# Software Engineering: OOP illustrated through Density Estimating Neural Networks

# **GANs**


**Albert Montillo, Son Nguyen**

UTSouthwestern

Departments: Lyda Hill Department of Bioinformatics, Radiology, and Advanced Imaging Research Center


May 8-12, 2023

SWE course

Lyda Hill Department of Bioinformatics

# Outline

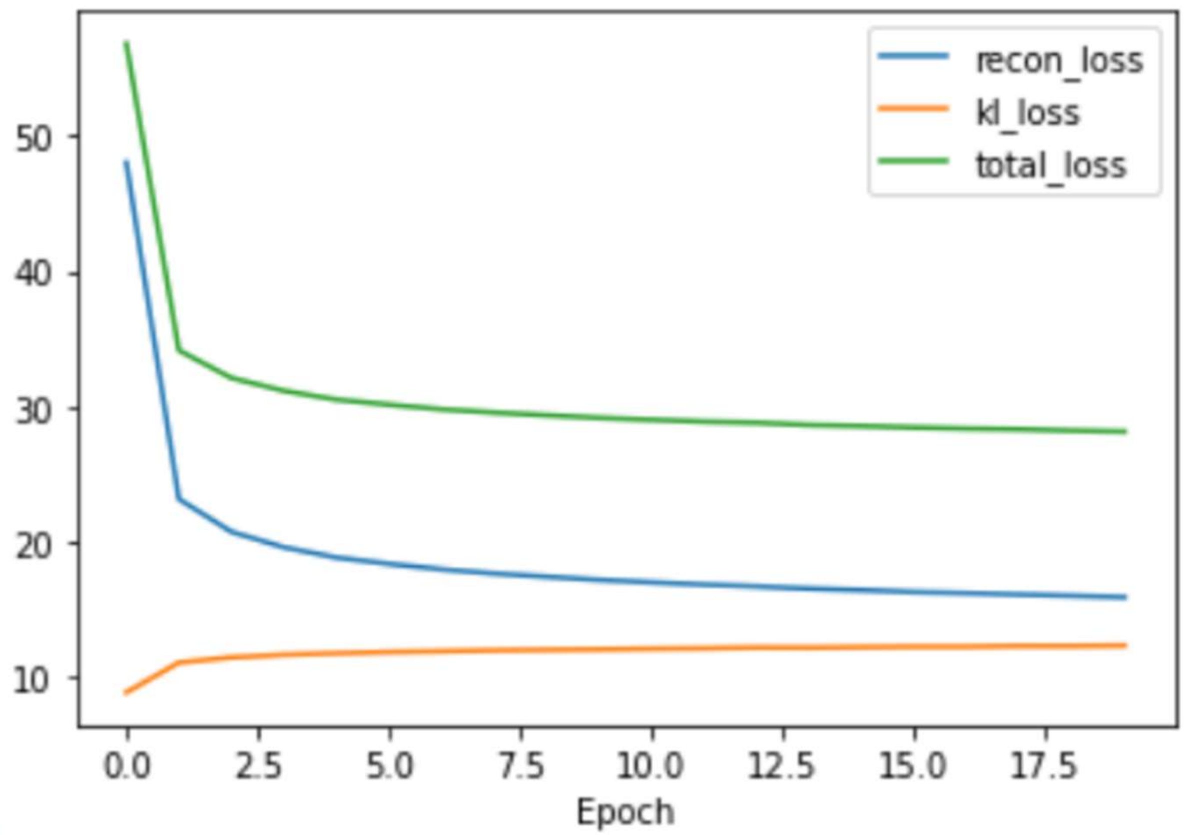
1. **Monday**
   1. **Review OOP, Image I/O, Keras**
   2. **New topic: Object Oriented  Variational Autoencoders (VAEs)**
2. **Tuesday**
   1. **Review observations on VAE**
   2. **New topic: Symbolic debugger: cond breakpoints and call stack traversal**
   3. **New topic: Object Oriented  Generative Adversarial Networks (GANs)**
3. **Wednesday**
   1. **Review GAN observations**
   2. **New topic: Object Oriented *conditional* VAEs (cVAE) and *Auxillary Classifier* GANs (AKA cGAN or acGAN)**
   3. **New topic: Motivate a possible combination of cVAE and cGAN**
4. **Thursday**
   1. **Review cVAE, cGAN observations**
   2. **Review cVAE-cGAN code**
   3. **New topic: Hyperparameter optimization**
   4. **New topic: training curve and latent space traversal and visualization**

# Observations from the Hands on VAE exercise

3

# VAE Training curves

1. **Shows evolution of reconstruction and regularizing prior ($D_{KL}$) loss as well as the total loss (their sum)**
2. **We observe that the majority of the learning took place in the first 10 epochs. We could train longer but diminishing returns**



# VAE Reconstructed images at epoch 1 epoch001_recons.png

Real:

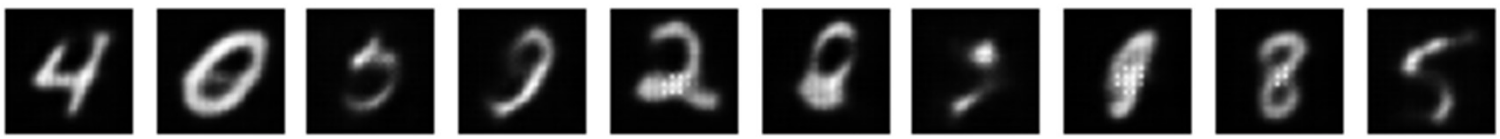
Recon:

# VAE Reconstructed images at epoch 10

# VAE Reconstructed images at epoch 20

# VAE Purely synthesized images at epoch 1 epoch001_fakes.png

**Note: we cannot steer (choose ) class label of the generated image. Images from 10 random Z's shown**

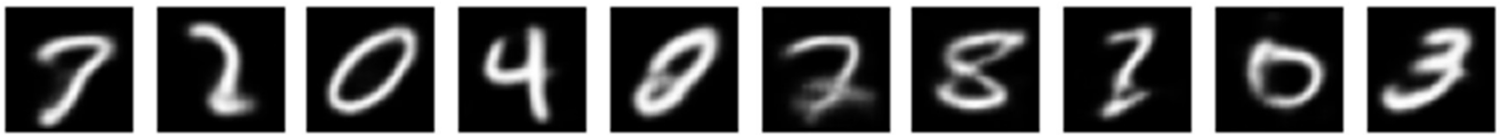


# VAE purely synthesized images at epoch 10

**Note: we cannot steer (choose ) class label of the generated image. Images from 10 random Z's shown**

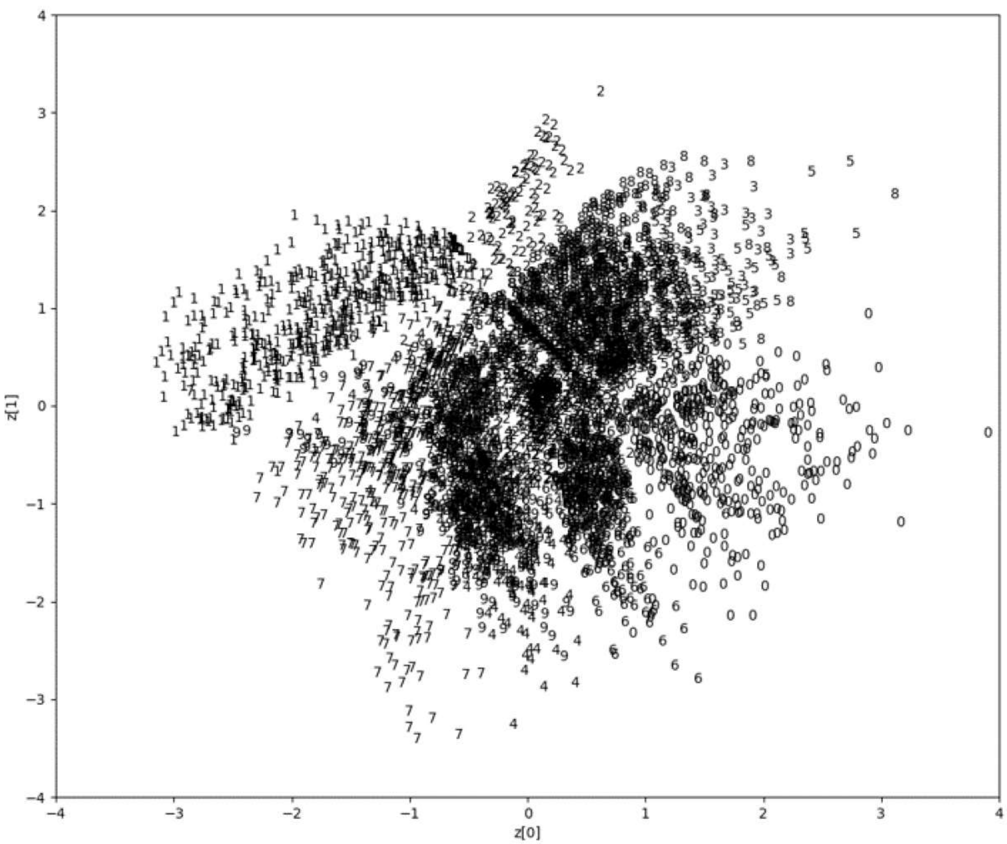
# VAE purely synthesized images at epoch 20

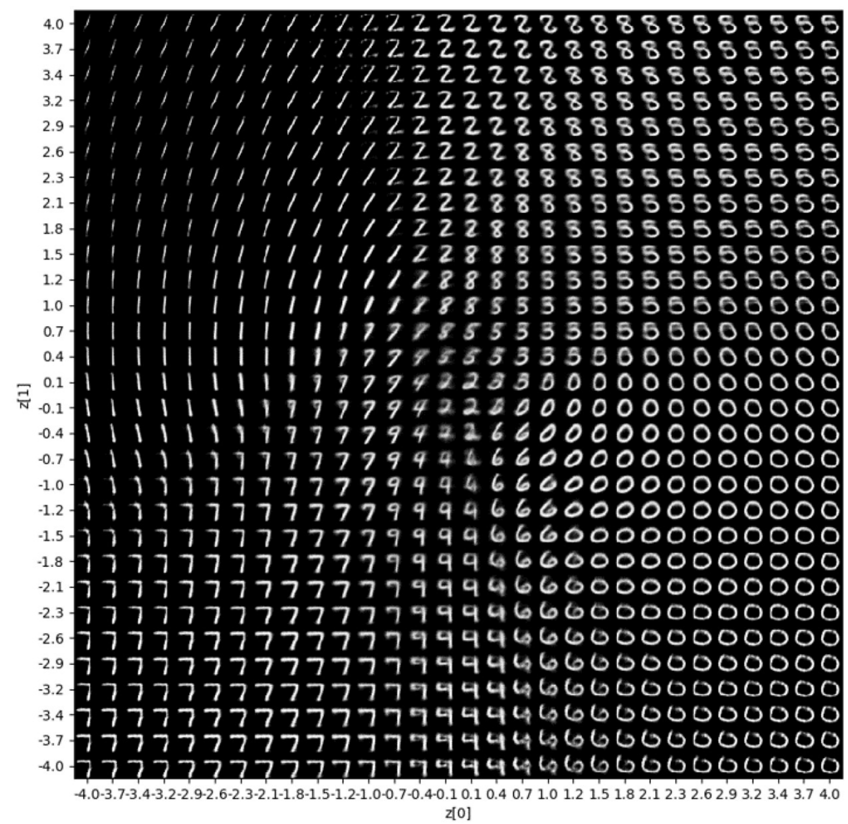
**Note: we cannot steer (choose ) class label of the generated image.**
**Images from 10 random Z's shown**



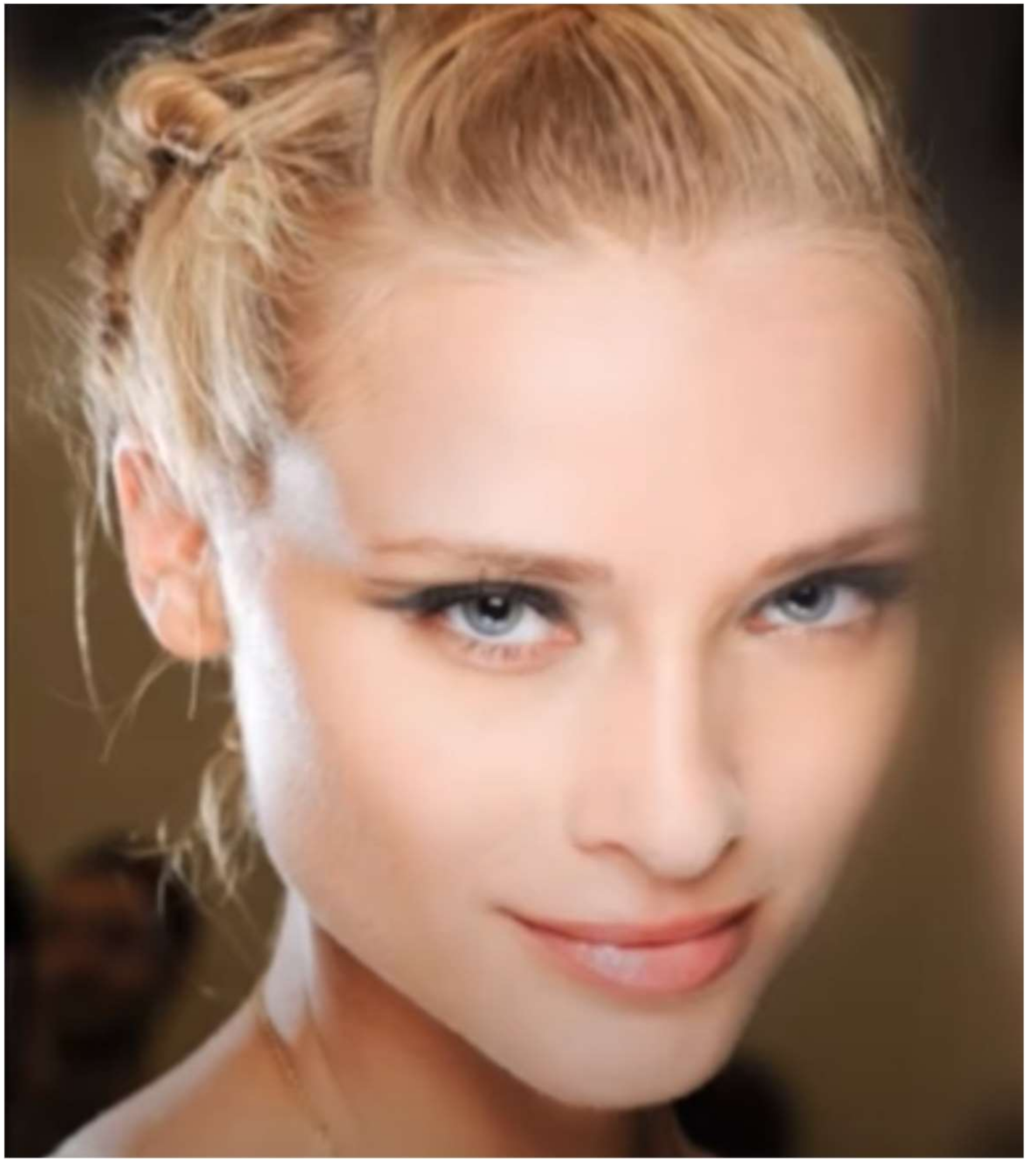
# Upshot of VAE results

- **Reasonable digits are produced**
- **Strokes are nice and straight, not wavy**
- **Borders of digits are blurry**
- **We can also traverse the Z space to explore nearby digits, which is smooth and contiguous through the prior we enforced.**
  - **Contrast with GAN ..**

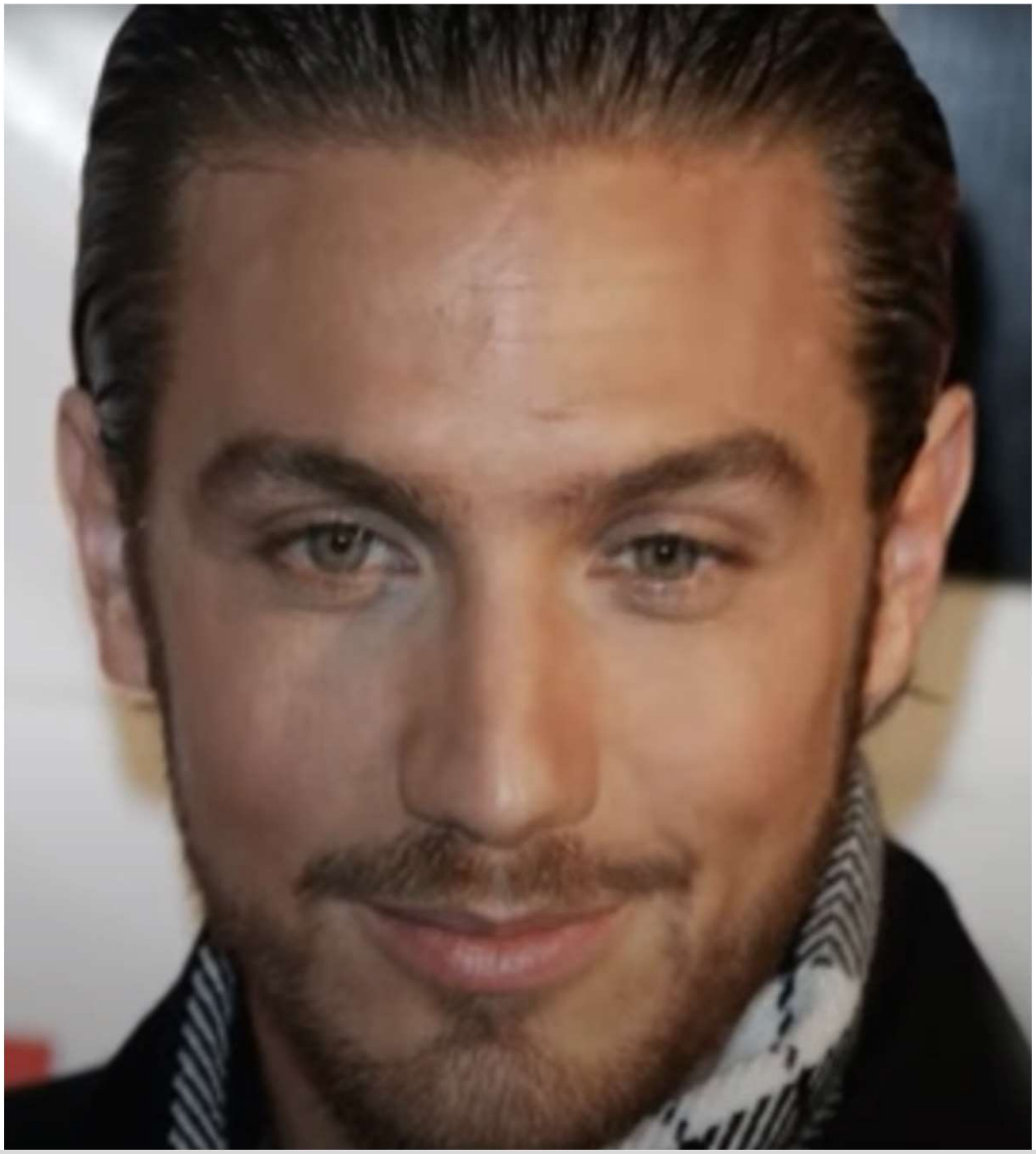


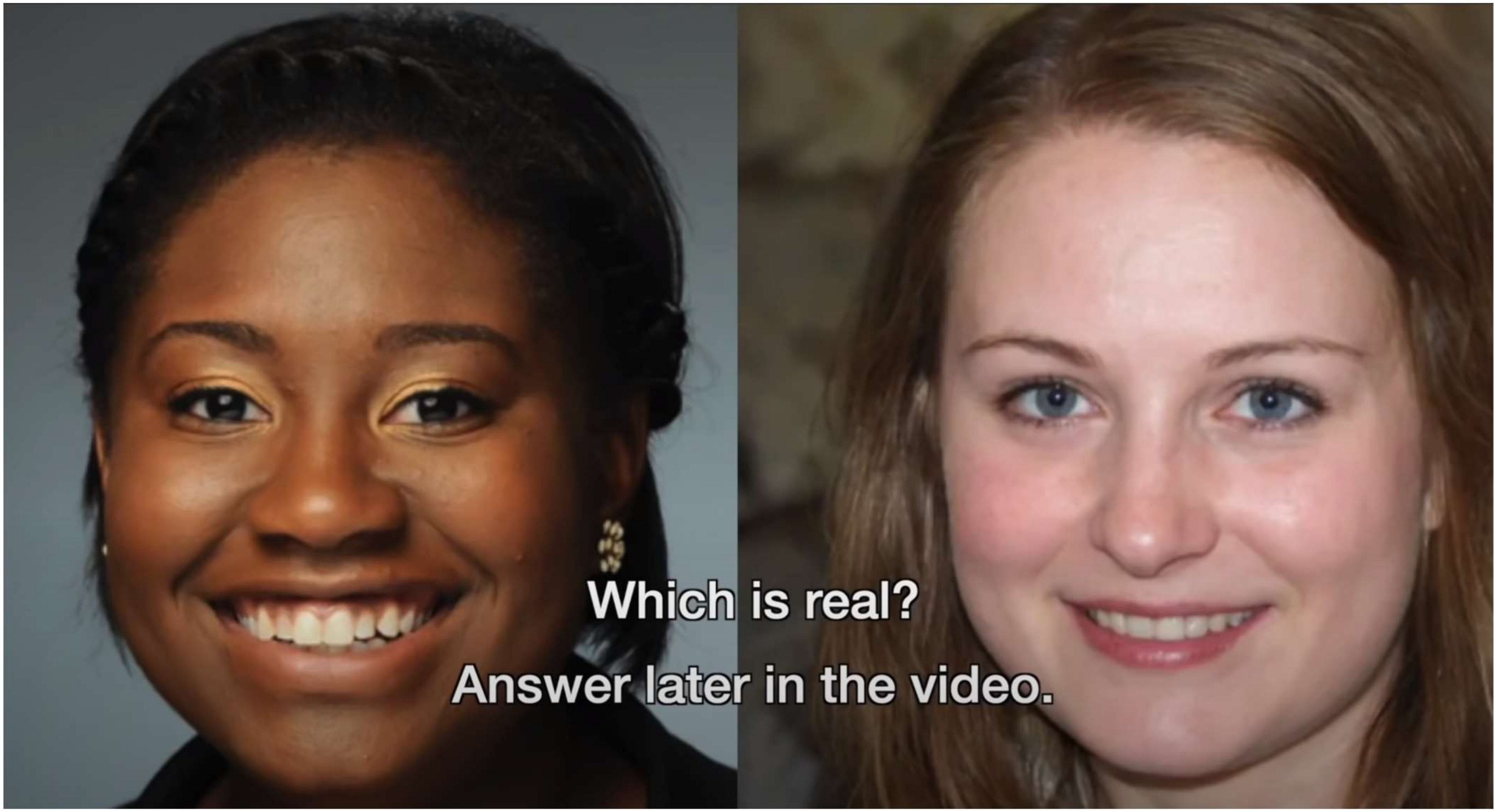



# Who is this?



# Who is this?






https://www.youtube.com/watch?v=ixgFtjfO_7Q&t=52s&ab_channel=DigitalEngine


https://www.youtube.com/watch?v=ixgFtjfO_7Q&t=52s&ab_channel=DigitalEngine

# Generative Adversarial Networks (GANs)

- **Introduced in 2014 by Ian Goodfellow**

- **Why do we need GANs? What are they useful for?**
  - **To generate new data to augment when labeled training data is scarce**
  - **To generate new data to be used as is (to understand a density, advertising)**

- **Why is it adversarial?**
  - **It is a competition between 2 components**

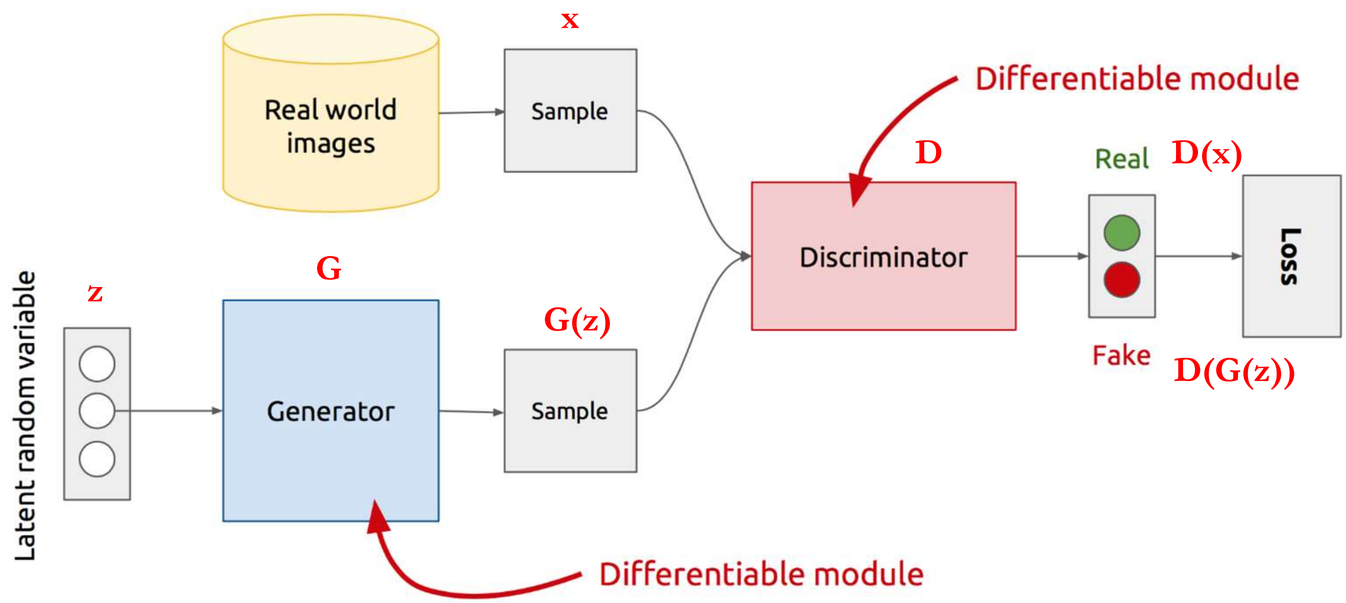
  - **Generator (counterfeiter) produces fake data by mimicking real data**
  - **Discriminator (cop) … distinguishes real from fake data (to catch the counterfeiter)**

- **Discriminator’s loss is passed to the Generator as an objective function**

## Steps to train a GAN

1. Define the problem
2. Define the GAN architecture
3. Train Discriminator to distinguish real from fake data
4. Train Generator to synthesize new data
5. Repeat steps 3 and 4 until convergence
6. When training is finished, synthesize new data from generator

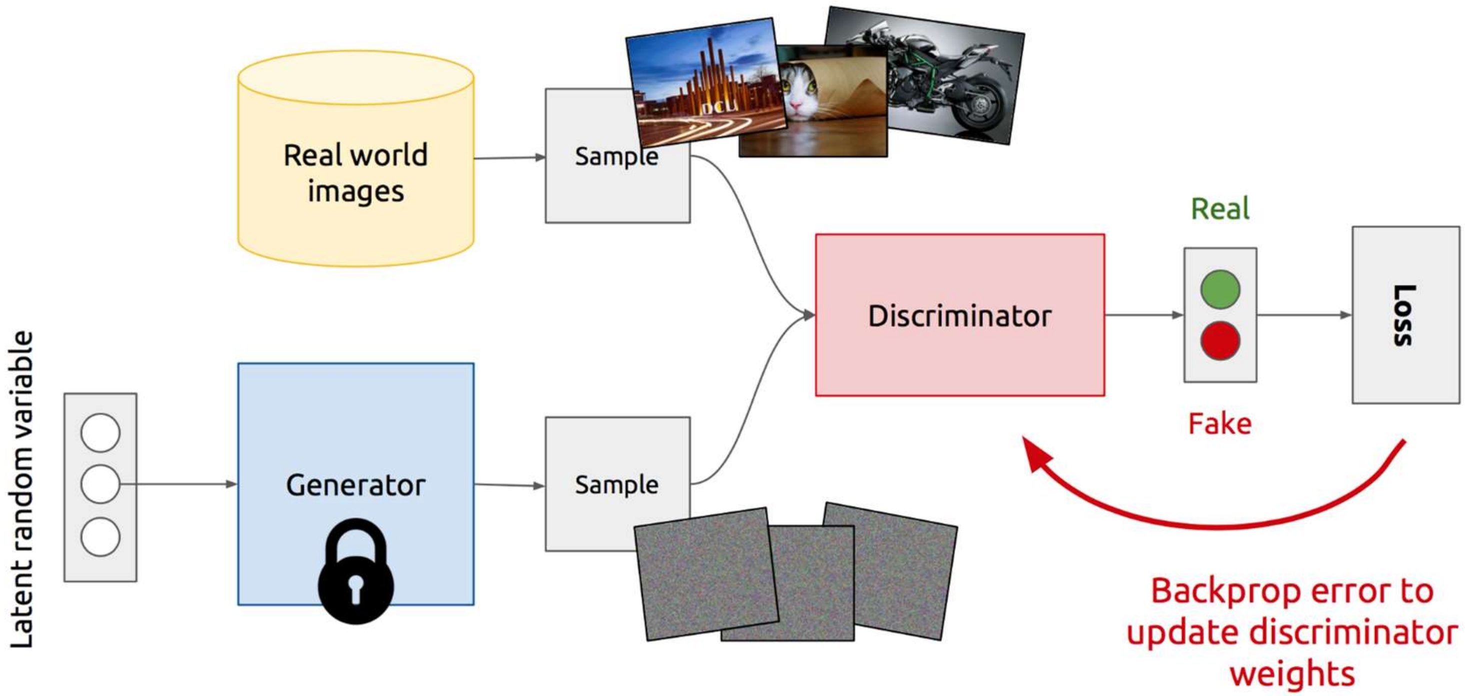
# Architecture of a GAN




- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

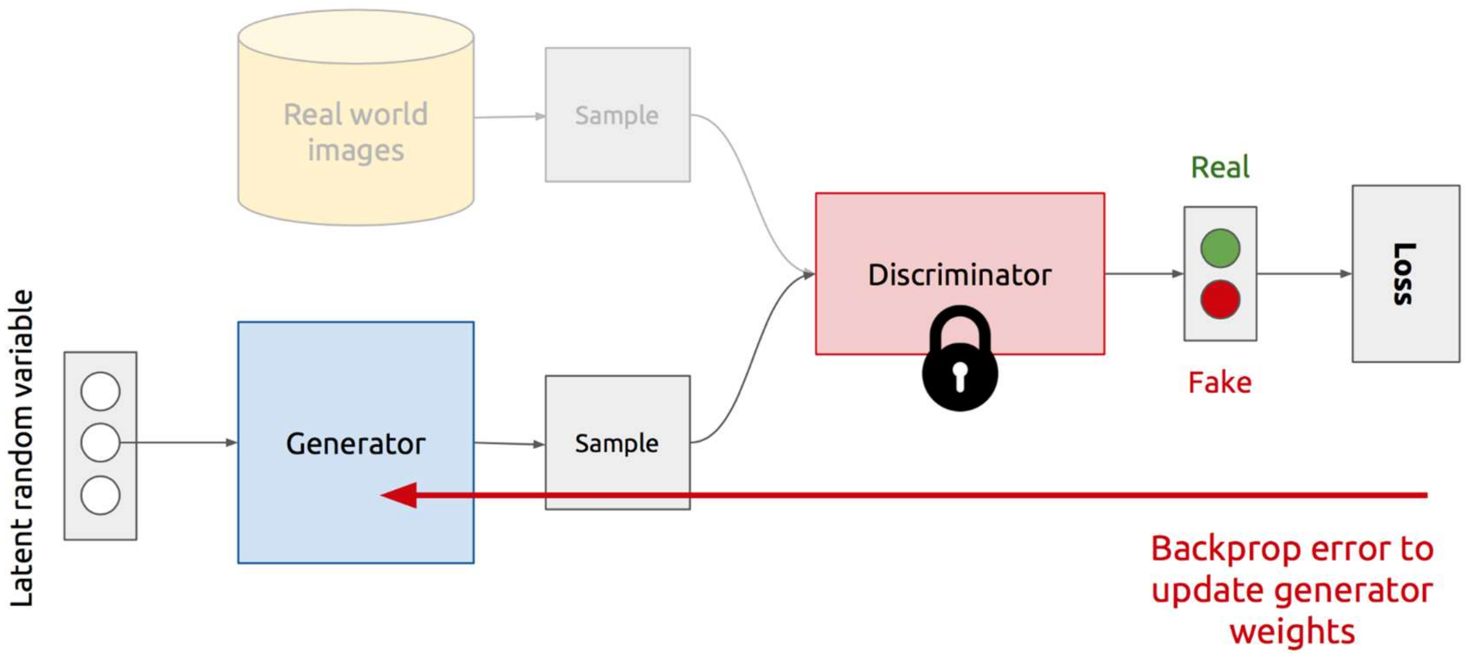
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

# Discriminator training




https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

# Generator training




https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

# Formulation of the GAN

$$\min_G \max_D V(D, G)$$

**It is formulated as a minimax game, where:**

- **The Discriminator is trying to maximize its reward $\boldsymbol{V(D, G)}$**
- **The Generator is trying to minimize Discriminator's reward (or maximize its loss)**

$$V(D, G) = \boxed{\mathbb{E}_{x \sim p(x)}[\log D(x)]} + \boxed{\mathbb{E}_{z \sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]}$$

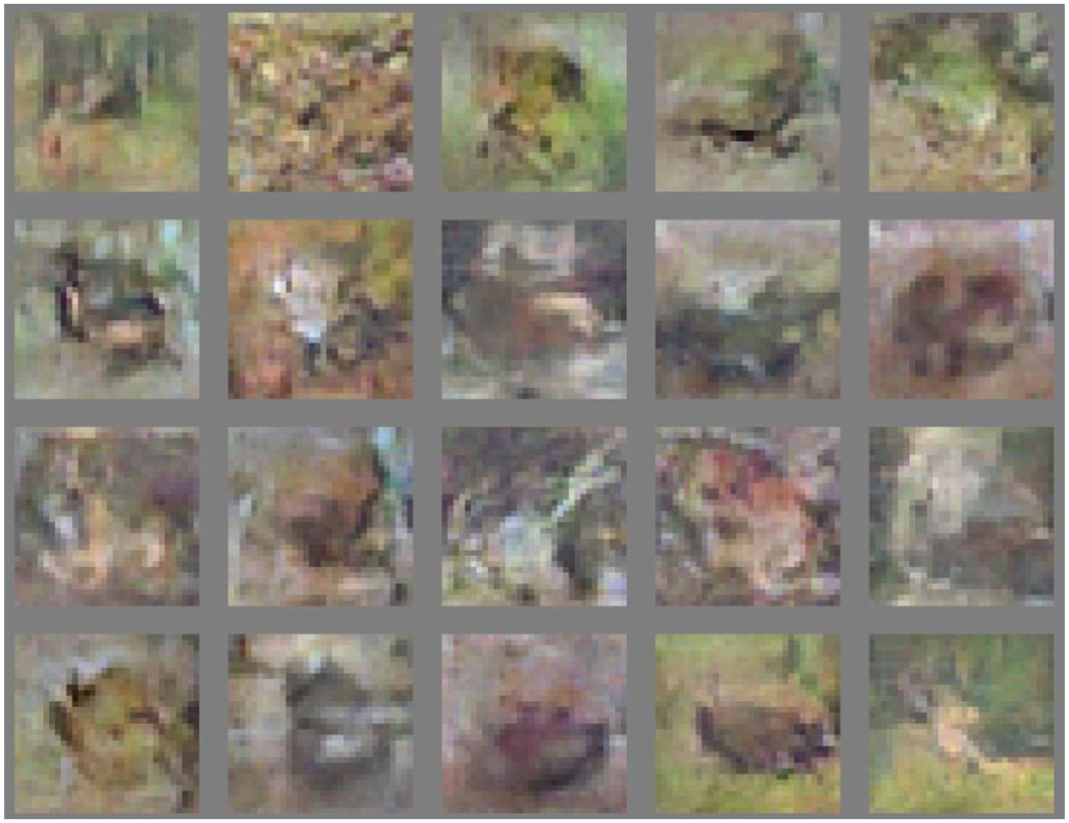
**The Nash equilibrium of this particular game is achieved at:**

- $P_{data}(x) = P_{gen}(x) \;\; \forall x$
- **D**$(x) = \frac{1}{2} \;\; \forall x$

Adapted from http://slazebni.cs.illinois.edu/spring17/lec11_gan.pptx

# Steps to train a GAN

**CIFAR**

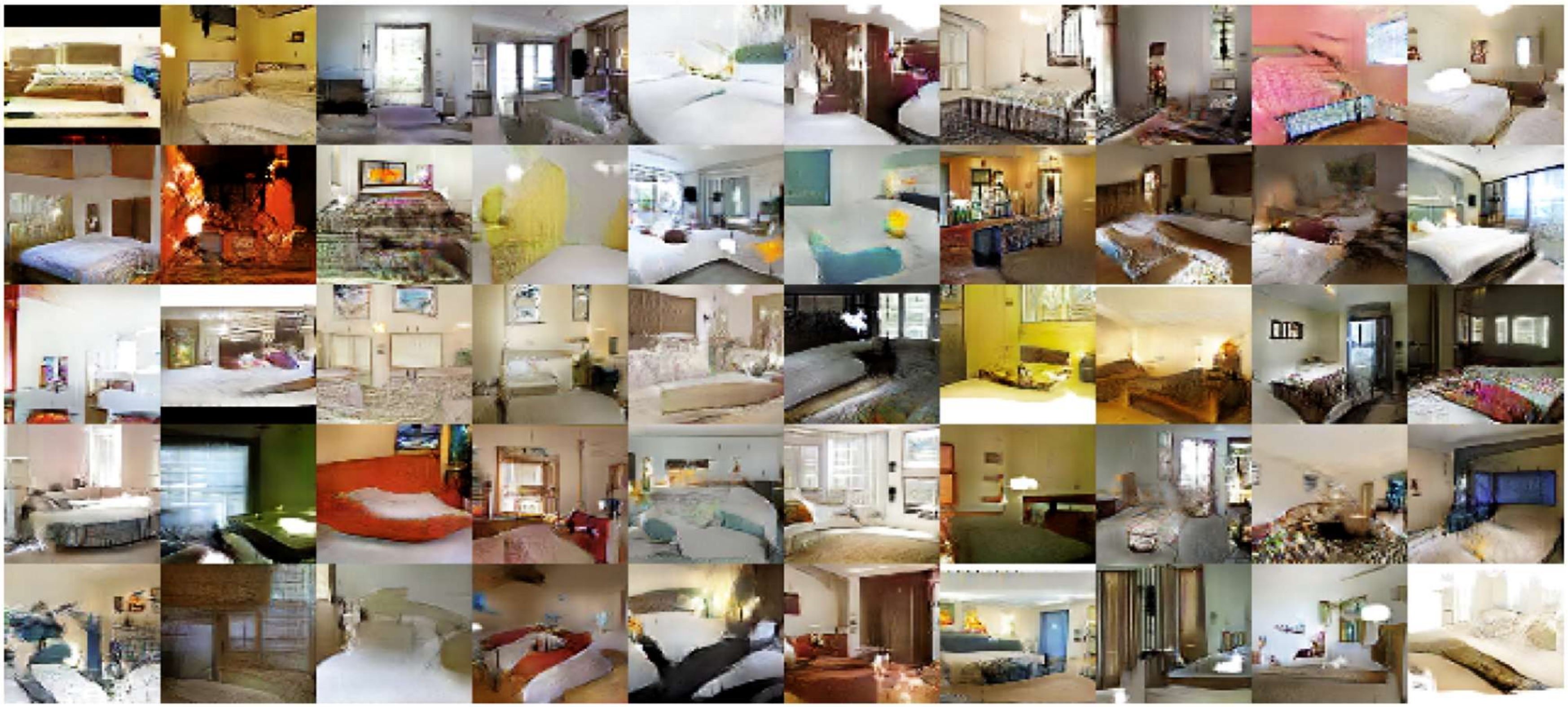


Goodfellow, Ian, et al. **"Generative adversarial nets."** *Advances in neural information processing systems*. 2014.


Deep Learning for Precision Health Lab
www.UTSouthwestern.edu/labs/Montillo
UTSouthwestern
Lyda Hill Department of Bioinformatics

# Steps to train a GAN

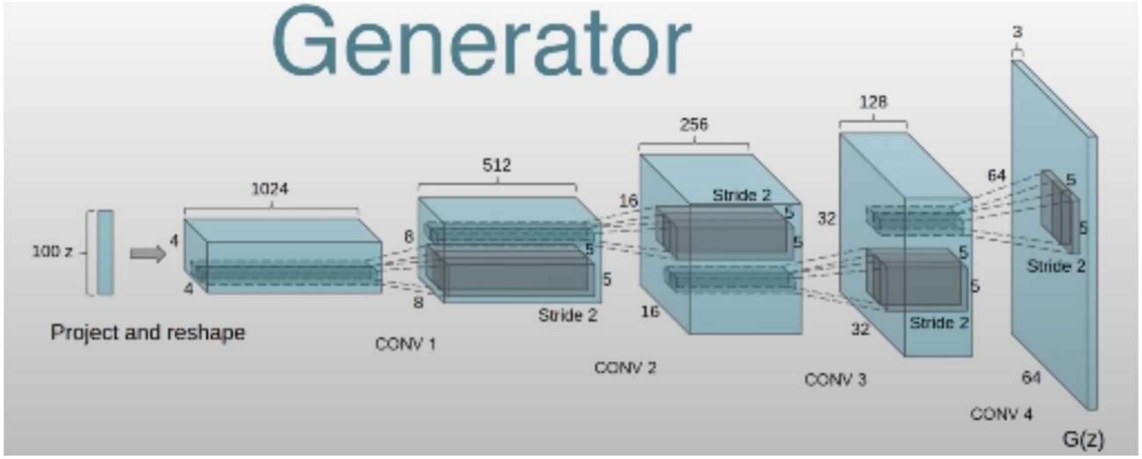
**DCGAN: Bedroom images**



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

# Deep convolutional GANs (DCGANs)

## Architecture




Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

**Key ideas**:

- Replace dense fully connected hidden layers with transposed convolutions
  - **Generator:** Strided transposed convolutions

- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Sigmoid for the output layer

# Deep convolutional GANs (DCGANs)

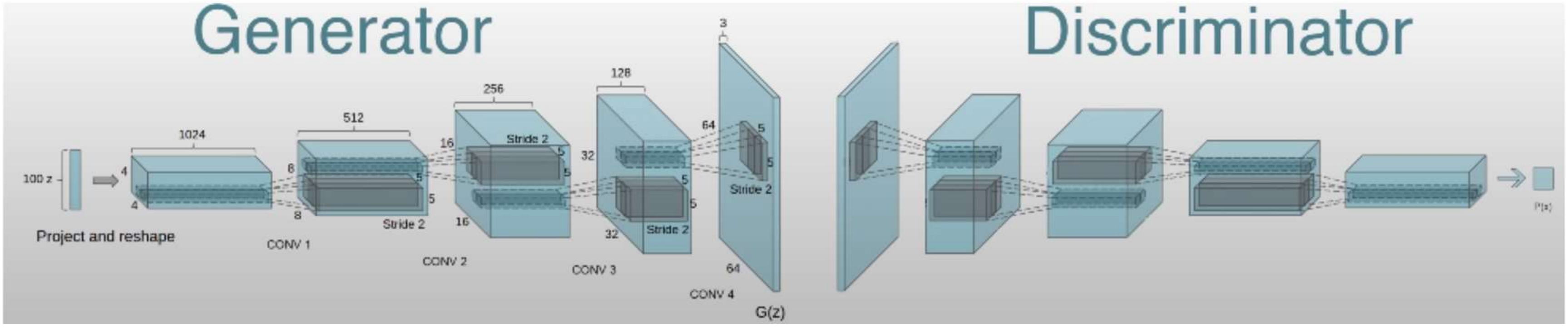
## Architecture


**Key ideas**:

- Replace dense fully connected hidden layers with **convolutions**
  - **Discriminator:** Strided **convolutions**
- Use Batch normalization after every layer
- **Inside Discriminator**
  - Use ReLU for hidden layers
  - Use Sigmoid for the output layer
- Output is probability of **real** image (authenticity classifier)

# Deep convolutional GANs (DCGANs)

## Architecture




**Key ideas**:

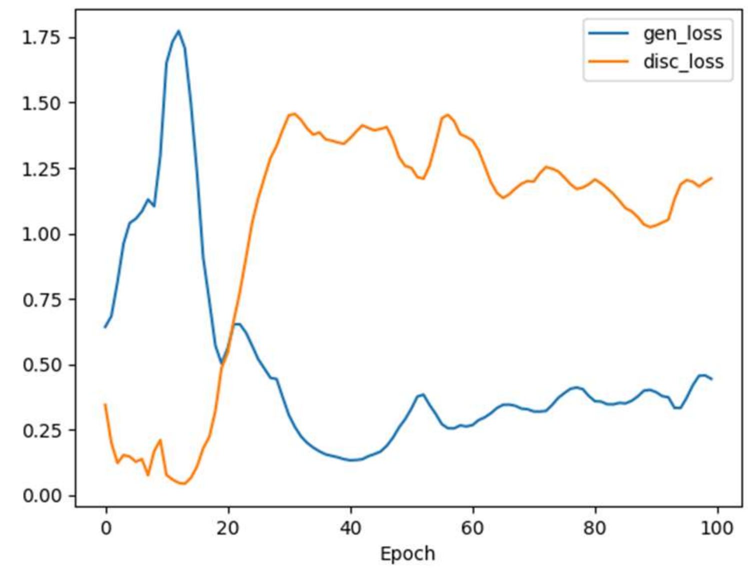
- Generator constructs an image from a random input (Z) with same dimensions as real images
- Discriminator receives images from real data and from generated
- Compatible networks
  - Generator produces images of a given size
  - Discriminator analyzes images of that size
- There is no weight tying.
- Freedom to construct Discriminator with different depth, so long as it is compatible with Generator

# Challenges when implementing GANs

## 1. Divergence

- Generated images get worse and worse with more training
- Reason: the learning rate for the Generator and Discriminator networks are not properly balanced
- Solutions:
  - decrease the discriminator learning rate to help achieve the necessary balance via hyperparameter optimization
    - Manual: graduate student descent
    - Automated: eg Bayesian Optimization
  - add noise to the real/fake labels when training the discriminator to restrain its learning.

- Example from GAN trained on AD. Required decreasing discriminator LR to 10% of the generator LR to get equilibrium in the losses. The two losses don't need to be equal in magnitude as long as they each stay roughly flat.

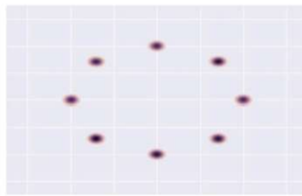


# Challenges when implementing GANs

## 2. Mode collapse

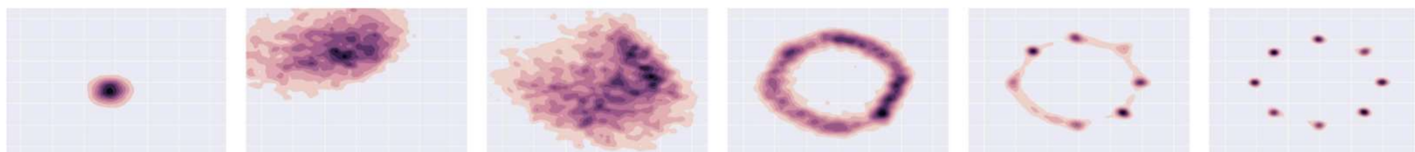
- **Example: given that real examples come from a 2D Gaussian Distribution (with 8 modes)**

Target



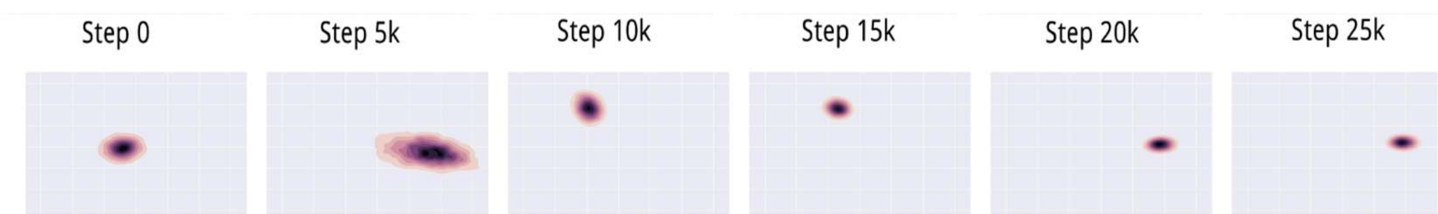
- **You would expect your generated distribution to learn to follow this distn.**

Expected



- **Instead: Generator fails to output diverse samples**
  - **During training generated samples move from one mode to another, and fail to capture all the modes .. Thus, you don't get diversity in the generated samples**

Output




Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

# Challenges when implementing GANs

**Solution: reward sample diversity**

**At Mode Collapse,**

- **Generator produces good samples, but a very few of them.**
- **Thus, Discriminator can’t tag them as fake.**

**To address this problem,**

- **Let the Discriminator know about this edge-case.**

**More formally,**

- **Let the Discriminator look at the entire batch instead of single examples**
- **If there is lack of diversity, it will mark the examples as fake**

**Thus,**

- **Generator will be forced to produce diverse samples.**

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

# Why use GANs for Generation?

**Can be trained using back-propagation for Neural Network based Generator/Discriminator functions.**

**Sharp images can be generated.**

**Fast to sample from the model distribution: *single* forward pass generates a *single* sample.**

# GANs Reading List

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative adversarial nets, NIPS (2014).

Goodfellow, Ian NIPS 2016 Tutorial: Generative Adversarial Networks, NIPS (2016).

Radford, A., Metz, L. and Chintala, S., Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434. (2015).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. Improved techniques for training gans. NIPS (2016).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

Zhao, Junbo, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016).

Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).

Liu, Ming-Yu, and Oncel Tuzel. Coupled generative adversarial networks. NIPS (2016).

Denton, E.L., Chintala, S. and Fergus, R., 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. NIPS (2015)

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., & Courville, A. Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016).

## Applications:

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004. (2016).

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. Generative adversarial text to image synthesis. JMLR (2016).

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). Face Aging With Conditional Generative Adversarial Networks. arXiv preprint arXiv:1702.01983.

# Exercise on object oriented GANs

# Walk through of Exercise 5: cVAEs and GANs

1. **Introduce the GAN code:**
   1. **Walk through the structure of gan.py at …/tasks/VAEGAN/vaegan**

   2. **Walk through today's caller program:**
      **train_gan_mnist.ipynb**

# Overview of the train_gan notebook

- **Given a complete notebook to run a GAN (train_gan_mnist.ipynb)**
- **Given a *partially* complete GAN  (gan.py)**
- **Your job:**
  - **Understand the calling code in the notebook  (.ipynb)**
  - **Understand the partial gan.py**
  - **We will review together what is missing and you will work on it together.**
- **Once you are done, test your GAN.**
- **Write down what you observe.**

# Structure of the GAN code file: gan.py

**Standard GAN**

- **Generator module**
  - **class Generator(tf.keras.Model)**

- **Discriminator module**
  - **class Discriminator(ft.keras.Model)**

- **GAN model**
  - **Contains one Generator and one Discriminator**

# Structure of the Generator module

- **Class Generator(tf.keras.Model)**

- **Methods**
  - **__init__()    … define the layers that you wish to use in the forward pass**
  - **call()   …. Define the forward pass .**
  - **get_config() and from_config() … helper methods that allow loading and saving trained models.**

# Structure of the Discriminator module

- **Class Discriminator(tf.keras.Model)**

- **Methods**
  - **__init__()    … define the layers that you wish to use in the forward pass**
  - **call()   …. Define the forward pass .**
  - **get_config() and from_config() … helper methods that allow loading and saving trained models.**

# Some programming tips

- **self.attribute … self refers to the current instance of the class.**

- **How forward pass is expressed:**
  - **x=inputs**
  - **x = self.layerName1(x)**
  - **x = self.layerName2(x)**

**Now x contains the inputs transformed by the two layers.**

## Some programming tips

**What are some helpful functions to help you understand what to do next?**

**variable.shape is your friend**

`Z.shape returns the dimensions of the variable, Z`

**whos variable is also your friend**

```
>> %whos
  Name          Size                Bytes  Class     Attributes
  Z            300x2                 4800  double
```

## Think about these questions:

- **You may observe something like this during training. It is to be expected. What happens with longer training?**

- **What do you observe about the appearance of the strokes of the digits?**
- **In the training curves plot, what is happening to the generator and discriminator losses?  What does that signify? Practical implication?**
- **What pros and cons of GAN final results do you observer over VAE (and cVAE) final results?**

# Acknowledgements

Albert Montillo, PhD, PI

Son Nguyen, PhD
Postdoc

Alex Treacher
PhD student

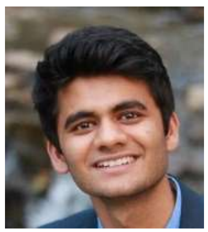
Aixa Andrade Hernandez, MS, PhD student

Austin Marckx
PhD student

Krishna Chitta
Res. Sci.

------------- Recent Alumni -------------

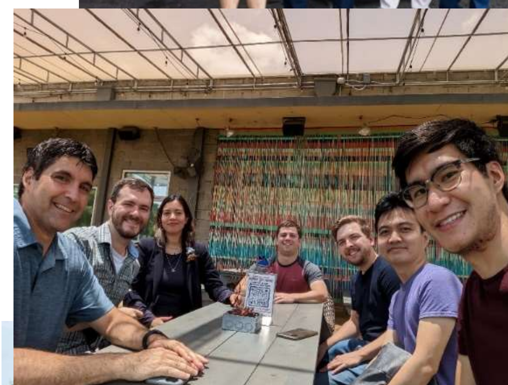
Atef Ali
Undergrad



Vyom Raval, BS
MD/PhD

Kevin Nguyen
MD/PhD student

Cooper Mellema
MD/PhD student




## Lab Funding

- **NIH/ NIGMS R01** ***Correcting Biases in Deep Learning***
- **King Foundation (PI) : Quantitative AD diagnostics.**
- **Lyda Hill Foundation (PI): Quantitative prognostics of Parkinson's disease**
- **NIH/ NIA R01 Blood Biomarkers for Alzheimer's and Parkinson's**
- **TARCC : Texas Alzheimer's Research and Care Consortium.**
- **NIH / NINDS F31 fellowship : Causal connectivity biomarkers for neurological disorders**

# Thank you!

**Email: Albert.Montillo@UTSouthwestern.edu**

**Github**: https://github.com/DeepLearningForPrecisionHealthLab

MegNET …. Artifact suppression
BLENDS …. fMRI augmentation
Antidepressant-Reward-fMRI …. response prediction
Parkinson-Severity-rsfMRI … disease trajectory prediction