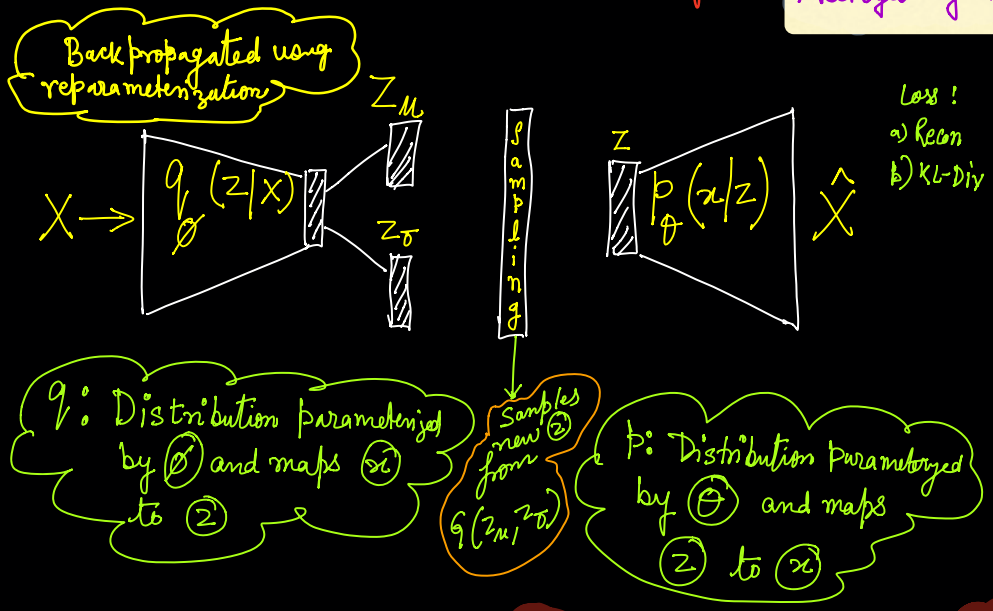
GAN's

Variational Auto Encoder (Summary)

Our Youtube Channel :

Deep learning for all at Manas lab at IIT Mandi.

Aditya Nigam - I.I.T Mandi.



ELBO

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

This need to maximized the ELBO

Maximized

Minimized

Known tractable distribution

$N(0,1)$

But these z's are coming from Encoder hence averaged over $q_\phi(z|x)$

This is the log likelihood of x given z. Maximizing it ensures z should have to generate x.

Minimizing this KL-Div makes $q(z)$ as close to $p(z) = N(0,1)$

(A) Problems with L_2 / MSE loss

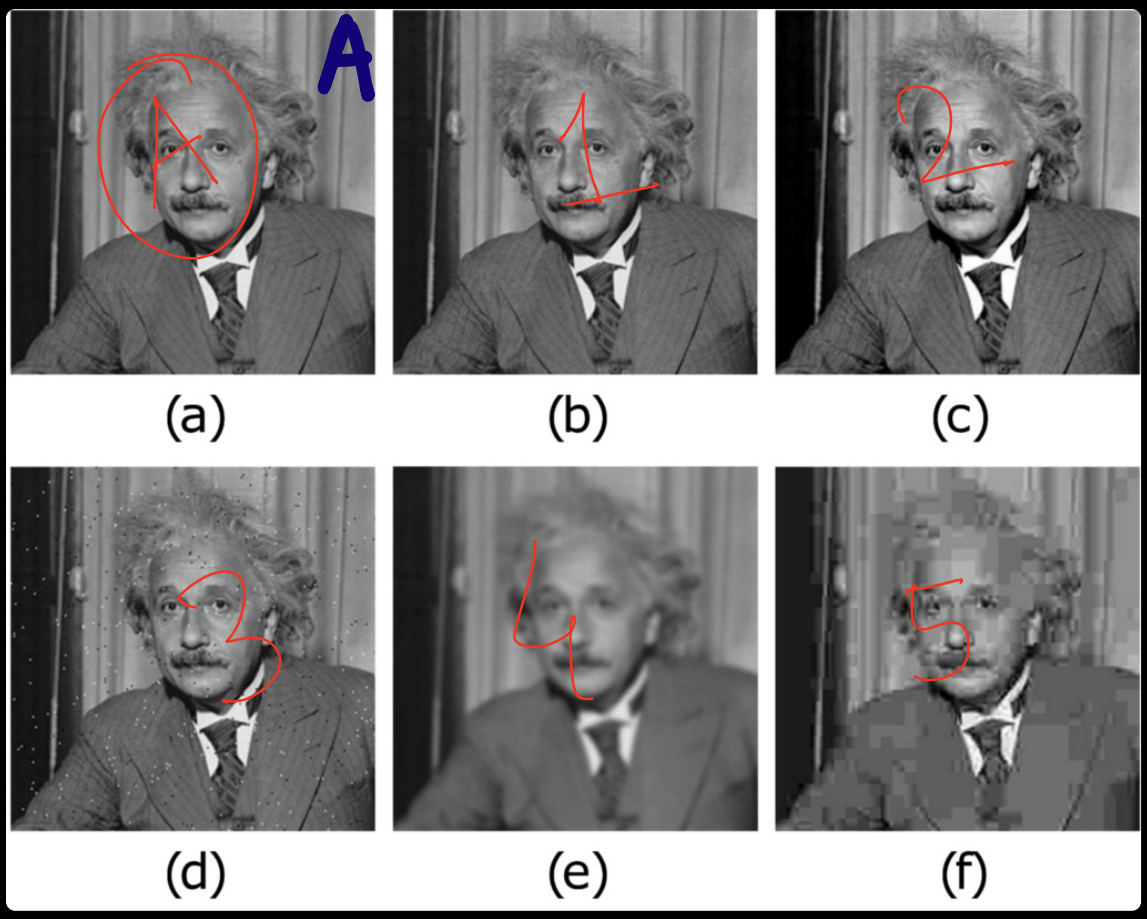


Image (A) is the original image. All the remaining images are transformed images in such a way that the MSE remains to be similar / same.

$$\mathcal{L} = \frac{1}{2 \cdot M \cdot N} \sum_{i}^{M} \sum_{j}^{N} \left(\hat{x}_{ij} - x_{ij} \right)^2$$

(Basic MSE/L_2 loss.)

$$\mathcal{L} \propto \frac{1}{2} \| \hat{x} - x \|$$

Let us look at Gaussian distribution

$$p(x \mid \mu, \sigma^2) = \frac{1}{Z} e^{\frac{-(\mu - x)^2}{2\sigma^2}}$$

Normalization Constant

Mean

(σ^2) Variance

Let us assume $\mu = \hat{x}$ and $\sigma^2 = 1$

$$\therefore p(x|\hat{x}) \propto e^{\frac{-1}{2}\|\hat{x} - x\|_2^2}$$

Taking log both side

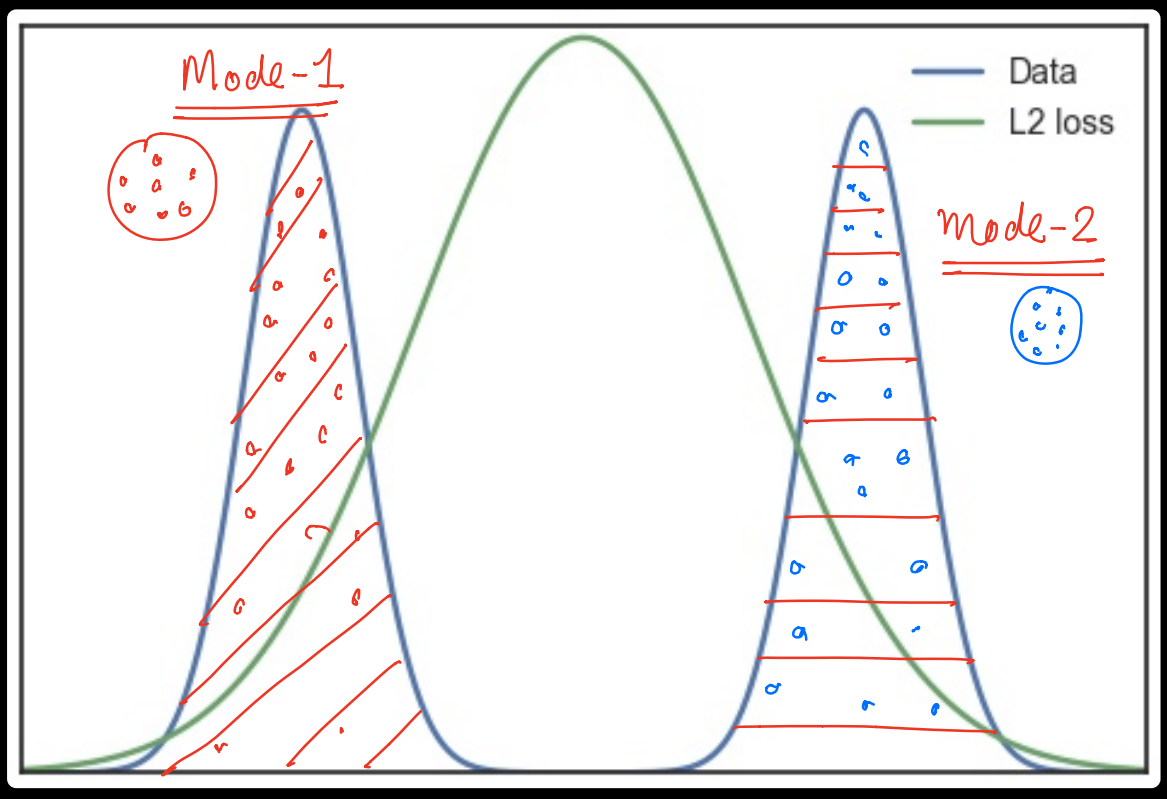
$$\log p(x|\hat{x}) \propto \frac{-1}{2}\|\hat{x} - x\|_2^2$$

$\Rightarrow$ Minimizing L_2/MSE loss is equivalent to maximizing the log-likelihood of Gaussian.

This assumes that our real data is coming from Gaussian distribution. But Gaussian is simple and a single peak distribution may not be able to represent complex image distribution.

In other words: What happen that our data is not following unimodal distribution but a Bi-modal distribution and we try to minimize (L_2) loss. This uses uni-modal gaussian to fit a Bimodal gaussian distribution.

Basically data is coming from 2 modes (Red) and (Blue) $\Longrightarrow$ 2 types of data.



L_2 have to minimize the reconstruction loss for both types of samples during the (optimization) training.

$\Longrightarrow$ Gaussian got Centered right in the middle

$\therefore$ If we sample from Gaussian, sample will come from the middle/mean of both

$\Longrightarrow$ Average of samples of 2 modes

Blurry image.

(B) KL-Div : This is a divergence and not Symmetric

(Issues) [Target] (P): Unknown distribution (we need to estimate)

[Source] (Q): We try to estimate (P) using (Q)

KL-Div ——— KL-Div (Reverse) : $KL(Q \,||\, P) = \sum Q(x) \log \frac{Q(x)}{P(x)}$

This was also utilized as a loss function for VAE.

$$\min\left(KL\left(Q_\phi(z) \,||\, P(z|x)\right)\right)$$

$Q_\phi(z)$: Parameterized by ϕ [Source]

$P(z|x)$: Unknown [Target]

KL-Div (Forward)

$KL(P||Q) = \sum P(x) \log \frac{P(x)}{Q(x)}$

Question: Why do VAE uses Reverse-KL Div not Forward KL-Div.

Forward KL-Div:

$$\min_{\phi} \{ KL(P \| Q_\phi) \}$$

(Unknown) — Estimating (P) via (Q) parameterized using ϕ.

Q ⇒ Tractable and Parameterized. Say unimodal Gaussian.

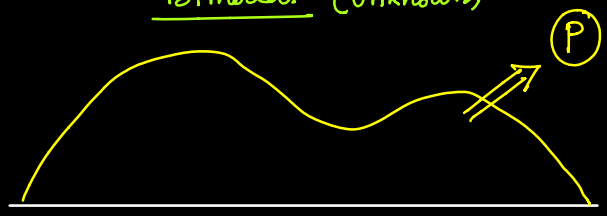
Parameterization of neural network to minimize the objective function.

$$\min_{\phi} \left(\sum_x P(x) \log \frac{P(x)}{Q(x)} \right)$$

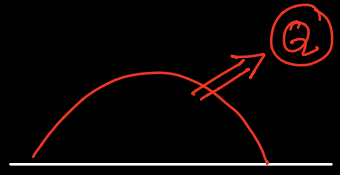
What happen when $Q(x) = 0$ and $P(x) \neq 0$ (some finite value)

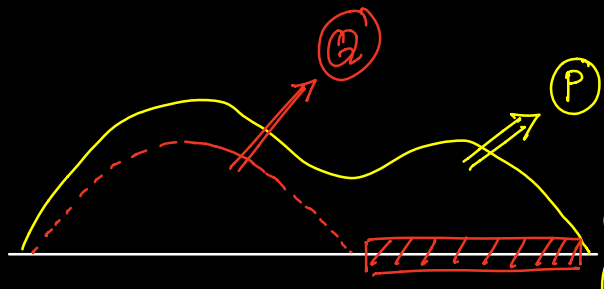
F-KL ⟶ tends to (∞) not defined.

Bimodal (unknown)



Unimodal (assumed)





$$\min_{\phi} [KL(P \| Q_\phi)]$$

⟹ Tends to infinity

Here $Q(x) = 0$ But $P(x) \neq 0$ ⇒ F-KL ⟶ ∞ (Not defined)

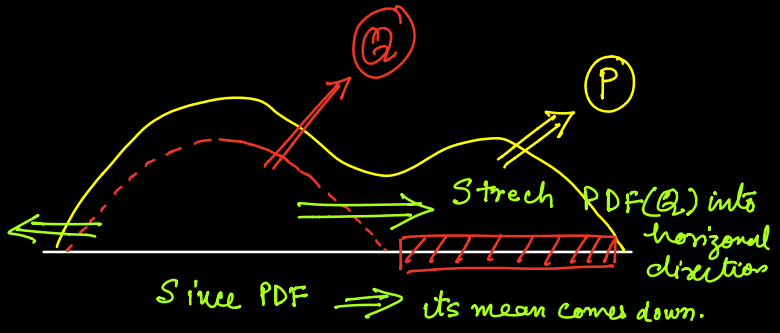
This will never let our neural network (NN) parameterized over ϕ to learn a mode of (P).

Very high (KL) (Unfavorable)

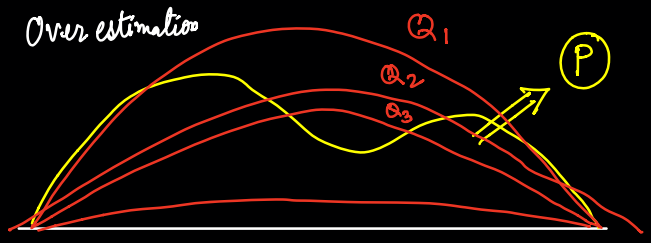
Using F-KLDiv

It avoids such conditions when $Q(x) = 0$ & $P(x) \neq 0$ therefore individual mode learning is avoided.

[Zero avoiding situation]



Since PDF ⟹ its mean comes down.



$\Rightarrow$ few allowed Q_i's (Q_1, Q_2, Q_3)

F-KL allows $P(x) = 0 \Rightarrow Q(x) = 0$

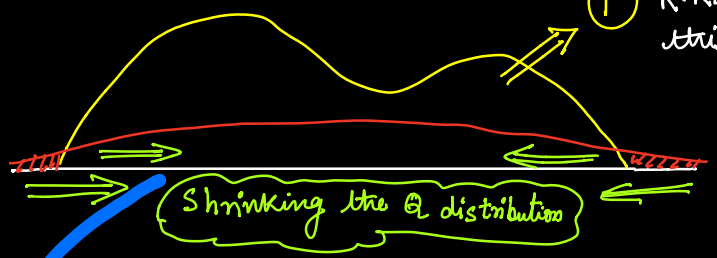
dis allows $Q(x) = 0$ But $P(x) \neq 0$

F-KL [forces (Q_ϕ) to stretch to cover entire (P) distribution] $\rightarrow$ Zero avoiding and Mean seeking.

Reverse KL-Div : $\min_\phi \{ KL(Q_\phi \| P) \}$

$$\min_\phi \left\{ \sum_x Q_\phi(x) \log\left(\frac{Q_\phi(x)}{P(x)}\right) \right\}$$

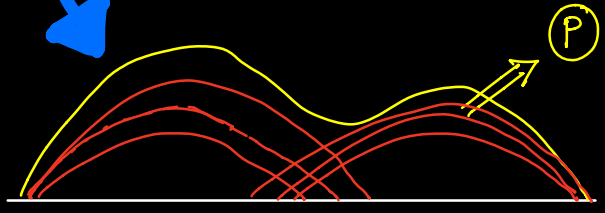
Q_ϕ = non zero and $P(x)$ = zero blow it up



$\Rightarrow$ Avoid long tails. Basically tries to shrink the tail as much as possible.

$\therefore$ F-KL $\Rightarrow$ Stretch Q and support long tail (Mean seeking) [over-estimating] (General average behavior)

R-KL $\Rightarrow$ Shrink Q and avoid long tail (mode seeking) [under-estimating] (estimating only some mode)

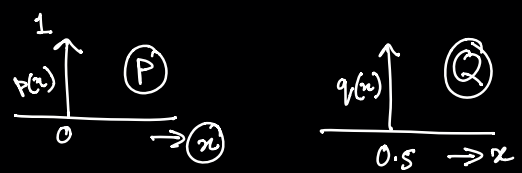


$\Rightarrow$ Mode seeking

Due to this mode seeking behaviour VAE uses R-KL. (F-KL) introduces more and more blur.

But KL-Div suffers from an inherent problem (both F/R).

We have two discrete PDF (P) and (Q).



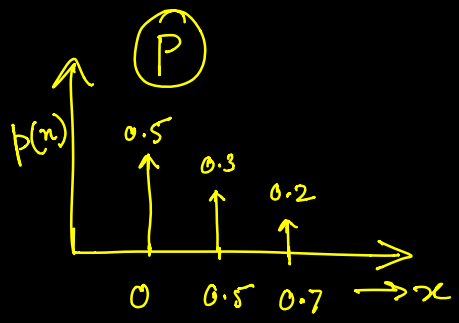
$$KL(P||Q) = P(x=0)\log\frac{P(x=0)}{Q(x=0)} = \infty$$

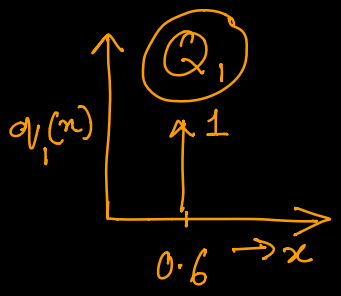
$$KL(Q||P) = Q(x=0.5)\log\frac{Q(x=0.5)}{P(x=0.5)} = \infty$$

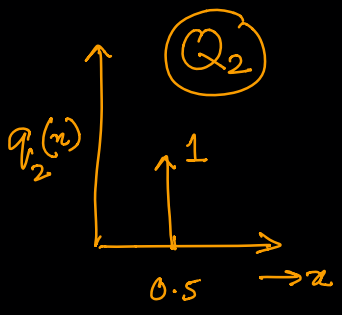
$\Rightarrow$ KL-Div fails to measure when there is NO-OVERLAP.

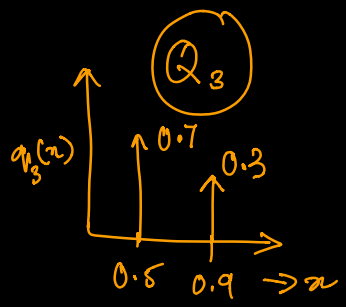
$\Rightarrow$ This is most of the times true especially at the start of the optimization. P $\equiv$ unknown, Q $\equiv$ Tractable / Parameterized

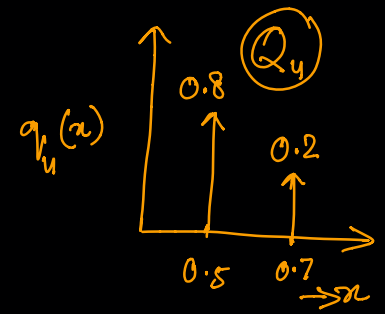
Basically (Q) is very far from the real distribution





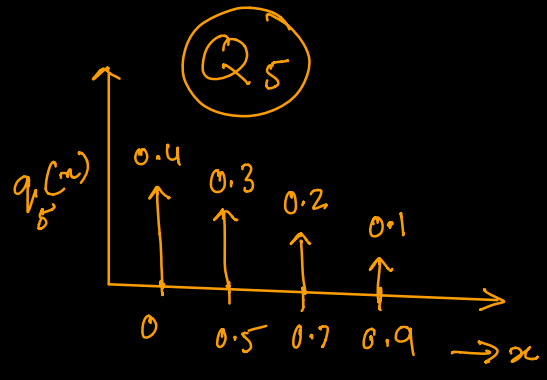


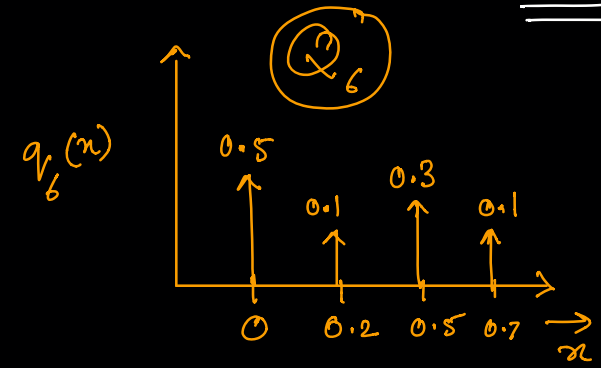




Compute all $KL(P||Q_i)$ & $KL(Q_i||P)$

12 Values





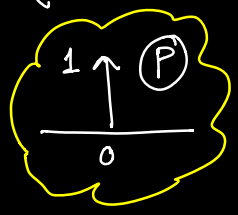
Inference about Containment.

JS- divergence :

$$JSD(P||Q) = \frac{1}{2}\left[KL(P||M) + KL(Q||M)\right]$$

where $M = \left(\frac{P+Q}{2}\right)$

Say for



$$KL(P||M) = \sum P(x) \log \frac{P(x)}{M(x)} = P(x=0) \log \frac{P(x=0)}{\frac{P(x=0)+Q(x=0)}{2}} = 1 \cdot \log \frac{1}{\frac{1}{2}} = \log 2$$

$$KL(Q||M) = \sum Q(x) \log \frac{Q(x)}{M(x)} = Q(x=0.5) \log \frac{Q(x=0.5)}{\frac{P(x=0.5)+Q(x=0.5)}{2}} = 1 \log \frac{1}{\frac{1}{2}} = \log 2$$

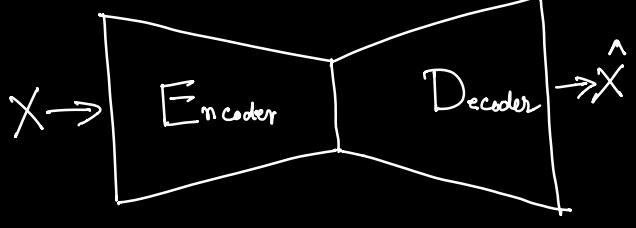
$$\therefore \boxed{JSD(P||Q) = \frac{1}{2}\left[\log 2 + \log 2\right] = \log 2}$$

This behaviour is required when we estimate (P) via (Q) as it rarely happen that (P) & (Q) Overlap right from the starting of optimization

Moving towards generative adversarial networks (GAN).

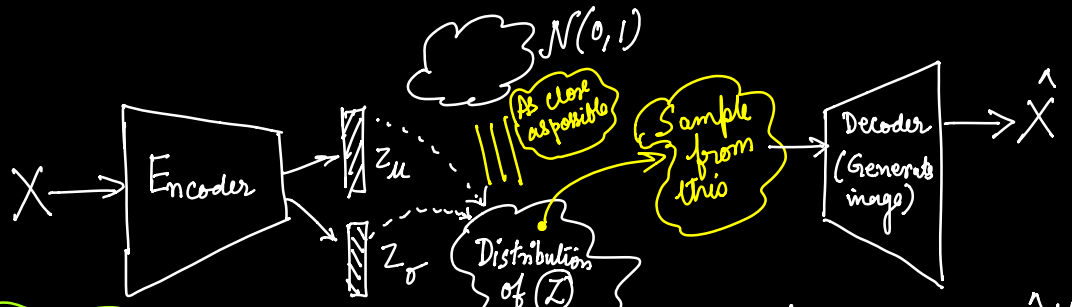
- Instead of MSE they utilizes adversarial training
- Instead of KLD they utilizes JSD.

Let us redraw Basic A.E



Loss : $\|X - \hat{X}\|_2$

VAE

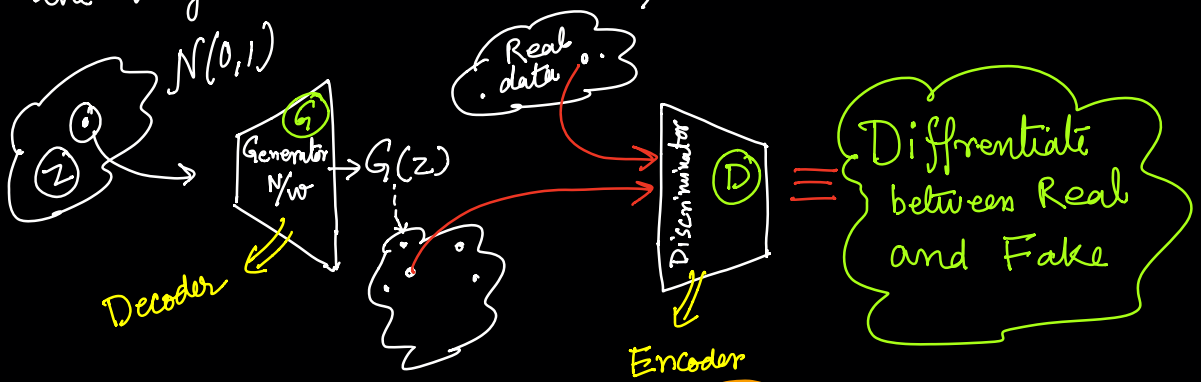


We are learning image (X) distribution via (Z) latent distribution that in turn uses N(0,1). Hence we are implicitly estimating image distribution (X).

$$\text{Loss} \Rightarrow \|X - \hat{X}\| + KLD(G(z_\mu, z_\sigma), N(0,1))$$

GAN [Generative Adversarial Network]

The sequence of Decoder (Generator) and Encoder (Discriminator) got changed. Their objective are also tweaked keeping the overall objective of image reconstruction being fixed. Instead of implicitly learning the image distribution (as in VAE), GAN's explicitly learns it.



Explicitly learning the image distribution

Just trained as a classifier N/w

Generative Adversarial Network (GAN's)

We wanted to generate a distribution that is very close to our dataset's distribution

Using adversarial training which is constant opposition

Deep N/ws

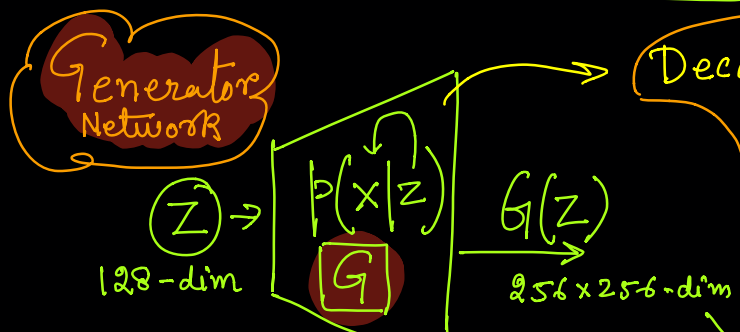
G and D are Competing

We device 2 network generator and discriminator and both of them provide constant opposition to each other

(*) **Adversarial training** (training with an adversarial loss) is essentially required as MSE introduces blurring effect.

Generator network can be seen as a function that takes a random vector (z) as an input and generates G(z) an image.

(z) (say 128-dim) [Essentially a decom-upsampling based NW] G(z) (say 256×256)



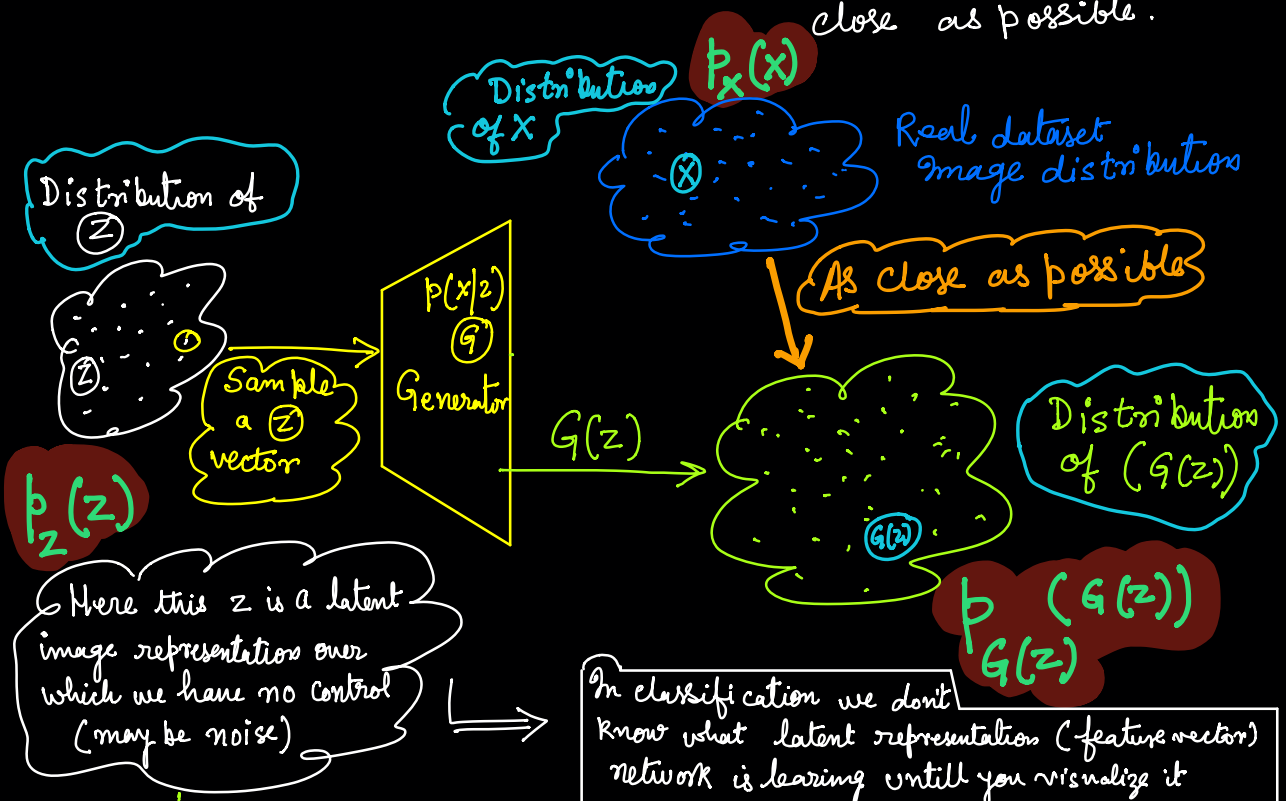
Deconvolution based upsampling network

This G is similar to the decoder network of autoencoder. There it has been trained by MSE loss (reconstruction error)

In autoencoder G is preceded with an encoder type of Network and trained end-to-end by MSE loss

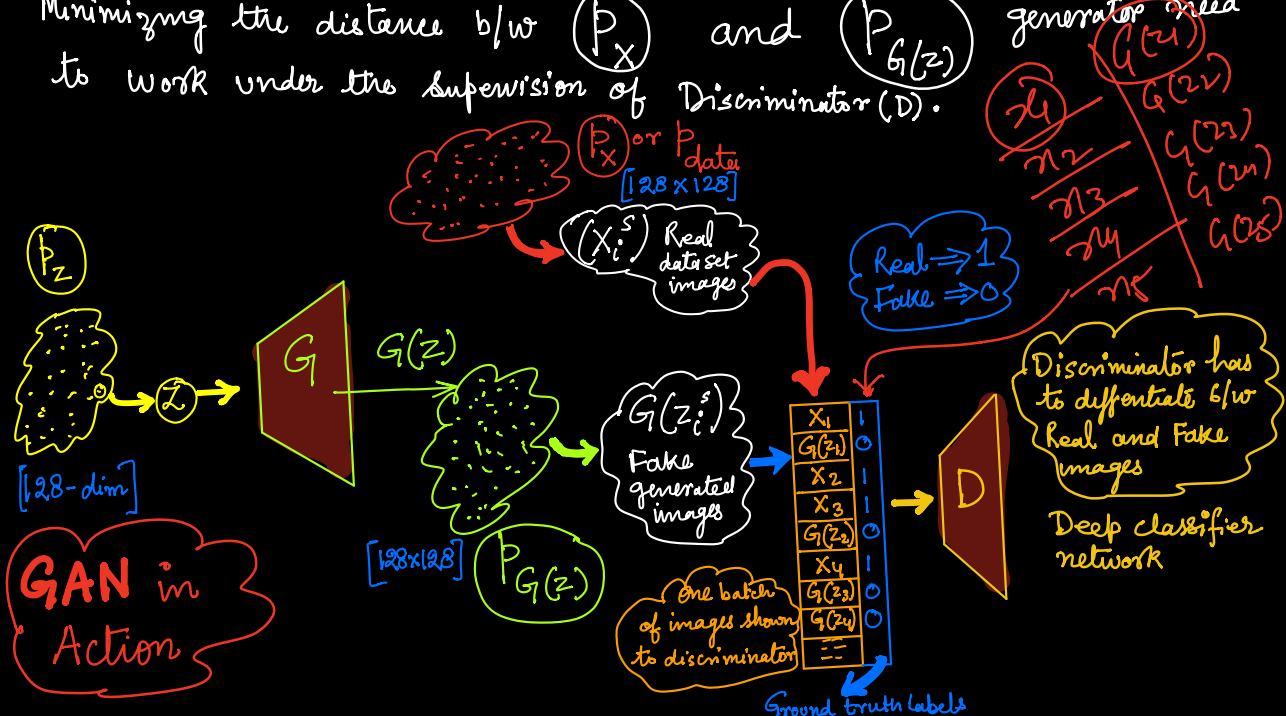
Instead of MSE we train using adversarial loss to avoid blurring.

Objective of Generator network : Given (z) random vector generate a real image as close as possible.



For GAN's, in order to know about (z) generate some image G(z).

Minimizing the distance b/w p_x and $p_{G(z)}$ generator need to work under the supervision of Discriminator (D).



Discriminator network needs to figure out features that contributes to natural images and can help to discriminate real/fake images

While training, the feedback of discriminator network need to be backpropagated to generator

This will help generator (G) to figure out the missing features. Basically (G) is trying to understand that what features current generator is interpreting as real.

Sucess of Discriminator (D) realized as loss to the generator (G) (needs to be minimized)

Adversarial game :

Job of generator (G) is to fool discriminator.

Job of discriminator (D) is to Judge the generator.

The discriminator (D) and generator (G) networks are trained alternatively (one after other), while keep the other one fixed. ⟹ Eventually generator (G) started to make image real wrt (D) trained over the dataset.

Loss function : Binary Cross entropy for classification

$$L(\hat{y}, y) = y \log \hat{y} + (1-y) \log (1-\hat{y})$$

Predicted label

Ground truth label

P	T.P	FN
N	FP	TN

Discriminator loss function : I/P ⇒ Either (X) OR G(z)

[Classifier N/w]

$\hat{y}$ O/P (Label) ⇒ D(X) OR D(G(z))

y Ground Truth (Label) ⇒ 1 (Real) OR 0 (Fake)

∴ $L(D(x), 1) = \log[D(x)]$

D need to maximize D(x)

D(x) maximization
|||
log [D(x)] maximization

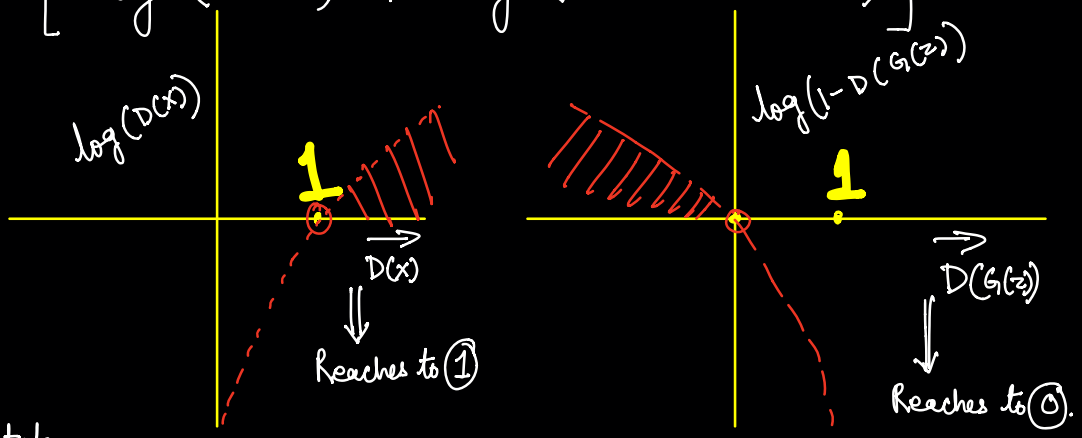
$L(D(G(z)), 0) = \log[1 - D(G(z))]$

D need to minimize D(G(z)) ≡ maximize 1 − D(G(z))

maximize log (1 − D(G(z)))

∴ **Discriminator loss fn** : (Single Sample)

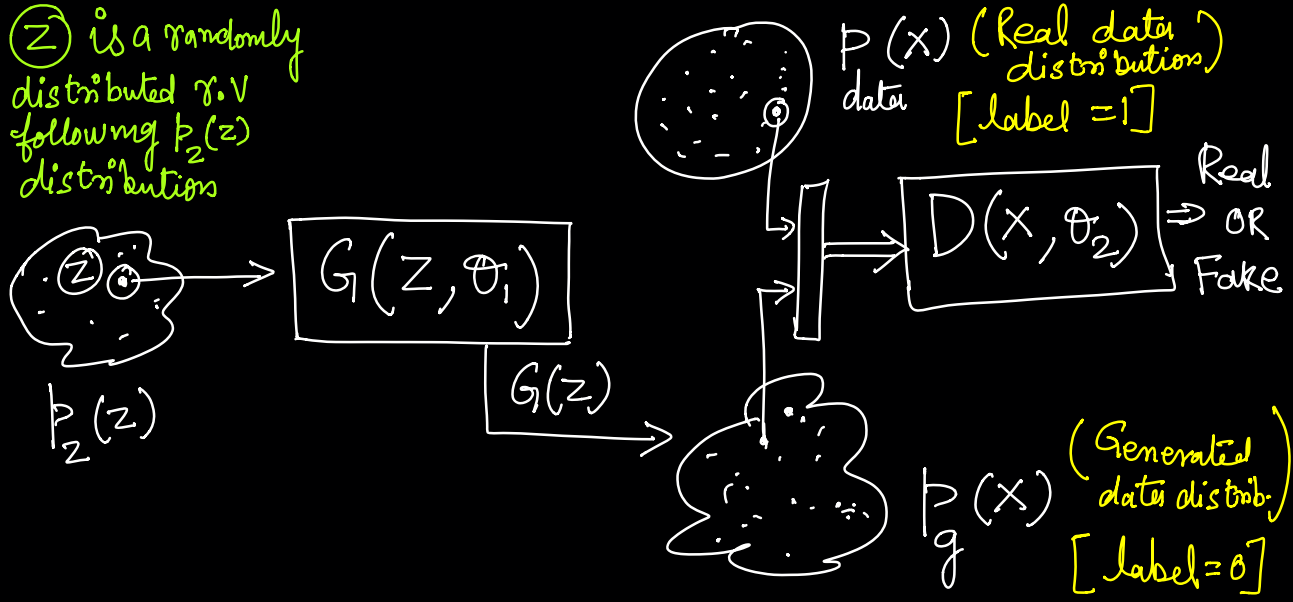
∴ $\max_{D} \left[\log(D(x)) + \log(1 - D(G(z))) \right]$



Over a batch

$$\max_{D} \left[\mathbb{E}_{x \sim p(x)\ \text{data}} (\log(D(x))) + \mathbb{E}_{z \sim p_z(z)} (\log(1 - D(G(z)))) \right]$$

Generator needs to fool Discriminator by generating imags as real as possible hence : $\min_{G}$ $\mathbb{E}_{z \sim p_z(z)}$ $(1 - D(G(z)))$ $(-A)$



Generator loss fn : I/p : $G(z)$

$(\hat{y})$ O/p : $D(G(z))$

(y) G/T : 0

∴ From binary cross entropy (Single Sample)

$$L(D(G(z)), 0) = \log(1 - D(G(z)))$$

(over a batch) $\Rightarrow \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$.

Since (G) is never going to see any real data, just for completeness one can write Generators complete loss fn as.

$$\min_{G} \left[\mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log(D(x))] + \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \right]$$

Total Combined Loss Fn :

$$\min_G \max_D V(D, G) =$$

$$\min_G \max_D \left(\mathbb{E}_{x \sim p(x)} [\log(D(x))] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \right)$$

Real data (data) — Random vector — Image generated by Random vector

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

10-June 2014

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Point-01 : First we will show that given any fixed (G), the optimally achieved D_G^* can be defined as :

$$D_G^*(x) = \arg\max_{(D)} V(D, G) \quad \text{(G fixed; Best possible (D))}$$

$$= \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Point-02 : Secondly for any given (say best (D_G^*)) The optimal G^* can be defined as:

$$G^* = \arg\min_{(G)} V(D_G^*, G)$$

(D_G^* fixed; to get best generator.)

⇒ It is been shown that this optimization have a unique solution achieved when $\boxed{p_g = p_{data}}$ desirable form the problem itself

⇒ Also it has been shown that it is equivalent to minimize $JSD(p_{data} \,||\, p_g)$.

Proving point-01 : For a given (G)

Our objective is to Get the optimal (D).

$$\therefore \quad D_G^* = \arg\max_{(D)} V(D, G)$$

Best possible (D) for a given (G)

$$\therefore D_G^* = \arg\max_{(D)} \left\{ \mathbb{E}_{x \sim p(x)_{data}} [\log(D(x))] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \right\}$$

$$\mathbb{E}_{p(x)}[X] = \int_x x \, p(x) \, dx$$

Expected value of R.V(x), following PDF $\Rightarrow p(x)$

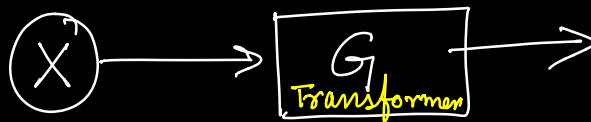
Hence $V(G, D)$:

$$V(G, D) = \underbrace{\int_x p(x)_{data} \cdot \log(D(x)) \, dx}_{(A)} + \underbrace{\int_z p_z(z) \cdot \log(1 - D(G(z))) \, dz}_{(B)} \qquad \text{Eq-01}$$

Random Variable Transformations :

Assuming invertible

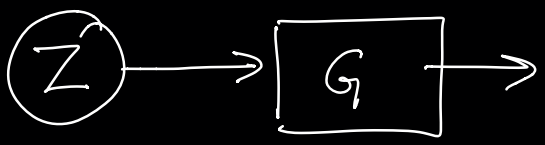
$X = G^{-1}(Y)$



$p_X(x)$ (Given)

$$\therefore \quad p_Y(y) = p_X\left(G^{-1}(y)\right) \cdot \frac{d}{dy}\left(G^{-1}(y)\right) \quad -(i)$$

Now Consider the case of GAN's.



$p_z(z)$

Using (i)

$$p_g(x) = p_z\left(G^{-1}(x)\right) \cdot \frac{d}{dx}\left(G^{-1}(x)\right) \quad -(ii)$$

(Now let us see the total derivation)

$V(G, D)$:

$$V(G,D) = \int_x \underbrace{p_{data}(x) \cdot \log\left(D(x)\right) dx}_{(A)} + \int_z \underbrace{p_z(z) \cdot \log\left(1 - D(G(z))\right) dz}_{(B)} \qquad \text{Eq-01}$$

Term (B):

$$\int_z p_z(z) \log(1 - D(G(z)))\, dz$$

Changing of variable

Let $z = G^{-1}(x) \quad \boxed{x = G(z)}$

$$\therefore dz = \frac{d(G^{-1}(x))}{dx} \cdot dx$$

$$\therefore \int_x p_z(G^{-1}(x)) \cdot \log(1 - D(x)) \cdot \frac{d(G^{-1}(x))}{dx} dx$$

$p_z(G^{-1}(x))$ and $\frac{d(G^{-1}(x))}{dx}\,dx$ $\Rightarrow$ $p_g(x)$ (using (ii))

$$\boxed{\int_x p_g(x) \log(1 - D(x))\, dx}$$

Hence finally $V(G, D)$:

$$\int_x p_{data}(x) \log(D(x))\, dx + \int_x p_g(x) \log(1 - D(x))\, dx$$

(only in x)

$$D_G^*(x) = \arg\max_{D} [V(D, G)]$$

optimal; fixed

Hence in order to maximize $V(D, G)$ we need to differentiate it wrt $[D(x)]$.

$$\therefore \frac{d(V(D,G))}{d[D(x)]} = \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0$$

$$\therefore \boxed{D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}}$$

Practically Cannot be Computed but analytically proved.

Now the role of Generator is reverse of (D). Hence for this best an optimally choosen (D_G^*), we need to obtain the optimal (G^*) by minimizing the loss fn $V(D, G)$ over all possible (G)'s for fixed (D_G^*)

$$\therefore G^* = \underset{G}{\operatorname{argmin}}\, V(D_G^*, G)$$

$$G^* = \arg\min_G \left\{ \int_x p_{data}(x) \log\left(D_G^*(x)\right) dx + \int_x p_g(x) \log\left(1 - D_G^*(x)\right) dx \right\}$$

$$= \arg\min_G \left\{ \int_x \left(p_{data}(x) \cdot \log\left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) + p_g(x) \cdot \log\left(\frac{p_g(x)}{p_{data}(x) + p_g(x)} \right) \right) dx \right\}$$

$$+ \log(2) \cdot p_{data}(x) - \log(2) \cdot p_{data}(x)$$

$$+ \log(2) \cdot p_g(x) - \log(2) \cdot p_g(x)$$

they got Combined

$$= \arg\min_{G} \left\{ \int_x \left(-\log(2)\left(p_{data}(x) + p_g(x)\right) \right.\right.$$

$$+ \; p_{data}(x) \cdot \log \frac{2 \cdot p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$\left.\left. + \; p_g(x) \cdot \log \frac{2 \cdot p_g(x)}{p_{data}(x) + p_g(x)} \right) dx \right.$$

$$\Rightarrow \int_x p_g(x) = \int_x p_{data}(x) = 1$$

$$\therefore G^* = \arg\min_G \left[\int_x \left(-\log(2) \cdot (2) \; + \right.\right.$$

$$p_{data}(x) \log \left(\frac{2 \cdot p_{data}(x)}{p_{data}(x) + p_g(x)} \right)$$

$$\left.\left. + \; p_g(x) \log \left(\frac{2 \cdot p_g(x)}{p_{data}(x) + p_g(x)} \right) \right) dx \right]$$

Hence

$$G^* = \arg\min_G \left[-\log(4) + \int_x p_{data}(x) \cdot \log\left(\frac{2 \cdot p_{data}(x)}{p_{data}(x) + p_g(x)} \right) dx \right.$$

$$\left. + \int_x p_g(x) \cdot \log\left(\frac{2 \cdot p_g(x)}{p_{data}(x) + p_g(x)} \right) dx \right]$$

$$= \arg\min_G \left[-\log(4) + 2 \cdot JSD\left(p_{data}(x), p_g(x) \right) \right]$$

This shows that G^* Optimization is equivalent to JSD minimization between $p_{data}(x)$, $p_g(x)$

$\Longrightarrow$ This implies that minimum JSD can be equal to (zero) that too when $\boxed{p_g(x) = p_{data}(x)}$

∴ After the above minimization Convergence,

$\boxed{JSD \to 0}$ $\boxed{p_g(x) \to p_{data}(x)}$

and $\boxed{V(D, G) \to -\log(4).}$

This shows that :

The global minima of the Criterion fn $V(D_G^*, G)$ is achieved iff $(p_g = p_{data})$ and the loss will be equal to $[-\log(4)]$

Hence the goal of GAN's is to generate the optimal (G^*) which is effectively done by minimizing the $JSD(p_{data}, p_g)$ and Converges to $\boxed{p_g = p_{data}}$.

$\therefore$ The Corresponding $D_G^* = \frac{p_{data}}{p_{data} + p_g} = \frac{1}{2}$ (loss) Can be

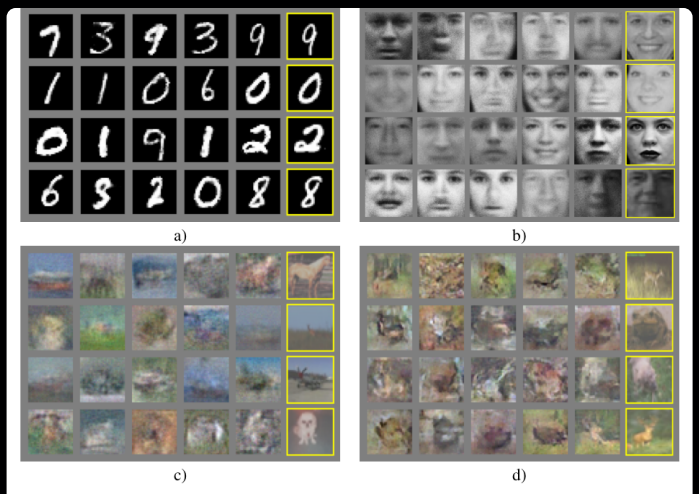


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

Problems/Challenges/Limitations of the Current GAN models [AK/JH]

(I) Diminishing Gradients

(*) Intially (G) is generating very poor samples and (D) will actually be trained fast. Training of (G) is also very slow. (x & $G(z)$ are very diffient and $D(x)$ & $D(G(z))$ → (1) Easy → (0))

(G) is learning from (D)'s mistakes that getting rarer

Loss for (D)

$$\max_D \left\{ \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z))) \right\}$$

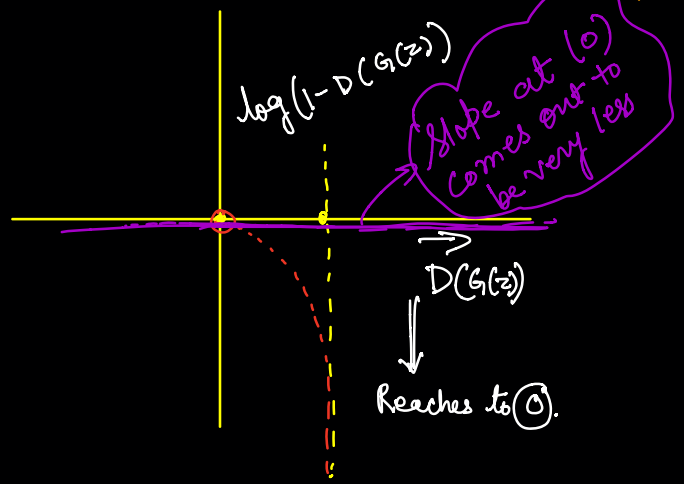
- x: Real
- (D) tries to make it close to (1)
- z: Random vector
- $G(z)$: Fake generator image
- (D) makes (0)
- (G) makes (1)

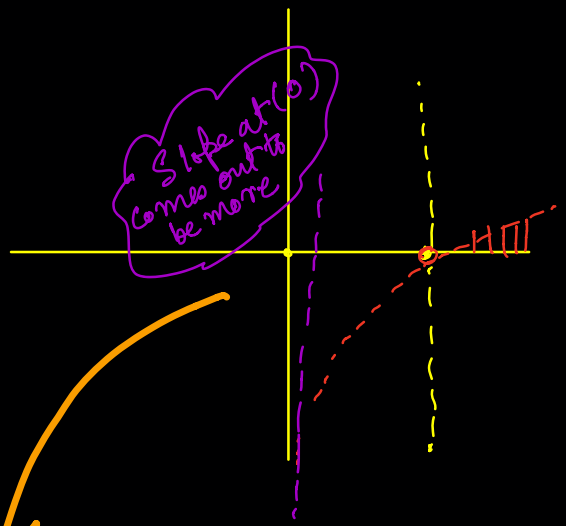
Loss for (G)

$$\min_G \left\{ \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z))) \right\}$$

(A) Early on the generator network training is very slow

Hence we need to tweak the loss so as to manage the derivate early on.





Equivalent to

$$\max_G \left\{ \mathbb{E}_{z \sim p_z(z)} \log[D(G(z))] \right\}$$

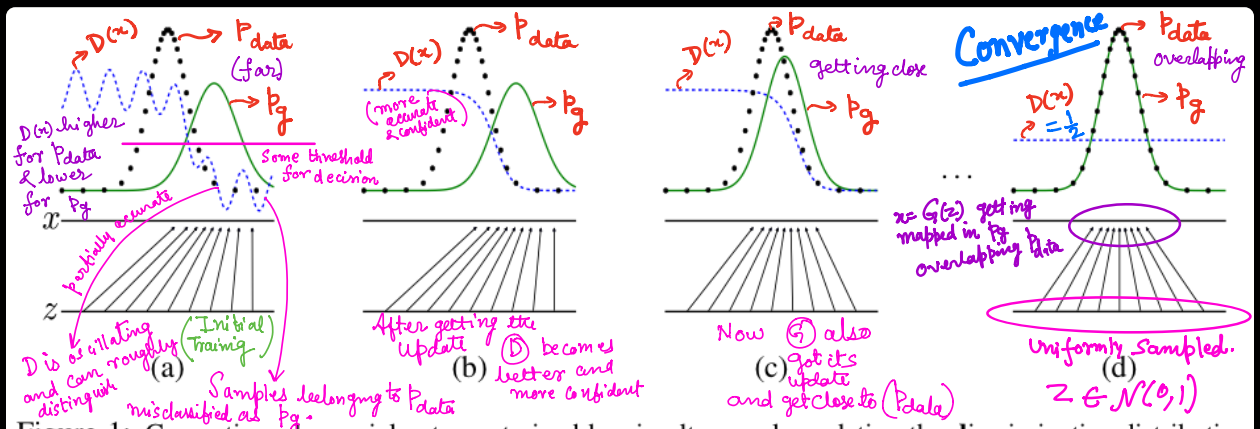


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.