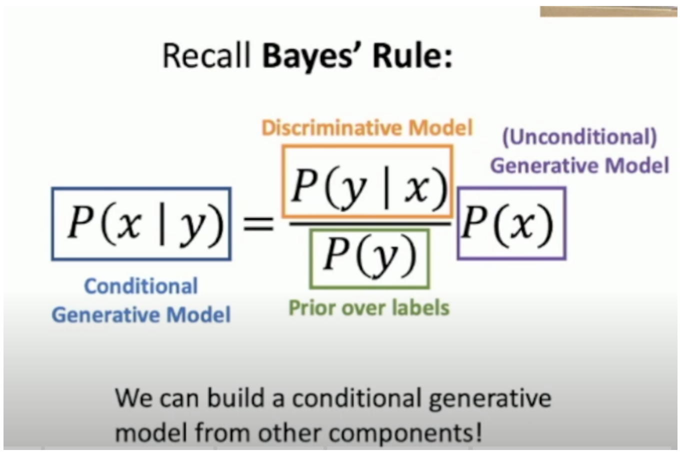
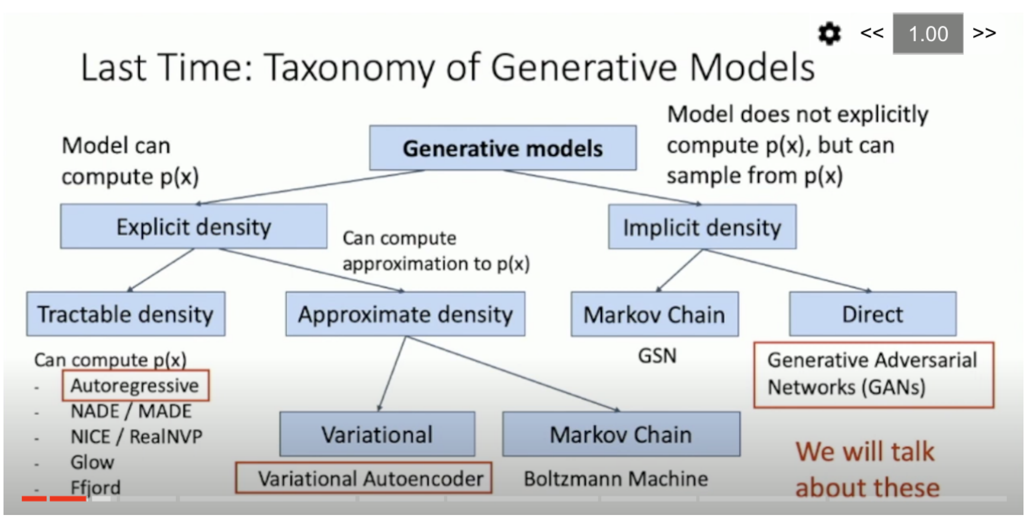
Generative Models, Part 2

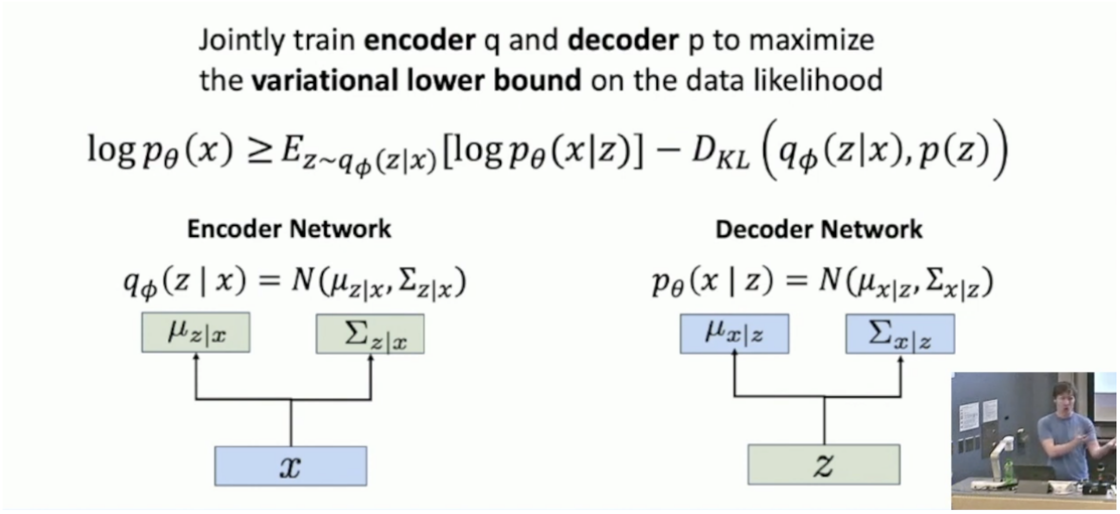
Discriminative Model:
- Learn distribution p(y | x)

Generative Model:
- Learn distribution p(x)
- Assign probability to each image in the universe

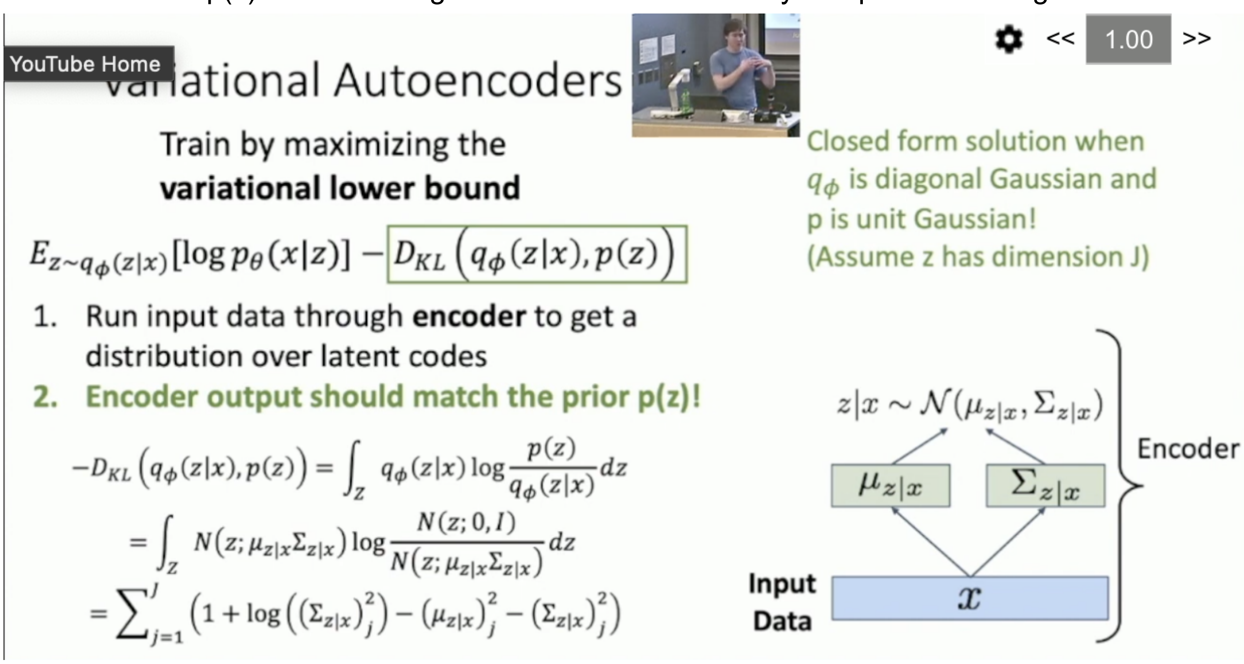
Conditional Generative Model:
- Learn p(x, y)
- Easy to build this type of model when we have built the other types of models
- 

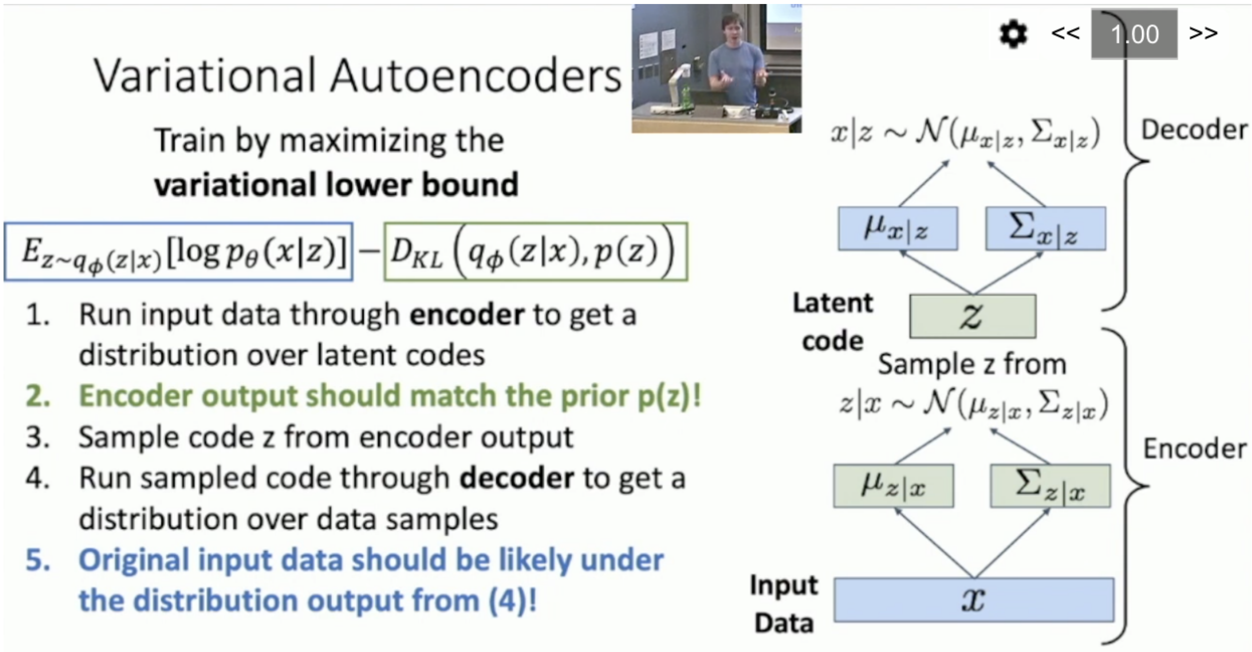


Variational Autoencoders:

- Gain: In addition to modeling likelihood of data, learn latent representation of data (z)
- 

- Maximize lower-bound on data-likelihood
- Encoder and decoder have to output probability distributions, not just a normal sample/vector
    - All probability distributions are diagonal gaussian distributions, so only need mu and sigmas to parametrize
- Prior on p(z) is said to be gaussian so we can actually compute KL-divergence





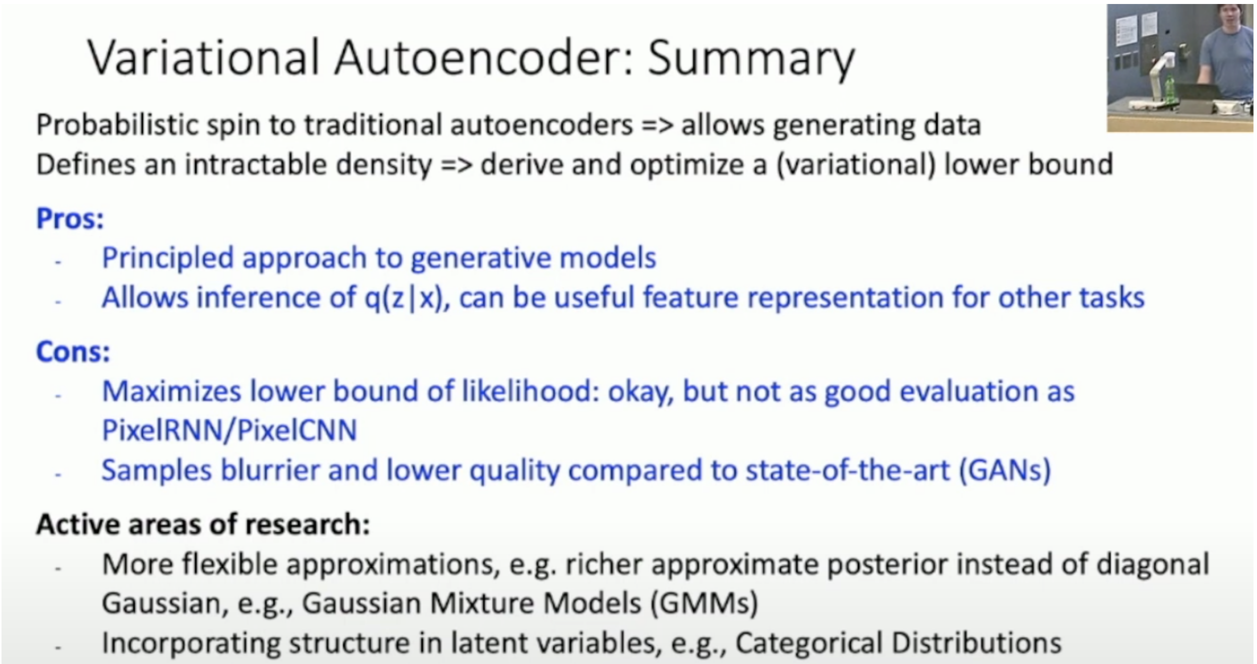
- Blue is data reconstruction term:
    - When you sample z from latent space, and then put that z vector back into the decoder, the original image should be most likely
    - Sort of like mean squared error loss in traditional autoencoder, incentivizes model to output the original input data

- Green term is regularization term
    - Predicted distribution should be simple and should be close to gaussian distribution

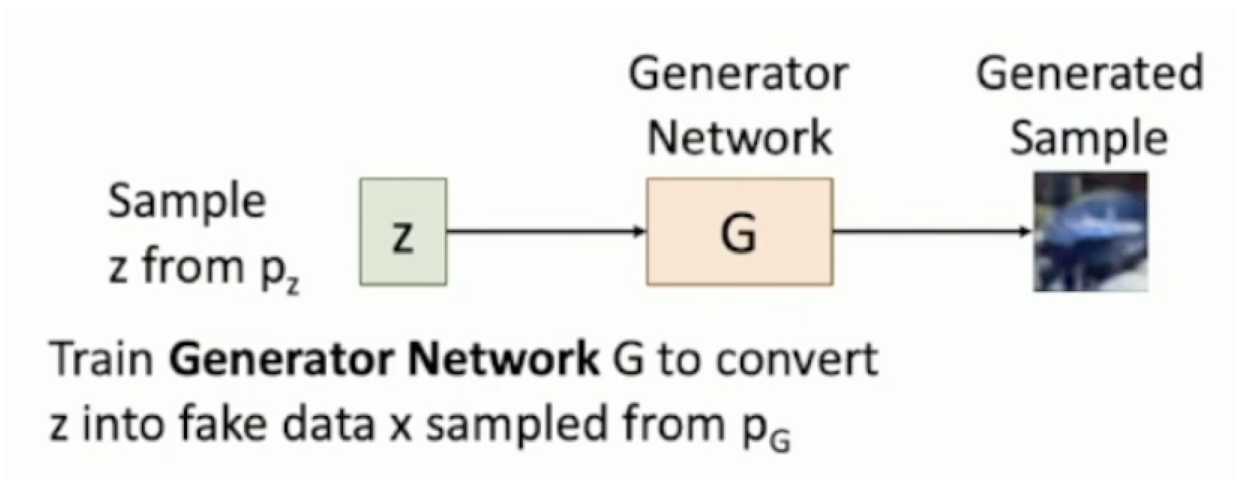
- Blue and green term are fighting against each other

- Can generate data by sampling from latent space once we have trained VAE
- Each dimension of z is independent (due to constraint that latent space followers diagonal gaussian)
- Can vary different elements of VAE to get smooth transitions between different aspects of the image

- Can also edit images by introducing perturbations in certain dimension of latent space



GANs

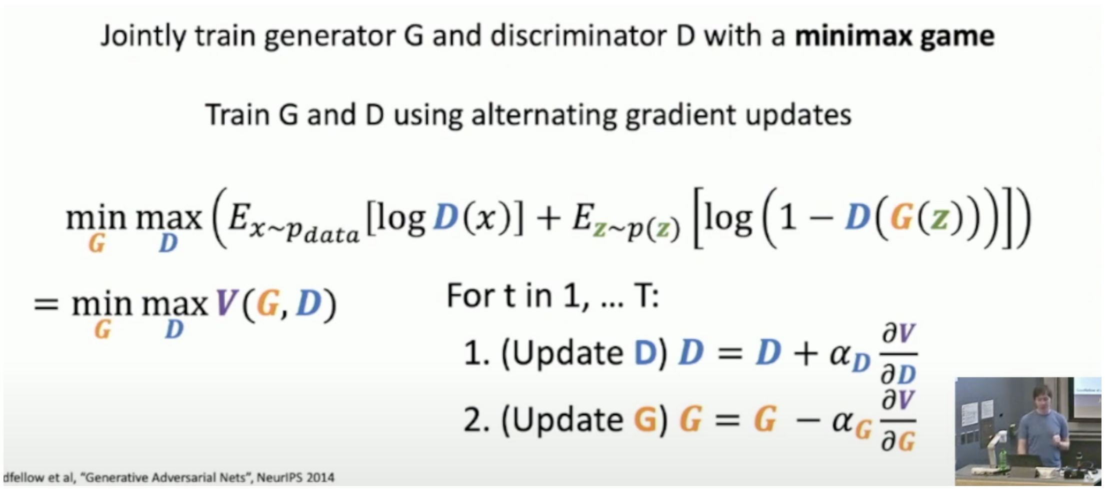
- Give up modeling p(x), but allow to draw samples from p(x)
- 

- Generator and discriminator are fighting against each other
- Hope is that samples from generator will end very similar to samples from actual data
- $$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

- D(x) = 1 for real data
- Generator does care about left-hand side

- Discriminator wants D(x) = 0 for fake data
- Right side incentivizes generator to be as good as possible

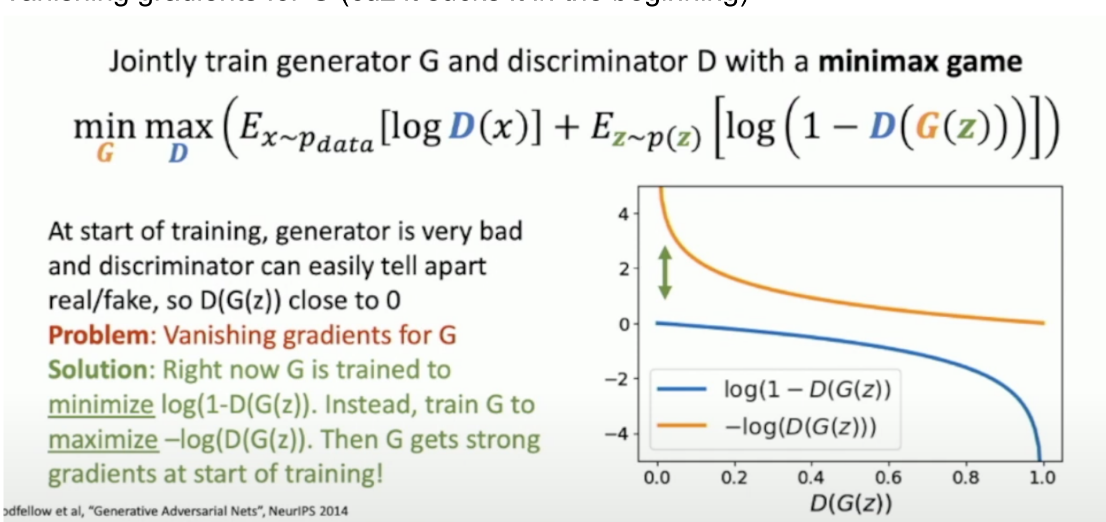
Use alternating gradient descent updates

- 

- Use gradient ascent for D (since we're trying to maximize)
- Use gradient descent for G (since we're trying to minimize)

Problem: Not minimizing any overall loss! No training curves that we can observe during training

Problem #2: D(G(z)) is close to zero
Problem: Vanishing gradients for G (cuz it sucks it in the beginning)

Solution: 

Is it supposed to be minimize or maximize :o???

Conditional GANs

- Given class