

Software Engineering: OOP illustrated through Density Estimating Neural Networks

Albert Montillo
UTSouthwestern

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



UNIVERSITY of PENNSYLVANIA



Rensselaer

Yale University

RUTGERS
UNIVERSITY



GE Global Research

MGH/HST Athinoula A. Martinos
Center for Biomedical Imaging



MASSACHUSETTS
GENERAL HOSPITAL



Harvard-MIT
Health Sciences & Technology



PENN
Radiology
University of Pennsylvania Health System

Microsoft
Research
COGNEX

SWE course

Lyda Hill Department of Bioinformatics

Outline

1. Monday : Object Oriented Variational Autoencoders (VAEs)

1. Self study read the slides (Monday_VAE..pptx) on your own
2. Start the exercise described on slides 47-54
3. Any concerns: email TAs and/or bring questions to Tuesday morning lecture

2. Tuesday

1. Ask your residual questions about VAEs after having attempted exercise Monday
2. New topic: Object Oriented Generative Adversarial Networks (GANs)
3. New topic: Symbolic debugger: cond breakpoints and call stack traversal

3. Wednesday

1. Review observations on VAEs and GANs
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

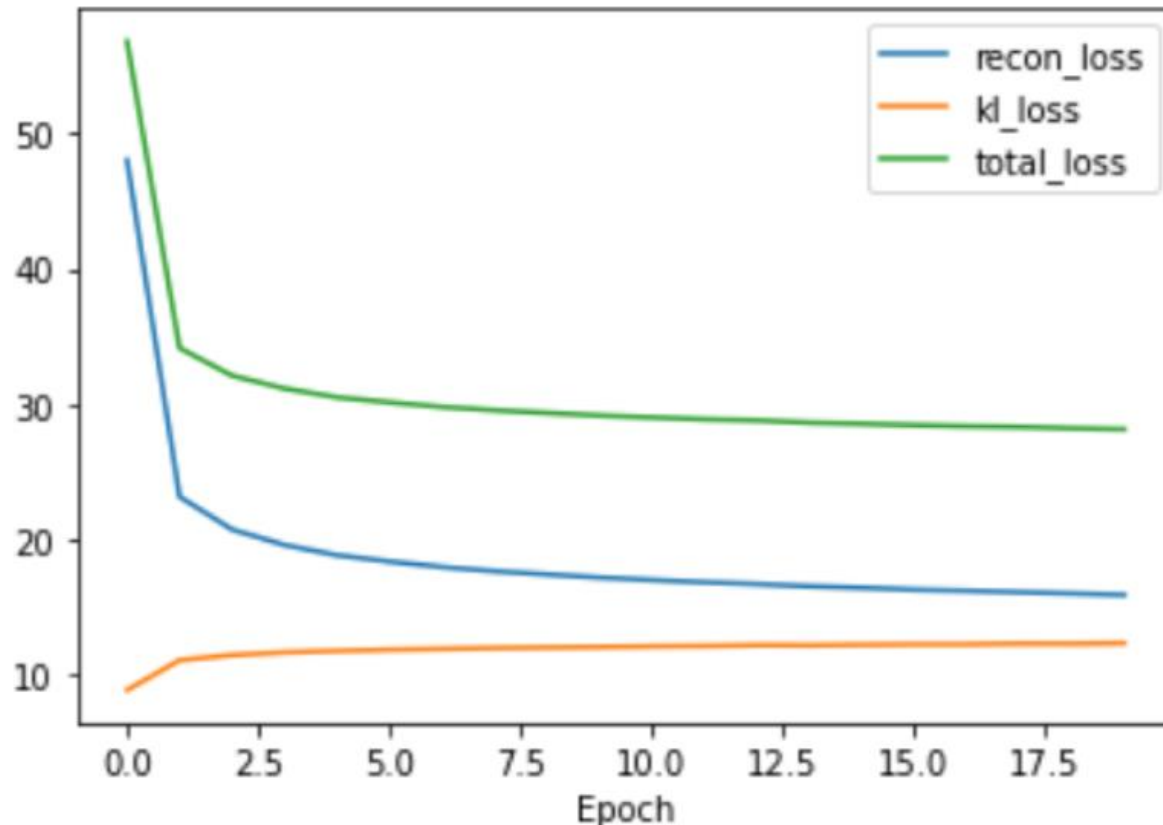


Observations from the Hands on VAE exercise

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



VAE Reconstructed images at epoch 10



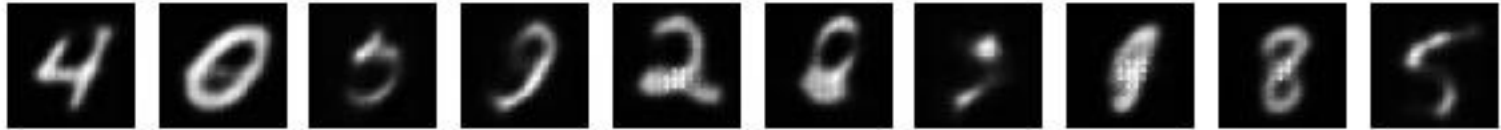
VAE Reconstructed images at epoch 20



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**



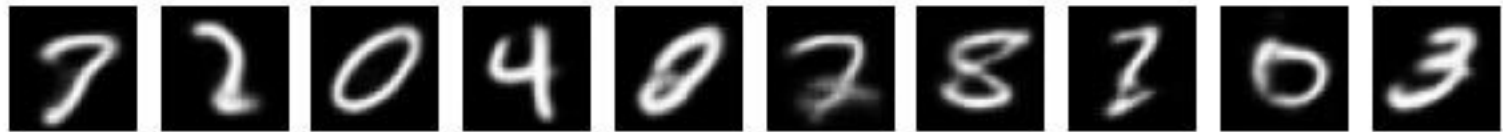
VAE purely synthesized images at epoch 10

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



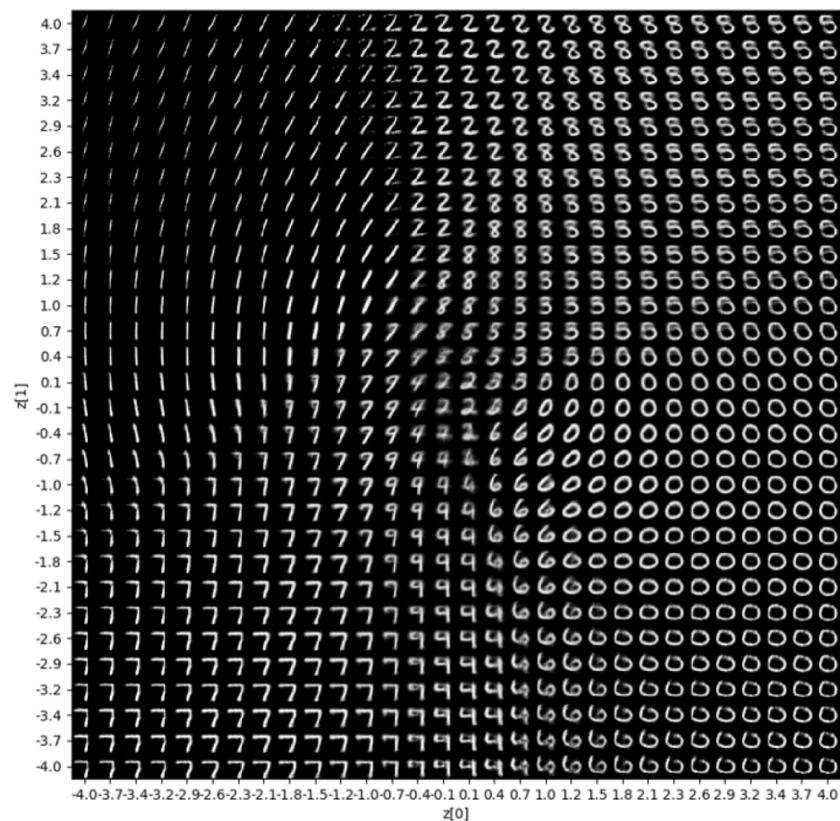
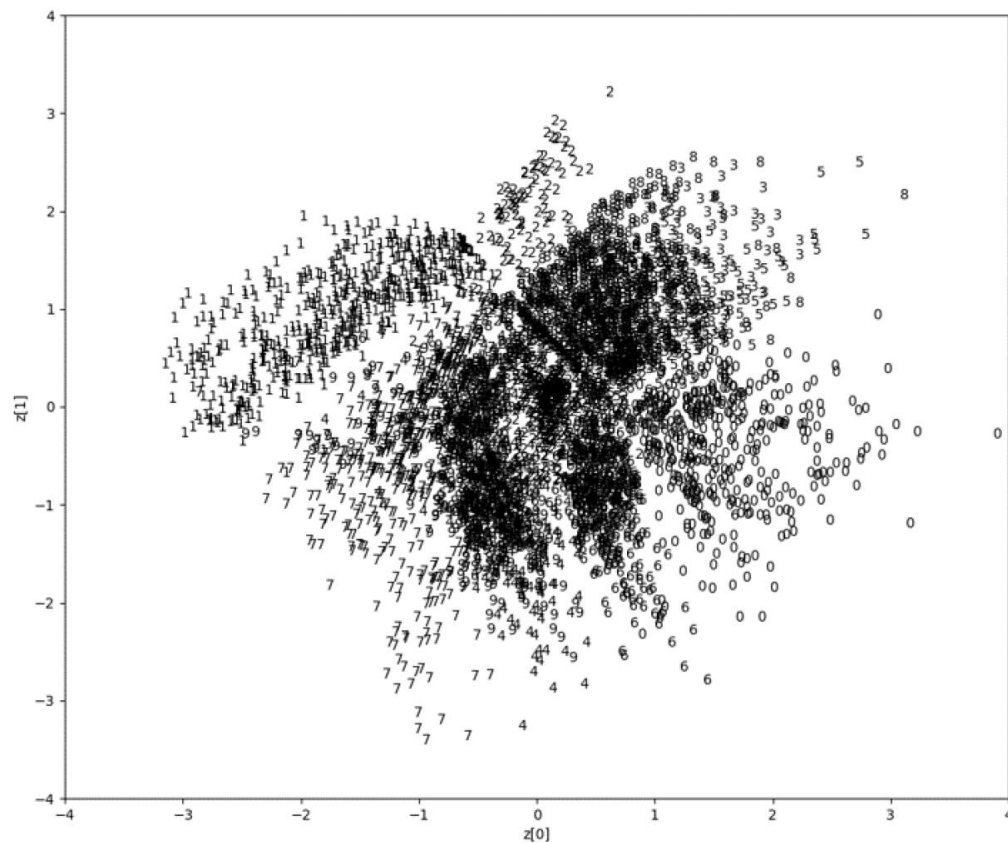
VAE purely synthesized images at epoch 20

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 ..

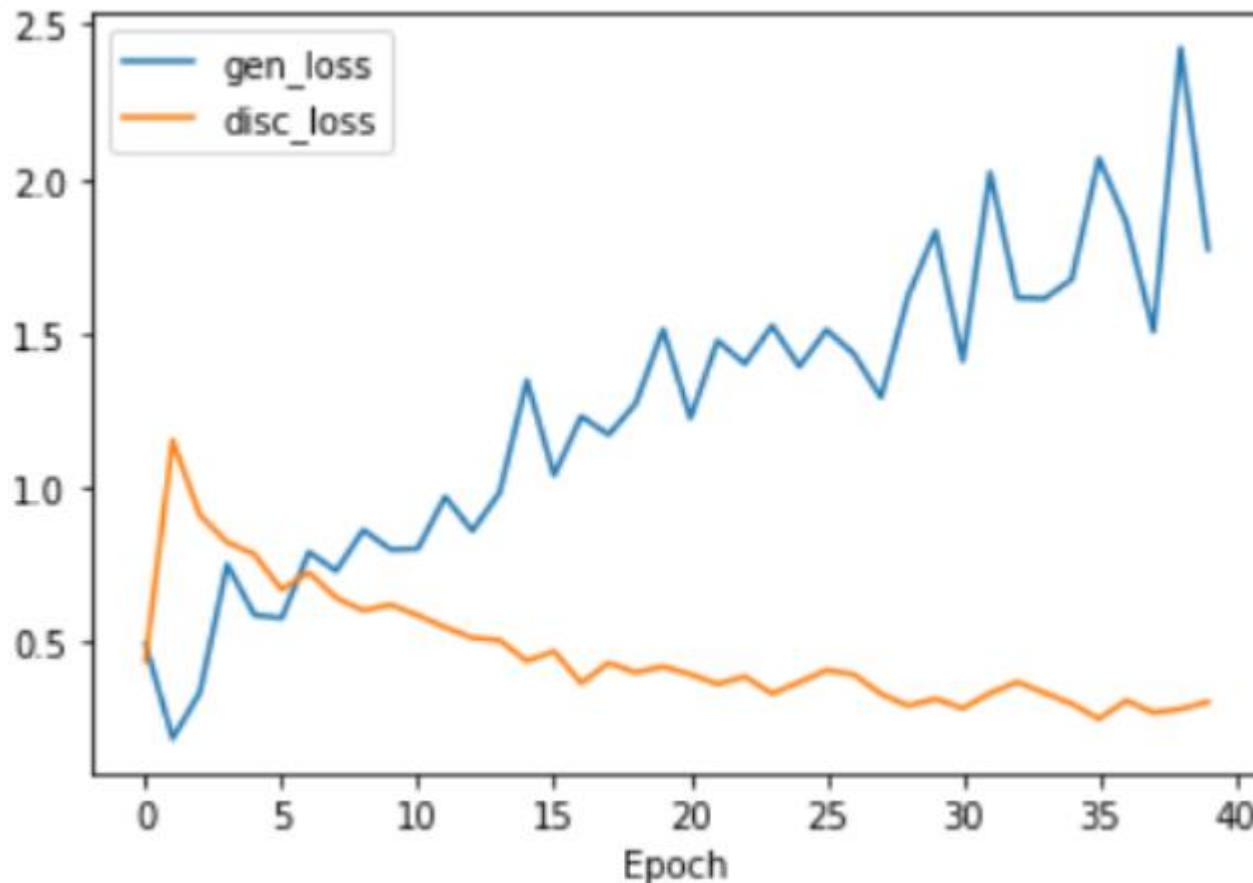


Observations from the Hands on exercise: GANs

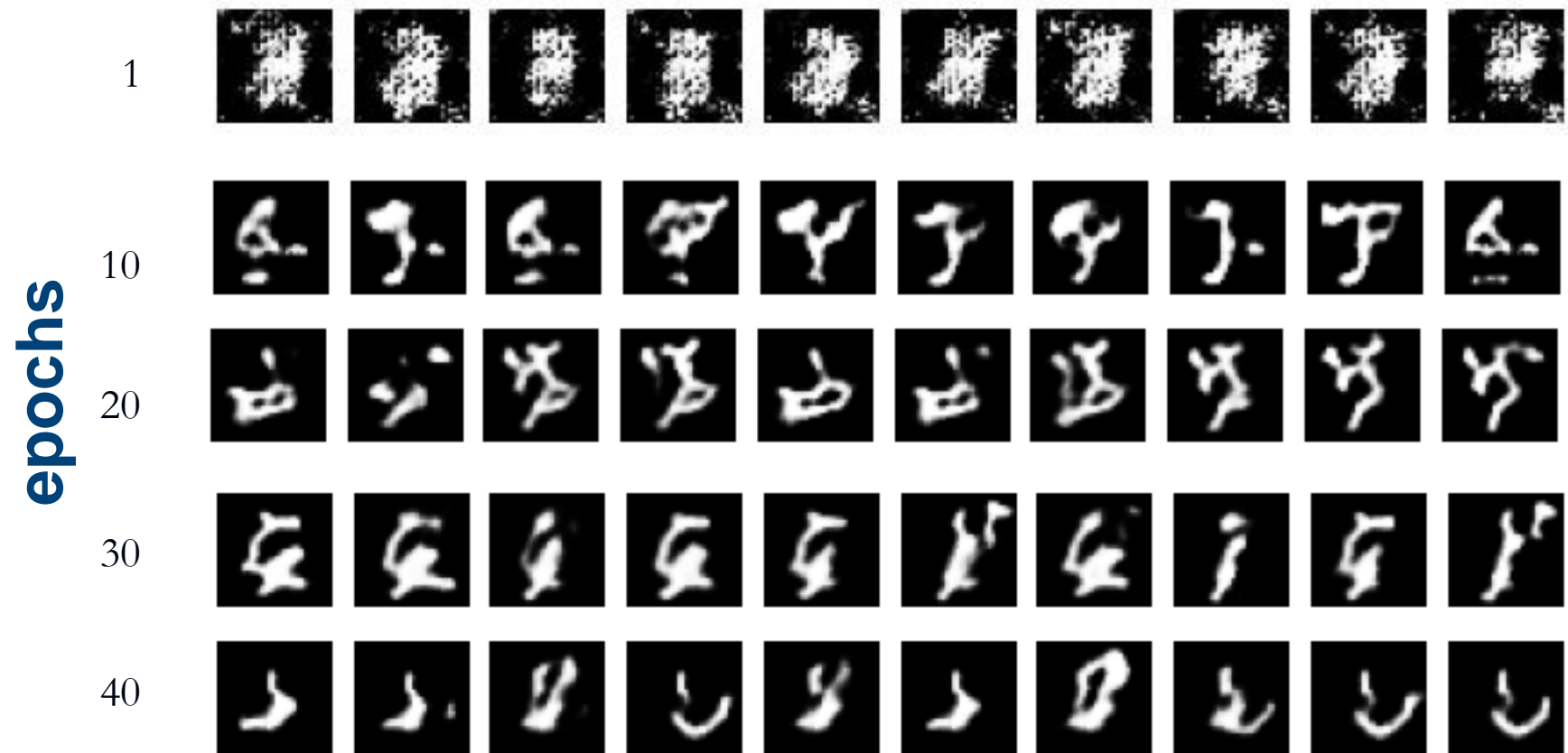
12

GAN Training curves

1. Shows evolution of generator and discriminator losses
2. We observe some divergence.. What would we do next? Later we will address



GAN purely synthesized images at epochs 1 ... 40



1. **Observations: digits are starting to appear, slowly**
2. **Clearly better at epoch 40 than 1, but more work needed.**
3. **Upshot: it can be challenging to get a simple GAN to converge**

New topic: Limitations of the VAE

- 1. The VAE condenses latent Z space, making no gaps such as dead zones of say representative vectors (e.g. non digits)**
- 2. While this is an improvement over AEs, it does NOT permit generation of a specific label of data on demand. You can sample latent space but have no idea what class of vector you will get (e.g., which digit)**
- 3. This causes two problems**
 - 1. it makes introspection of the learned latent space (density) more difficult**
 - 2. It reduces the usability of the decoder to generate specific labels (classes) of data (e.g., a specific digit cannot be guaranteed)**
- 4. Conditional VAE (cVAE) will remedy those limitations**

Changes to VAE to make it conditional. The cVAE

1. **Key idea: concatenate the label to the input vector and encode/decode the now expanded input vector**
2. **Concatenate the class label when training the encoder**
For images, the labels are concatenated as a per-pixel, one-hot encoded class label. That is every pixel is labeled individually.
3. **Concatenate the class label when training the decoder**
The latent z vector representation is concatenated with the one-hot encoded class label.

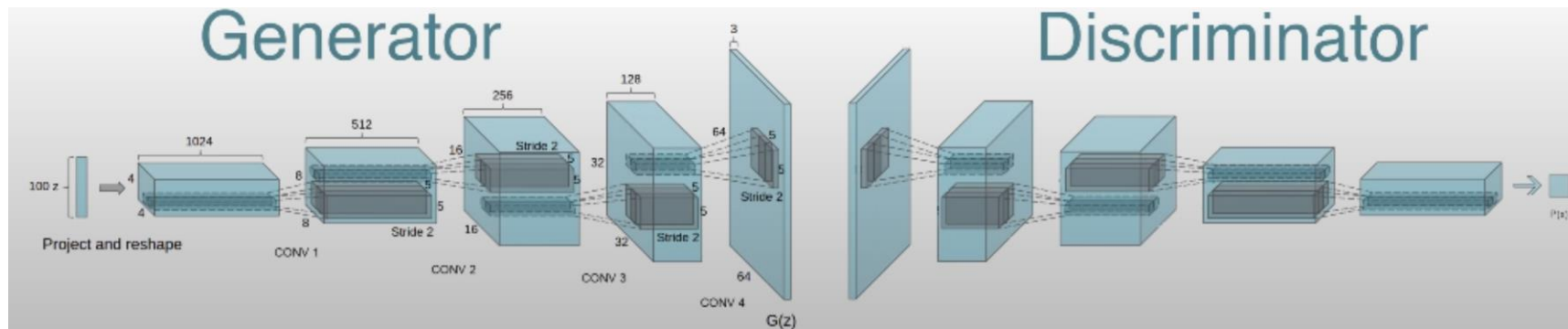
New topic: conditional auxiliary classifier GAN

The GAN has limitations that we have just seen.

In a largely analogous way that the cVAE can alleviate limitations of the VAE, so too can a conditional auxiliary classifier GAN alleviate key limitations of the GAN.

Recall: Deep convolutional GANs (DCGANs)

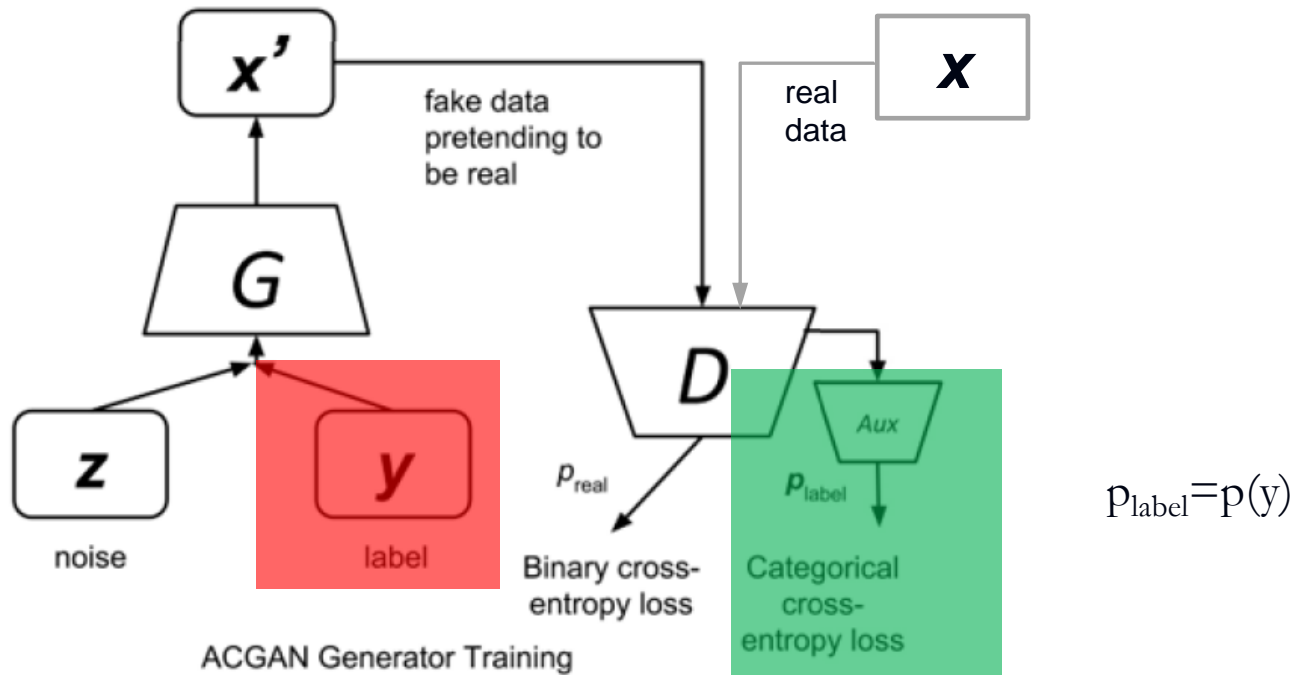
Architecture



Key ideas:

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

Extension: conditional Auxiliary Classifier GAN



- Generator gets an additional input: class label and it not only tries to fool the Discriminator but also aims to generate data that matches the given class label.
- Discriminator (now multitasking) still aims to distinguish which inputs it sees are real and which are fake but is also aims to classify each image into a class label (it outputs a predicted class label).

Why use cGANs for Generation?

All the benefits of GANs:

- Can be trained using back-propagation for Neural Network based Generator/Discriminator functions.
- Images can be generated with sharp (non blurry) edges.
- Fast to sample from the model distribution: *single* forward pass generates a *single* sample

Plus, we attain the following new capabilities:

- Generator is now steerable: generate new data for a specified class
- Discriminator can be used for decision making

cGANs 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., Houthoofd, 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 Onel 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 5

Part 1: implement cVAE

Part 2: experiment with cGAN

Walk through of Exercise cVAEs and cGANs

Introduce the code:

1. Point out where the cVAE is to be implemented in vae.py
class ConditionalVAE(VAE):
2. Point out the provided cGAN in gan.py
Play with this fully implemented code.
class ConditionalGAN(GAN):
3. Walk through vaegan\conditional\callbacks.py
4. Walk through today's caller programs:
 1. train_cvae_mnist.ipynb
 2. train_cgan_mnist.ipynb
5. Tomorrow: we will provide a fully implemented cVAE-cGAN to experiment with.

Walk through of the cVAE Exercise

1. Introduce the cVAE code:

1. Walk through the structure of /code
 1. At the level of /code are the *caller programs* (jupyter notebooks)
2. Walk through today's caller program: `code/train_cvae_mnist.ipynb`
 1. It uses MNIST data as the training set.
 2. Purpose is to apply the cVAE class you complete to estimate the density of MNIST images and be able to reconstruct new digit images **that you can specify**.
 3. It outputs reconstructed results throughout and at the end of training here:
`code/outputs/mnist_cvae`
Folder contains reconstructed real digits and synthesized digits (next slides)
 4. Note: `HINTS_SWE.ipynb` contains helpful hints for this course.
3. Class hierarchy (illustrating inheritance) is in `code/vaegan`
 1. Walk through `code/vaegan/vae.py`
 1. Overall structure, use of code folding (Ctrl+K,J and Ctrl+K,1 or 2)
 2. Use VSCode to edit .py use Firefox to edit Jupyter

... Walk through of the VAE Exercise

Walk through exercise in Syllabus

Complete the Exercise 5 “ToImplement” occurrences in vae.py
Show this in VSCode..

- 1. ToImplement Exercise5a**
def make_conditional_input(self, images, labels):
- 2. ToImplement Exercise5b**
def train_step(self, data):

Note: /code/AD contains caller programs for applying what we build to AD

Overview of the train_cvae notebook

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

Structure of the **provided** cGAN in the file: gan.py

Standard GAN

- **Generator module**
 - `class Generator(tf.keras.Model)`
- **Discriminator module**
 - `class Discriminator(tf.keras.Model)`
- **GAN model**
 - Contains one Generator and one Discriminator

Advanced: Auxillary Classifier GAN

- **MultitaskDiscriminator module**
- **conditonalGAN model ... an auxillary classifier GAN.**
 - Generates images and predicts their class label
- Run a `git pull` command to retrieve an updated gan.py file that includes now for you the cGAN fully implemented to study and then run. **Write down/snap shot of what you observe.**

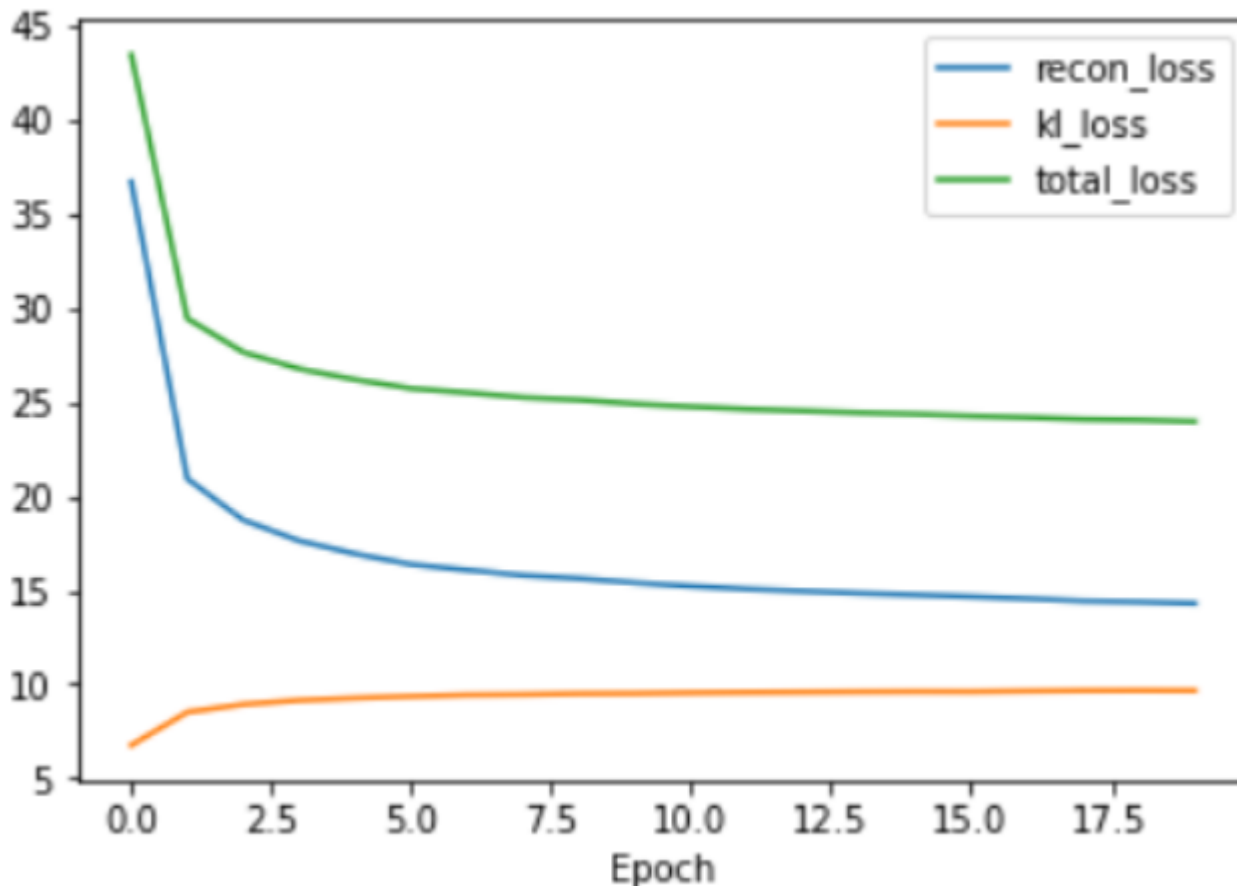
Hands on exercise

- 1. Finish implementing cVAE
subclass your VAE
implementation and
create a cVAE**
- 2. Run the provided cGAN.**

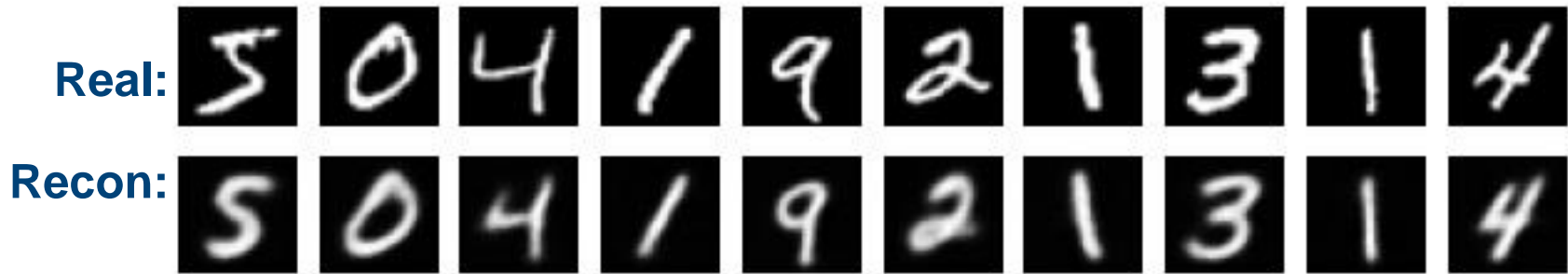
34

Conditional VAE (cVAE) Training curves

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



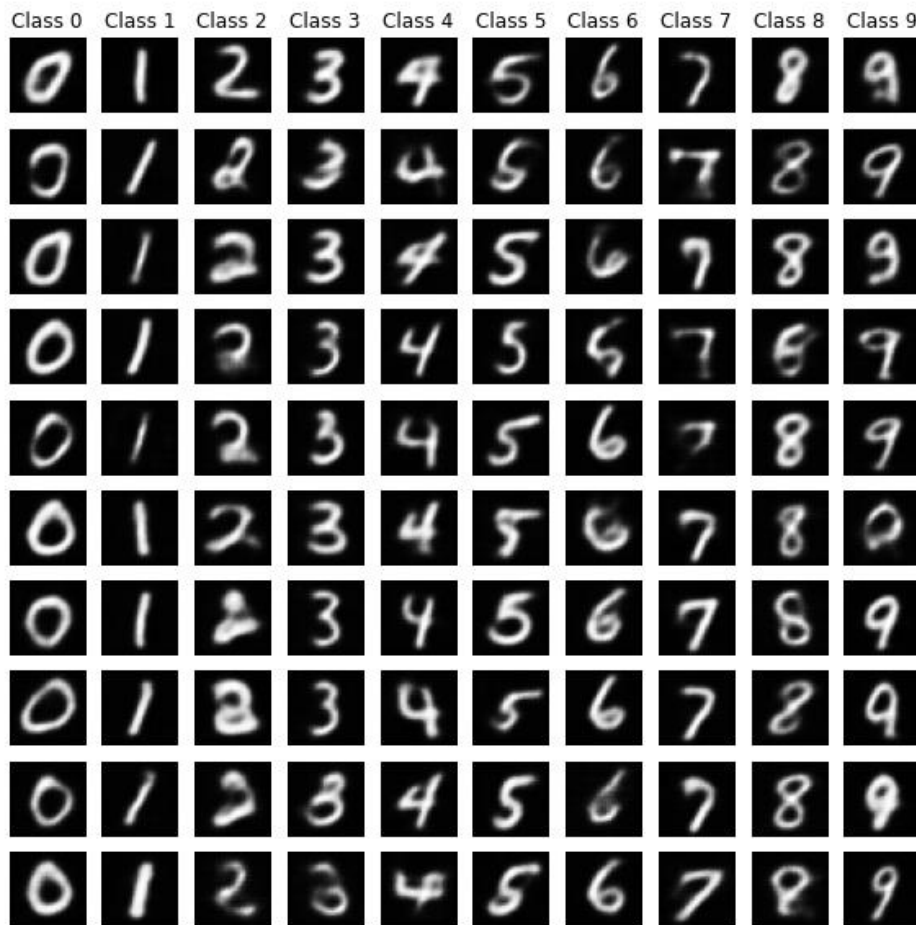
Conditional VAE Reconstructed images at epoch 1



What do you observe as the # epochs increases?

Conditional VAE Purely synthesized images at epoch 1

Note: Steerable class label (column), 100 different random z's



How do the results change as you train for more and more epochs?

Think about these questions:

For both the cVAE and cGAN

- When you train longer, are better and better digits produced? Is there a limit?
- What do you observe about the appearance of the strokes of the digits?

For the cGAN:

- In the training curves plot, what is happening to the generator and discriminator losses? What does that signify? Practical implication?

Overall

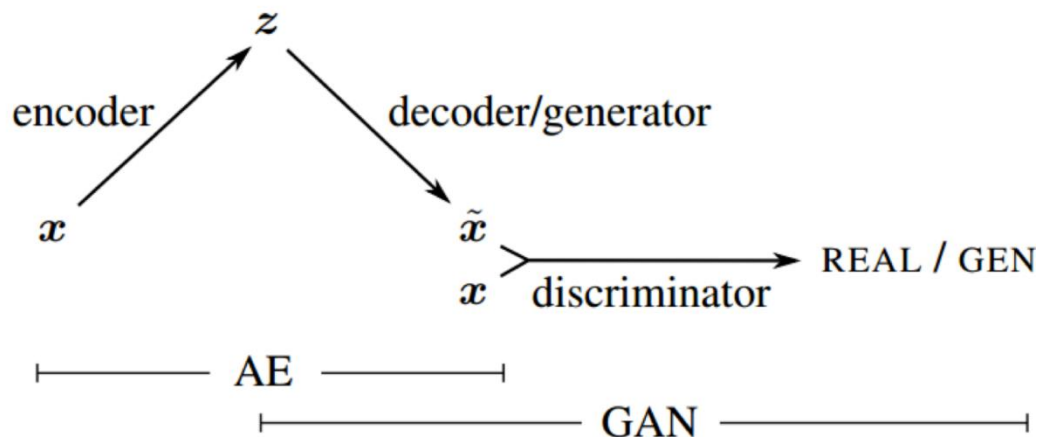
- What practical value does *conditional* VAE add over VAE?
- What practical value does *conditional aux. classifier* GAN add over GAN and over cVAE?
- Qualitatively compare your generated images across the models you have learned in this week: What pros and cons of the results do you see from VAE, GAN, cVAE, and cGAN?

Combining VAEs and GANs: The cVAE~cACGAN

- **Motivation:**
 - Want to generate images that have sharp edges of the GANs as this is a property of convincing real-world images.
 - Also want to smoothly traverse the learned space of images to understand better our data which is a property of cVAEs
- **How can we get the best of both worlds?**
- **Observation 1: Loss functions are different**
 - The pixel-wise loss from the VAE is to blame for the blurry images. Not much penalty for blurry edges. Does not encourage crisp, sharp pen strokes.
 - The GAN has a semantic loss from the extraction of semantic features through the layers of the Discriminator. This semantic loss is based on convolution filters which at the lowest level are edge detectors. They encourage sharp edges. This is the reason for the sharper (less blurry) images from the GAN.


cVAE~cACGAN

- **Observation 2: Compatible/duplicate parts**
 - Decoder of a VAE performs largely the same job as the generator of the GAN
 - They both take a compact, latent z and construct an image



VAE-GAN architecture, the discriminator from GAN takes input from VAE's decoder

- **Observation 3: value of linkage**
 - The encoder of the VAE allows us to find the location in the compressed latent space of any (real) input image, this also helps us understand the learned embedding.
 - Backpropagating the discriminator loss and pixel-wise recon loss back to the encoder/decoder allows to retain sharpness and high-quality images whose embedding space we can traverse and explore.



**Tomorrow we will provide a fully
implemented cVAE-cGAN for
MNIST and for Alzheimer's
dataset**

41

Request (optional) Run cVAE-cGAN on Alzheimer's dataset

42

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



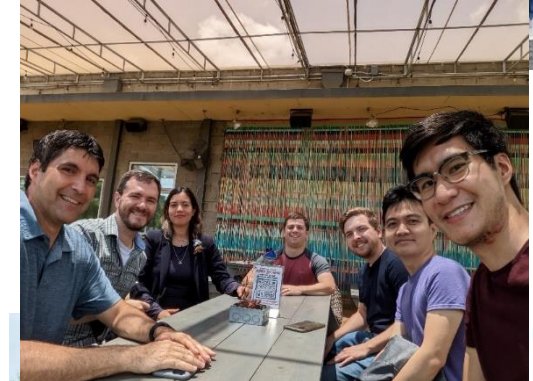
Vyom Raval, BS
MD/PhD



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



End of presentation
