BOSTON UNIVERSITY

Deep Learning for Data Science
DS 542

Lecture 21
Unsupervised Learning
And
Variational Autoencoders

Slides originally by Thomas Gardos.

Images from Understanding Deep Learning unless otherwise cited.

Last Time

- Adversarial Inputs
- Generative Adversarial Networks ← example of unsupervised learning

This Time

Unsupervised Learning

- Taxonomy
- Generative models
- Quantifying performance

Two new kinds of generative models

- Normalizing flows → next time
- Variational autoencoders

Supervised Learning

Any time that

- We are provided input/output pairs
- And asked to build a model generalizing them

Unsupervised learning

- Everything else? Not quite.
- Self/semi-supervised learning used inconsistently.
    - Sometimes partially supervised.
    - Sometimes deriving targets for unsupervised data.
- Reinforcement learning is pretty different. Will come back to that later.

Unsupervised Learning

- Learning problems where an input/output relation was not provided.
  - Often not a specific function to learn.
- General task is “learn the distribution”.
  - Calculate mean and standard deviation technically qualifies.
  - But usually we want something that can match the distribution a lot better.

Unsupervised Learning → Supervised Learning?

Previously saw next token prediction with LLMs

- Was this supervised or unsupervised?

Unsupervised Learning → Supervised Learning?

Previously saw next token prediction with LLMs

- Was this supervised or unsupervised?
    - Unsupervised data set - lots of text.
    - Extracted lots of supervised problems - pieces of text and next tokens.
    - Fine tuning GPT 4 → ChatGPT has more explicit supervision.

Generation by discriminating what to generate next 🤔

Supervised vs. Self/Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is a label

Goal: Learn function to map

$$x \rightarrow y$$

Applications: Classification, regression, object detection, semantic segmentation, etc.

Self/Unsupervised Learning

Data: x

x is data, no labels! Or labels part of the data

Goal: Learn the hidden or underlying structure of the data.

Applications: Clustering, dimensionality reduction, compression, find outliers, generating new examples, denoising, interpolating between data points, etc.

Related split: did humans decide the labels or targets?

Latent Variables

- What is a latent variable?
  - Invisible but underlying truth behind what's going on?
- Latent variable → observations?
  - Often lower dimension than our observations.
  - Observation ~ f(latent)
  - But not always
- Observation → latent variable?
  - K-means mapping data to cluster id
  - Often will want to infer latents from observations (like inverting GAN)

Will be saying observation a lot today to distinguish "visible" data from inferred latents.

Generative Models

If you have

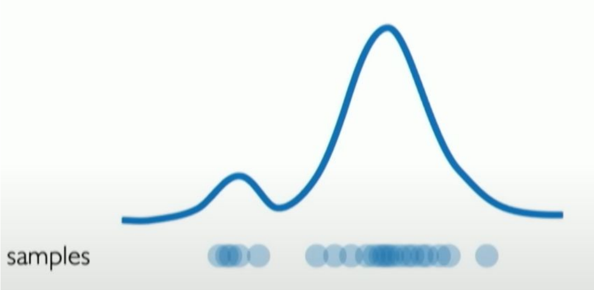
1. Probability distribution of latent variables
2. Function mapping latent variables to observations

You basically have a generative model.

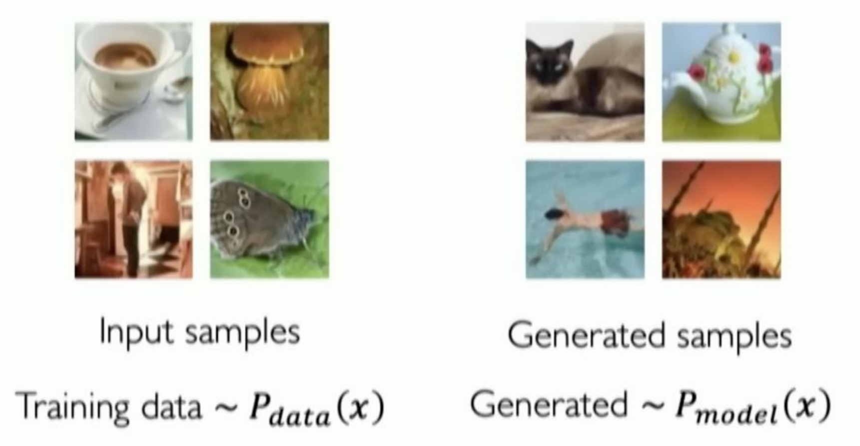
Generative Modeling

Goal: Take as input training samples from some distribution and learn a model that represents that distribution

Probability Density Estimation



Sample Generations



How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?

Why generative models? Debiasing

Capable of uncovering **underlying features** in a dataset

Homogeneous skin color, pose

VS

Diverse skin color, pose, illumination

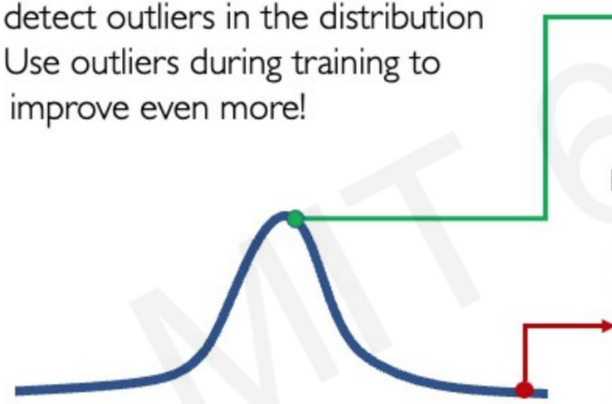
How can we use this information to create fair and representative datasets?

Amini et al, "Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure," 2019

Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:
(1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training

Edge Cases

Harsh Weather

Pedestrians

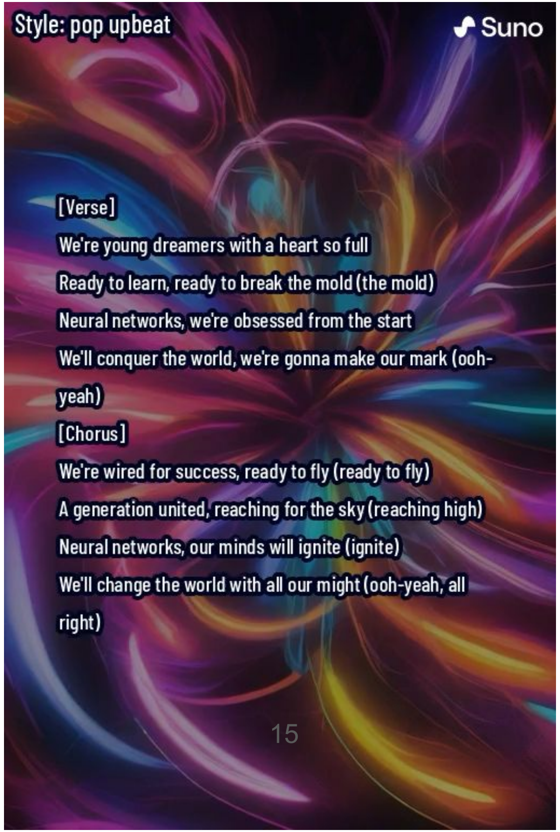
A. Amini et al, “Variational Autoencoder for End-to-End Control of Autonomous Driving with Novelty Detection and Training De-biasing,” *2018*

More outlier examples

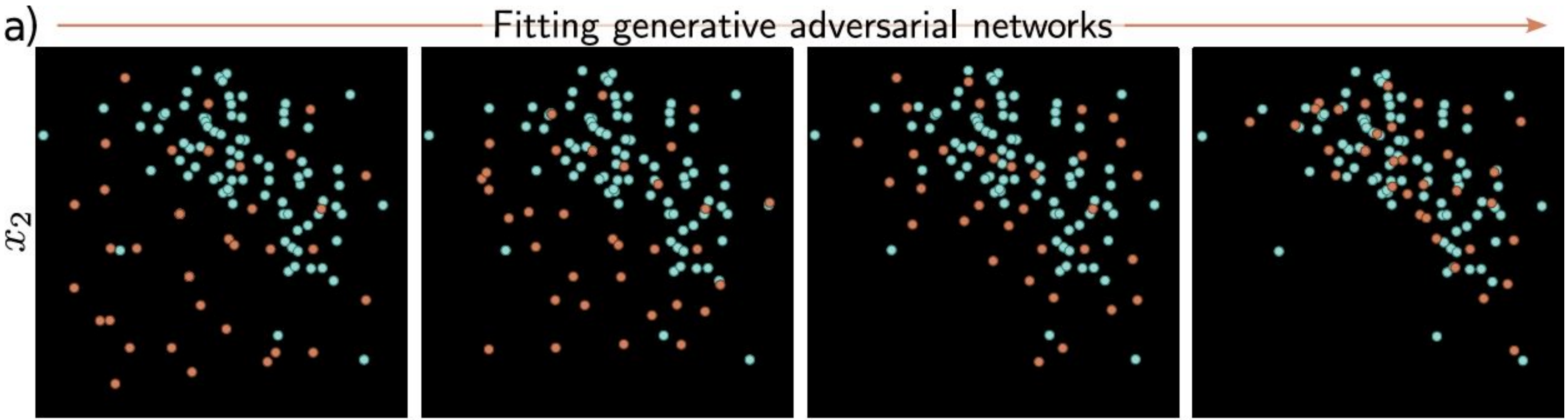
YouTube Video, Feb. 2020 -- https://www.youtube.com/watch?v=hx7BXih7zx8&t=514s

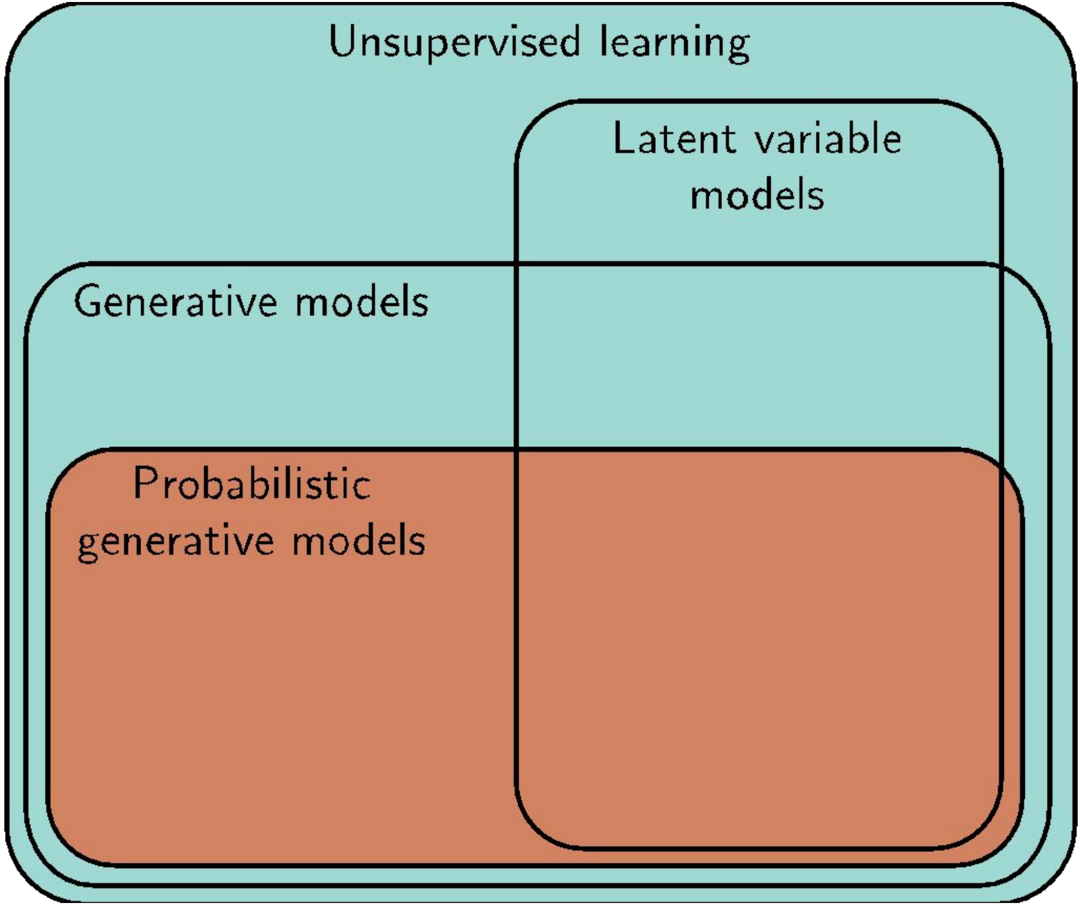
Why generative models?
image, video and audio creation

A teenage superhero fighting crime in an urban setting shown in the style of claymation.



Write a short pop song about students wanting to learn about neural networks and do great things with them.


a)
Fitting generative adversarial networks
x_2



Generative = can generate new examples

Probabilistic = can assign probability to data examples

Probabilistic Generative Models

Key distinction

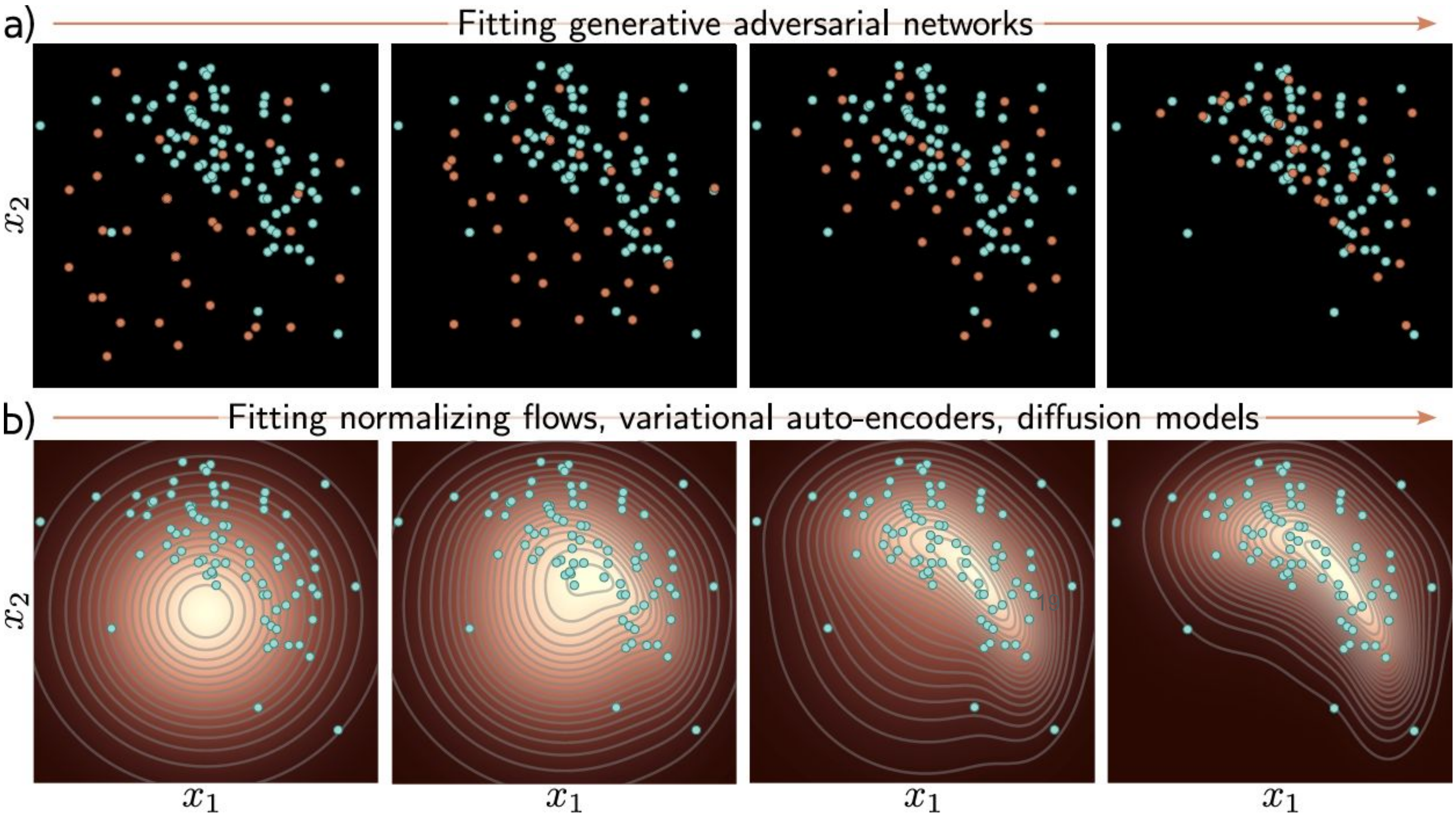
- Can assign probability to observations (conditioned on model parameters)

Can't you get this from the latent probabilities and latent to observation mapping?

Not always easy to invert...

Standard optimization:

- Maximize probability of observations
- Requires direct calculation of observation probability from model parameters?
- Implicitly suppresses dissimilar possibilities…


a)
Fitting generative adversarial networks
b)
Fitting normalizing flows, variational auto-encoders, diffusion models
x_2
x_2
x_1
x_1
x_1
x_1

Probabilistic Generative Models

Since we can calculate probabilities for observations,

- We can compare different models
  - Which model makes the test data more likely?
- We can quantify how unlikely an observation is…
  - So is it an outlier?

Examples of Probabilistic Generative Models

- Normalizing flows (next week)
- Variational autoencoders
- Diffusion models (next week)

Probabilistic models

- Maximize log likelihood of training data

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log[\Pr(x_i | \phi) \right]$$

- Find the parameters, ϕ, of some parametric probability distribution so that the training data is most likely under that distribution

What makes a good model?

- Efficient sampling:
  - Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.

What makes a good model?

- High-quality sampling:
  - The samples should be indistinguishable from the real data that the model was trained with.
  - This is broadly getting better as we train bigger models.

What makes a good model?

- Coverage:
  - Samples should represent the entire training distribution. It is insufficient to only generate samples that all look like a subset of the training data.
  - GANs have trouble with this since their generator training does not directly see the training data…

What makes a good model?

- Well-behaved latent space:
  - Every latent variable z should correspond to a plausible data example x and smooth changes in z should correspond to smooth changes in x.
  - Usually this is the case. Just ignore the 6 fingered hands?

What makes a good model?

- Interpretable latent space:
  - Manipulating each dimension of z should correspond to changing an interpretable property of the data. For example, in a model of language, it might change the topic, tense or degree of verbosity.

This is stronger than having a well-behaved latent space, since changes in a particular direction need to be semantically similar.

What makes a good model?

- Efficient likelihood computation:
    - If the model is probabilistic, we would like to be able to calculate the probability of new examples efficiently and accurately.

WTB: a probability calculator that identifies fake news as low probability.

Do we have good models?

	GANs	VAEs	Flows	Diffusion
Efficient sampling	✓	✓	✓	✗
High quality	✓	✗	✗	✓
Coverage	✗	?	?	?
Well-behaved latent space	✓	✓	✓	✗
Interpretable latent space	?	?	?	✗
Efficient likelihood	n/a	✗	✓	✗

How to measure performance within or between categories?

- Open research area.

Quantifying Performance - Test Likelihood

How likely is the the test data given our model? (Throwback to loss functions)

$$\sum_{i=1}^{I} \log[\Pr(x_i | \phi)$$

See also perplexity if working with text.

Quantifying Performance - Inception Score

Grading via another model

- Usually the Inception model for ImageNet
- Want generated images to have a single very likely classification.
- But average flat classification across generated images.
- Formal formula checking KL-divergence between those on a per-generated image basis…

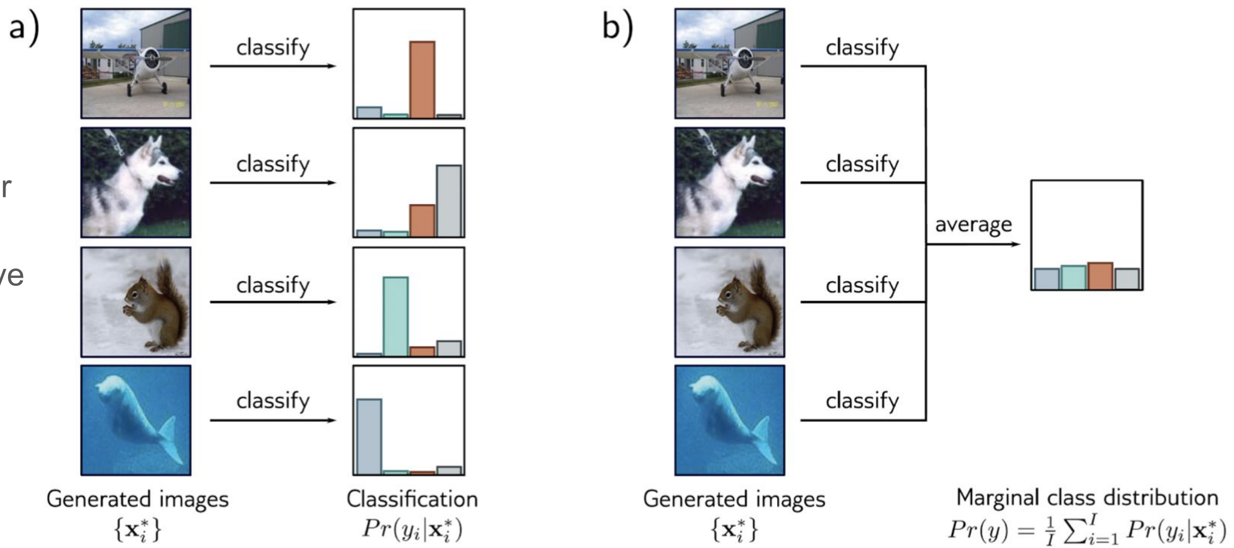


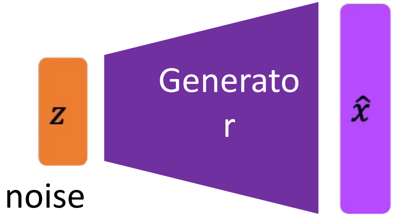
Figure 14.4 Inception score. a) A pretrained network classifies the generated images. If the images are realistic, the resulting class probabilities $Pr(y_i|\mathbf{x}_i^*)$ should be peaked at the correct class. b) If the model generates all classes equally frequently, the marginal (average) class probabilities should be flat. The inception score measures the average distance between the distributions in (a) and the distribution in (b). Images from Deng et al. (2009).

Quantifying Performance - Fréchet Inception Distance

Another visual similarity metric based on Inception model (others can be used).

- Map generated images to distribution of Inception features.
- Model the distribution of Inception features as a multivariate normal distribution.
- Compare two such distributions with the Wasserstein distance (metric)
    - Also called “earth mover’s distance”
    - Smaller is better.
    - Closed form solution from multivariate normal assumption.

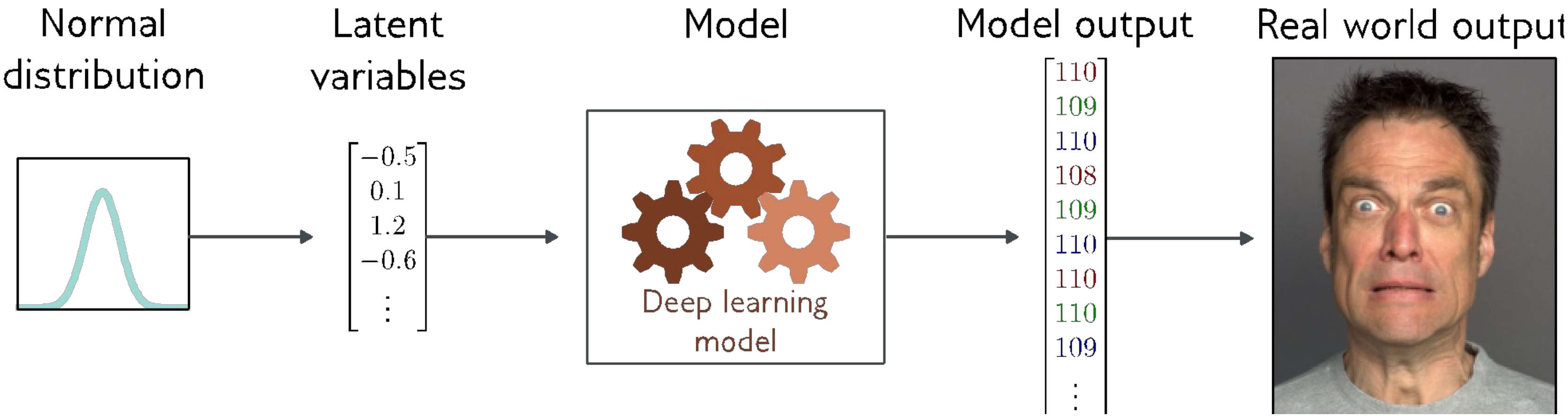
General Idea of GANs



- Don’t try to build a probability model directly
- Learn a transformation from a sample of noise to look similar to training data distribution

Left GANs vulnerable to mode collapse where only some of the distribution is replicated.

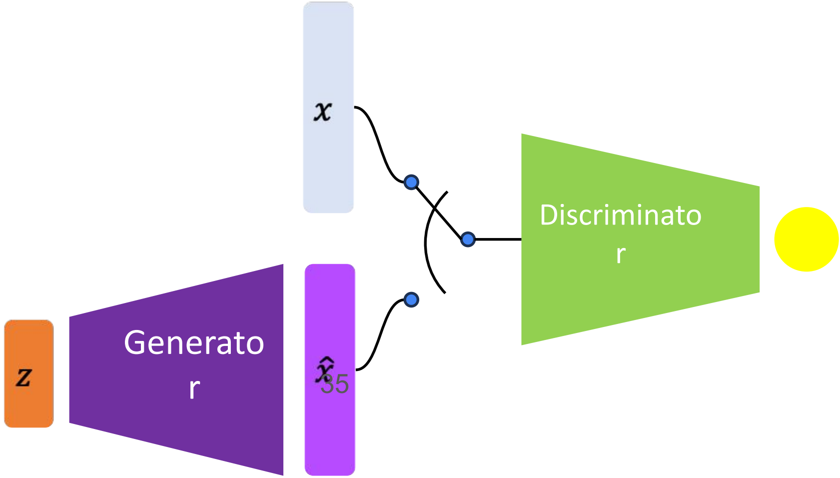
Latent variable models



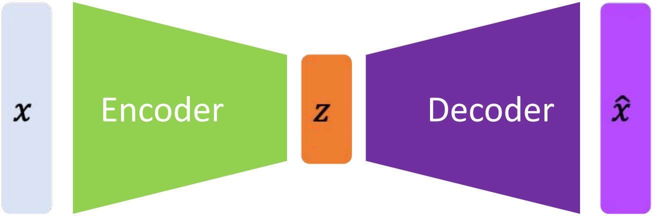
Latent variable models map a random “latent” variable to create a new data sample

Latent Variable Models

Generative Adversarial Networks



Autoencoders and Variational Autoencoders (VAEs)



Latent Variable Models

Informally speaking, different levels of latent variables…

- Latent variable directly determines observations
  - e.g. x = f(z)
- Latent variable determines distribution of observations
  - e.g. x ~ Norm[f_mu(z), f_sigma2(z)]
- These levels aren't really different -
  - An extremely tight distribution ~ a fixed prediction
  - A fixed prediction + noise ~ a distribution

Variational Autoencoders (VAEs)

Goal is to learn the probability distribution from observed data

Can sample the distribution, but not evaluate probabilities exactly.

Variational Autoencoder

Variational Inference: A method from machine learning that approximates probability densities through optimization.

Autoencoder: A type of artificial neural network used to learn efficient codings of unlabeled data in an unsupervised manner.

VAE is an autoencoder whose encodings distribution is regularized during the training to ensure that its latent space has good properties allowing us to generate new data.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Autoencoder: A type of artificial neural network used to learn efficient codings of unlabeled data in an unsupervised manner.

Variational Inference: A method from machine learning that approximates probability densities through optimization.

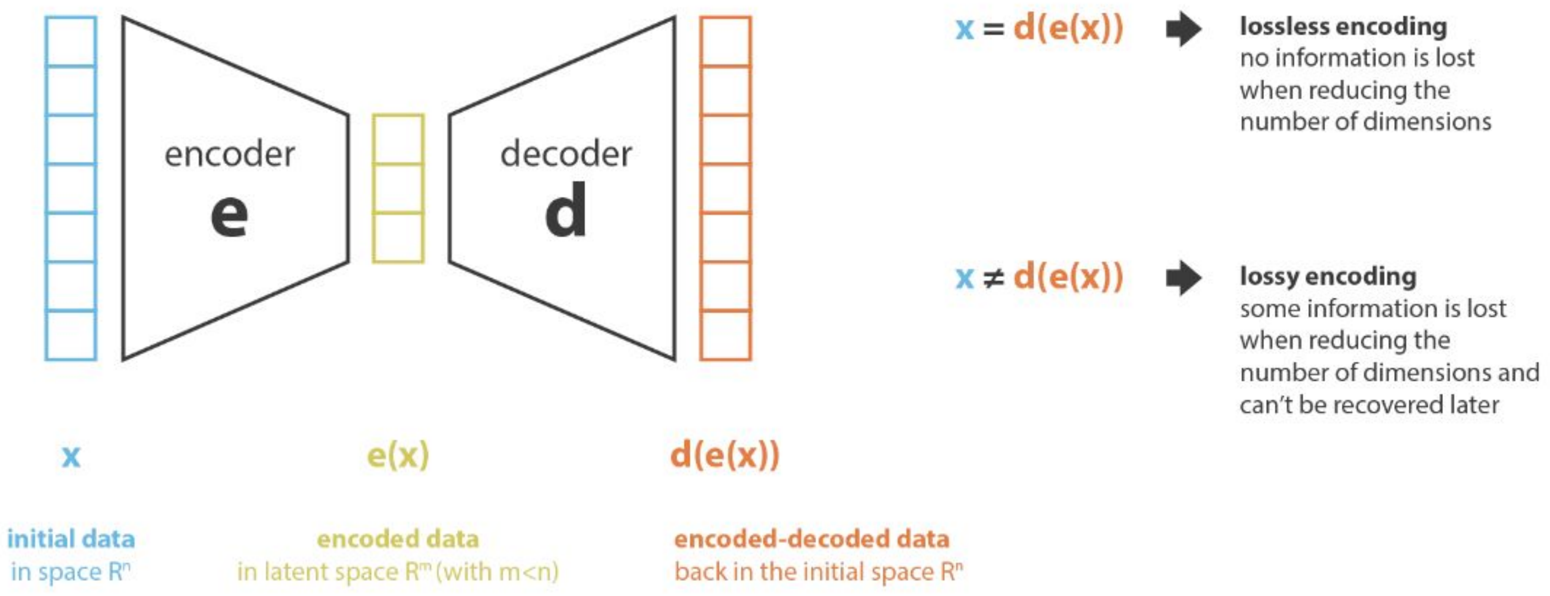
Bayesian since joint density is decomposed into prior and posterior density distributions using Bayes Rule:

$$p(\mathbf{z},\mathbf{x}) = p(\mathbf{x}|\mathbf{z})\,p(\mathbf{z})$$

Outline

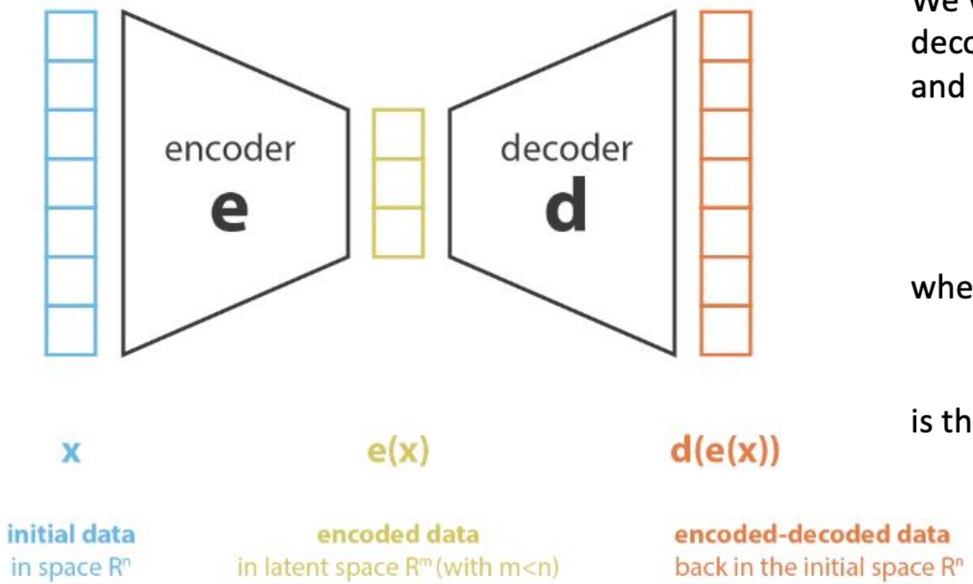
- Autoencoder and its limitations
- Intuition behind VAEs
- Derivation of VAE
- Example applications

Dimensionality reduction with an autoencoder



Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Dimensionality reduction with an autoencoder



We want to find the best encoder, **e**, and decoder, **d**, to minimize the error between x and d(e(x)).

$$(e^*, d^*) = \underset{(e,d) \in E \times D}{\operatorname{argmin}} \ \epsilon(x, d(e(x)))$$

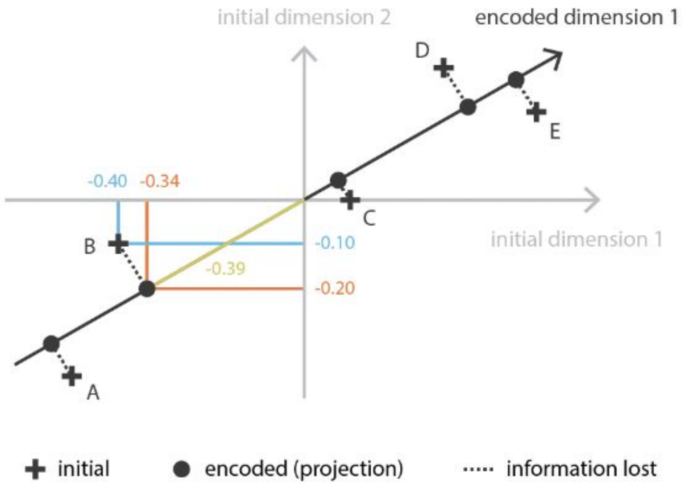
where

$$\epsilon(x, d(e(x)))$$

is the reconstruction error.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Dimensionality reduction with Principal Component Analysis (PCA)



$n_d = 2 \;\; n_e = 1$

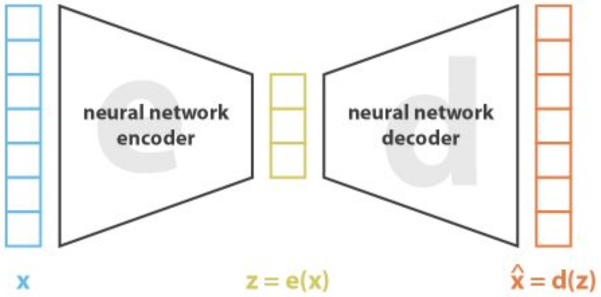
Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

Project the n_d-dimensional features onto an orthogonal n_e-dimensional subspace that minimizes Euclidean distance.

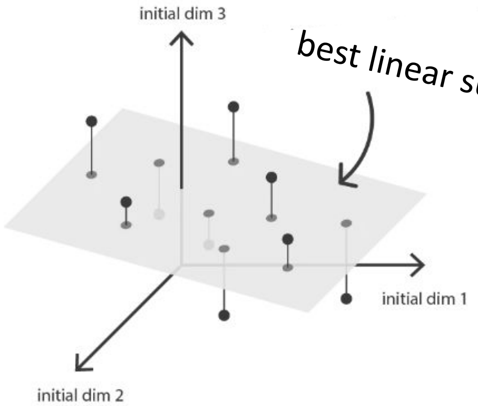
Linear Transformation!!

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

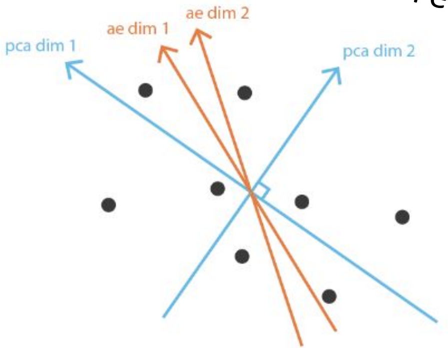
Neural Network Autoencoder – 1 Linear Layer



We could define encoder and decoder to each have one linear layer (no activation function), but it wouldn't necessarily converge during training to PCA solution.



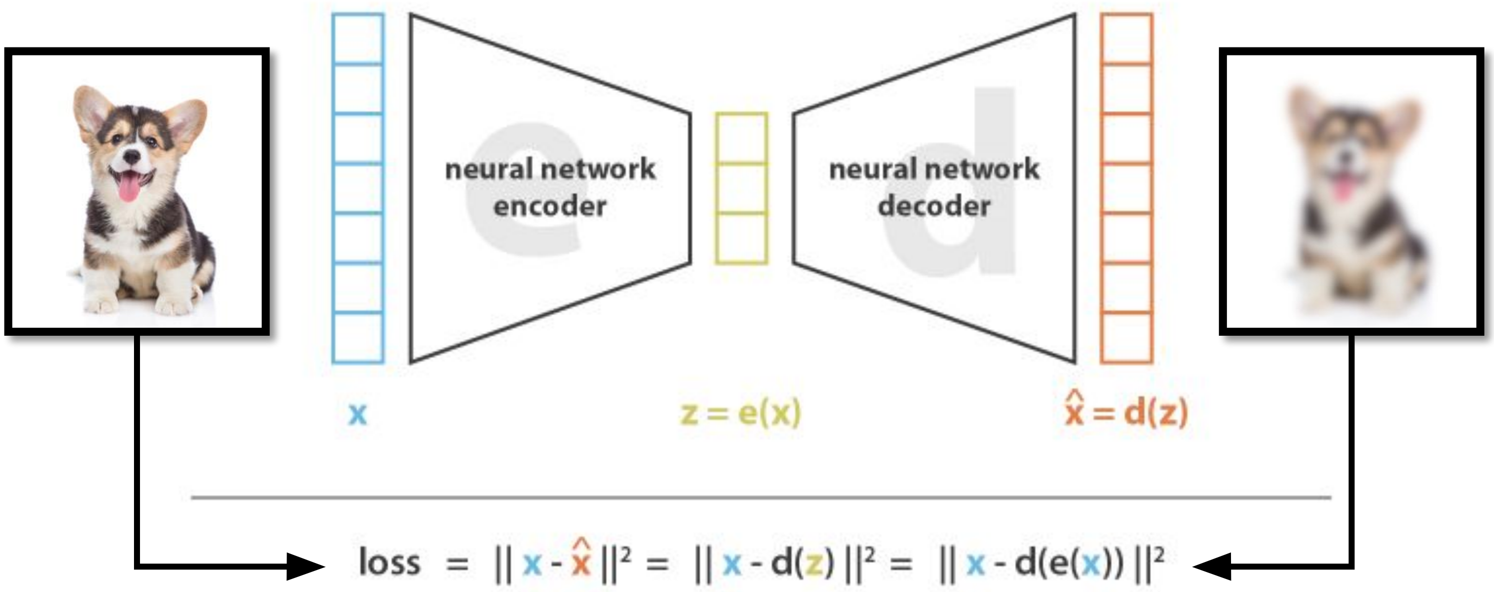
best linear subspace



AE can end up with any basis

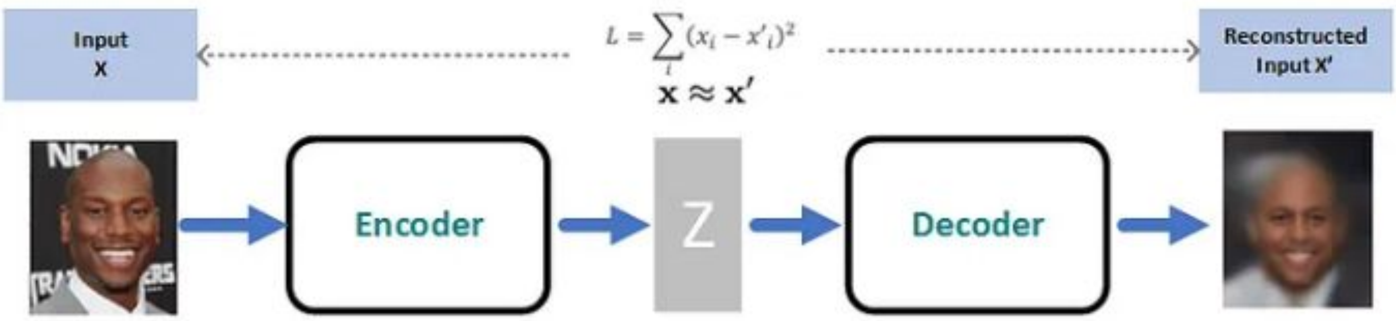
Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Neural Network Autoencoder



Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

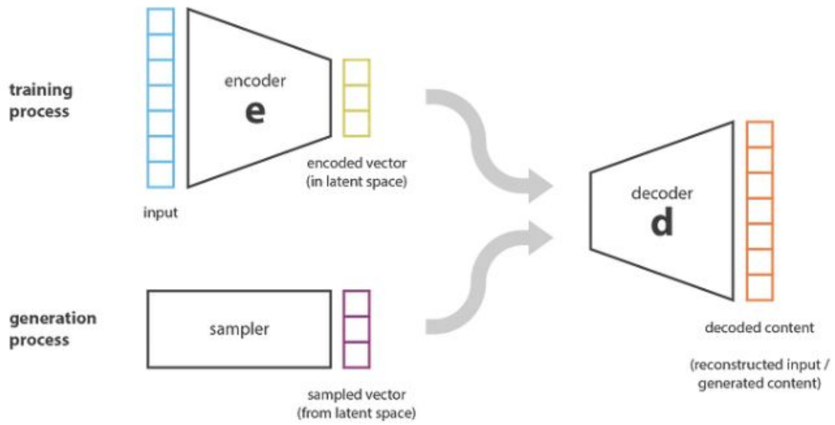
Autoencoder Reconstruction



Trained on CelebA dataset.

Kana, "Variational Autoencoders (VAEs) for Dummies -- Step by Step Tutorial", 2020

Can we generate new samples with autoencoder?



Train encoder and decoder as autoencoder.

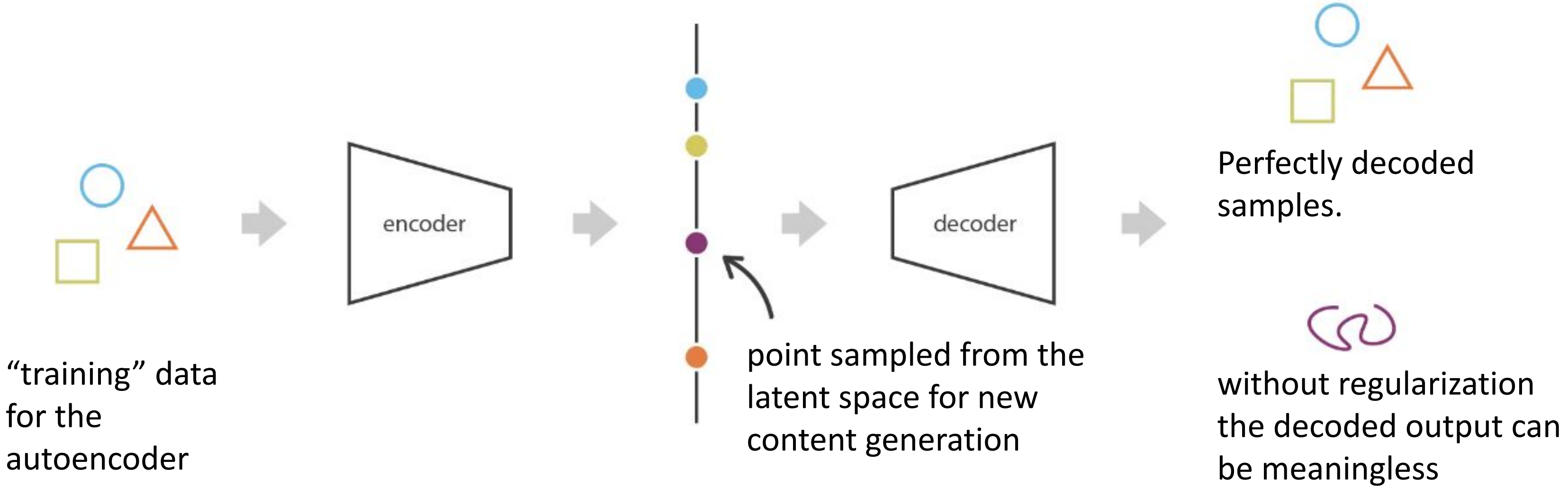
Randomly select a different point in the latent space.

Provide as input to the decoder to generate an output.

Will this produce a good quality output? Why?

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Extreme case: Memorization



Encoder and decoder are so powerful that they can fully memorize the data.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Outline

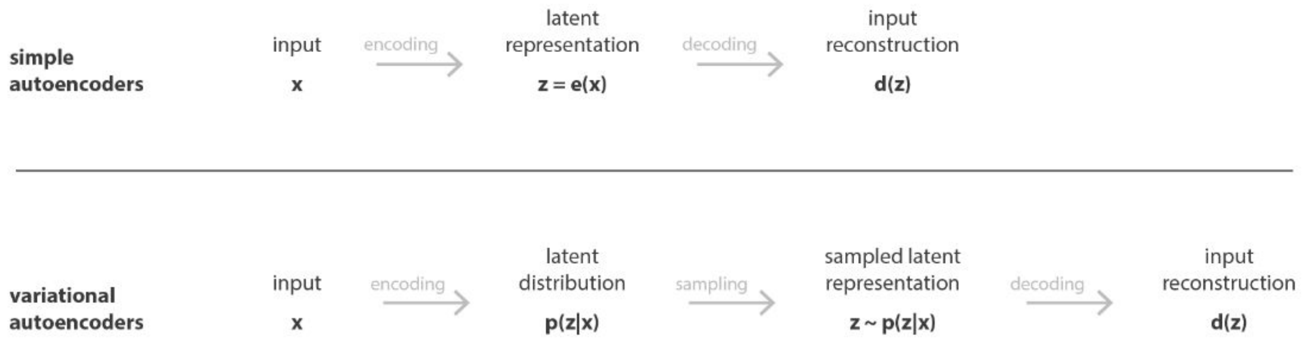
- Autoencoder and its limitations
- Intuition behind VAEs
- Derivation of VAE
- Example applications

Variational Autoencoder…

…is an autoencoder whose training is *regularized* to avoid overfitting and ensure that the *latent space has good properties* that enable generative process.

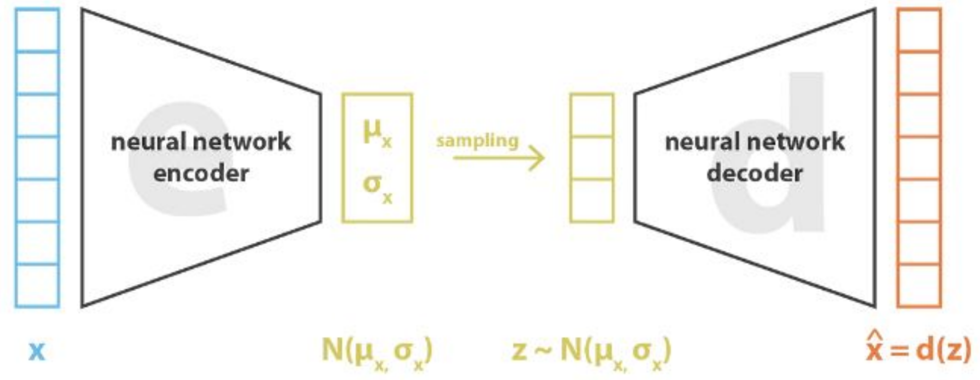
Instead of encoding as a *single point*, encode it as a *distribution* over the latent space.

Assume distributions are normal.



Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

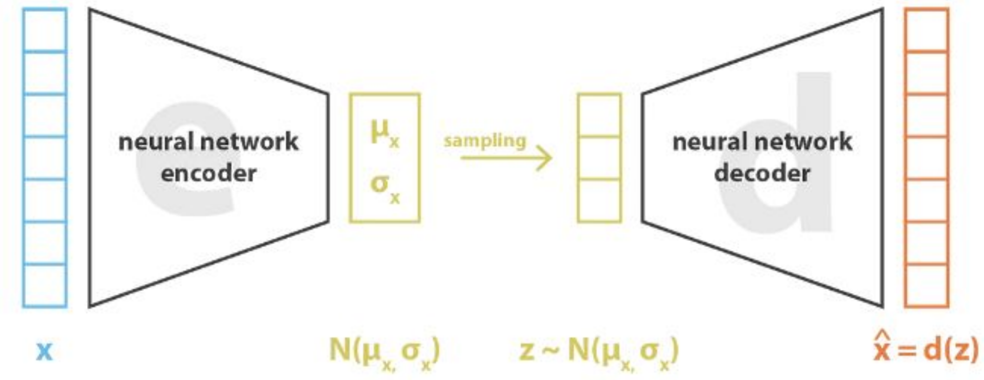
Variational Autoencoder



$$\text{loss} = \| x - \hat{x} \|^2 + \text{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \text{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

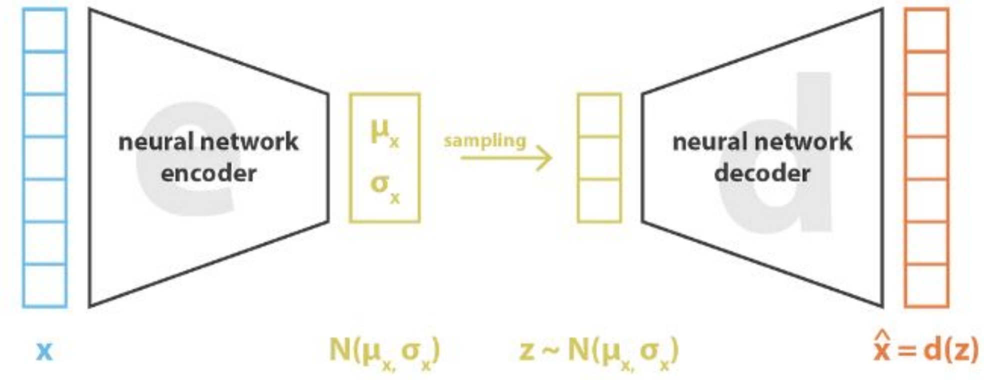
Variational Autoencoder



Encoder is emitting μ_x vector and σ_x diagonal vector for independent gaussians densities.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

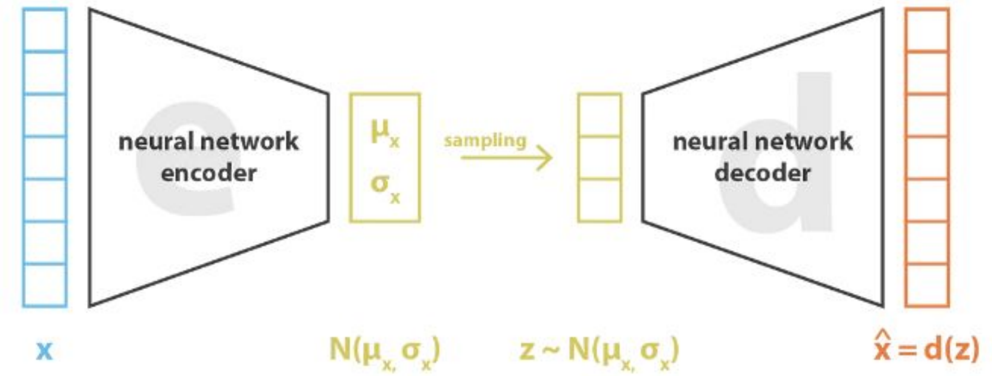
Variational Autoencoder



We then sample z from the multivariate Normal.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

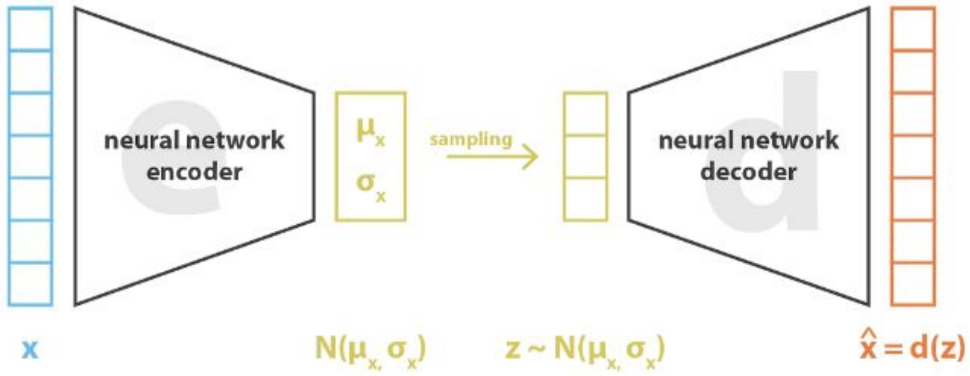
Variational Autoencoder



Then input z to the decoder
network to produce output.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Variational Autoencoder

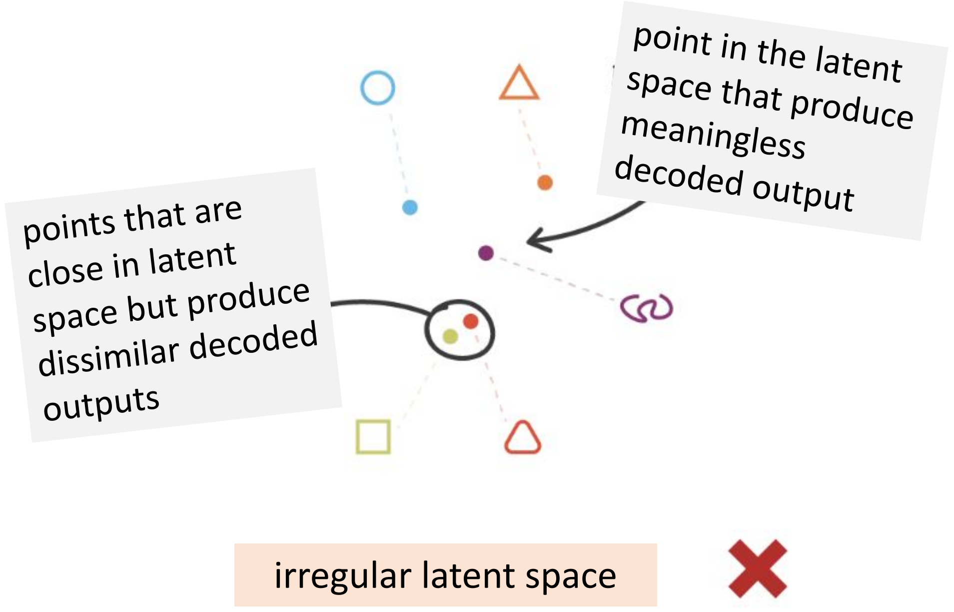


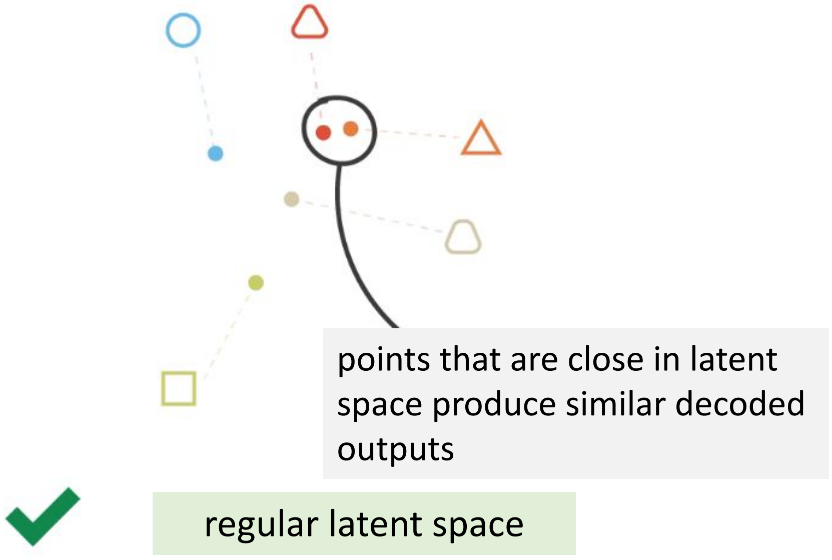
$$\text{loss} = \underbrace{\| x - \hat{x} \|^2}_{\text{L2 Loss}} + \underbrace{\mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]}_{\text{Kulback-Leibler divergence}} = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

The loss is now the L2 loss as with the autoencoder, but with an additional KL-divergence term as regularizer.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

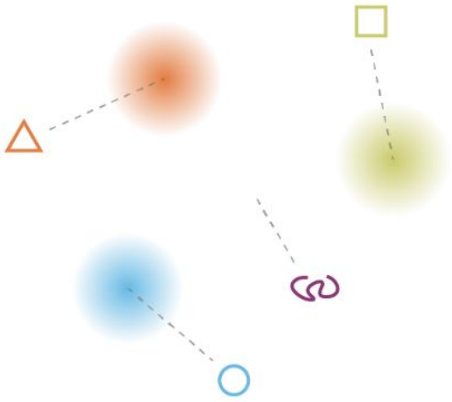
Intuitions about Regularization



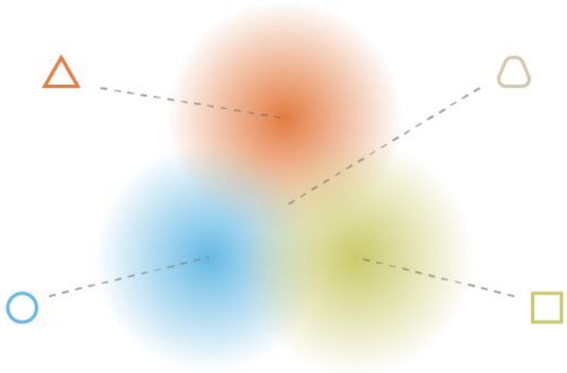


Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Encoding to Normal distributions is not enough



without regularization



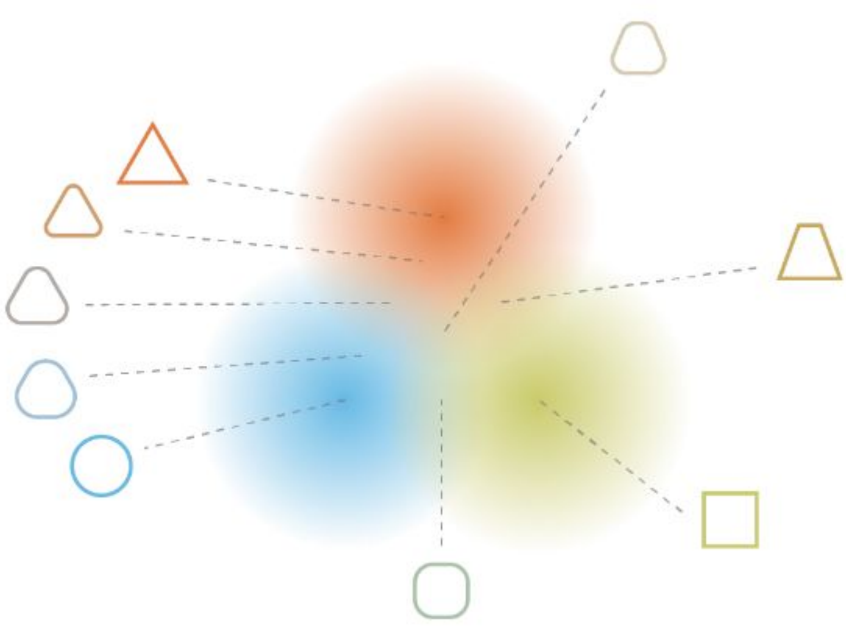
with regularization

We have to regularize the means and the covariances too!
Regularize to a standard normal.

$$\text{loss} = \|x - \hat{x}\|^2 + \mathrm{KL}[\,N(\mu_x, \sigma_x), N(0, I)\,]$$

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Benefit of regularization



The continuity and completeness obtained from regularization tends to create a “gradient” over the information encoded in latent space.

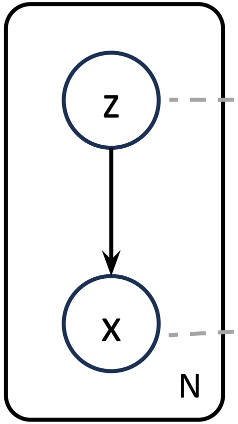
Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Dall-E 3

Outline

- Autoencoder and its limitations
- Intuition behind VAEs
- Derivation of VAE
- Example applications

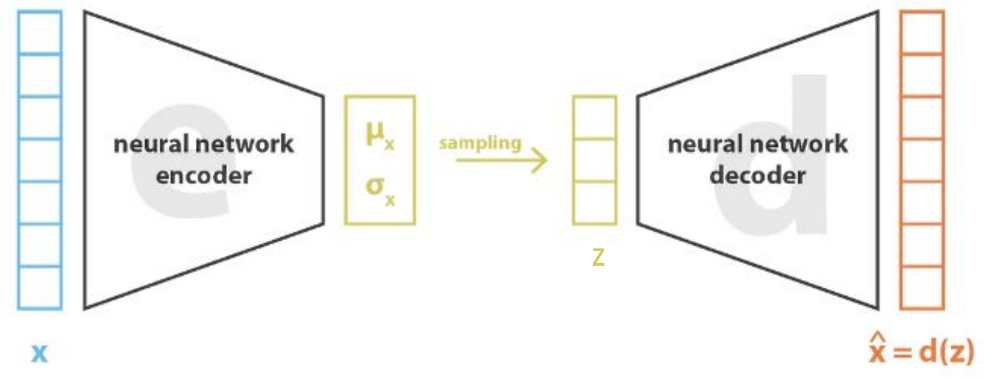
Preliminaries: Bayesian Models



$p(z)$ *prior* – prior knowledge or belief about z

$p(x|z)$ *likelihood* – probability of a sample given z

Bayesian Inference

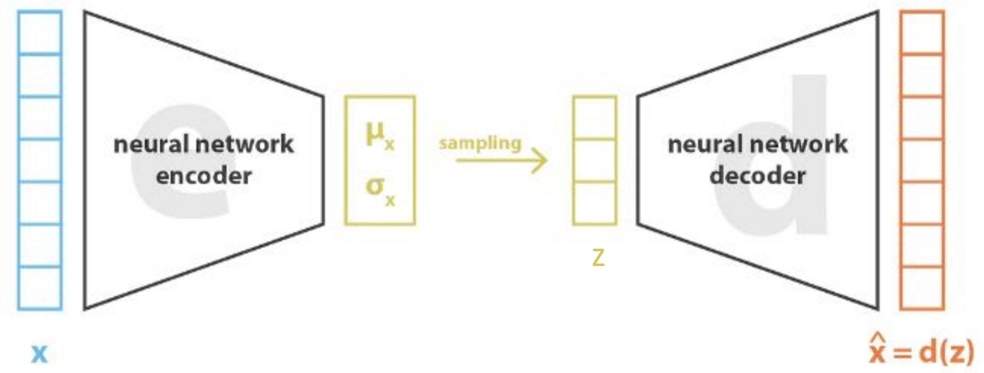


Probabilistic Encoder
$p(z|x)$

Probabilistic Decoder
$p(x|z)$

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Bayesian Inference



Probabilistic Encoder

$p(z|x)$

posterior – update our knowledge of z given a new sample

Probabilistic Decoder

$p(x|z)$

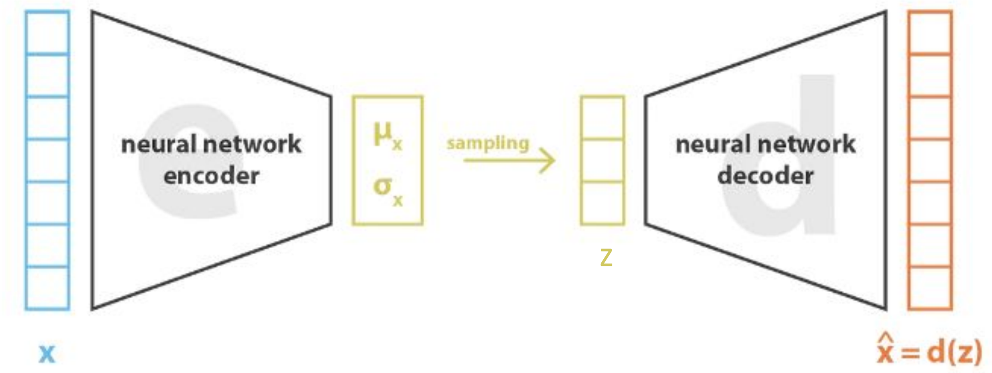
likelihood – probability of a sample given z

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

We can relate the *posterior* to the *likelihood* via **Bayes Theorem.**

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Bayesian Inference



Probabilistic Encoder

$$\underbrace{p(z|x)}_{\textit{posterior}}$$

posterior – update our knowledge of z given a new sample

Probabilistic Decoder

$$\underbrace{p(x|z)}_{\textit{likelihood}}$$

likelihood – probability of a sample given z

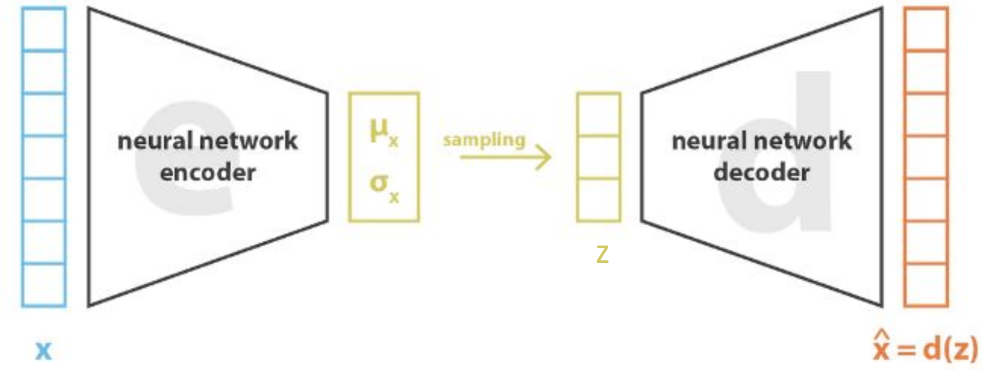
$$\underbrace{p(z|x)}_{\textit{posterior}} = \frac{\overbrace{p(x|z)}^{\textit{likelihood}}\overbrace{p(z)}^{\textit{prior}}}{\underbrace{p(x)}_{\textit{evidence}}}$$

prior – prior knowledge or belief about z

evidence – probability distribution of our observed data

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Bayesian Inference



Probabilistic Encoder

$$\underbrace{p(z|x)}$$

posterior – update our knowledge of z given a new sample

Probabilistic Decoder

$$\underbrace{p(x|z)}$$

likelihood – probability of a sample given z

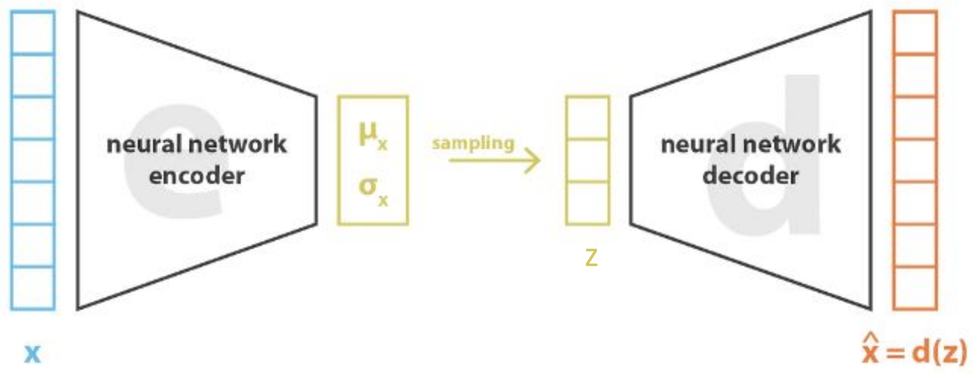
$$\underbrace{p(z|x)}_{\textit{posterior}} = \frac{\overbrace{p(x|z)}^{\textit{likelihood}}\overbrace{p(z)}^{\textit{prior}\text{ – prior knowledge or belief about z}}}{p(x)}$$

$$= \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

We can't calculate the integral directly, but we can approximate it using *variational inference*

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Simplifying Assumptions



Probabilistic Encoder
$p(z|x)$
posterior

Probabilistic Decoder
$p(x|z)$
likelihood

Assume that the *prior* is a standard Gaussian

$$p(z) \equiv \mathcal{N}(0, I)$$

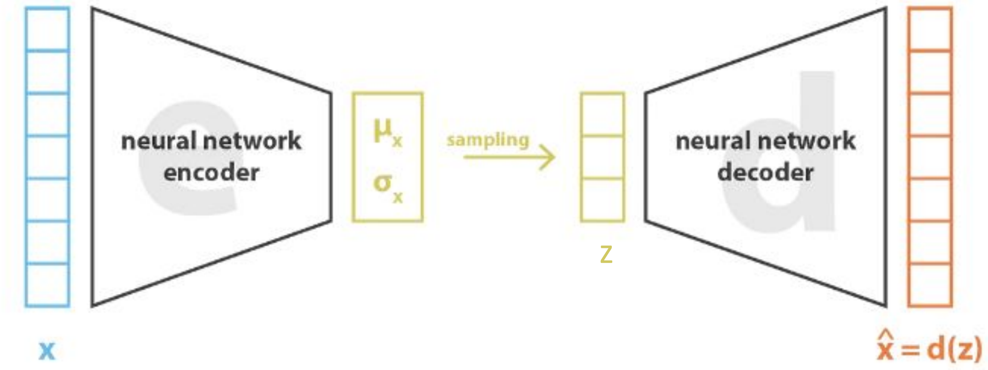
And *likelihood* is a Gaussian

$$p(x|z) \equiv \mathcal{N}(f(z), cI)$$

where $f \in F$ is a family of functions we will specify later and $c > 0$.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

Variational Inference Formulation



Probabilistic Encoder

$$\underbrace{p(z|x)}_{\textit{posterior}}$$

Probabilistic Decoder

$$\underbrace{p(x|z)}_{\textit{likelihood}}$$

We are going to approximate *posterior* to parameterized set of Gaussians.

Approximate $p(z|x)$ by a Gaussian $q_x(z)$.

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

where $g \in G$ and $h \in H$ are a family of functions we will define shortly.

Rocca, "Understanding Variational Autoencoders (VAEs)", 2019

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$(g^*, h^*) = \underset{(g,h)\in G\times H}{\arg\min}\ KL(q_x(z), p(z|x))$$

We want to find the best functions, g and h, to minimize the KL-divergence from the posterior $p(z|x)$.

C.5.1 Kullback-Leibler divergence

The most common measure of distance between probability distributions $p(x)$ and $q(x)$ is the *Kullback-Leibler* or KL divergence and is defined as:

$$D_{KL}\left[p(x)||q(x)\right] = \int p(x)\log\left[\frac{p(x)}{q(x)}\right]dx. \tag{C.28}$$

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$\begin{aligned}(g^*, h^*) &= \underset{(g,h)\in G\times H}{\arg\min}\ KL(q_x(z), p(z|x)) \\ &= \underset{(g,h)\in G\times H}{\arg\min} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right)\end{aligned}$$

- Rewriting KL divergence as Expectation,
- log of division is difference of the logs
- substituting for the posterior using Bayes Theorem

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$(g^*, h^*) = \underset{(g,h)\in G\times H}{\arg\min}\ KL(q_x(z), p(z|x))$$

$$= \underset{(g,h)\in G\times H}{\arg\min} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right)$$

$$= \underset{(g,h)\in G\times H}{\arg\min} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x))\right)$$

- log of product becomes sum of logs
- log of division becomes difference of logs

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$\begin{aligned}
(g^*, h^*) &= \underset{(g,h)\in G\times H}{\arg\min}\; KL(q_x(z), p(z|x)) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\; \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\; (\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x))) \\
&= \underset{(g,h)\in G\times H}{\arg\max}\; (\mathbb{E}_{z\sim q_x}(\log p(x|z)) - KL(q_x(z), p(z)))
\end{aligned}$$

- negating and converting from argmin to argmax
- collecting terms to form KL divergence

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$
\begin{aligned}
(g^*, h^*) &= \underset{(g,h)\in G\times H}{\arg\min}\ KL(q_x(z), p(z|x)) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\ \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\ (\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x))) \\
&= \underset{(g,h)\in G\times H}{\arg\max}\ (\underbrace{\mathbb{E}_{z\sim q_x}(\log p(x|z))} - \underbrace{KL(q_x(z), p(z))})
\end{aligned}
$$

Maximize the expected log likelihood.

Minimize the difference between the approximate posterior and the prior.

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

Variational Inference

$$
\begin{aligned}
(g^*, h^*) &= \underset{(g,h)\in G\times H}{\arg\min}\; KL(q_x(z), p(z|x)) \\
&= \underset{(g,h)\in G\times H}{\arg\min} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right) \\
&= \underset{(g,h)\in G\times H}{\arg\min} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x))\right) \\
&= \underset{(g,h)\in G\times H}{\arg\max} \left(\mathbb{E}_{z\sim q_x}(\log p(x|z)) - KL(q_x(z), p(z))\right) \\
&= \underset{(g,h)\in G\times H}{\arg\max} \left(\underbrace{\mathbb{E}_{z\sim q_x}\left(-\frac{||x - f(z)||^2}{2c}\right)} - KL(q_x(z), p(z))\right)
\end{aligned}
$$

Log of the Gaussian likelihood $p(x|z) \equiv \mathcal{N}(f(z), cI)$.

This brings our function, f, into the equation, so...

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

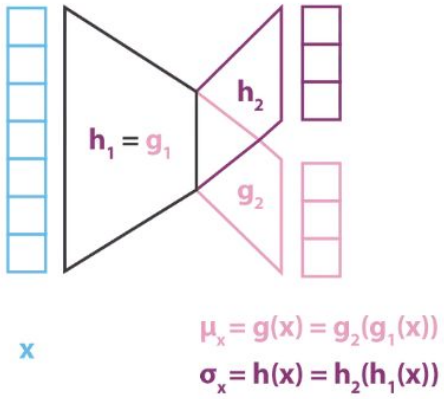
Variational Inference

We are looking for optimal f^*, g^* and h^* such that

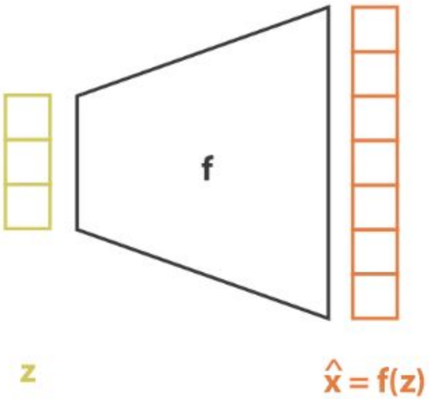
$$(f^*, g^*, h^*) = \underset{(f,g,h)\in F\times G\times H}{\arg\max} \left(\mathbb{E}_{z\sim q_x} \left(-\frac{||x - f(z)||^2}{2c} \right) - KL(q_x(z), p(z)) \right)$$

Note that the constant, c, determines the balance between reconstruction error and the regularization term given by KL divergence.

Enter the Neural Networks



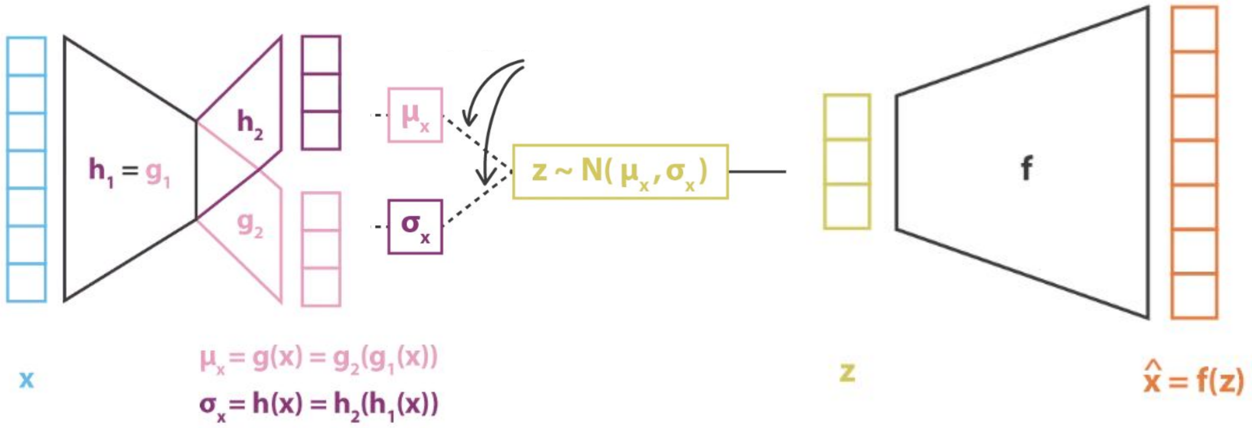
Encoder produces the mean and variance.



Decoder reconstructs the input (during training)

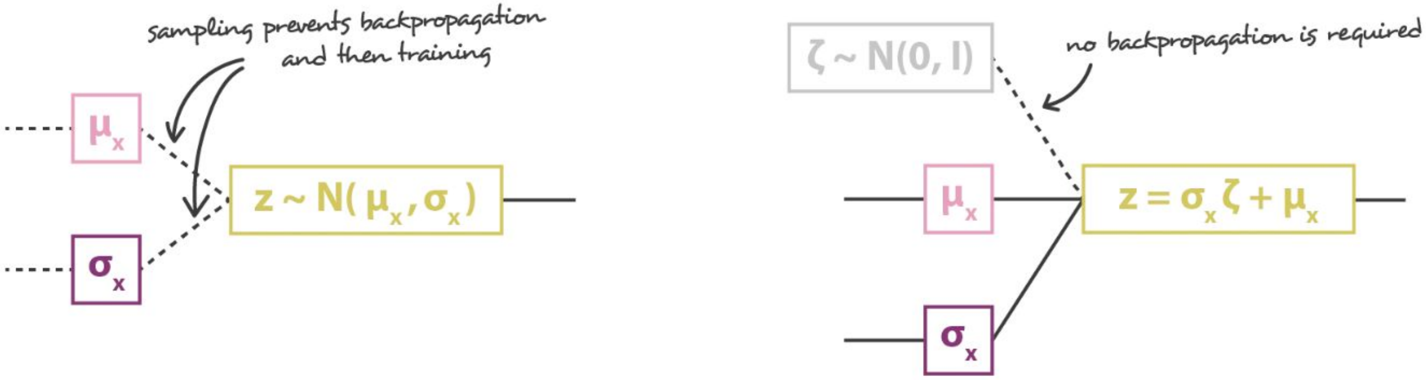
But one more problem to solve

STOP

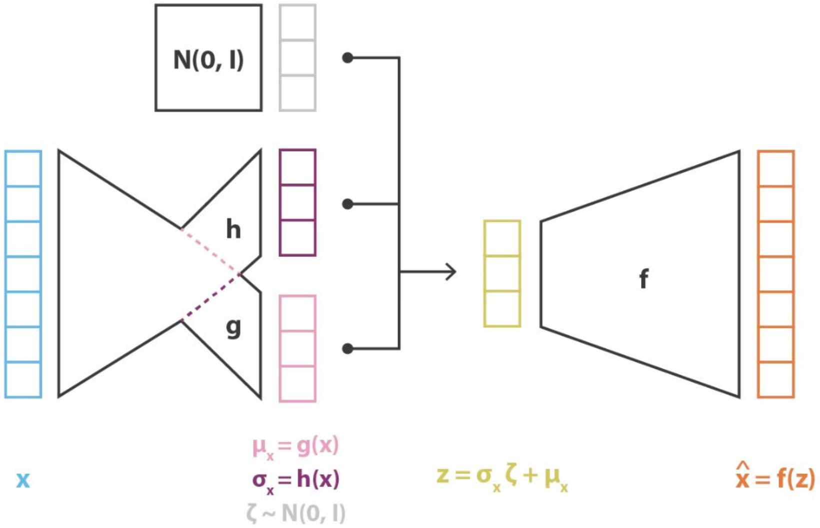


We can’t backpropagate through the sampling step.

Use the reparameterization trick



Putting it all together



$$\text{loss} = C \, || \, x - \hat{x} \, ||^2 + \text{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,] = C \, || \, x - f(z) \, ||^2 + \text{KL}[\, N(g(x), h(x)), N(0, I) \,]$$

We use a Monte-Carlo approximation to the expectation of reconstruction loss

Convert C = 1/(2c).

We have as trainable neural network!

Probability Distribution Divergence Measures

C.5.1 Kullback-Leibler divergence

The most common measure of distance between probability distributions $p(x)$ and $q(x)$ is the *Kullback-Leibler* or KL divergence and is defined as:

$$D_{KL}\big[p(x)||q(x)\big] = \int p(x) \log\left[\frac{p(x)}{q(x)}\right] dx. \tag{C.28}$$

C.5.2 Jensen-Shannon divergence

The KL divergence is not symmetric (i.e., $D_{KL}[p(x)||q(x)] \neq D_{KL}[q(x)||p(x)]$). The Jensen-Shannon divergence is a measure of distance that is symmetric by construction:

$$D_{JS}\Big[p(x)\big|\big|q(x)\Big] = \frac{1}{2}D_{KL}\left[p(x)\bigg|\bigg|\frac{p(x)+q(x)}{2}\right] + \frac{1}{2}D_{KL}\left[q(x)\bigg|\bigg|\frac{p(x)+q(x)}{2}\right]. \tag{C.30}$$

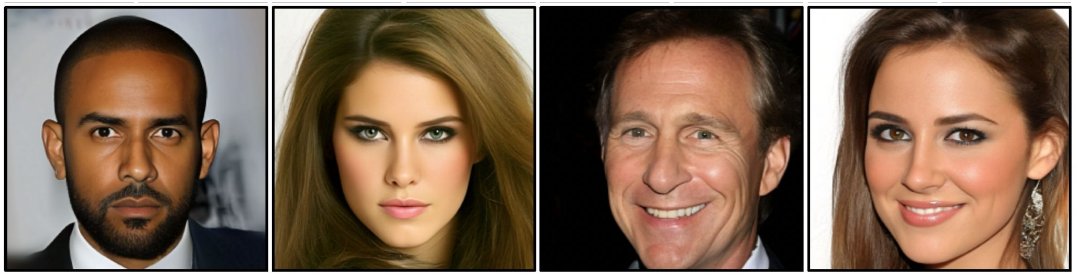
It is the mean divergence of $p(x)$ and $q(x)$ to the average of the two distributions.

Dall-E 3

Outline

- Autoencoder and its limitations
- Intuition behind VAEs
- Derivation of VAE
- Example applications

Generating high quality images



Vahdat & Kautz (2020) "NVAE: A deep hierarchical variational autoencoder"

Resynthesizing real data with changes

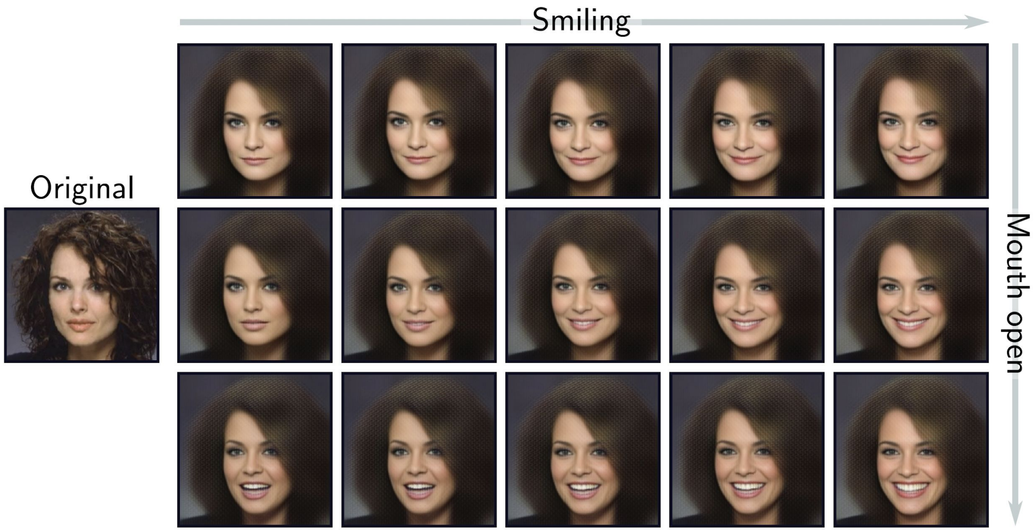
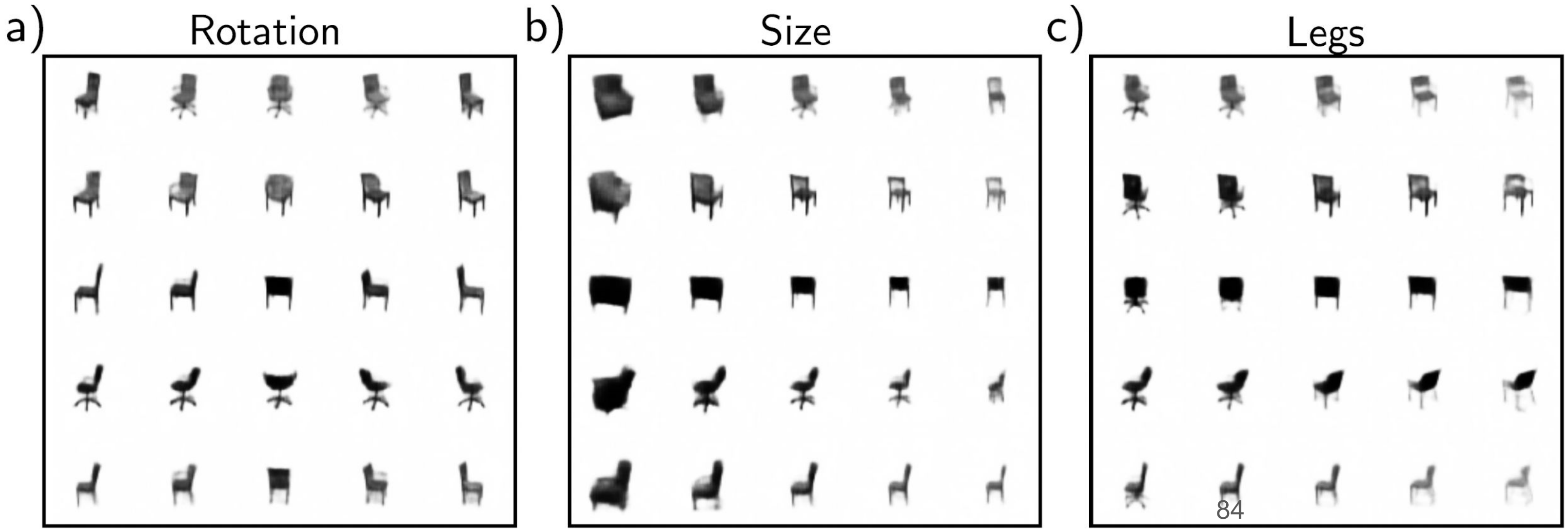
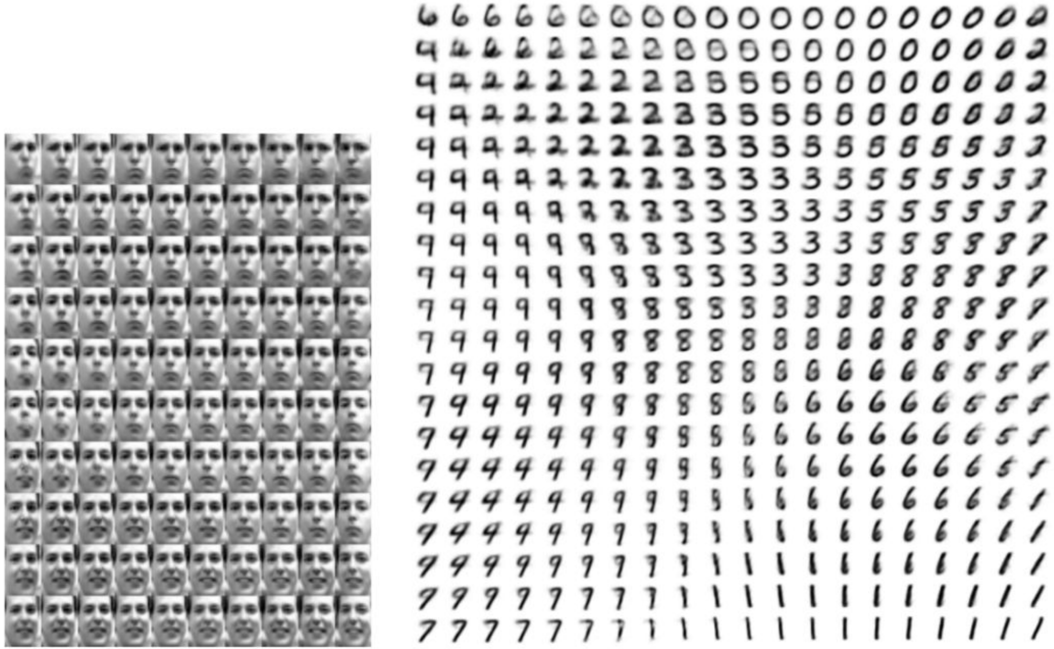


Figure 17.13 Resynthesis. The original image on the left is projected into the latent space using the encoder, and the mean of the predicted Gaussian is chosen to represent the image. The center-left image in the grid is the reconstruction of the input. The other images are reconstructions after manipulating the latent space in directions representing smiling/neutral (horizontal) and mouth open/closed (vertical). Adapted from White (2016).

Disentanglement of the latent space



Chen et al (2021) “Cross-layer distillation with semantic calibration.”

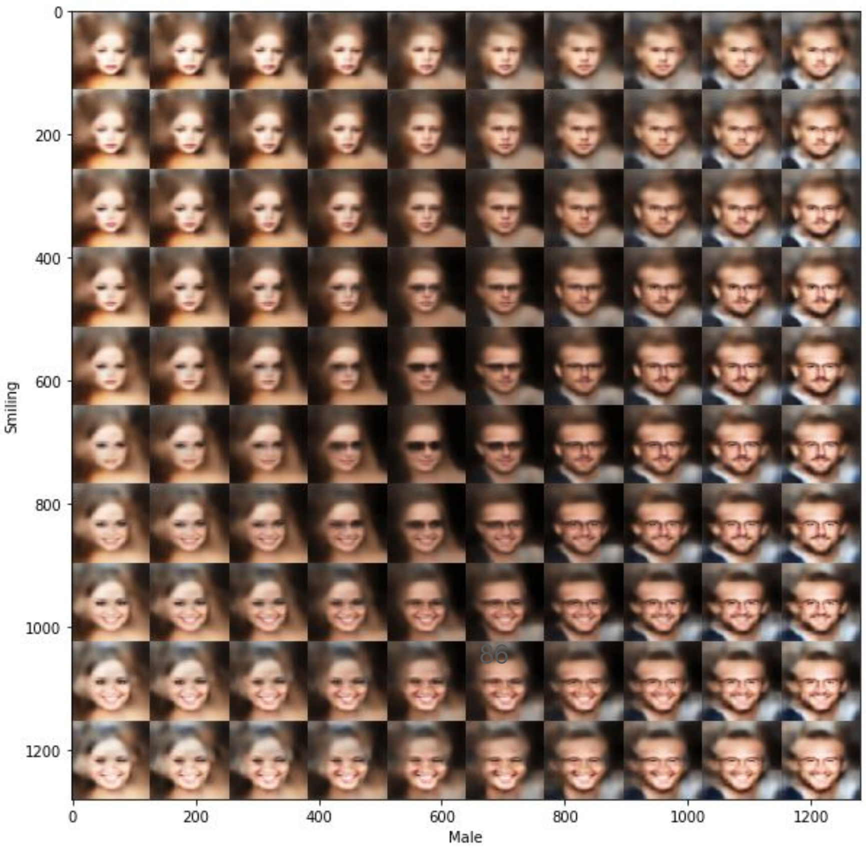


(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Kingma and Welling, “Auto-Encoding Variational Bayes.”, 2013.

Conditional VAEs



Example from
https://towardsdatascience.com/variational-autoencoders-vaes-for-dummies-step-by-step-tutorial-69e6d1c9d8e9

Debiasing

Capable of uncovering **underlying features** in a dataset

vs

Homogeneous skin color, pose

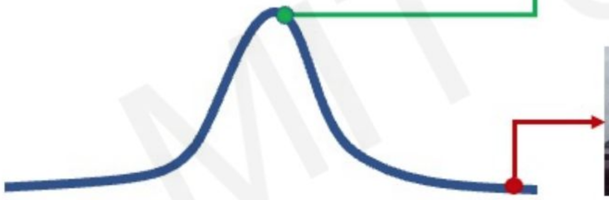
Diverse skin color, pose, illumination

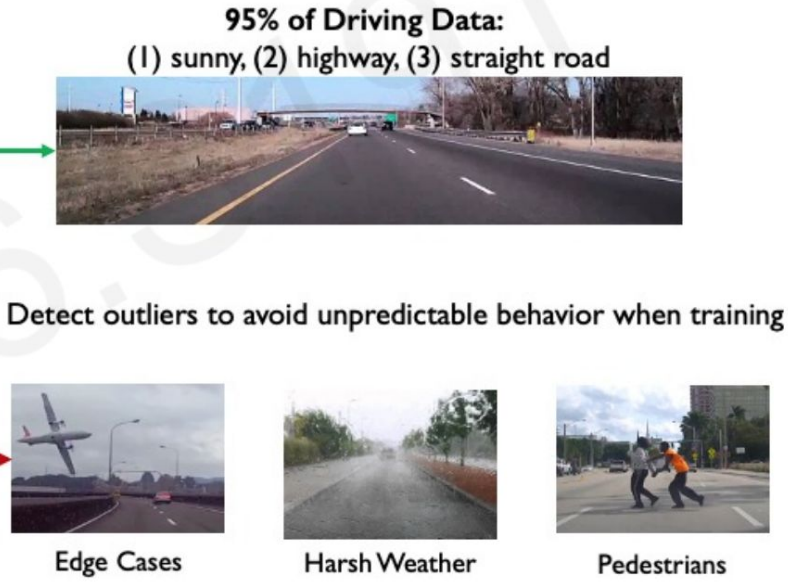
How can we use this information to create fair and representative datasets?

Amini et al, "Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure," 2019

Outlier Detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!





A. Amini et al, "Variational Autoencoder for End-to-End Control of Autonomous Driving with Novelty Detection and Training De-biasing," *2018*

Next Week

- Normalizing Flows (easy inversion / probabilities)
- Diffusion Models (high quality / fast / easy to steer)

Feedback?