolutional generative adversarial network to learn a manifold of normal anatomical variability, accompanying a novel anomaly scoring scheme based on the mapping from image space to a latent space. Applied to new data, the model labels anomalies, and scores image patches indicating their fit into the learned distribution. Results on optical coherence tomography images of the retina demonstrate that the approach correctly identifies anomalous images, such as images containing retinal fluid or hyperreflective foci.

https://arxiv.org/pdf/1703.05921.pdf




AnoGAN

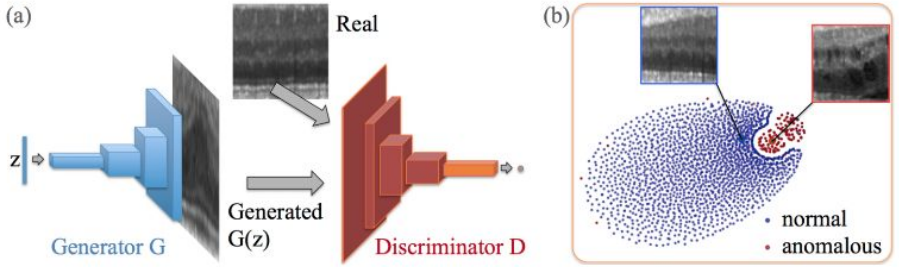



**Fig. 2.** (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

AnoGAN

# Reinforcement Learning

# Reinforcement Learning (RL)

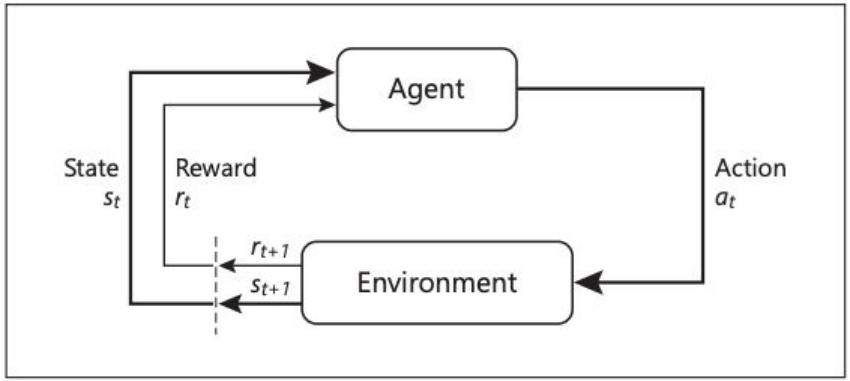
- A subfield of AI that provides tools to optimize **sequences of decisions** for **long-term outcomes**

- Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal

- The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them
  - Lots of interacting with environment
  - Lots of trial and error
  - A decision will affect not only the next action, but actions after that as well

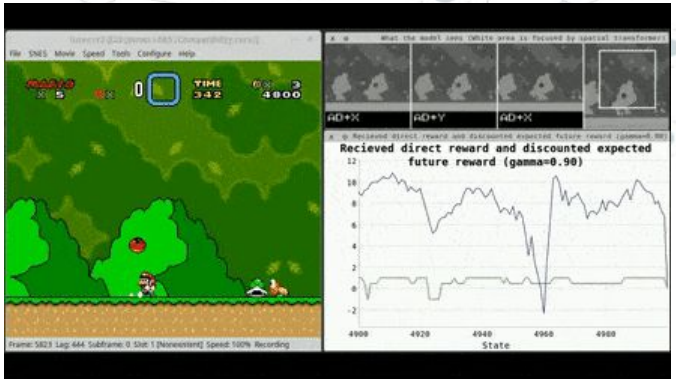
# RL

- Framework
  - Trial and error search, and delayed reward
  - Input: sequences of interactions (called **histories**) between the decision maker and their environment
  - At every decision point, the RL algorithm chooses an **action** according to its policy and receives new observations and immediate outcomes (often called **rewards**)

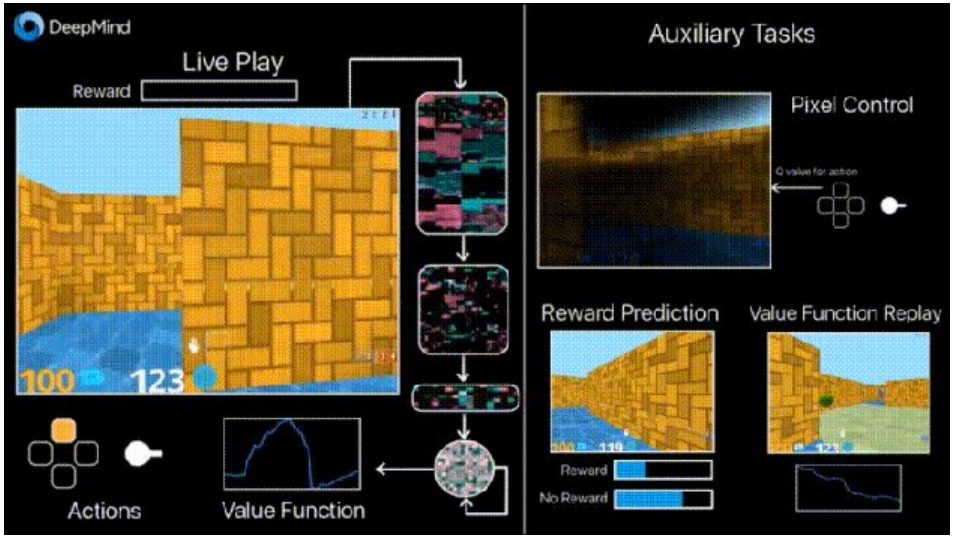


# RL

- Has been really popular with games
  - AlphaGo is better than the best Go players in the world







# RL in Healthcare

- Still a recent method being applied in healthcare contexts

- Examples
  - Optimizing antiretroviral therapy in HIV
  - Tailoring antiepilepsy drugs for seizure control
  - Determining the best approach to managing sepsis

- Rather than a one-time prediction, RL affects a patient's future health and future treatment options
  - Long-term effects are more difficult to estimate

https://www.nature.com/articles/s41591-018-0310-5

https://towardsdatascience.com/a-review-of-recent-reinforcment-learning-applications-to-healthcare-1f8357600407

# Sepsis Example

- There is wide variability in the way clinicians make decisions about sepsis management
  - Can RL help with this?

- **History**: may include a patient's vital signs and laboratory tests

- **Actions**: all the treatments available to the clinician, including medications and interventions

- **Rewards**: require clinician input - they should represent the achievement of desirable tasks, such as stabilization of vital signs or survival at the end of the stay
  - Short-term: liberation from mechanical ventilation
  - Long-term: prevention of permanent organ damage
  - https://arxiv.org/pdf/1711.09602.pdf

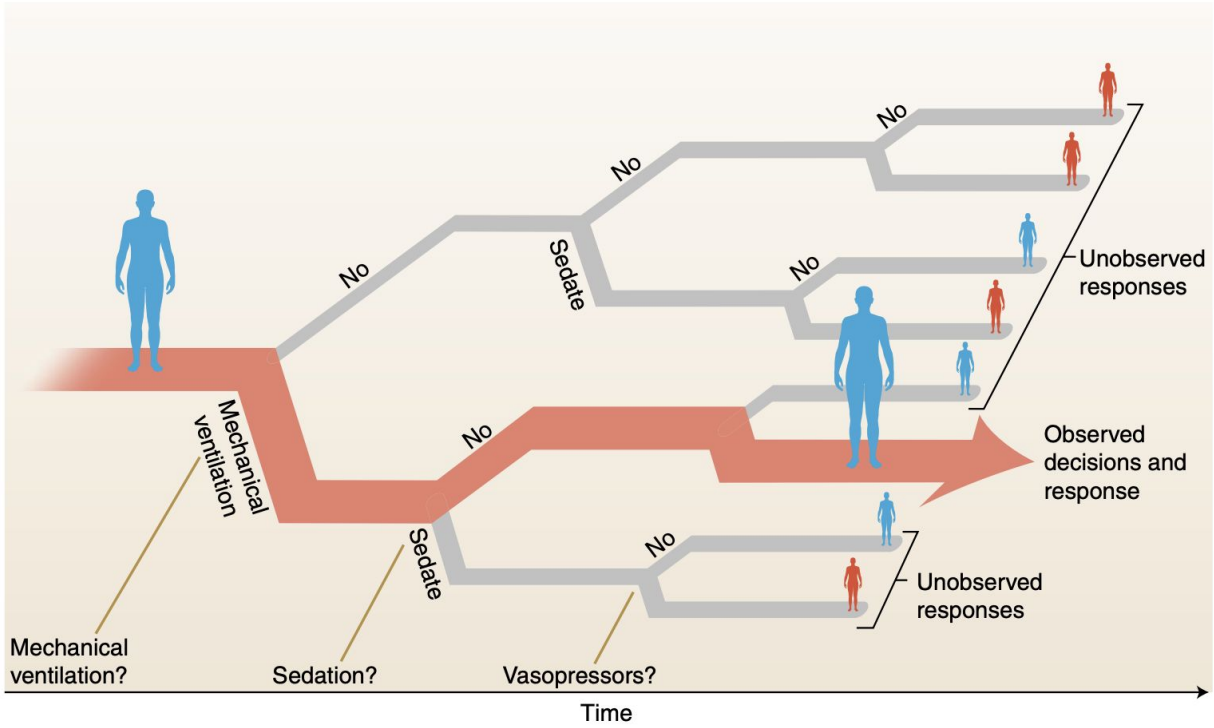



**Fig. 1 | Sequential decision-making tasks.** To perform sequential decision making, such as for sepsis management, treatment-effect estimation must be solved at a grand scale—every possible combination of interventions could be considered to find an optimal treatment policy. The diagram shows the scale of such a problem with only three distinct decisions. Blue and red people denote positive and negative outcomes, respectively. Credit: Debbie Maizels/Springer Nature

# Challenges

- We only observe one set of actions and rewards for each patient
  - We can't keep trying different combinations of actions to optimize a reward - forced to use previous observational data, called "off-policy" learning
- We don't observe everything going on in the body
  - We also don't observe the values we do record (blood pressure, etc.) at every time step (dynamic data)
- It's difficult to find a reward function
  - How do we balance short and long-term rewards?
- Need a ton of data, which is difficult to come by

# Questions to Consider

- Is the AI given access to all variables that influence decision making?
  - Confounding variables
  - Can lead to confounding in the short term and long term

- How big is your effective sample size?
  - Most approaches for evaluating RL policies from observational data weigh each patient's history on the basis of whether the clinician decisions match the decisions of the policy proposed by the RL algorithm

  - The reliability (variance) of the treatment-quality estimate depends on the number of patient histories for which the proposed and observed treatment policies agree—a quantity known as the effective sample size

  - The possibilities for mismatch between the actual decision and the proposed decision grow with the number of decisions in the patient's history, and thus RL evaluation is especially prone to having small effective sample sizes

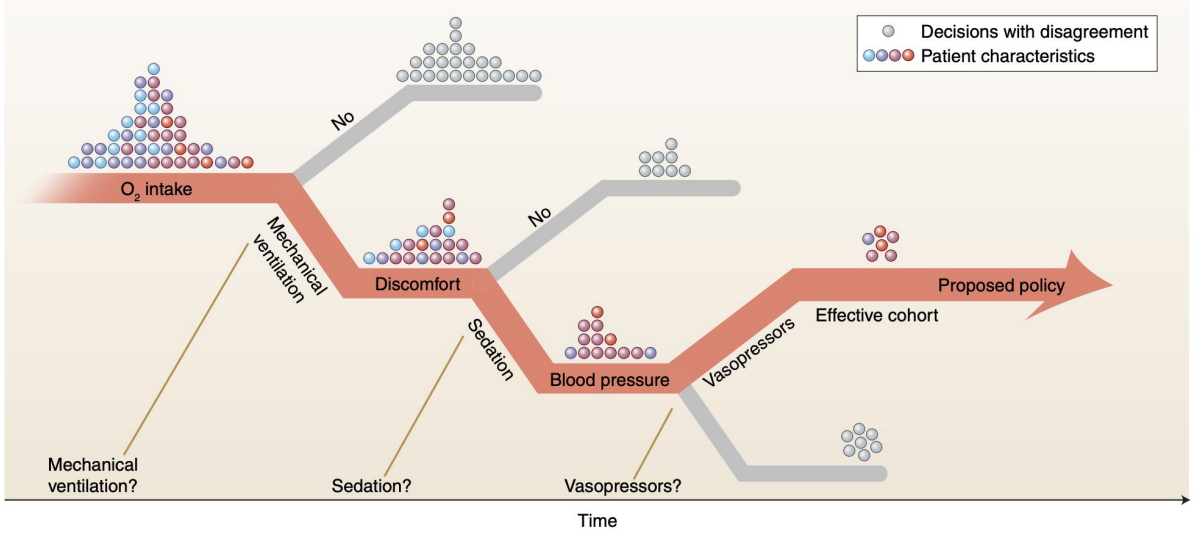



**Fig. 2 | Effective sample size in off-policy evaluation.** Each dot represents a single patient at each stage of treatment, and its color indicates the patient's characteristics. The more decisions that are performed in sequence, the likelier it is that a new policy disagrees with the one that was learned from. Gray decision points indicate disagreement. Use of only samples for which the old policy agrees with the new results in a small effective sample size and a biased cohort, as illustrated by the difference in color distribution in the original and final cohort. Credit: Debbie Maizels/Springer Nature

# Questions to Consider

- Will the AI behave prospectively as intended?
  - Errors in problem formulation or data processing can lead to poor decisions
  - Simplistic reward functions may neglect long-term effects for meaningless gains: for example, rewarding only blood pressure targets may result in an AI that causes long-term harm by excessive dosing of vasopressors
  - Errors in data recording or preprocessing may introduce errors in the reward signal, misleading the RL algorithm
  - The learned policy may not work well at a different hospital or even in the same hospital a year later if treatment standards shift

# RL in Medicine

- RL in medicine seems promising, but difficult
- May help guide clinicians in treatment decisions based on all of a patient's history and not their immediate symptoms/responses to treatment

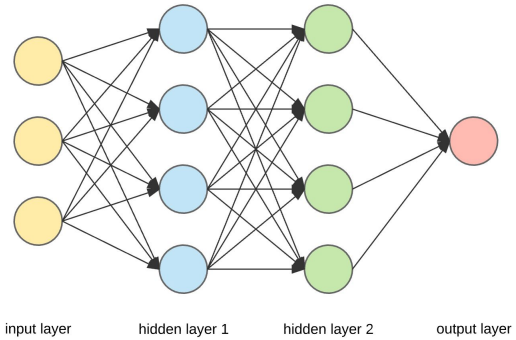
# Course Review

# What have we learned?

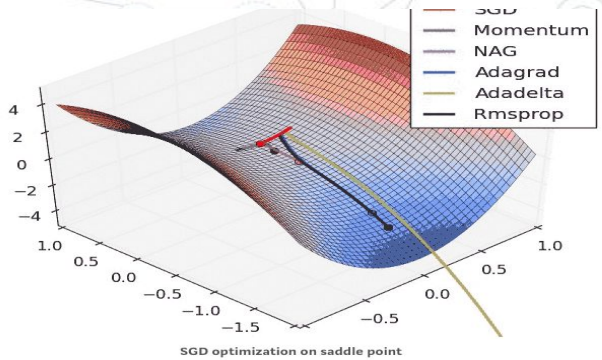
Quite a bit!

- Multilayer perceptrons → CNNs → RNNs → Advanced architectures

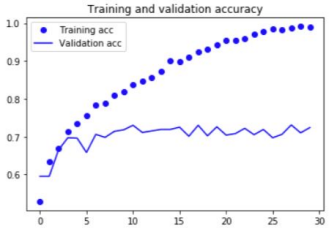
- Network architecture
  - Hidden units
  - Layers
  - Activation function
  - Loss functions
  - Optimization algorithms
  - Batch size



# What have we learned?




- How networks learn
  - Gradient descent/ascent
  - Backpropagation
  - Forward pass and backward pass
  - Visualizing filters
  - Dense layer vs convolution layer vs RNN/LSTM/GRU layer

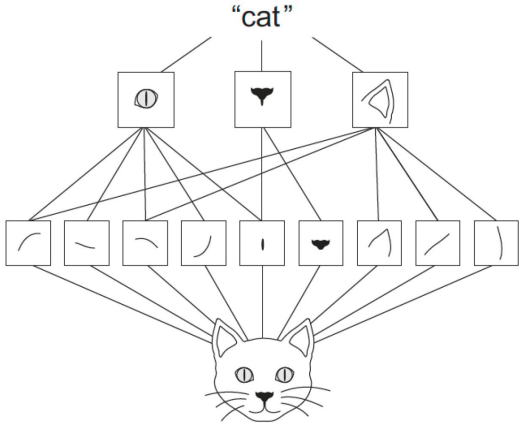


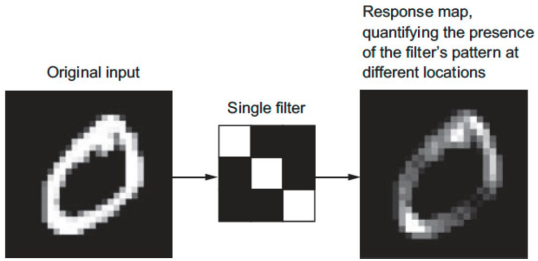


- Model performance
  - Underfitting vs overfitting
  - Regularization techniques
    - Dropout, L2 and L1 norms, network size
  - Bias/variance tradeoff

# What have we learned?

- CNNs
  - Padding
  - Pooling
  - Strides
  - Filters
  - Translation invariance
  - Hierarchical learning
  - Lower layer representations vs higher layer representations
  - Data format (3D vs 4D tensors)
  - Object detection and localization
  - Face recognition
  - 1D CNN for sequential data
  - Landmark detection
  - Data augmentation
  - Neural style transfer

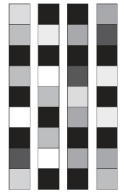


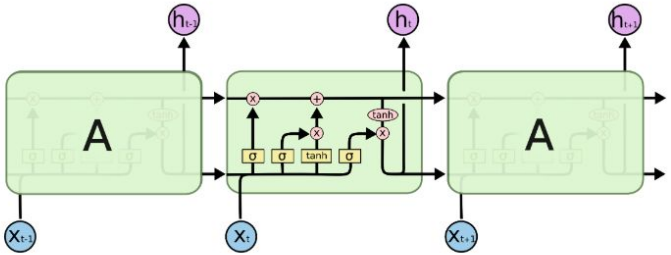
# What have we learned?

- RNNs
  - Different types of sequential data
  - How RNNs preserve order in this type of data
  - SimpleRNN vs LSTM vs GRU layers
  - Tokens and tokenization
  - One-hot encoding and hashing
  - Word embeddings
  - Word2Vec and Glove
  - Time series data
  - Recurrent dropout
  - Text generation
  - Bidirectional recurrent layers

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



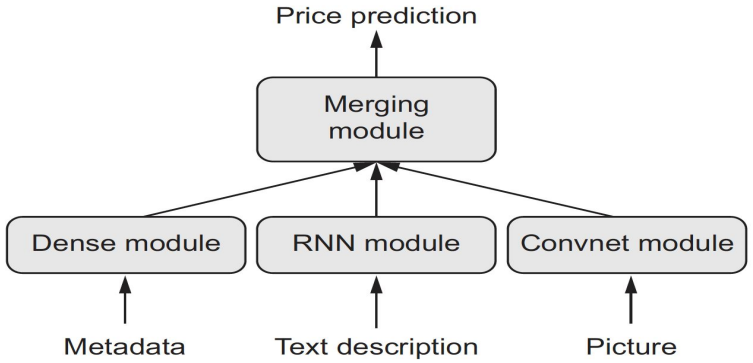
Word embeddings:
- Dense
- Lower-dimensional
- Learned from data




The repeating module in an LSTM contains four interacting layers.

# What have we learned?

- Advanced network architectures
  - One-to-many (multi-output/multi-head models)
  - Many-to-many
  - Many-to-one (multi-modal models)
  - Directed acyclic graphs
- Advanced architecture patterns
  - Batch normalization
  - Hyperparameter optimization
  - Model ensembling
- Implementation in Keras
  - Tensor (data) manipulation
  - Sequential model
  - MLP, CNN, RNN
  - Using GCP
  - Functional API



# What have we learned?

- Advanced topics
  - Variational autoencoders (VAEs)
  - Generative adversarial networks (GANs)
  - Reinforcement learning (RL)
  - DeepDream
  - Neural style transfer
  - Text generation