PS4 CS 7643 Adrian Kulla

2.1 Loss would be the reconstruction error.

$$Loss = \sum_{x \in dataset} \| x - g_\beta(f_\alpha(x)) \|^2$$

2.2 It's not practical because there can be gaps in the latent space distribution where there are no training samples. In a vanilla auto encoder, the encoder maps inputs to specific points in the latent space without enforcing a valid distribution. Hence, randomly sampling from the latent space, one might sample from regions that don't correspond to true data points, which can result in unrealistic outputs.

So, vanilla auto encoders don't learn like a proper generative model.

They're still useful to:

- reduce dimensionality from x inputs to learn from a latent representation (z)
- so can compress data
- can be used for pretraining for future downstream tasks.
- Latent encoding can capture features of data for feature learning
- can be used for denoising to remove noise.

PS4 CS7643 Adrian Kulula

2.3

$$D_{KL}(p(x)\|q(x)) = E_{x\sim p(x)}[\log(p(x))] - E_{x\sim p(x)}[\log(q(x))]$$

It measues how much extra information is needed to encode samples from distribution p assuming code is optimized for distribution q.
So, it measures the difference between cross entropy of (p and q) and the entropy of p. It approximates the lost information when one uses "q" to approximate "p" distributed data.
It will always be non-negative and will be zero iff $p(x)$ and $q(x)$ are aligned so it acts as a distance measure between two distributions.

- - - - - - - - - - - -

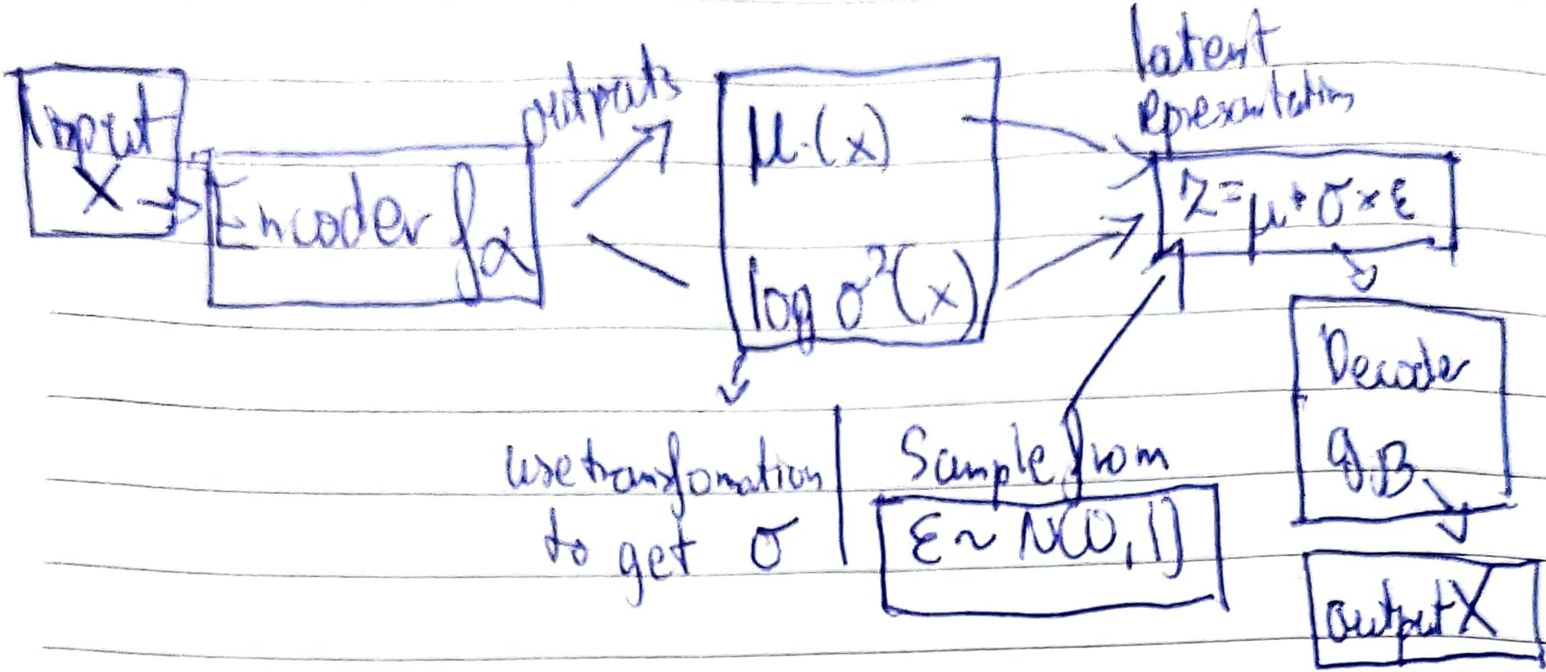
The encoder f_α should output the parameters of a Gaussian distribution, i.e. mean vector $\mu_{f\alpha}(x)$ and variance vector $\sigma^2_{f\alpha}(x)$ for input x.
KL divergence term should capture the encoded distribution's difference from standard normal distribution.
encoded distr^n = $q_{f\alpha}(z|x) = N(\mu_{f\alpha}(x), \sigma^2_{f\alpha}(x))$
$p(z) = N(0, I)$ which is the desired prior distribution. Answer below:

So $D_{KL}(N(\mu_{f\alpha}(x), \sigma^2_{f\alpha}(x)) \| N(0, I))$

is the KL divergence term

PS4 CS7643 Adrian Kulla

2.4



2.5 $p_\theta(x, z) = p_\theta(x|z) \times p_\theta(z)$ Bayes' rule

$$ELBO = E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)]$$

$\log p_\theta(x)$ is constant w.r.t. z

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$

$$p_\theta(x) = \int p_\theta(z, x)\, dz$$

Continuing:

$$\log (p_\theta(x)) = E_{z \sim q_\phi}(z|x)[\log p_\theta(x)]$$

$$= E_{z \sim q_\phi(z|x)}[\log (p_\theta(x, z) / p_\theta(z|x)]$$

using $p_\theta(z|x) = p_\theta(x, z) / p_\theta(x)$ Hence,

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(z, x) - \log p_\theta(z|x)]$$

$$= E_{z \sim q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x) + \log q_\phi(z|x) - \log p_\theta(z|x)]$$

Hence

$$\log p_\theta(x) = ELBO + E_{z \sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z|x)]$$

PS4 CS 7643 Adrian Kulla

2.5 continued

second term is KL divergence:

$D_{KL}(q_\phi(z|x) \| p_\theta(z|x))$, which is always non-negative

Hence,

$\log p_\theta(x) = ELBO + D_{KL}(q_\phi(z|x) \| p_\theta(z|x)) \geq ELBO$

2 b

$$ELBO = E_{z \sim q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)]$$

$$= E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]$$
$$+ E_{z \sim q_\phi(z|x)}[\log p_\theta(z)]$$
$$- E_{z \sim q_\phi(z|x)}[\log q_\phi(z|x)]$$

combine term 2 and 3

$$\Rightarrow ELBO = E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] \leftarrow ①$$
$$- E_{z \sim q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z)}\right] \leftarrow ②$$

Part ① is the reconstruction term similar to loss term in standard autoencoder. It measures how well we can reconstruct x from latent variable z. Maximizing this term ensures the decoder learns to reconstruct $g_\theta(z)$ i.e. match to actual input data x

Part ② is the KL divergence $D_{KL}(q_\phi(z|x) \| p_\theta(z))$ which one wants to minimize (and hence maximize ELBO) so that $q_\phi(z|x)$ is close to $p_\theta(z)$ which would allow one to generate samples from $p_\theta(z)$ and decoding

PS4 CS7643 Adrian [illegible]

2.6 continued

Maximizing ELBO encourages low reconstruction error so the decoder can learn to reconstruct actual data.

The latent space is regularized to a known prior distribution using KL divergence so one can generate data by sampling.

This ensures the VAE is a good autoencoder by closely reconstructing actual inputs and also a generative model by sampling data from prior distribution.

3.1 Generator 'G' -> tries to produce synthetic data to fool the discriminator
Discriminator 'D' -> tries to distinguish between real and generated samples
So sets up a minimax two player game:

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

as generator wants to minimize discriminator's ability to distinguish between samples and the discriminator wants to maximize its ability to correctly classify generated vs. real samples.
Nash Equilibrium occurs when the generator produces data indistinguishable from real data and the discriminator can do no better than random guessing ($D(x) = 0.5$)

3.2

Generators: G: $(X \to Y)$ and F: $(Y \to X)$

Discriminators: D_Y distinguishes real Monet paintings from fake Monet style images i.e. receives y and $\hat{y} = G(x)$

D_X distinguishes real photos from fake photos i.e distinguish whether $x, \hat{x}$ are real/~~fake~~ $\to \hat{y} = G(x)$

So there are G: $(X \to Y)$ which transforms regular images to Monet paintings and F: $(Y \to X)$ which transforms Monet paintings to regular images $\hat{x} = F(y)$

There are two separate two player games

G vs. D_Y => generator G tries to create Monet style images to fool D_Y

F vs D_X => generator F tries to create regular images that fool D_X.

Nash Equilibria:

G produces images that D_Y can't distinguish from real Monet paintings.

F produces regular images that D_X can't distinguish from real images.

In both cases discriminators are reduced to random guessing (0.5 probability).

PS4 CS 7643 Adrian Kulala

3.3

So cycle consistency enforces that
$F(G(x)) = x$ -> image -> Monet -> image should return to original
$G(F(y)) = y$ -> Monet -> image -> Monet should return to original.
Without cycle consistency translations input identity wouldn't be constrained to preserve info.
So generators could map any arbitrary input to any output to fool the discriminator. mapping between

With cycle consistency network learns to perform a style change while preserving the identity of original image.

Without cycle consistency generator could create images that resemble the target domain but change the content completely. Many images could map to same output image such that the mapping would lose the structure between domains.

G is the generator/mapping function from regular images X to Monet paintings Y.
F is the generator/mapping function from Monet paintings Y to real images X.

So in conclusion, one preserves content but transforms style.