CPSC 440: Advanced Machine Learning Project

Variational Autoencoder (VAE)

Elvis Cai, Yilin Yang

27910165, 24754350

Motivating Problem

- Density Estimation for complex images
  - One of the hottest topics in ML
  - We need a state-of-the-art algorithm for image reconstruction and image generation.

Input Image:

Reconstructed Image:

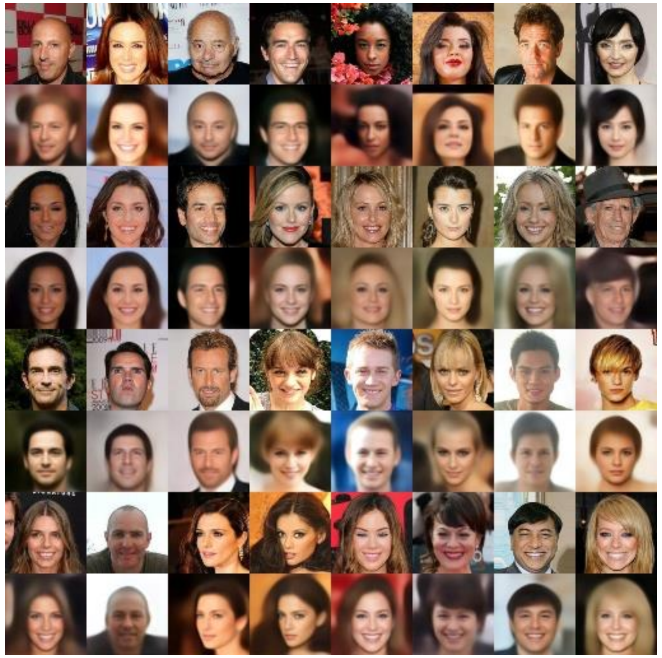


Image Reconstruction

Image Generation

Image source: https://github.com/orgoro/style-vae

VAE motivations

- Density Estimation
  - Modelling $p(x)$, $p(y)$, or $p(x, y)$
  - Tasks: Clustering, Dimension reduction, Outlier detections
  - Methods: Mixture Models, PCA
- Neural Networks
  - Modelling $p(y|x)$, but can model $p(x)$ (autoencoder)
  - Tasks: Classification(mostly)
  - Methods: Convolution NN, Recurrent NN
- Variational Autoencoder
  - Auto-Encoding Variational Bayes by DP Kingma, M Welling has been cited 14000+ times since 2013: https://arxiv.org/pdf/1312.6114.pdf
  - Idea: Neural Network for Density Estimation
  - Pros:
    - Density Estimation to Optimization, can utilize gradient descent
    - Generative model trained by neural network

VAE structure (more on this later):

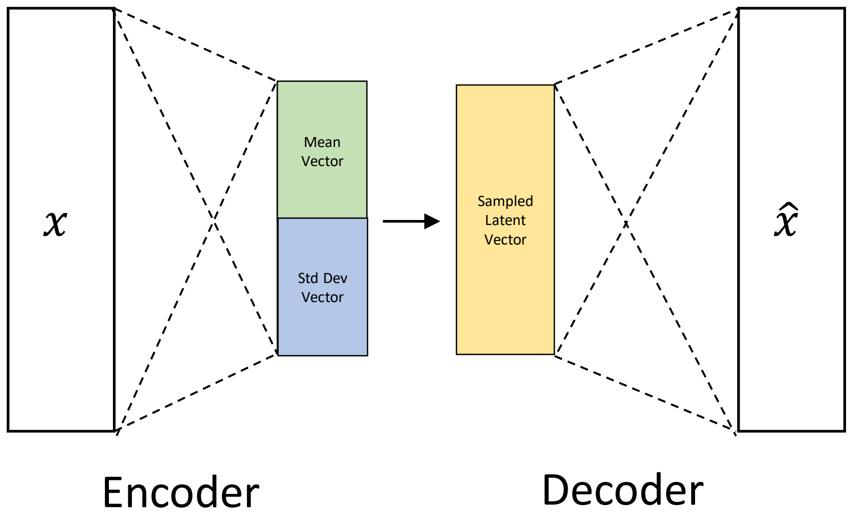


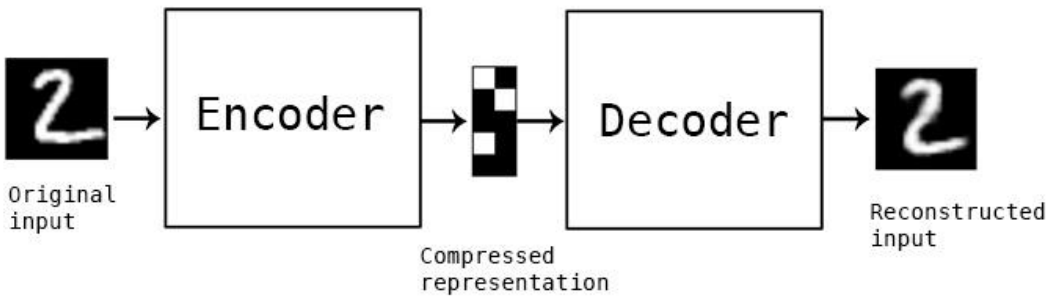
Image source: https://github.com/wojciechmo/vae

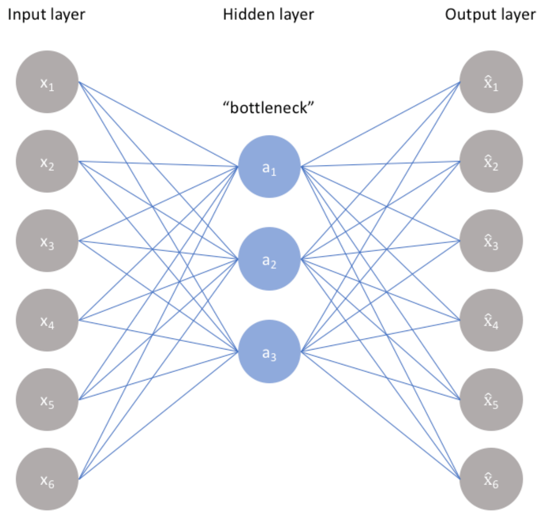
VAE Ingredients

- Autoencoder (Easy)
  - Network Structures
  - Loss Functions
- Variational Inference (Harder)
  - Motivations
  - KL Divergence

The AE in VAE: Autoencoder

- Data goes through a bottle neck, and AE attempts to reconstruct Data.





- Objective function: $\min|x - \hat{x}|$
- Encoder: a NN that maps Input x to latent variable z
- Decoder: a NN that maps z to $\hat{x}$
- If there is no nonlinearity, and one hidden layer: (Almost) PCA
  - With nonlinearity: generalization of PCA
- Applications: dimensionality reduction, denoising etc.
- Tends to overfit and maps each input to a z point.
  - (more on this in extra slides)

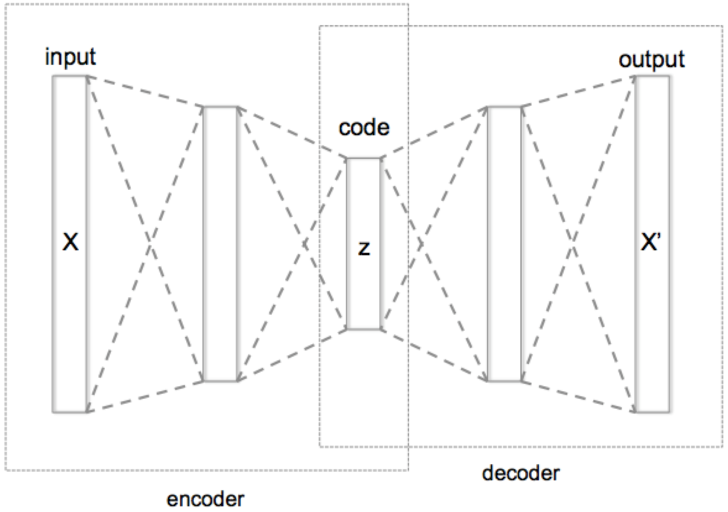


Image source: https://www.jeremyjordan.me/autoencoders/ and https://blog.keras.io/building-autoencoders-in-keras.html

Variational Inference Motivation: Bayesian Posterior Estimation

- Bayesian Inference: compute $p(z|x)$, using Bayes Formula: $p(z|x) = \frac{p(X|Z)p(z)}{p(X)}$
- We often only takes : $p(z|x) \propto p(x|z)p(z)$
- Issue: To find the exact posterior, according to the formula, we will need to compute the mariginal distribution: $\mathrm{p(x)} = \int_z p(x|z)p(z)dz$, which is often intractable.

- Approximating the Posterior:
  - Monte Carlo methods: (Recall)
    - Gibbs Sampling
    - Metropolis-Hastings
    - High variance, slow convergence rate, unbiased
  - Variational methods:
    - Low variances, fast convergence rate, Biased

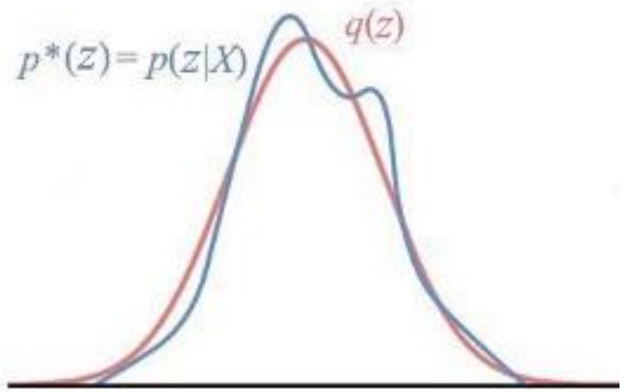
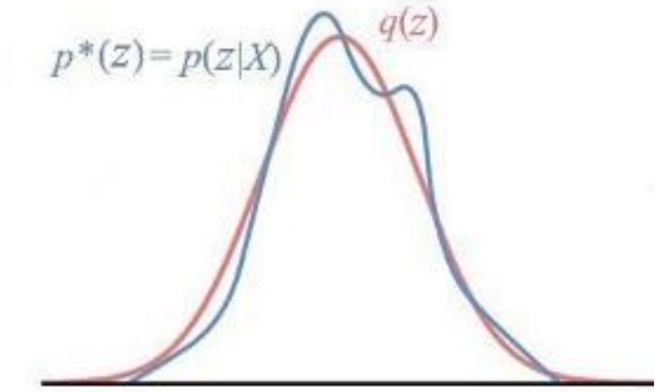


Image source: https://jonathan-hui.medium.com/machine-learning-variational-inference-273d8e6480bb

KL Divergence



- To approximate the posterior distribution $p(z|x)$, estimate it with a tractable distribution, $q(z)$
- Choice of $q(z)$? => A Gaussian distribution (gives a closed form solution)
- How good is $q(z)$? => KL (Kullback-Leibler) divergence measures difference between two distributions
  - Reverse-KL Divergence: $KL(p(z|x) \parallel q(z)) = \int_{-\infty}^{\infty} q(x) log \frac{q(z)}{p(z|x)}$
- Objective function: $\mathrm{argmin}_{q(z)} KL\big(p(z|x) \parallel q(z)\big)$

Image source: https://jonathan-hui.medium.com/machine-learning-variational-inference-273d8e6480bb

KL Divergence Details

$$KL(q(z)||p(z|x)) = \int_{-\infty}^{\infty} q(z) log \frac{q(z)}{p(z|x)} dz$$

$$= E_q[log(q(z))] - E_q[log(p(z|x))]$$

Some math

$$= \boxed{E_q[log(q(z))] - E_q[log(p(z,x))]} + \boxed{log(p(x))}$$

Define: -ELBO[q(z)]

Intractable
But constant

- ELBO: evidence lower bound (Variational lower bound)
- More details in the additional slide

$$\Rightarrow \quad log(p(x)) = KL(q(z)||p(z|x)) + ELBO[q(z)]$$

$$\Rightarrow \quad argmin_{q(z)}\ KL(q(z)||p(z|x)) == argmax_{q(z)}\ ELBO[q(z)]$$

Break

- Next, we move on to VAE

VI to VAE

$$argmin_{q(z)}\ KL(q(z)||p(z|x)) == argmax_{q(z)}\ ELBO[q(z)]$$

More math
(Details in the additional slide)

$$= argmax_{q(z)}\ -KL(q(z)||p(z)) + E_q[log(p(x|z)]$$

Take $q(z) == q(z|x)$, the distribution is dependent on x

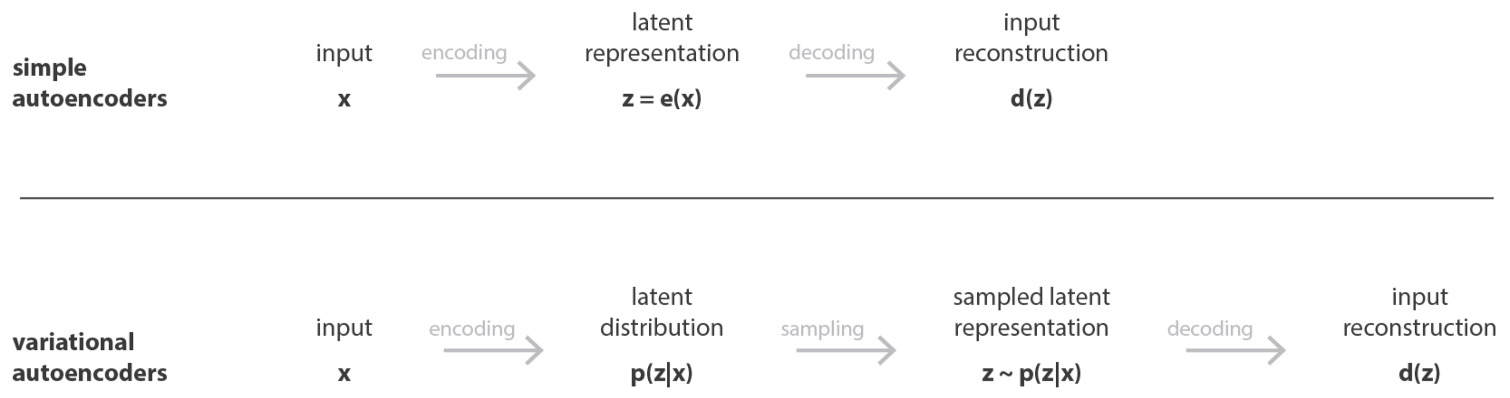
$$argmax_{q(z|x)}\ E_{q(z|x)}[log(p(x|z))] - KL(q(z|x)||p(z))$$

Reconstruction loss KL Loss

- Encoder be $q(z \mid x)$ that generates a Gaussian Distribution
- Decoder be $p(x \mid z)$, that maps z to $\hat{x}$ (Same as AE)
- Take $p(z) \sim N(0,1)$, i.e., we want the distribution of latent variable z to be as close to a standard normal distribution as possible
- You will see the benefit of the two Gaussian assumptions in the homework

VAE Implementation

- The main issue is that so far, we haven't introduced the non-deterministic (random-ness) aspect of VAE!

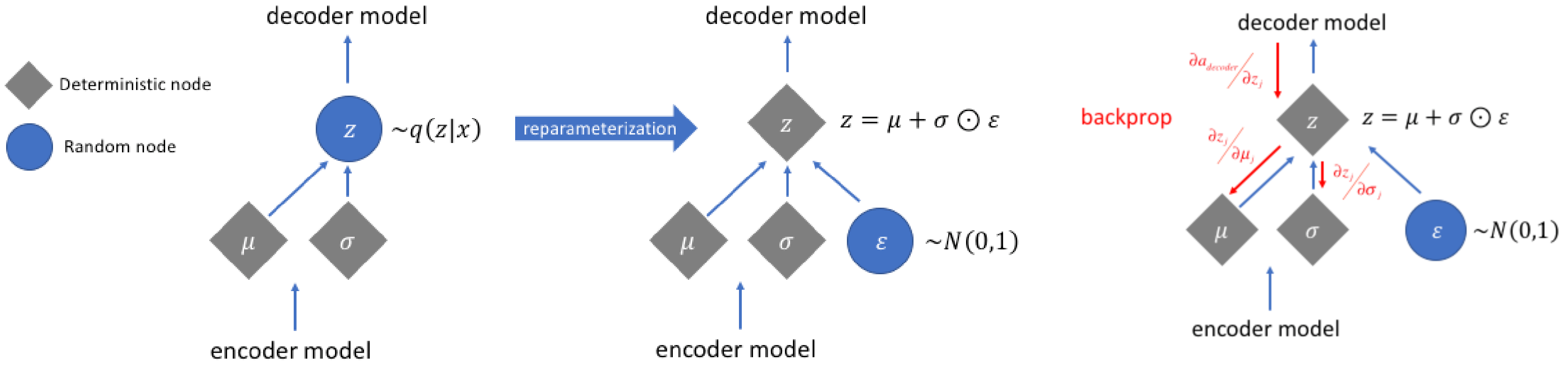


- We sample from the latent distribution to get a sampled latent vector z as input from the decoding NN.
  - Without this step we are basically doing an AE
- We also choose a training example x in each iteration
- Therefore, we train VAEs with stochastic gradient.

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

Further Issues Introduced by Sampling Latent Distribution

- The sampling method leads to a problem in backpropagation:



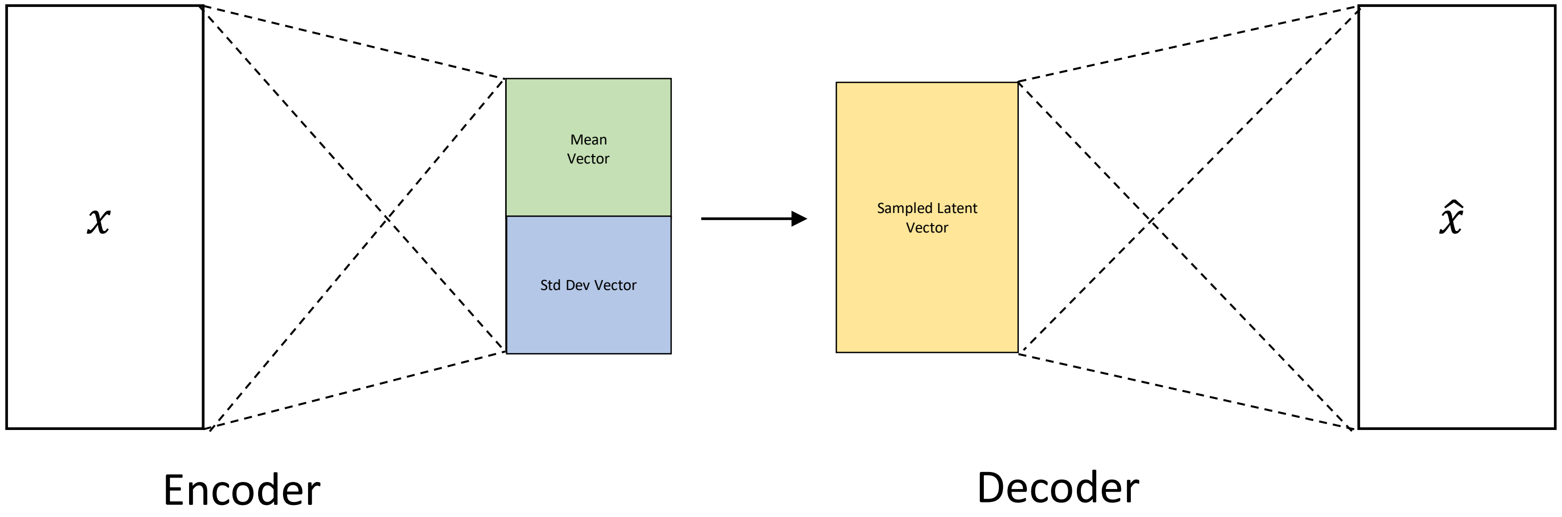
- As the z nodes are random, we cannot use backpropagation.

- "reparameterization trick": randomly sample ε from $N(0, I)$, and then by $z = \mu + \sigma \odot \varepsilon$, shift the randomly sampled ε by the latent distribution's mean and scale it by the latent distribution's variance.
  - Monte Carlo approximation of the gradient.

- Thus, We can finally use stochastic gradient to train VAE.

Image source: https://www.jeremyjordan.me/variational-autoencoders/

VAE Summary

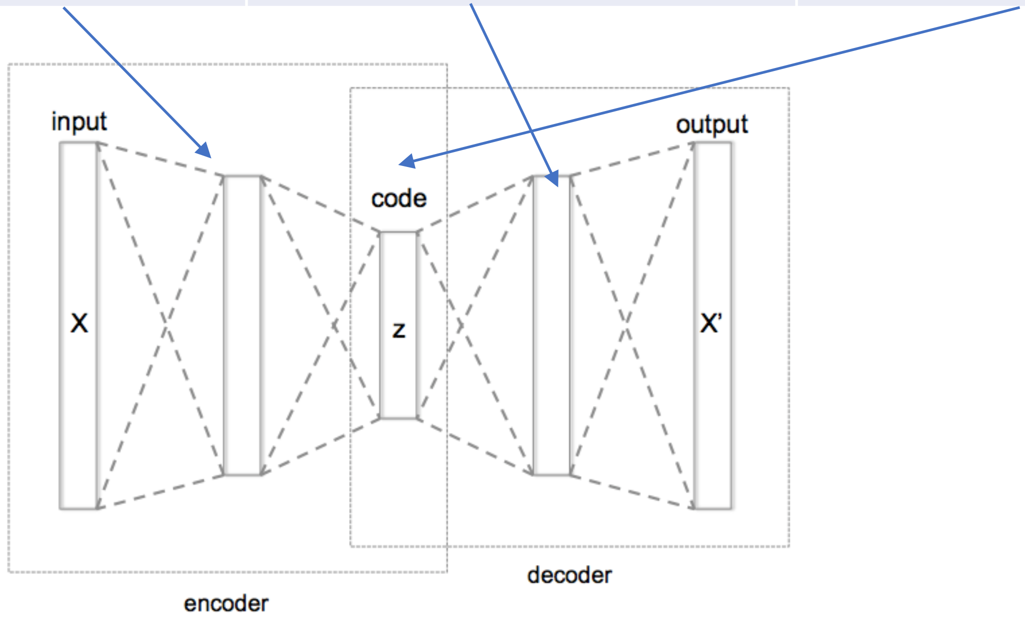


- Objective function: $argmax_{q(z|x)}\ E_{q(z|x)}[log(p(x|z))] - KL(q(z|x)||p(z))$
- Encoder: $q(z|x)$
- Decoder: $p(x|z)$
- Latent variables: $q(z)$

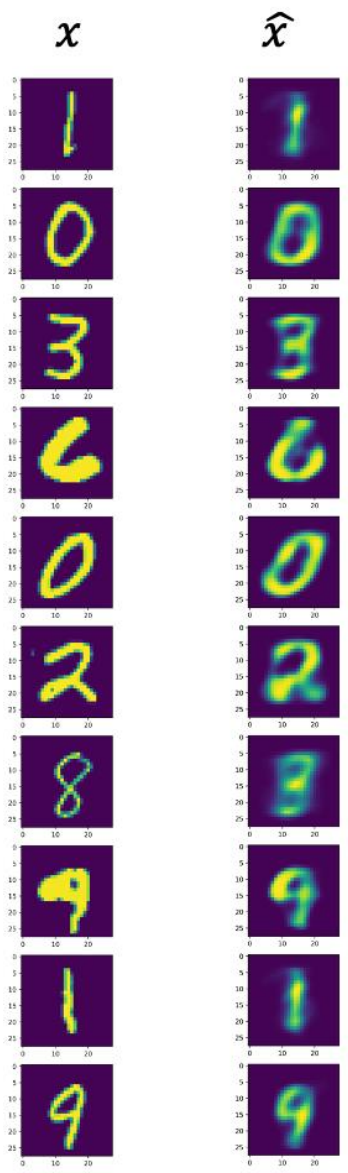
AE, Deep AE, VAE, Deep VAE on MNIST

Experimental Model Definitions

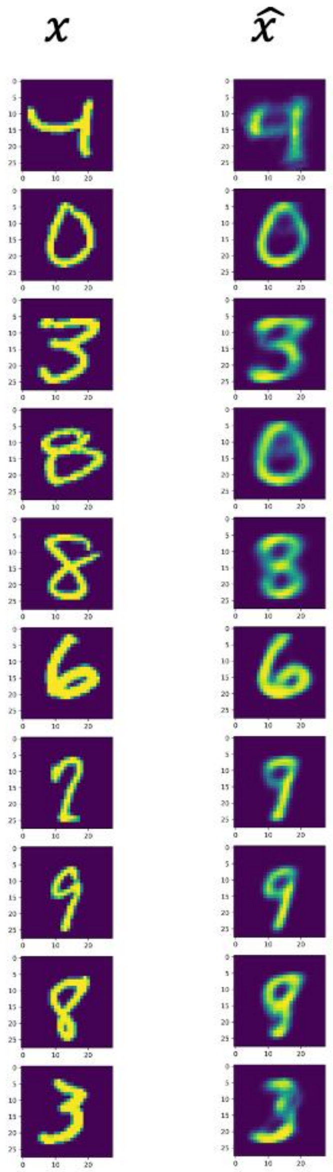
Model	Hidden Layers (encoder)	Hidden Layers (decoder)	Latent Layer dimensions	Loss
AE non-deep	[0]	[0]	[8]	Binary_Crossentropy
AE deep	[64, 32]	[32,64]	[8]	Binary_Crossentropy
VAE non-deep	[0]	[0]	[8(mean), 8(cov)]	BC + KL_Loss
VAE deep	[64, 32]	[32,64]	[8(mean), 8(cov)]	BC + KL_Loss



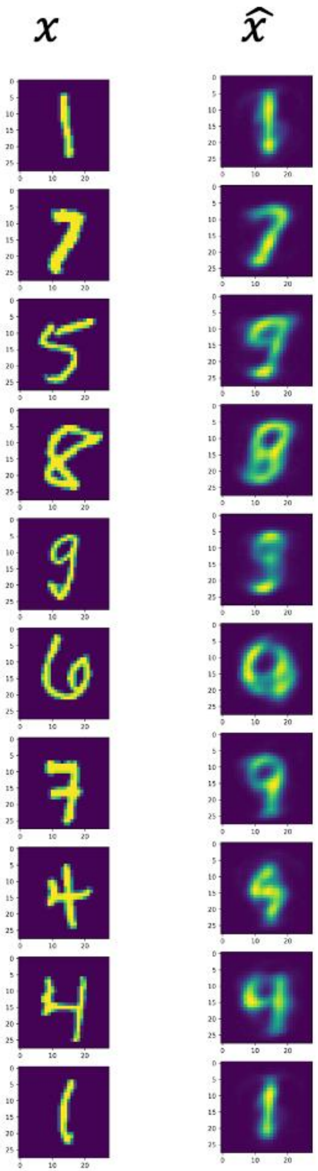
AE non-deep
Loss 0.1734



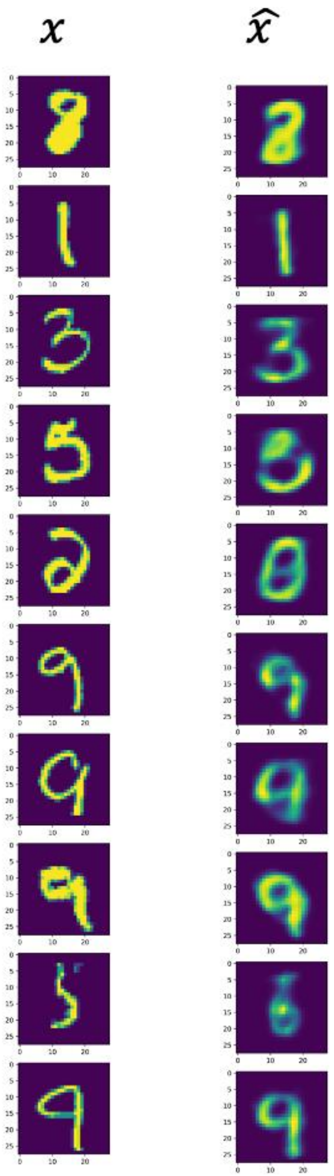
AE deep
loss 0.1477



VAE non-deep
loss 153.3227

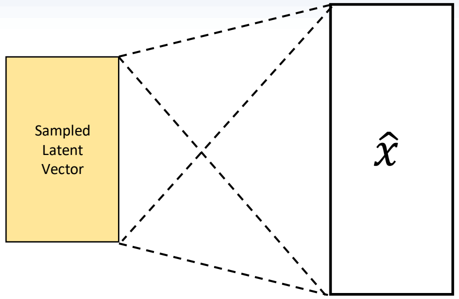


VAE deep
loss 129.3765

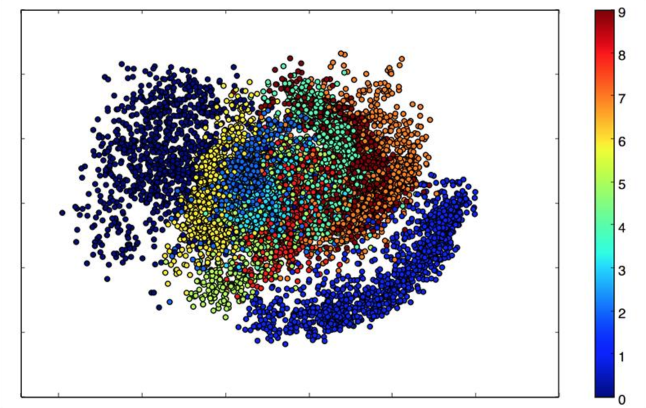


Inference Task: VAE as a Generative Model

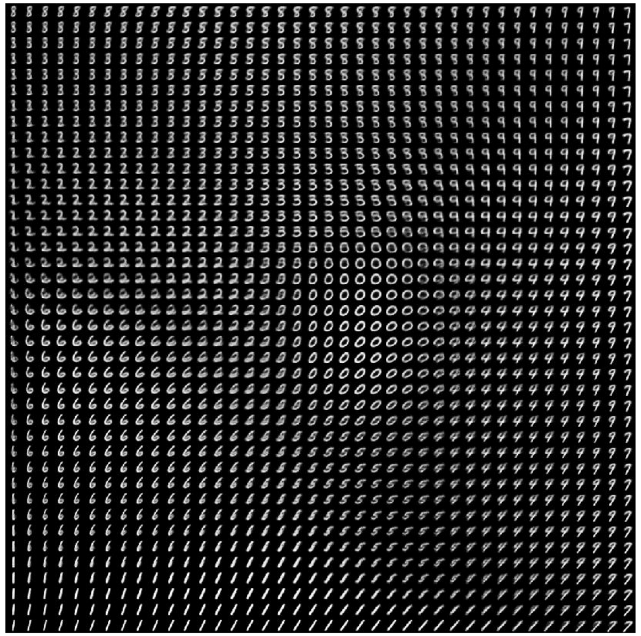
- Once the VAE is trained, we can forget about the encoder!
  - Since our objective is to generate unseen x, we don't need the encoder.



Decoder



- We can sample from the latent distribution and directly feed it to the decoder.
- Left is a cool MNIST 2D Visualization of latent space



- This was sampled from a two-dimensional Gaussian (outputted by the encoder) and inputted into decoder network.

Image source: https://blog.keras.io/building-autoencoders-in-keras.html and https://www.jeremyjordan.me/variational-autoencoders/

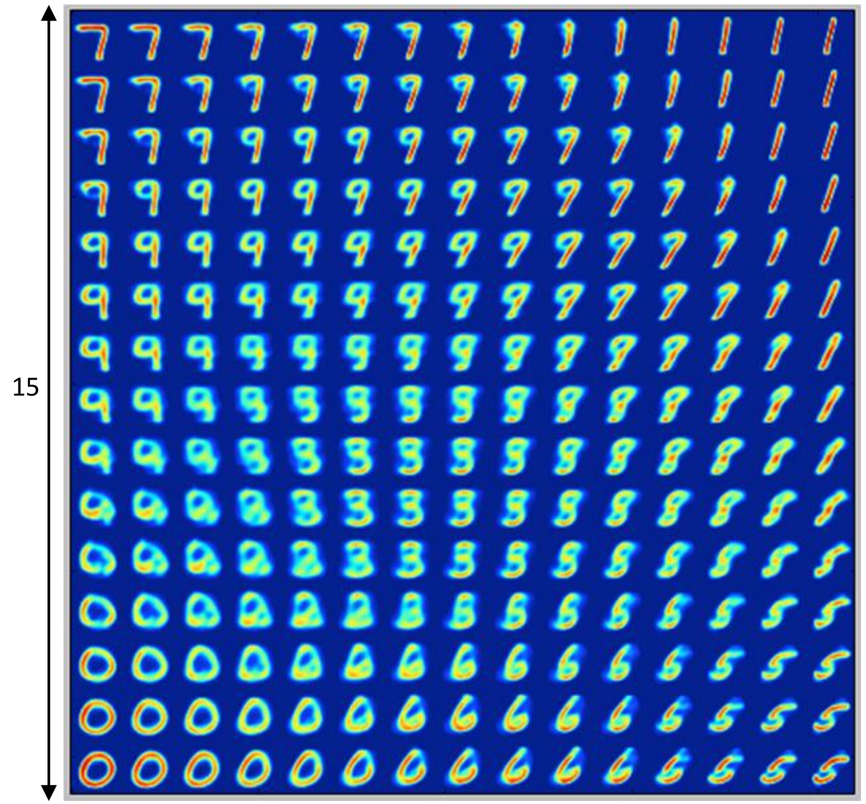
Generative model code:

- Implementing the previous slide in code:

```
// we want to sample various datapoints for 2D space:
grid_x = array from -15 to 15 (15 for example)
grid_y = array from -15 to 15

for i in grid_x:
   for j in grid_y:
      z_sample = [i, j]
      x_decoded = decoder.predict(z_sample)
      // format the x_decoded to the correct dimensions
      digit = x_decoded[0].reshape(digit_size, digit_size)
      // ADD the decoded digit to plot
// show plot
```

The resulting plot will look like this:



- You can also just sample n random points in 2D space to generate n samples instead of a grid.

Source: https://blog.keras.io/building-autoencoders-in-keras.html

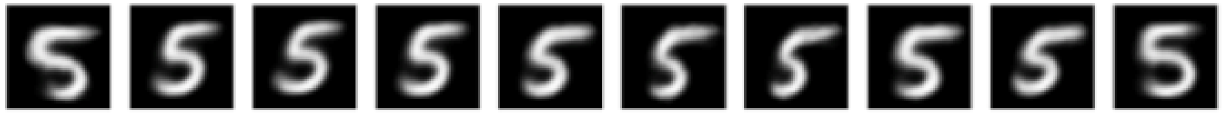
Conditional VAE (Discriminative Modelling)

- New Problem: How do we make use of class labels? – Conditioning encoder and decoder on the class labels

- New Objective function, conditioning on class label c, becomes:

$$\log P(X|c) - D_{KL}[Q(z|X,c)||P(z|X,c)] = E[\log P(X|z,c)] - D_{KL}[Q(z|X,c)||P(z|c)]$$

- Consequences:
  - Decoder becomes $P(X|z,c)$, meaning that given the latent variable z, and class label c, we could generate samples of a particular class. For example, if we want to generate more $c = 5$



  - Better image reconstructions:

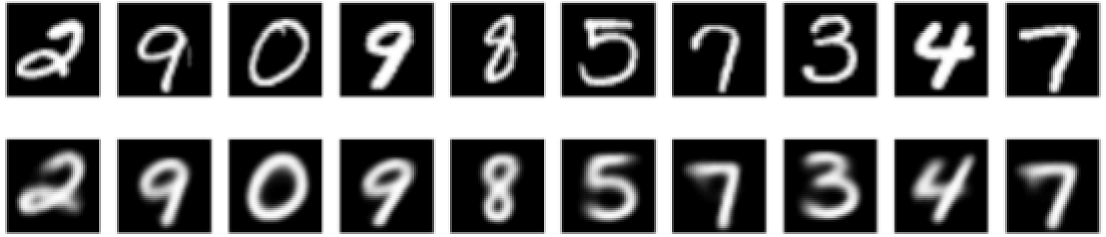


Image & Equation source: https://wiseodd.github.io/techblog/2016/12/17/conditional-vae/

VAE further results: MusicVAE

- Link: https://www.youtube.com/watch?v=G5JT16flZwM&t=7s
- Check out more at: https://magenta.tensorflow.org/music-vae
- This algorithm provides easy interpolations of simple loops for musicians

Summary

- VAE Ingredients
  - AE
  - Variational Inference
    - KL-Divergence
- Training VAE
- VAE on MNIST
- VAE as a Generative Model
- VAE as a Discriminative Model

KL Divergence Derivation

KL derivations – ELBO: Let $p(z|x)$ and $q(z)$ be the two targeting distributions, want to calculate $KL(q(z)||p(x|z))$

$$KL(q(z)||p(z|x)) = \int_{-\infty}^{\infty} q(z) log \frac{q(z)}{p(z|x)} dz$$

$$= \int_{-\infty}^{\infty} q(z) log(q(z)) - \int_{-\infty}^{\infty} q(z) log(p(z|x)) dz$$

$$= E_q[log(q(z))] - E_q[log(p(z|x))]$$

$$= E_q[log(q(z))] - E_q[log \frac{p(z,x)}{p(x)}]$$

$$= E_q[log(q(z))] - E_q[log(p(z,x))] + E_q[log(p(x))]$$

$$= E_q[log(q(z))] - E_q[log(p(z,x))] + log(p(x))$$

Let $ELBO[q] = -E_q[log(q(z))] + E_q[log(p(z,x))]$

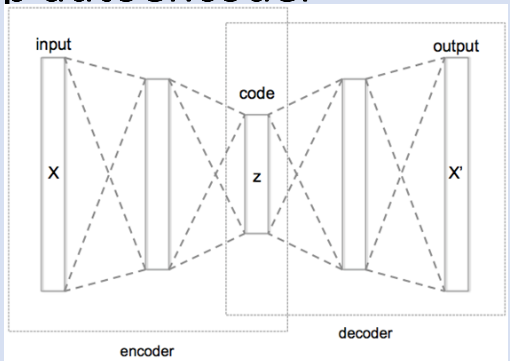
$log(p(x)) = KL(q(z)||p(z|x)) + ELBO[q(z)]$

$=> argmin_{q(z)}\ KL(q(z)||p(z|x)) == argmax_{q(z)}\ ELBO[q(z)]$

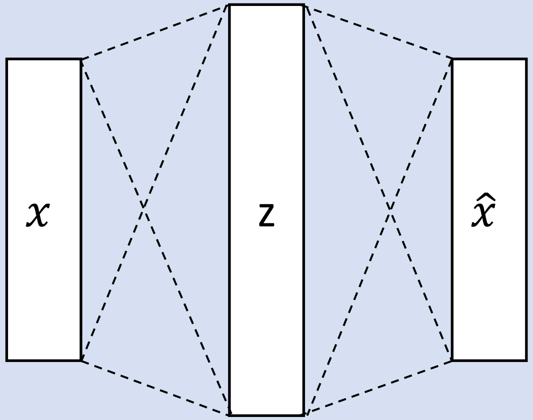
- ELBO: evidence lower bound (Variational lower bound)
- As $\log(p(x))$ is fixed, maximizing the ELBO gives more information on p(x), we can do this by minimizing KL.
- Good explanation from Information to KL: https://youtu.be/uaaqyVS9-rM?t=583 (Good Video about VAE)

Variations on Autoencoder

- We can add more layers: Deep autoencoder



  - Can yield better results than shallow models.
- Overcomplete autoencoder



- Issue: there's always a trivial solution!
  - We can: regularise OR add noise to input ($x + \epsilon$) to avoid it: noise reduction

Image source: https://en.wikipedia.org/wiki/File:Autoencoder_structure.png

Variational Inference to VAE

Looking back at the Variational Inference function:

$$argmin_{q(z)}\ KL(q(z)||p(z|x)) = argmax_{q(z)}\ ELBO[q(z)]$$

$$= argmax_{q(z)}\ E_q[log(p(z,x))] - E_q[log(q(z))]$$

$$= argmax_{q(z)}\ E_q[log(\frac{p(x|z)p(z)}{p(x)})] - E_q[log(q(z))]$$

$$= argmax_{q(z)}\ E_q[log(p(x|z)] + E_q[log(p(z))] - E_q[log(p(x)] - E_q[log(q(z))]$$

$$= argmax_{q(z)}\ -E_q[log(q(z))] + E_q[log(p(z))] + E_q[log(p(x|z)] - E_q[log(p(x)]$$

$$= argmax_{q(z)}\ \ -KL(q(z)||p(z)) + E_q[log(p(x|z)] - constant$$

$$= argmax_{q(z)}\ \ -KL(q(z)||p(z)) + E_q[log(p(x|z)]$$

Assume instead we approximate $q(z)$ with $q(z|x)$, we get:

$$E_{q(z|x)}[log(p(x|z))] - KL(q(z|x)||p(z))$$

- $E_{q(z|x)}[log(p(x|z))]$ can be thought of as construction error
- The decoder network is deterministic, $p(x|z) = \ p(x|\hat{x})$
  - If $p(x|\hat{x})$ is gaussian, we have the term $e^{-|x\ -\hat{x}|^2}$ so we observe that this is:
  - $\min|x\ - \hat{x}|^2\ -\ KL\ (q(z|x)\ ||\ p(z))$ = L2-loss + KL regularization
  - This is like autoencoder (reconstruction loss) + regularizer where the regularizer constraints q to be similar to p.
- Variational Inference: we are using $q(z\mid x)$ to approx. the posterior $p(z\mid x)$

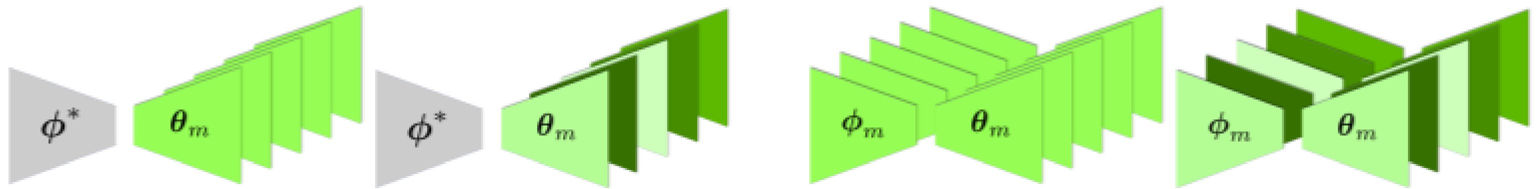
Bayesian VAE

- New method proposed in a 2019 paper: Bayesian Variational Autoencoders for Unsupervised Out-of-Distribution Detection by Erik Daxberger, José Miguel Hernández-Lobato (Link: https://arxiv.org/pdf/1912.05651.pdf)

- The generative process draws a $z \sim p(z)$ from its prior and a $\theta \sim p(\theta|D)$ from its posterior (D here is training data), and then generates $x \sim p(x|z, \theta)$

- We now need Bayesian inference on two posteriors $x \sim p(x|z, \theta)$ and $\theta \sim p(\theta|D)$

- The paper proposes two variants to achive this inference:



(a) Variant 1 with a shared encoder ϕ^* and M decoders (b) Variant 2 as an ensemble of M VAEs (ϕ_m, θ_m)

Figure 1: Illustrations of the (a) first and (b) second BVAE variant (with $M = 5$), with (left) *agreement* and (right) *disagreement* in their likelihoods $p(\mathbf{x}^*|\theta_m)$ (as encoded by color intensity).