VAE

KL-divergence is used to measure difference between two distributions

It can be used to measure divergence b/w Learned latent variable distribution and the normal Guassian.

$$D\left(q_\phi(z/x) \,||\, p(z)\right) = -\frac{1}{2}\sum_{j=0}^{K-1}\left(\sigma_j + \mu_j^2 - 1 - \log\sigma_j\right)$$

↑ Inferred Latent distribution

↑ Fixed prior on latent distribution.

When we use normal distribution as prior

Using mean and σ as input and computing distance metric that captures this divergence.

Properties expected from regularization:

1- Continuity :. Points that are close in latent space. should result in similar reconstruction after decoding.

2. Completeness :. When we sample from the latent space, it should result in meaningful content after decoding.

if we donot regularize
there could be a scenario where
there are two points in latent space
that are close to each other but
they are not properly ~~decoded~~. closed when they are decoded
or a point selected
randomly may not decode properly.

Mean and variance

For VAE: Loss function has both
reconstruction loss and regularization
function.

If there is no ~~VAE~~ regularization
function then model will try to
reduce reconstruction loss. even though
we are encoding the latent varible
via mean and variance

Consequences:

1) Latent varible variance may end
up being very small. This may lead
to pointed distribution

2) Means that are totally divergent from each other, which will Lead to discontinuity in the latent space

Even though we may be optimizing the reconstuction loss.

This is why regularization is important. Normal prior will encourage the learned latent variable distributions to overlap in latent space.

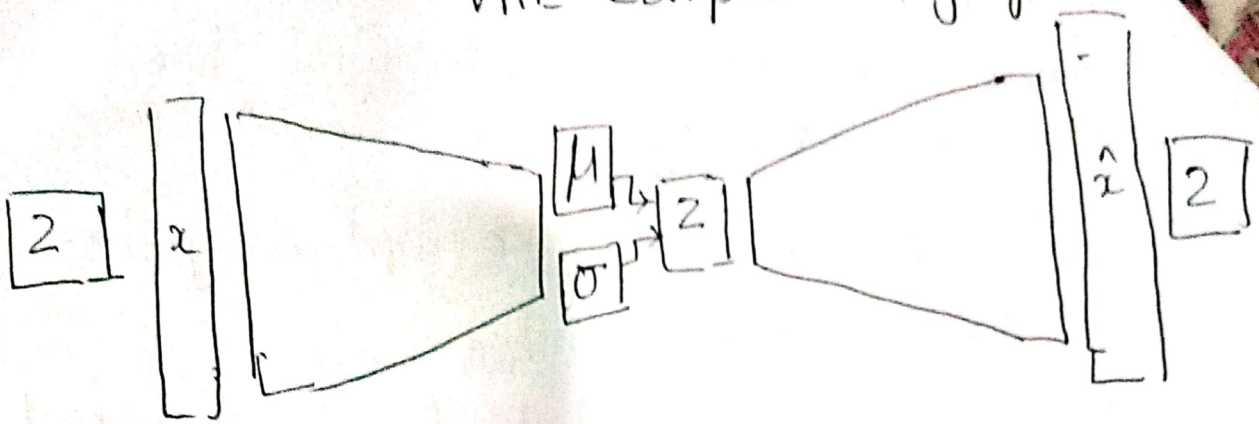
$\mu = 0$, $\sigma^2 = 1$,

Center means regularize the variance.

Points and distances in the latent Space has some relationship in the generated input.

Trade off exists between regularization and reconstruction, if we regularize more than it will affect the quality of the generated input.

VAE Computation graph



Encoder computes $q_\phi(z/x)$

Decoder computes: $p_\theta(x/z)$

$L(\phi, \theta, x)$ = reconstruction loss + regularization term

Problem: How to backpropagate?
gradient through sampling

Because we have introduced stochastic sampling layers, we cannot backpropagate through them as they have notion of stochasticity (having a random probability probability distribution)

This is because backpropagation requires deterministic behaviour.

Cannot compute gradient through stochastic sampling.

Reparameterize the Sampling layer

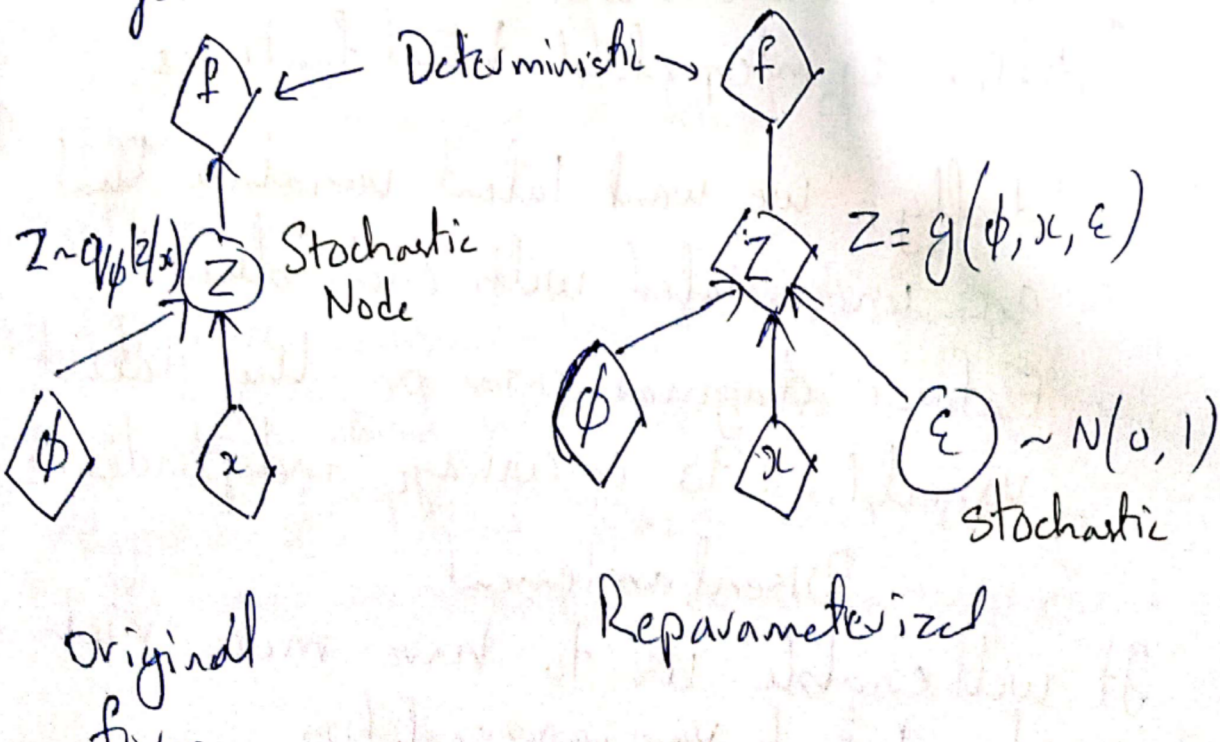
- Consider the sampled latent vector z as a sum of
  - a fixed μ vector
  - fixed σ vector, scaled by random constants drawn from prior distribution

$$\Rightarrow z = \mu + \sigma \odot \epsilon$$

where $\epsilon \sim N(0,1)$

we still have stochasticity which is introduced by random constant ϵ

But ϵ is not in the bottleneck latent layer.



VAEs: Latent perturbation

A side effect on imposing distributional priors is that we can actually sample latent variables and individually tune them. while keeping all the other variables fixed.

We can tune the value of a particular latent variable and run the decoder each time this ~~decoder~~ variable is changed to generate a new reconstructed output.

Different dimensions of z encodes different interpretable latent features

Ideally, we want latent variables that are uncorrelated with each other
Enforce diagonal prior on the latent variables to encourage independence

Disentanglement

It will enable us to have more rich and complex latent ~~var~~ representation

Latent space disentanglement with β-VAEs

Standard VAE Loss

$$\mathcal{L}(\theta, \phi; x, z, \beta) = \underbrace{E_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction}} - \beta \underbrace{D_{KL}(q_\phi(z|x) \| p(z))}_{\text{Regularization}}$$

Control strength of regularization term

Increasing β encourages disentanglement

$\beta > 1$: constrain latent bottleneck, encourage efficient latent encoding → disentangled

Lab

- Automatic Debiasing of facial classification system

Build up a learned latent distribution of face data and use this to identify regions of latent space that are going to be over represented or under represented.

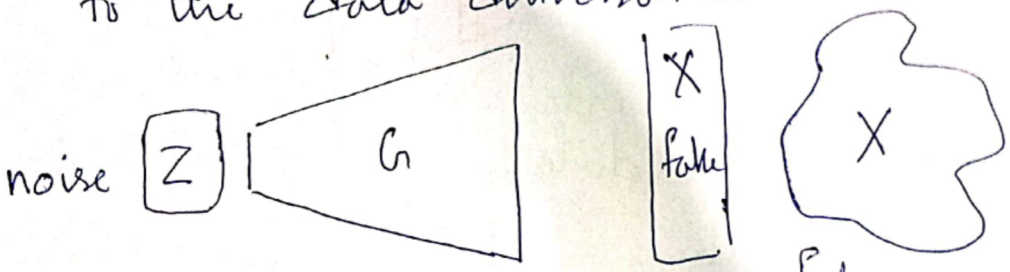
VAE Summary

1- Compress representation of world to something we can use to learn
2- Reconstruction allows for unsupervised learning.
3- Reparameterization trick to train end to end
4- Interpret hidden latent variables using perturbation
5- Generating new examples.

Generative Adversarial Networks

Idea:. Instead of modeling density just sample to generate new instances.

Solution:. Work through some approximation of distribution or sample from some noise and learn a transformation to the data distribution.



Generator Network G

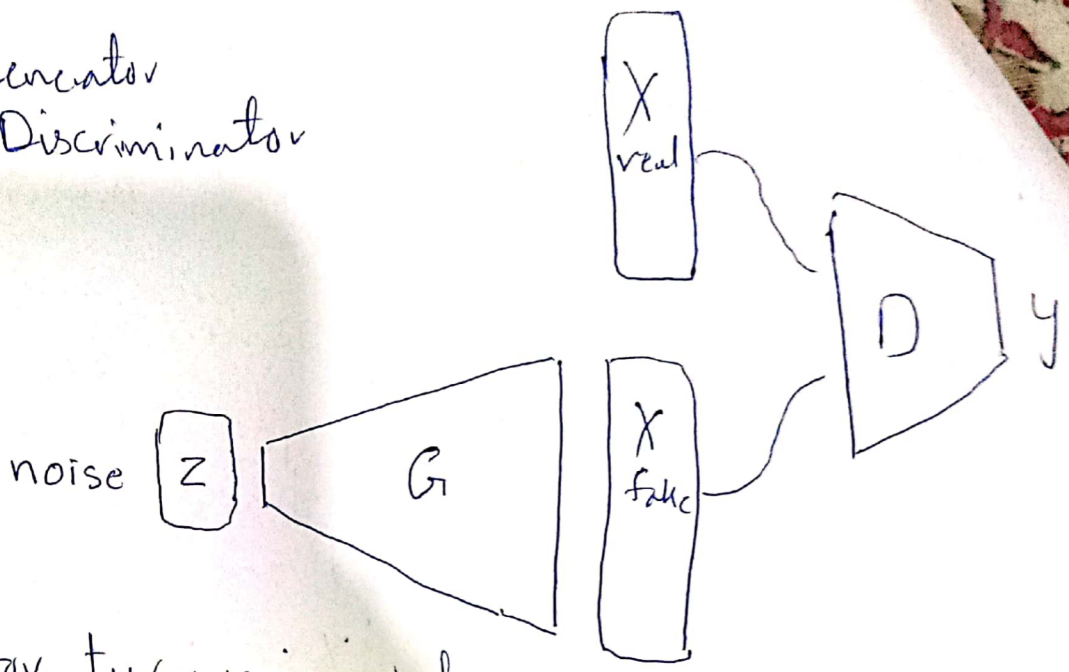
Fake sample from learned representation of data distribution

Start from simple noise (random) and try to build GAN that can learn from functional distribution to go from noise to data distribution.

After learning this functional generative mapping, we can create fake data which is close to the actual data

GAN has two components.

- Generator
- Discriminator



generator turns noise into an intimation of the data to try to trick the discriminator

The discriminator tries to identify real data from fakes created by the generator

G:. from random noise to produce imitation of data

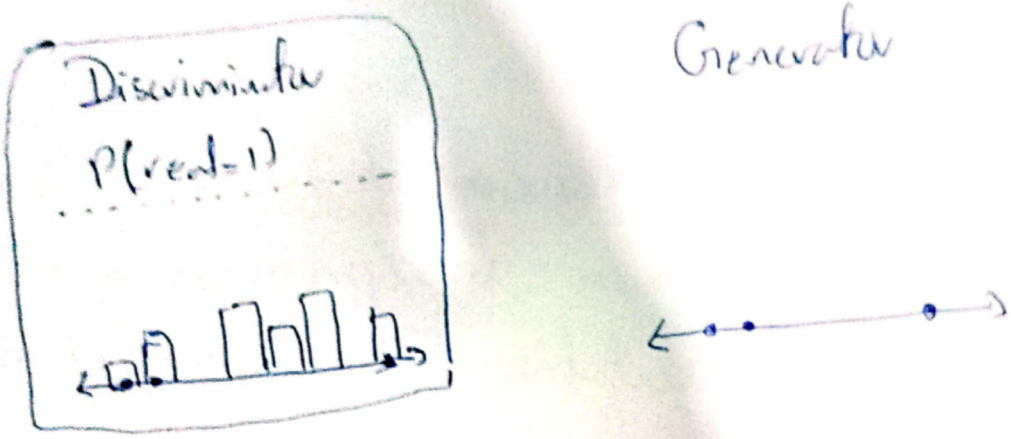
D:. take real and fake. Distinguish

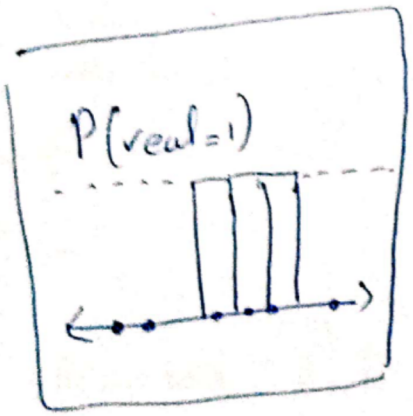
During training G and D will compete.

D: will get better in classification

G: will get better in production

D will try to force the G to be better in fooling D.





generator will then try to improve so that the synthetic data may lie close to the real data

Discriminator will then receive new pixels.

Discriminator: tries to identify the synthesized instances.

G: tries to synthesize fake instances that fool D.

Training: adversarial objectives for D and G

Global optimum: G reproduces the true data representation. D cannot discriminate b/w fake and real.

Loss function: Cross Entropy

D: Maximize that the fake data is identified as fake

G(z): o/p of Generator

D(G(z)): Discriminator's estimate of the probability that a fake instance is actually fake

D(x): is the discriminator's estimate of the probability that a real instance is fake.

$$\arg\max_D E_{z,x}\left[\underbrace{\log D(G(z))}_{\text{Fake}} + \underbrace{\log(1-D(x))}_{\text{Real}}\right]$$

Maximize probability

Generator:- taking random noise
it cannot affect $D(x)$ directly
G will have adversarial affect to G.
$D(x)$ is solely based on the real data
G will try to minimize goal of D

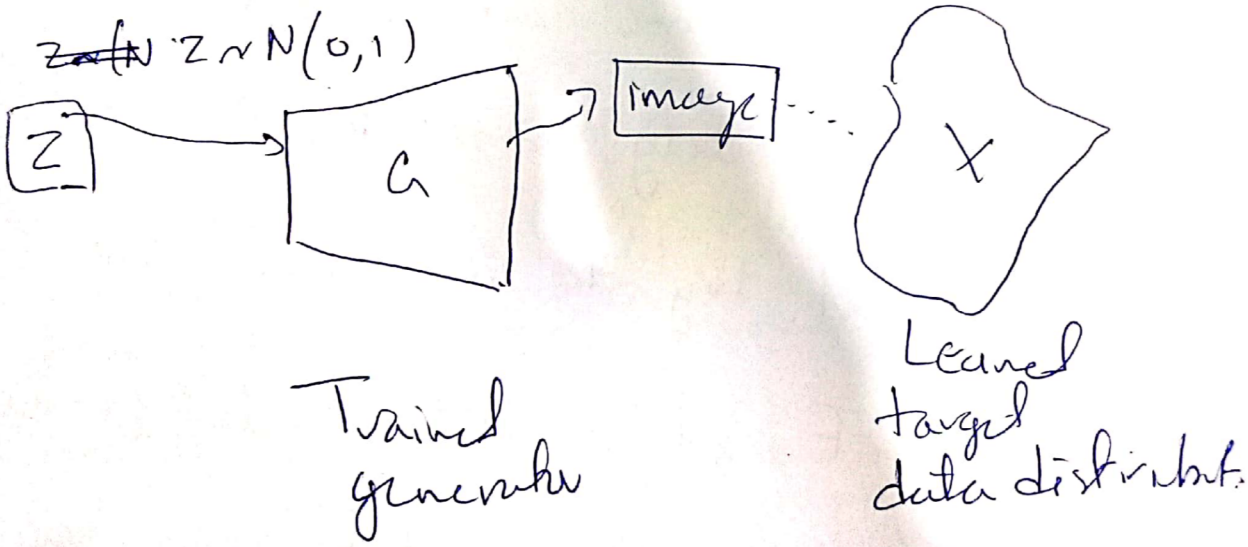
$$\arg\min E_{z,x}\left[\log D(G(z)) + \log(1-D(x))\right]$$

try to synthesize fake instances
that fool the discriminator
D will try to be at best
to discriminate b/w real vs fake

$$\arg\min_G \max_D E_{z,x}\left[\log D(G(z)) + \log(1-D(x))\right]$$

After training G can produce new data instances that have never been seen before.

It is effectively learning a transformation of noise to a target data distribution



different points yields different instance

- traverse and interpolate

synthetic examples