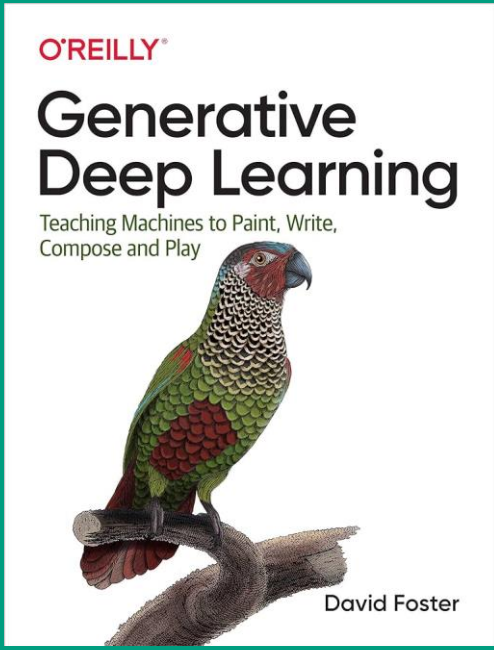
Norwegian University of Life Sciences

Generative Models

Autoencoders (AE) &
Variational Autoencoders (VAE)

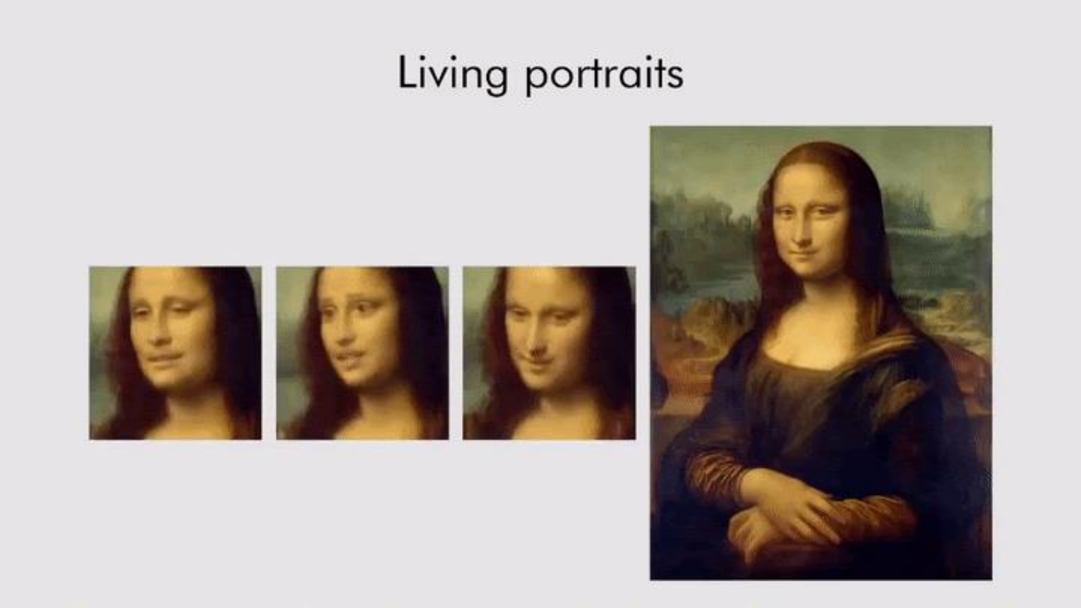
Chapter 3



Motivation (1)

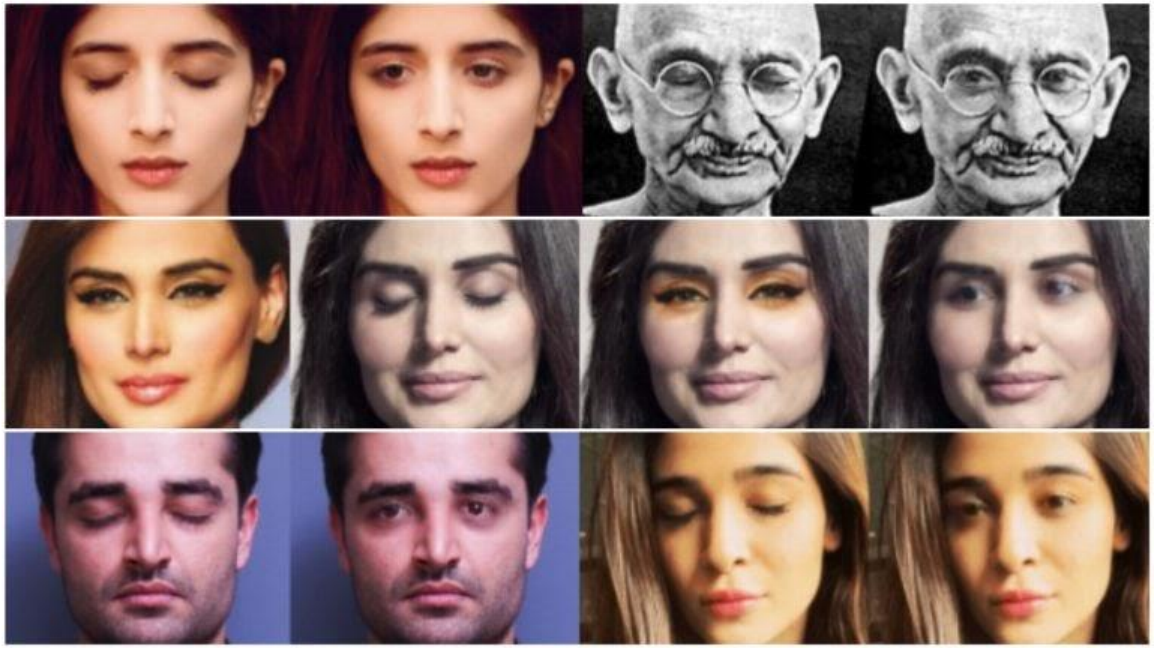
NVIDIA AI

Motivation (2)



DeepFake, Samsung AI

Motivation (3)



Exemplar Generative Adversarial Networks (ExGANs, Facebook)

Problem Definition

The Generative Modeling Framework

- We have a dataset of observations X
- We assume that the observations have been generated according to some unknown distribution, P_{data}
- A generative model P_{model} tries to mimic P_{data}
  - If we achieve this goal, we can sample from P_{model} to generate observations that appear to have been drawn from P_{data}.
- We are impressed by P_{model} if:
  - Rule 1: It can generate examples that appear to have been drawn from P_{data}.
  - Rule 2: It can generate examples that are suitably different from the observations in X.
- In other words, the model shouldn't simply reproduce things it has already seen.

Probabilistic Generative Models

Problem Description:

- It's 2047, and you've just been appointed to create new fashion trends, who are particular about their style.
- You must design new looks that are similar to existing ones but not identical.
- You're given a dataset of 50 fashion styles

Task:

- Generate 10 new looks for the Fashion Police to review, experimenting with hairstyles, hair color, glasses, and clothing

Dataset – 50 Samples (images)

Source: Images sourced from https://getavataaars.com.

Probabilistic Generative Models

Dataset Features:

- 6 different hair styles
- 7 different hair colors
- 3 different kinds of top type
- 4 different kinds of clothing
- 8 different clothing colors

Dataset – 50 Samples (images)

Source: Images sourced from https://getavataaars.com.

Probabilistic Generative Models

Dataset Features:

- 6 different hair styles
- 7 different hair colors
- 3 different kinds of top type
- 4 different kinds of clothing
- 8 different clothing colors

Dataset

- There are:

  $7 \times 6 \times 3 \times 4 \times 8 = 4{,}032$ different combinations

Source: Images sourced from https://getavataaars.com.

Probabilistic Generative Models

- 6 different hair styles
- 7 different hair colors
- 3 different kinds of top type
- 4 different kinds of clothing
- 8 different clothing colors
- There are:

  7 × 6 × 3 × 4 × 8 = 4,032 different combinations

The first 10 observations in the Wrodler face dataset

face_id	accessoriesType	clothingColor	clothingType	hairColor	topType
0	Round	White	ShirtScoopNeck	Red	ShortHairShortFlat
1	Round	White	Overall	SilverGray	ShortHairFrizzle
2	Sunglasses	White	ShirtScoopNeck	Blonde	ShortHairShortFlat
3	Round	White	ShirtScoopNeck	Red	LongHairStraight
4	Round	White	Overall	SilverGray	NoHair
5	Blank	White	Overall	Black	LongHairStraight
6	Sunglasses	White	Overall	SilverGray	LongHairStraight
7	Round	White	ShirtScoopNeck	SilverGray	ShortHairShortFlat
8	Round	Pink	Hoodie	SilverGray	LongHairStraight
9	Round	PastelOrange	ShirtScoopNeck	Blonde	LongHairStraight

Source: Generative Deep Learning,´David Foster

Probabilistic Generative Models

- 6 different hair styles
- 7 different hair colors
- 3 different kinds of top type
- 4 different kinds of clothing
- 8 different clothing colors
- There are:

  $7 \times 6 \times 3 \times 4 \times 8 = 4{,}032$ different combinations

The first 10 observations in the Wrodler face dataset

face_id	accessoriesType	clothingColor	clothingType	hairColor	topType
0	Round	White	ShirtScoopNeck	Red	ShortHairShortFlat
1	Round	White	Overall	SilverGray	ShortHairFrizzle
2	Sunglasses	White	ShirtScoopNeck	Blonde	ShortHairShortFlat
3	Round	White	ShirtScoopNeck	Red	LongHairStraight
4	Round	White	Overall	SilverGray	NoHair
5	Blank	White	Overall	Black	LongHairStraight
6	Sunglasses	White	Overall	SilverGray	LongHairStraight
7	Round	White	ShirtScoopNeck	SilverGray	ShortHairShortFlat
8	Round	Pink	Hoodie	SilverGray	LongHairStraight
9	Round	PastelOrange	ShirtScoopNeck	Blonde	LongHairStraight

- The problem is that we do not know P_{data} explicitly — all we have to work with is the sample of observations X generated by P_{data}.
- The **goal** of generative modeling is to use these observations to build a P_{model} that can accurately mimic the observations produced byP_{data}.

Source: Generative Deep Learning,´David Foster

Naive Bayes

We make the naive assumption that each feature x_j is independent of every other feature x_k.

$$p(x_j \mid x_k) = p(x_j)$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

Naive Bayes

topType	n	$\hat{\theta}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{\theta}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{\theta}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{\theta}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{\theta}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

topType hairColor accessoriesType clothingType clothingColor

p(LongHairStraight, Red, Round, ShirtScoopNeck, White) = ????

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{p}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{p}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{p}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{p}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{p}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

$$p(\overset{\text{topType}}{\text{LongHairStraight}}, \overset{\text{hairColor}}{\text{Red}}, \overset{\text{accessoriesType}}{\text{Round}}, \overset{\text{clothingType}}{\text{ShirtScoopNeck}}, \overset{\text{clothingColor}}{\text{White}}) = \underbrace{0.46 \times 0.16 \times 0.44 \times 0.38 \times 0.44} = = 0.0054$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{\theta}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{\theta}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{\theta}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{\theta}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{\theta}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

$$p(\overset{\text{topType}}{\text{LongHairStraight}}, \overset{\text{hairColor}}{\text{Red}}, \overset{\text{accessoriesType}}{\text{Round}}, \overset{\text{clothingType}}{\text{ShirtScoopNeck}}, \overset{\text{clothingColor}}{\text{White}}) = \underbrace{0.46 \times 0.16 \times 0.44 \times 0.38 \times 0.44}_{} = = 0.0054$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{p}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{p}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{p}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{p}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{p}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

$$p(\overset{\text{topType}}{\text{LongHairStraight}}, \overset{\text{hairColor}}{\text{Red}}, \overset{\text{accessoriesType}}{\text{Round}}, \overset{\text{clothingType}}{\text{ShirtScoopNeck}}, \overset{\text{clothingColor}}{\text{White}}) = 0.46 \times 0.16 \times 0.44 \times 0.38 \times 0.44 = = 0.0054$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{\theta}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{\theta}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{\theta}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{\theta}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{\theta}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

topType hairColor accessoriesType clothingType clothingColor

p(LongHairStraight, Red, Round, ShirtScoopNeck, White) = $\underbrace{0.46 \times 0.16 \times 0.44 \times 0.38 \times 0.44}$ = = 0.0054

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{p}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{p}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{p}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{p}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{p}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

$$p(\overset{\text{topType}}{\text{LongHairStraight}}, \overset{\text{hairColor}}{\text{Red}}, \overset{\text{accessoriesType}}{\text{Round}}, \overset{\text{clothingType}}{\text{ShirtScoopNeck}}, \overset{\text{clothingColor}}{\boxed{\text{White}}}) = \underbrace{0.46 \times 0.16 \times 0.44 \times 0.38 \times \boxed{0.44}} = = 0.0054$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{\theta}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{\theta}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{\theta}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{\theta}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{\theta}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

Source: Generative Deep Learning,´David Foster

Naive Bayes

$$p(\overset{\text{topType}}{\text{LongHairStraight}}, \overset{\text{hairColor}}{\text{Red}}, \overset{\text{accessoriesType}}{\text{Round}}, \overset{\text{clothingType}}{\text{ShirtScoopNeck}}, \overset{\text{clothingColor}}{\text{White}}) = 0.46 \times 0.16 \times 0.44 \times 0.38 \times 0.44 = = 0.0054$$

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k)$$

topType	n	$\hat{\theta}$
NoHair	7	0.14
LongHairBun	0	0.00
LongHairCurly	1	0.02
LongHairStraight	23	0.46
ShortHairShortWaved	1	0.02
ShortHairShortFlat	11	0.22
ShortHairFrizzle	7	0.14
Grand Total	*50*	*1.00*

hairColor	n	$\hat{\theta}$
Black	7	0.14
Blonde	6	0.12
Brown	2	0.04
PastelPink	3	0.06
Red	8	0.16
SilverGrey	24	0.48
Grand Total	*50*	*1.00*

clothingColor	n	$\hat{\theta}$
Black	0	0.00
Blue01	4	0.08
Grey01	10	0.20
PastelGreen	5	0.10
PastelOrange	2	0.04
Pink	4	0.08
Red	3	0.06
White	22	0.44
Grand Total	*50*	*1.00*

accessoriesType	n	$\hat{\theta}$
Blank	11	0.22
Round	22	0.44
Sunglasses	17	0.34
Grand Total	*50*	*1.00*

clothingType	n	$\hat{\theta}$
Hoodie	7	0.14
Overall	18	0.36
ShirtScoopNeck	19	0.38
ShirtVNeck	6	0.12
Grand Total	*50*	*1.00*

In generative models, this means the model can create new combinations of features that weren’t in the original dataset, but it still assigns a likelihood (nonzero probability) to them. This allows the model to generate realistic new examples based on patterns it has learned, even if they weren’t seen during training.

Source: Generative Deep Learning,´David Foster

Naive Bayes (from features -> Pixels)

32 * 32 pixels images = 1024 pixels or features

the total number of combinations is:

$$256^{1024}$$

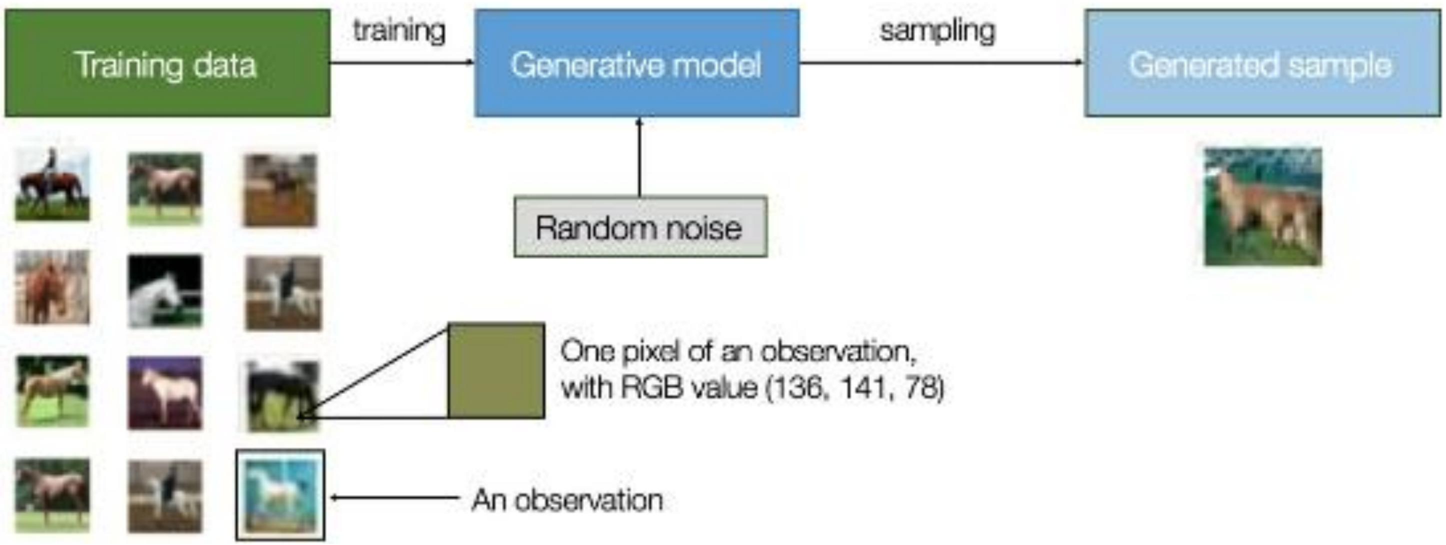
Source: Images sourced from https://getavataaars.com.

Naive Bayes (from features -> Pixels)

The ten new pixel styles, generated by the Naive Bayes model

- Naive Bayes can fail in image generation because it assumes pixels are independent, which isn't true in images where neighboring pixels are highly related.
- It also struggles with high-dimensional data (like 1,024 pixels) and can't capture complex or non-linear relationships between pixels. More advanced models, are better suited for handling the problem

Generative Models



Foster, David. Generative Deep Learning . O'Reilly Media. Kindle-Version.

Generative Models Challenges

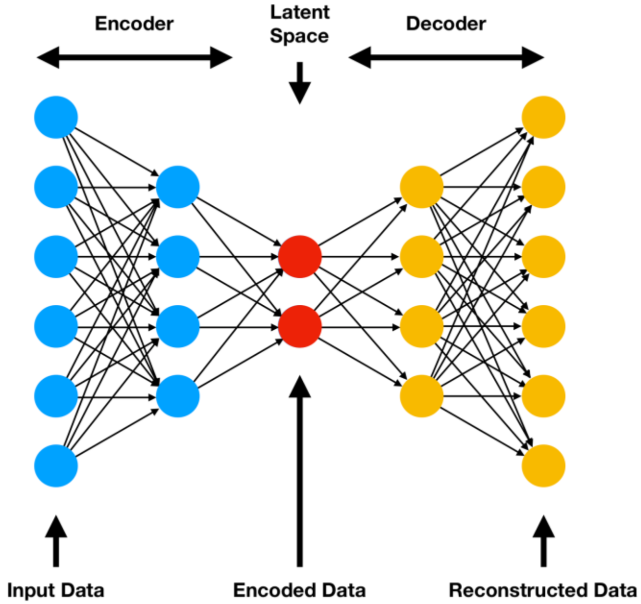
- How does the model cope with the high degree of conditional dependence between features?
- How does the model find one of the tiny proportion of satisfying possible generated observations among a high-dimensional sample space?

Autoencoders

- An autoencoder is a type of artificial neural network used for unsupervised learning. Its primary goal is to learn a compressed representation of input data, and it does this by encoding the data into a lower-dimensional latent space and then decoding it back to its original form. The entire process is meant to capture the most salient features of the data.

The autoencoder structure:

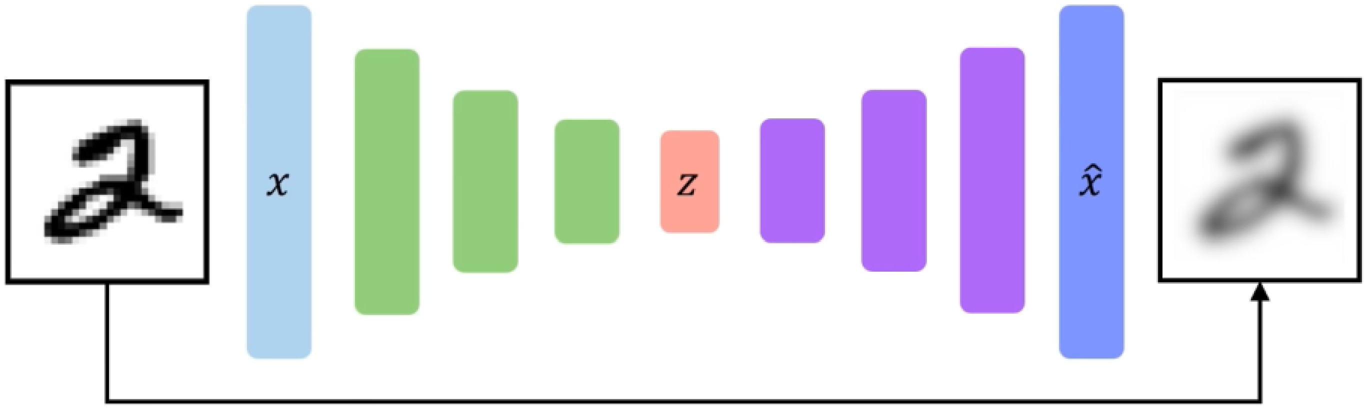
- **Encoder:** This part of the network compresses the input into a latent-space representation. It encodes the input data as an internal fixed-size representation in reduced dimensionality.
- **Latent Space:** This is the compressed representation of the input data. It holds the key features necessary to reconstruct the input data.
- **Decoder:** This part of the network reconstructs the input data from the internal representation. It maps the encoded data back to the original space.



https://www.compthree.com/blog/autoencoder/

Autoencoders

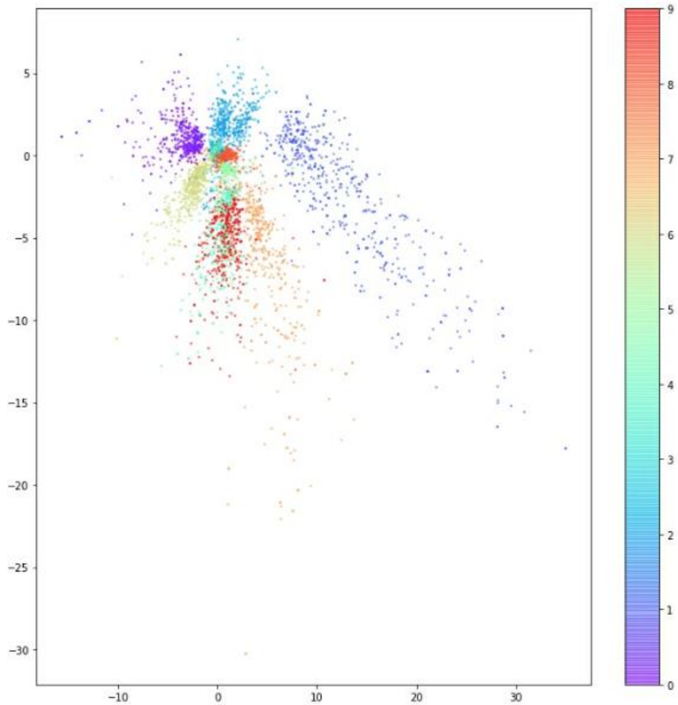
- An encoder network that compresses high-dimensional input data into a lower-dimensional representation vector
- A decoder network that decompresses a given representation vector back to the original domain



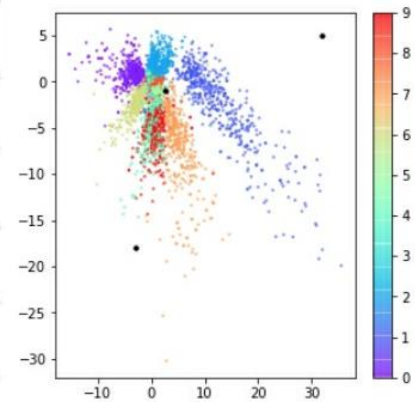
$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Image source: MIT 6.S19

Autoencoders



The latent space of the autoencoder colored by digit



Some poorly generated images

Autoencoders

Examples of Applications and:

- Autoencoders are used to reduce the size of data for faster storage and transmission.
- Autoencoders can learn to remove noise from corrupted data (e.g., denoising images).
- Autoencoders can identify unusual or rare patterns (anomalies) in data because they will struggle to reconstruct patterns that are not similar to the training data.
- They help extract meaningful features from data, often used as a pre-training step in more complex models.

Class Activity

Problem

You work at a hospital where doctors rely on **medical imaging** (e.g., MRI) to diagnose patients. Most of the scans show **healthy tissue**, but sometimes there are **anomalies**, like tumors, fractures, or other abnormal conditions, that need to be flagged for further review.

Challenge
The challenge is that **abnormalities are rare** in the dataset, and doctors are often overloaded with reviewing scans manually. You need to develop a system that can automatically **detect anomalies** in medical images by learning what healthy tissue looks like and flagging anything that looks unusual.

Goal
The system should automatically identify whether a new medical image contains an anomaly (e.g., a tumor or lesion). However, since most images show healthy tissue and there are very few examples of abnormal scans, you will need to build a model that learns the patterns of **normal (healthy) scans** and flags anything that doesn't match those patterns as potentially abnormal

Autoencoders

- See the AutoEncoder Example in Gnerative_Models_AE_VAE.ipynb

Variational Autoencoders

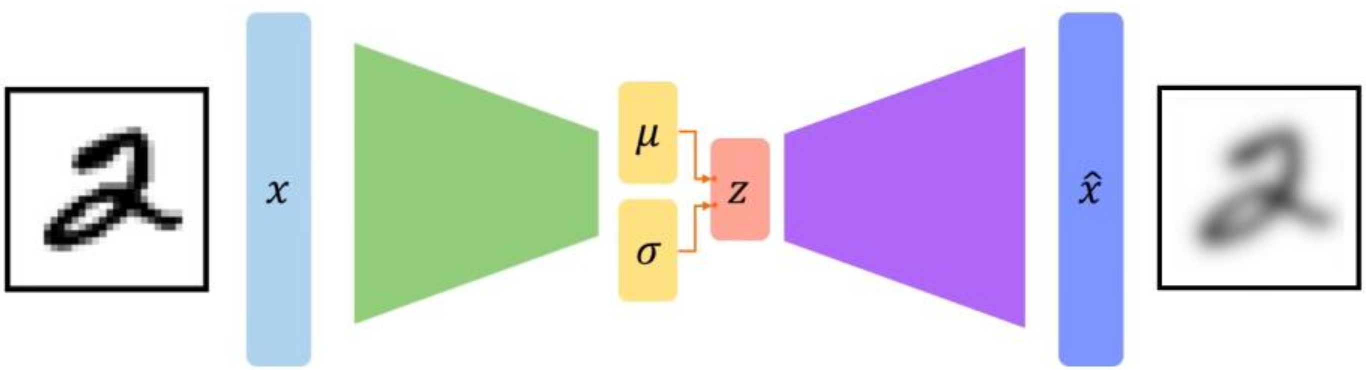
Variational Autoencoders (VAEs) are a type of autoencoder that introduces probabilistic reasoning and optimization techniques from variational inference to create a generative model. While standard autoencoders are trained to minimize the reconstruction error between the input data and their output, VAEs aim to generate new samples that could have been produced by the input data.

The variational autoencoder structure:

Encoder: Like traditional autoencoders, VAEs have an encoder that maps the input data to a latent space. However, instead of encoding the input as a single fixed point in the latent space, the VAE encoder outputs parameters of a probability distribution (usually Gaussian) over the latent space.

Sampling: A sample is drawn from this distribution to provide a randomized latent space representation of the input. This introduces a stochastic element that aids in generating diverse outputs.

Decoder: The sampled latent point is then passed through the decoder to generate a reconstruction of the input



Variational Autoencoders

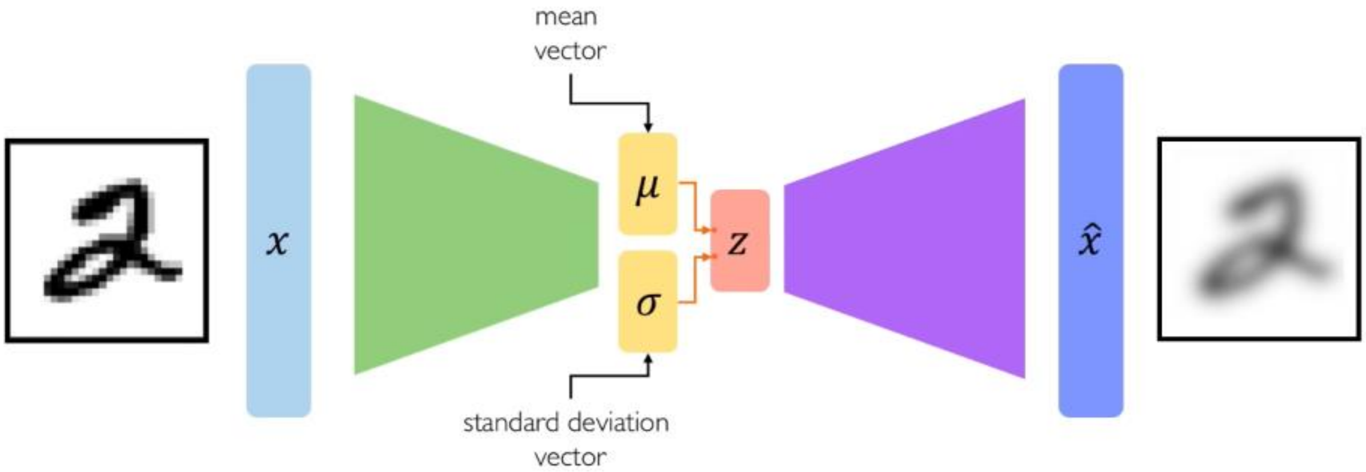
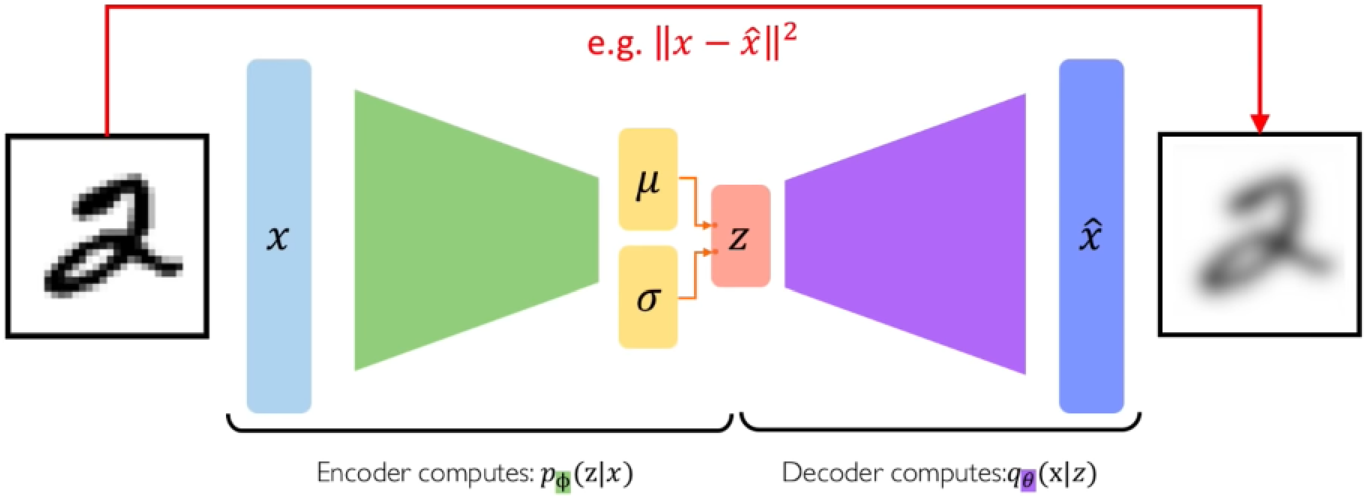


Image source: MIT 6.S19

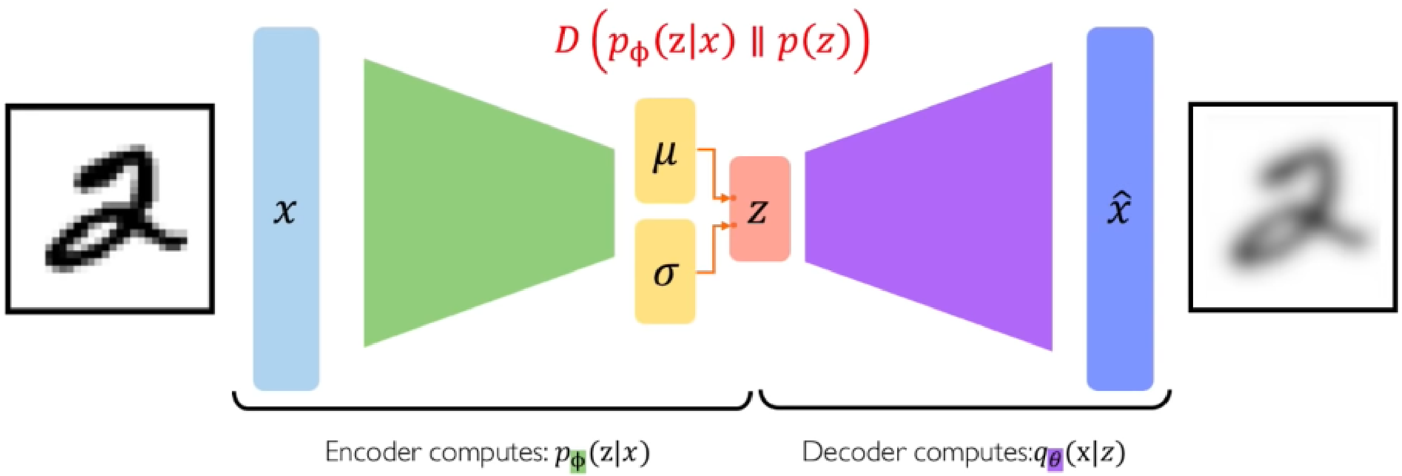
Variational Autoencoders



$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

Image source: MIT 6.S19

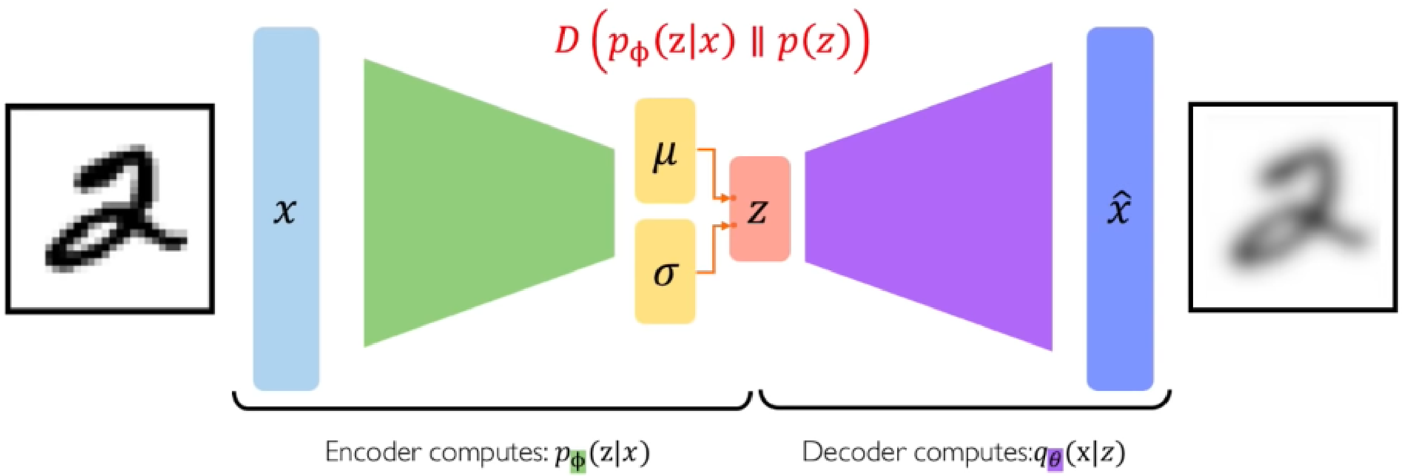
Variational Autoencoders



$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + \boxed{(\text{regularization term})}$$

Image source: MIT 6.S19

Variational Autoencoders



$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + \boxed{(\text{regularization term})}$

KL divergence has the closed form:

```
kl_loss = -0.5 * sum(1 + log_var - mu ^ 2 - exp(log_var))
```

or in mathematical notation:

$$D_{KL}[N(\mu,\ \sigma \parallel N(0,\ 1)] = \frac{1}{2}\sum(1 + log(\sigma^2) - \mu^2 - \sigma^2)$$

The sum is taken over all the dimensions in the latent space.

Image source: MIT 6.S19

Variational Autoencoders - Reparameterization

- **Objective:** Sample from while allowing backpropagation.

Key Idea:

$$\cancel{z \sim \mathcal{N}(\mu, \sigma^2)}$$

Consider the sampled latent vector $\boldsymbol{z}$ as a sum of

- a fixed $\boldsymbol{\mu}$ vector,
- and fixed $\boldsymbol{\sigma}$ vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

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

Image source: MIT 6.S19

Variational Autoencoders - Reparameterization

- **Objective:** Sample from while allowing backpropagation.

$$z_{log\_var} = \log(\sigma^2)$$
$$\sigma^2 = exp(z_{log\_var})$$
$$\sigma = exp(0.5 * z_{log\_var})$$

Key Idea:

$$\cancel{z \sim \mathcal{N}(\mu, \sigma^2)}$$

Consider the sampled latent vector $\boldsymbol{z}$ as a sum of

- a fixed $\boldsymbol{\mu}$ vector,
- and fixed $\boldsymbol{\sigma}$ vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

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

Image source: MIT 6.S19

Variational Autoencoders - Reparameterization

- **Objective:** Sample from while allowing backpropagation.

$$z_{log\_var} = \log(\sigma^2)$$
$$\sigma^2 = exp(z_{log\_var})$$
$$\sigma = exp(0.5 * z_{log\_var})$$

z_log_var: This represents the log variance (log(σ^2)) of the latent variables. We use the log variance instead of the variance for numerical stability and to ensure the variance is always positive when we convert it back using an exponential function.

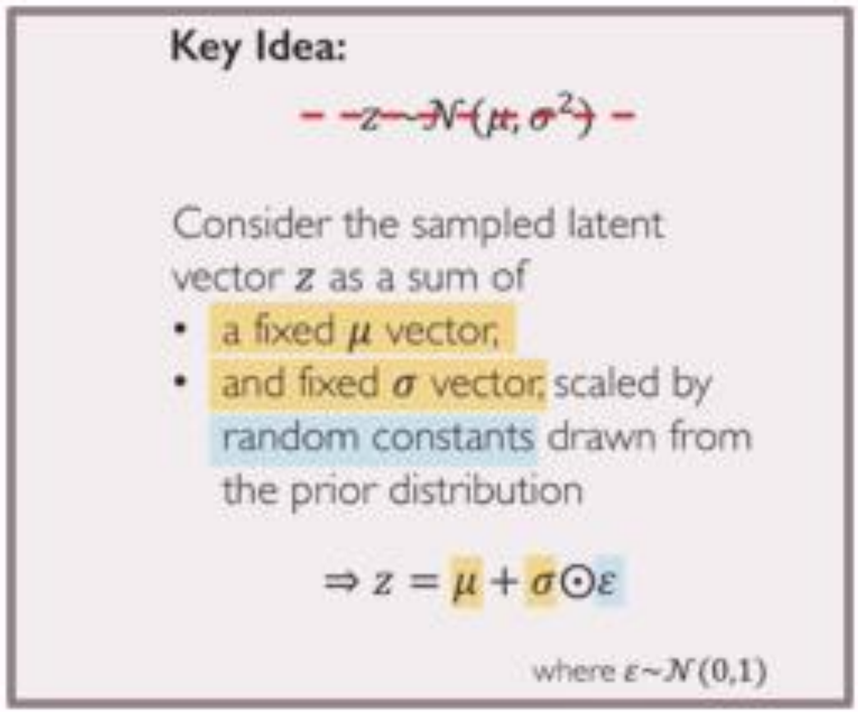


Image source: MIT 6.S19

Variational Autoencoders - Reparameterization

- **Objective:** Sample from while allowing backpropagation.

$$z_{log\_var} = \log(\sigma^2)$$
$$\sigma^2 = exp(z_{log\_var})$$
$$\sigma = exp(0.5 * z_{log\_var})$$

z_log_var: This represents the log variance (log(σ^2)) of the latent variables. We use the log variance instead of the variance for numerical stability and to ensure the variance is always positive when we convert it back using an exponential function.

Key Idea:

$$\cancel{z \sim \mathcal{N}(\mu, \sigma^2)}$$

Consider the sampled latent vector z as a sum of

- a fixed μ vector,
- and fixed σ vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

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

Epsilon: This is a random noise sampled from a standard normal distribution. It introduces stochasticity into our model, allowing us to sample different points from the latent space distribution encoded by z_mean and z_log_var.

Image source: MIT 6.S19

Variational Autoencoders

$$D\left(p_{\phi}(z|x) \parallel p(z)\right)$$

Inferred latent distribution

Fixed prior on latent distribution

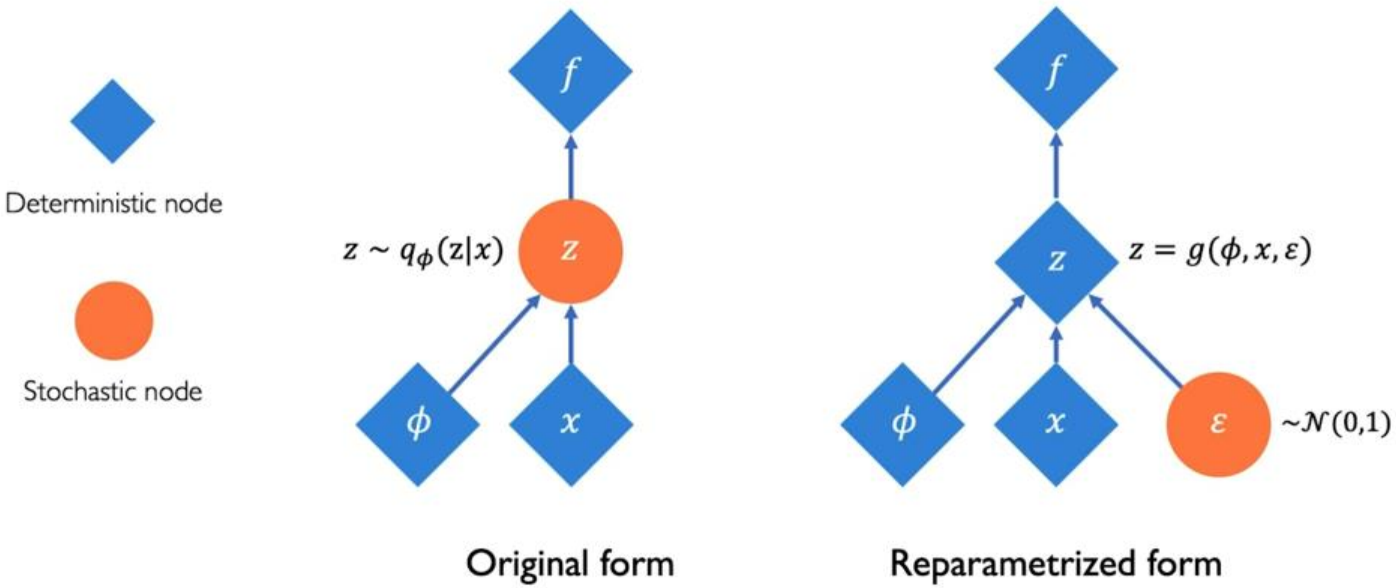
Common choice of prior:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (ie. memorizing the data)

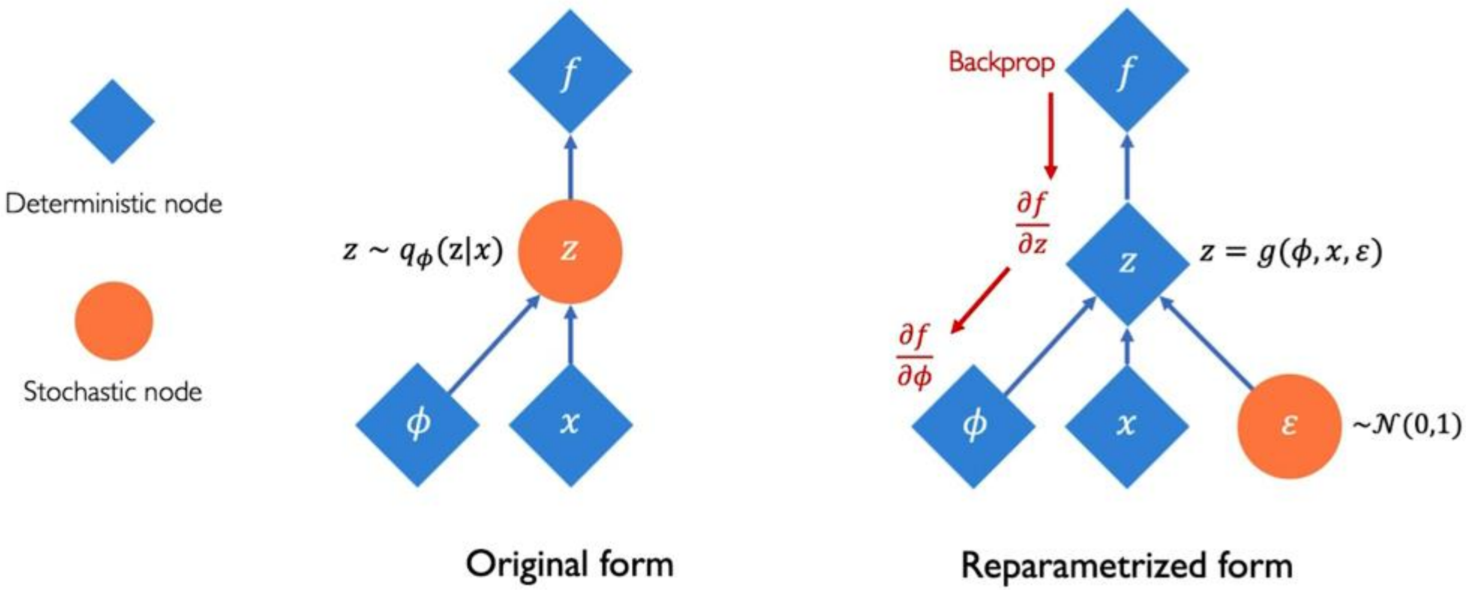
Source: MIT 6.S19

Variational Autoencoders



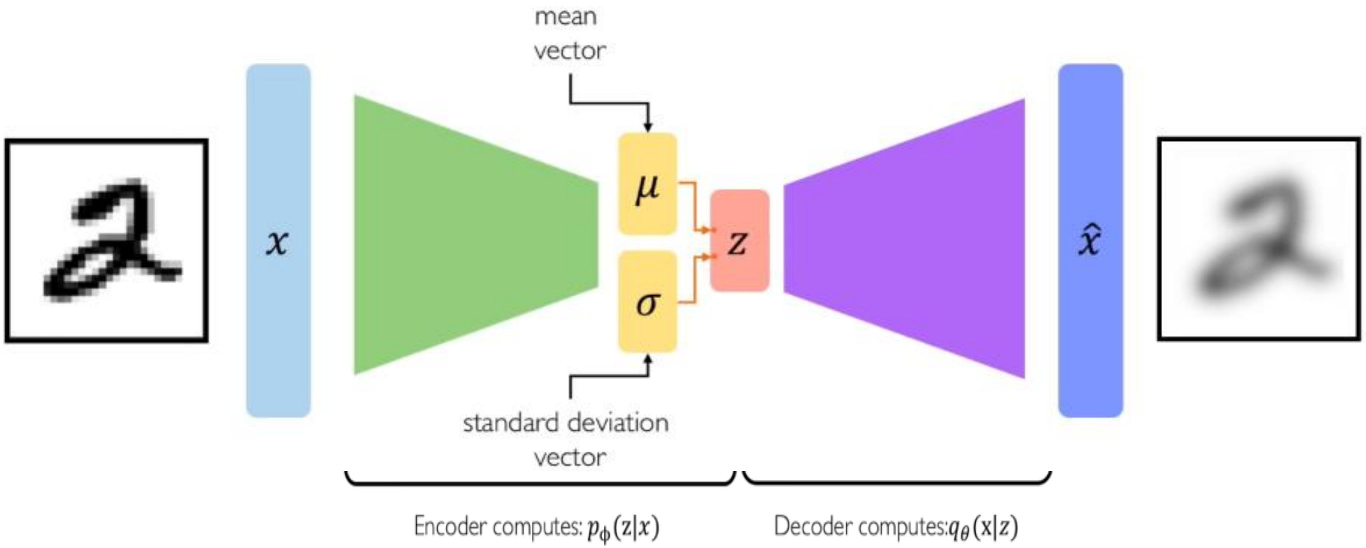
Source: MIT 6.S19

Variational Autoencoders



Source: MIT 6.S19

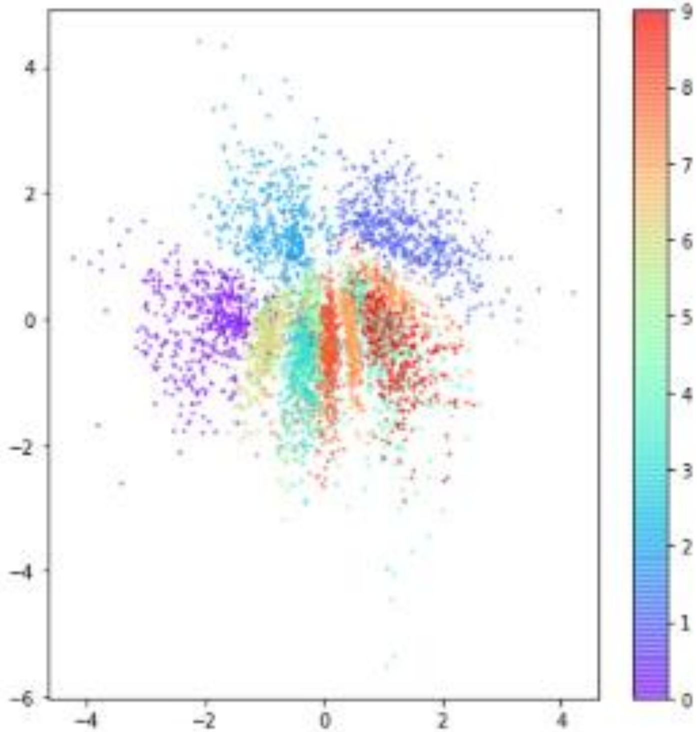
Variational Autoencoders



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{regularization term})$$

A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.

Variational Autoencoders



The latent space of the VAE colored by digit

Source: Generative Deep Learning,´David Foster

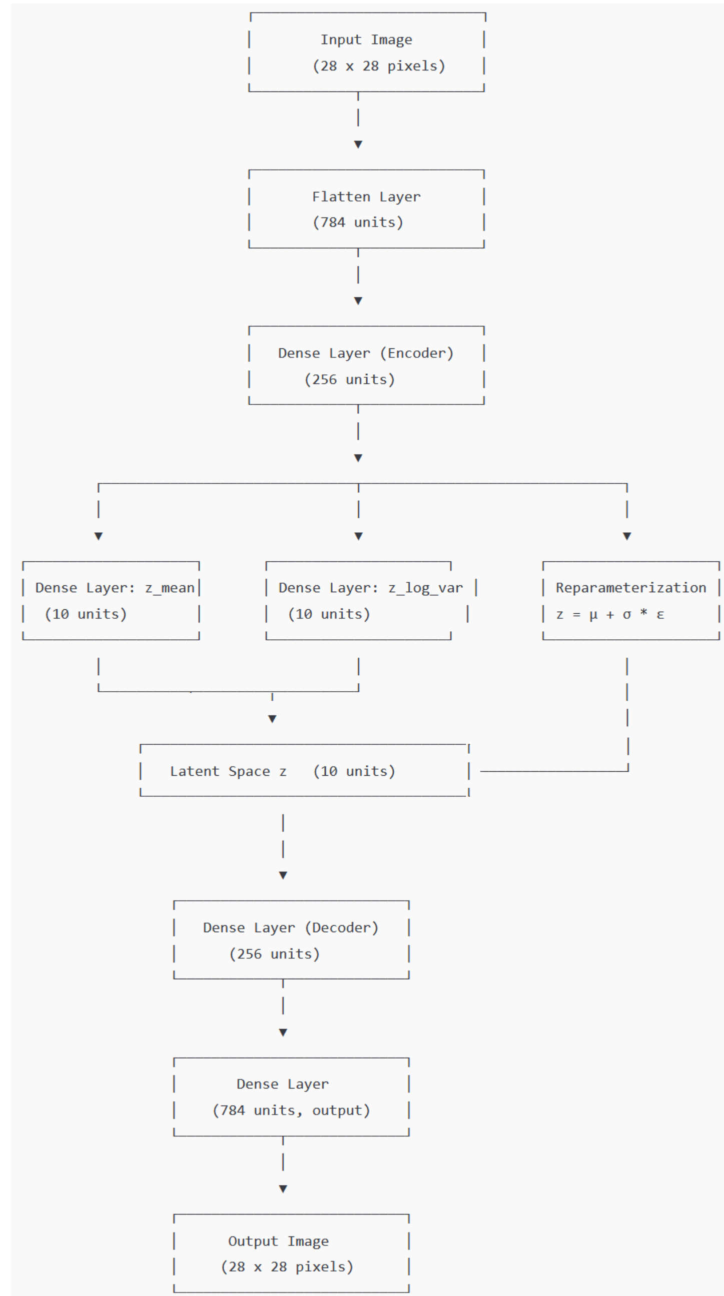
Variational Autoencoders

- See the Variational AutoEncoder Example in Gnerative_Models_AE_VAE.ipynb

Image source: MIT 6.S19

Variational Autoencoders – Class Activity

- Calculate the number of trainable parameters in the given VAE.



Variational Autoencoders – Class Activity

Discuss how Variational Autoencoders (VAEs) can be used to generate realistic scenarios for autonomous vehicle training and testing.