- Gibbs sampling
- VAEs
- GANs

13/04/2024

Tutorial On Generative Model

By – Mohd Ayyoob (Mtech CSA)

Gibbs sampling ✓

$P(X, Y)$ ① $\rightarrow$ $P(X, Y)$ — not access
— highly complicated

$\underbrace{P(X|Y), \; P(Y|X)}$ $\rightarrow$ ✓

$P(X=0, Y=0) = 0.3$

(x_t, y_t)

$X: \{0,1\}$ Algo:

$Y: \{0,1\}$

	X: 0	X: 1
Y: 0	✓	✓
Y: 1		

$P(X,Y)$

$(x_0, y_0) \sim \pi(\;\;)$

for t in rang(T):

$\quad x_t \sim P(X | Y_{t-1})$

$\quad Y_t \sim P(Y | X_t)$

$\rightarrow P(X, Y)$

$\{(0,0)\ (0,1)\ (1,0)\}$

$\longrightarrow P(X,Y)$

Y \ X	0	1	
0	0.1	0.4	0.5
1	0.3	0.2	

$(0,1) \sim 0.35$

0.1	0.2	0.3	0.4

$(0,1)$

$P(X|Y)$

Y \ X	0	1
0	$\frac{1}{5}$	$\frac{4}{5}$
1	$\frac{3}{5}$	$\frac{2}{5}$

$P(X|Y=0)$

$P(X|Y=1)$

$P(Y|X)$

Y \ X	0	1
0	$\frac{1}{4}$	$\frac{2}{3}$
1	$\frac{3}{4}$	$\frac{1}{3}$

z — latent variable

$z^i \sim p_\theta(z)$ prior $(z) \sim N(0, I)$

$\boxed{\text{intractable}}$

① first sample z

② sample x given z

$$\left\{\begin{array}{l} z^i \sim p_\theta(z) \\ x^i \sim p_\theta(x \mid z = z^i) \end{array}\right.$$

intractable

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^i) \rightarrow \int p_\theta(x^i|z)\, p_\theta(z) \cdot dz$$

VAE
GAN $\Big\}$ generate samples

gibbs sampling ✓

Variational Auto encoders

Latent variable modeling

$$x \longrightarrow y$$

$$x \longrightarrow z \longrightarrow y$$

Normal Auto-Encoder

→ identity function → unsupervised manner

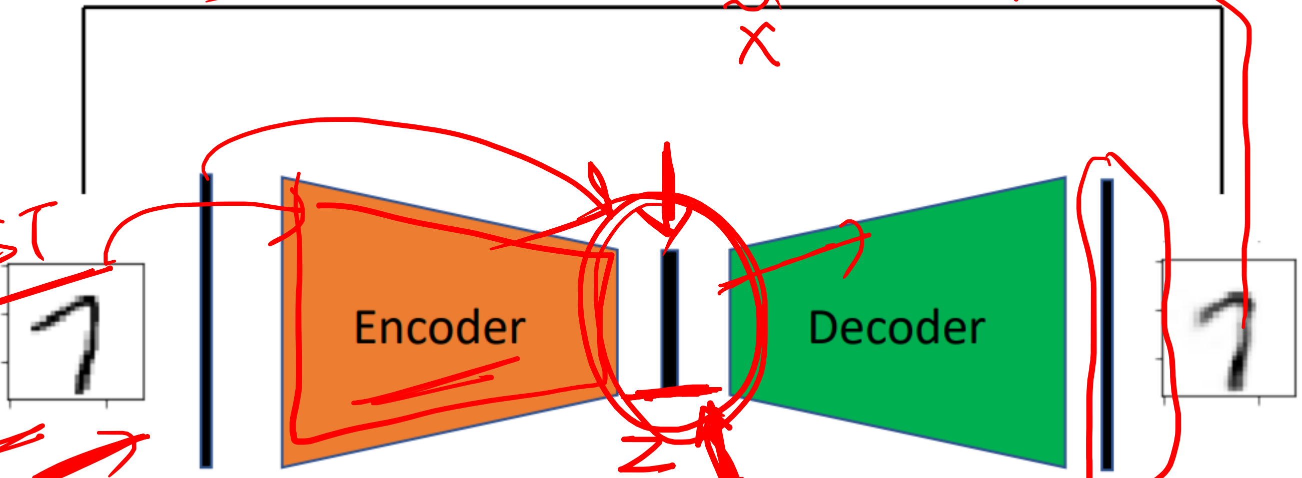
Relatiⁿ b/w Autoencoders & PCA

Minimize squared error loss:

$$\mathcal{L} = ||\mathbf{x} - \underbrace{Dec(\overset{z}{Enc}(\mathbf{x}))}_{\tilde{x}}||_2^2$$

MSE

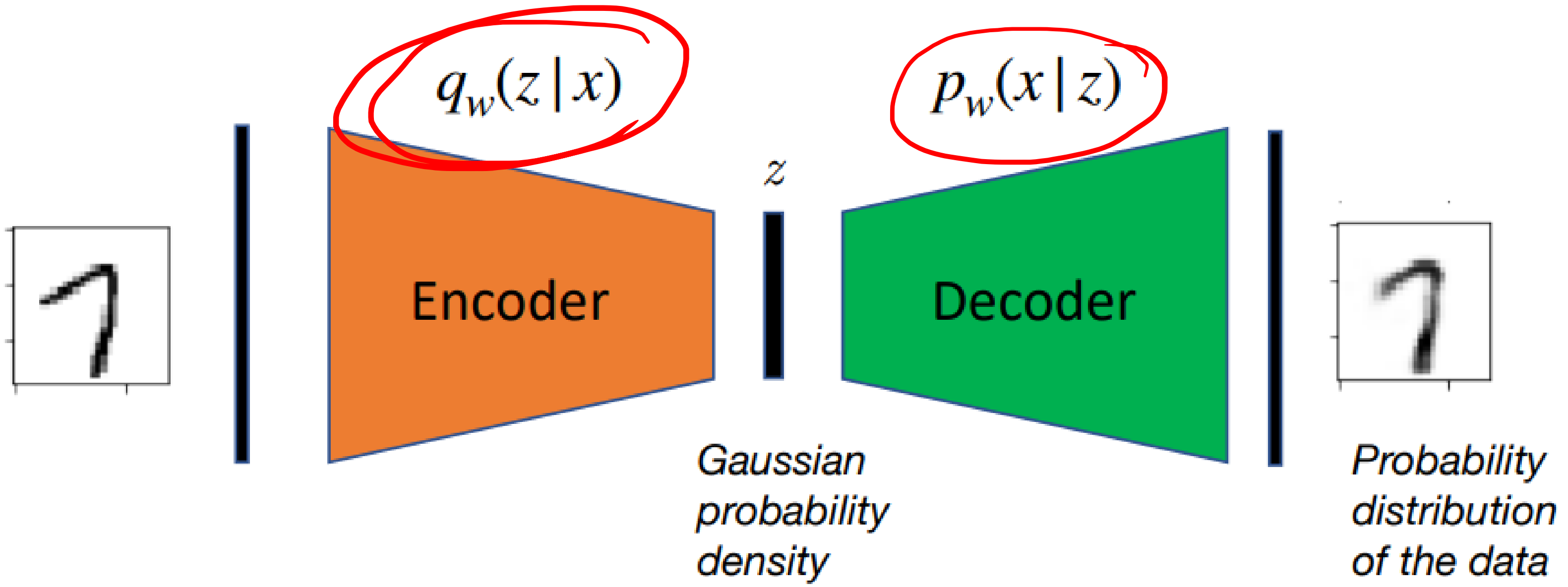
MNIST



z

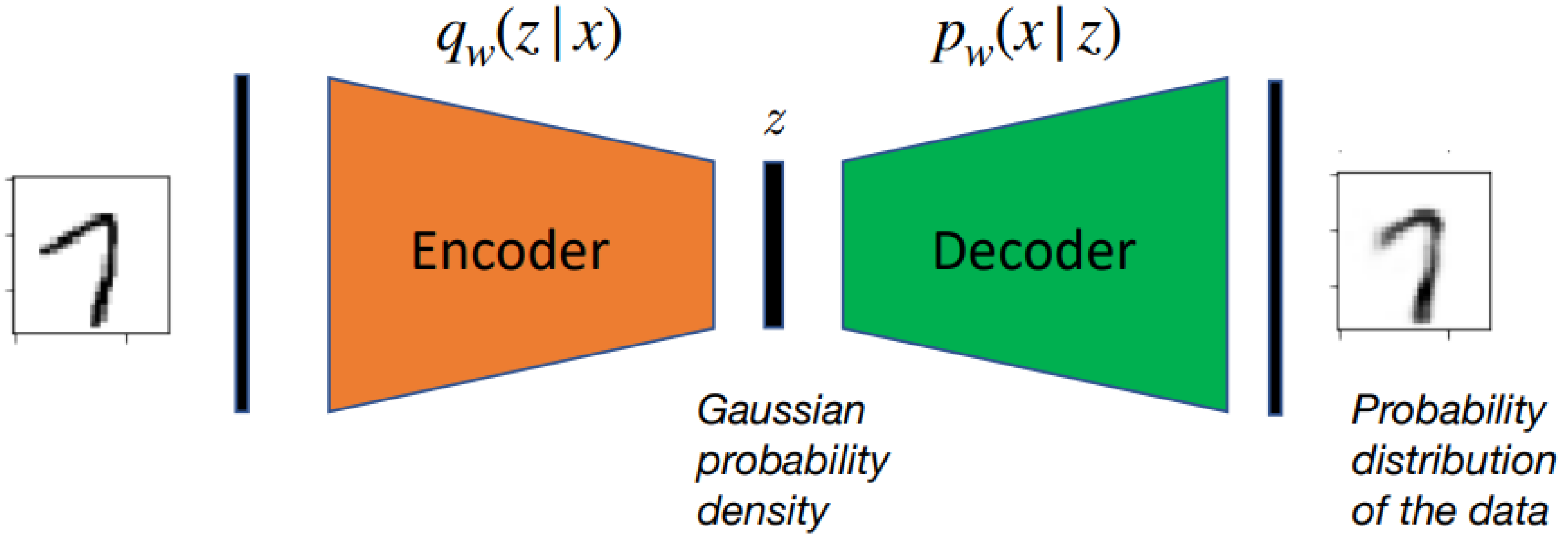
(hidden)

Variational Auto-Encoder



Variational Auto-Encoder

$$\mathcal{L} = -\mathbb{E}_{z\sim q_w(z|x^{[i]})}\left[\log p_w\left(x^{[i]}|z\right)\right] + \mathbf{KL}\left(q_w\left(z|x^{[i]}\right)\|p(z)\right)$$

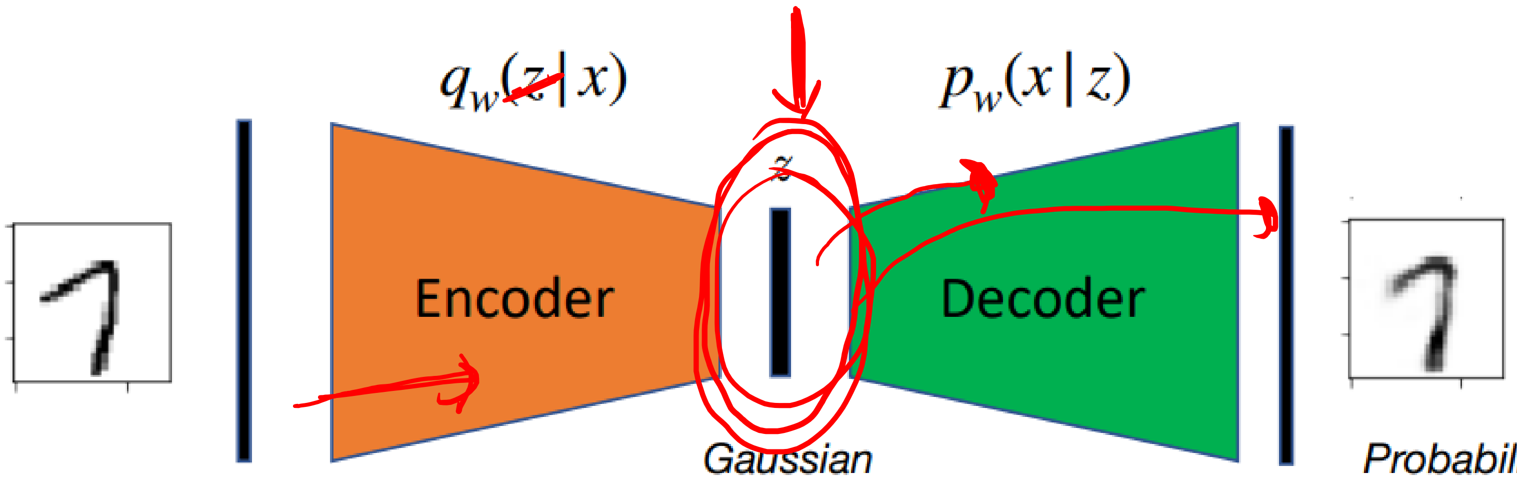


Variational Auto-Encoder

reconstruction loss

$$\mathcal{L} = -\overbrace{\mathbb{E}_{z \sim q_w(z \mid x^{[i]})} \left[\log p_w\left(x^{[i]} \mid z\right)\right]}^{\text{reconstruction loss}} + \mathbf{KL}\left(q_w\left(z \mid x^{[i]}\right) \| p(z)\right)$$

Expected neg. log likelihood
term; wrt to encoder distribution



Gaussian probability density

Probability distribution of the data

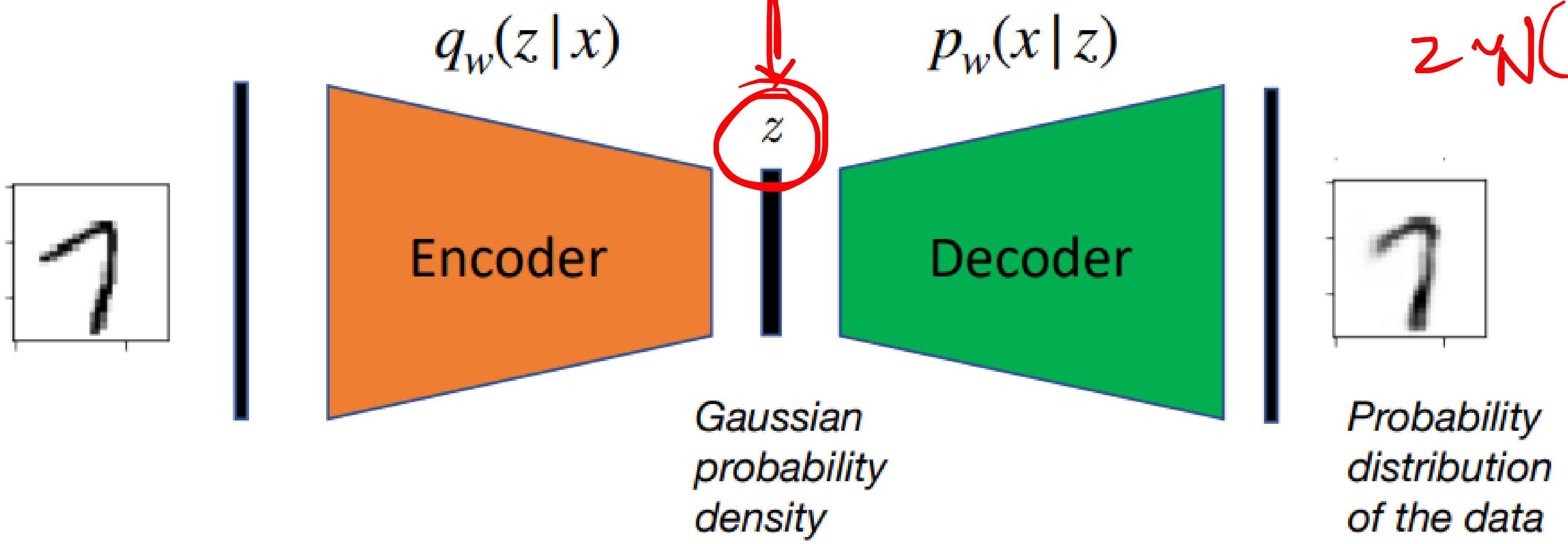
Variational Auto-Encoder

$$\mathcal{L} = -\mathbb{E}_{z \sim q_w(z \mid x^{[i]})}\left[\log p_w\left(x^{[i]} \mid z\right)\right] + \mathbf{KL}\left(q_w\left(z \mid x^{[i]}\right) \| p(z)\right)$$

Expected neg. log likelihood term; wrt to encoder distribution

Kullback-Leibler divergence term where $p(z) = \mathcal{N}\left(\mu = 0, \sigma^2 = 1\right)$

$N(0, I)$



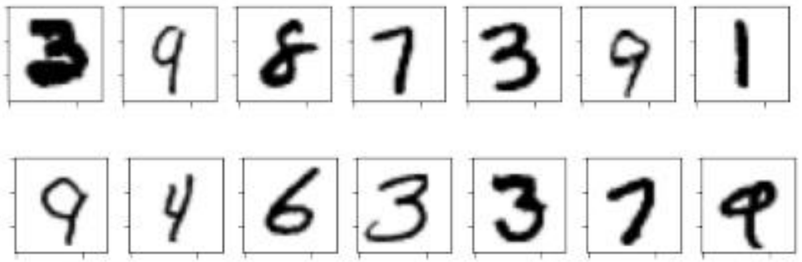
$z \sim N(0, I)$

$N(0, I)$

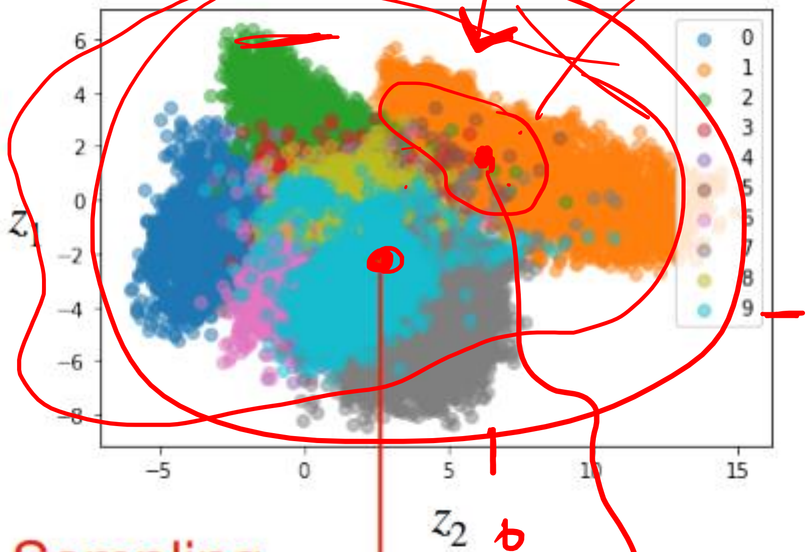
Sampling

Sampling from Normal Auto Encoders

(5,4) N(0,I)



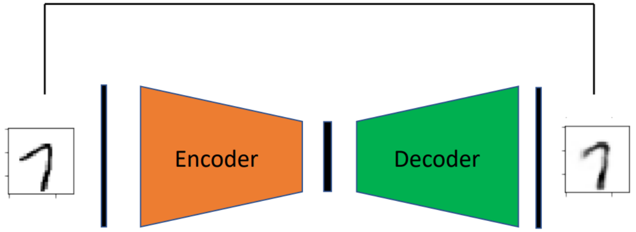
Encoding



Sampling
& Decoding

Minimize squared error loss:

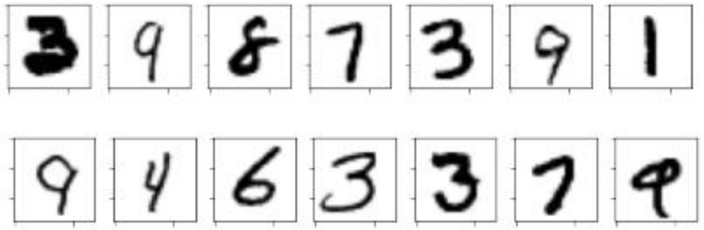
$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2$$



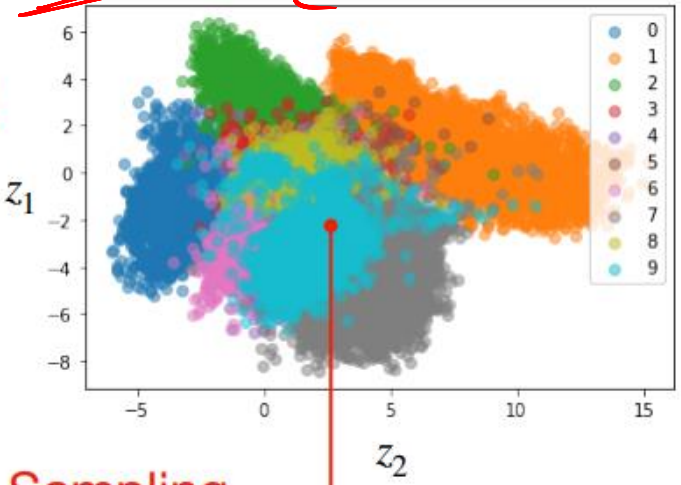
Sampling from Normal Auto Encoders

2D

$z \sim N(0, I)$

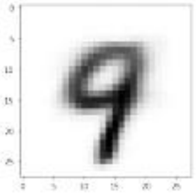


Encoding



Sampling
& Decoding

$z \sim N(0, I)$



Challenge: regular autoencoders are difficult to sample from, because

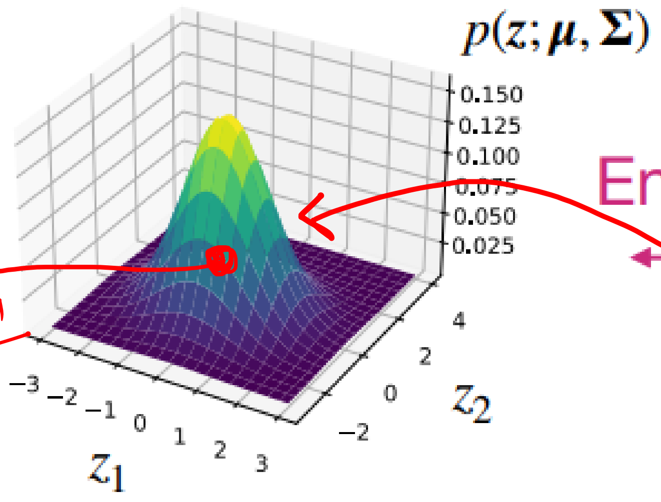
1. oddly shaped distribution, hard to sample in a balanced way
2. distribution not centered at (0, 0)
3. distribution not necessarily continuous (hard to see here in 2D, but a big problem in higher dimensional latent spaces)

z

$N(0, I)$

Sampling from Variational Auto Encoders

2



decoder

Encoding

2D

$$\begin{array}{ccc} & \sigma_1 & \sigma_2 \\ \sigma_1 & 1 & 0 \\ \sigma_2 & 0 & 1 \end{array}$$

d-dimensional probability density for multivariate Gaussian

$$p(z;\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(z-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(z-\boldsymbol{\mu})\right)$$

$$Z \sim \mathcal{N}(0,\boldsymbol{I})$$

with $\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$

Sampling from Variational Auto Encoders

Reparamaterization

Input

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

Where $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$

$$\Sigma \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

F.P FP

$\epsilon_1, \epsilon_2 \sim N(0,1)$

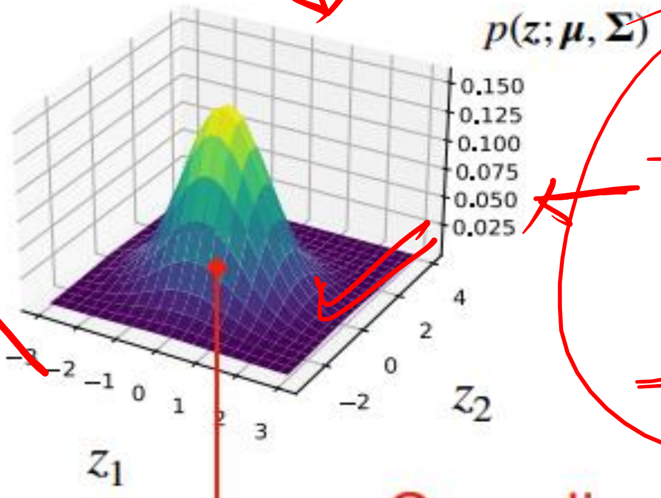
$z = \mu + \sigma \epsilon$

$N(0,1)$

No Covariance Matrix needed , as it assume diagonal covariance

Thus, we only need a mean and a variance vector, no covariance matrix

$p_q(x|z)$

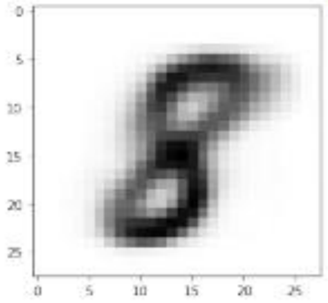


$N(\mu, \sigma)$

$\approx$

$N(0, I)$

Sampling & Decoding



Sampling from Variational Auto Encoders

$$z = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$$

Sampled from standard multivariate normal distribution in each forward pass

Where $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$

$$\epsilon_1, \epsilon_2 \sim N(0,1)$$

But why ϵ? Continuous distribution; VAE must ensure that points in neighborhood encode the same image so that when decoding they produce the same image

Think of these as parameter vectors included in training & backpropagation

Sampling from Variational Auto Encoders

Instead of using a variance vector, $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$ NN

we use the
<u>log-var vector</u>
to allow for positive and negative values: $\log(\boldsymbol{\sigma}^2)$

Why can we do this?

$$\log(\sigma^2) = 2 \cdot \log(\sigma)$$

$$\log(\sigma^2)/2 = \log(\sigma)$$

$$\sigma = e^{\log(\sigma^2)/2}$$

$z = \mu + \sigma \cdot \epsilon$

So, when we sample the points, we can do

$$\boxed{z = \mu + e^{\log(\sigma^2)/2} \cdot \epsilon}$$

Loss Calculation

1) Minimize squared error loss: (ensures good reconstruction)

$$\mathcal{L}_1 = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2 = \sum_{i=1}^{d} (x_i - x_i')^2$$

$\boxed{z \sim N(0, I)}$

2) Minimize KL divergence: (ensures latent space is continuous and standard normal distributed)

Derivation

$$\mathcal{L}_2 = D_{KL}\left[N(\mu, \sigma) || N(0, 1)\right] = -\frac{1}{2} \sum \left(1 + \log\left(\sigma^2\right) - \mu^2 - \sigma^2\right)$$

Overall loss: $\mathcal{L} = \alpha \cdot \mathcal{L}_1 + \mathcal{L}_2$

Generative Adverserial Networks

$P(X)$ $\quad X : (x_1\ x_2\ x_j \ldots)$

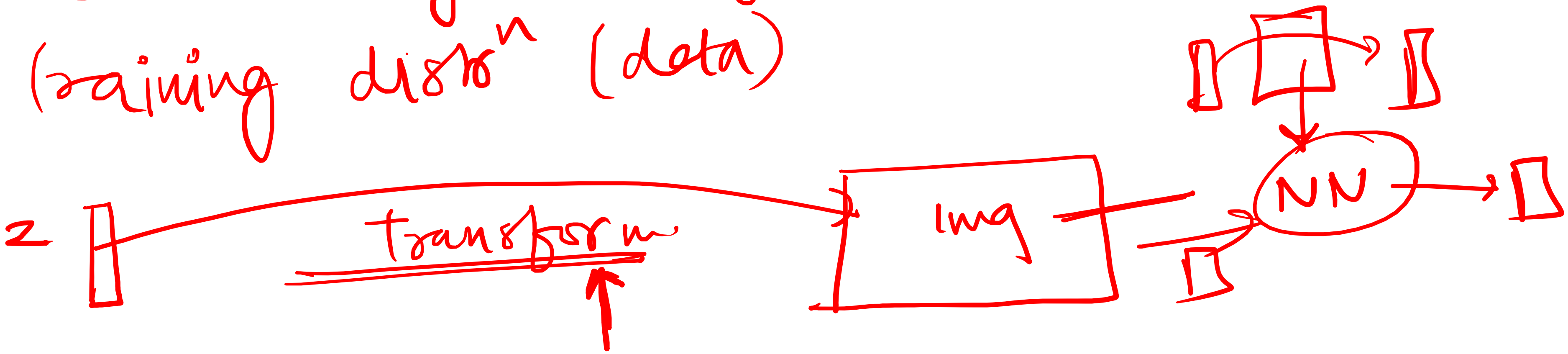
VAE and $p(z|x)$

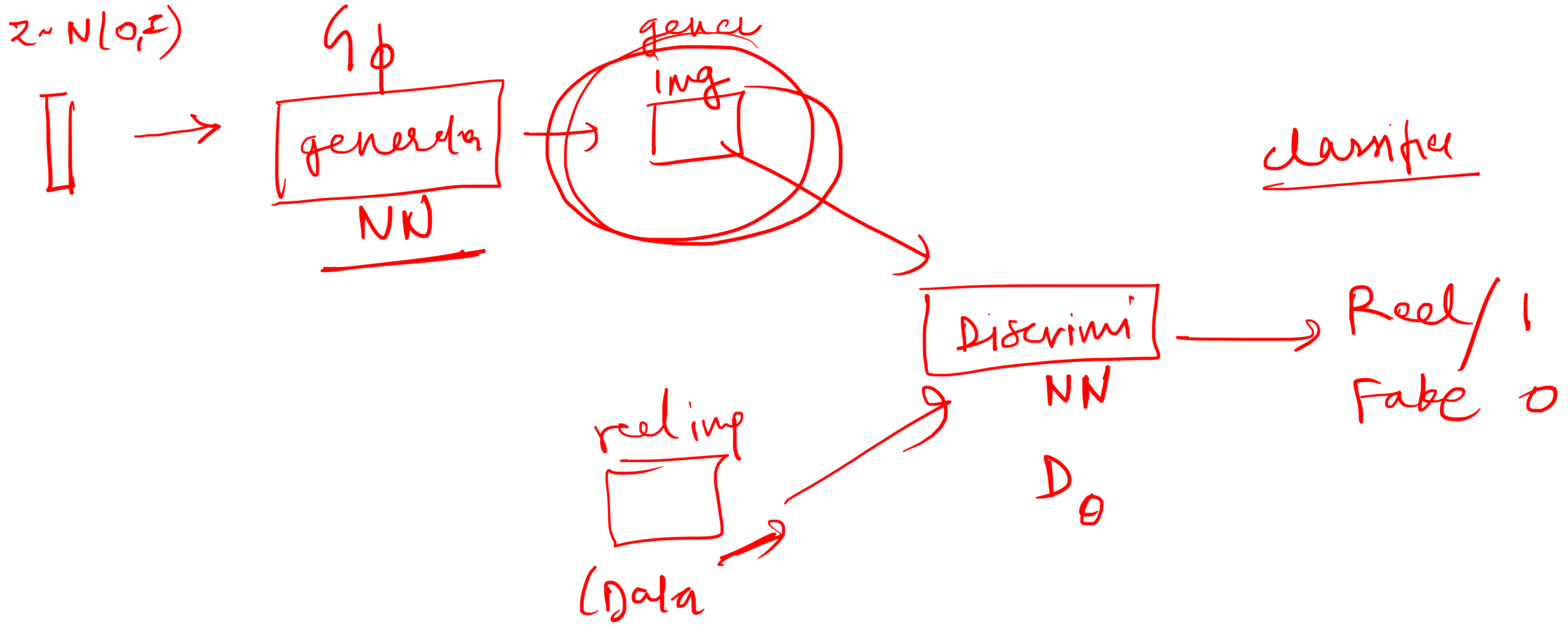
$p(x|z)$

GAN: avoid explicit learning of distr^n

① sample from $z \sim N(0, I)$

② start learning transform^n from this z to training distr^n (data)





z ~ N(0, I)
Gϕ
generator
NN
gener
img
classifier
Discrimi
NN
Dθ
Real/ 1
Fake 0
real img
(Data

$G_\phi(\ )$ generator

$D_\theta(\ )$ discriminator

$G_\phi(z) \rightarrow \tilde{x}$

$z \sim N(0, I)$

$D_\theta(x)$ / $D_\theta(G_\phi(z)) = D_\theta(\tilde{x})$

x: training

$D_\theta(G_\phi(z)) \uparrow$

discriminator

$D_\theta(x)$ = close to 1

$D_\theta(\underbrace{G_\phi(z)}_{\tilde{x}})$ = close to 0

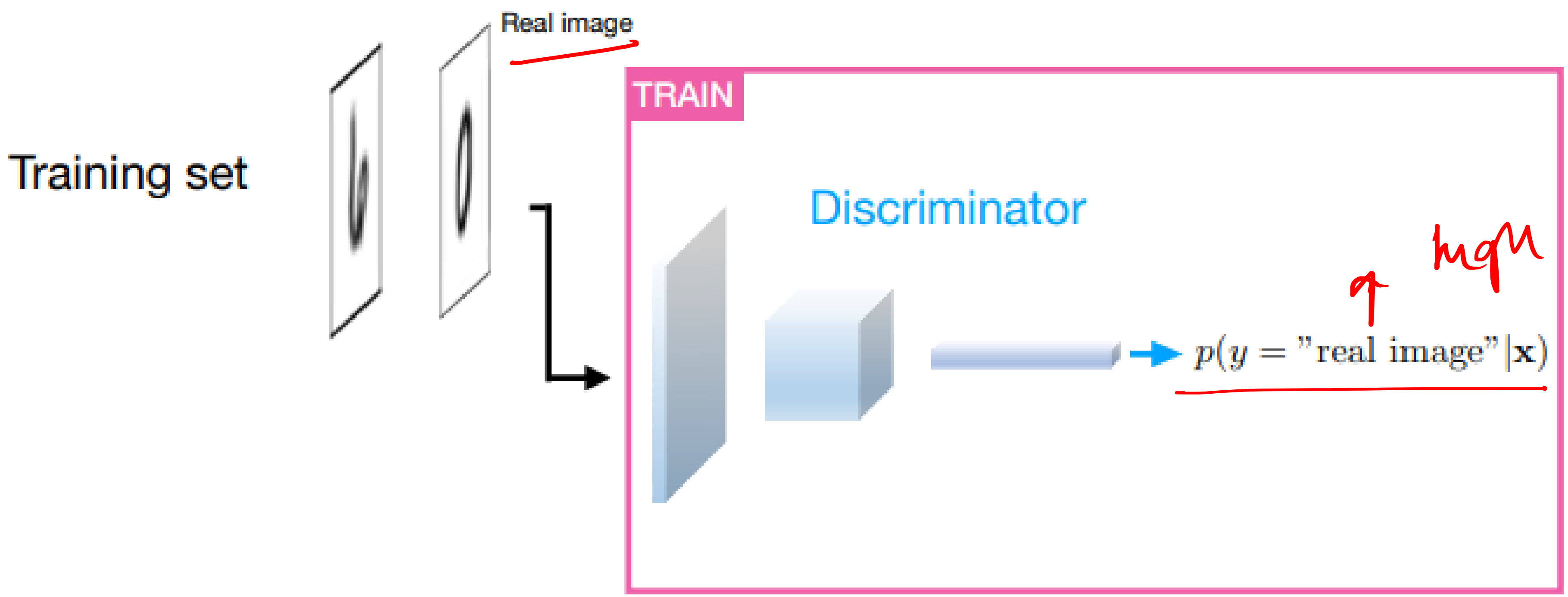
generator

$G_\phi(z) = \tilde{x}$

$D_\theta(\tilde{x})$ = close to 1

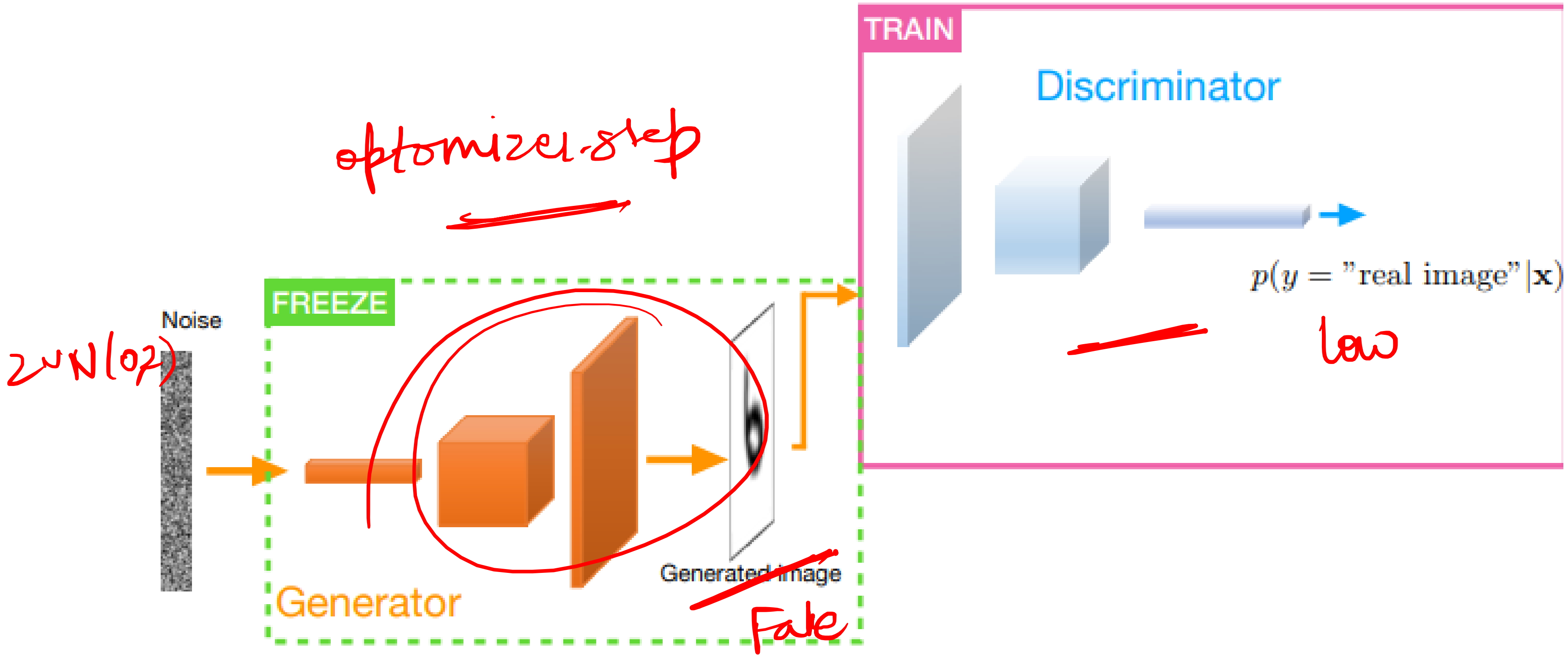
although $\tilde{x}$ is fake D_θ say it Real.

Step 1: Train Discriminator



Train to predict that real image is real

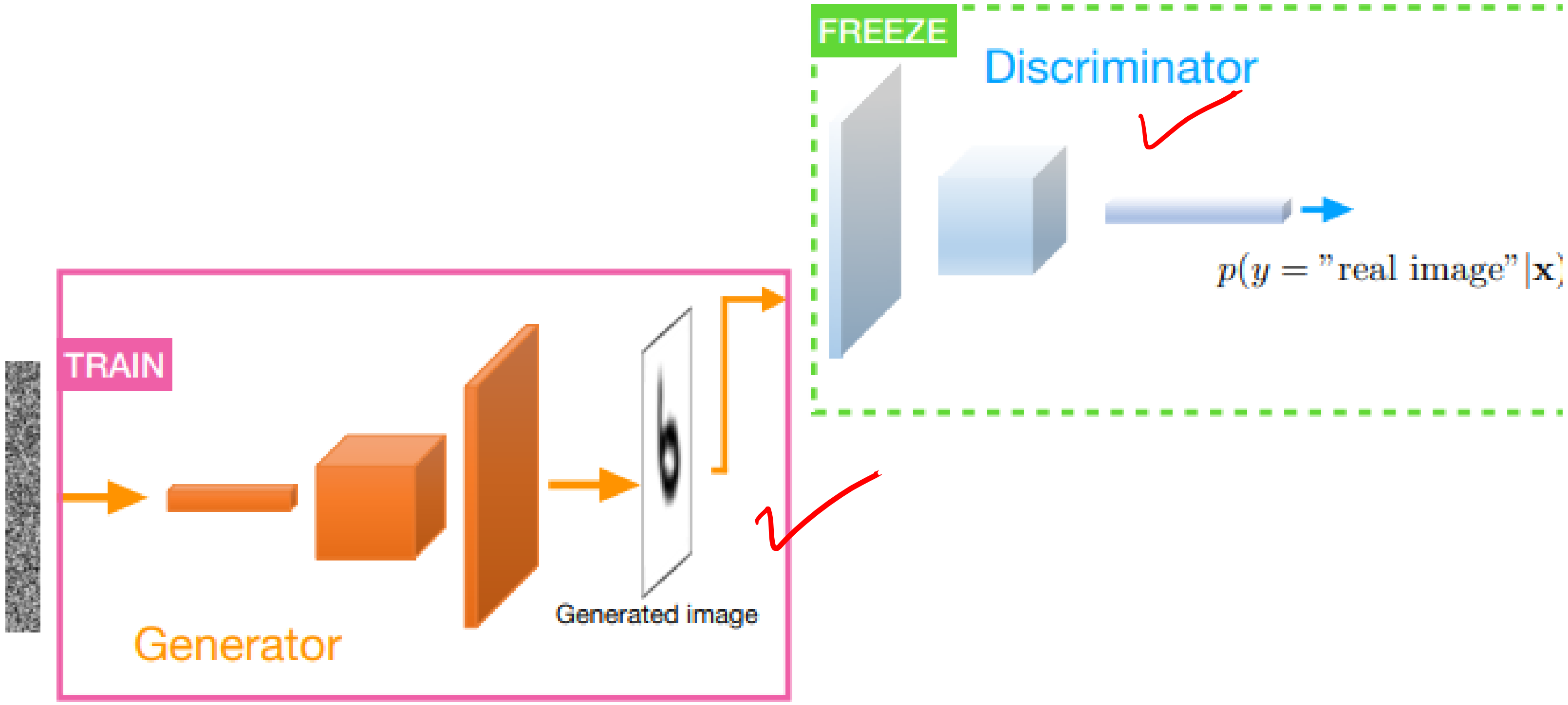
Step 1: Train Discriminator



Train to predict that fake image is fake

Step 2: Train Generator

$D_\theta(G_\phi(z))$ ↑



Train to predict that fake image is real

Discriminator gradient for update (gradient ascent):

predict well on real images
=> want probability close to 1

predict well on fake images
=> want probability close to 0

$$\nabla_{\mathbf{W}_D} \frac{1}{n} \sum_{i=1}^{n} \left[\overbrace{\log D\left(\boldsymbol{x}^{(i)}\right)} + \log \overbrace{\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)} \right]$$

$\tilde{x}$ (generated)

Generator gradient for update (gradient descent):

predict badly on fake images
=> want probability close to 1

$$\nabla_{\mathbf{W}_G} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}\right)$$

$\downarrow D(G(z^i))$

$\uparrow (1 - D(G(z_i)))$

issues

- Oscillation between generator and discriminator loss
- Mode collapse (generator produces examples of a particular kind only)
- Discriminator is too strong, such that the gradient for the generator vanishes and the generator can't keep up
- Discriminator is too weak, and the generator produces non-realistic images that fool it too easily (rare problem, though)

Discriminator gradient for update (gradient ascent):

$$\nabla_{\mathbf{W}_D} \frac{1}{n} \sum_{i=1}^{n} \left[\log \overbrace{D\left(\boldsymbol{x}^{(i)}\right)}^{\substack{\text{predict well on real images} \\ \text{=> want probability close to 1}}} + \log \left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\substack{\text{predict well on fake images} \\ \text{=> want probability close to 0}}}\right)\right]$$

~~Generator gradient for update (gradient descent):~~

$$\cancel{\nabla_{\mathbf{W}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\substack{\text{predict badly on fake images} \\ \text{=> want probability close to 1}}}\right)}$$

Do gradient ascent with

$$\nabla_{\mathbf{W}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

And flip labels