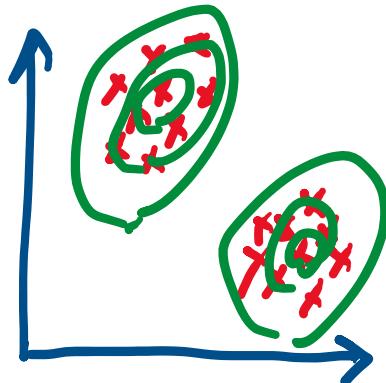


Deep Generative Models

Generative modeling

- Learns $P(X, Y)$ or $P(X|Y)$
 - Can sample (generates) X from $P(X|Y)$
 - Y is the controlling parameter
- What is $P(X|Y)$?
 - Most of the time it's assumed to be a parametric distribution



Generating real world data

- Let's say we want to generate faces
 - How do we fit a distribution to it?

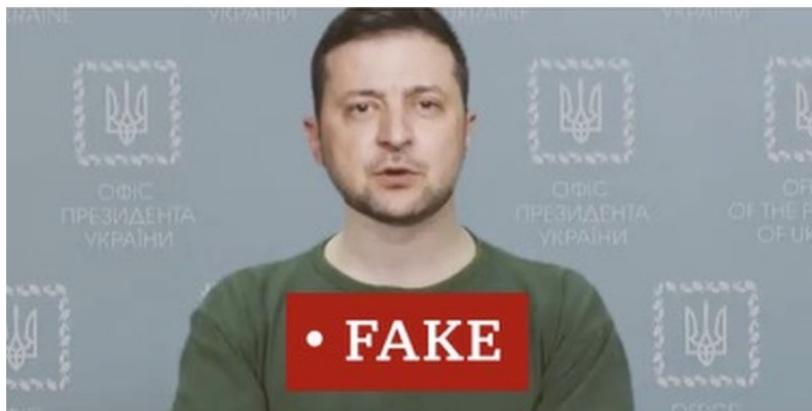
Deepfake presidents used in Russia-Ukraine war

By Jane Wakefield
BBC Technology

5 days ago



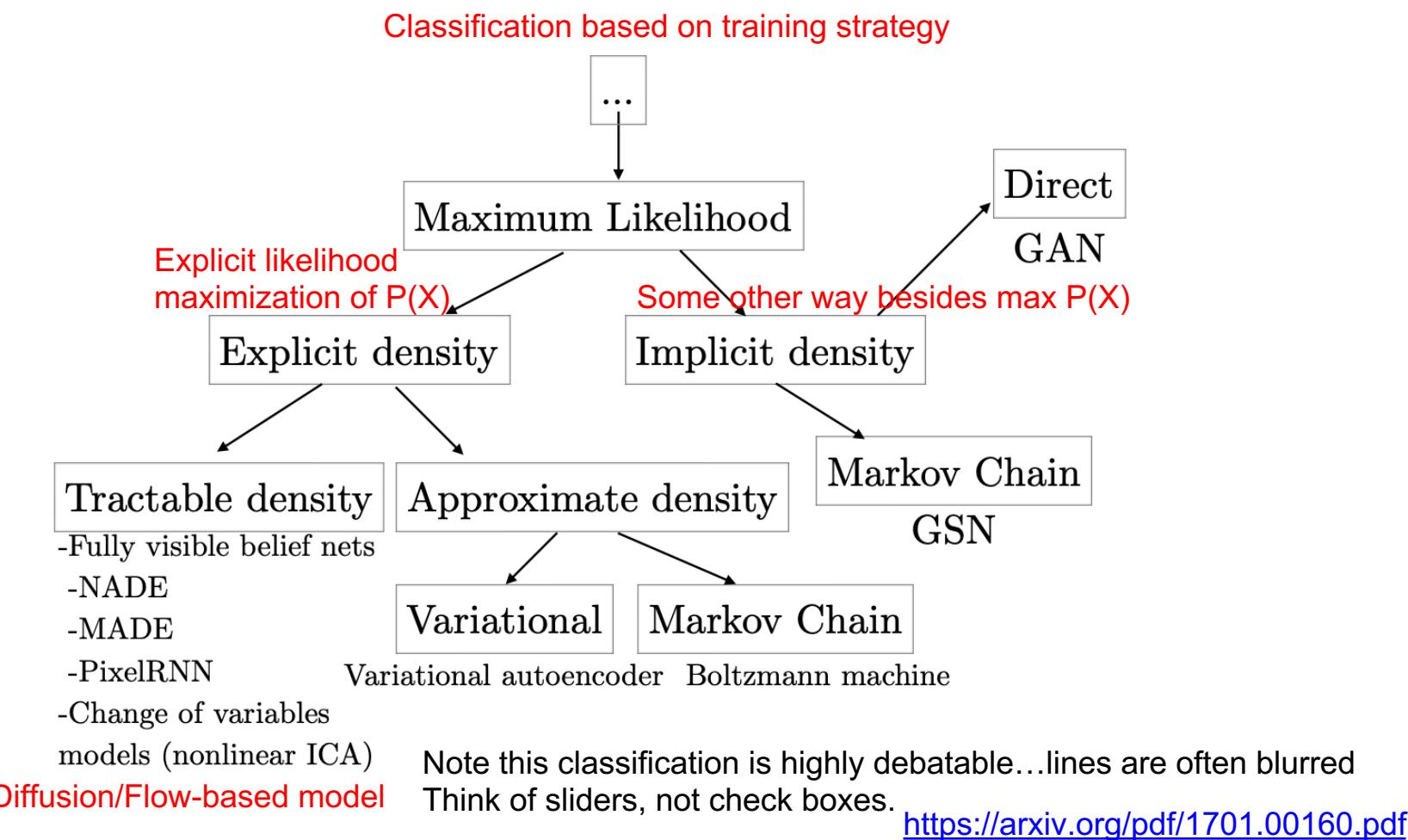
Russia-Ukraine war



<https://www.bbc.com/news/technology-60780142>

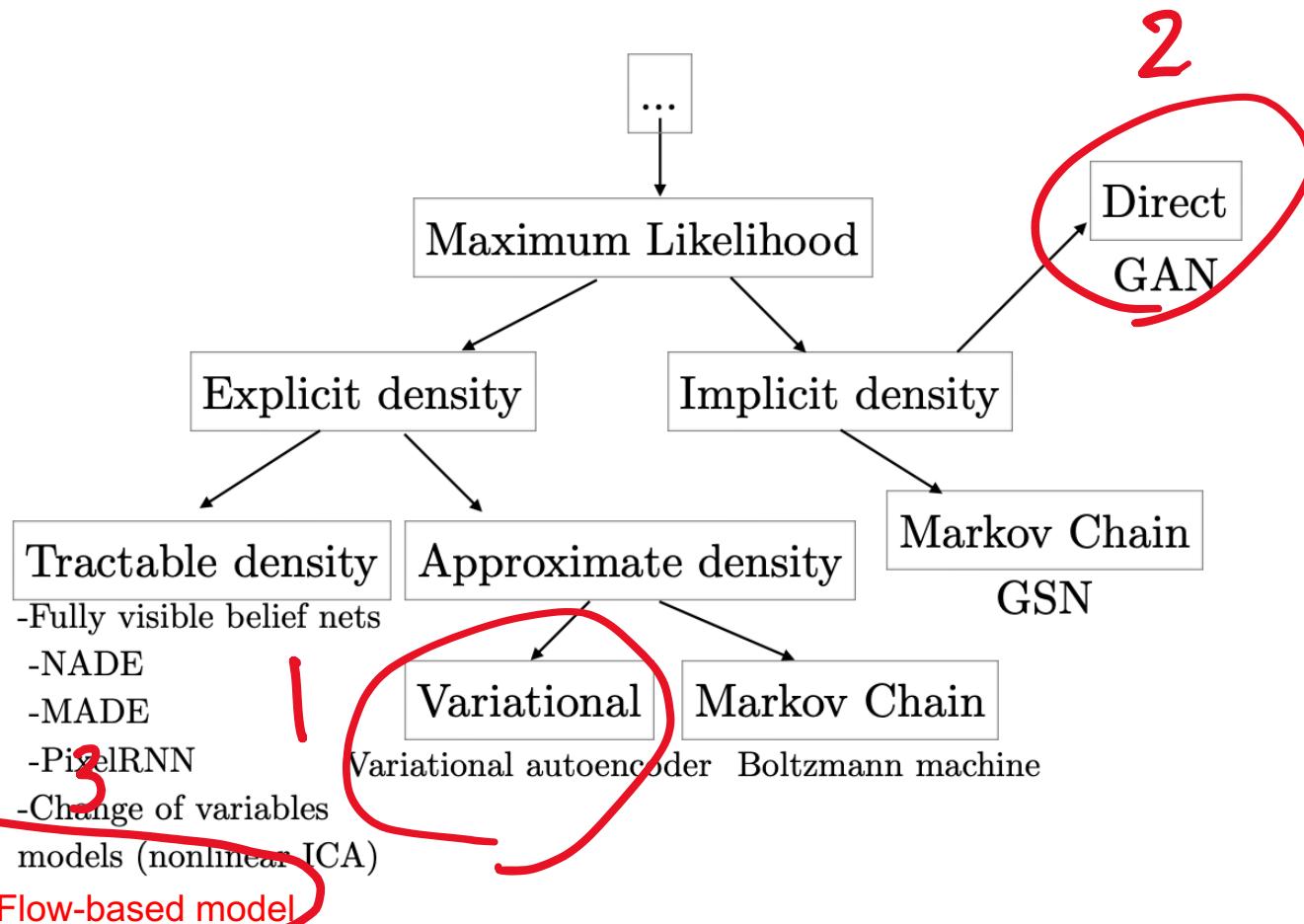
Implicit generative models

- We can generate without explicitly modeling $P(X)$



Deep Generative Models

- A deep learning model that can be used to generate X

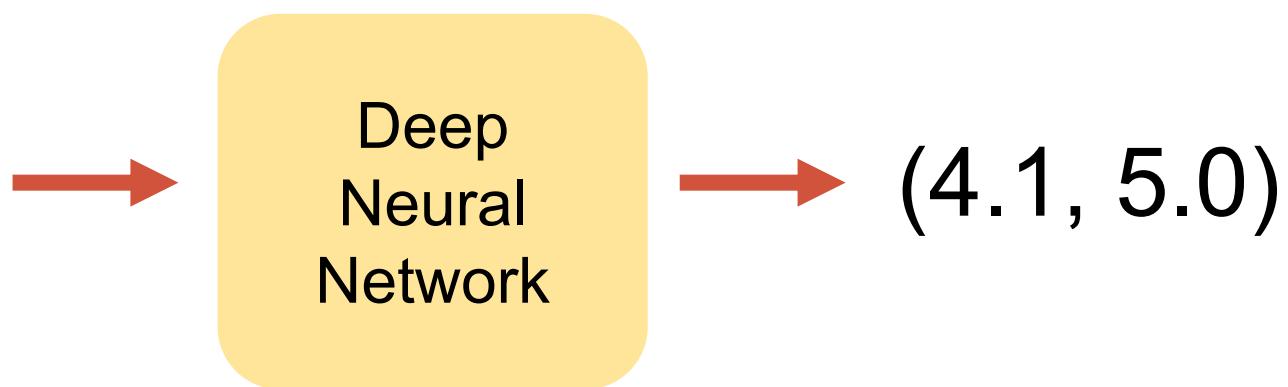
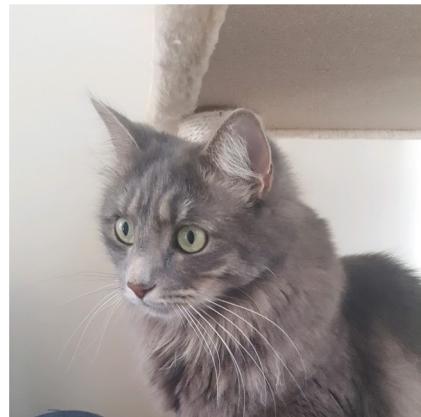


Deep generative models

- Autoencoders
 - Variational autoencoders
- GAN
- Diffusion/Flow

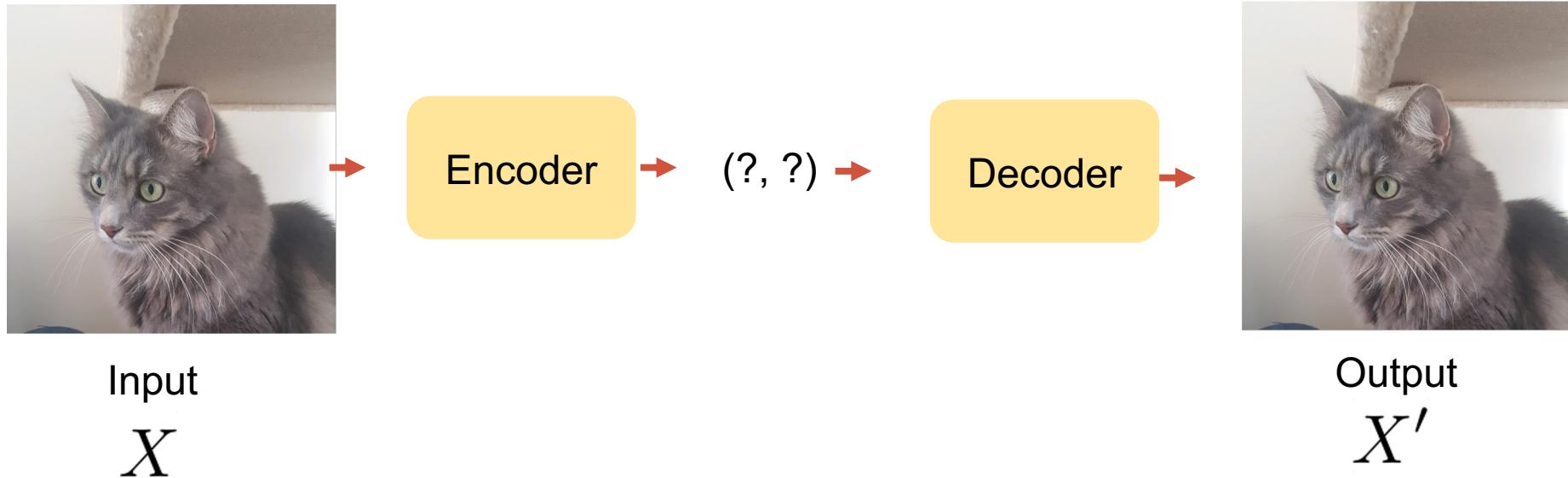
DNN as a representation learning model

We know that DNN can model highly complex mapping.



Autoencoder

Auto means self!

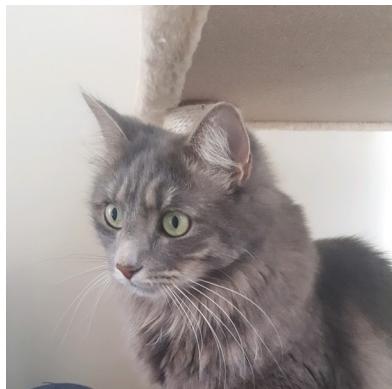


Now we can train the network with L2 Loss:

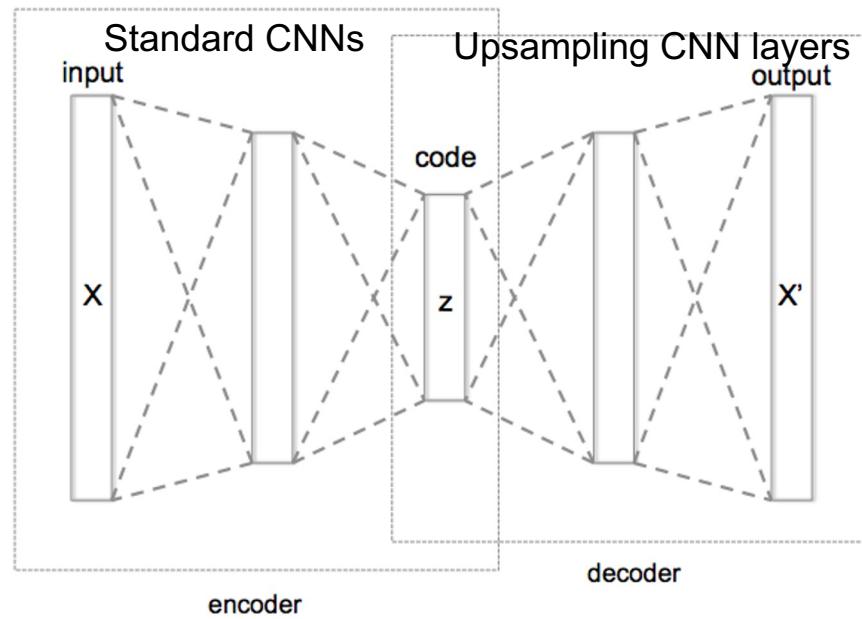
$$\|X - X'\|^2$$

Autoencoder

Auto means self!



Input
 X

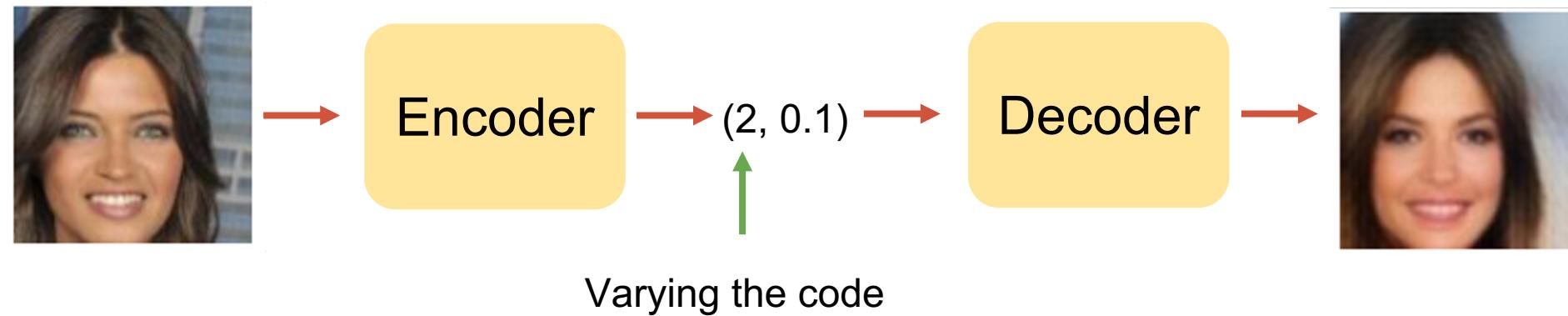


Output
 X'

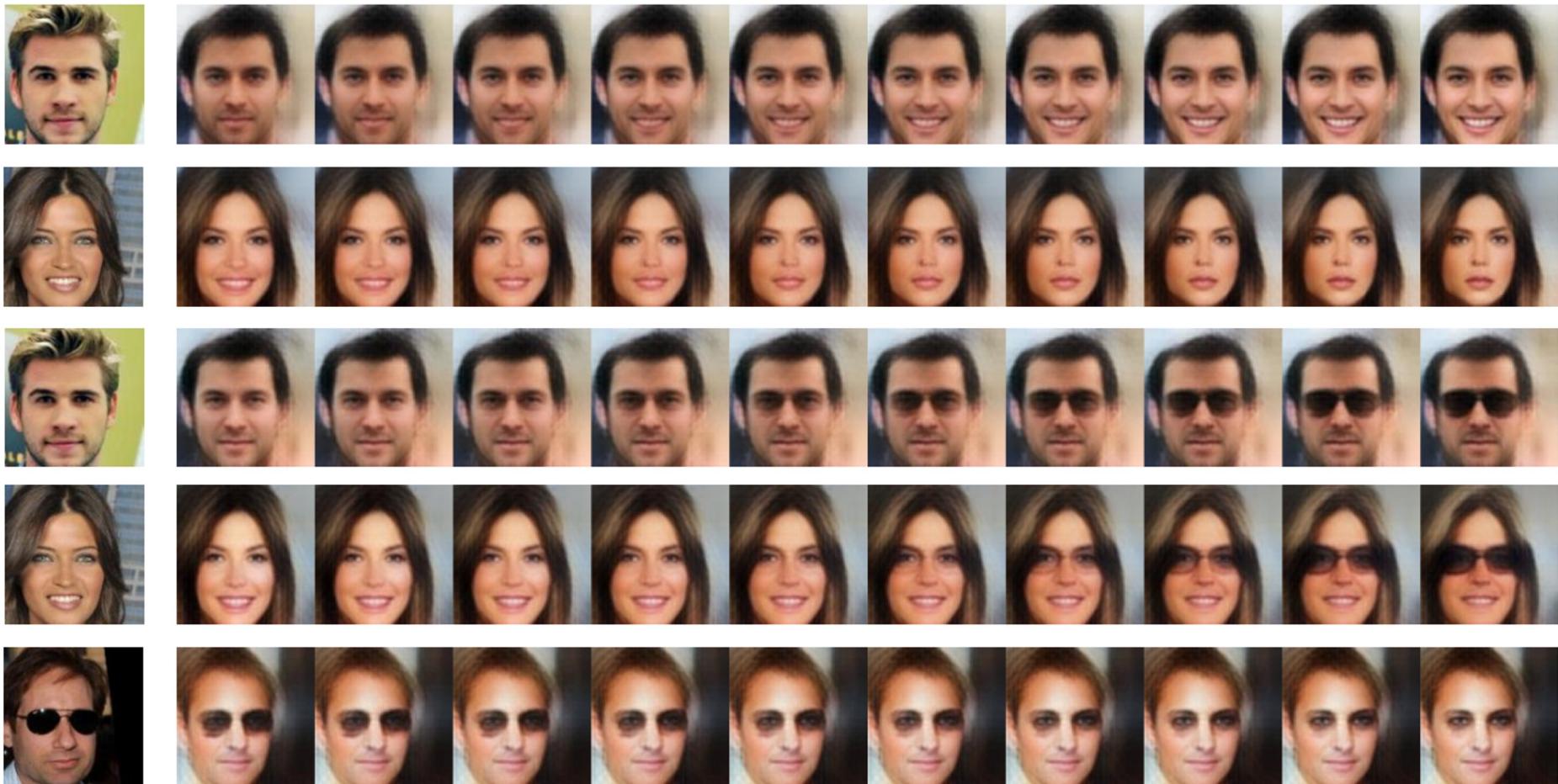
Now we can train the network with L2 Loss:

$$\|X - X'\|^2$$

Inspecting attributes learned from AE



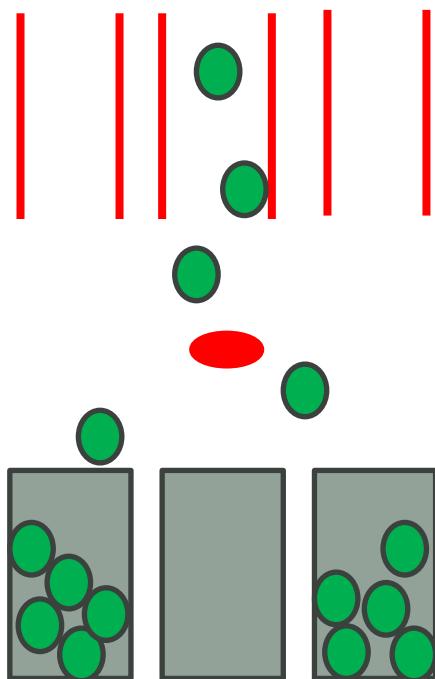
Attributes from autoencoder



Deterministic mapping

Autoencoder is good, but this learns a **deterministic** mapping from $X \xrightarrow{\text{encode}} Z \xrightarrow{\text{decode}} X'$ (point estimate)

Let's say the loss function is L2 $\|X - X'\|^2$

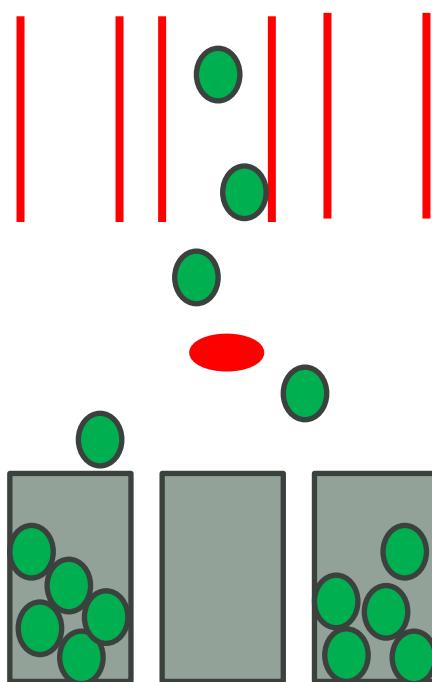


A deterministic mapping can lead to improbable answers. We need some **non-deterministic** mapping (distribution learning)

Distribution learning

One simple way to think of deep distribution learning is we're learning a mapping to a distribution instead

$[X_1, \dots, X_n] \rightarrow \text{distribution} \rightarrow [X'_1, \dots, X'_m]$



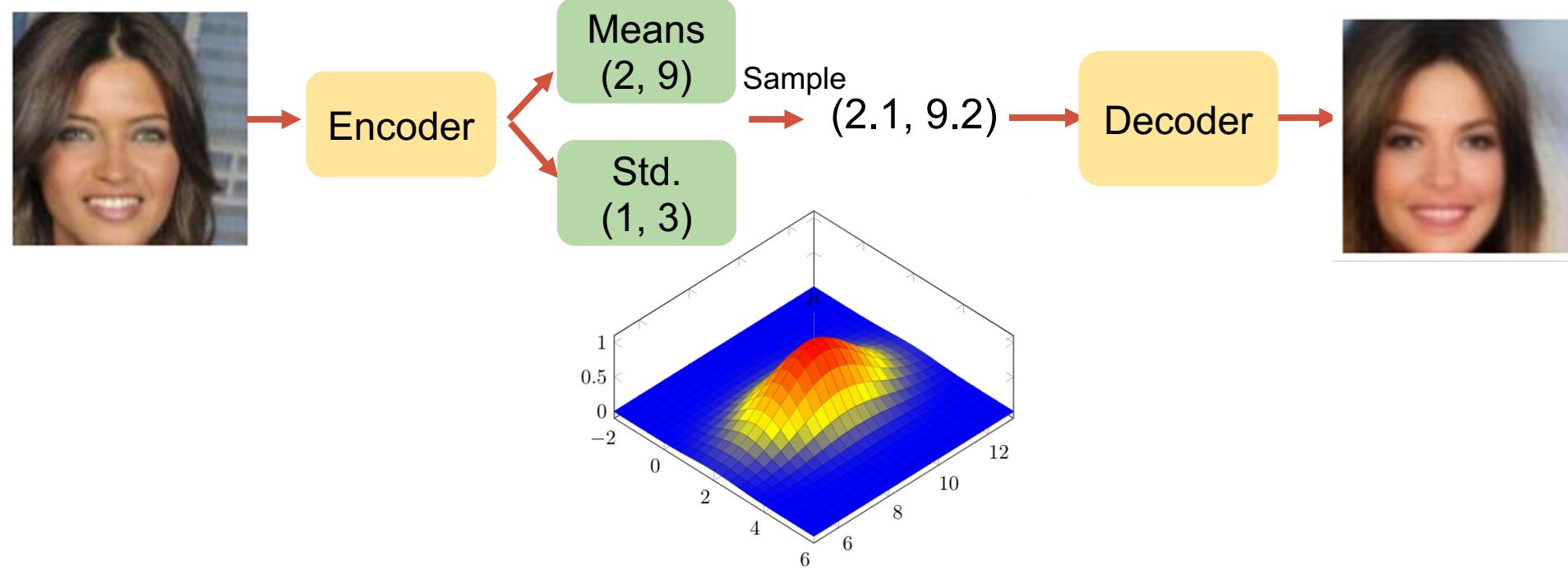
Variational Autoencoder

Original and by-product motivations:

0. Learning a non-deterministic mapping
1. Allows approximation of intractable posterior.
2. Generate photos from the decoder

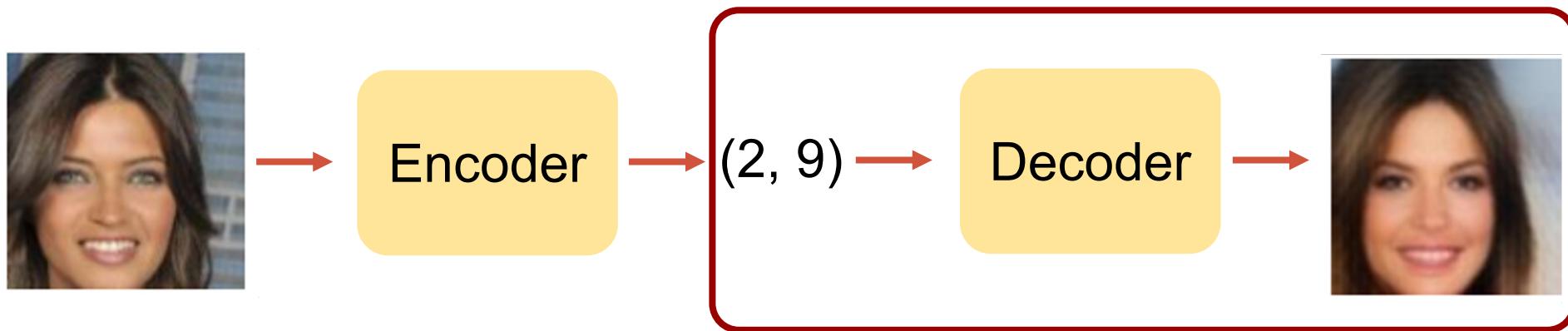
Motivation #0 – learning a non-deterministic mapping

Probabilistic AE



The encoder learns the parameters of a distribution instead
Then, we sample to generate.

Motivation #2 -- Generating Photos

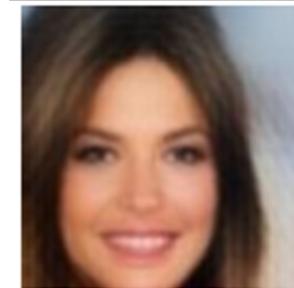
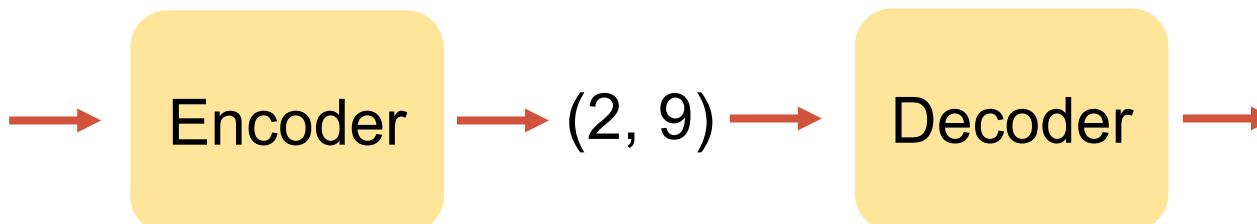


Use this part to generate an image from random code.

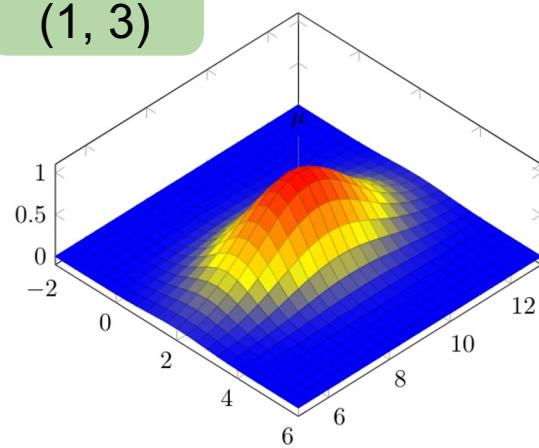
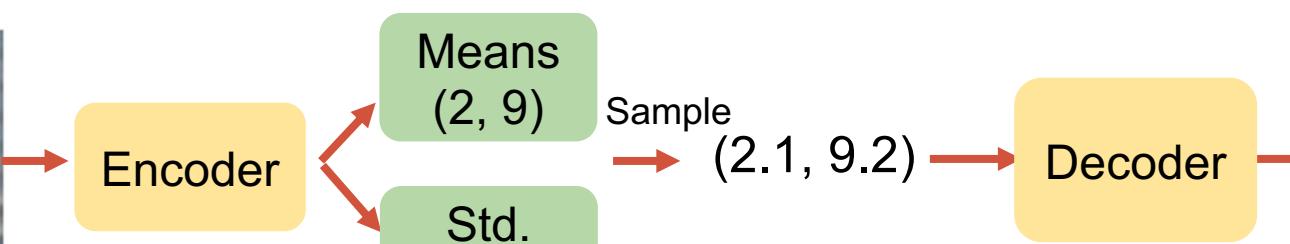
But how do you generate the two numbers? Uniform? Gaussian?
Encoder could output two numbers with some crazy distribution.

VAE allows us to force the latent code to follow some distribution like a unit normal distribution.

Standard AE

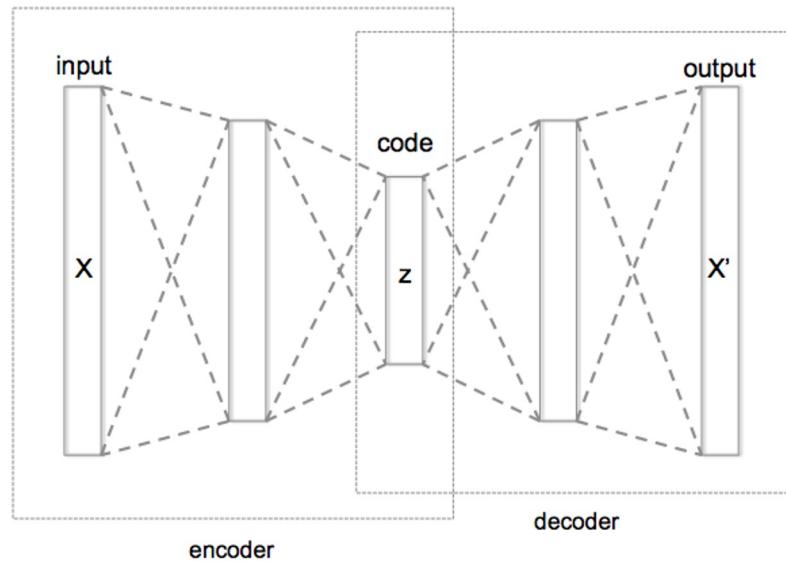


Probabilistic AE



Motivation #1 -- Solving Intractable Inference

Autoencoder can be viewed from a probabilistic perspective.



However, a standard AE isn't quite "probabilistic."
I.e., the encoder deterministically outputs z given x . Rather we want a distribution of the code (posterior $p(z|x)$).

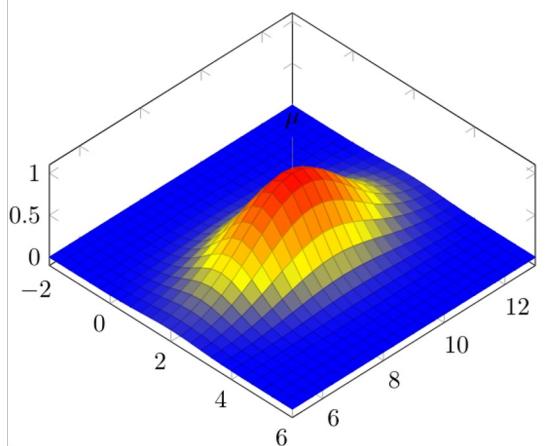
Autoencoding Variational Bayes: Bayesian Review

One important inference problem : Posterior inference

$$\underline{p(\mathbf{z} \mid \mathbf{x})}$$

x Observed data

z Latent variable
(hidden code)



Bayesian Review

One important inference problem : Posterior inference

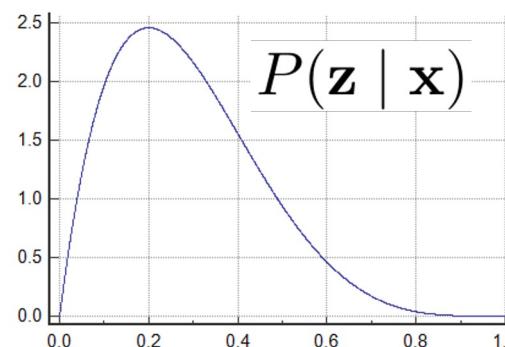
$$p(\mathbf{z} \mid \mathbf{x})$$

x Observed data

2 heads out of 10 coin flips
(Binomial Dist)

z Latent representation
(hidden code)

The coin fairness
 $z = 0.5$, fair coin
 $z = 1$, always head
 $z = 0$, always tail



Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x})$$

x Observed data



z Latent representation
(hidden code)

Class label cat, color,
can fly?, ...

$$p(\mathbf{z} \mid \mathbf{x})$$

Tells you **not only** what animal is the most likely,
but also how “orange cat” it is, how “white dog” it is, etc --
the probability of all hidden code

Variational Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\boxed{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}}$$

Hard to compute because \mathbf{z} can be high-dimensional

E.g. $\mathbf{z} = \{z_1, z_2, z_3, \dots\}$

z_1 : class label {cat, dog, rabbit, rat}

z_2 : color, a real number

z_3 : ability to fly {yes, no}

$$\int d\mathbf{z} = \sum_{z_1} \int_{z_2} \sum_{z_3} \dots dz_2$$

Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\boxed{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}}$$

A solution:

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$

with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$

and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$
with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$
and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Best $q^*(\mathbf{z} \mid \mathbf{x})$ is the one closest to $p(\mathbf{z} \mid \mathbf{x})$

A solution: $q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})]$

- But why not $\overline{\text{KL} [p(\mathbf{z} \mid \mathbf{x}) \parallel q(\mathbf{z} \mid \mathbf{x})]}$ or $D(p(\mathbf{z} \mid \mathbf{x}), q(\mathbf{z} \mid \mathbf{x}))$?
- How is this related to autoencoder?

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Because doing this (i.e., using $\text{KL}(q \parallel p)$):

$$q^*(\mathbf{z} \mid \mathbf{x}) = \underset{q}{\operatorname{argmin}} \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})]$$

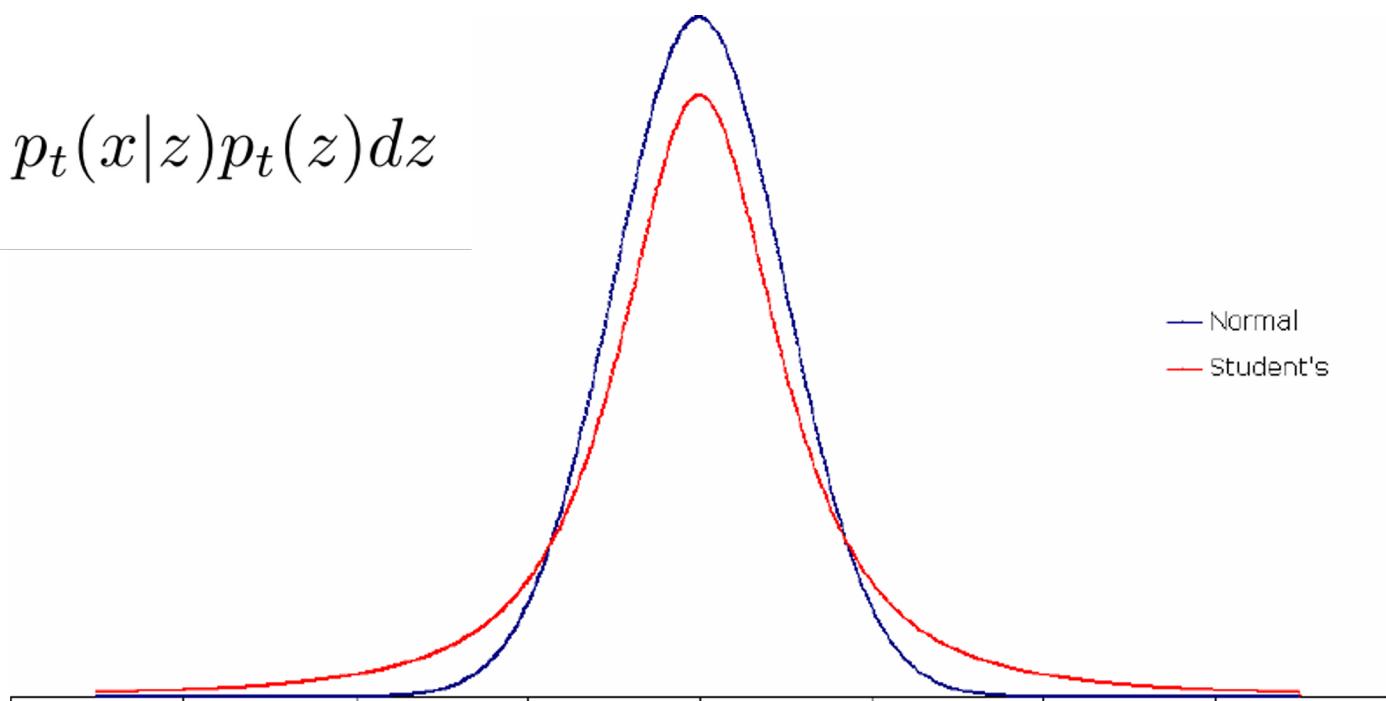
Corresponds to maximizing the evidence lower bound (ELBO)!

Digression: Model evidence

Model evidence (or marginal likelihood) is simply $p(\mathbf{x})$

$$p_g(x) = \int p_g(x|z)p_g(z)dz$$

$$p_t(x) = \int p_t(x|z)p_t(z)dz$$



EM algorithm

$p(x)$ - evidence

- Goal of EM : $\log \sum_k p(x, k; \theta) \geq \sum_k Q(k) \log (p(x, k; \theta)/Q(k))$
- Maximize the ELBO instead
- Initialize Θ
- Repeat till convergence
 - Expectation step (E-step) : estimate the conditional expectation $Q(k) = p(k|x; \theta)$ using the current θ .
 - Maximization step (M-step) : Estimate new Θ given by maximizing the ELBO given current $Q(k)$

$$E_{k \sim Q} \left[\log \frac{p(x, k)}{Q(k)} \right]$$

$p(x|Y)$ - likelihood

$p(Y|X)$ - posterior

Proof

This KL is hard to compute

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})]$$

$$\text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] = \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z} \mid \mathbf{x})]$$

$\nwarrow q(\mathbf{z} \mid \mathbf{x}) \quad \text{(by definition)}$

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

(expanding $P(z \mid x)$, $E_q[\log p(x)] = \log p(x)$)

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

Constant w.r.t q. So, we can safely kill it

$$\min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] \iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Questions: 1. What's this new term? 2. Is it easier to evaluate?

Proof

$$\min_q \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] \iff \min_q \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Let's interpret this new term:

From previous slide, we have:

$$\log p(\mathbf{x}) = \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

We know $\text{KL} \geq 0$, always!

$$\log p(\mathbf{x}) \geq \boxed{-\mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]}$$

(Lower bound of model evidence)

$$\begin{aligned} \min_q \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] &\iff \max_q -\mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] \\ &\iff \max_q \text{ELBO} \end{aligned}$$

But how to compute this ELBO term then?

Maximizing ELBO

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})]$$

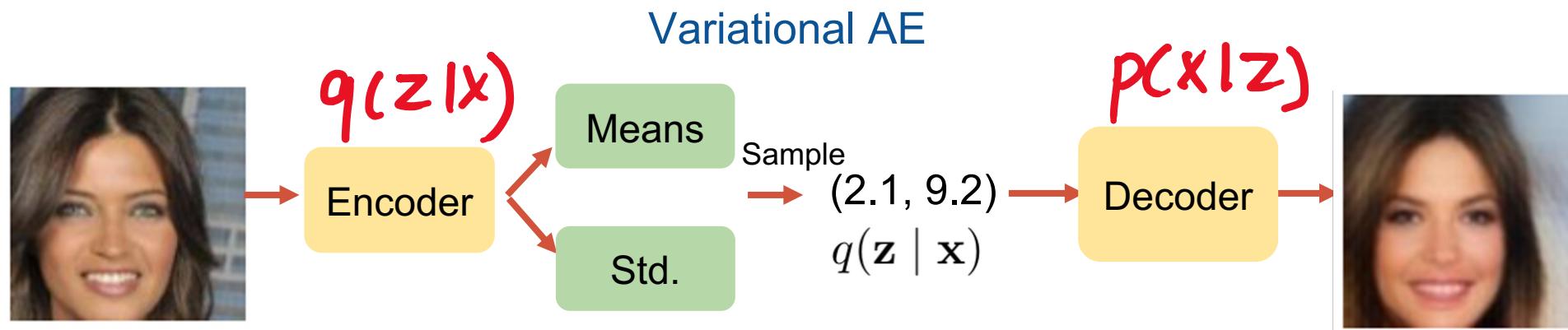
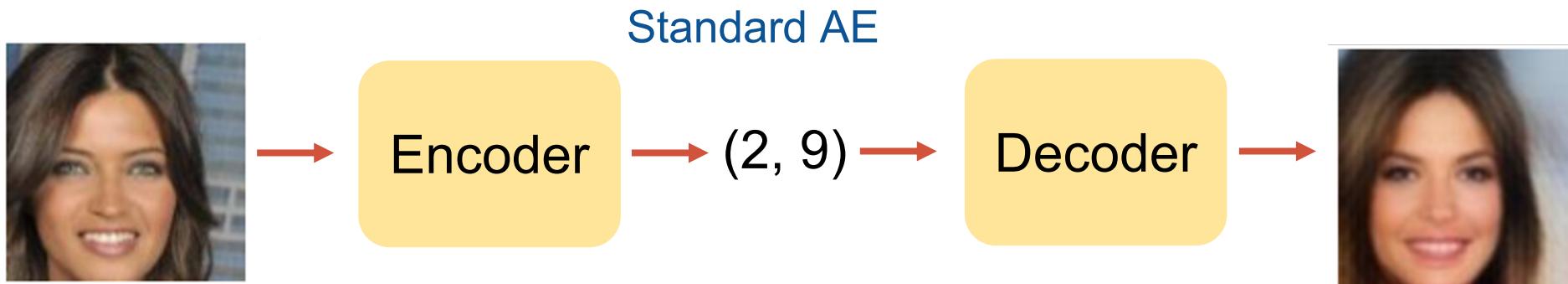
$$\max_q \text{ELBO} = \max_q -\mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{z})]$$

$$\iff \min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

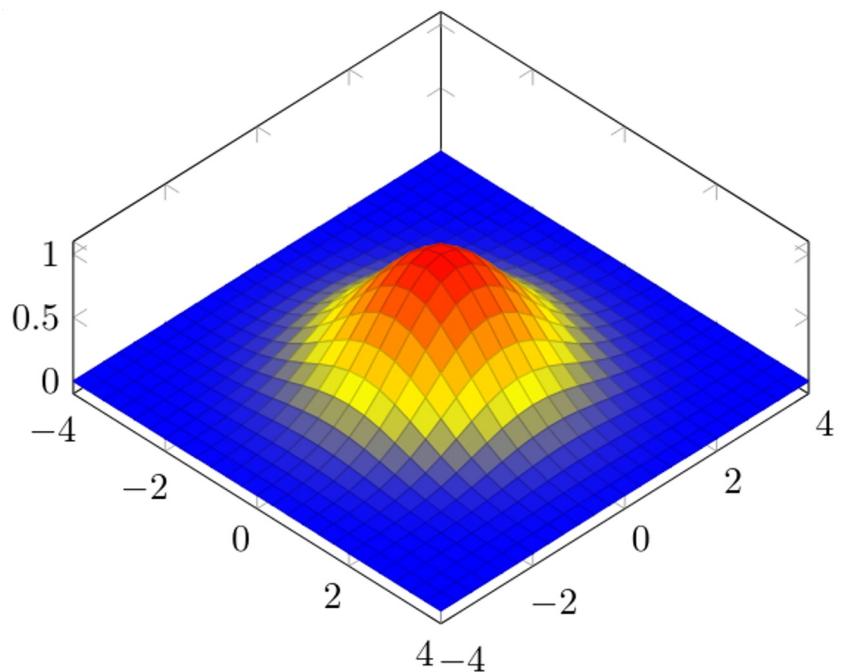
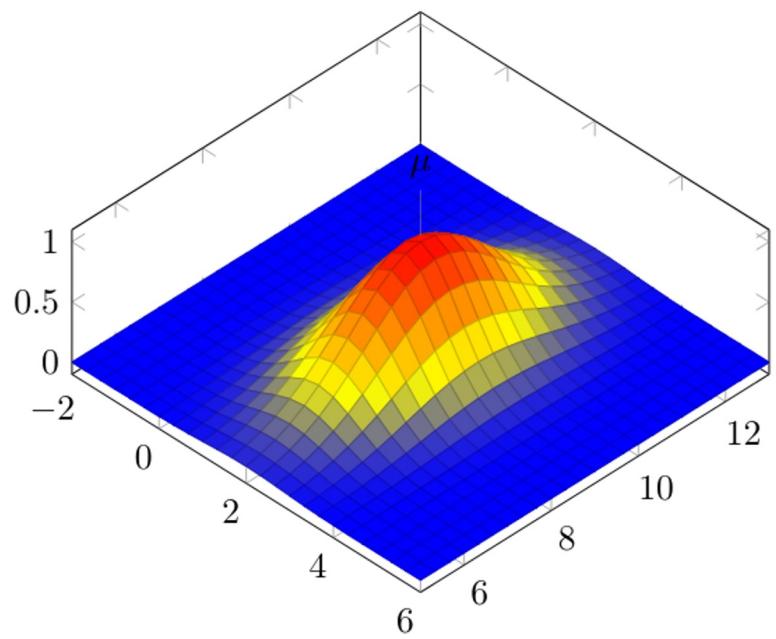
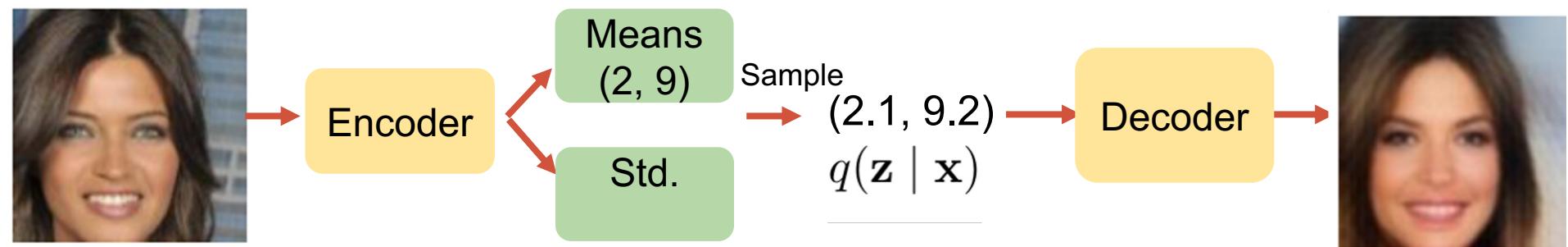
We're free to pick



*Output as a sample
from $P(x | z)$.

$$\min_q \text{KL}[q(\mathbf{z} | \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} | \mathbf{z})]$$

sample from $q(z|x)$



$q(\mathbf{z} | \mathbf{x})$ is forced to be close to $p(\mathbf{z}) = \mathcal{N}(0, I)$

$$\min_q \text{KL}[q(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} | \mathbf{z})]$$

What is this second term?

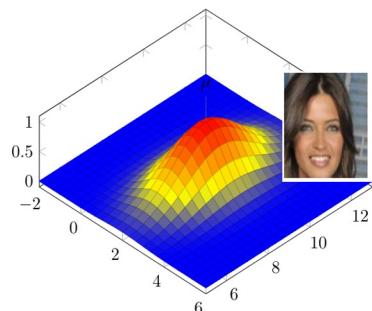
Second Term: Data Log Likelihood

$$\min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

If we also model $p(\mathbf{x} \mid \mathbf{z})$ as a Gaussian distribution, where the mean = \mathbf{u} is the output of the decoder:

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

(Let's consider a single pixel case
 \mathbf{u} is the output from the decoder
 x is the ground-truth pixel)

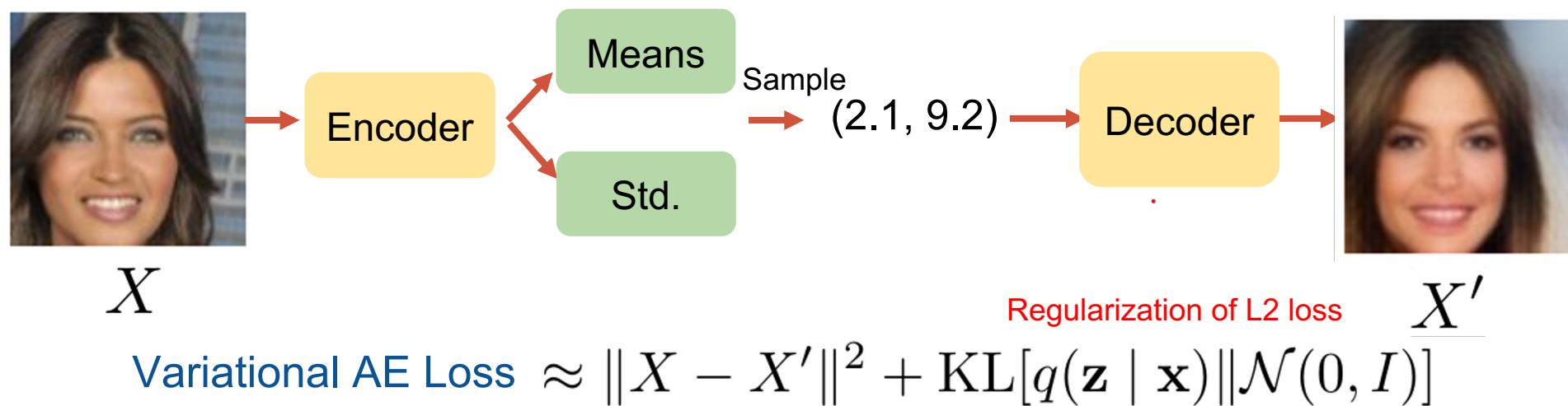
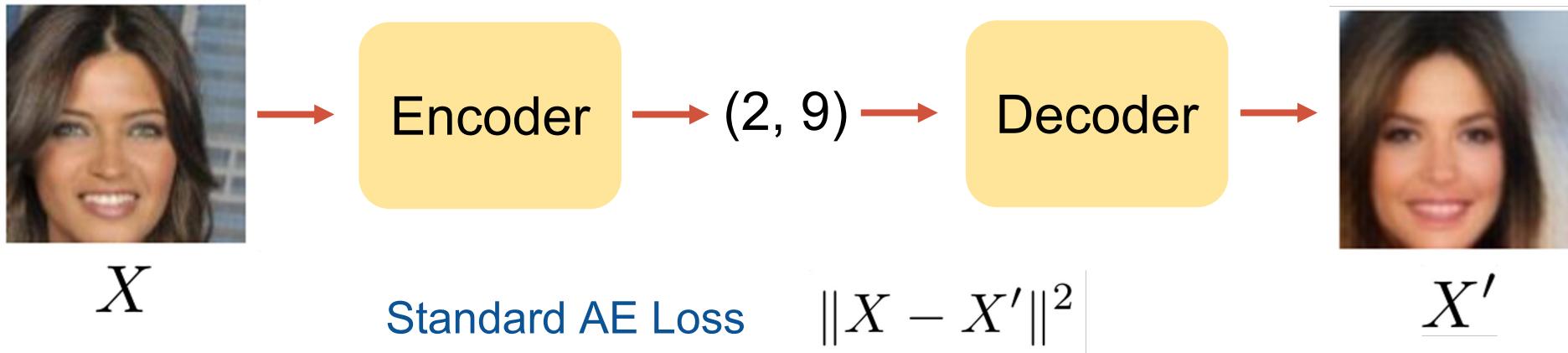


$$\propto \exp(-(x - u)^2)$$

$$-\log[\exp(-(x - u)^2)] = (x - u)^2 \quad (\text{with constants hidden})$$

Becomes the standard L2 error

Yet another reason why assuming Gaussian is nice



Precisely, $\min_{q_\phi(\mathbf{z} \mid \mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$

Implementation of KL term

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

approximate via minibatch

- You can approximate the KL term by computing in the minibatch
- If you assume both q and p are Gaussians (vanilla VAE) the equation simplifies
 - <https://mr-easy.github.io/2020-04-16-kl-divergence-between-2-gaussian-distributions/>

When q is $\mathcal{N}(0, I)$, we get,

Note the notation difference (p and q)

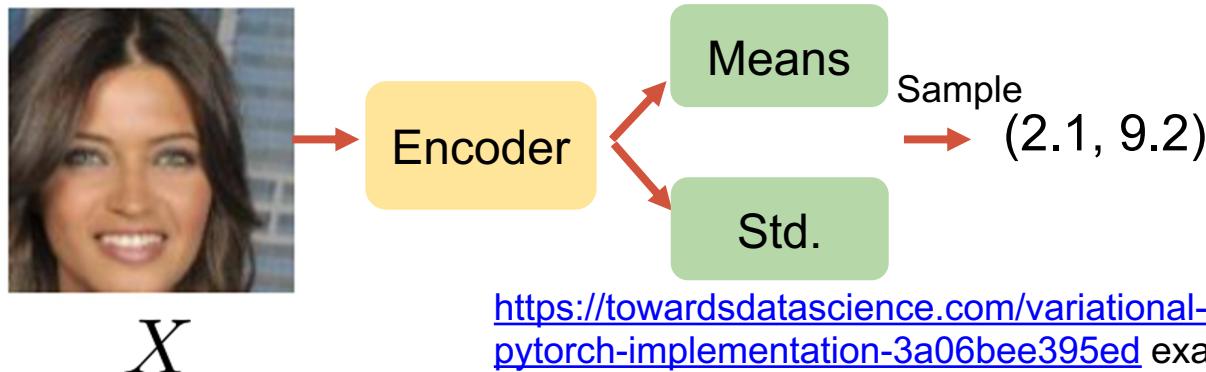
$$D_{KL}(p||q) = \frac{1}{2} [\boldsymbol{\mu}_p^T \boldsymbol{\mu}_p + \text{tr}\{\Sigma_p\} - k - \log |\Sigma_p|]$$

Implementation of KL term

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

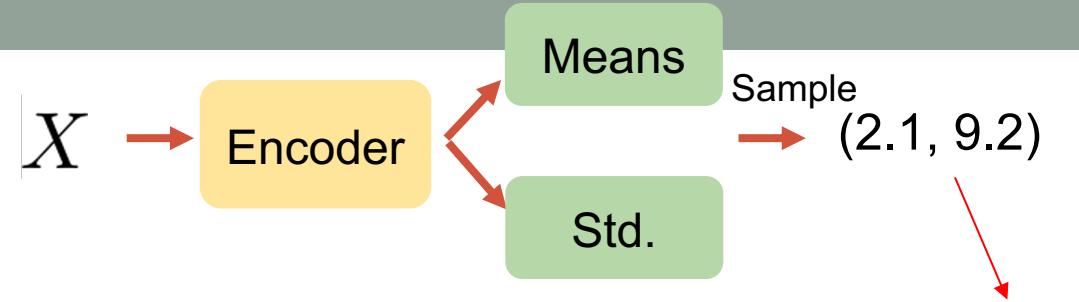
approximate via minibatch

- General form, we have to approximate KL via sampling
- $\text{KL}[q|p] = \mathbb{E}_{z \sim q(z|x)} [\log q(z|x) - \log p(z)]$
- For image x_i in minibatch
 - Compute $q(z|x)$ by passing through encoder network
 - Sample $z_i \sim q(z|x)$
 - Accumulate $[\log q(z_i|x) - \log p(z_i)]$



<https://towardsdatascience.com/variational-autoencoder-demystified-with-pytorch-implementation-3a06bee395ed> example code

Last Problem



$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

Estimate in minibatch

When we need to compute the gradient with respect to the network parameters (I.e., ϕ, θ) for SGD, we're struck because we can't backprop through the sampling for computing $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}$

Example:

$$z = y^2$$

$$z \sim \mathcal{N}(y, 1)$$

$$z \sim q_\phi(z \mid x)$$

$$\frac{dz}{dy} = 2y$$

$$\frac{dz}{dy} = ??$$

$$\frac{dz}{d\phi} = ??$$

Reparameterization Trick

$$z \sim \mathcal{N}(\mu, \sigma)$$

$$z = \epsilon_i \times \sigma + \mu$$

$$\frac{\partial z}{\partial \mu} = ??$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

$$\frac{\partial z}{\partial \sigma} = ??$$

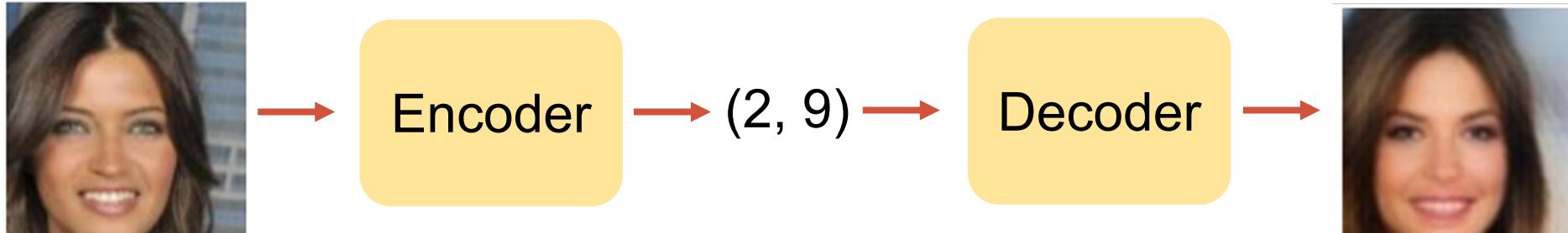
$$\frac{\partial z}{\partial \mu} = 1$$

$$\frac{\partial z}{\partial \sigma} = \epsilon_i$$

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

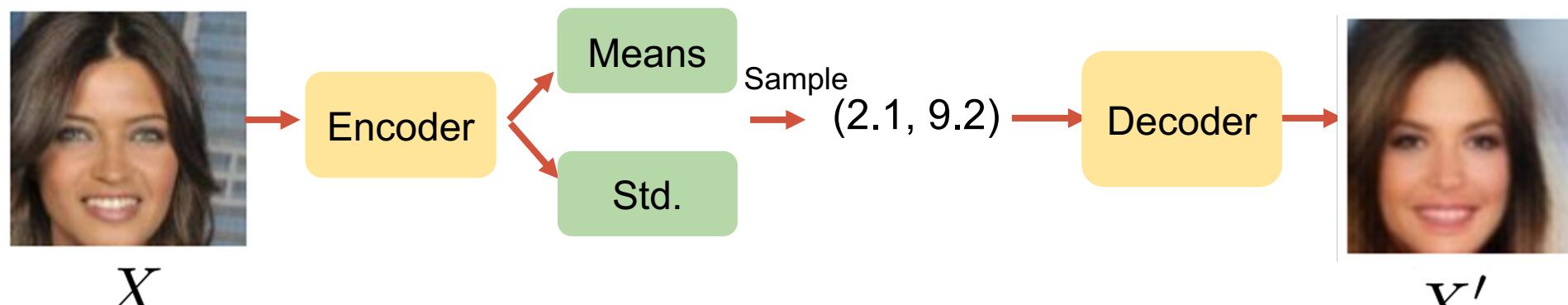
$$\iff \min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x} \mid \hat{\mathbf{z}})]$$

$$\hat{\mathbf{z}} = \epsilon \cdot \sigma + \mu, \quad p(\epsilon) \sim \mathcal{N}(0, I)$$


 X

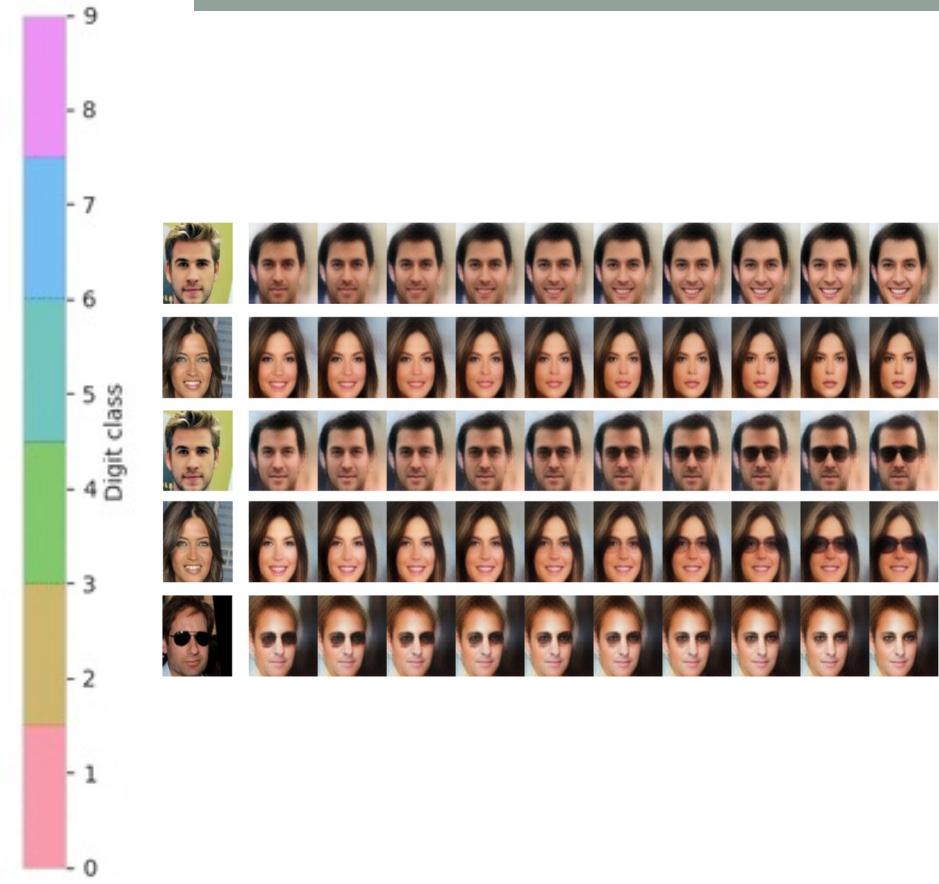
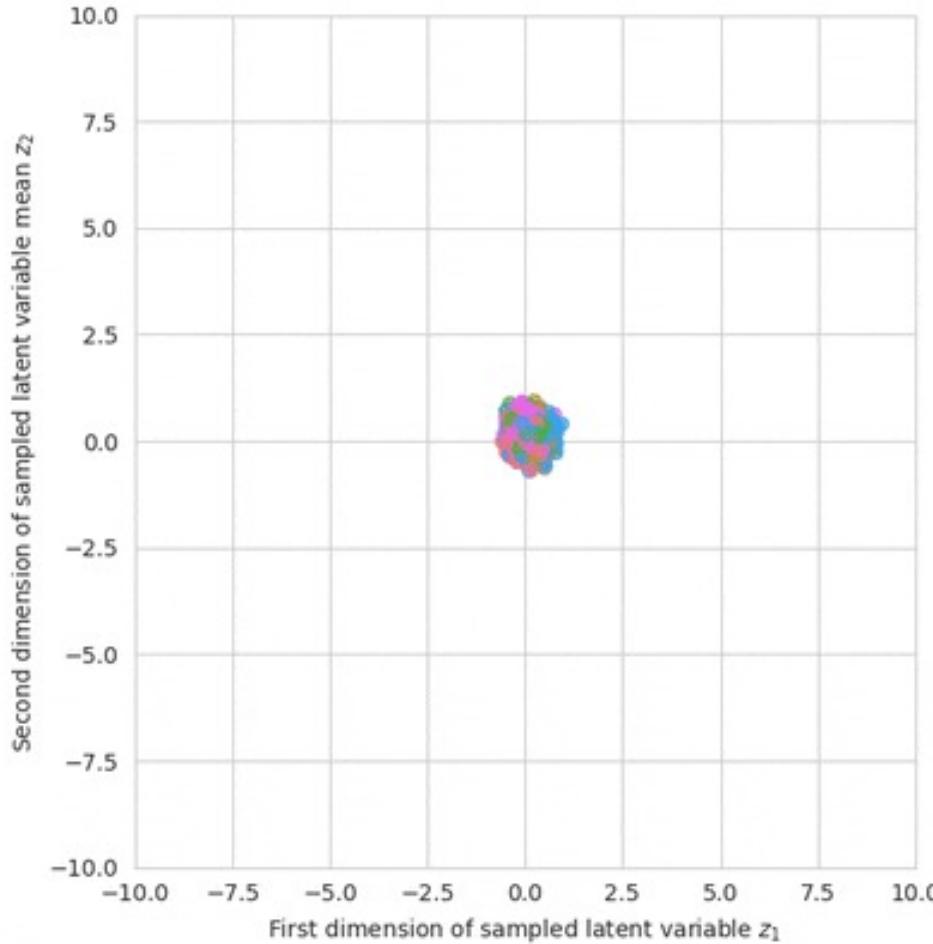
Standard AE Loss

$$\|X - X'\|^2$$

 X'

 X

$$\text{Variational AE Loss} \approx \|X - X'\|^2 + \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel \mathcal{N}(0, I)]$$

More info: <https://arxiv.org/abs/1606.05908>



Notes: when we compresses data, meaningful “codes” tend to emerge.
The $P(z) \sim N(0,1)$ clusters the data and allow for easy interpolation between images

VAE today

- Research focuses on disentangle meaning into different dimension of z (z_1 = hair color, z_2 = glasses, z_3 = age)
 - Controllable generation
- Improve quality of images

Disentangling Representation: β -VAE

We want compact meaningful representation.

But also disentangled representation

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Skin color

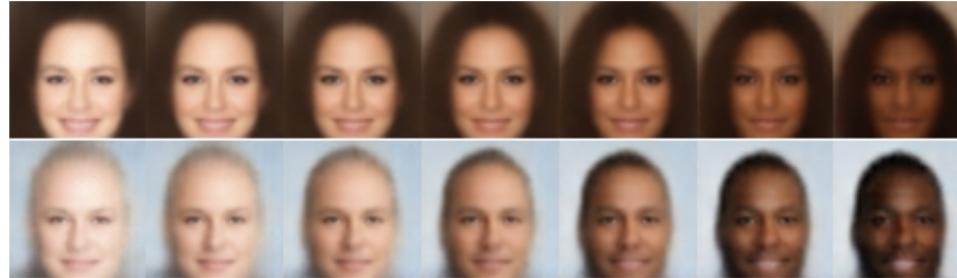
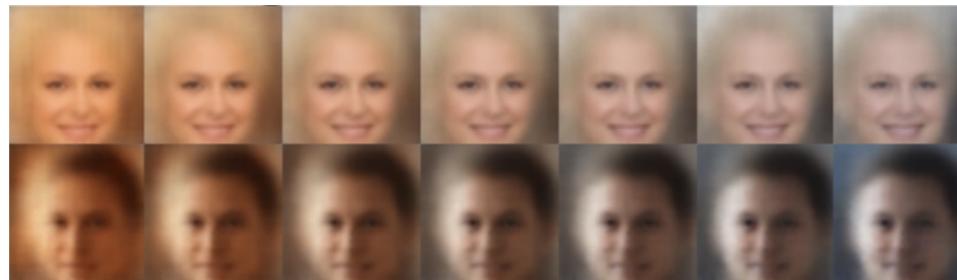
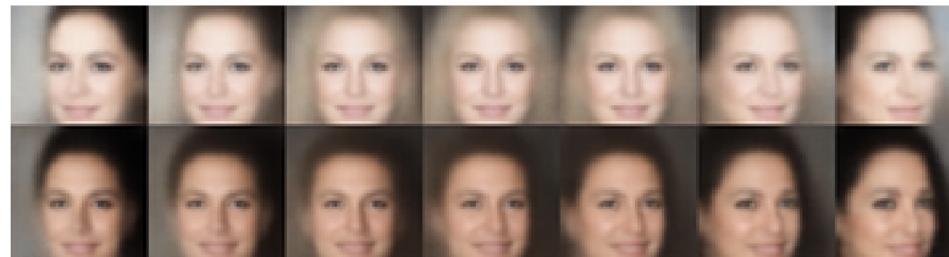


Image
Saturation



Head
Angle



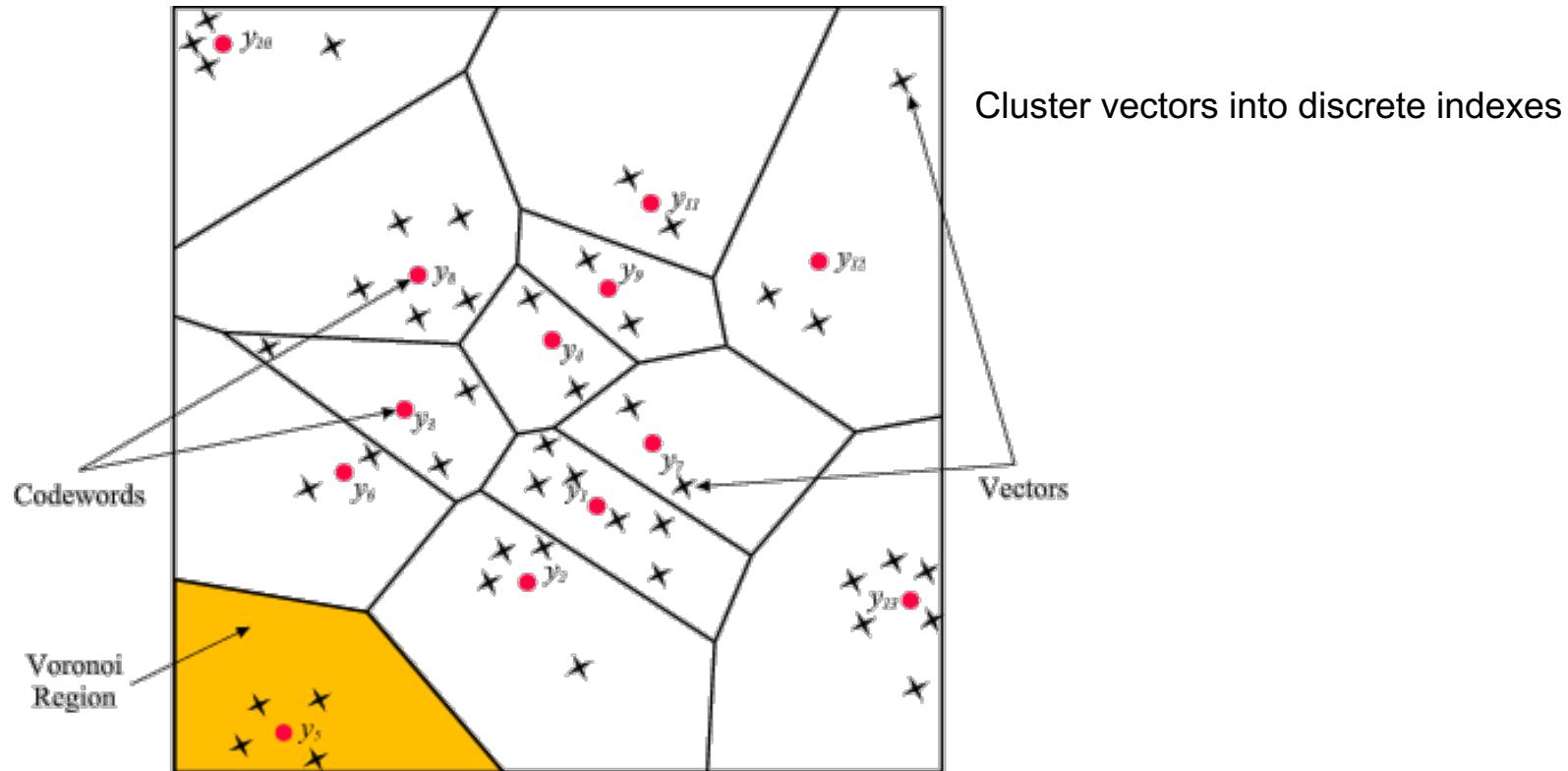
Age / Gender



VQ-VAE

<https://arxiv.org/abs/1711.00937>

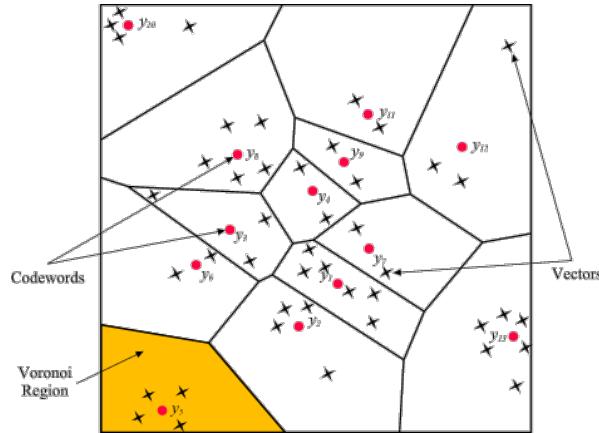
- Learn discrete z by using vector quantization



<http://www.mqasem.net/vectorquantization/vq.html>

VQ-VAE

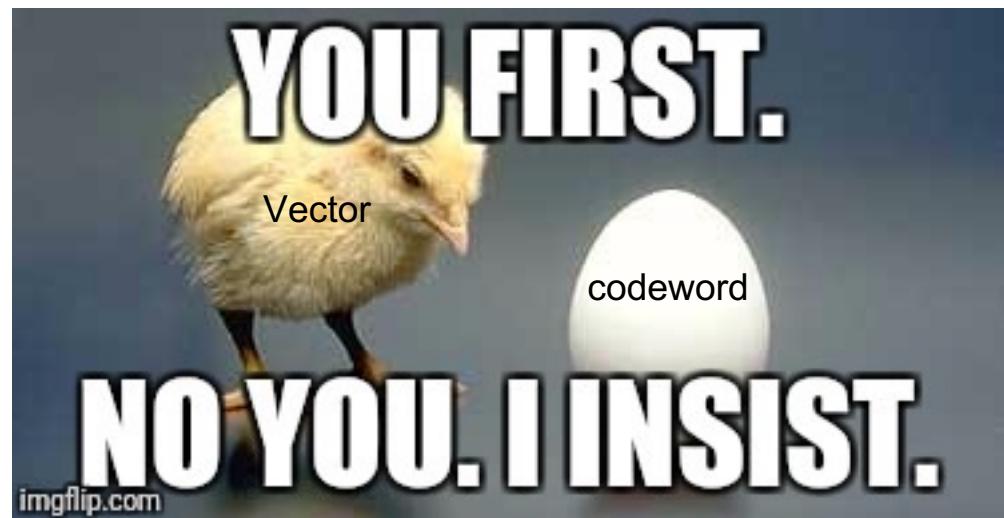
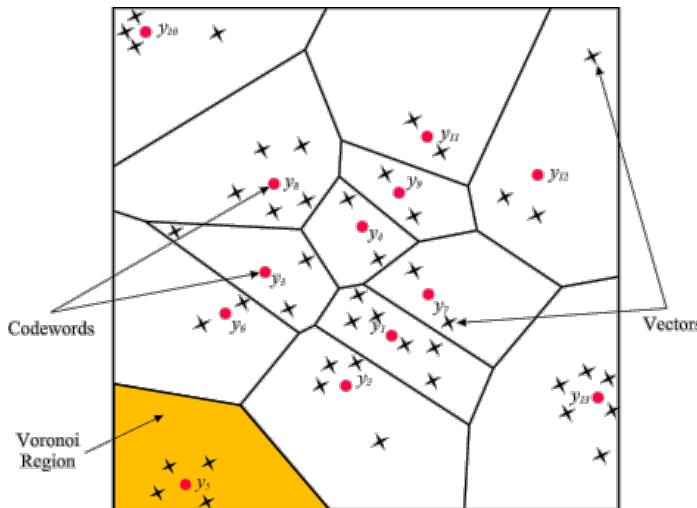
- An image can have multiple codes
- Encoder produces multiple vectors -> multiple codes
 - Ex. Image encodes into grid of NxN vector
- 32x32 grid vectors with 512 codes each
 - $= 512^{32 \times 32}$ possibilities



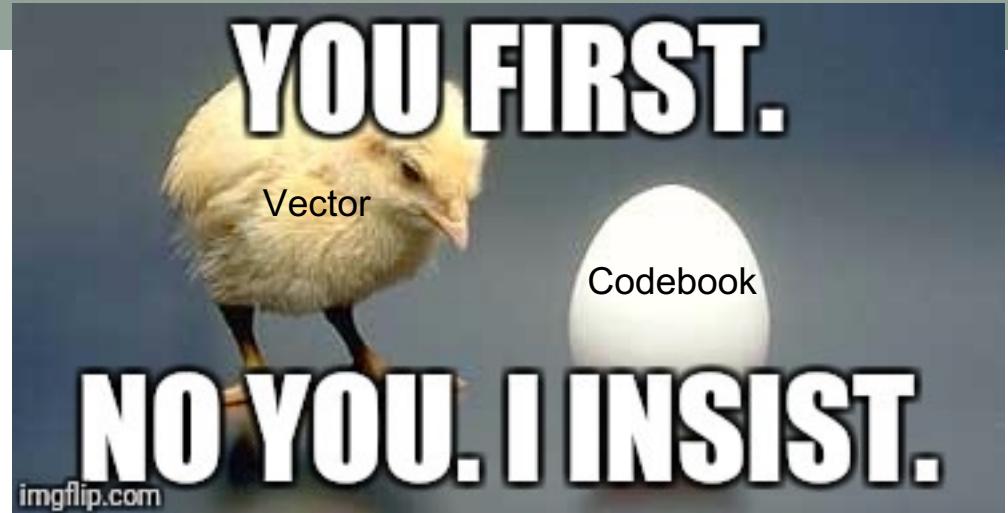
5 (girl), 10 (fire), 3 (road)

Learning the codebook

- Ideally, we want to have the encoded vectors be closed to the codewords
- And we want codewords that are good representations of encoded vectors



Overall loss



1.Codebook can reconstruct

2.Moves codebook to vector

3.Moves vector to codebook

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2,$$

Normal reconstruction loss

Encoded vector

Code book

Hyperparameter to
balance the loss

Stop gradient. No gradient flow
through this term
Pytorch .detach() see homework ☺

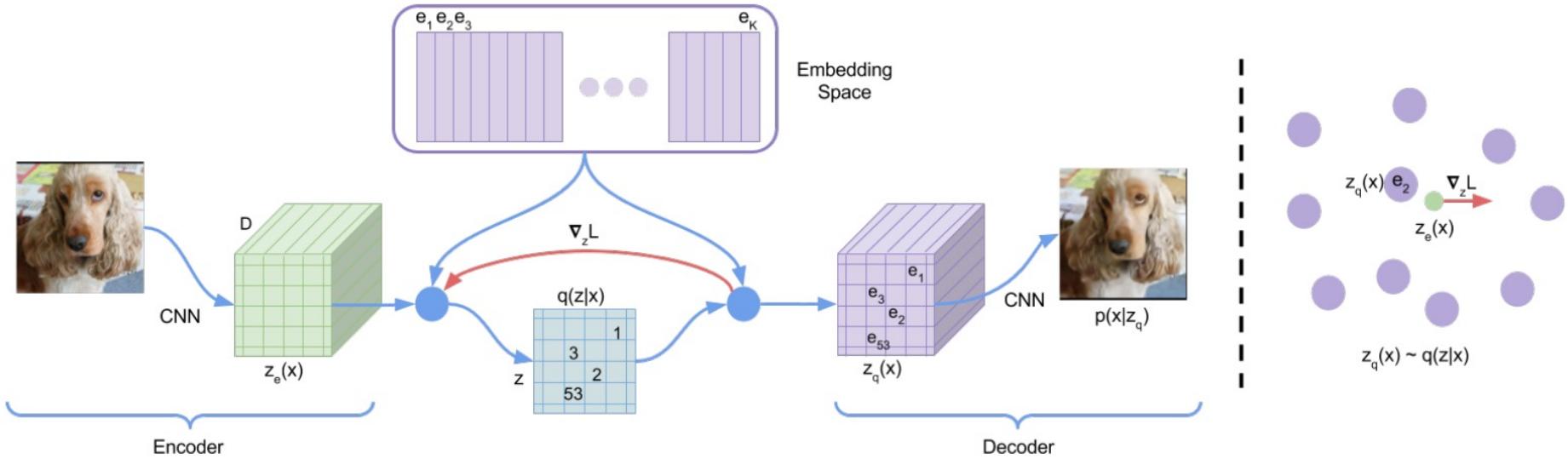


Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2 . The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

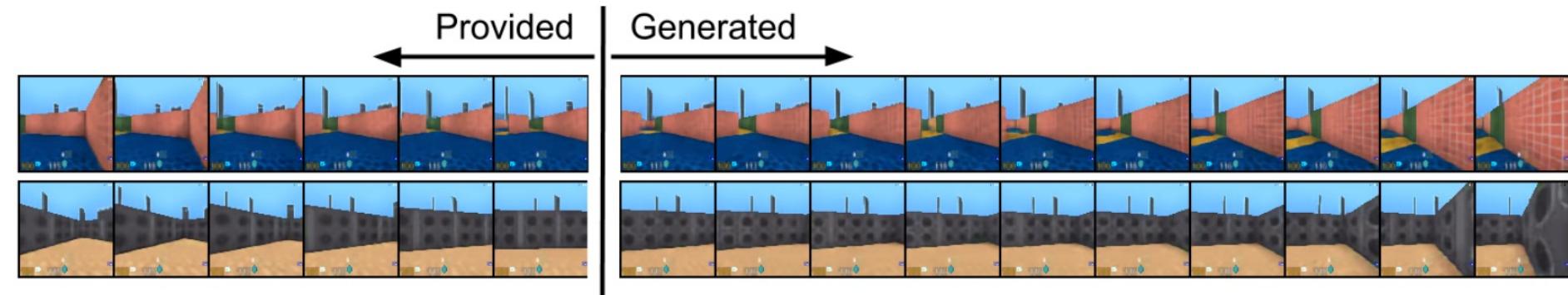
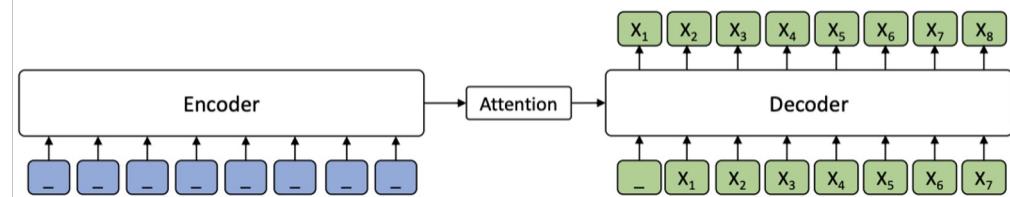


Figure 7: First 6 frames are provided to the model, following frames are generated conditioned on an action. Top: repeated action "move forward", bottom: repeated action "move right".

Dall E



Transformer (GPT3) + VQ VAE

More explanation here

<https://ml.berkeley.edu/blog/posts/dalle2/>



(a) a tapir made of accordian.
a tapir with the texture of an
accordian.

(b) an illustration of a baby
hedgehog in a christmas
sweater walking a dog

(c) a neon sign that reads
“backprop”. a neon sign
that reads “backprop”. backprop
neon sign

(d) the exact same cat on the
top as a sketch on the bottom

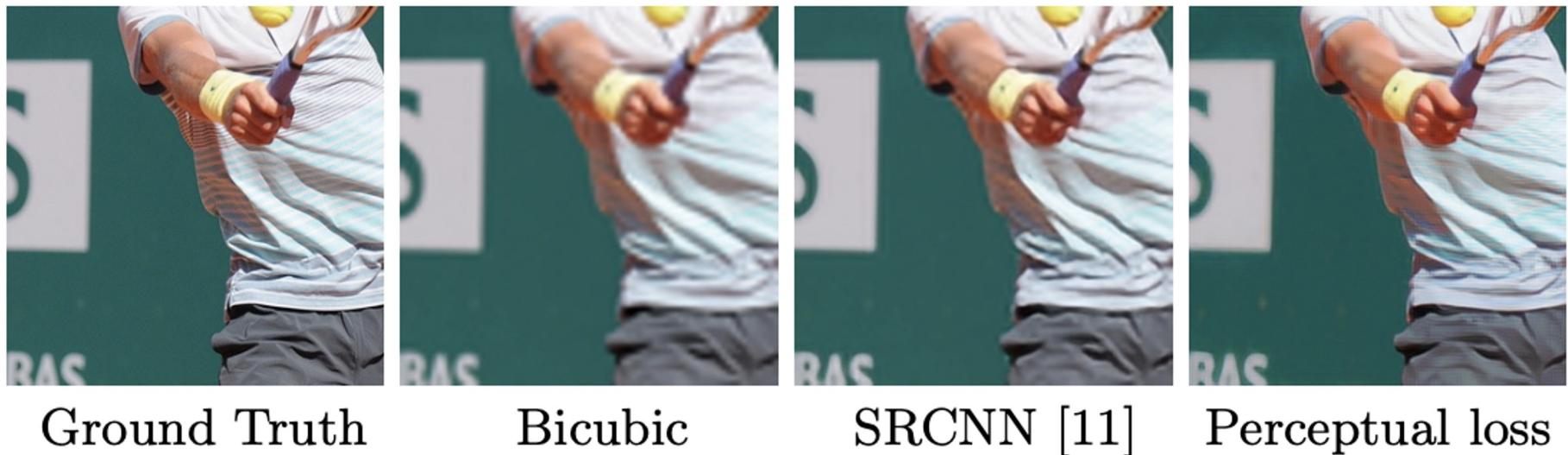
Figure 2. With varying degrees of reliability, our model appears to be able to combine distinct concepts in plausible ways, create anthropomorphized versions of animals, render text, and perform some types of image-to-image translation.

<https://openai.com/blog/dall-e/>

<https://arxiv.org/pdf/2102.12092.pdf>

Perceptual loss

Typical Autoencoder uses per pixel loss (for example L2)
Might not capture the important content



Perceptual loss

Measure **semantic** differences

Use the feature extracted from a pre-trained object classifier as a representation for semantic

Use the L2 between features as an additional loss

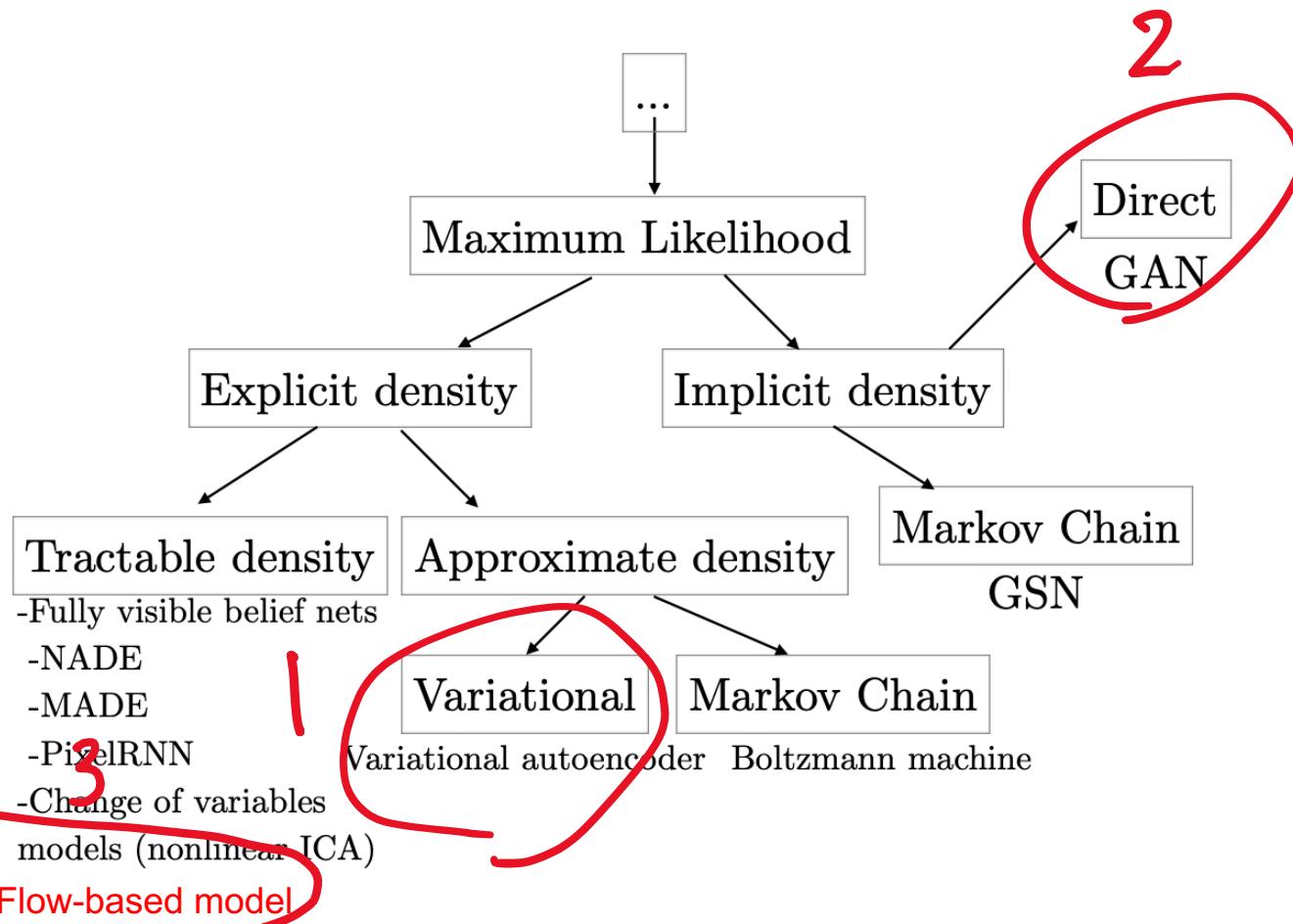
$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Deep generative models

- Autoencoders
 - Variational autoencoders
- GAN
- Diffusion/Flow

Deep Generative Models

- A deep learning model that can be used to generate X



Generating Images

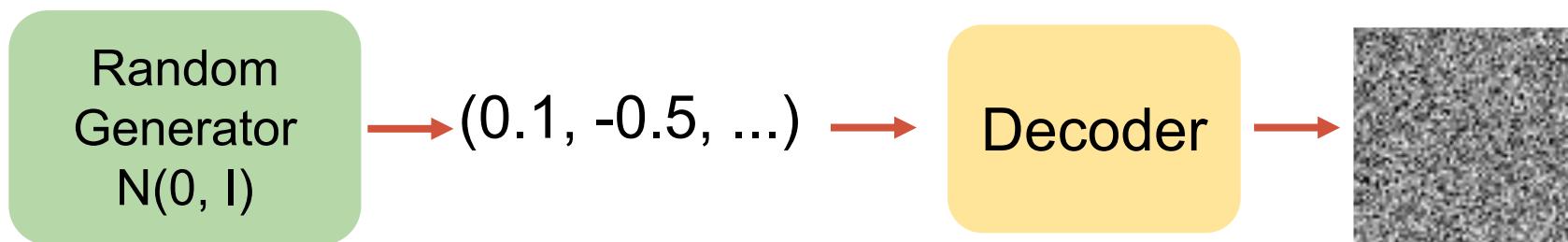
VAE can be used to generate images by feeding numbers into the decoder.

Are there other ways?

Given a collection of, say, faces, how can we generate new faces?

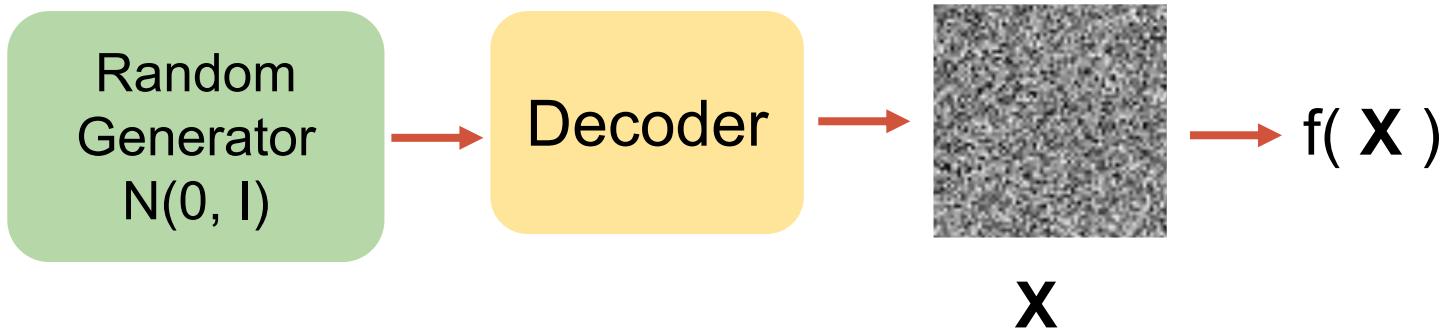
Maybe let's start with a decoder, because we need something that outputs an image anyway.

Generating Realistic Faces



What kind of loss should we use?

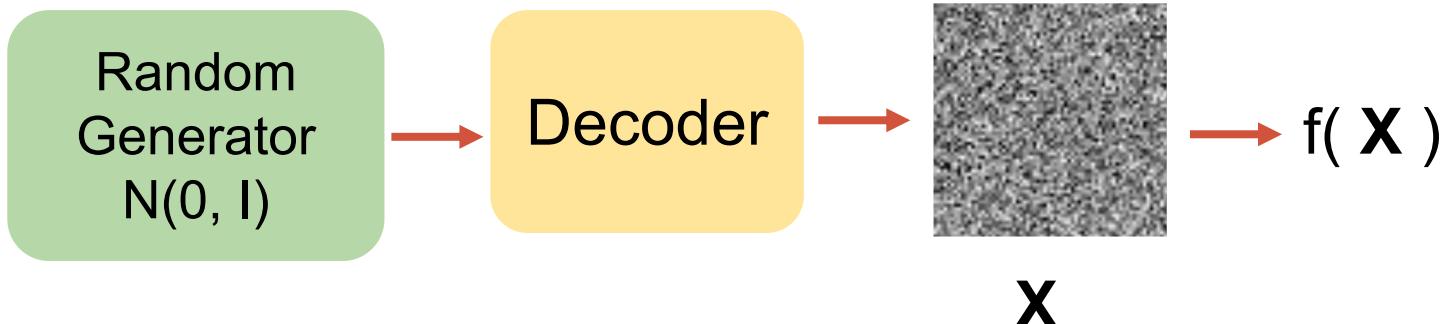
Generating Realistic Faces



We'll be done if we can come up with a smooth function f , that takes in the output image x and outputs $[0, 1]$ where:

$$f(\text{face image}) = 1 \quad f(\text{non-face image}) = 0.2 \quad f(\text{noise texture}) = 0$$

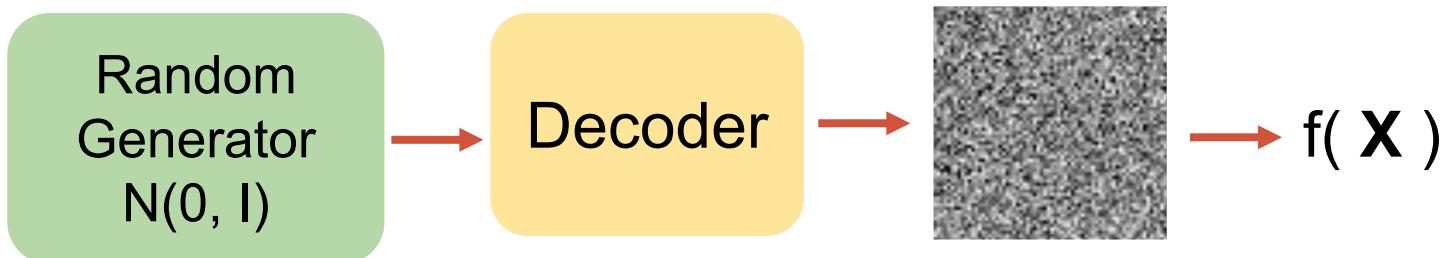
Generating Realistic Faces



$$f(\text{Image of Face}) = 1 \quad f(\text{Image of Tap}) = 0.2 \quad f(\text{Noise Image}) = 0$$

Given this f , we can compute its gradient with respect to the decoder parameters and use SGD to train. Done!

Generating Realistic Faces



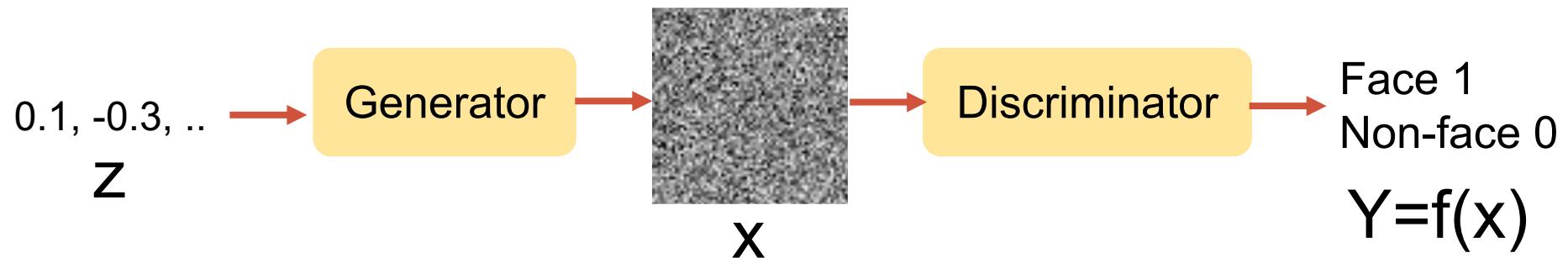
Solution: We will use another network as f . This network will predict the probability of x being a face.

Training this f is a simple supervised learning!

All face photos -> positive examples

All non-face photos -> negative examples

Generating Realistic Faces

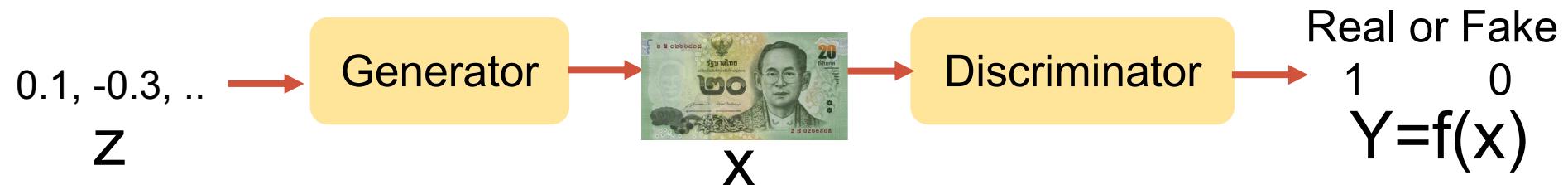


Generative Adversarial Networks (GAN)



- Consider a money counterfeiter
 - He wants to make fake money that looks real
 - There's a police that tries to differentiate fake and real money.
- The counterfeiter is the **adversary** and is **generating** fake inputs. – Generator network
- The police tries to discriminate between fake and real inputs. – Discriminator network

Generative Adversarial Networks (GAN) – minimax loss



- Generator (Money Faker):

- Maximize Y

$$\min_G \mathbb{E}_{\mathbf{z}} [\log(1 - D(G(\mathbf{z})))]$$

- Discriminator (Police):

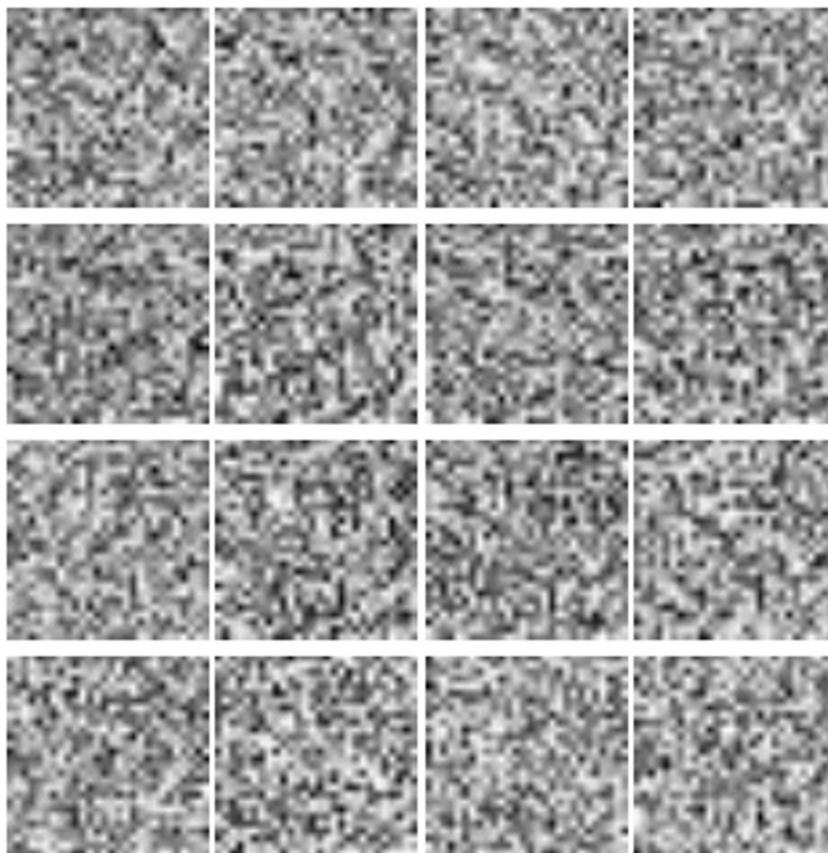
- For real images => Maximize Y
 - For generated images from the faker => Minimize Y

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

This comes from binary cross entropy loss

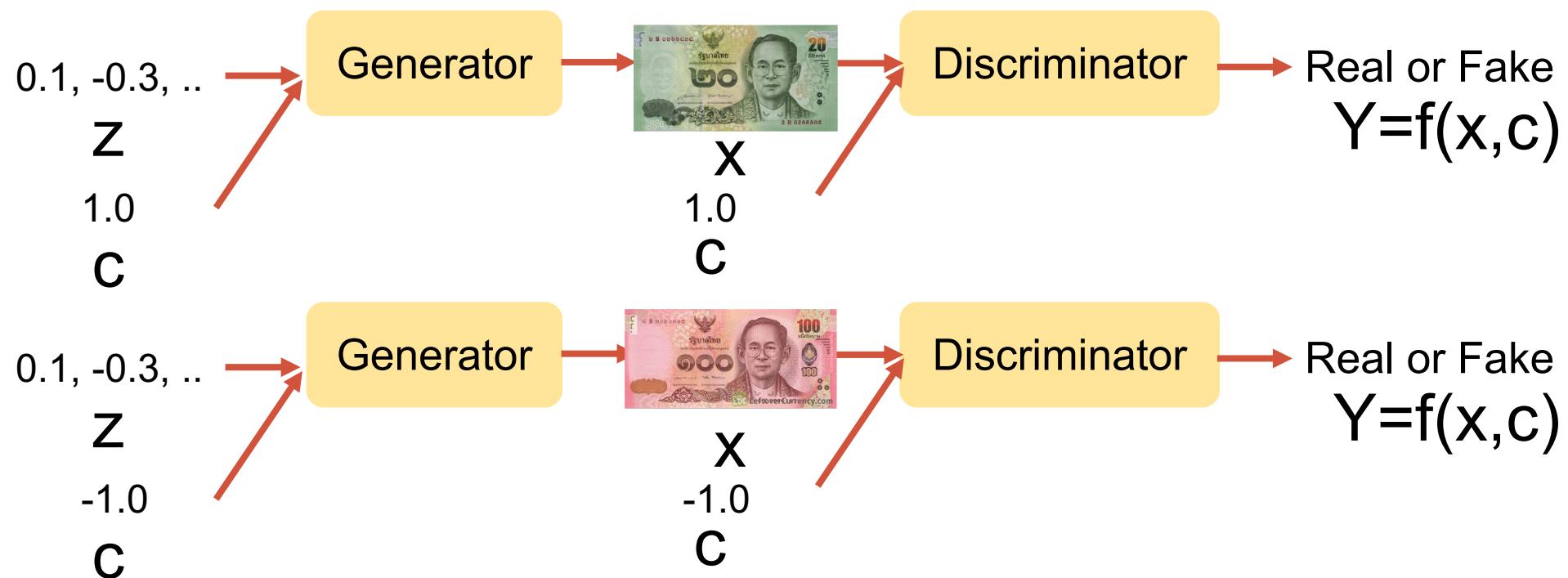
GAN example

Generator output starts from random noise and gets better as we train.



Conditional GAN

- Can condition GAN to generate a certain trait/class



Just concatenate C to the noise and the discriminator input
Discriminator also checks if it's the correct class

Mode collapse

Model only learn a couple types (modes) of inputs



<https://arxiv.org/pdf/1701.07875.pdf>



Ian Goodfellow

@goodfellow_ian

4.5 years of GAN progress on face generation.

arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434

arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196

arxiv.org/abs/1812.04948



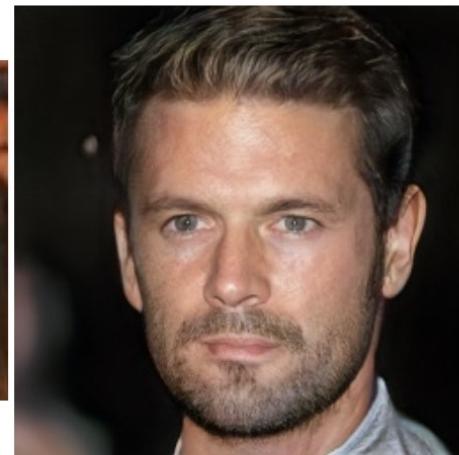
2014



2015



2016



2017

Progressive GAN

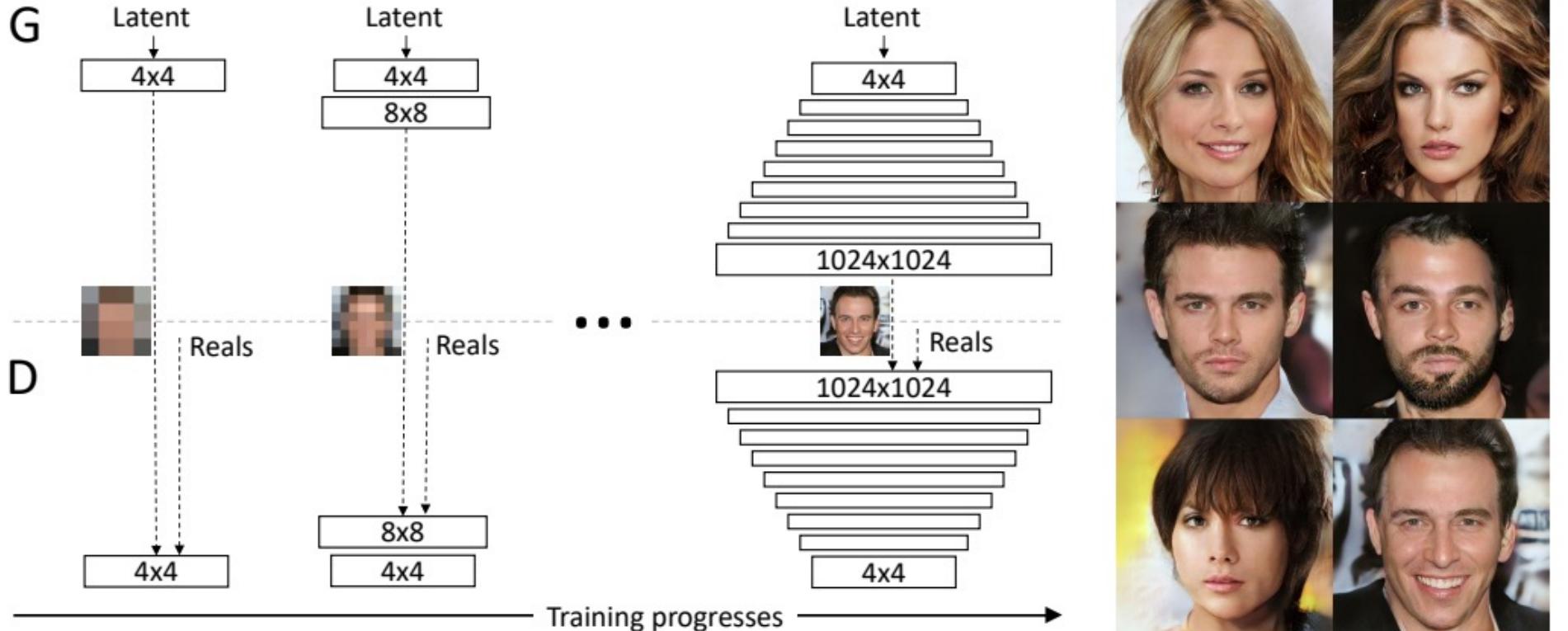


2018
Style GAN

More realistic, higher res, more controllable

Progressive GANS

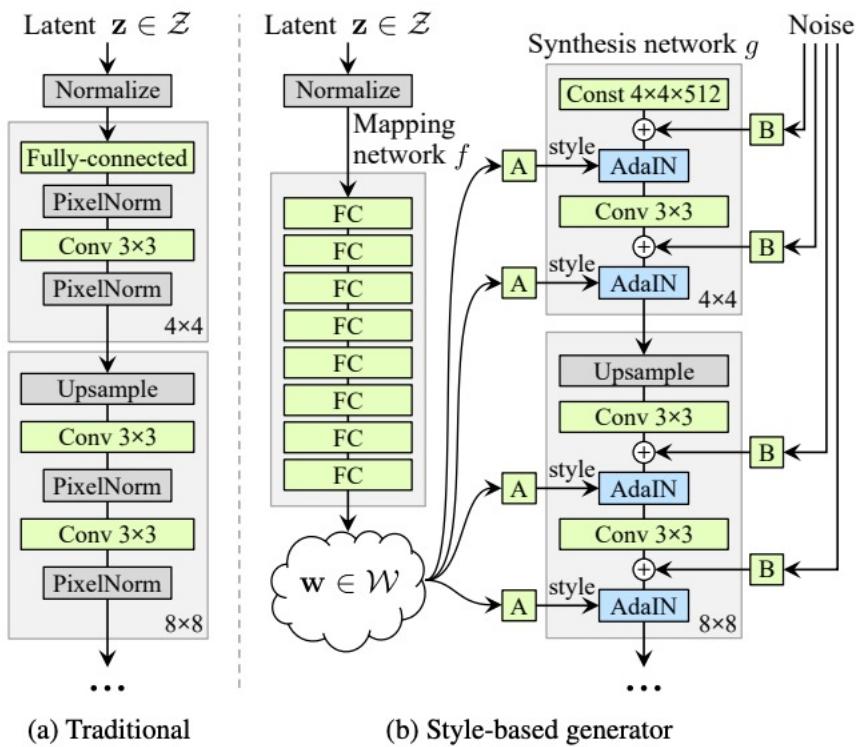
Newer GANs are usually trained from low res to high res



Style-GAN (V1)



- Based on progressive GAN
- Inspired from style transfer network (no actual style transfer here)
- Embedded “Styles” from z
- Use styles to control the generator



$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}, \quad (1)$$

Instance norm

Style value

GAN output in paper



Whom being stillness
consists.



Your GAN output



Extra reading

<https://machinelearningmastery.com/generative-adversarial-network-loss-functions/>

GANs Loss Formulations

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

$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

Discriminator

Generator

| GAN | DISCRIMINATOR LOSS | GENERATOR LOSS |
|----------|---|---|
| MM GAN | $\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$ | $\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$ |
| NS GAN | $\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$ | $\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[\log(D(\hat{x}))]$ |
| * WGAN | $\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d}[D(x)] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$ | $\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$ |
| WGAN GP | $\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g}[(\nabla D(\alpha x + (1 - \alpha)\hat{x}) _2 - 1)^2]$ | $\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$ |
| * LS GAN | $\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})^2]$ | $\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[(D(\hat{x} - 1))^2]$ |
| * DRAGAN | $\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)}[(\nabla D(\hat{x}) _2 - 1)^2]$ | $\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$ |
| BEGAN | $\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d}[x - \text{AE}(x) _1] - k_t \mathbb{E}_{\hat{x} \sim p_g}[\hat{x} - \text{AE}(\hat{x}) _1]$ | $\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\hat{x} - \text{AE}(\hat{x}) _1]$ |

Are GANs Created Equal? A Large-Scale Study [Lucic et al. 2018]

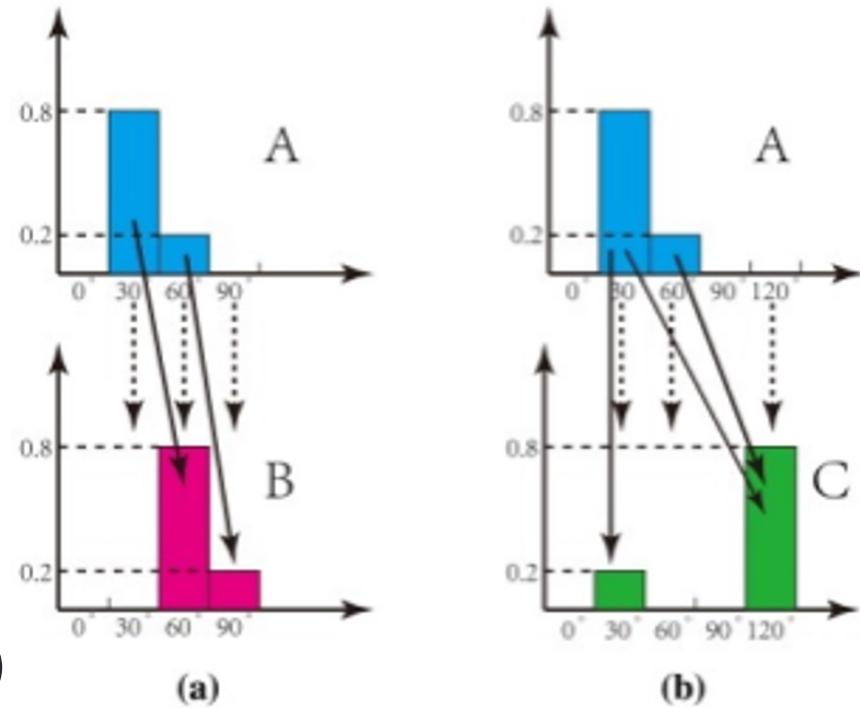
Wasserstein GAN (WGAN)

Wasserstein distance? (Earth mover distance)

Energy required to move mass to make two distributions look the same

WGAN
approximates and minimizes
this distance

(Note vanilla GAN minimizes
Jensen-Shannon divergence)



WGAN

Discriminator as a **critic** (no fake/real sigmoid) but output a score (linear activation at output layer)

WD has better gradient and convergence

| | Discriminator/Critic | Generator |
|------|---|--|
| GAN | $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)})))]$ | $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (D(G(\mathbf{z}^{(i)})))$ |
| WGAN | $\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$ | $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$ |

WGAN

Approximating EMD requires the critic model to be a k-Lipschitz function

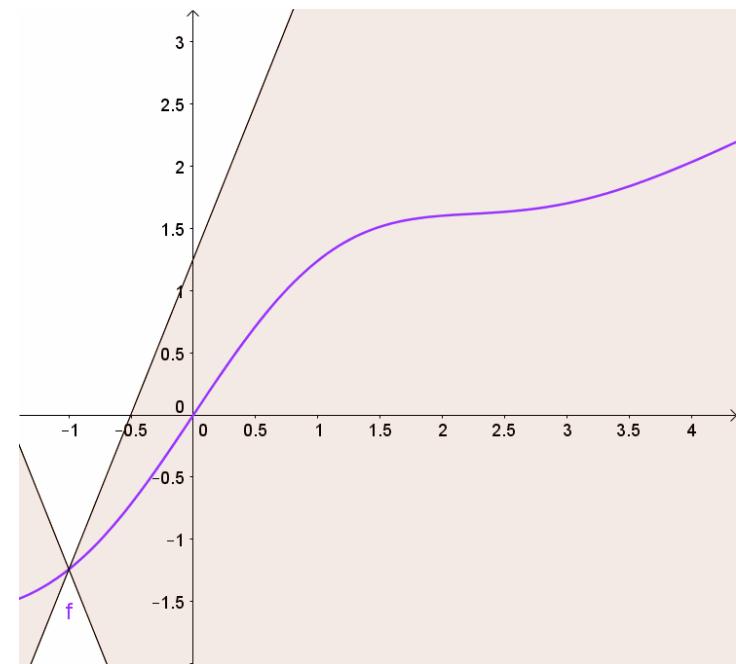
k-Lipschitz?

Bounded in slope of k

$$|f(a) - f(b)| < k |a - b|$$

Example

$f(x) = 5x$ is 5-Lipschitz



https://en.wikipedia.org/wiki/Lipschitz_continuity

WGAN to WGAN-GP

To make k-Lipschitz

WGAN caps the weights of all layers to 1

WGAN-GP improves and add **Gradient Penalty** to reduce the weights instead

A differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere.

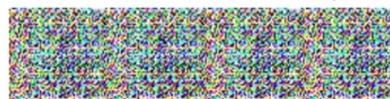
$$L = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{Our gradient penalty}}.$$

DCGAN**LSGAN****WGAN (clipping)****WGAN-GP (ours)**

Baseline (G : DCGAN, D : DCGAN)



G : No BN and a constant number of filters, D : DCGAN



G : 4-layer 512-dim ReLU MLP, D : DCGAN



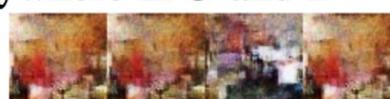
No normalization in either G or D



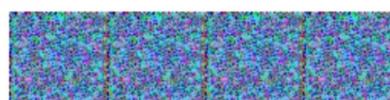
Gated multiplicative nonlinearities everywhere in G and D



tanh nonlinearities everywhere in G and D

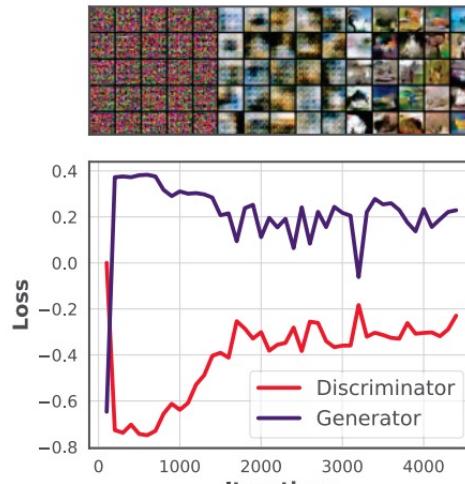


101-layer ResNet G and D

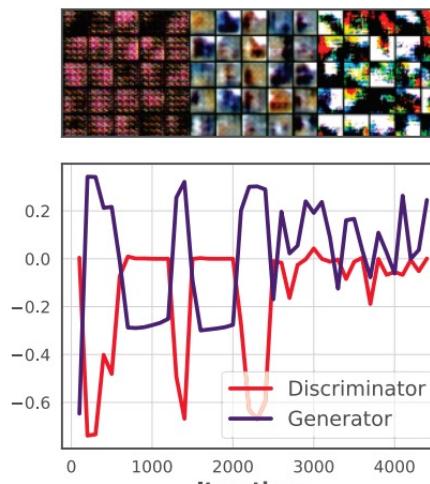


GAN without competition

- Minimax nature of GAN makes it hard to train
 - Optimize the duality gap instead



(a) Convergence



(b) Divergence

Figure 1. Loss curves throughout the training progress of WGAN
on the CIFAR-10 dataset.

Generative Minimization Networks: Training GANs Without Competition

<https://arxiv.org/pdf/2103.12685.pdf>

On Characterizing GAN Convergence Through Proximal Duality Gap

<http://proceedings.mlr.press/v139/sidheekh21a/sidheekh21a.pdf>

$$\begin{aligned} DG^\lambda(\theta_d, \theta_g) &= V_{D_w}(\theta_g) - V_{G_w}^\lambda(\theta_d), \text{ where} \\ V_{D_w}(\theta_g) &= \max_{\theta'_d \in \Theta_D} V(D_{\theta'_d}, G_{\theta_g}) \\ V_{G_w}^\lambda(\theta_d) &= \min_{\theta'_g \in \Theta_G} V^\lambda(D_{\theta_d}, G_{\theta'_g}) \end{aligned}$$

Estimate V_D by optimizing discriminator to estimate worst case discriminator

Estimate V_G by optimizing generator to estimate worst case generator

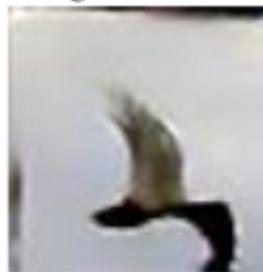
Optimize the gap

Other uses

- StackGAN: text to photo

(a) StackGAN
Stage-I
64x64
images

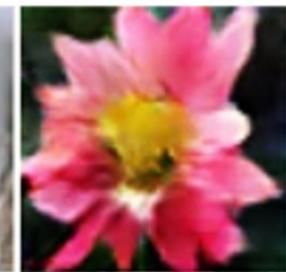
This bird is white with some black on its head and wings, and has a long orange beak



This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face



This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments



(b) StackGAN
Stage-II
256x256
images

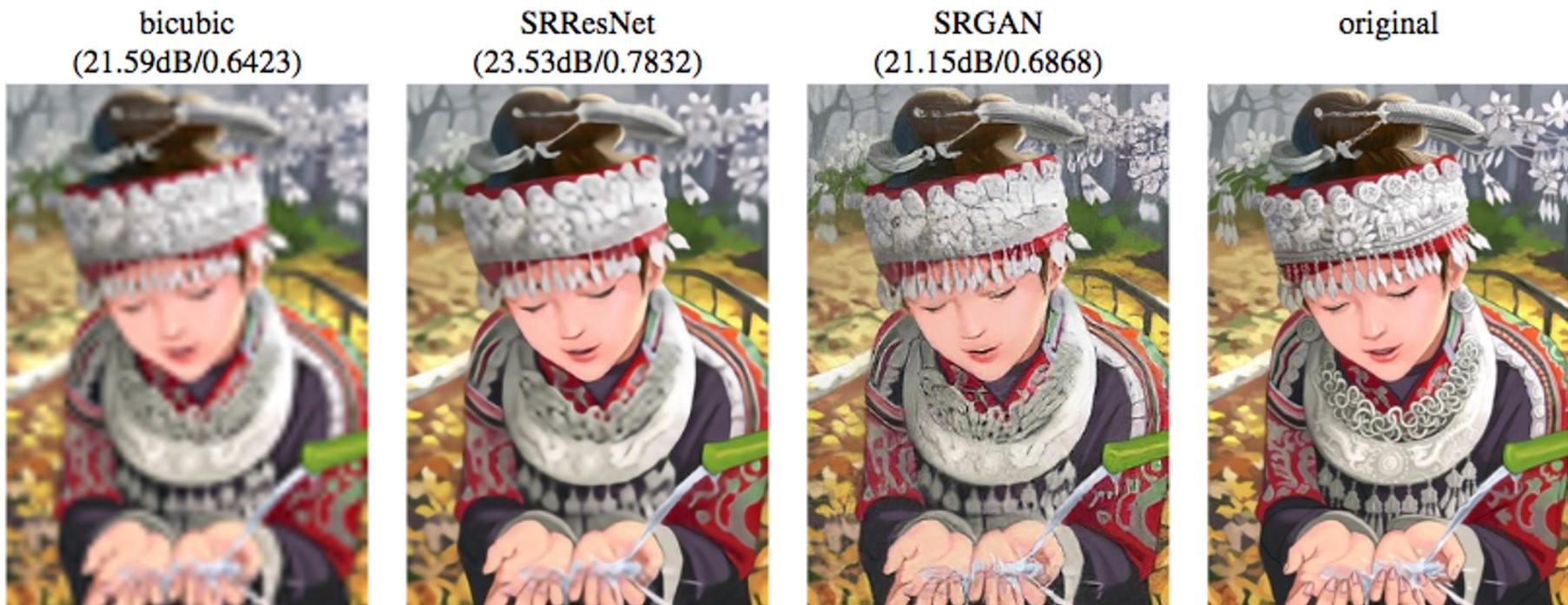


(c) Vanilla GAN
256x256
images



SRGAN

- <https://arxiv.org/pdf/1609.04802.pdf>

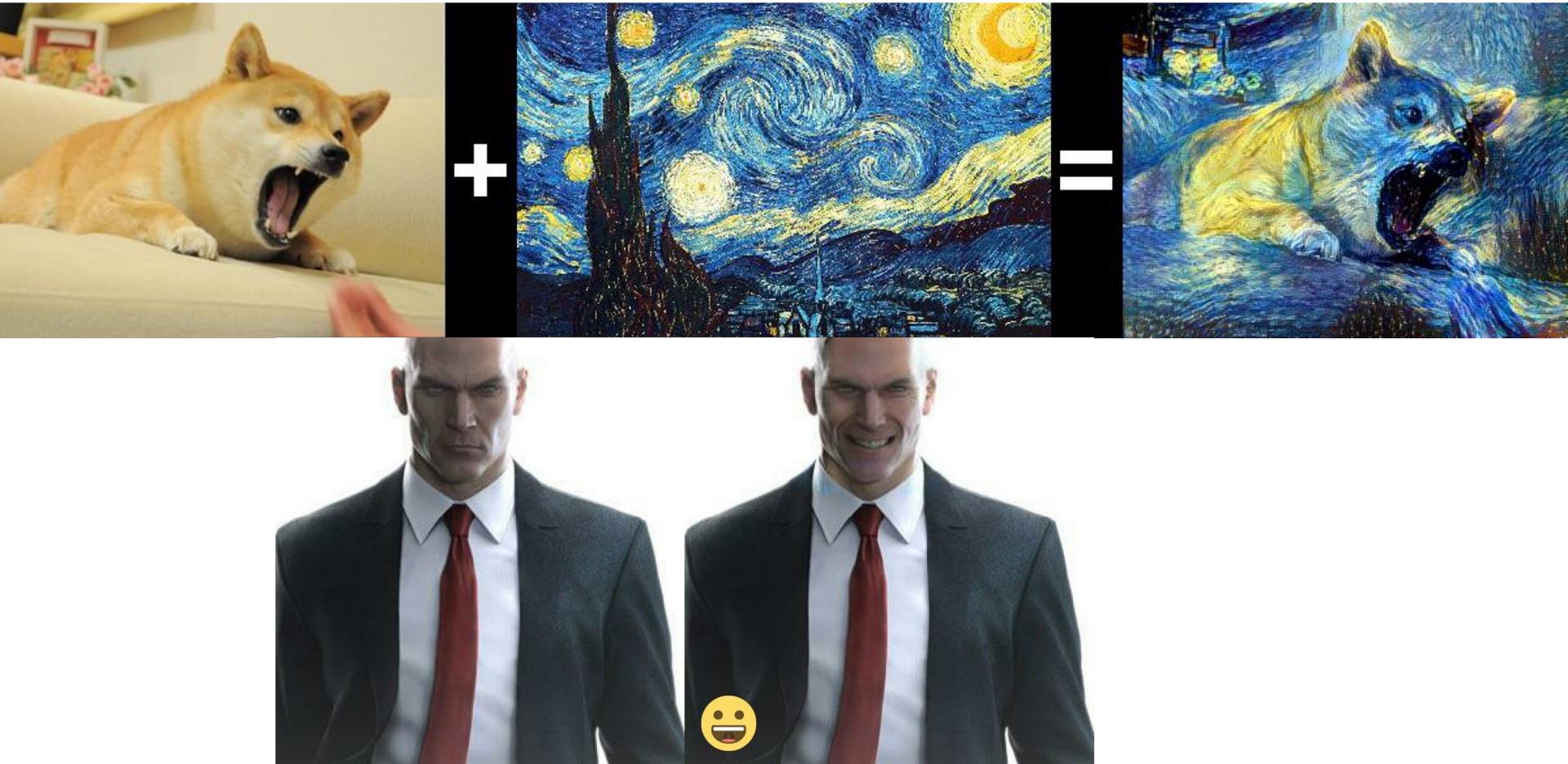


Content-aware Filling



Globally and Locally Consistent Image Completion [Iizuka et al.,

Style transfer with cycleGAN



Zebras ↗ **Horses**



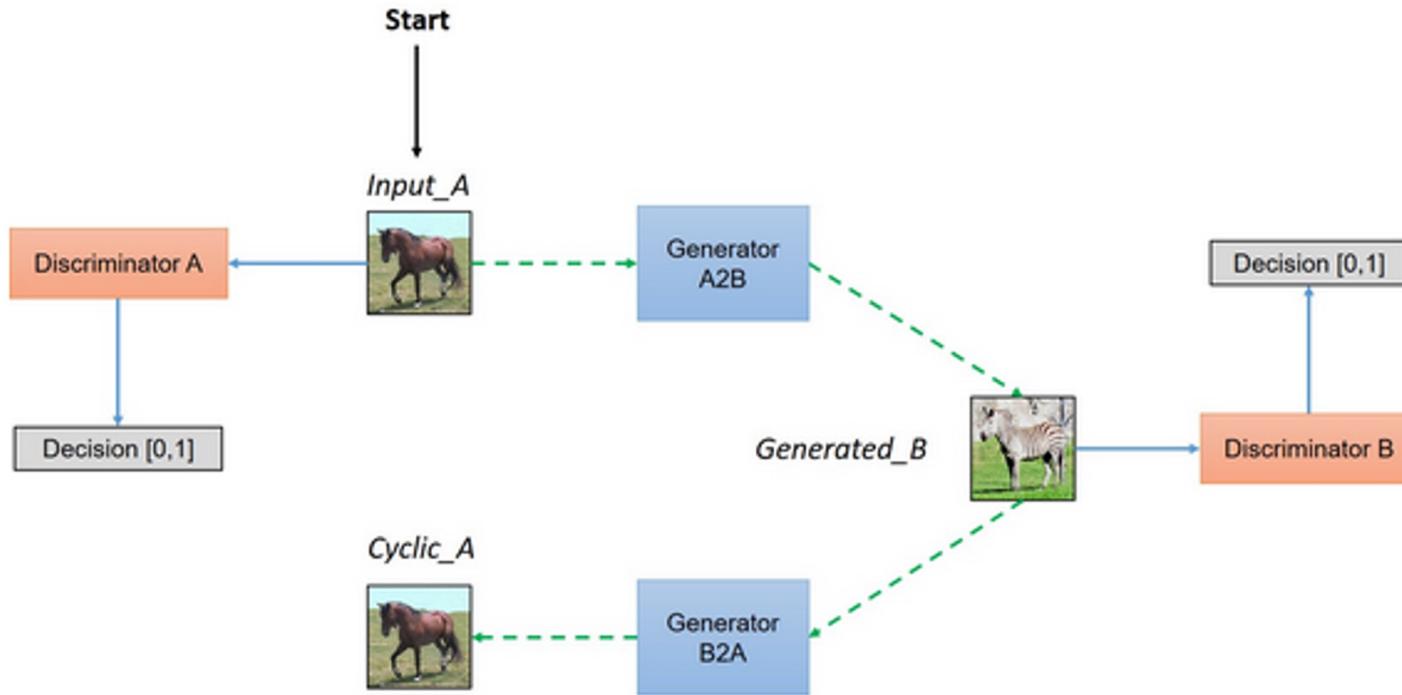
zebra → horse



horse → zebra

Cycle consistency

Note, you don't
need a paired
dataset



Cycle consistency

Machine translation with no parallel text

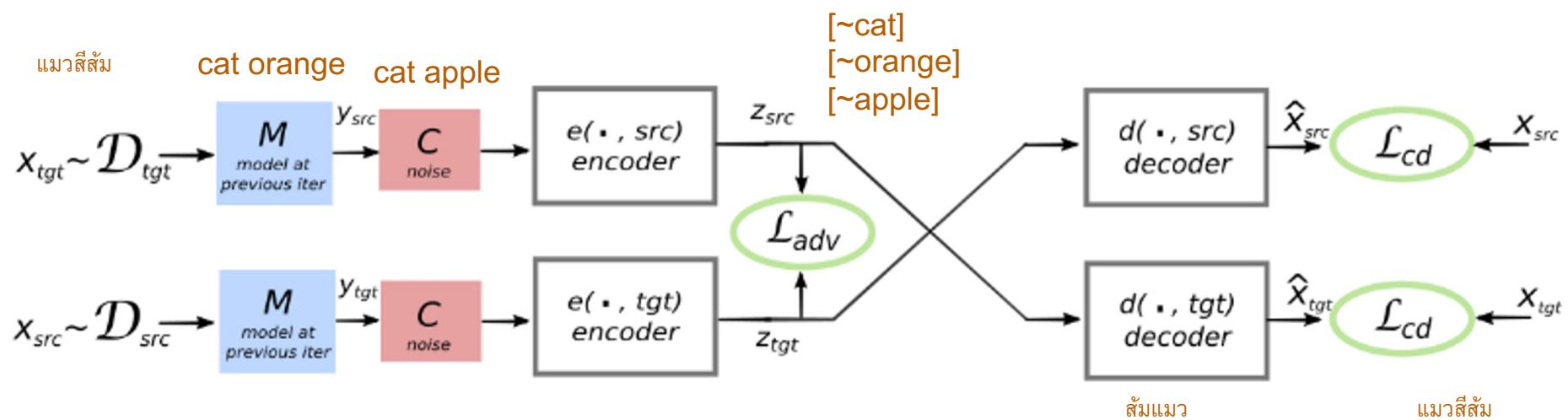
- MT usually requires parallel text

This is a cat

นี่คือแมว

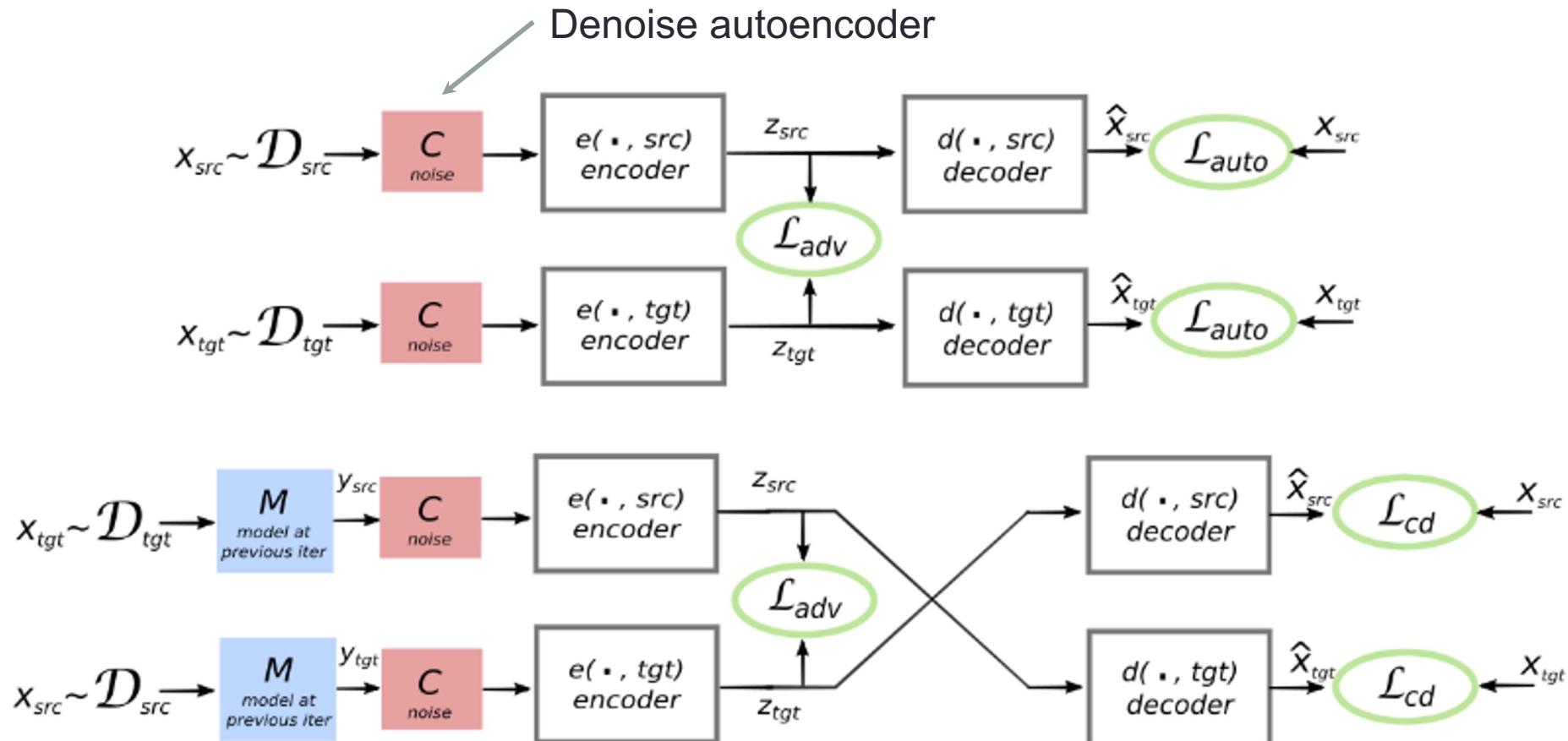
- Most of the time we don't have parallel text. Just text.
- Can we still do MT?
 - Use GANs + Autoencoders

Denoise autoencoder



Use GAN Loss (\mathcal{L}_{adv}) to enforce that source and target language pairs share the same distributions.

Then enforce the translation distribution matches.



Results

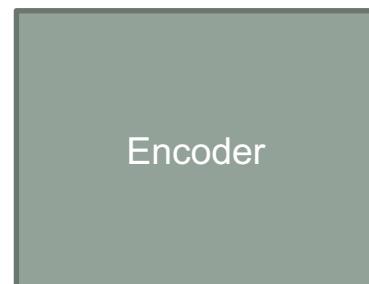
| | Multi30k-Task1 | | | | WMT | | | |
|--------------------------|----------------|-------|-------|-------|-------|-------|-------|-------|
| | en-fr | fr-en | de-en | en-de | en-fr | fr-en | de-en | en-de |
| Supervised | 56.83 | 50.77 | 38.38 | 35.16 | 27.97 | 26.13 | 25.61 | 21.33 |
| word-by-word | 8.54 | 16.77 | 15.72 | 5.39 | 6.28 | 10.09 | 10.77 | 7.06 |
| word reordering | - | - | - | - | 6.68 | 11.69 | 10.84 | 6.70 |
| oracle word reordering | 11.62 | 24.88 | 18.27 | 6.79 | 10.12 | 20.64 | 19.42 | 11.57 |
| Our model: 1st iteration | 27.48 | 28.07 | 23.69 | 19.32 | 12.10 | 11.79 | 11.10 | 8.86 |
| Our model: 2nd iteration | 31.72 | 30.49 | 24.73 | 21.16 | 14.42 | 13.49 | 13.25 | 9.75 |
| Our model: 3rd iteration | 32.76 | 32.07 | 26.26 | 22.74 | 15.05 | 14.31 | 13.33 | 9.64 |

Adversarial debiasing

- Use the discriminator to force the encoder to drop bias information



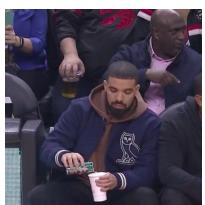
Training data



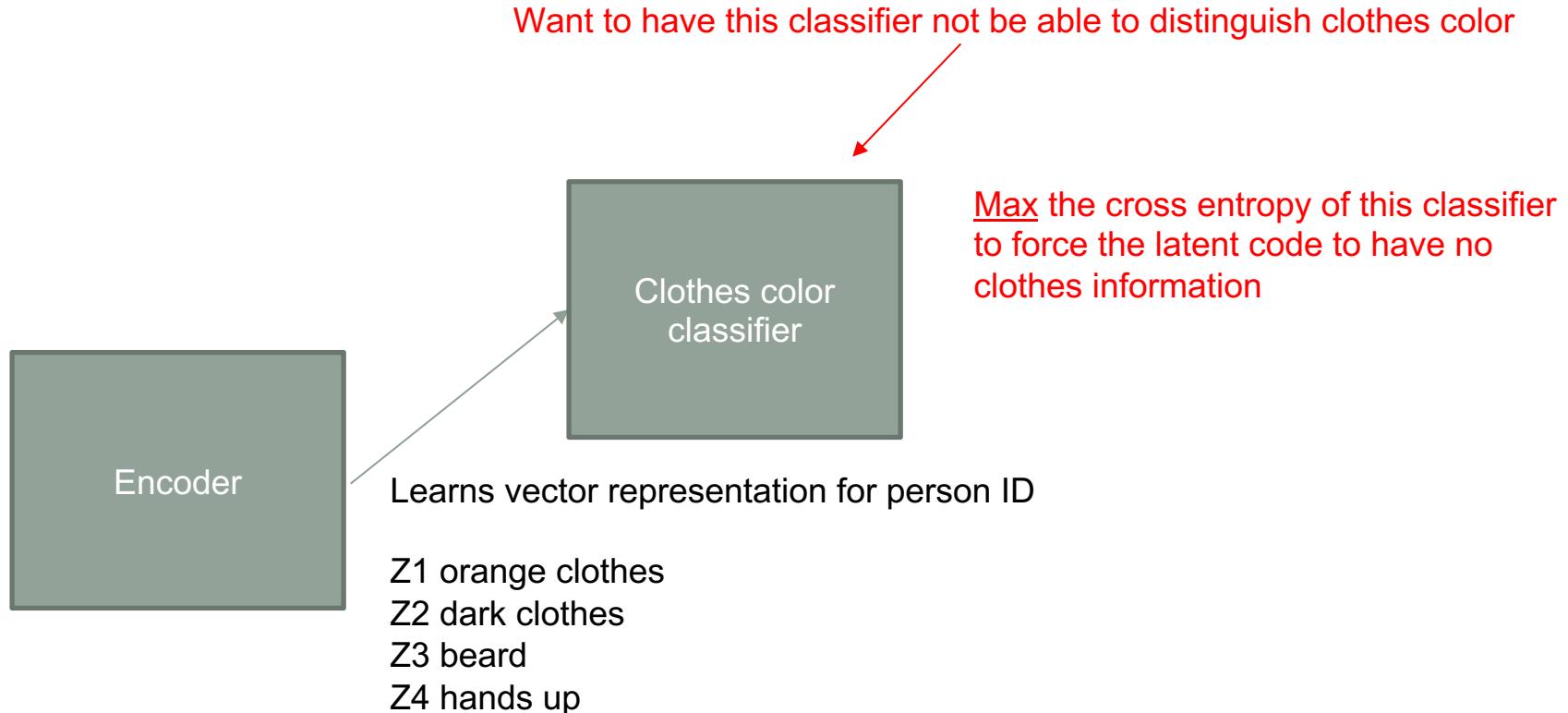
Learns vector representation for person ID

Z1 orange clothes
Z2 dark clothes
Z3 beard
Z4 hands up

How to remove unwanted information in latent code?



Adversarial debiasing



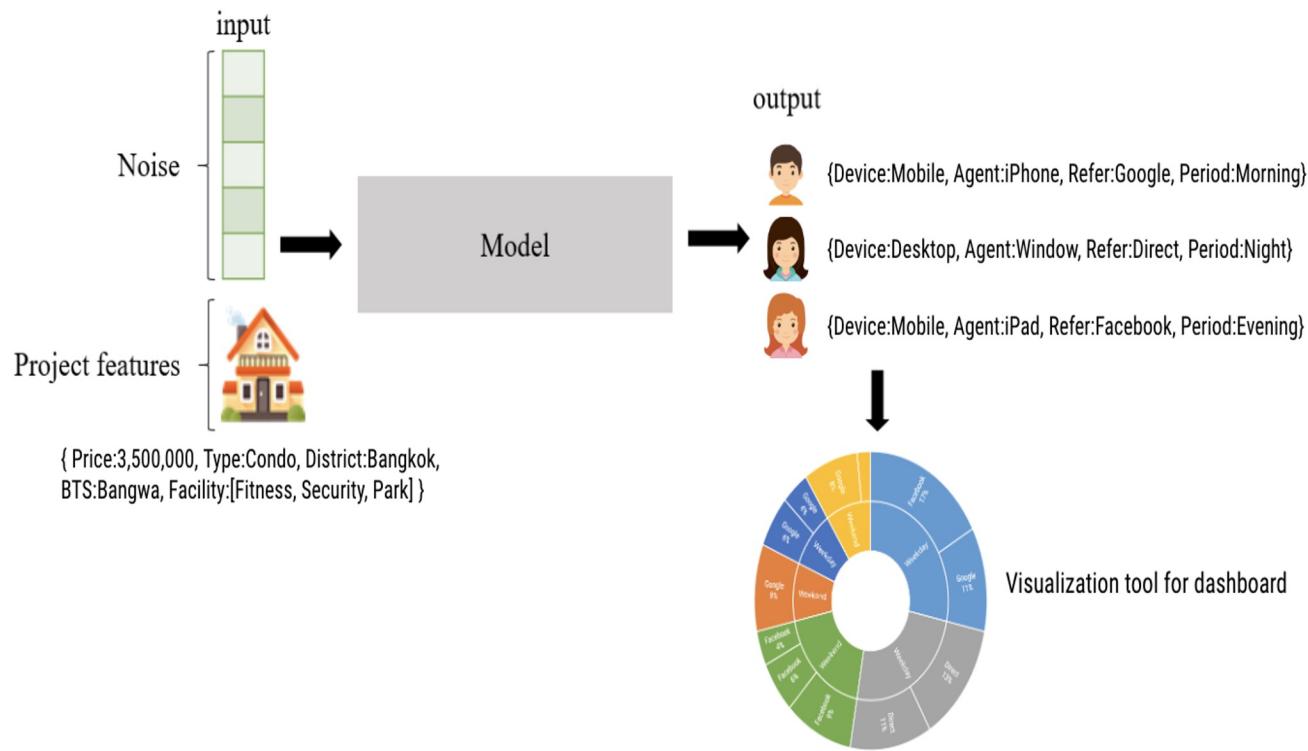
Reading <https://arxiv.org/pdf/1801.07593.pdf>

Example application usages

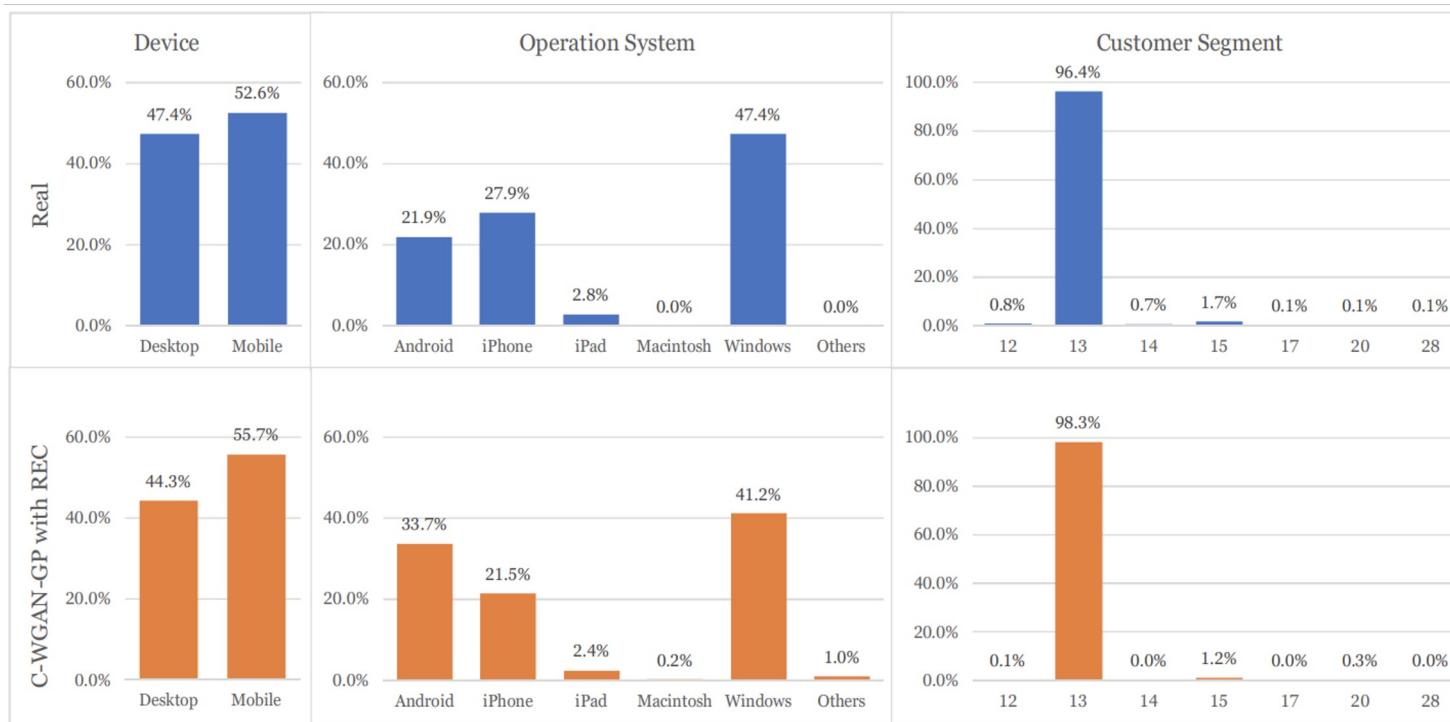
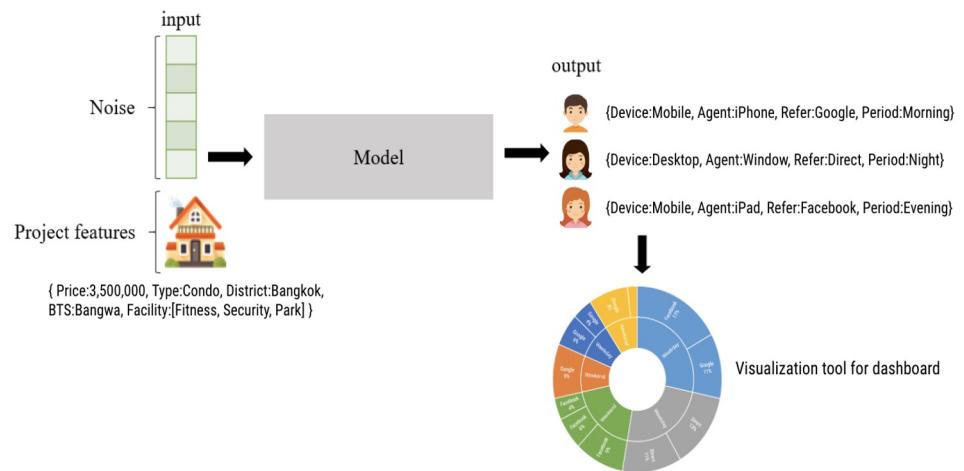
Bioinformatics <https://www.biorxiv.org/content/10.1101/2021.04.14.439903v1.full.pdf>

Face recognition <https://arxiv.org/pdf/1911.08080.pdf>

New product development



Predicting users

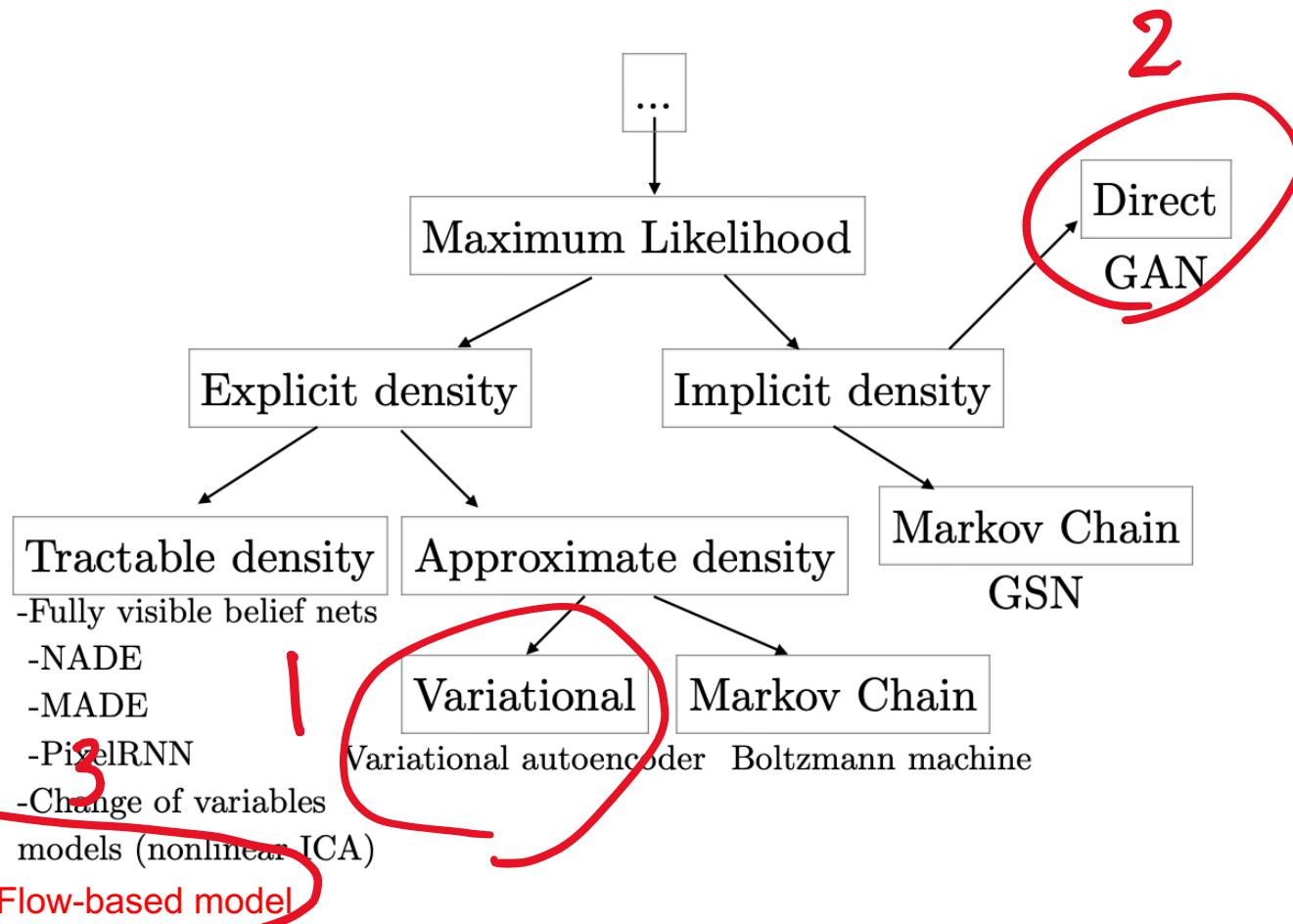


GAN vs VAE

- Typically GAN provides better quality
- Typically VAE provides more controllability
 - Representation more disentangled
 - Can easily interpolate two inputs
- VAE can encode nice embeddings/features for other task
- Research exist to fill both gaps
- GANs are harder to train than VAE

Deep Generative Models

- A deep learning model that can be used to generate X



Announcing the project tournament

Grand finals

17/5/24 Afternoon (Online presentation)
6 finalists
Mandatory attendance (participation score for asking questions)
Winning team receives a special prize!



บໍານ

ບໍານ

| |
|--|
| Assigned to houses |
| Proposal feedback with coaches (discord) - til 4/4/24 |
| Project update (online) - 18/4/24 |
| Project presentation (video) – deadline 13/5/24 |
| Q/A period with coaches (discord) - during 14/5/24 |
| Finalist announcement 15/5/24 |

(participation score for asking questions)

Project update (18/4/24)

- Goal: at least have the first baseline
- Content
 - Setup of the problem (dataset, etc.)
 - Methods tried
 - Results
 - Plan
- Presentation 10 mins + 5 mins question

Project update (18/4/24)

- Goal: at least have the first baseline or at least a stupid baseline
- Content
 - Setup of the problem (dataset, etc.)
 - Metrics
 - Methods tried
 - Results
 - Plan
- Presentation 10 mins + 5 mins question
- Must have enough detail for your classmates to understand
 - If you are working based on some technique never taught in class you must explain the technique

Project presentation (13/5/24)

- Goal: mostly finish*
- Content
 - Setup of the problem (dataset, etc.)
 - Metrics
 - Methods tried
 - Results
 - Error analysis (which method is better, what can it do well/not well, if possible try to say why)
- Presentation video 15 mins