Software Engineering: OOP illustrated through Density Estimating Neural Networks

Albert Montillo
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

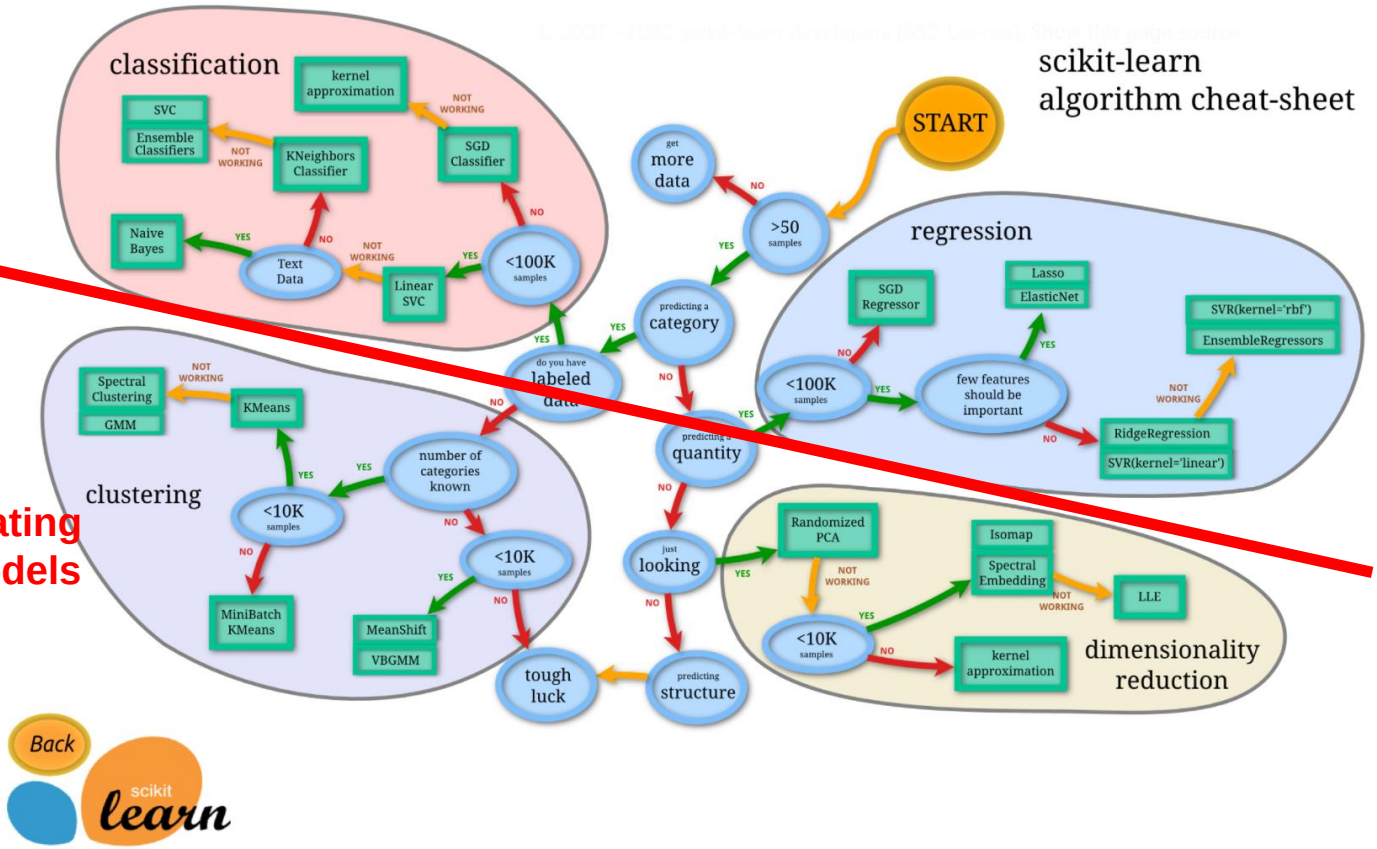
Observations from the weekend exercises

How much information is available for machine learning?

- **Pure supervised learning (cherry)**
- **Semi-supervised learning (icing)**
- **Unsupervised learning (cake)**

Taxonomy of machine learning algorithms

- **Supervised algorithms (labeled)**
- **Unsupervised (unlabeled)**
  - **density estimating**
  - **generative models**



Taxonomy of Generative Models

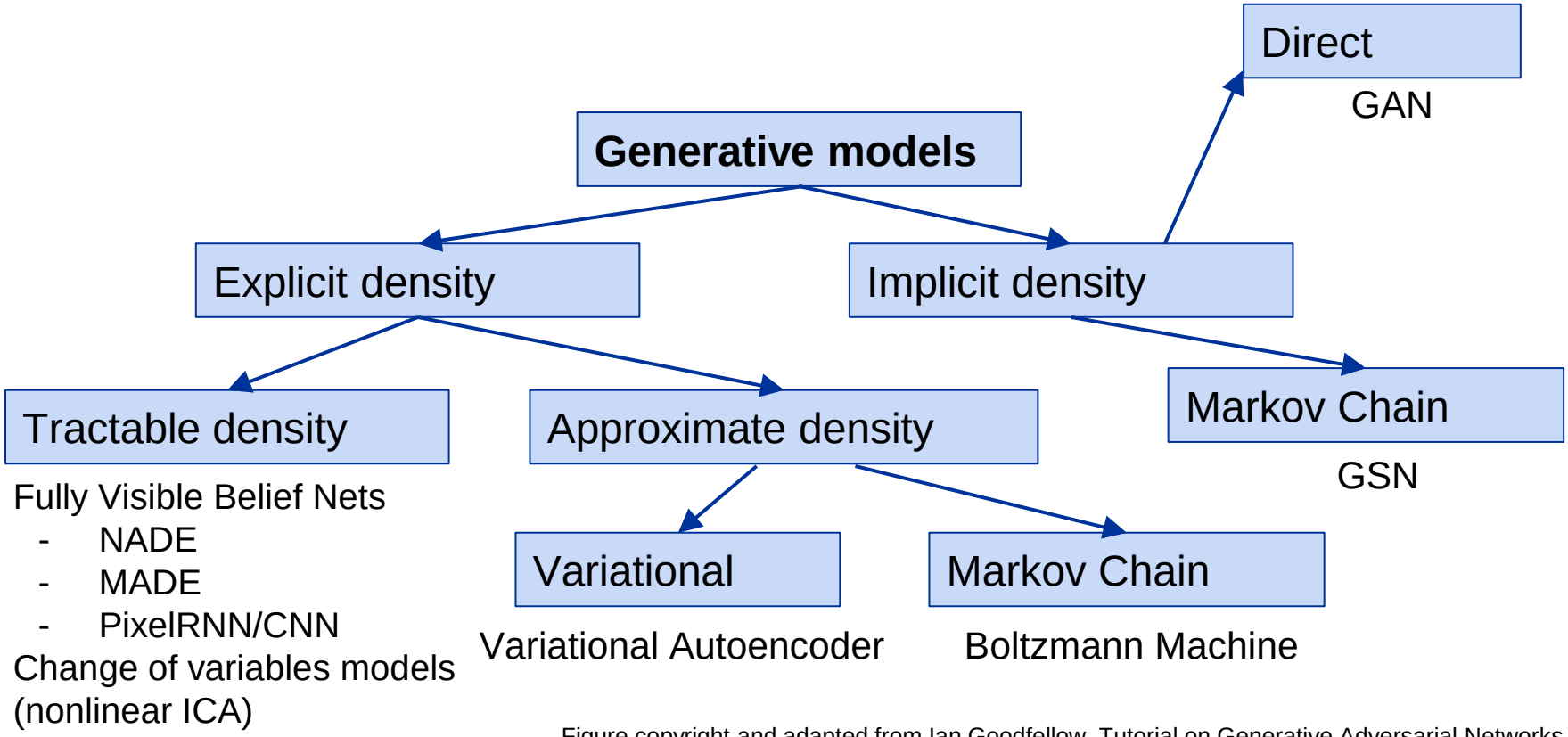


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Taxonomy of Generative Models

We will discuss 2 of the most popular types of generative models

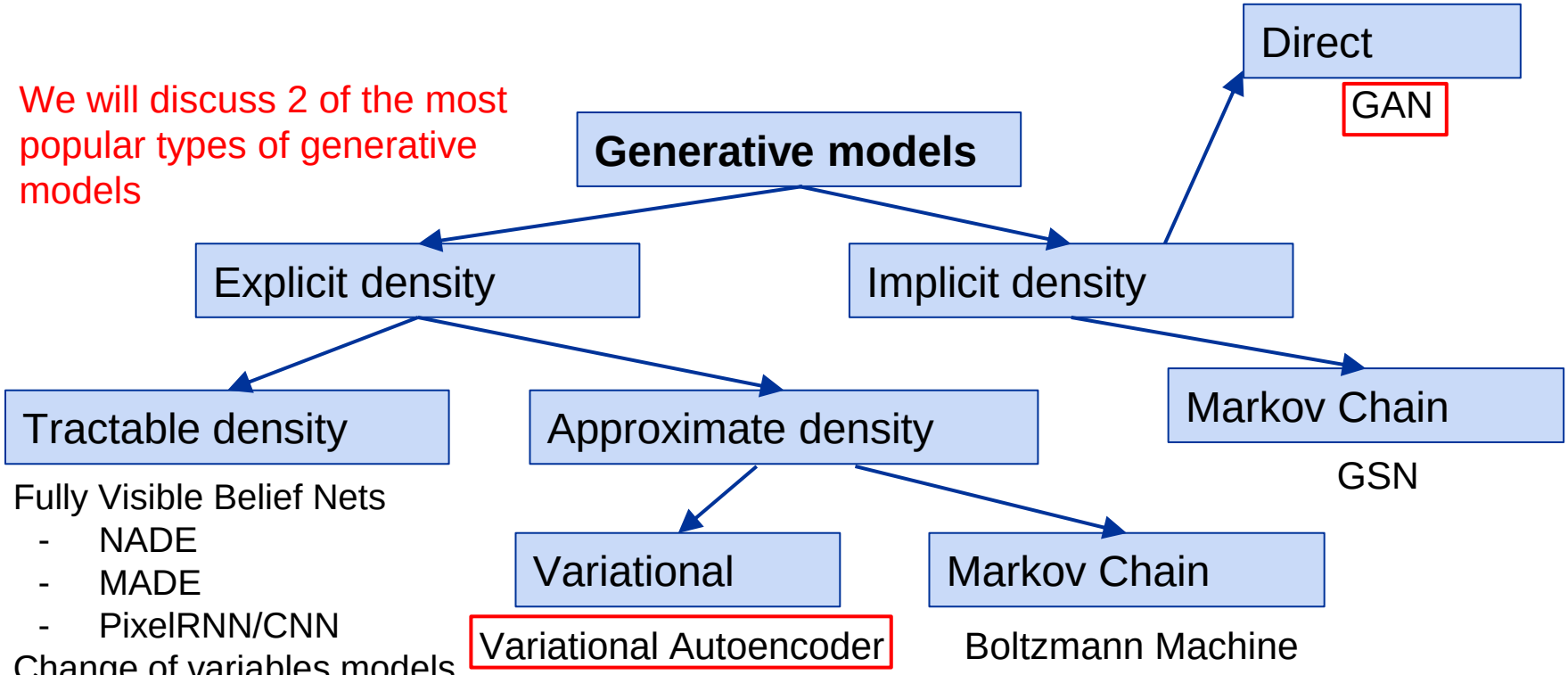
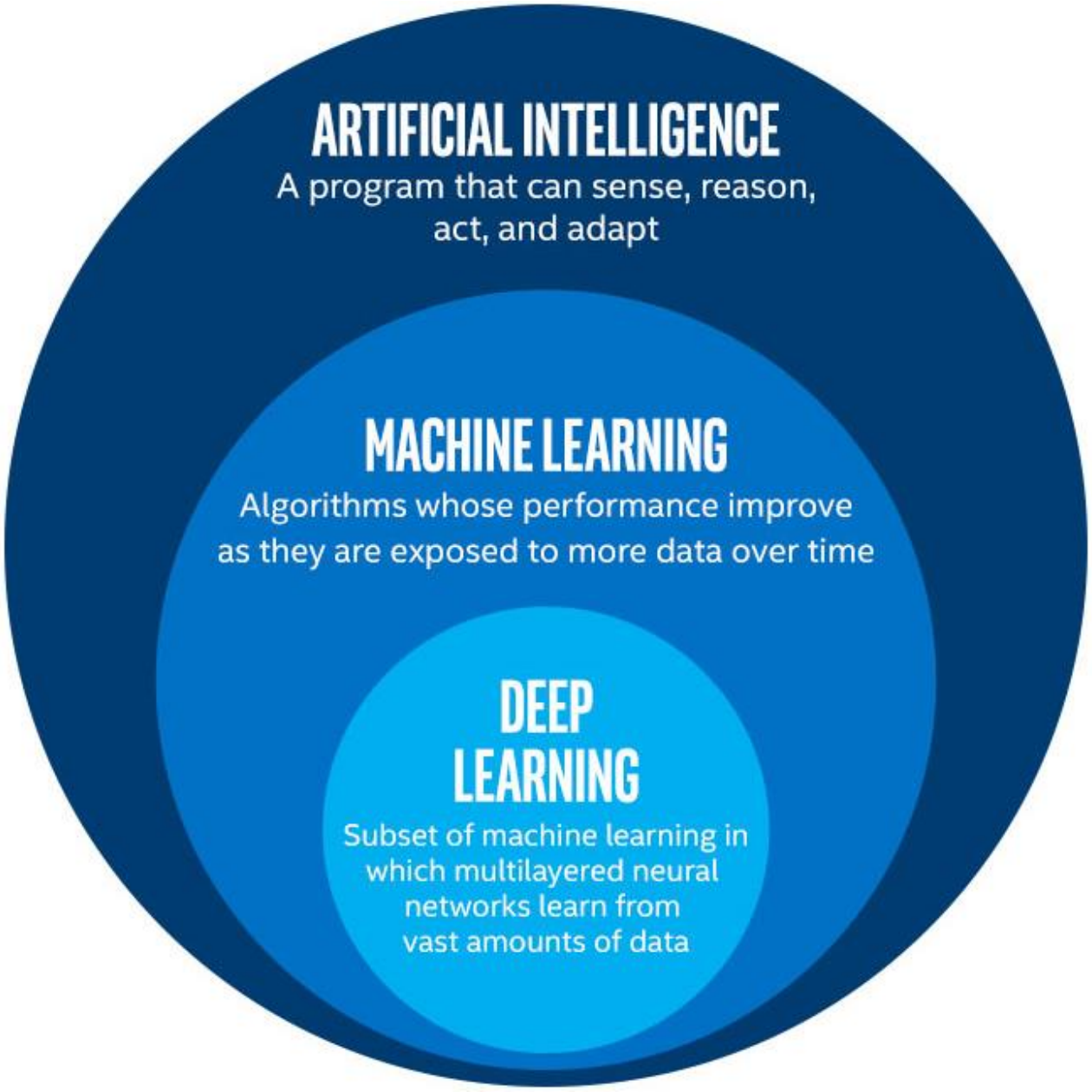


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

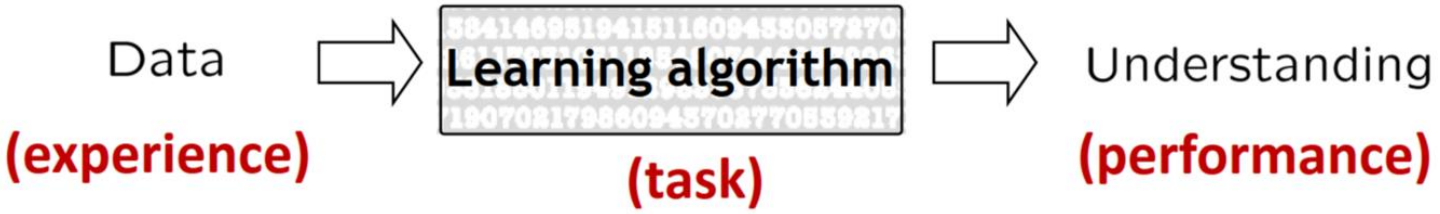
How do GANs and VAEs relate to AI?



Core attribute of machine learning algorithms

Algorithms whose

- **performance improves**
- **at some task**
- **with increasing experience**



Datasets

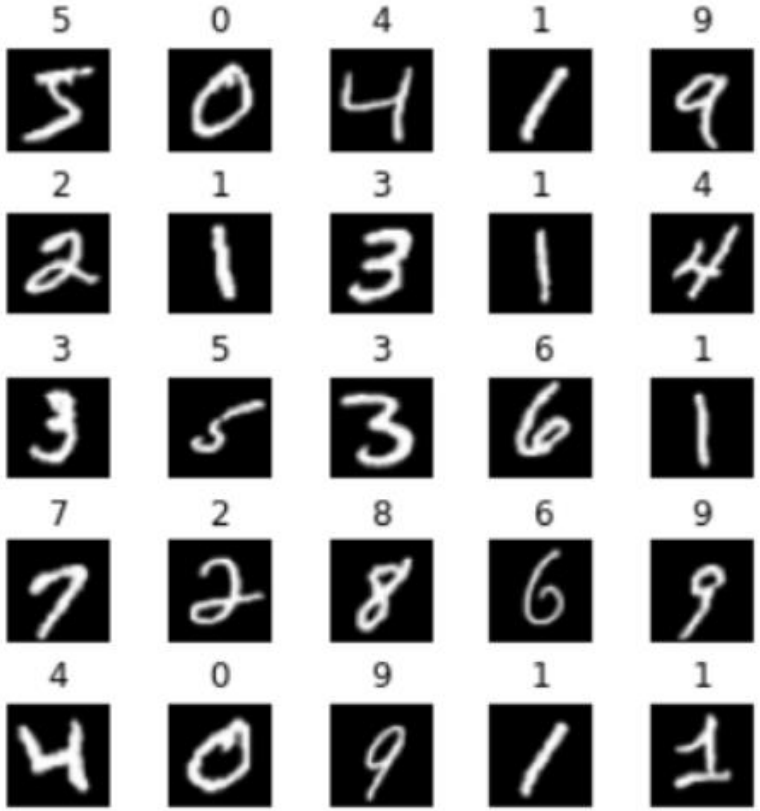
1. **MNIST dataset**

2. **Alzheimer's dataset**

MNIST

1. **60,000 images of handwritten digits**

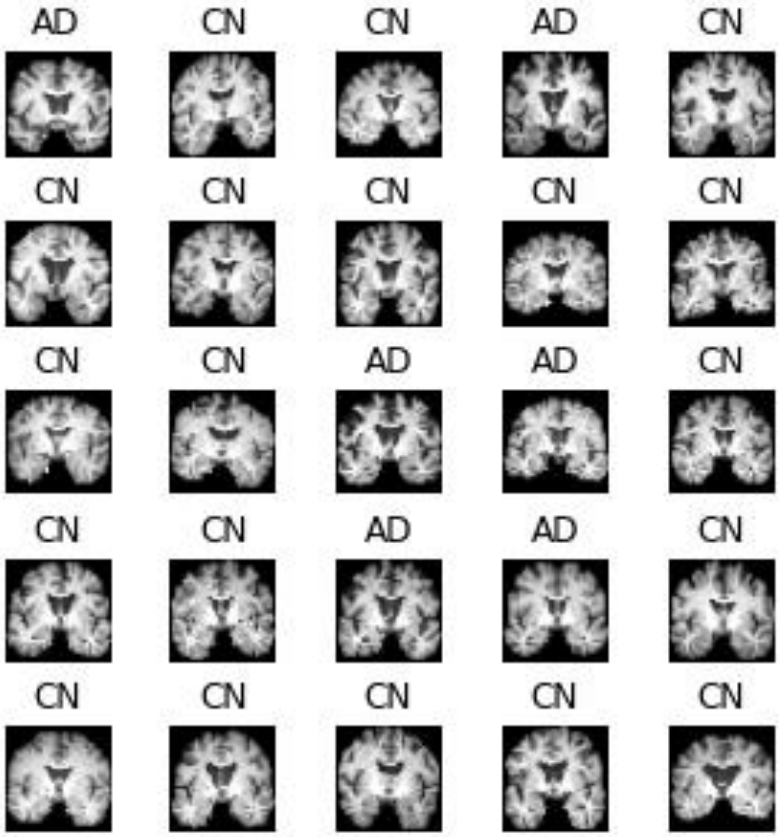
2. **Example images**



Alzheimer's Disease

1. **T1w MRI of cognitively normal and Alzheimer's Disease patients**

2. **Example images**



Evolution of autoencoders

1. **Autoencoders**

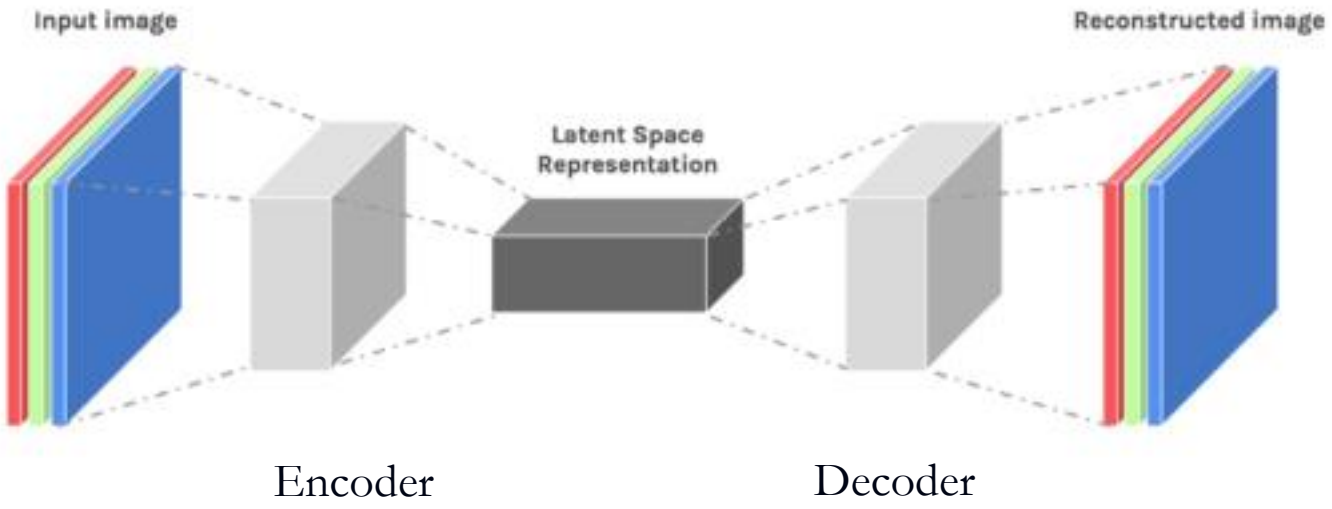
2. **Variational autoencoders**

Autoencoders (AEs)

Autoencoders are designed to reproduce their input through training

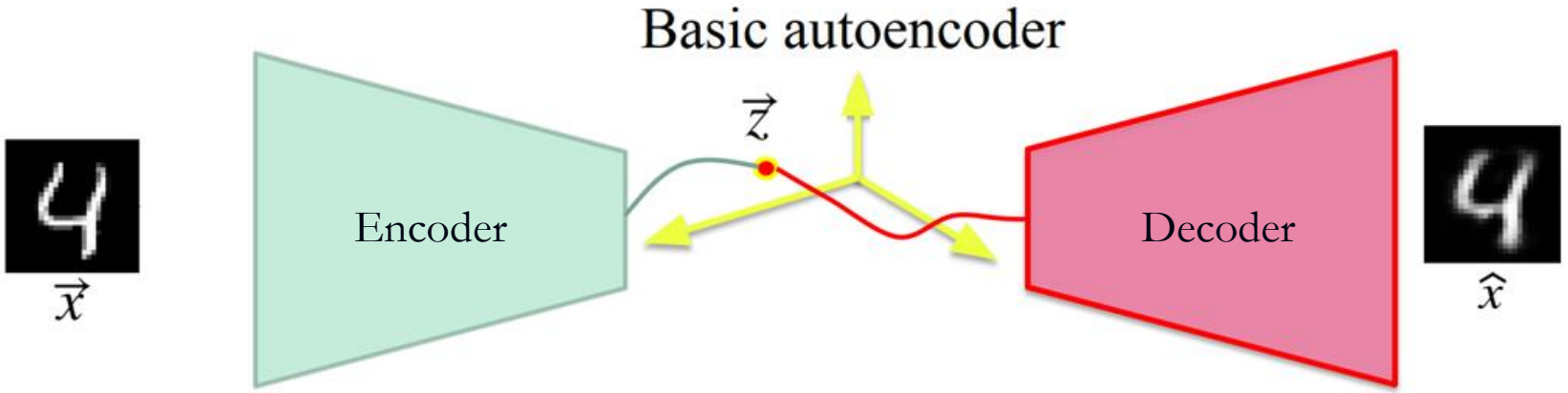
- **Can be used for any data, but especially useful for images.**
- **Why do that? You don't have labels (of the images)**
- **1) learn to denoise your images**
- **2) learn a compact representation**

Composed of Encoder and Decoder subnetworks:



https://www.edureka.co/blog/autoencoders-tutorial/

Comparison AEs and Variational autoencoders (VAEs)

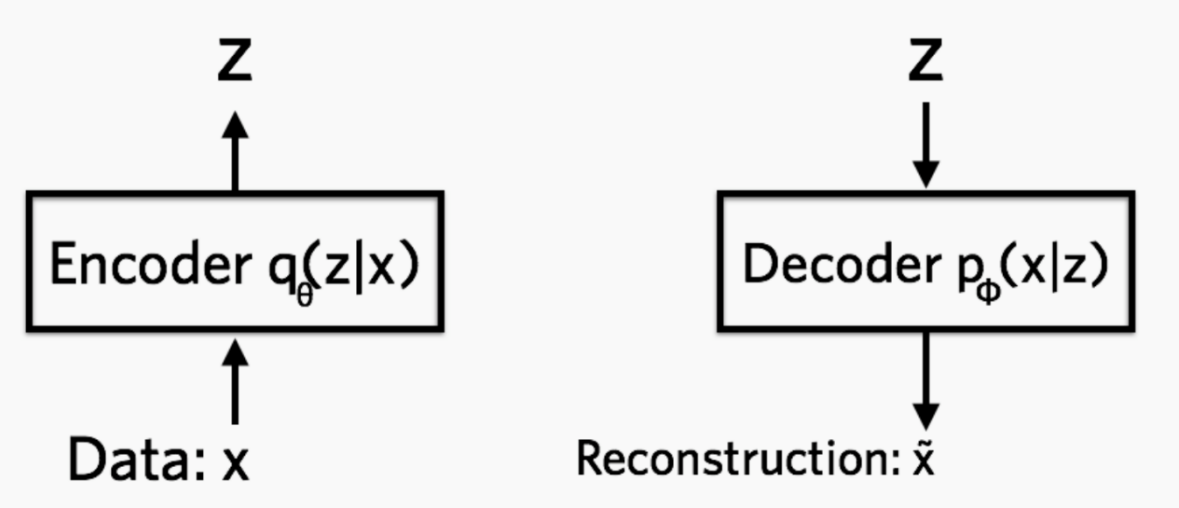


Jodoin, Deep learning for medical imaging school 2021

Variational Autoencoder (VAE)

Key idea: make both the encoder and the decoder probabilistic.

I.e., the latent variables, z, are drawn from a probability distribution depending on the input, X, and the reconstruction is chosen probabilistically from z.



https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

VAE Encoder

The encoder takes input and returns parameters for a probability density (e.g., Gaussian): I.e., $q_\theta(z \mid x)$ mean and covariance matrix.

We can draw from this distribution to get values of the representation of *z* in the lower-dimensional space.

Implemented via a neural network: each input *x* gives a vector mean and diagonal covariance matrix that determine the Gaussian density

Parameters θ **for the encoder network need to be learned – need to set up a loss function.**

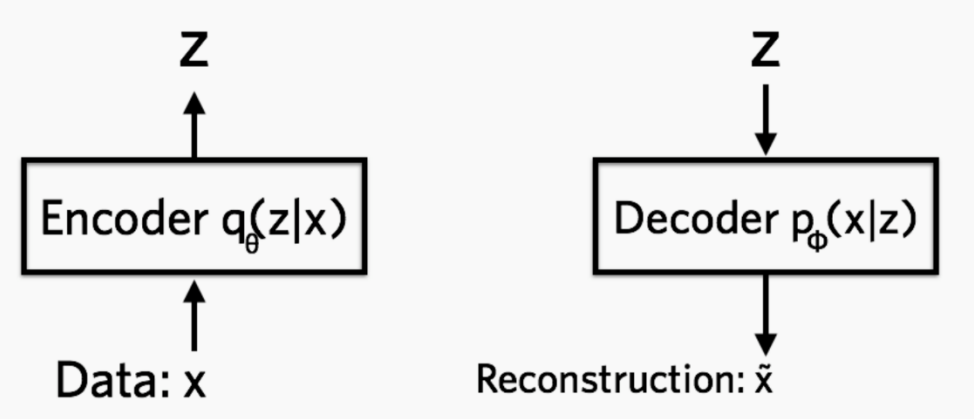
https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

VAE Decoder

The *decoder* takes latent variable *z* and returns parameters for a distribution. E.g., $p_\phi(x|z)$ gives the mean and variance for *each pixel* in the output.

Reconstruction $\tilde{x}$ is produced by sampling.

Implemented via neural network, i.e. the parameters ϕ of the decoder network are also learned.



VAE loss function

What is the loss function for autoencoder? L_2 distance between output (reconstruction) and input (original).

For VAE, we need to learn parameters of two probability distributions. For a single input, x_i, we maximize the expected value of returning x_i or equivalently, minimize the expected negative log likelihood.

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

This takes expected value wrt *z* over the current distribution $q_\theta(z|x_i)$ of the loss $-\log p_\phi(x_i|z)$

This loss term ensures that combination of the probabilistic encoder/decoder produce a reconstruction similar to the input.

https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

VAE loss function

Problem: the weights may adjust to memorize input images via *z*. I.e., inputs, that we regard as similar, may end up with very different distributions in *z* space.

We prefer continuous latent representations to give meaningful parameterizations. E.g., smooth changes from one digit to another.

Solution: Try to force $q_\theta(z|x_i)$ to be close to a standard normal (or some other simple density). This will ensure training images are mapped to proximal distributions.

VAE loss function

For a single data point x_i we get the loss function

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

The first term promotes recovery of the input. This is the data fidelity term that we saw earlier.

The second term keeps the encoding continuous – the encoding is compared to a fixed prior, $p(z)$, regardless of the input, which inhibits memorization (overfitting the data which prevents generalization).

With this loss function the VAE can (almost) be trained using gradient descent on minibatches.

We just need to determine how to implement the two loss terms in our loss function.

Implementing the second term

There is a closed form soln to KL divergence for the std normal dist, that you need to implement in the exercises:

$$-\mathbb{KL}((q_{\phi}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}))) = \int q_{\boldsymbol{\theta}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\right) d\mathbf{z}$$

$$= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)$$

Kingma and Welling 2014, ICLR, Auto-encoding Variational Bayes

Implementing the first term

For a single data point x_i we get the loss function

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

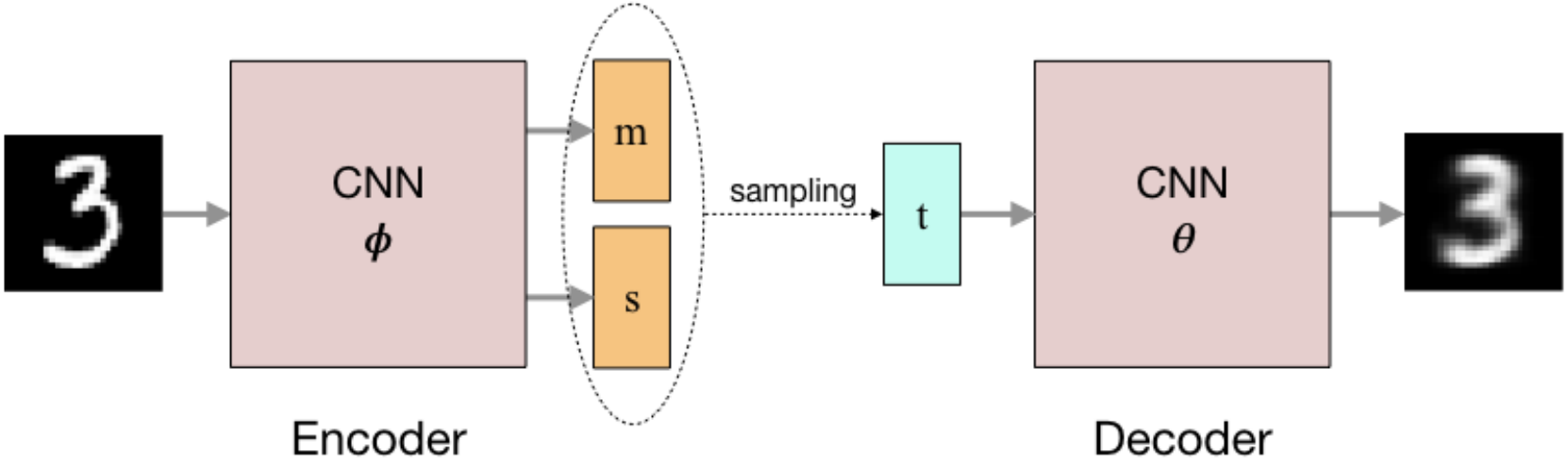
Problem: The expectation would usually be approximated by drawing samples and then averaging. This is not differentiable wrt θ and ϕ.

... Implementing the first term

Problem: The expectation would usually be approximated by drawing samples and averaging. This is not differentiable wrt θ and ϕ.

For example, for images a (convolutional) VAE's encoder is a CNN that learns a the means (m) and log variances (s) of the distn in z-space.

The CNN decoder, requires drawing from that learned distn, to recon the image.



https://nbviewer.jupyter.org/github/krasserm/bayesian-machine-learning/blob/master/variational_autoencoder.ipynb

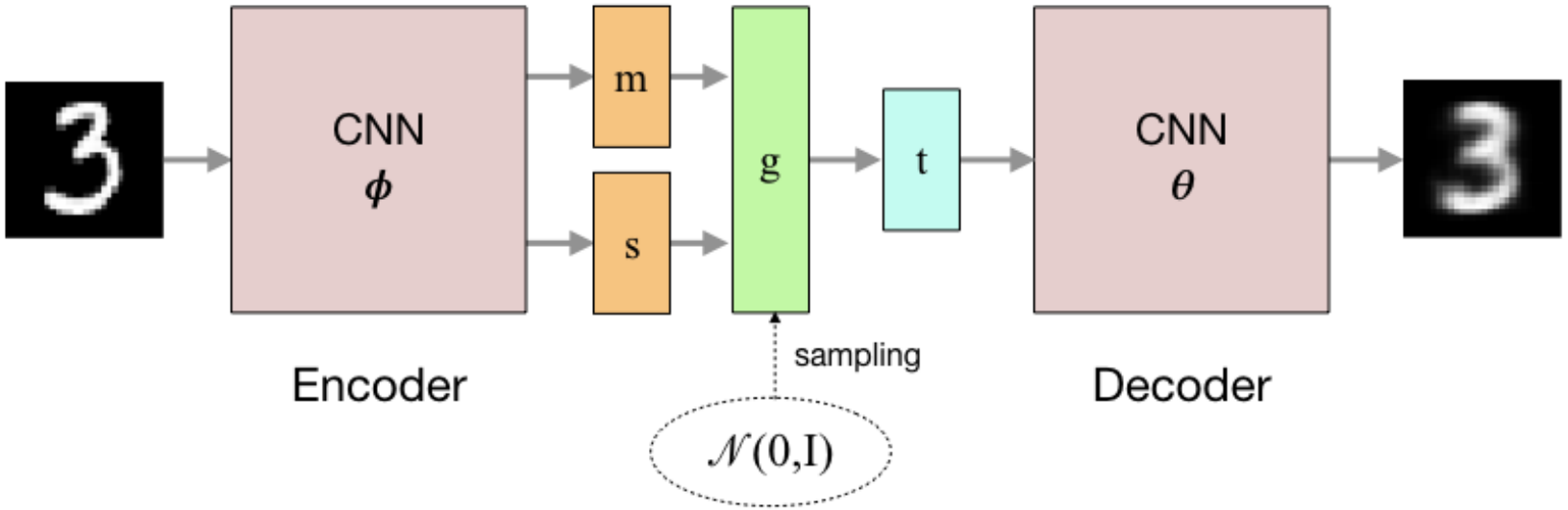
... Implementing the first term

Solution is a reparameterization:

If *z* ~ $N(\mu(x_i), \Sigma(x_i))$**, then we can sample from *z* using**

$$z = \mu(x_i) + \sqrt{(\Sigma(x_i))}\,\epsilon,$$

where ϵ **~ *N*(0,1). So we can draw samples from *N*(0,1), which doesn't depend on the parameters.**



https://nbviewer.jupyter.org/github/krasserm/bayesian-machine-learning/blob/master/variational_autoencoder.ipynb

Using the fully trained VAE model

After training, $q_\theta(z|x_i)$ is close to a standard normal, *N*(0,1), which is easy to sample.

Using a sample of *z* from $q_\theta(z|x_i)$ as input to sample from $p_\phi(x|z)$ gives an approximate reconstruction of x_i, at least in expectation.

If we sample any *z* from *N*(0,1) and use it as input to sample from $p_\phi(x|z)$ then we can approximate the entire data distribution *p*(*x*). That is we can generate new samples that look like the input but aren't in the input.

We can also walk along the dimensions of Z, (say in a grid) and display the reconstructions as we go, to visualize the learned meaning of Z space.

Exercise on object oriented VAEs

Walk through of the VAE Exercise

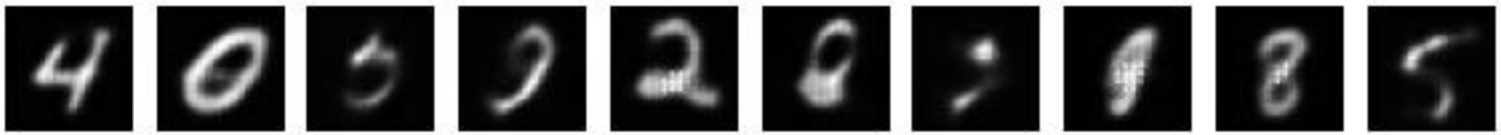
1. **Introduce the VAE code:**
   1. Everyone: Copy the code: /archive/course/SWE22/shared/week2/code
      1. cd to the folder of your local git repo in ~trainXX
      2. cp –R /archive/course/SWE22/shared/week2/code .
   2. Walk through the structure of /code
      1. At the level of /code are the *caller programs* (jupyter notebooks)
   3. Walk through today's caller program: code/train_vae_mnist.ipynb
      1. It uses MNIST data as the training set.
      2. Purpose is to apply the VAE class you complete to estimate the density of MNIST images and be able to reconstruct new digit images.
      3. It outputs reconstructed results throughout and at the end of training here:
         code/outputs/mnist_vae
         Folder contains reconstructed real digits and synthesized digits (next slides)
      4. Note: HINTS_SWE.ipynb contains helpful hints for this course.
   4. 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
   5. Note: /code/AD contains caller programs for applying what we build to AD

VAE Reconstructed images at epoch 1
epoch001_recons.png

What do you get as the # epochs increases?

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



What do you get as the # epochs increases?

… Walk through of the VAE Exercise

2. Walk through exercise 3 in Syllabus on MS Teams

3. Complete the Exercise 3 “ToImplement” occurrences in vae.py

1. **Show this in VSCode..**

4. You will need to git add and commit the new files (after you complete the exercises)

Acknowledgements


Albert Montillo, PhD, PI


Son Nguyen, PhD Postdoc


Alex Treacher PhD student

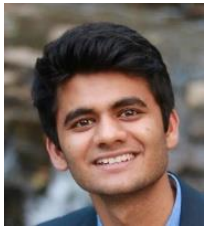

Aixa Andrade Hernandez, MS, PhD student


Austin Marckx PhD student


Krishna Chitta Res. Sci.

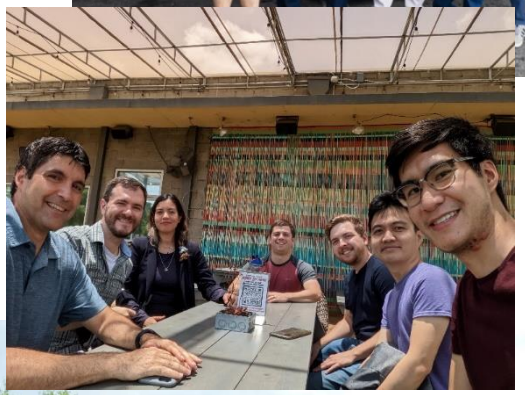
---------- Recent Alumni ----------


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