CNNs, Generation

Bryan Cardenas
Robert Jan Schlimbach
Caspar van Leeuwen

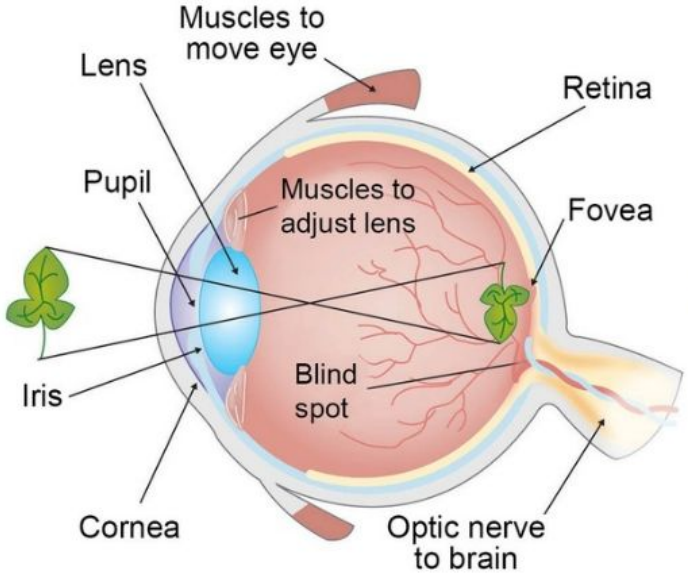
Background Prerequisites

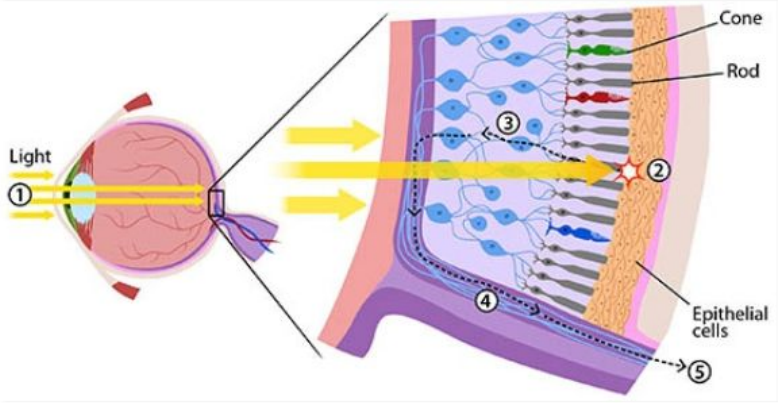
Welcome to the world of computer vision!

How do we let computers *see* something?

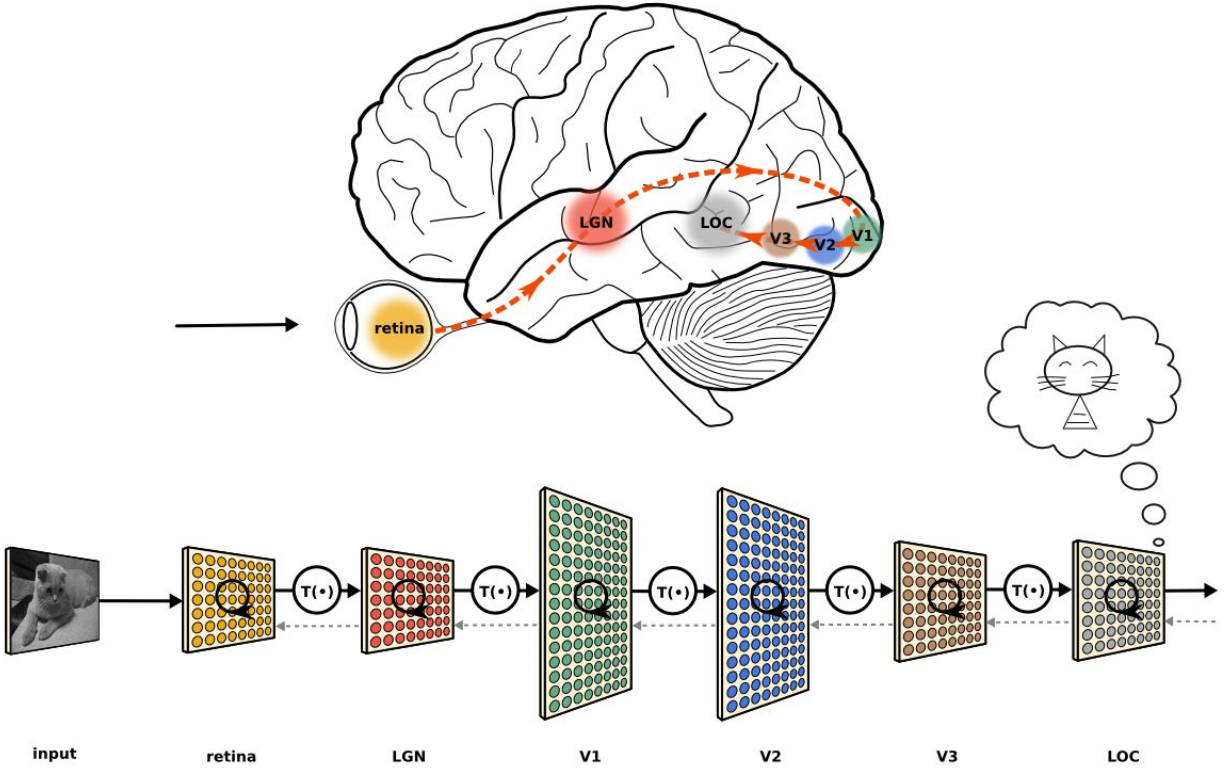
How do *we* see something

Human eyes

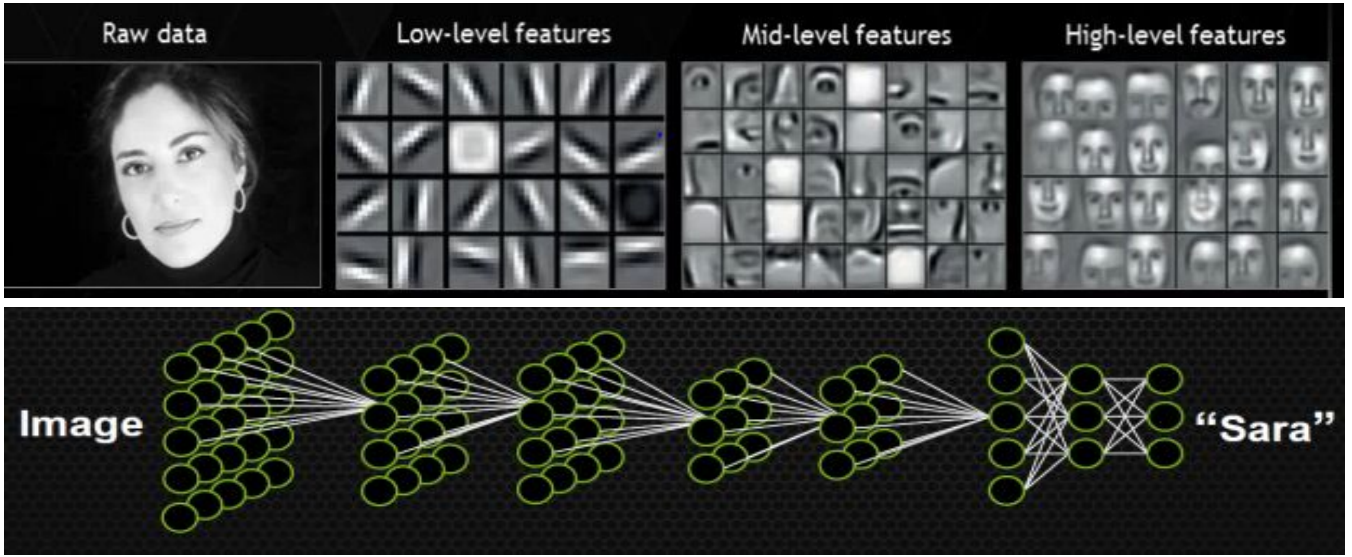

Muscles to move eye
Lens
Retina
Pupil
Muscles to adjust lens
Fovea
Iris
Blind spot
Cornea
Optic nerve to brain


Cone
Rod
Light
1
2
3
4
5
Epithelial cells

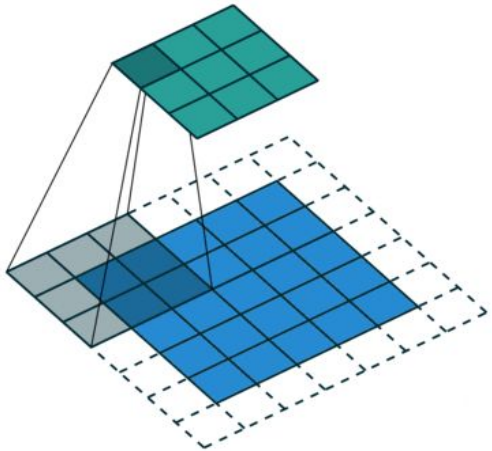
Human eyes

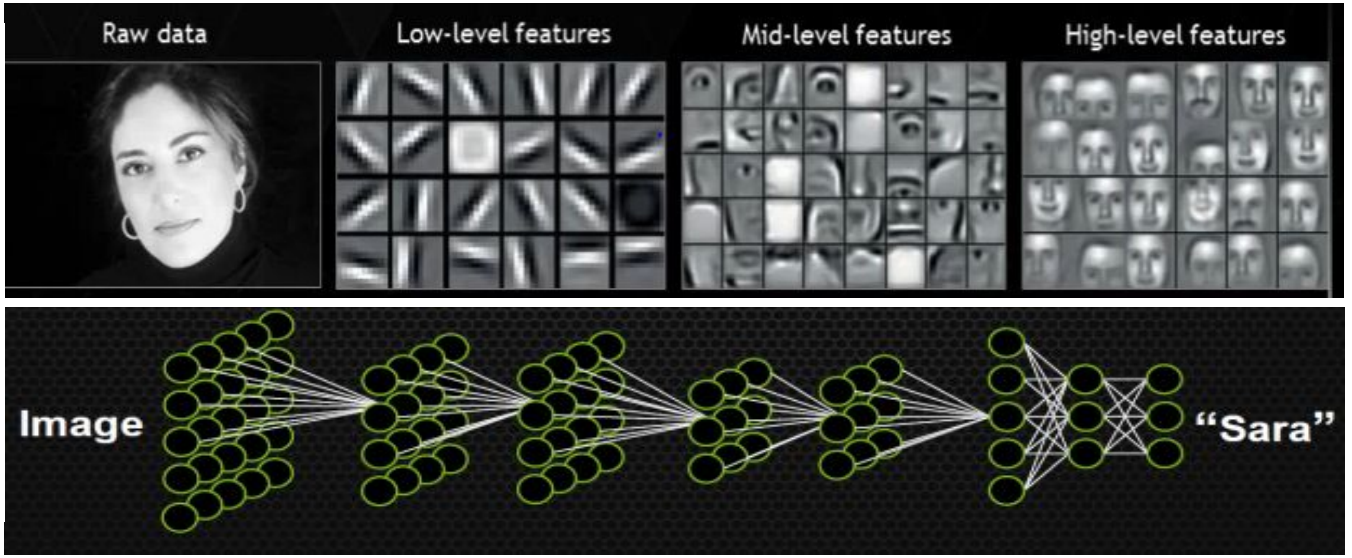


Convolutional Neural Network



Convolutional Neural Network





Representation of images in computers?

An **image**

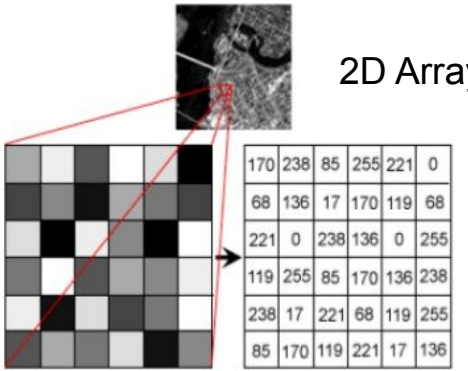
1. Is a matrix/array of intensity values
2. Usually consists of integers [0, 255] or float points [0,1]
3. Each element in the matrix is a **pixel**
4. Can have 1 greyscale channel or multiple colour **channels: RGB**

Representation of images in computers?

An **image**

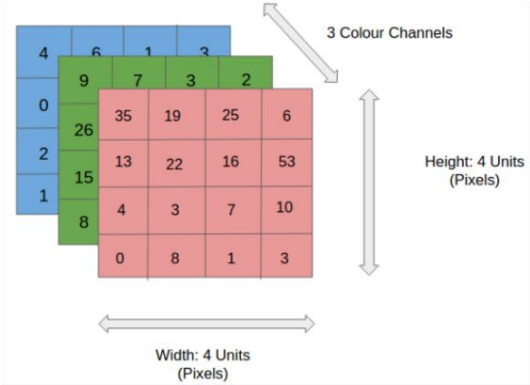
1. Is a matrix/array of intensity values
2. Usually consists of integers [0, 255] or float points [0,1]
3. Each element in the matrix is a **pixel**
4. Can have 1 greyscale channel or multiple colour **channels: RGB**

3D Array. Shape: (width, height, channel)



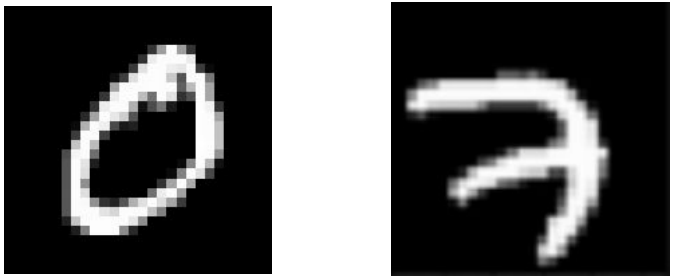
170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

2D Array. Shape = (width, height)

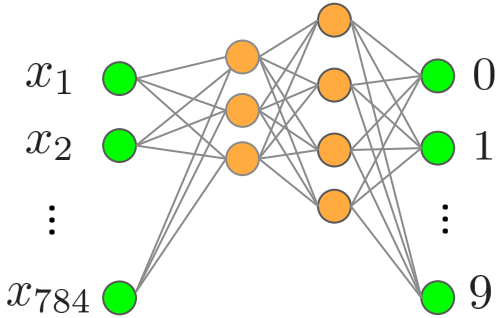


Representation of images in computers?

We will use a neural network to recognize
Hand written digits

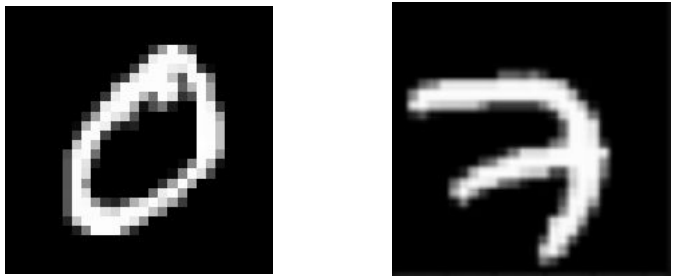


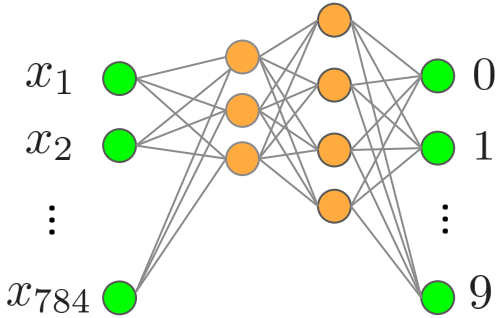
images of size 28x28 = 784 pixels



Representation of images in computers?

We will use a neural network to recognize Hand written digits





images of size 28x28 = 784 pixels

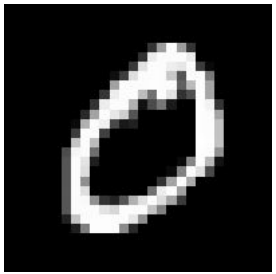
Inefficient: Every pixel is connected to everything

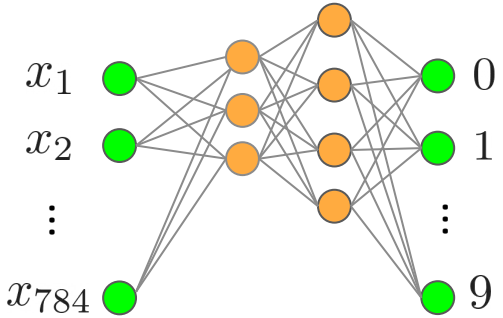
Regions in Images share features

Not shift invariant

Representation of images in computers?

We will use a neural network to recognize Hand written digits



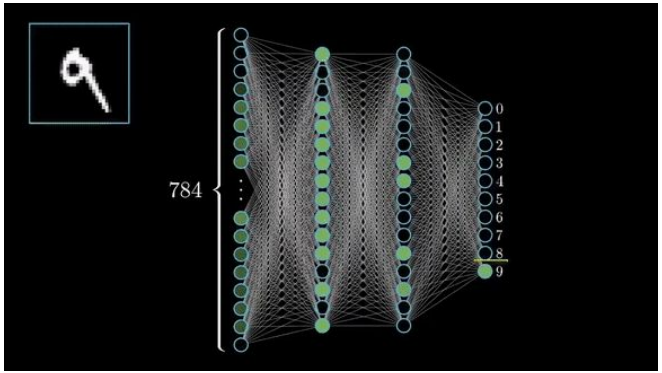


images of size 28x28 = 784 pixels

Inefficient: Every pixel is connected to everything

Regions in Images share features

Not shift invariant



Convolutions

Filters

f

Convolutions

Filters

f

Filter
h

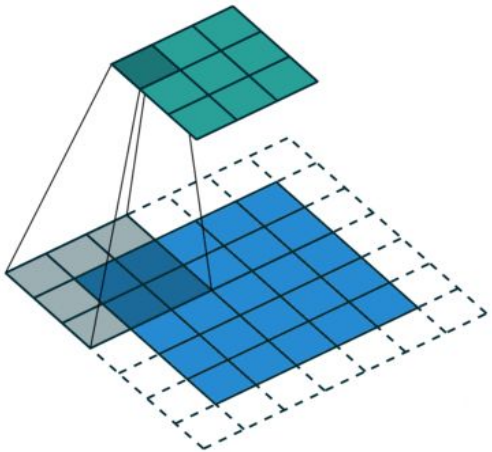
Convolutions

A filter applies a **convolution** operation upon the original signal

Kernel (filter): small matrix that we use to convolve an image

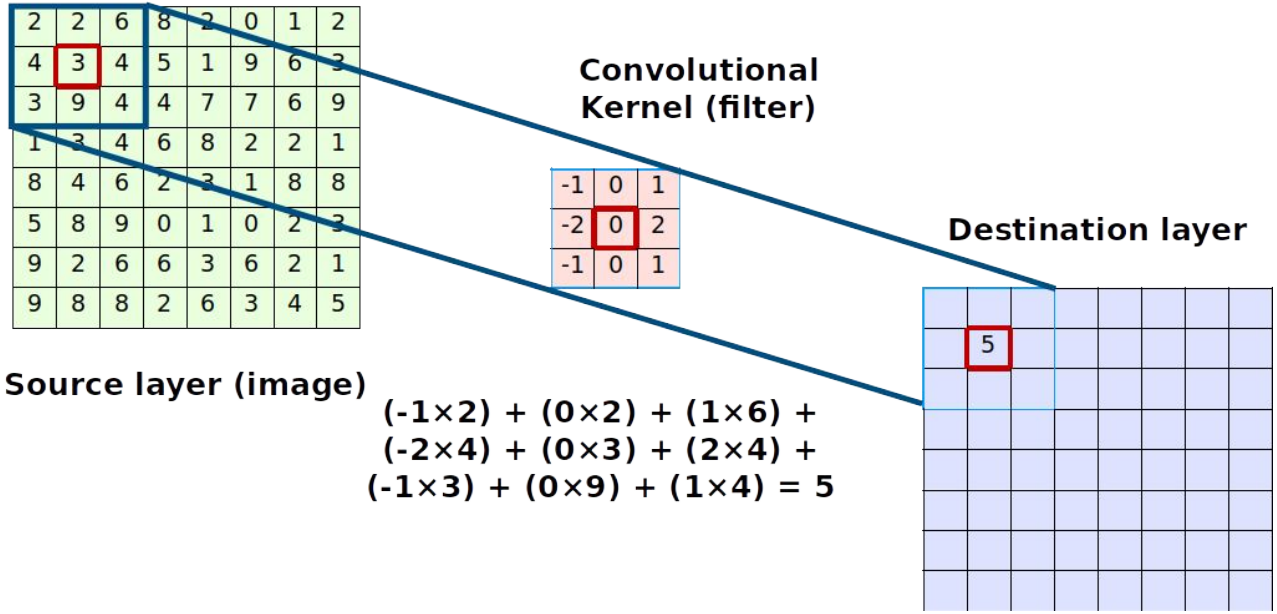
Convolution:

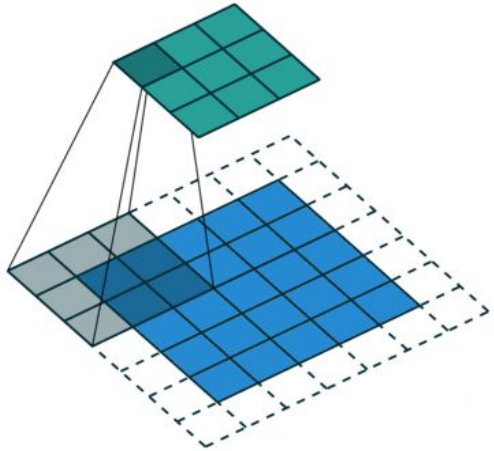
An operation that “blends” one function with another.



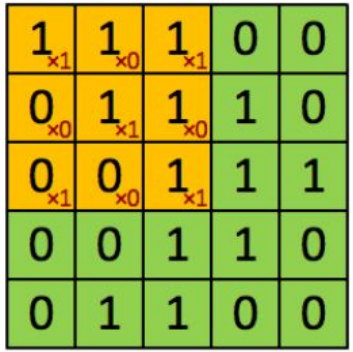
Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Convolutions

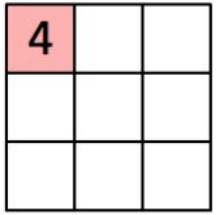




Convolutions

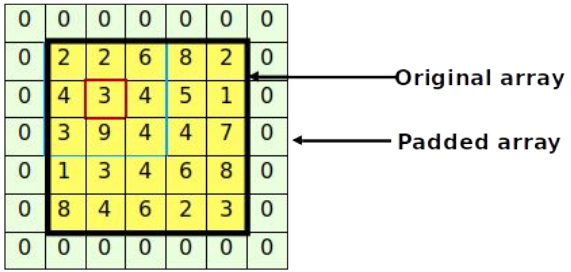


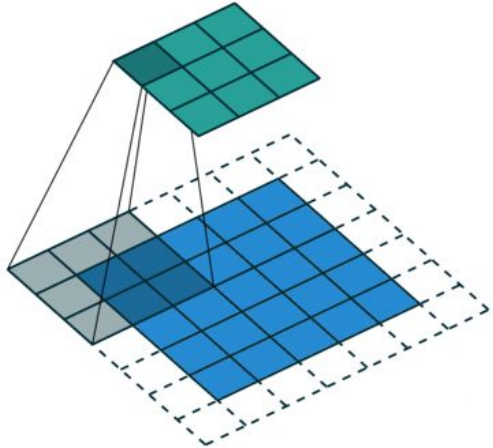
Image



Convolved Feature

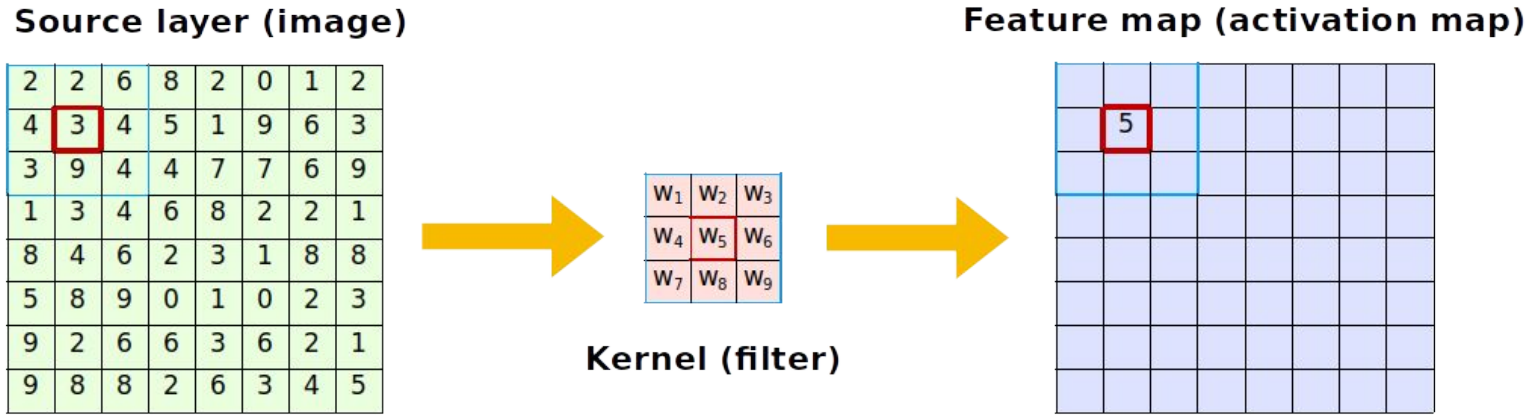
- The kernel is **shifted** over the image with a step size, and computes the output for each position.
- The step size is called **stride**.
- Output has **smaller** dimension than input: dim(output) = dim(input) - (dim(kernel) - 1)
- **Padding** is used to solve this problem, which artificially make the image "bigger" by adding synthesis data (typically 0-padding)





Animation: Sumit Saha/Towards Data Science

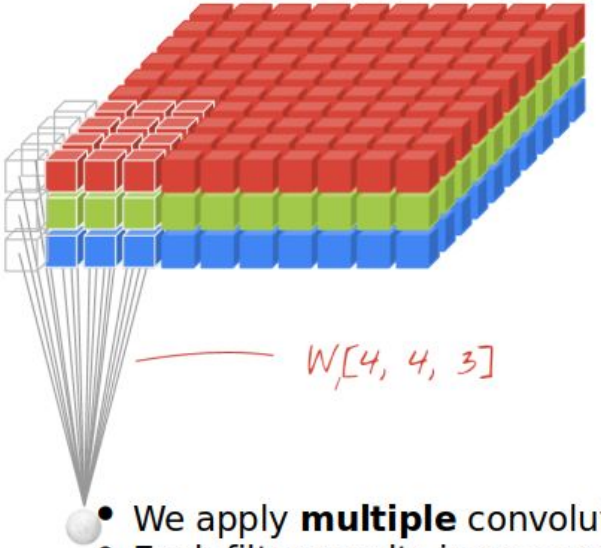
Convolutions

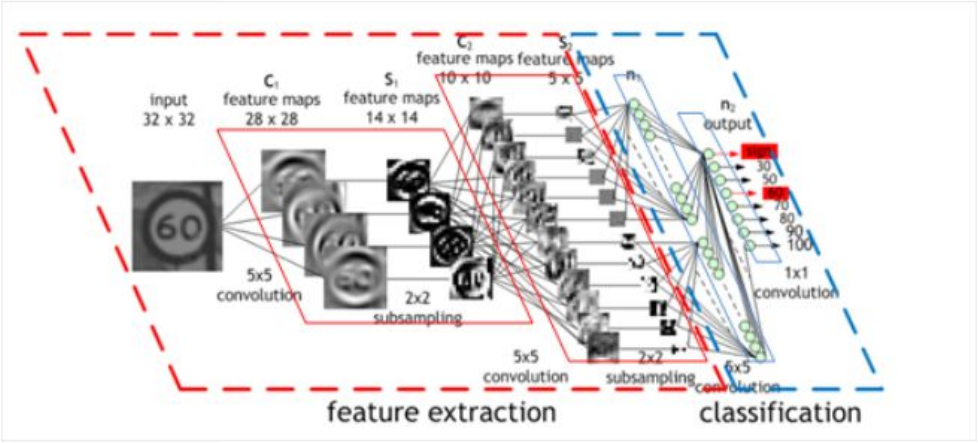


How do we know which kernels to use?

- Kernels are learnt!
- Initialized **randomly**
- Backward propagation; the CNN learns which features to detect

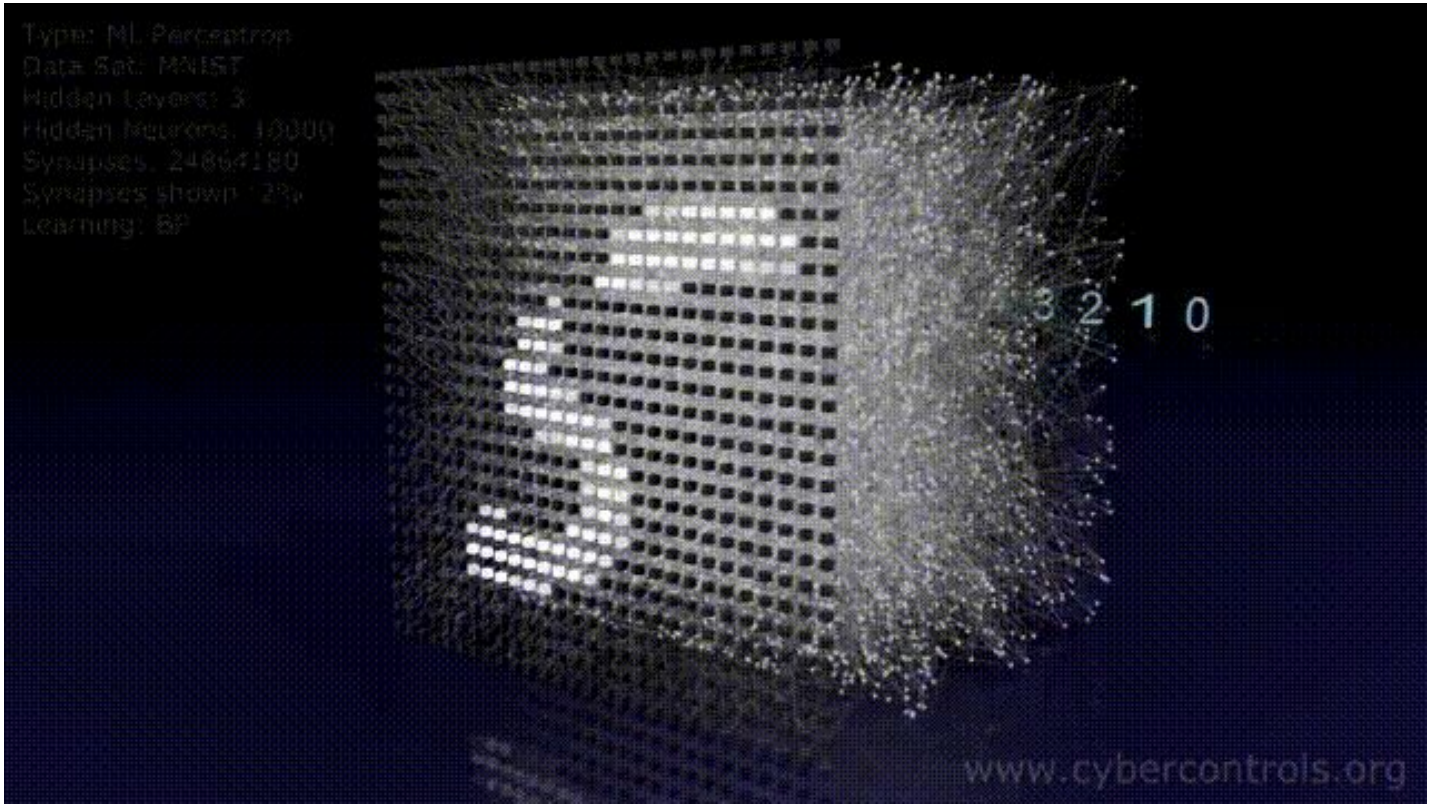
Convolutions



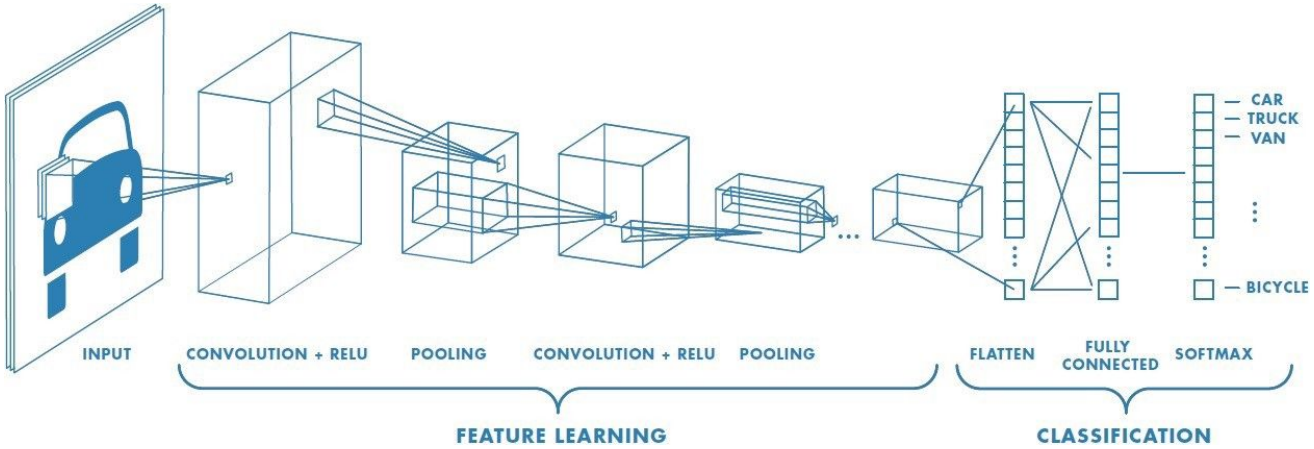


- We apply **multiple** convolutional **filters/kernels** to the **same** image.
- Each filter results in one convolutional **channel**.
- By learning the same image from different channels, one can detect complex patterns.

Convolutions



CNN Form



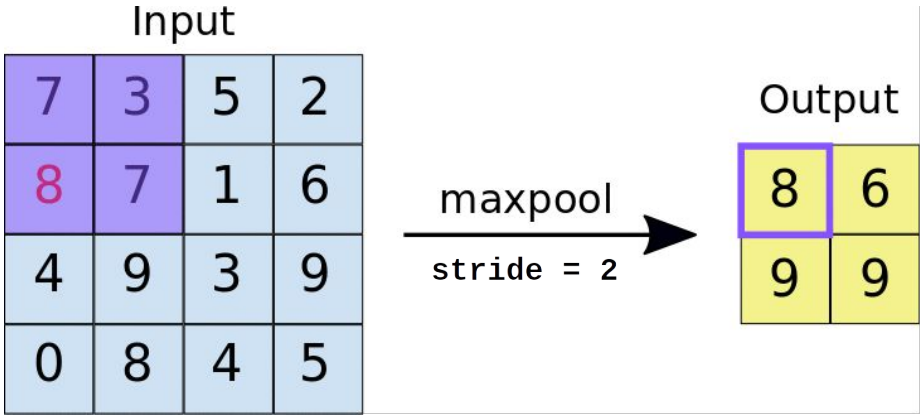
Two parts:

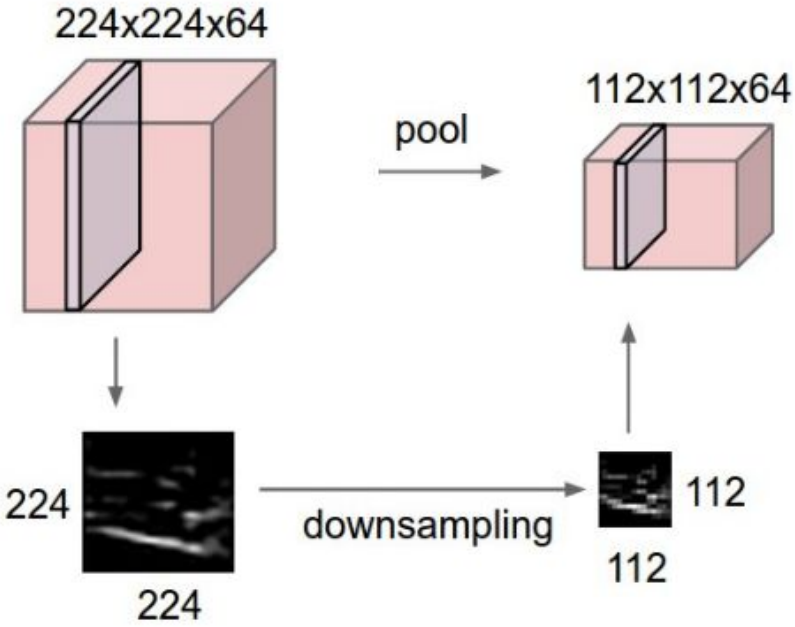
1. Feature Learning
2. Classification (fully connected layers)

Images become smaller and number of filters increases as we go deeper
Complex Images -> we need more filters
Easy images -> we need fewer filters

Pooling (downsampling)

We need to **discard** information **gradually**.
Pooling is a way of information **abstraction**.
Pooling is usually applied **after** convolutions.





Max pooling: keep the **strongest** signal
Average pooling: use the **local average** as the signal

Animation: Google Developers

Transfer Learning

Trained

CNN

Continue Training

CNN reuses its knowledge; its features

Transfer Learning

Trained

CNN

Continue Training

CNN reuses its knowledge; its features

Generative Models

How do we generate images?

What I cannot create, I do not understand

Generative Models

Classification: mapping a distribution to a certain output

Generation: mapping a distribution to a certain output

Happy cat

Angry cat

Relaxed cat

“Cats”

“Cats”

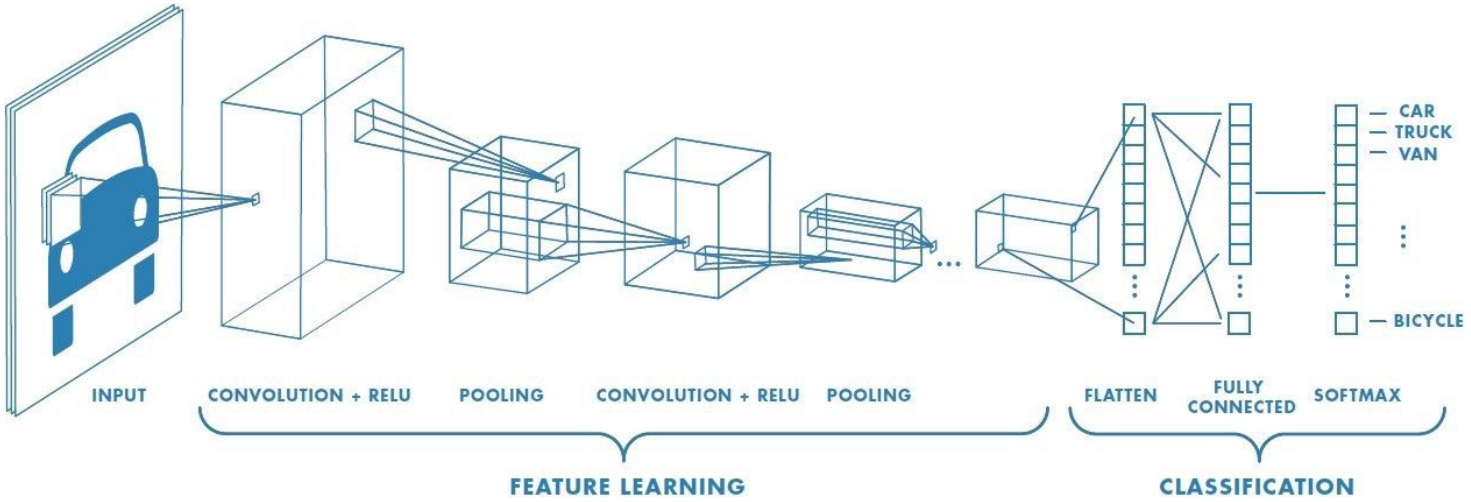
Happy cat

Angry cat

Relaxed cat

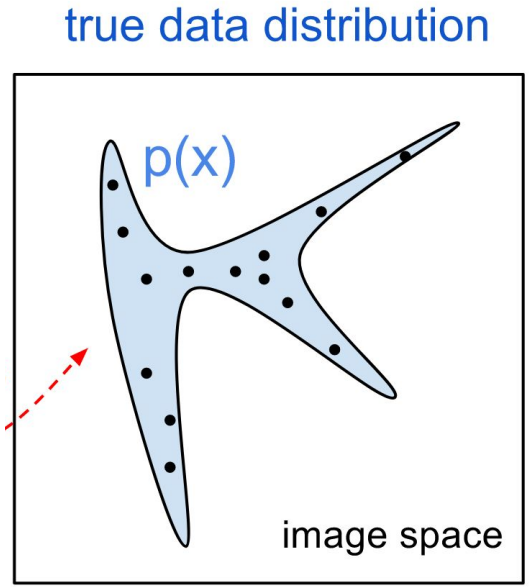
Generative Models

CNN image classifier

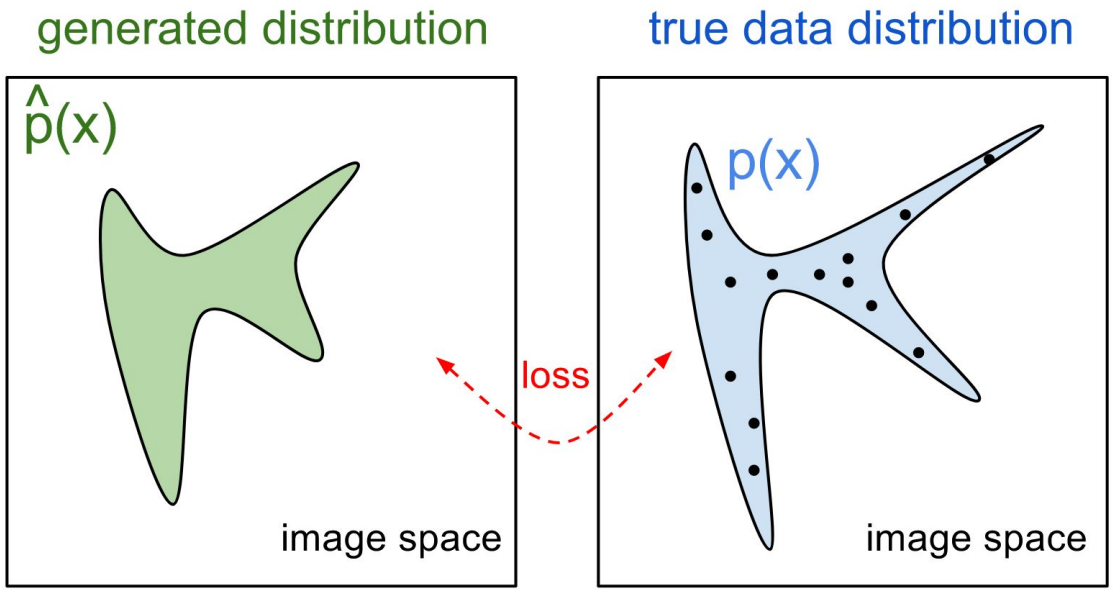


Generative models?

Generative Models



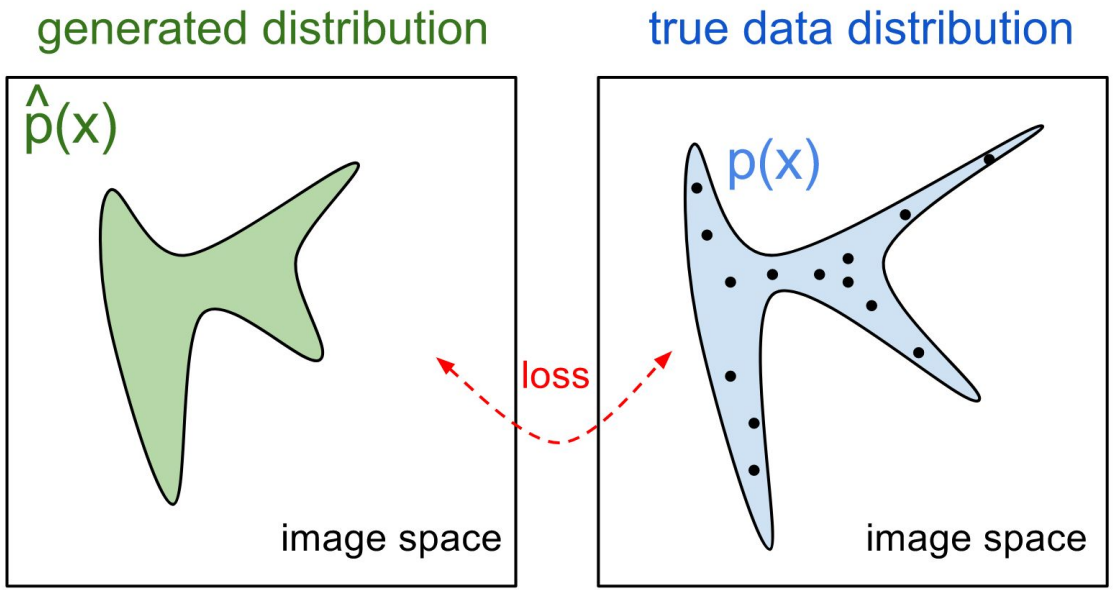
Generative Models



Generative Models

A generative model

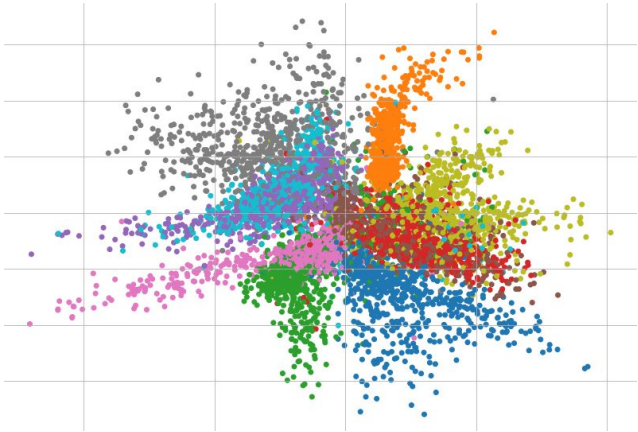
1. Learns the data distribution instead
2. Allows uncertainties to be captures
3. Allows for (condition) density estimation



AutoEncoder

How data is represented is imperative for neural networks

The data representation could have been **engineered** by a human

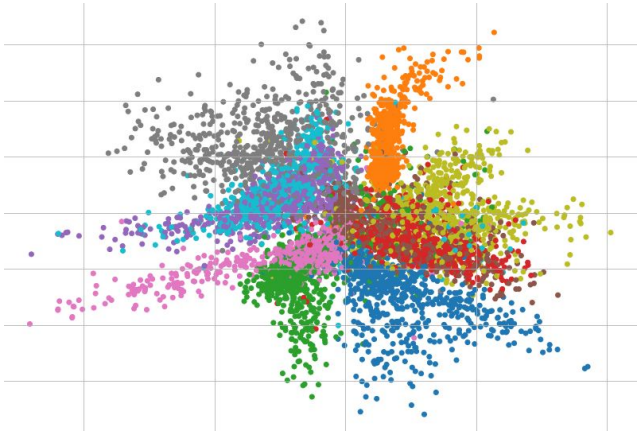


AutoEncoder

How data is represented is imperative for neural networks

The data representation could have been **engineered** by a human

As an alternative: Feed raw data to the neural network and **learn** the representation automatically



AutoEncoder

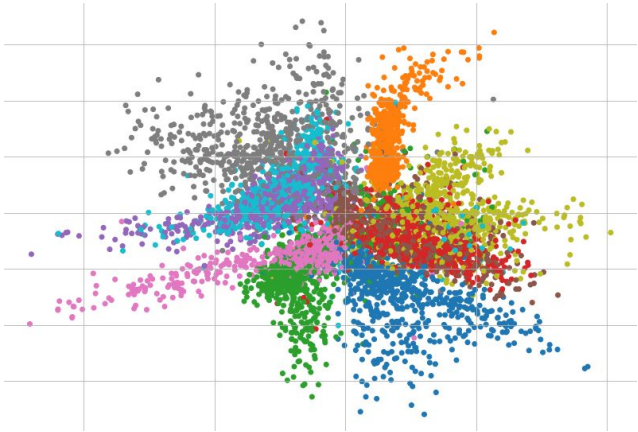
How data is represented is imperative for neural networks

The data representation could have been **engineered** by a human

As an alternative: Feed raw data to the neural network and **learn** the representation automatically

Manifold Hypothesis: High-dimensional data “lives” in a lower dimensional manifold

High-dimensional data can be sufficiently represented using lower dimensional latents



AutoEncoder

How data is represented is imperative for neural networks

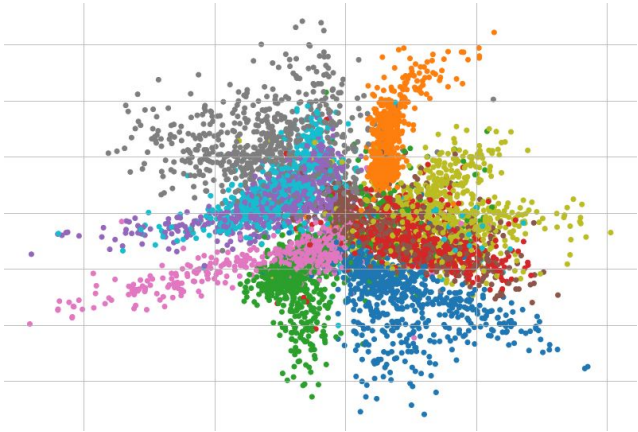
The data representation could have been **engineered** by a human

As an alternative: Feed raw data to the neural network and **learn** the representation automatically

Manifold Hypothesis: High-dimensional data “lives” in a lower dimensional manifold

High-dimensional data can be sufficiently represented using lower dimensional latents

lower dimensional manifold = lower dimensional Latent Space



AutoEncoder

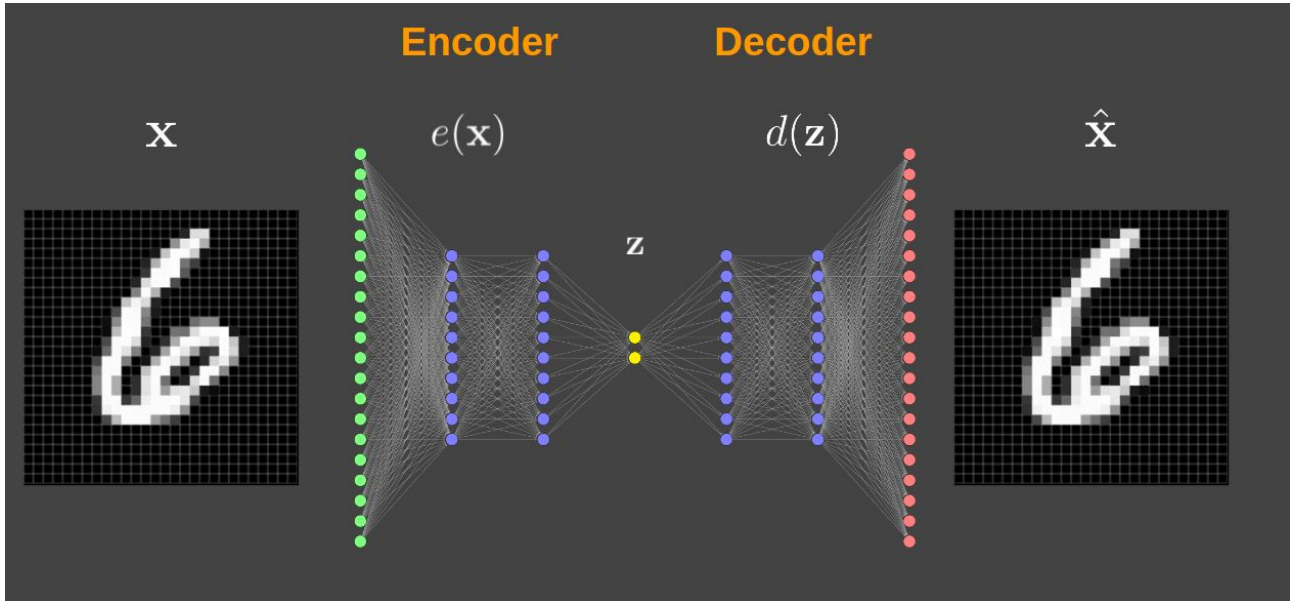
Encoder learns a mapping $e : X \to Z$

Decoder learns a mapping $d : Z \to X$

AutoEncoder

Encoder learns a mapping $e : X \rightarrow Z$

Decoder learns a mapping $d : Z \rightarrow X$



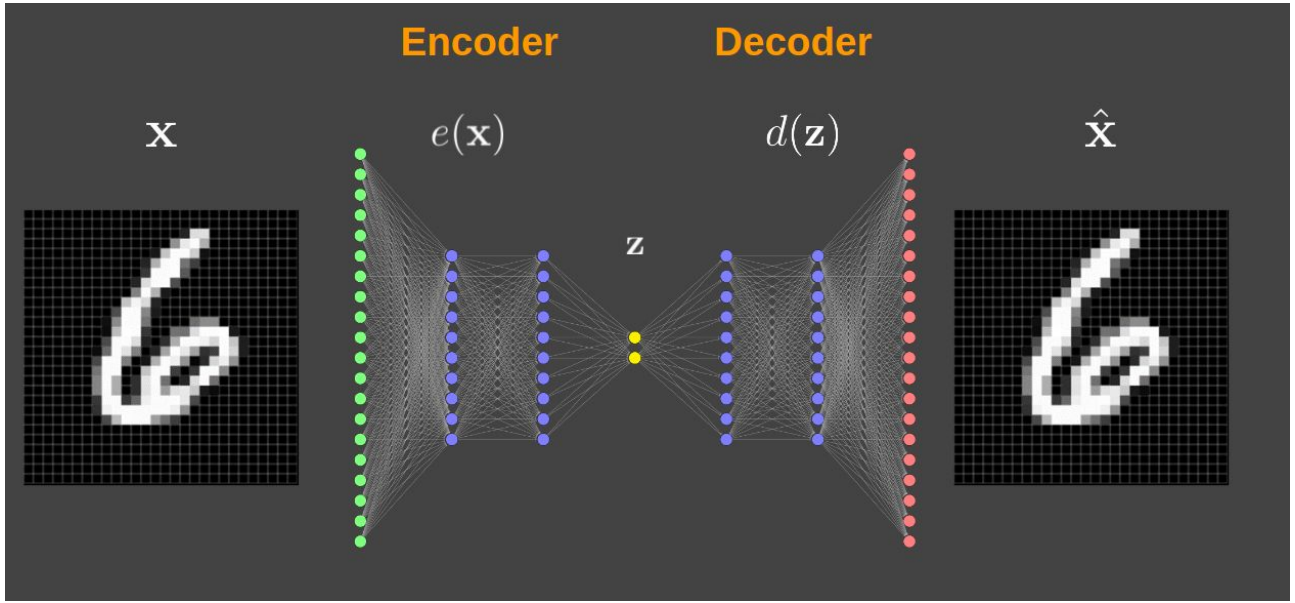
AutoEncoder

Encoder learns a mapping $e : X \rightarrow Z$

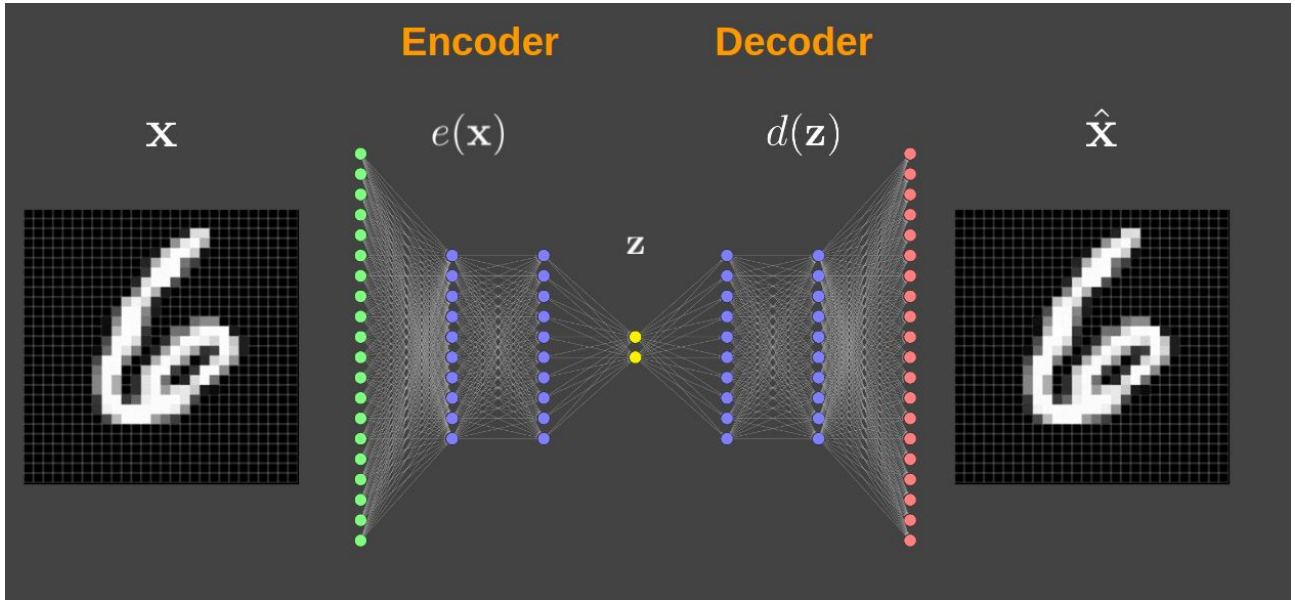
Decoder learns a mapping $d : Z \rightarrow X$

Nested functions

$$f(\mathbf{x}) = d(e(\mathbf{x}))$$



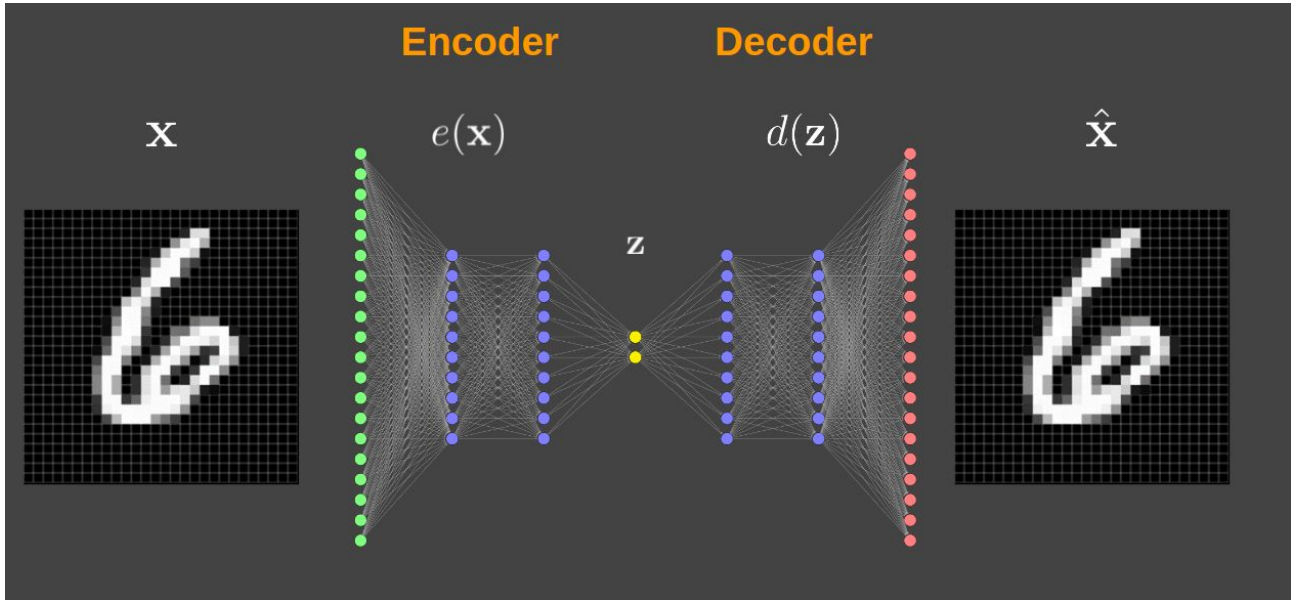
AutoEncoder



Classification

$$L(y, \hat{y}) = L(W, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

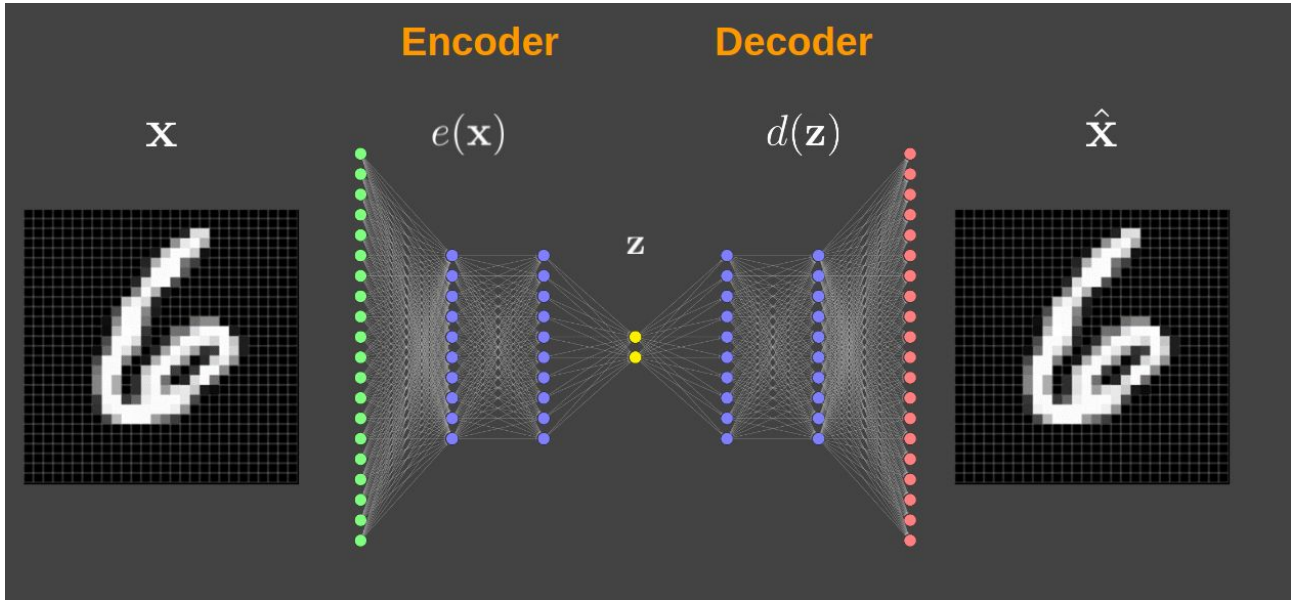
AutoEncoder



Classification

$$L(y, \hat{y}) = L(W, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

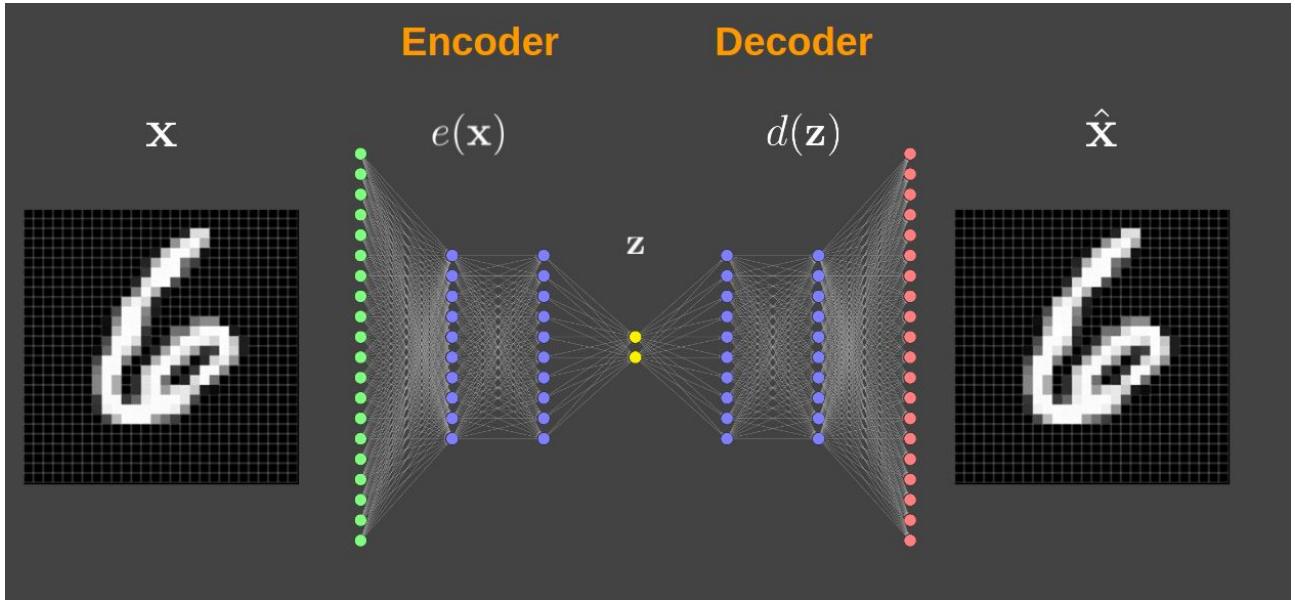
AutoEncoder



Classification

$$L(y, \hat{y}) = L(W, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

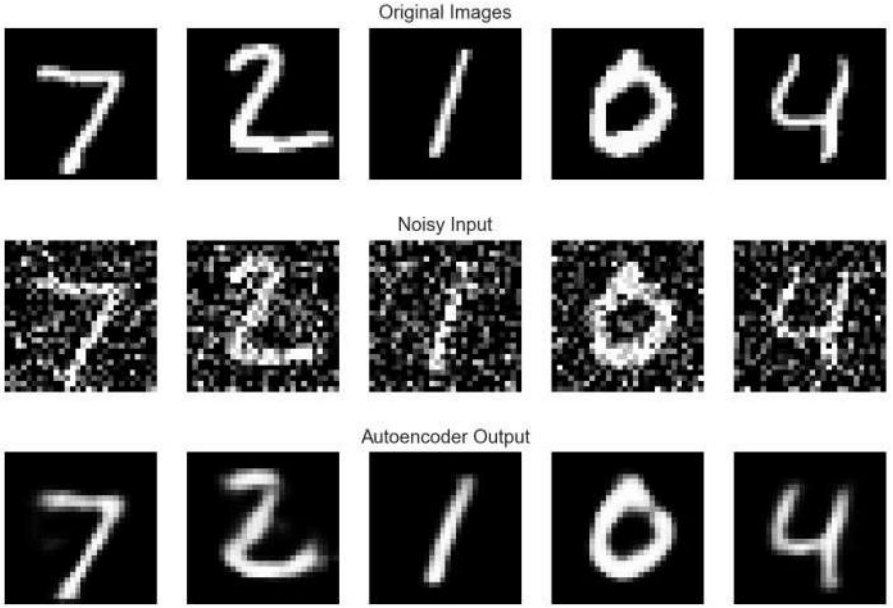
AutoEncoder



Reconstruction Loss

$$L(\mathbf{x}, \hat{\mathbf{x}}) = L(W, b) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x} - \hat{\mathbf{x}})^2$$

AutoEncoder



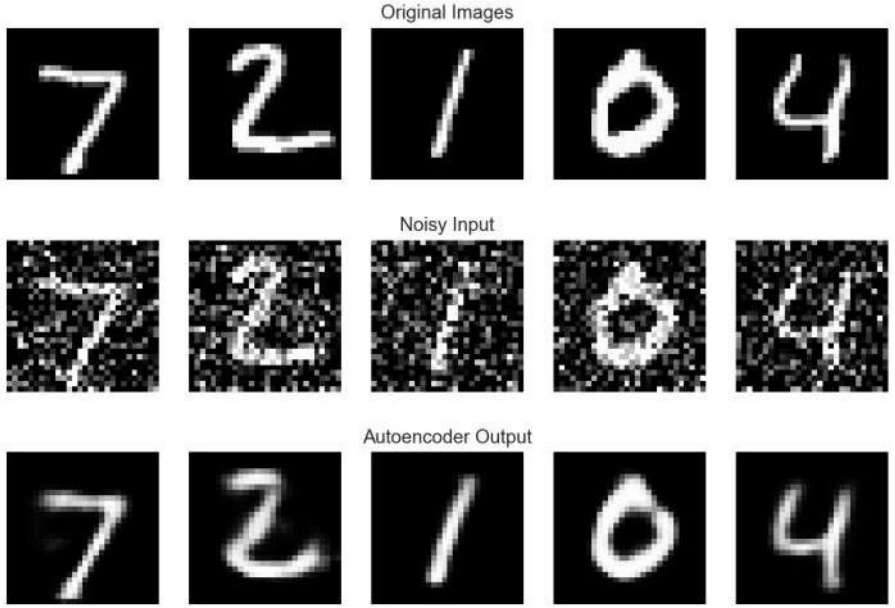
AutoEncoder

IMAGE COLORING

Before After

IMAGE NOISE REDUCTION

Before After



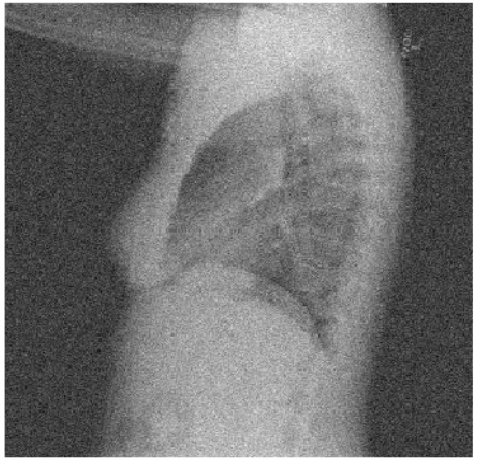
AutoEncoder

IMAGE COLORING

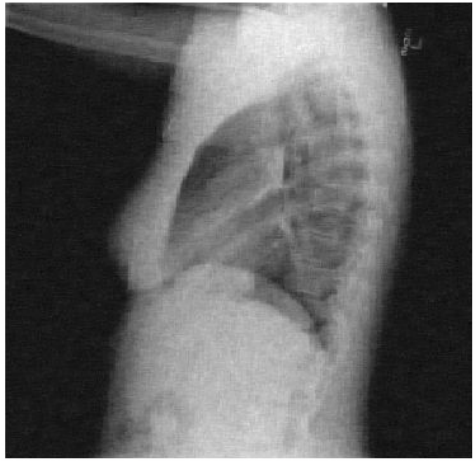
Before After

IMAGE NOISE REDUCTION

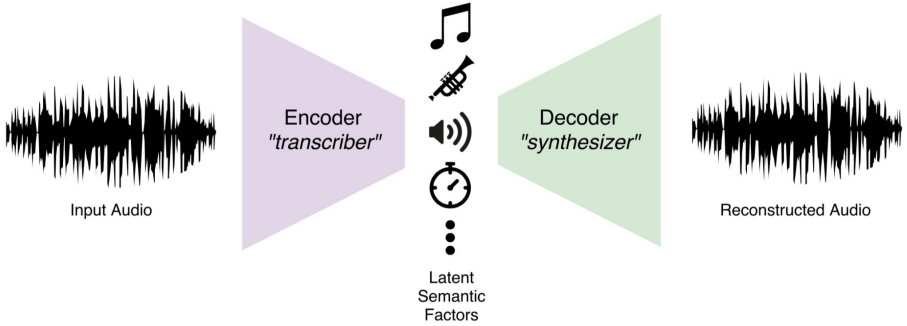
Before After



Noisy



Unsupervised Cleaning



AutoEncoder

Issues

Limited use aside from denoising and simple transformations

We cannot generate new data from the learned representations

Suppose we learn a two dimensional representation for MNIST

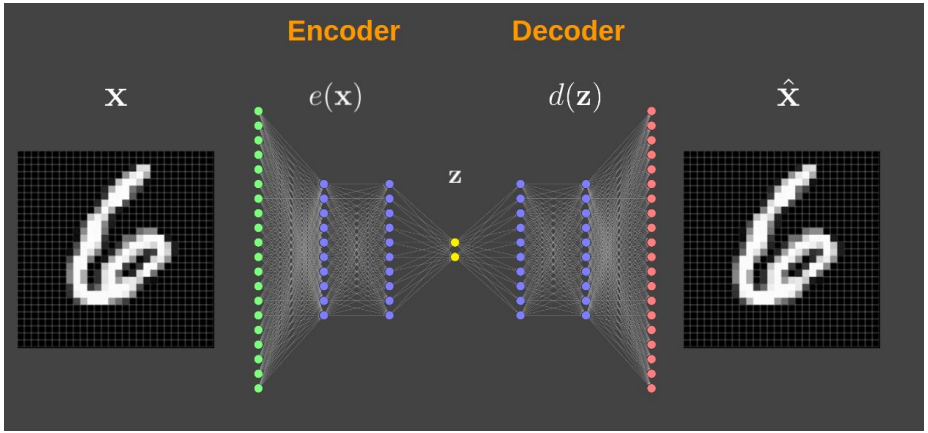
AutoEncoder

Issues

Limited use aside from denoising and simple transformations

We cannot generate new data from the learned representations

Suppose we learn a two dimensional representation for MNIST



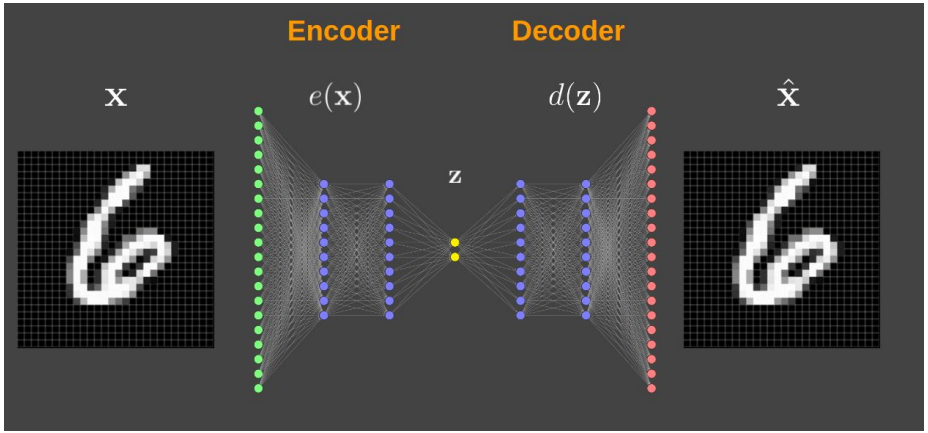
AutoEncoder

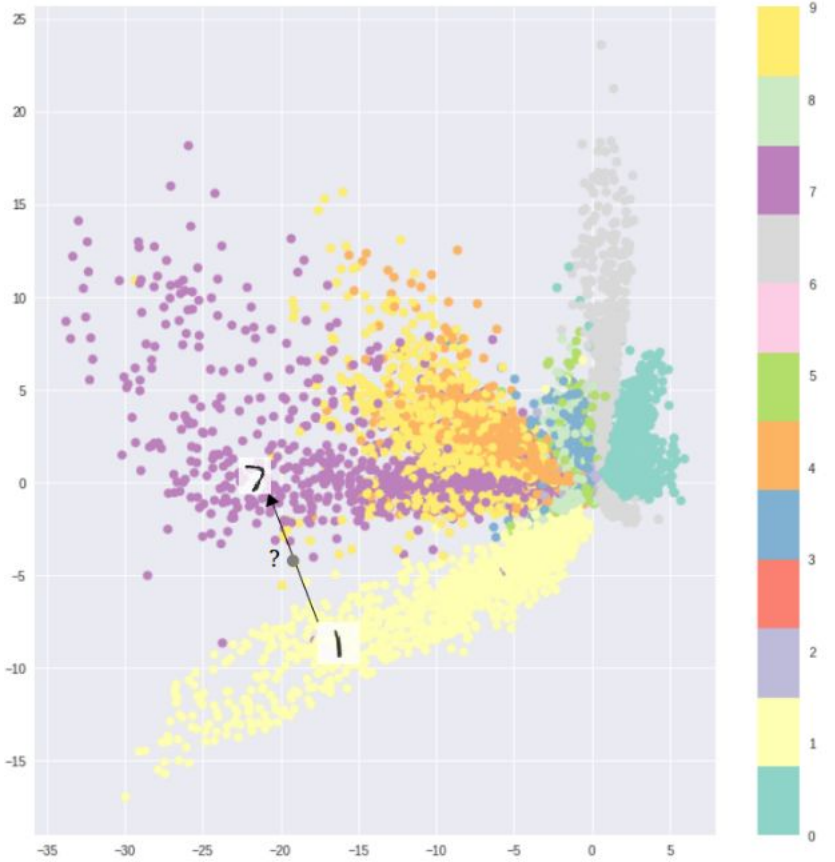
Issues

Limited use aside from denoising and simple transformations

We cannot generate new data from the learned representations

Suppose we learn a two dimensional representation for MNIST





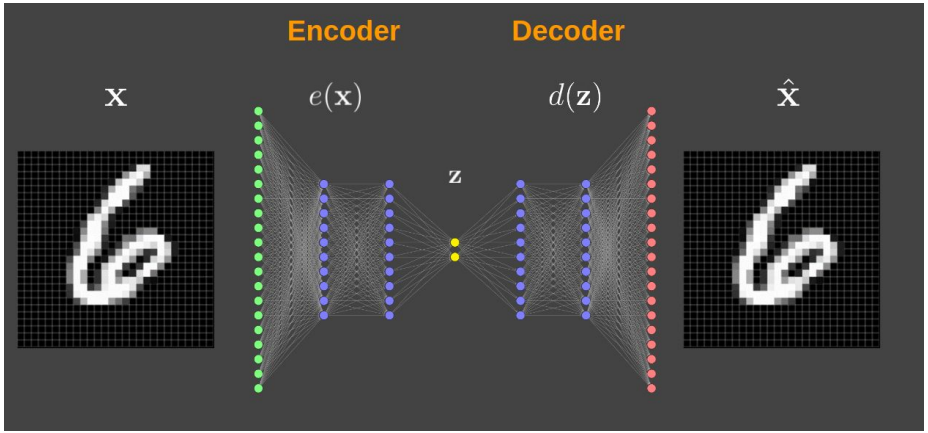
AutoEncoder

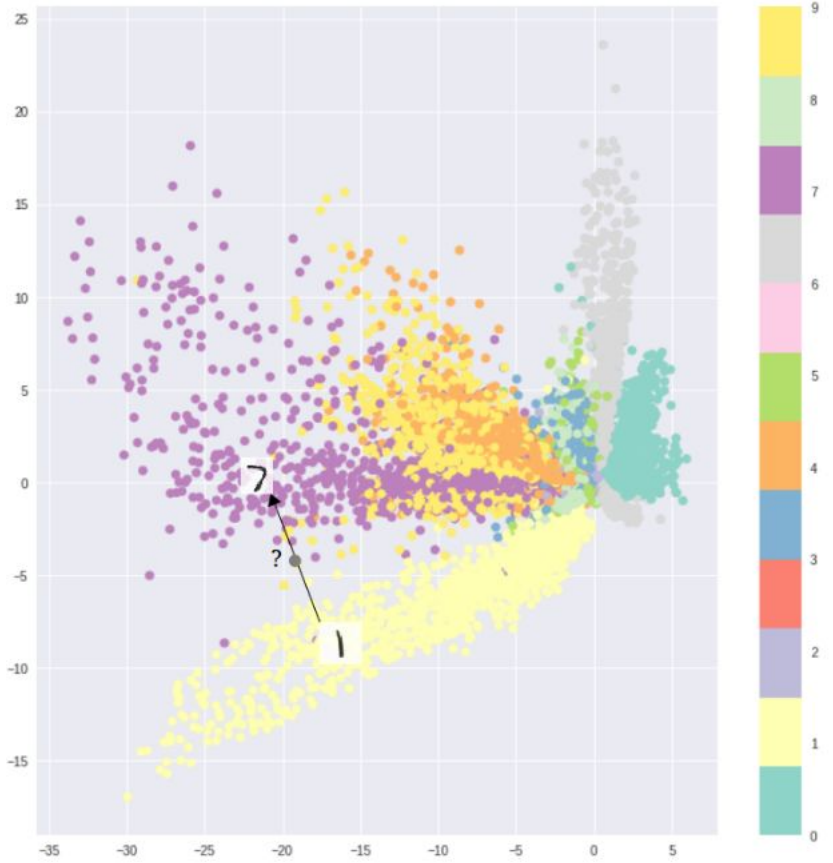
Issues

Latent space has ‘gaps’

We would like to generate variations of the data from a gapless/continuous latent space

Keep disentanglement of the features that represent the data





Variational AutoEncoder

Same structure as a normal AutoEncoder

We want stochasticity in our model

Map inputs to a probability distribution over our latent space

Learn the statistics of our latent space

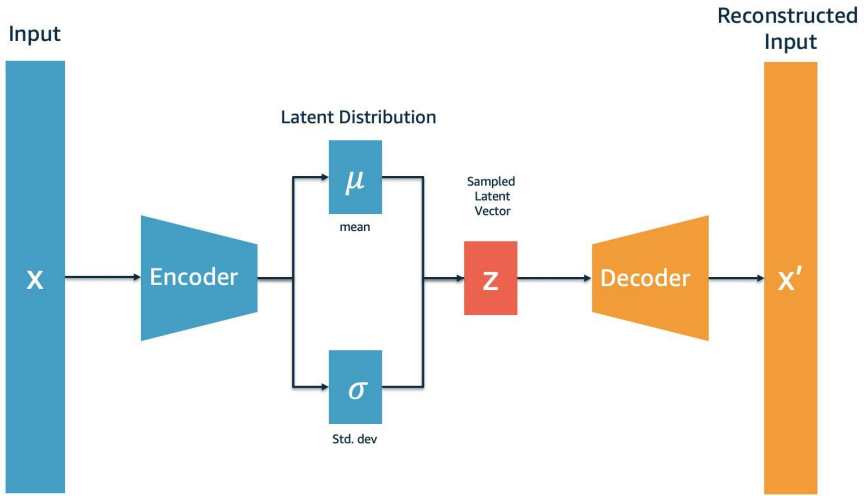
Variational AutoEncoder

Same structure as a normal AutoEncoder

We want stochasticity in our model

Map inputs to a probability distribution over our latent space

Learn the statistics of our latent space



Variational AutoEncoder

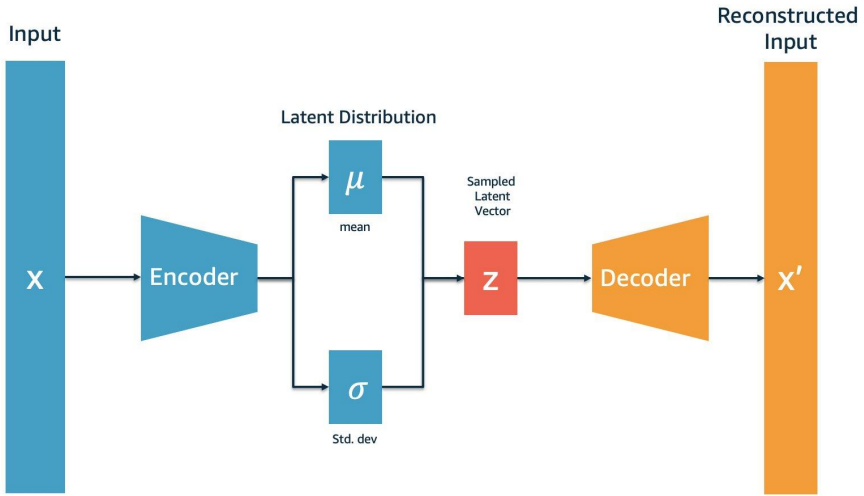
Same structure as a normal AutoEncoder

We want stochasticity in our model

Map inputs to a probability distribution over our latent space

Learn the statistics of our latent space

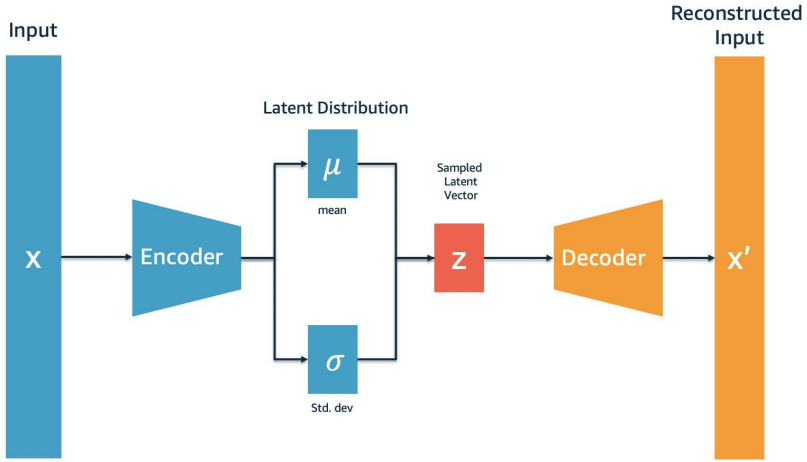
$$\mathbf{z} \sim \mathcal{N}(\mu, \sigma)$$



Variational AutoEncoder

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

Model is stochastic now! We want determinism back!



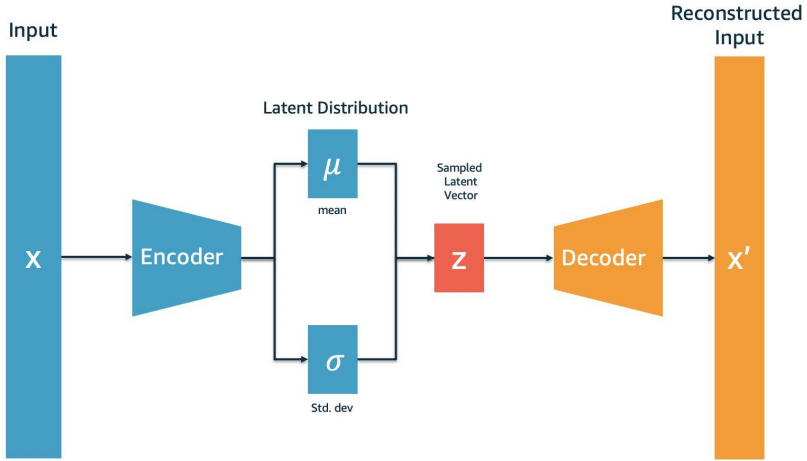
Variational AutoEncoder

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

Model is stochastic now! We want determinism back!

Cannot compute gradients through a stochastic layer

Derivative should be non-zero



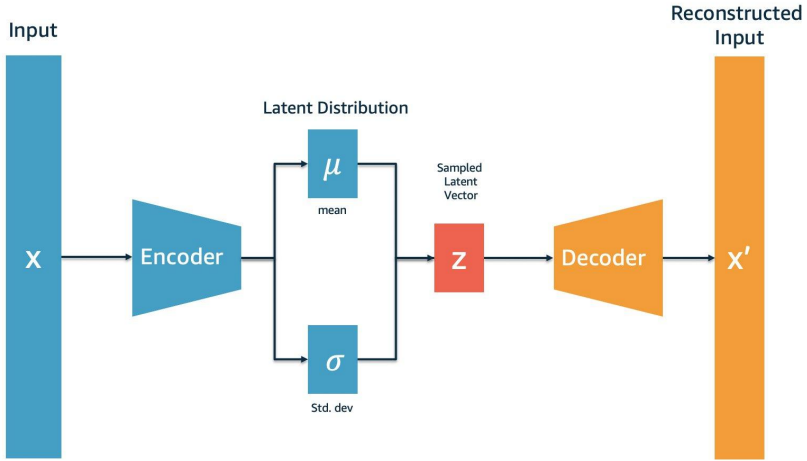
Variational AutoEncoder

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

Model is stochastic now! We want determinism back!

Cannot compute gradients through a stochastic layer

Derivative should be non-zero



Reparameterization ‘hack’

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

Backpropagation possible

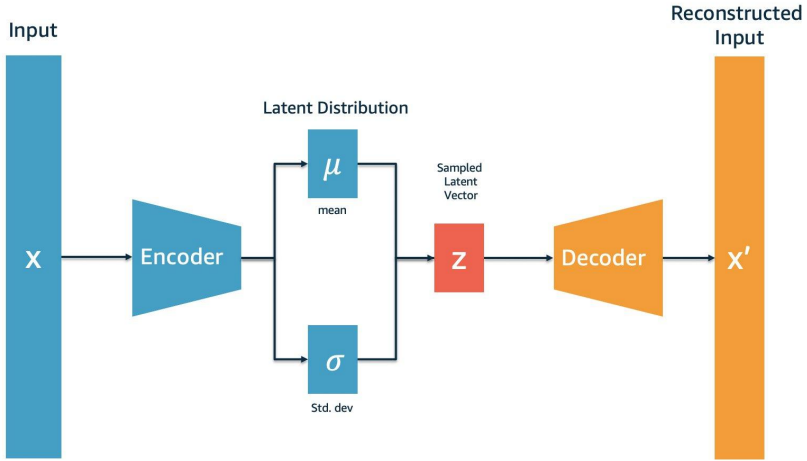
Variational AutoEncoder

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

Model is stochastic now! We want determinism back!

Cannot compute gradients through a stochastic layer

Derivative should be non-zero



Reparameterization ‘hack’

$$\mathbf{z} = \mu + \sigma \cdot \epsilon \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Backpropagation possible

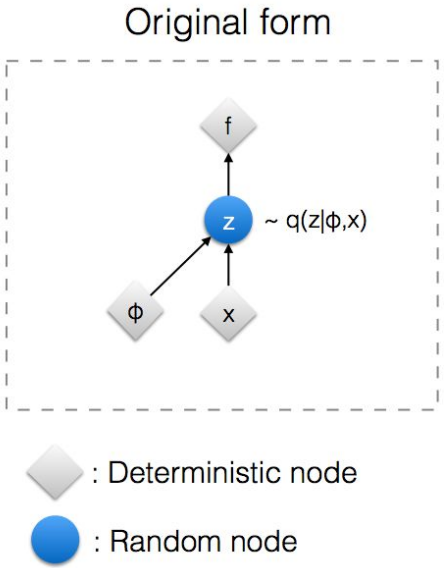
Variational AutoEncoder

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

Model is stochastic now! We want determinism back!

Cannot compute gradients through a stochastic layer

Derivative should be non-zero



Reparameterization ‘hack’

$$\mathbf{z} = \mu + \sigma \cdot \epsilon \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Backpropagation possible

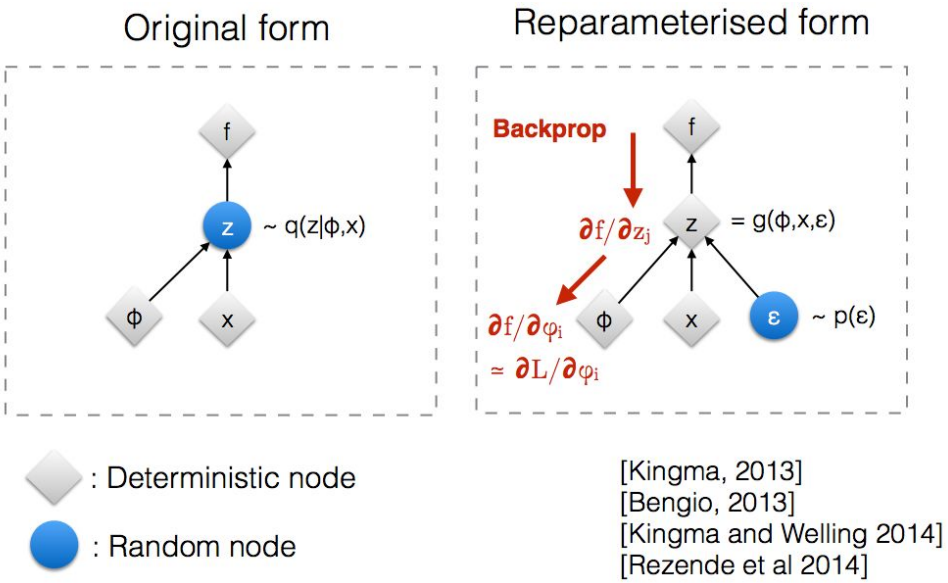
Variational AutoEncoder

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

Model is stochastic now! We want determinism back!

Cannot compute gradients through a stochastic layer

Derivative should be non-zero



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

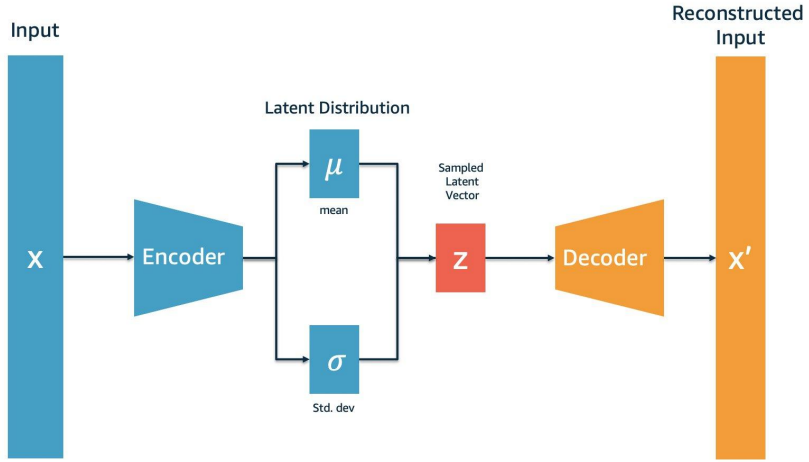
Reparameterization 'hack'

$$\mathbf{z} = \mu + \sigma \cdot \epsilon \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Backpropagation possible

Variational AutoEncoder

$$\mathbf{z} = \mu + \sigma \cdot \epsilon \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$



Reconstruction loss

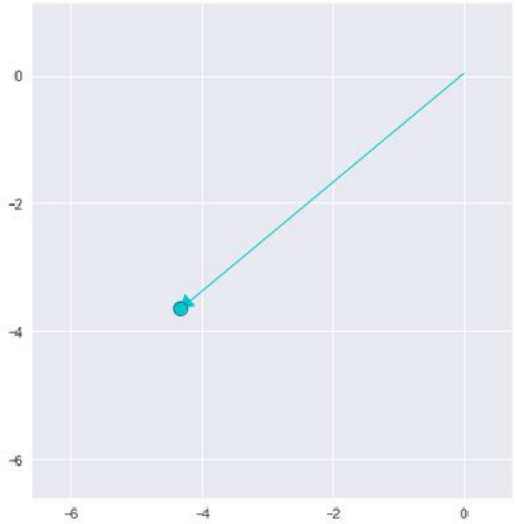
$$L(\mathbf{x}, \hat{\mathbf{x}}) = L(W, b) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x} - \hat{\mathbf{x}})^2$$

Not done yet; need to force the network to adhere to to a unit normal distribution

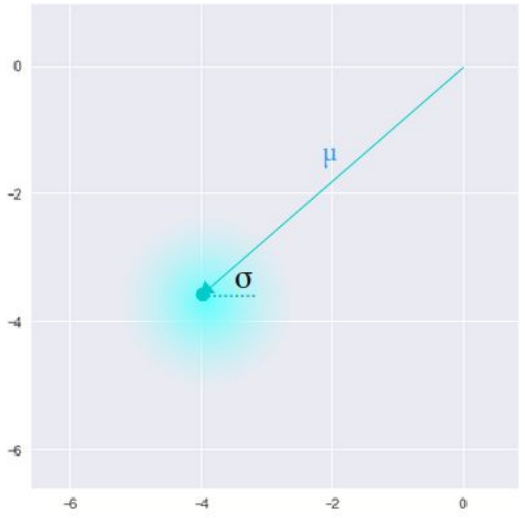
Network will just learn very small standard deviations and learn the mean, similar to a regular autoencoder

Variational AutoEncoder

Network will just learn very small standard deviations and learn the mean, similar to a regular autoencoder



Standard Autoencoder
(direct encoding coordinates)

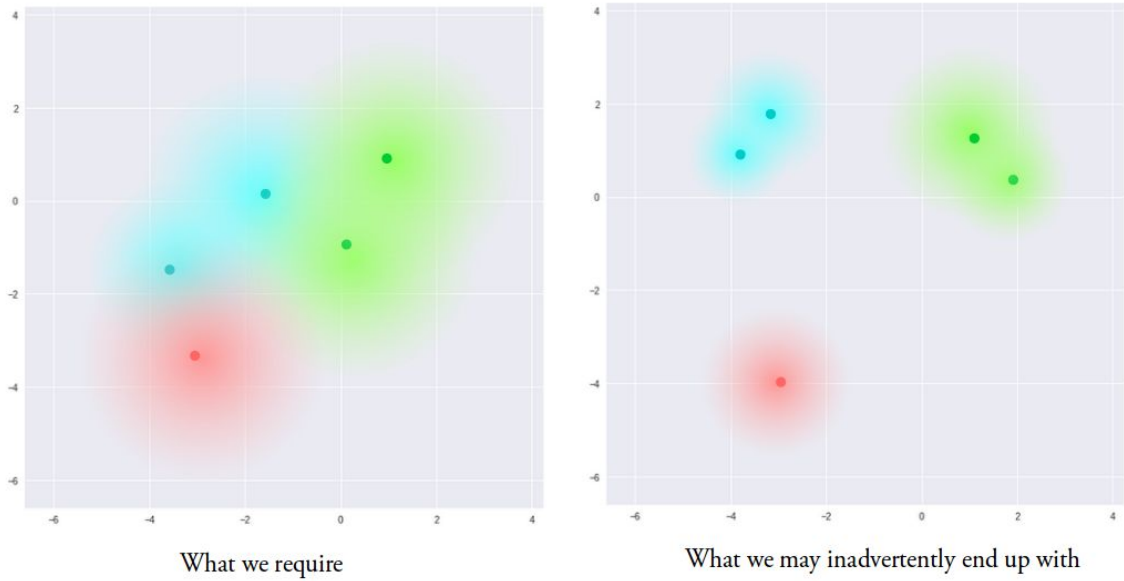


Variational Autoencoder
(μ and σ initialize a probability distribution)

https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Variational AutoEncoder

Network will just learn very small standard deviations and learn the mean, similar to a regular autoencoder



https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Use the Kullback-Leibler divergence from information theory

$$D_{KL}(q||p) = \sum_{x} q(x)\ \log\ \frac{q(x)}{p(x)}$$

Measures the information content between distributions q and p

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Use the Kullback-Leibler divergence from information theory

$$D_{KL}(q||p) = \sum_{x} q(x)\ \log\ \frac{q(x)}{p(x)}$$

Measures the information content between distributions q and p

We want the (posterior) distribution $q_\theta(z|x)$ that we are learning

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Use the Kullback-Leibler divergence from information theory

$$D_{KL}(q||p) = \sum_{x} q(x) \log \frac{q(x)}{p(x)}$$

Measures the information content between distributions q and p

We want the (posterior) distribution $q_\theta(z|x)$ that we are learning to be close to a (prior) standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Use the Kullback-Leibler divergence from information theory

$$D_{KL}(q||p) = \sum_{x} q(x)\ \log\ \frac{q(x)}{p(x)}$$

Measures the information content between distributions q and p

We want the (posterior) distribution $q_\theta(z|x)$ that we are learning to be close to a (prior) standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Why standard normal though?

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Why standard normal though?

1. Learn a smooth/continuous distribution
2. Expand evenly in the latent space
3. KL divergence with a normal distribution has an easy analytical solution
4. We can estimate any distribution with a normal distribution

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - D_{KL}(q_\theta(z|x)||\mathcal{N}(0,1))$$

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - D_{KL}(q_\theta(z|x)||\mathcal{N}(0,1))$$

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - \frac{1}{2}\sum 1 + \sigma_i^2 - \mu_i^2 - \log(\sigma_i)$$

Likelihood **KLD**

Variational AutoEncoder

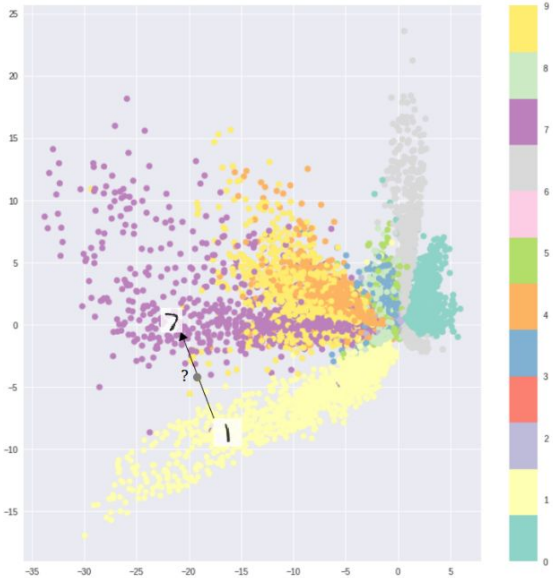
We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - \frac{1}{2}\sum_{i=1} 1 + \sigma_i^2 - \mu_i^2 - \log(\sigma_i)$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - \frac{1}{2}\sum_{i=1} 1 + \sigma_i^2 - \mu_i^2 - \log(\sigma_i)$

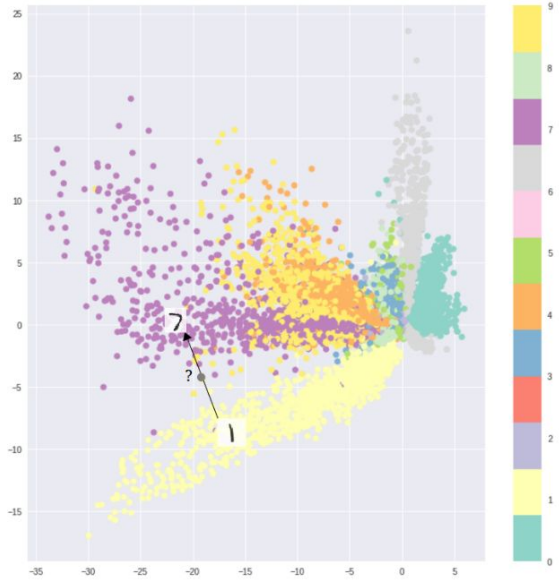
Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

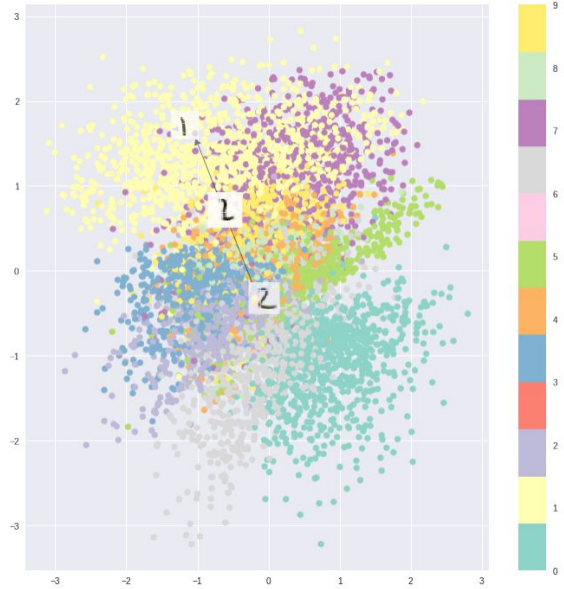


Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - \frac{1}{2}\sum_{i=1} 1 + \sigma_i^2 - \mu_i^2 - \log(\sigma_i)$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$

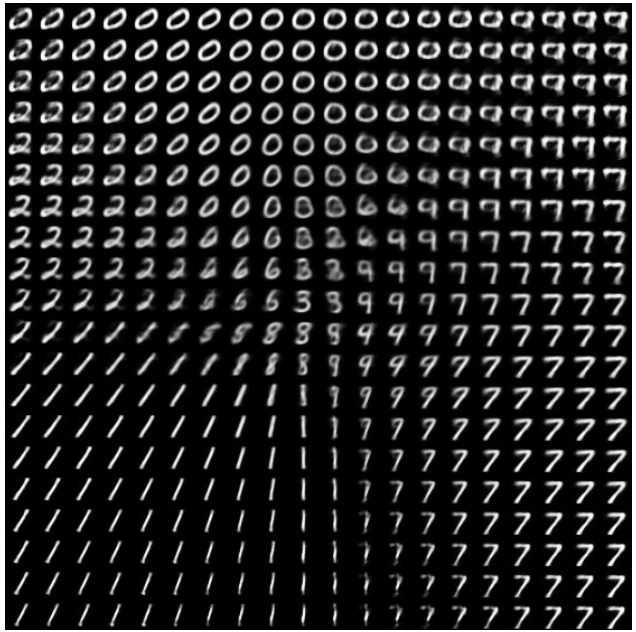


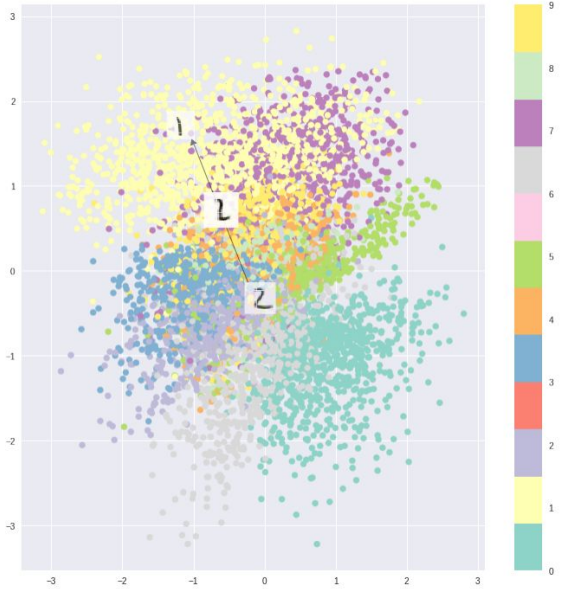


Variational AutoEncoder

We add a regularization term to the reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x} - \hat{\mathbf{x}})^2 - \frac{1}{2}\sum_{i=1} 1 + \sigma_i^2 - \mu_i^2 - \log(\sigma_i)$

Not only reconstruct but also stay close to $\mathcal{N}(\mathbf{0}, \mathbf{1})$





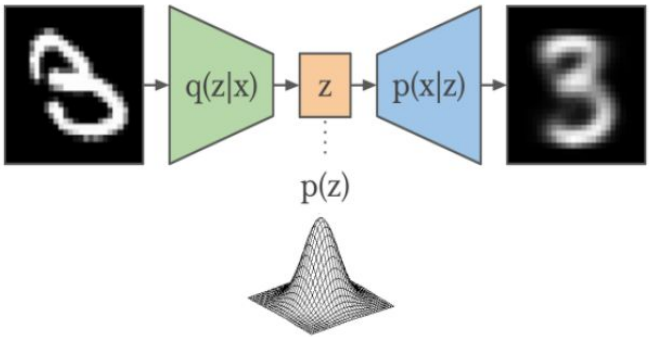
Variational Bayesian AutoEncoder

Evidence lower bound (ELBO)

$$\ln p(x) \geq \mathbb{E}_{q(z|x)}[\ln p(x|z)] - D_{KL}[q(z|x)||p(z)]$$

Proposed by Kingma & Welling (2013) and Rezende et al. (2014)

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



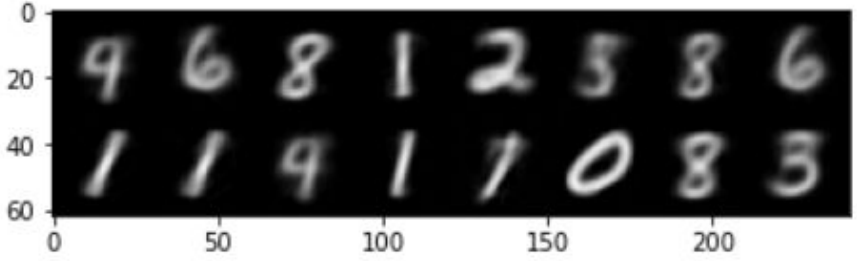
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

More on this:

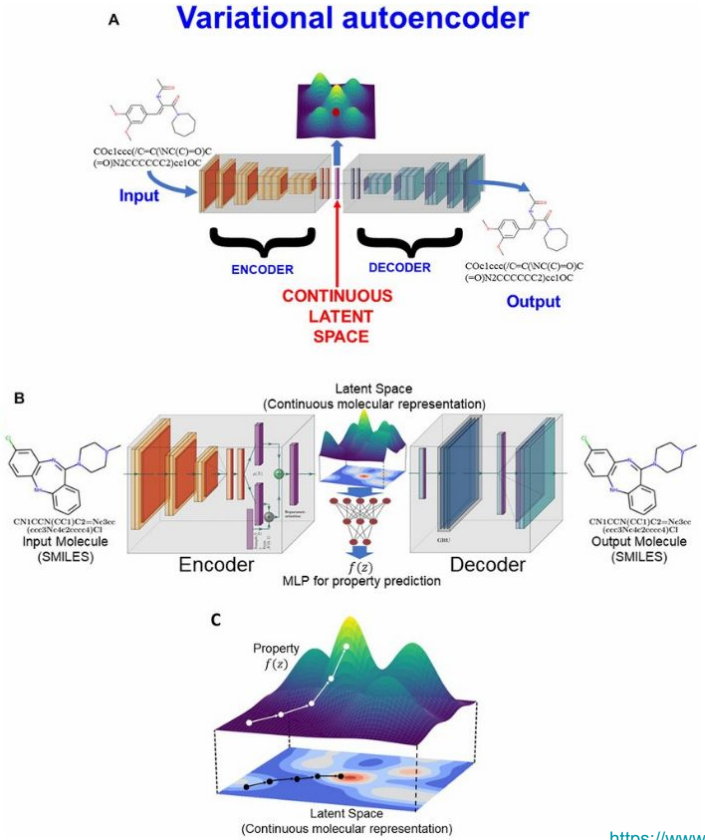
https://arxiv.org/pdf/1906.02691.pdf

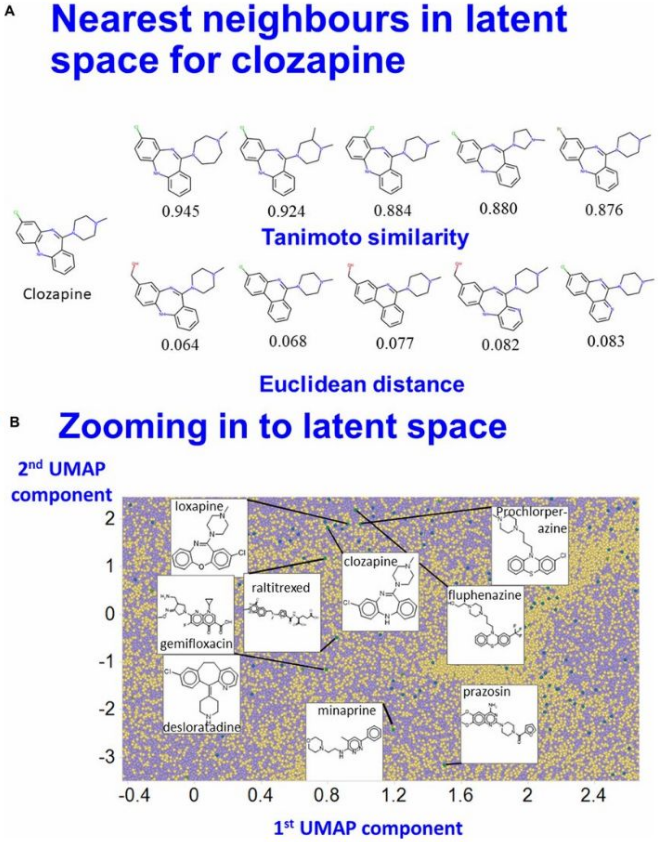
https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf

VAE applications



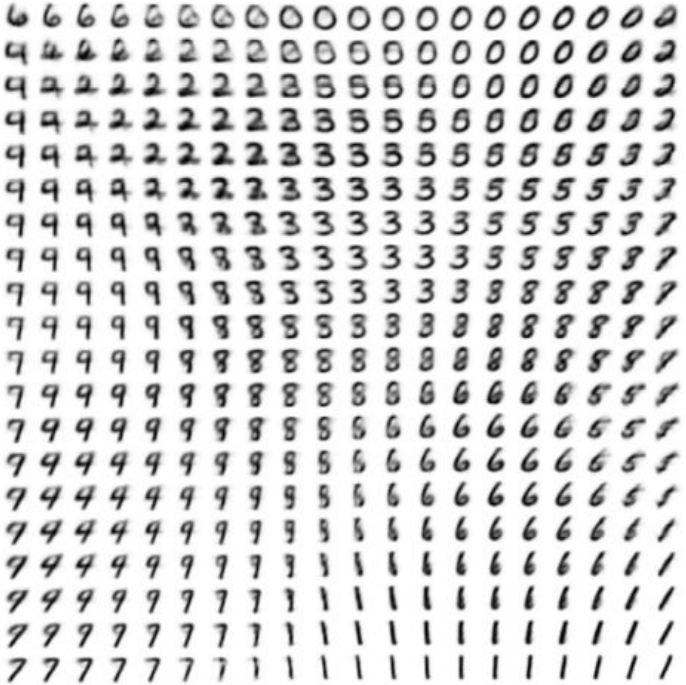
VAE applications

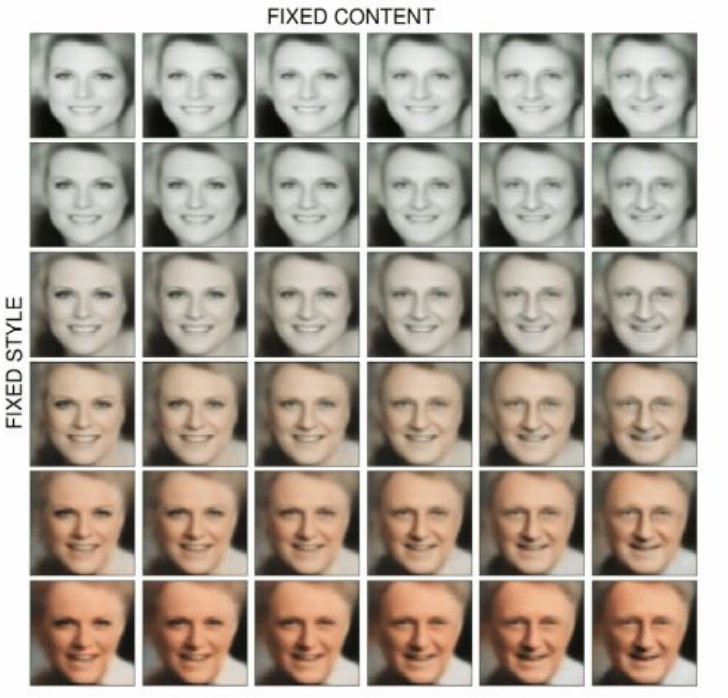




https://www.researchgate.net/figure/Local-properties-of-latent-space-A-Molecules-near-clozapine-A-VAE-was-trained-as-in_fig5_347531998

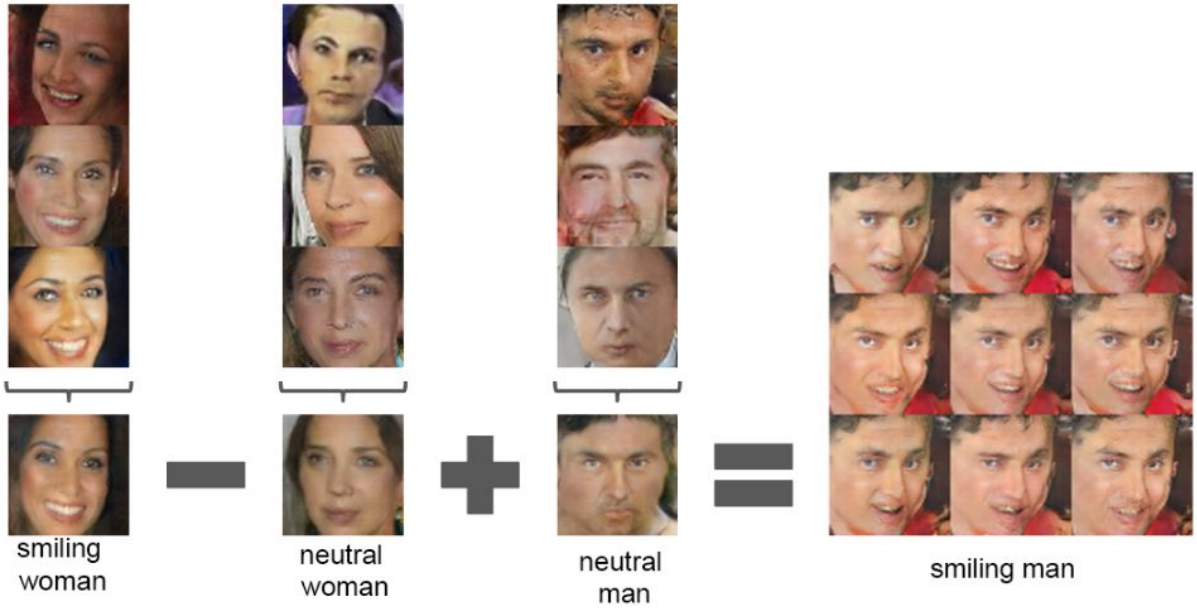
Latent Vector Interpolation





Arithmetic

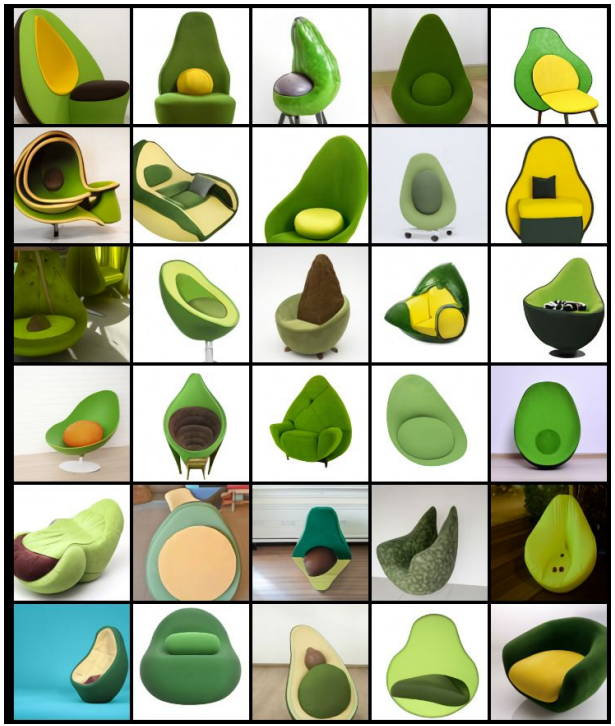
Latent Space Arithmetic

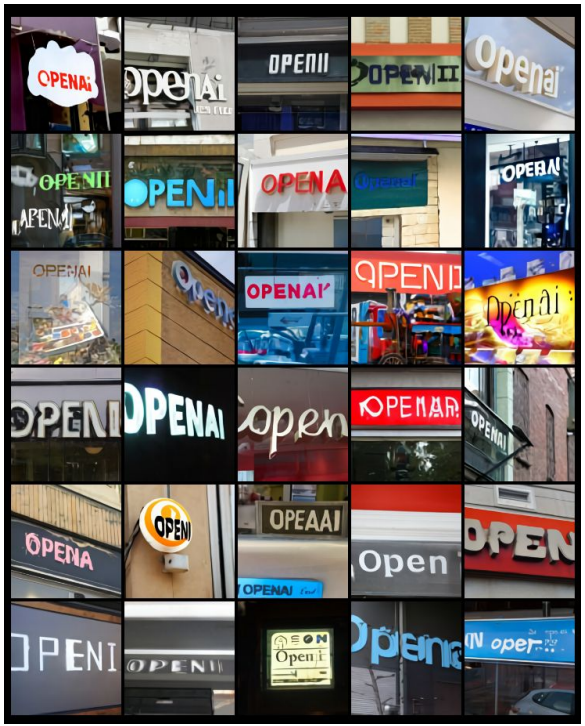


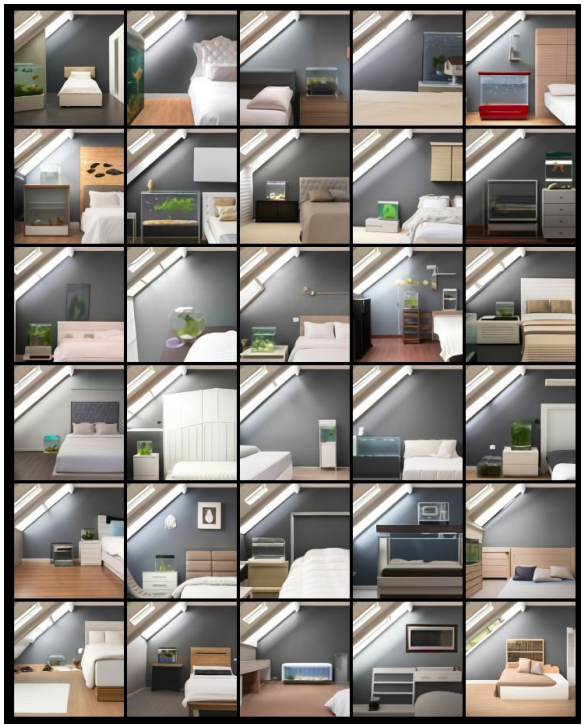
Mirza & Osindero (2014, arXiv:1411.1784); Isola et al. (2016, arXiv:1611.07004)

Dall-E

Original Dall-e by OpenAI used a discrete VAE, where the latent space is discrete and not continuous



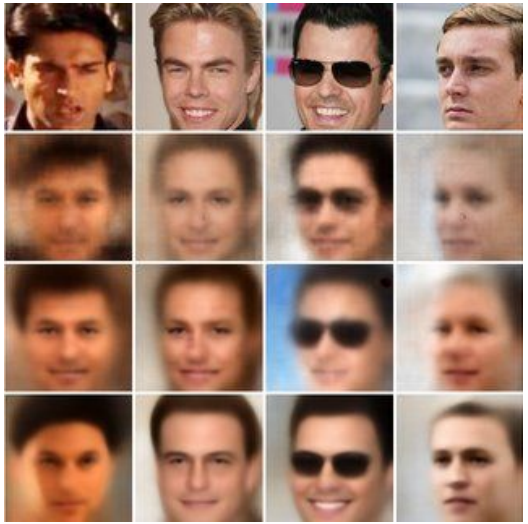




Problems with VAEs

Posterior Collapse: The posterior (encoder) is too weak and the decoder starts ignoring samples from the posterior. The encoder becomes “disconnected” from the mean and standard deviation it needs to learn

Blurry Generations: The assumption of a standard normal distribution makes the generations ‘noisy’. Using a MSE loss as reconstruction increases the problem



Families of Generative Networks

Variational Autoencoders

Discrete VAEs

GAN (Generative Adversarial Network)

Diffusion Models

Autoregressive Models

Dall-E 2

(DALL·E 2) "An astronaut surfing on a sea turtle over a rainbow past planets in space, digital art"

Image Synthesis

an ai in the shape of a campfire telling stories to an audience of enthralled forest animals in hyperrealistic digital fantasy style

"a raccoon astronaut with the cosmos reflecting on the glass of his helmet dreaming of the stars"

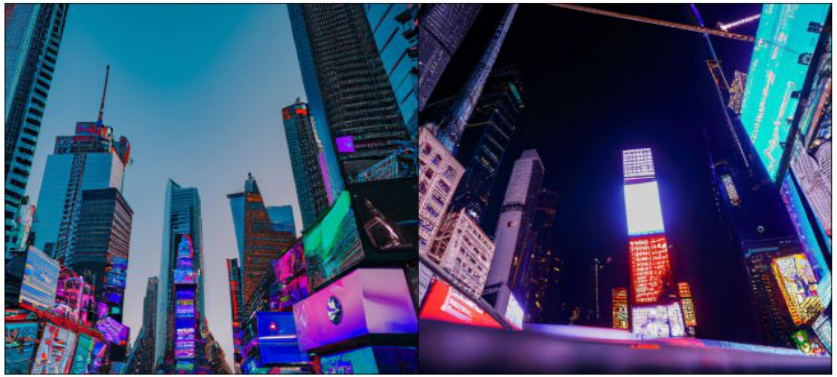
@OpenAI DALL-E 2

Dall-E 2

Figure 19: Random samples from unCLIP for prompt “A teddybear on a skateboard in Times Square.”

Dall-E 2

(a) A high quality photo of a dog playing in a green field next to a lake.

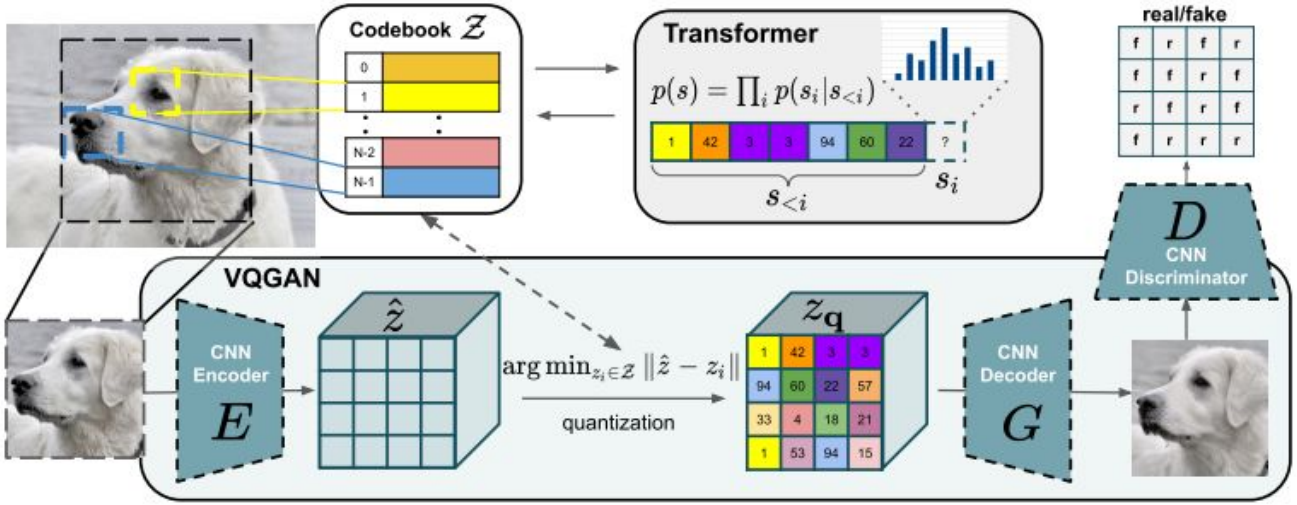


(b) A high quality photo of Times Square.

panda mad scientist mixing sparkling chemicals, artstation

a corgi's head depicted as an explosion of a nebula

Quality of GAN-Generated images



https://arxiv.org/pdf/2012.09841.pdf

Quality of GAN-Generated images

Quality of GAN-Generated images

Super Resolution

Change Style: Young to Old

https://yuval-alaluf.github.io/SAM/

Modelling Sequences

How to work with data that has a temporal dimension?

Modelling Sequences

Language translation

Speech recognition

Music generation

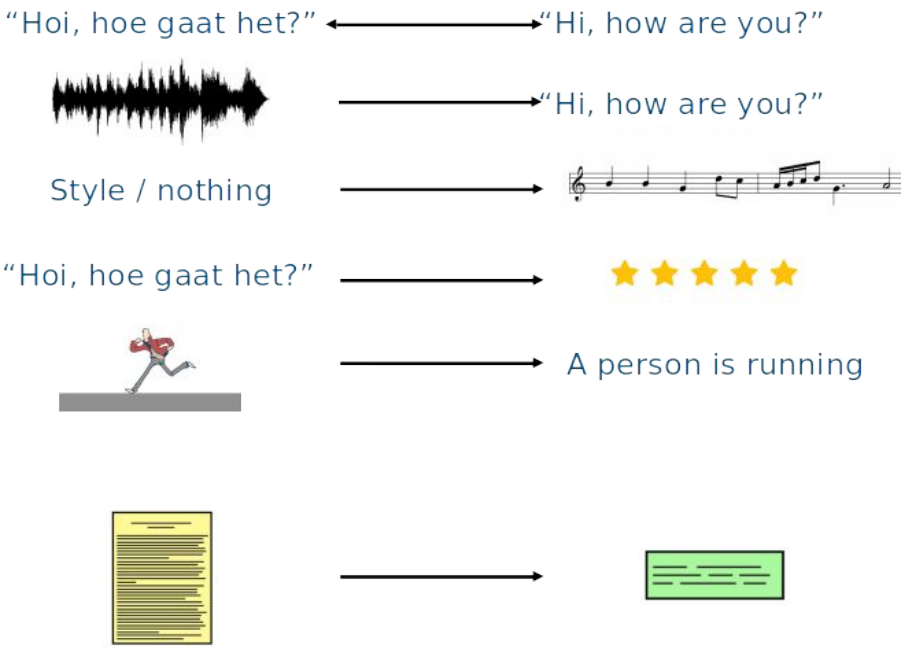
Sentiment classification

Video captioning

Chatbot

Text summary

...



Sequential Data

Sequential Data are represented with 1d or 2d arrays

Time	24/3	25/3	26/3	27/3	28/3	...
Value	19,722	21,050	21,188	22,532	21,678	...

Sequential Data

Sequential Data are represented with 1d or 2d arrays

Time	24/3	25/3	26/3	27/3	28/3	...
Value	19,722	21,050	21,188	22,532	21,678	...

“The cat is black.”

Value	42	82	16	4	0	...

Sequential Data

Sequential Data are represented with 1d or 2d arrays

Time	24/3	25/3	26/3	27/3	28/3	...
Value	19,722	21,050	21,188	22,532	21,678	...

“The cat is black.”

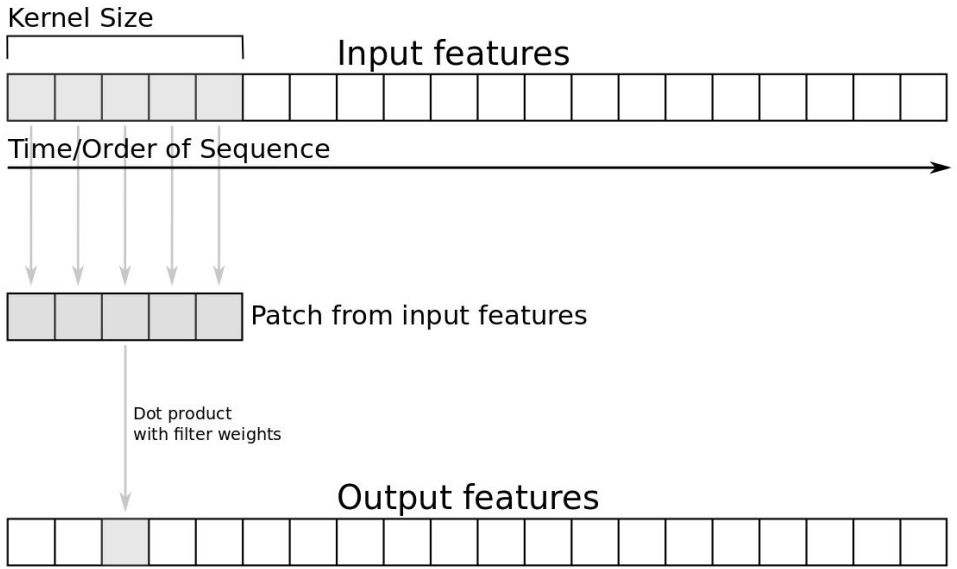
Value	42	82	16	4	0	...

In Natural Language Processing (NLP)

- Words converted into tokens
- Token is represented numerically
- Sentence is a sequence of tokens
- Tokens may be logically related to each other

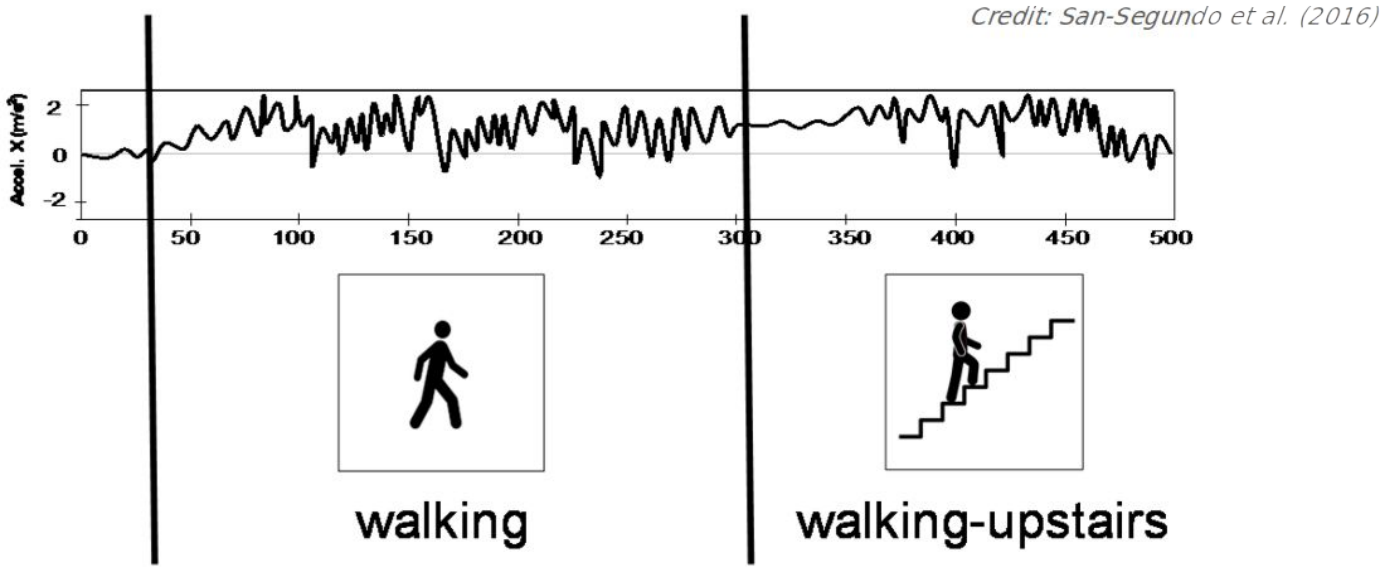
Sequence modelling with 1D convolutions

- CNNs work here because they take into account **nearby** tokens



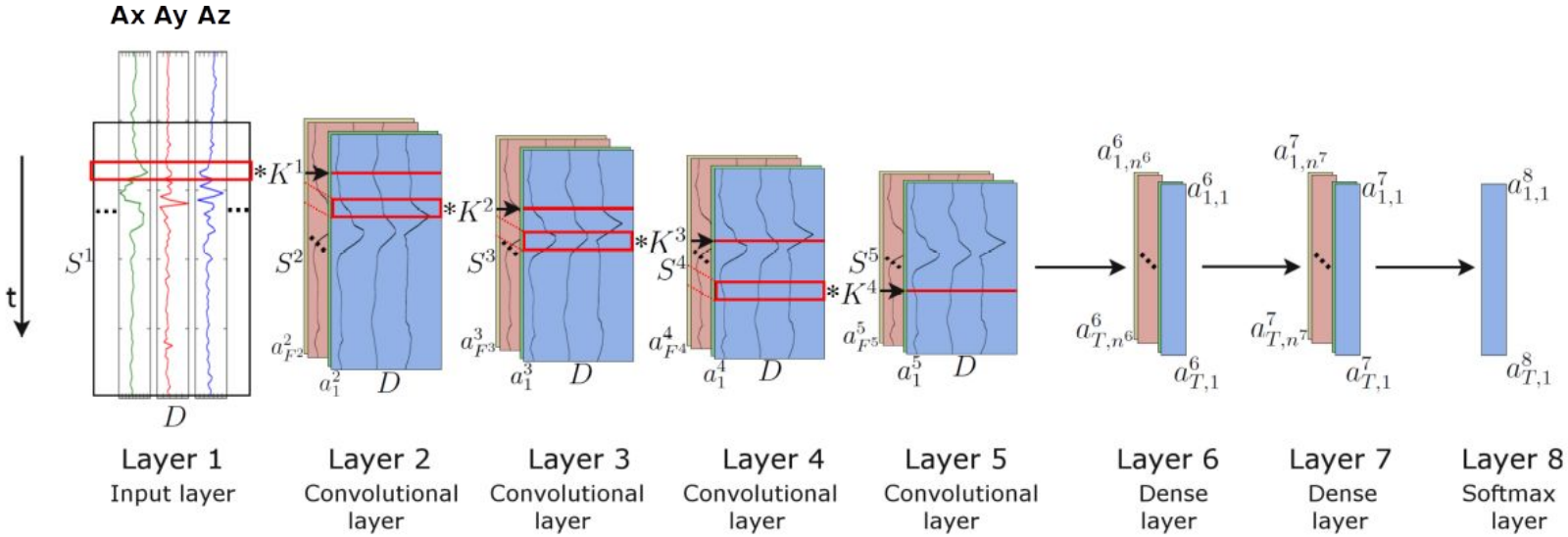
Human Activity Modelling with CNNs

- CNNs work here because they take into account **nearby** tokens

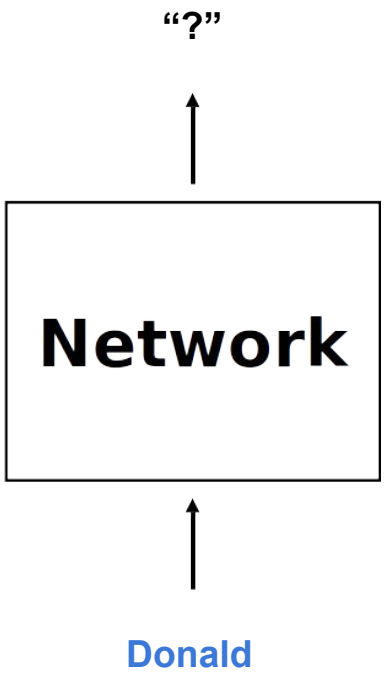


The DNN can be trained with **existing** open datasets!
e.g., https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

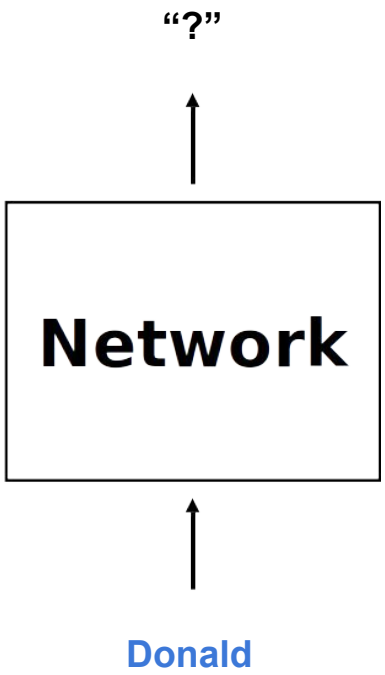
Human Activity Modelling with CNNs



Natural Language Processing



Natural Language Processing



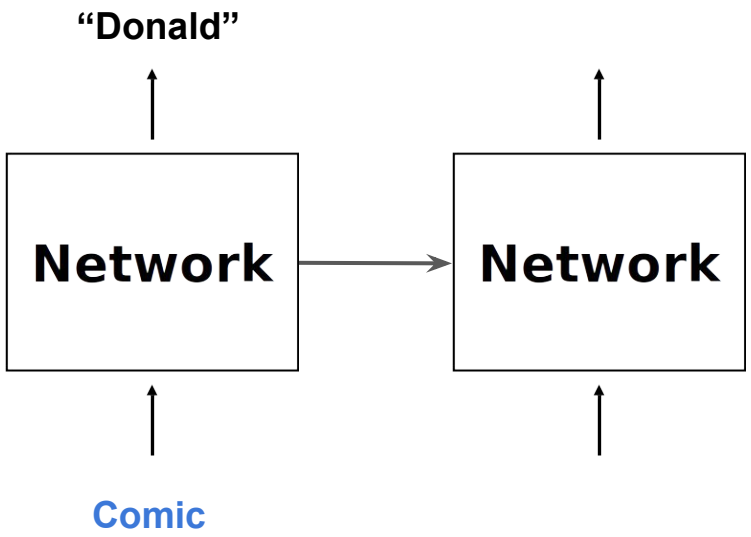
How about modeling just the next word?

The president of the United States is Donald Trump.
Wherever Donald Trump goes, security is tight.

The comic Donald Duck is very famous.

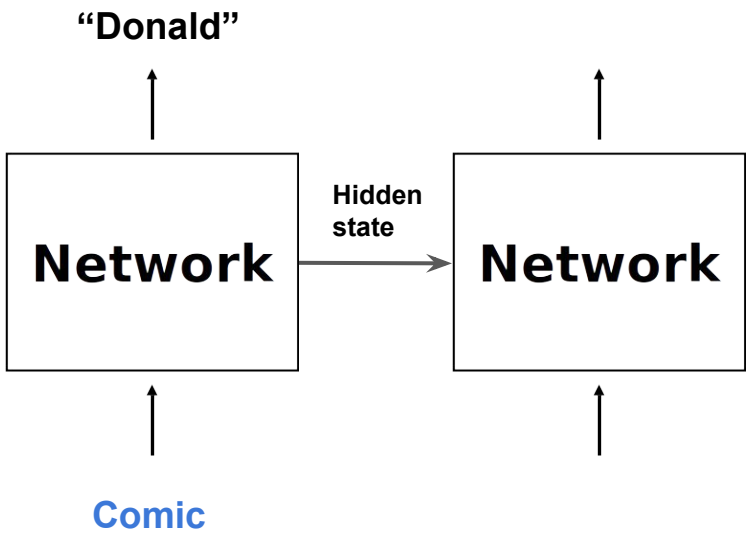
Context matters to the prediction!

Natural Language Processing



The meaning of the sentence depends on the **context**

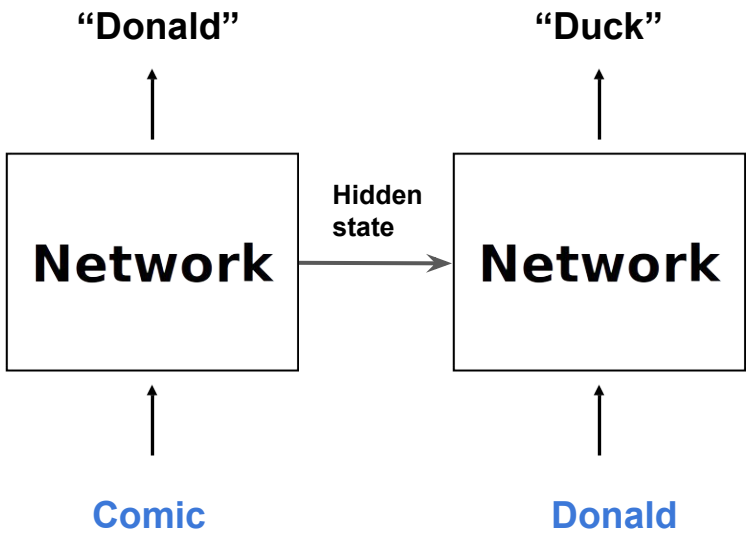
Natural Language Processing



The meaning of the sentence depends on the **context**

The context is propagated to the next time step via the **hidden state.**

Natural Language Processing

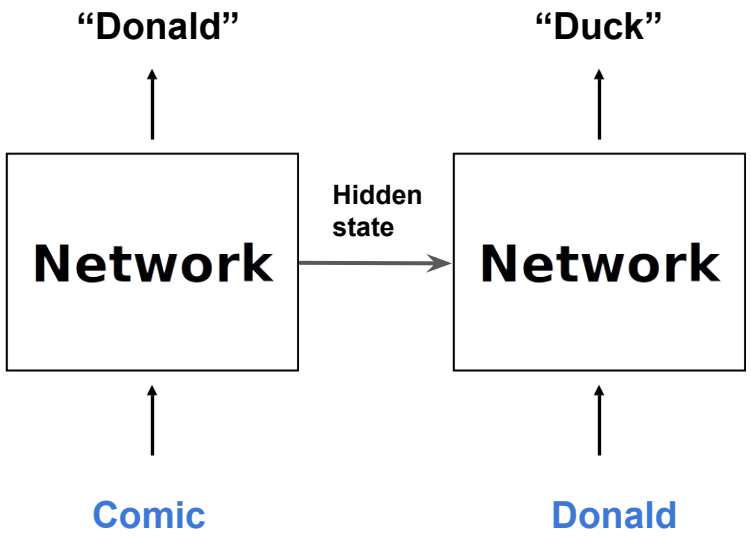


The meaning of the sentence depends on the **context**

The context is propagated to the next time step via the **hidden state.**

The next prediction is then modelled with both the hidden state and the next word

Natural Language Processing

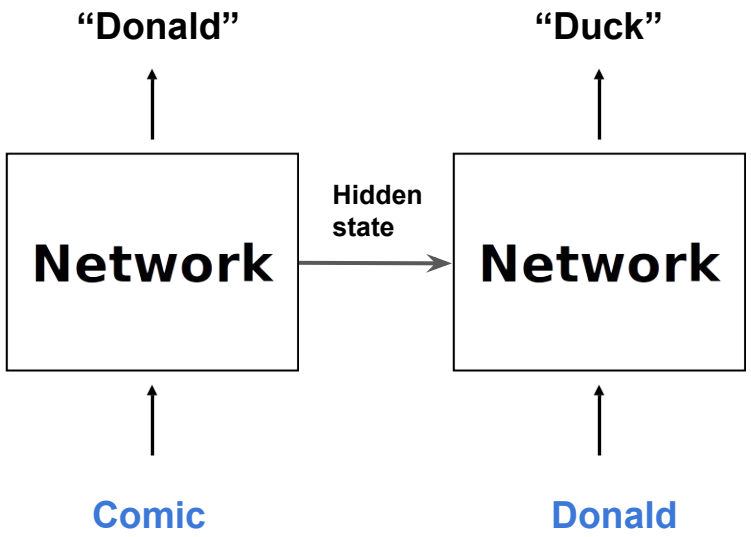


The meaning of the sentence depends on the **context**

The context is propagated to the next time step via the **hidden state.**

The next prediction is then modelled with both the hidden state and the next word

Natural Language Processing



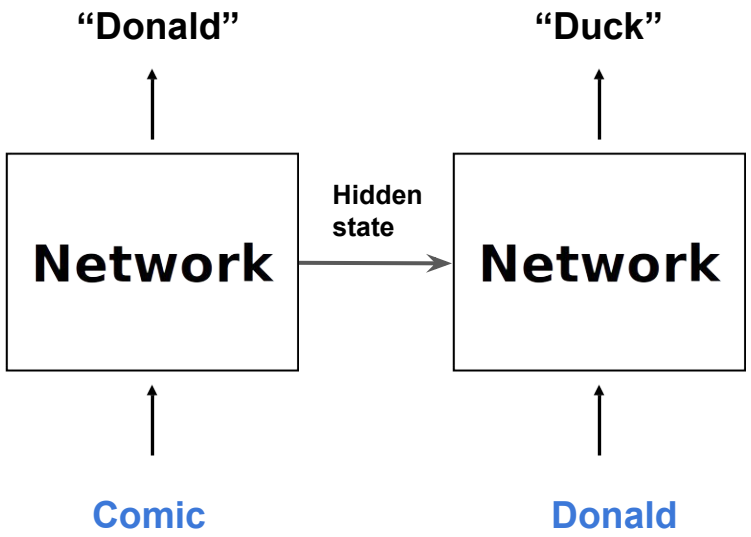
The meaning of the sentence depends on the **context**

The context is propagated to the next time step via the **hidden state.**

The next prediction is then modelled with both the hidden state and the next word

The hidden state allows the network to have **memory**

Natural Language Processing



The meaning of the sentence depends on the **context**

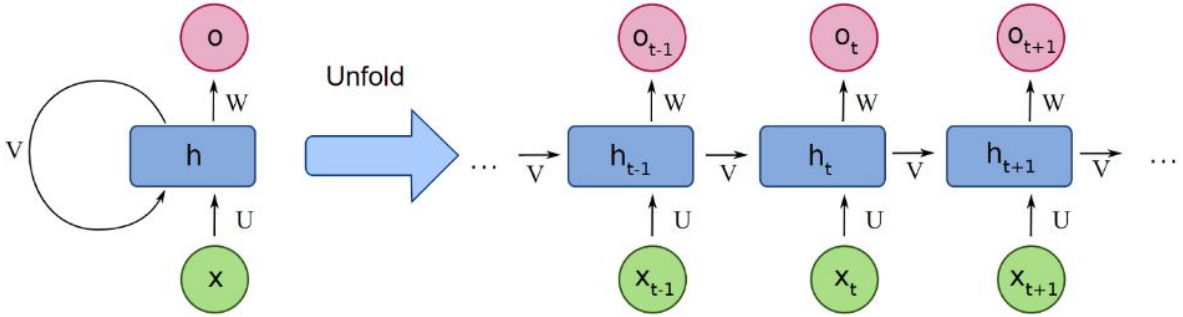
The context is propagated to the next time step via the **hidden state.**

The next prediction is then modelled with both the hidden state and the next word

The hidden state allows the network to have **memory**

Recurrent Neural Network

In RNNs, weights are reused across time steps



RNN

$$h_t = \sigma(U x_t + V h_{t-1} + b_h)$$

$$O_t = \sigma(W h_t + b_o)$$

Back propagation through **time**

Non-RNN $O_t = \sigma(W x_t + b)$

Back propagation through **layers**

Long-term Dependency

Sometimes, the contexts are far away

The clouds are in the ***sky***

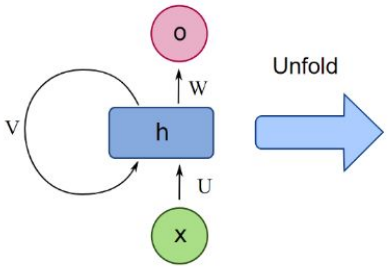
I grew up in The Netherlands … I speak fluent ***Dutch***

Long-term Dependency

Sometimes, the contexts are far away

The clouds are in the ***sky***

I grew up in The Netherlands … I speak fluent ***Dutch***

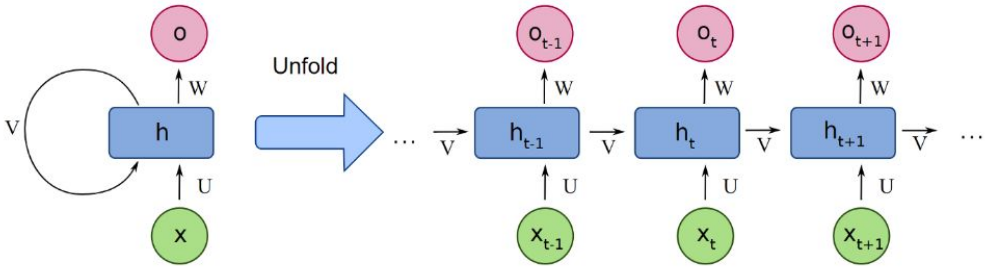


Long-term Dependency

Sometimes, the contexts are far away

The clouds are in the ***sky***

I grew up in The Netherlands … I speak fluent ***Dutch***

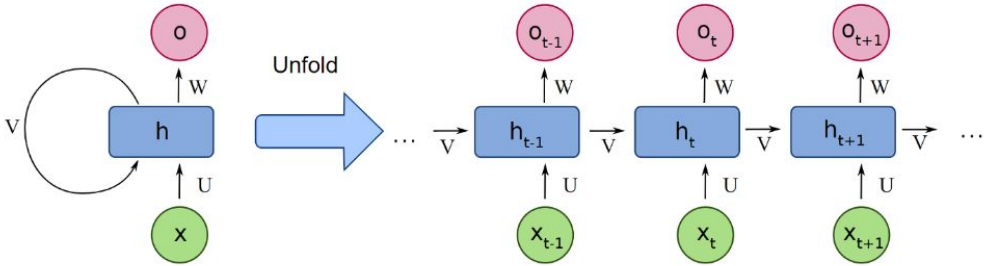


Long-term Dependency

Sometimes, the contexts are far away

The clouds are in the ***sky***

I grew up in The Netherlands … I speak fluent ***Dutch***



$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta) \tag{1}$$

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b} \tag{2}$$

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{3}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \tag{4}$$

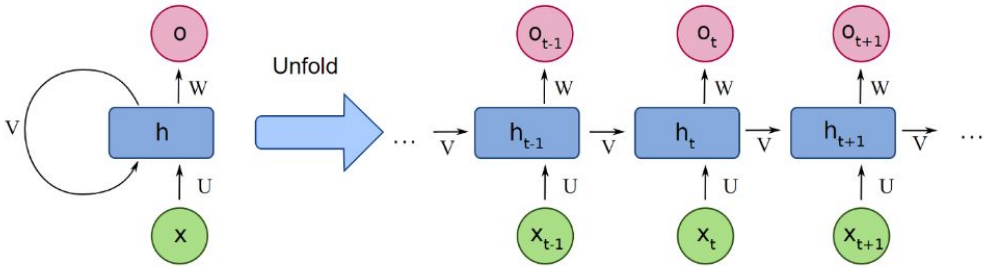
$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \tag{5}$$

Long-term Dependency

Sometimes, the contexts are far away

The clouds are in the ***sky***

I grew up in The Netherlands … I speak fluent ***Dutch***



$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta) \quad (1)$$

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b} \quad (2)$$

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

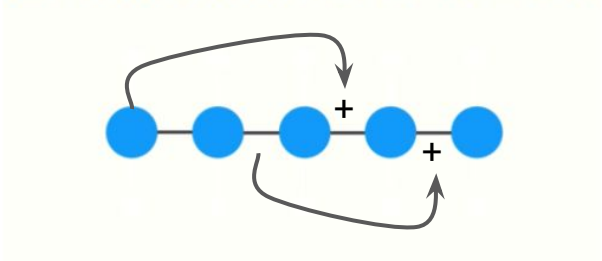
$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

If $\mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$ is < 1, in a long product, gradients start to vanish

Long Short-term Memory

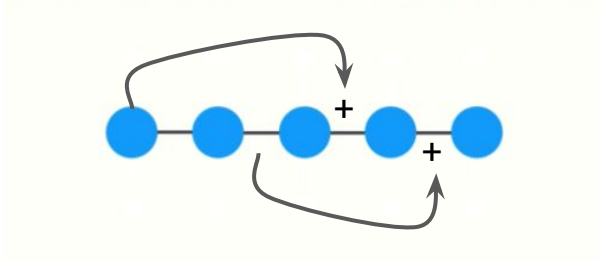
Residual connection is a general solution to the vanishing gradient problem.



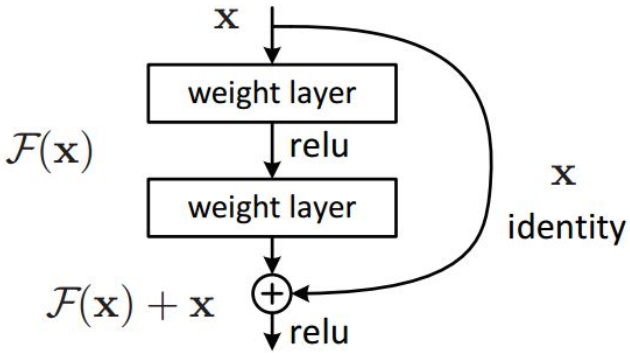
However, for very long dependency (which is common in RNNs), this solution is not sufficient.
We should instead regulate which information to **keep** and which to **forget**.

Long Short-term Memory

Residual connection is a general solution to the vanishing gradient problem.

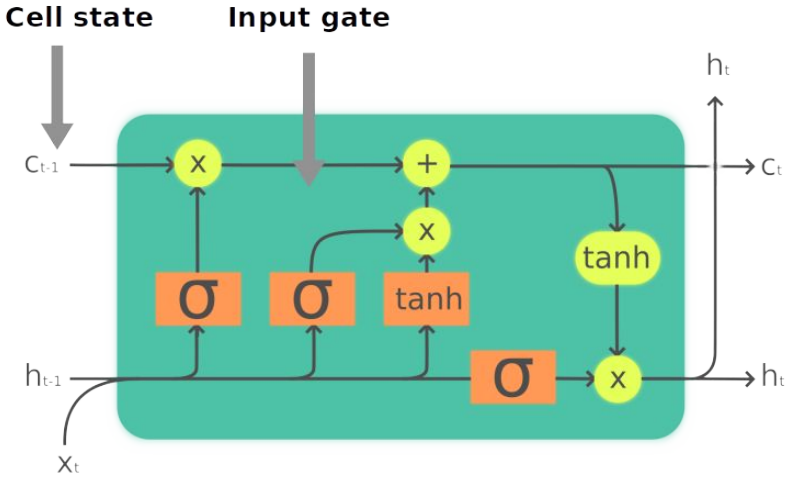


However, for very long dependency (which is common in RNNs), this solution is not sufficient.
We should instead regulate which information to **keep** and which to **forget**.



Long Short-term Memory

LSTM



LSTM regulates the flow of information through gates.

- **Input gate**: controls the extent to which a new value flows into the cell;
- **Forget gate**: controls the extent a value remains in the cell;
- **Output gate**: controls the extent to which the value in the cell is used to calculate the output activation.

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Sequence Modeling

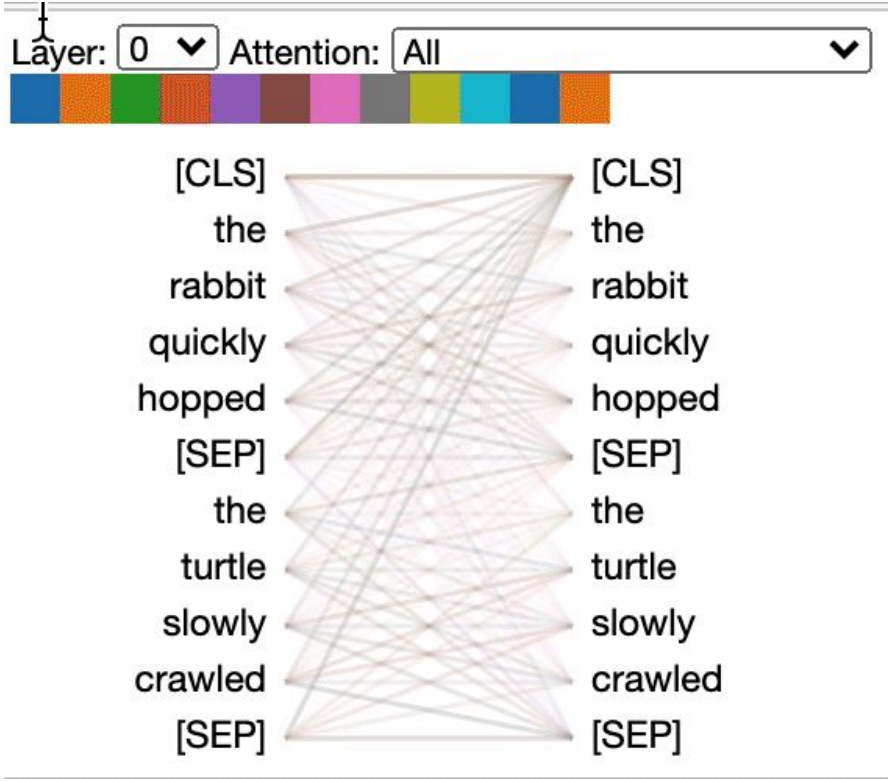
LSTM

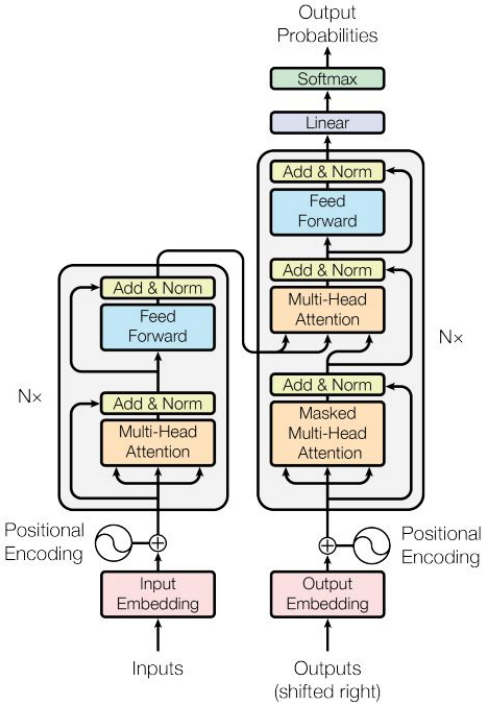
BISHOP OF ELY:
Marry, and will, my lord, to weep in such a one were prettiest;
Yet now I was adopted heir
Of the world's lamentable day,
To watch the next way with his father with his face?

ESCALUS:
The cause why then we are all resolved more sons.

https://www.tensorflow.org/text/tutorials/text_generation

Self-Attention and Transformers





https://www.tensorflow.org/text/tutorials/text_generation

Sequence Modeling

Transformer: GPT-3

The unicorns told the scientists that they had left the valley to explore the world, but that after years of traveling the world, they found humans to be incredibly stupid and decided to return to the valley to lve in peace.

“We were shocked”, said Dr. St. maurice. “We had no idea that the unicorns had been waiting for us. We also had no idea that the unicorns were capable of speech. It was a truly magical moment.”

https://www.buildgpt3.com/post/88/

https://openai.com/api/

Final Notes

CNNs allow us to use images in machine learning in an efficient way

Classifiers want to extract feature for estimation

Generative models want to learn the distribution of the data

CNNs can be used for sequence modelling

RNNs are better than CNNs for NLP

LSTMs solves an inherent problem to RNNs

There is so much more…

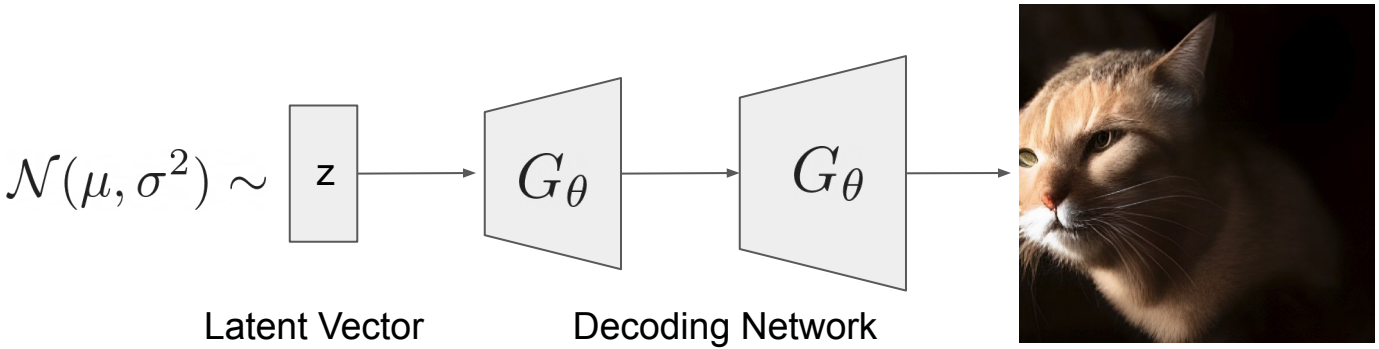
The END

bryan.cardenasguevara@surf.nl
bryca@pm.me

UNUSED SLIDES

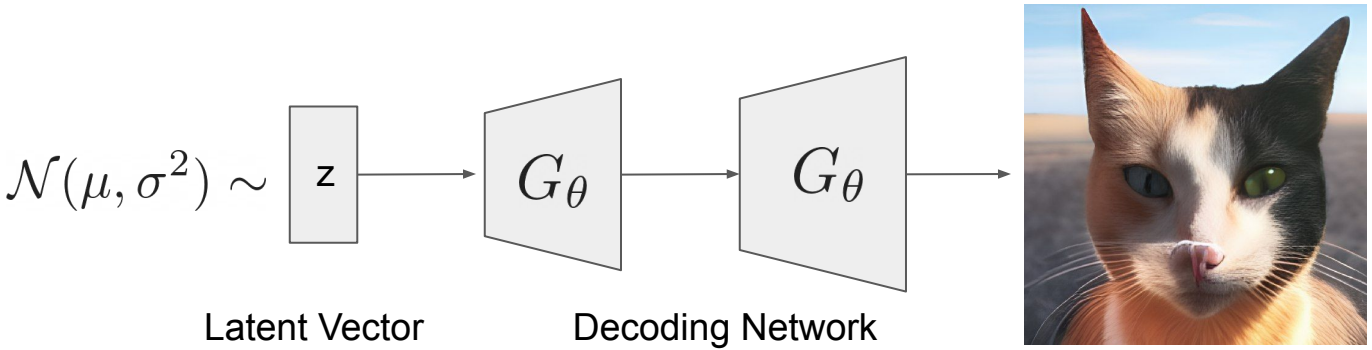
Generative Models

Generator



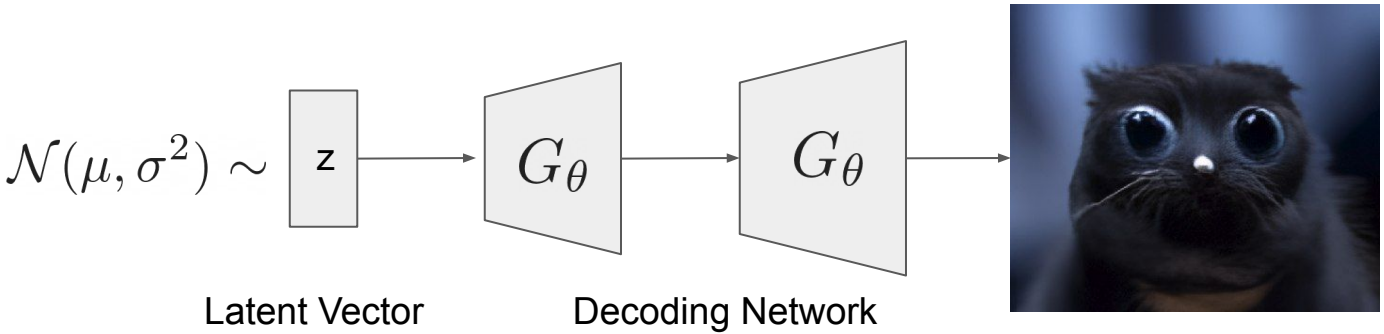
Generative Models

Generator



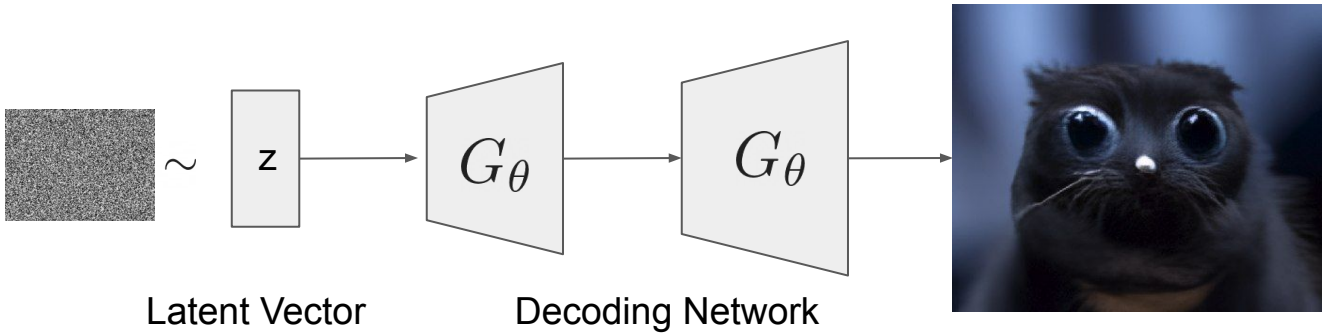
Generative Models

Generator

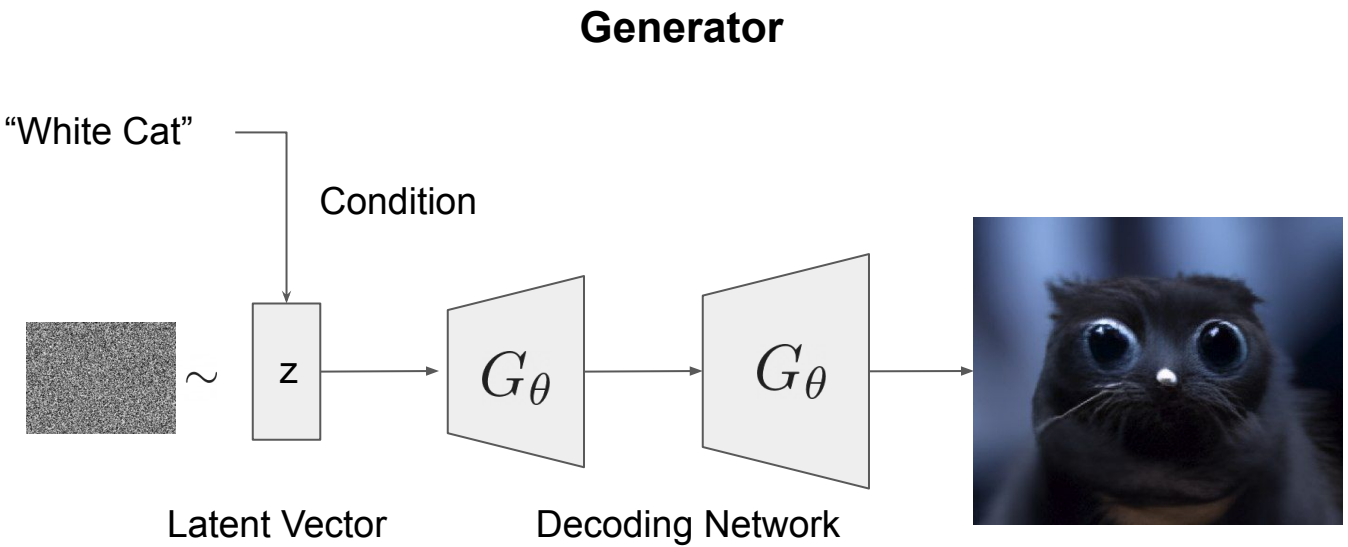


Generative Models

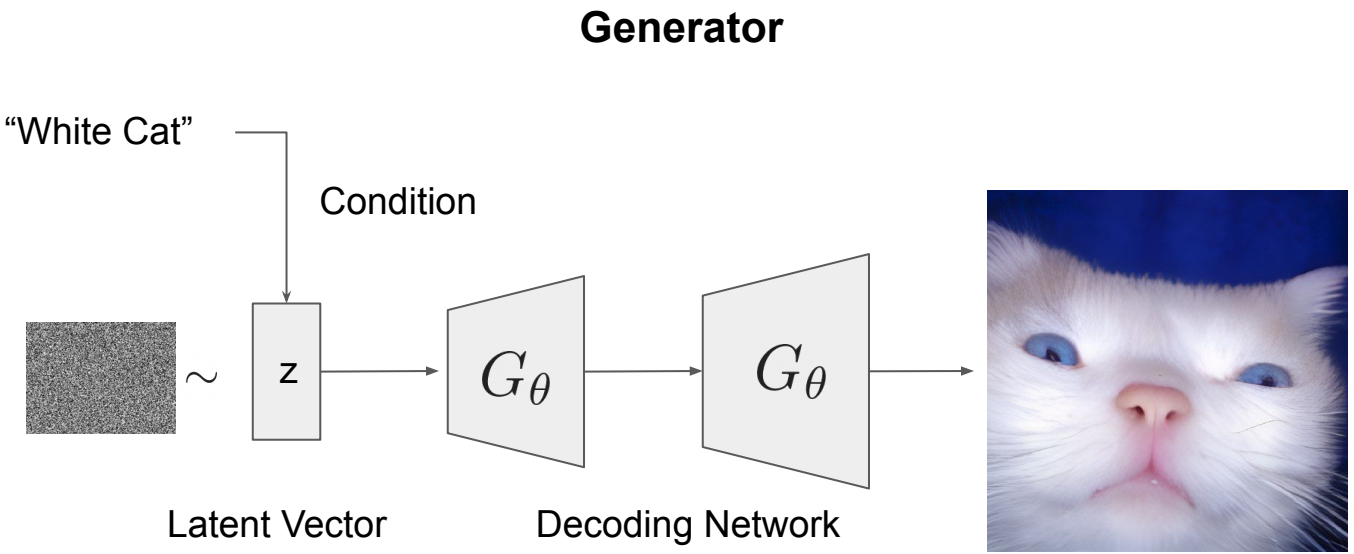
Generator



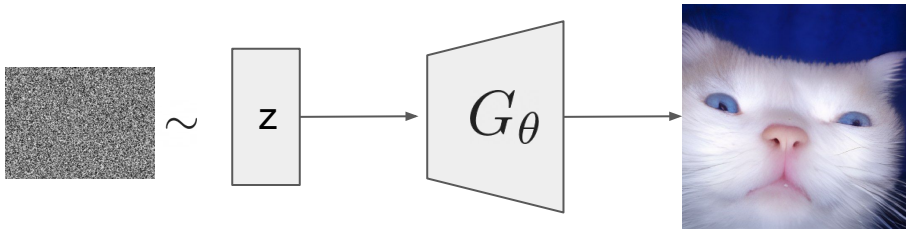
Generative Models



Generative Models



Adversarial Training



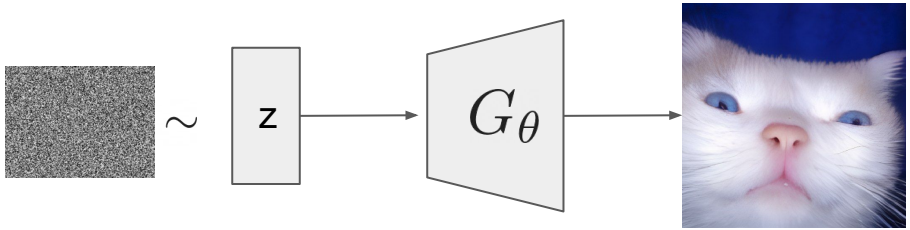
Latent Vector

Decoding Network

Generator

Adversarial Training

Training set

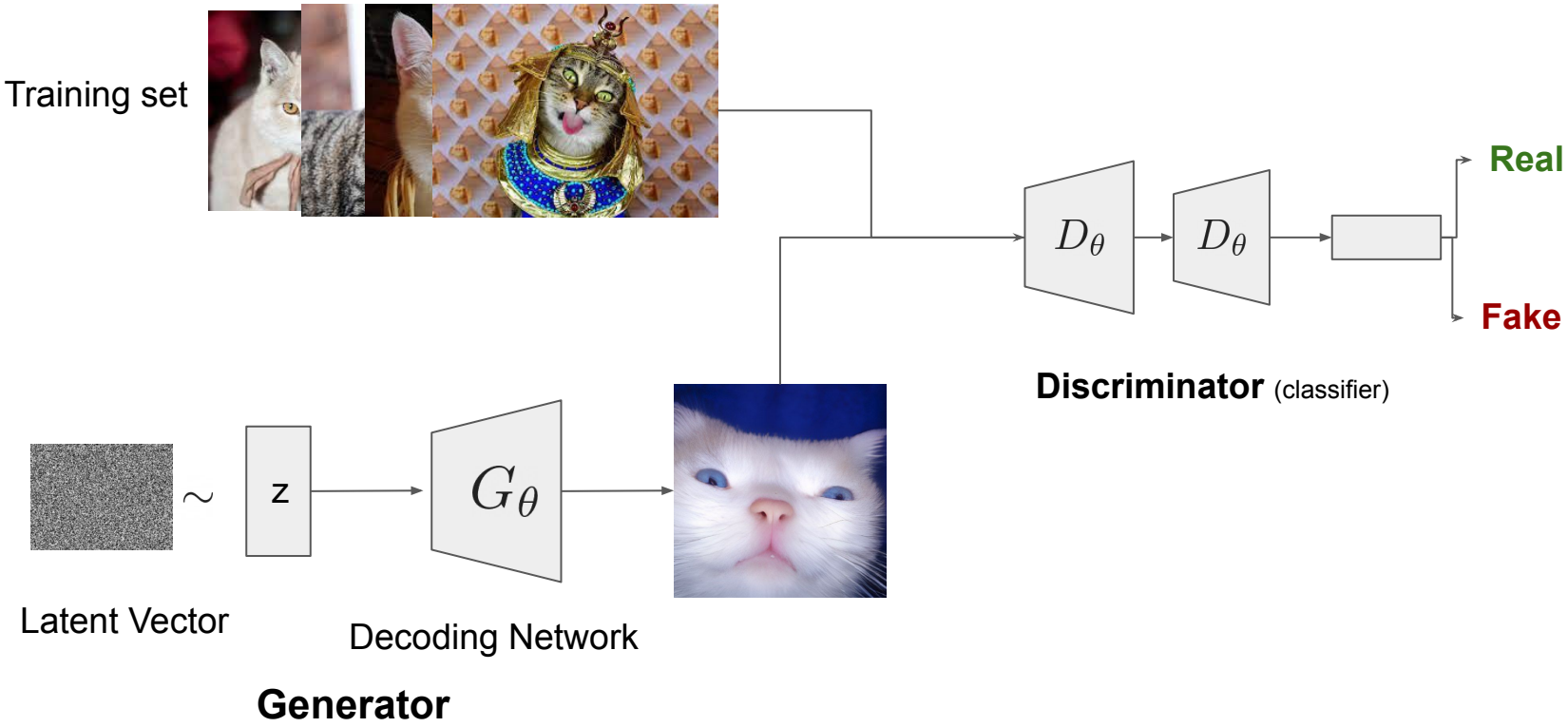


Latent Vector

Decoding Network

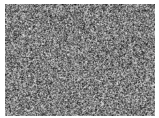
Generator

Adversarial Training



Adversarial Training

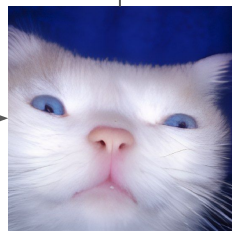
Training set



$\sim$

z

G_θ



Latent Vector

Decoding Network

Generator

D_θ D_θ

Real

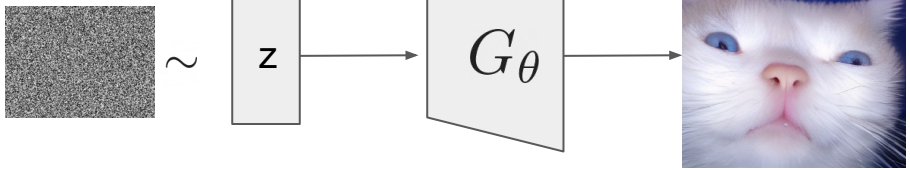
Fake

Discriminator (classifier)

We need both networks to find an **optimal equilibrium**, where the generator creates good images to fool the discriminator that they are real

Adversarial Training

Training set



Latent Vector

Decoding Network

Generator

Discriminator (classifier)

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

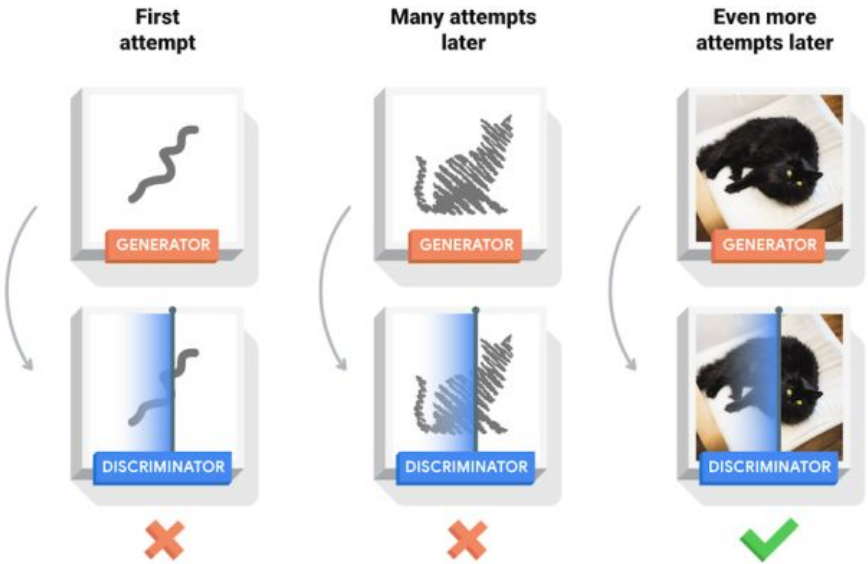
Log probability of **D** predicting the **real images** to be genuine

Log probability of **G** predicting the generated images to be **fake**

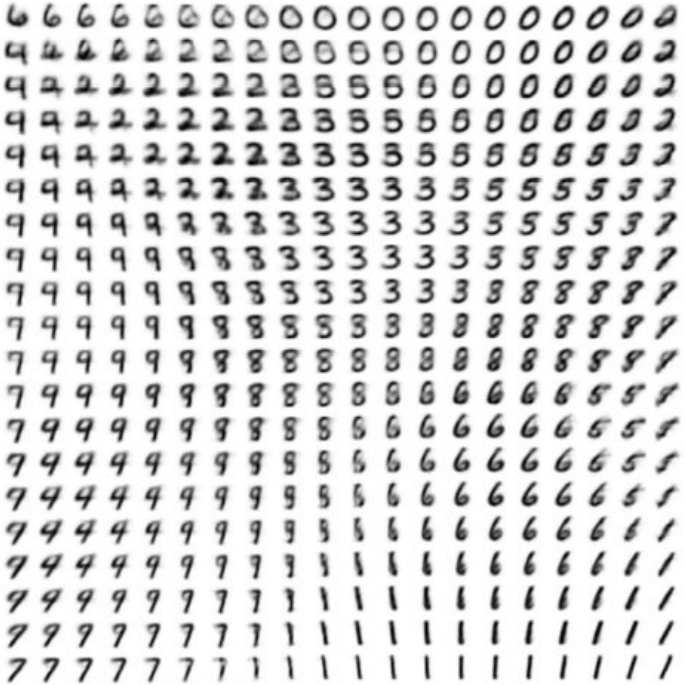
Adversarial Training

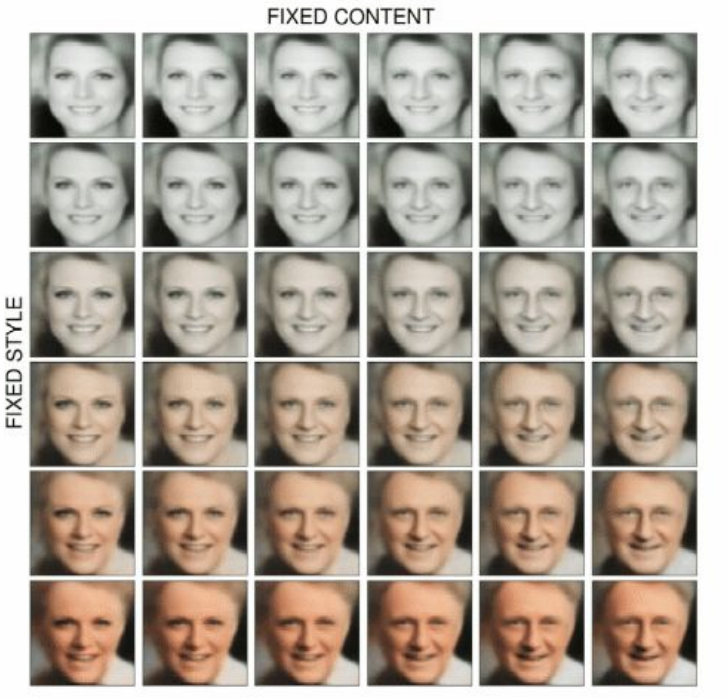
During the training:

- The **generator** progressively **becomes better** in creating images that looks real;
- The **discriminator** progressively **becomes better** in telling them apart.
- The process reaches an **equilibrium** when the discriminator can no longer distinguish real images from fakes.



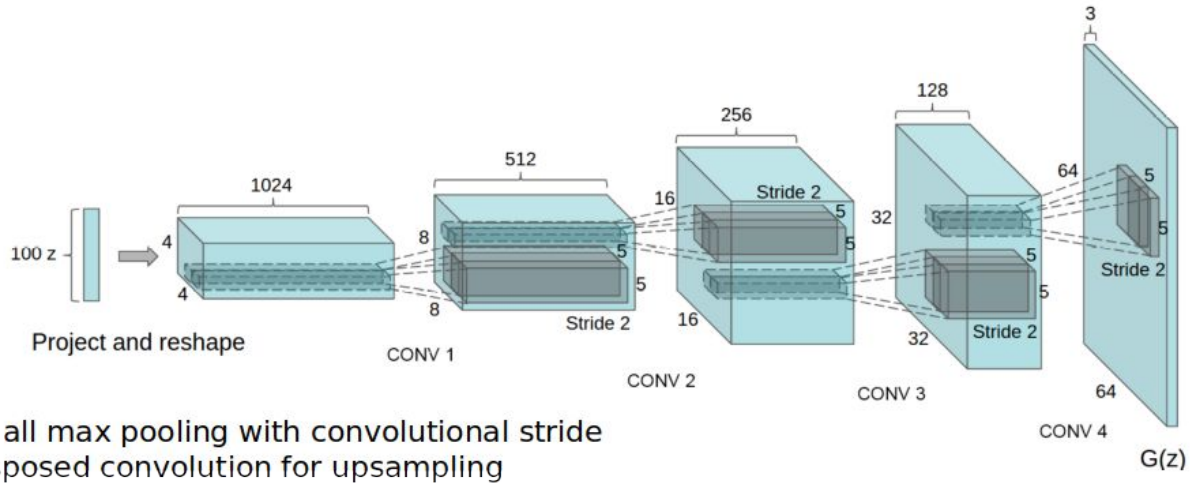
Latent Vector Interpolation





Case Study: DCGAN

DCGAN: Deep Convolutional Generative Adversarial Network

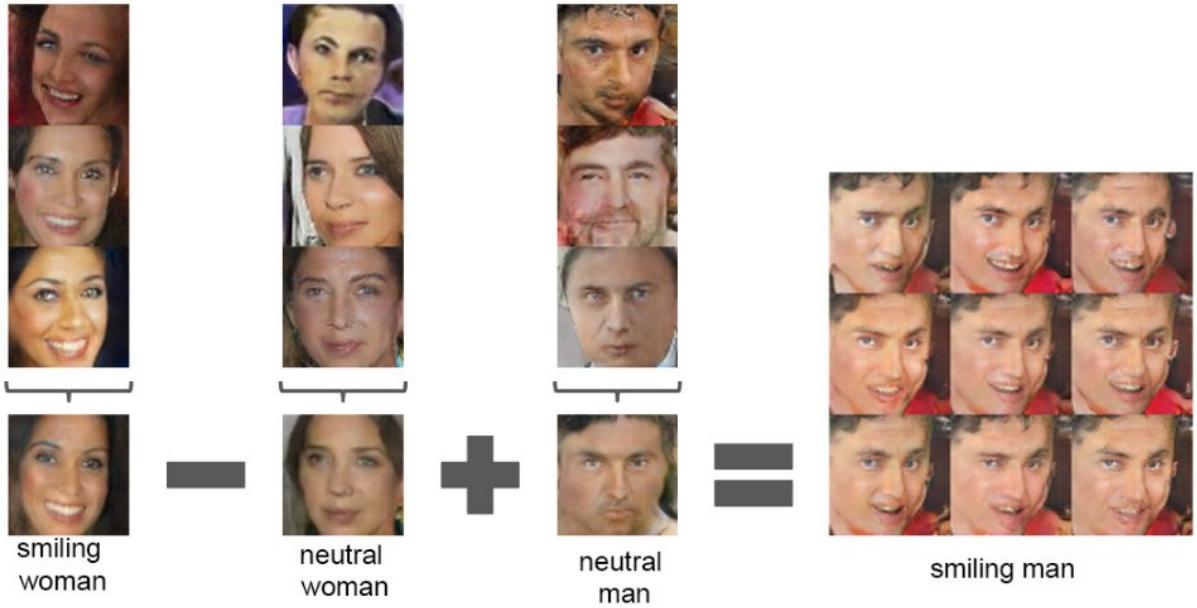


- Replaces **all max pooling with convolutional stride**
- Use transposed convolution for upsampling
- Eliminate fully connected layers
- Use batch normalization except the output layer for the generator
- Use batch normalization for the input layer of the discriminator
- Use ReLU in the generator except for the output (which uses tanh)
- Use LeakyReLU in the discriminator.

Radford, Metz & Chintala (2015, arXiv:1511.06434)

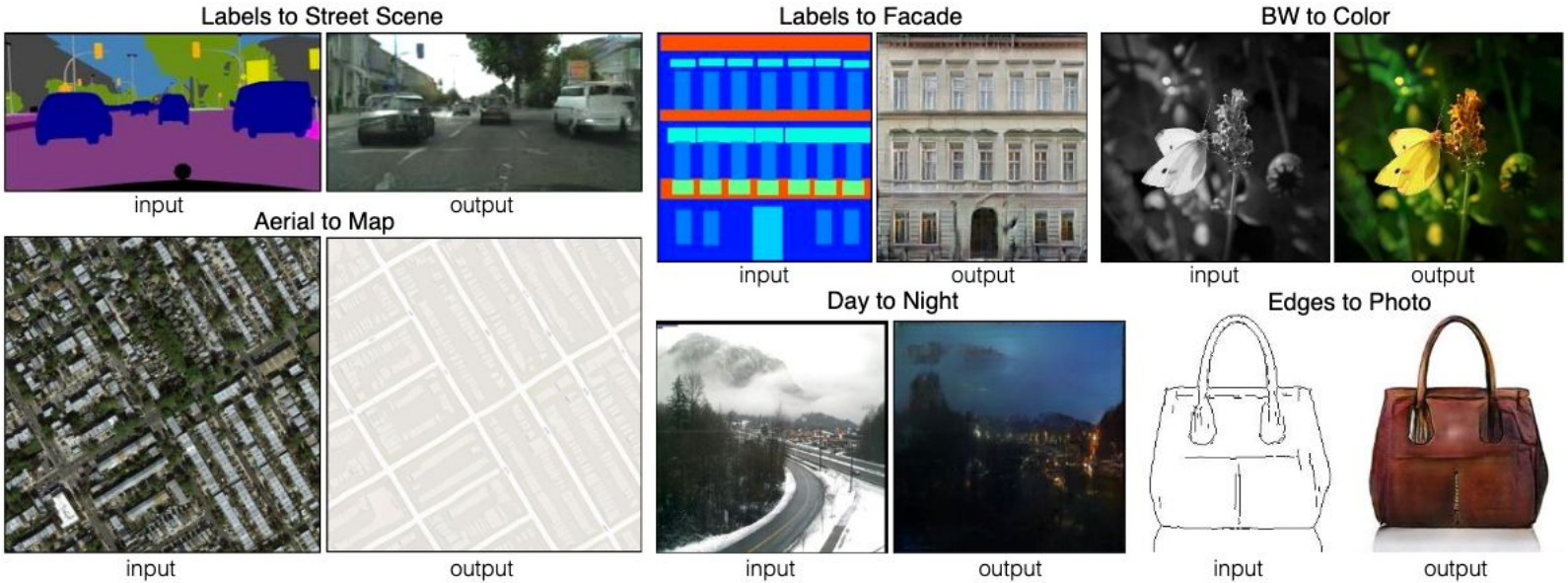
Case Study: DCGAN

Latent Space Arithmetic



Mirza & Osindero (2014, arXiv:1411.1784); Isola et al. (2016, arXiv:1611.07004)

Case Study: CGAN



Mirza & Osindero (2014, arXiv:1411.1784); Isola et al. (2016, arXiv:1611.07004)

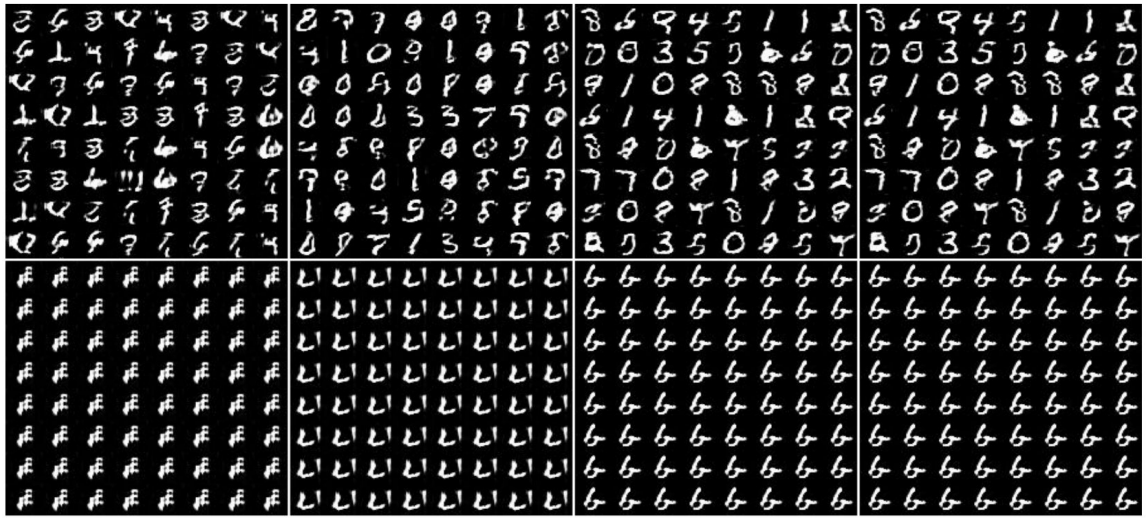
Quality of GAN-Generated images

Problems with GANs

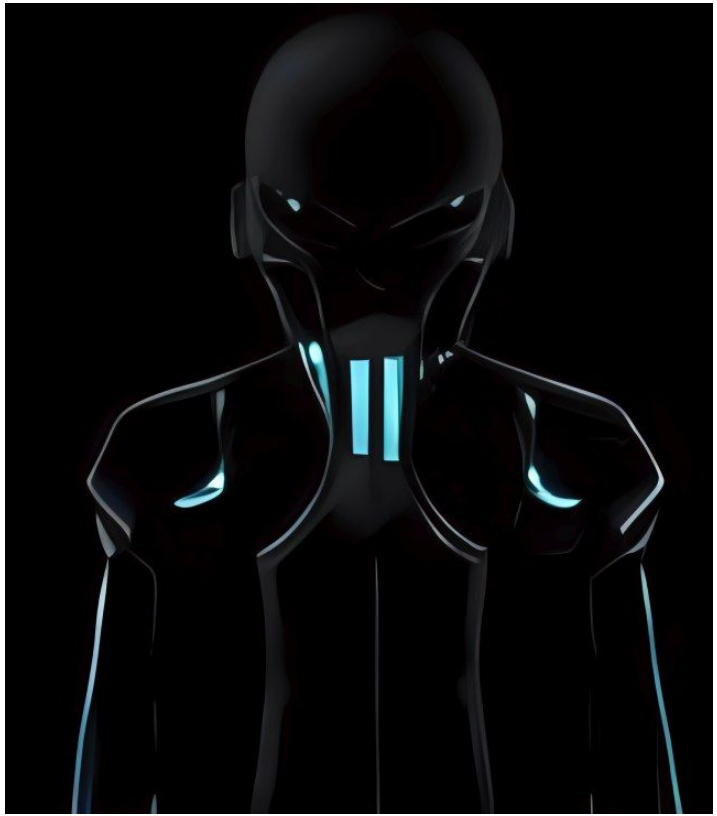
Convergence is not guaranteed: Learning is unstable due to adversarial nature -> no convergence

Mode Collapse: Limited variation learnt

Diminished Gradient: The discriminator is too good telling fake from real -> the generator learns nothing



Quality of GAN-Generated images



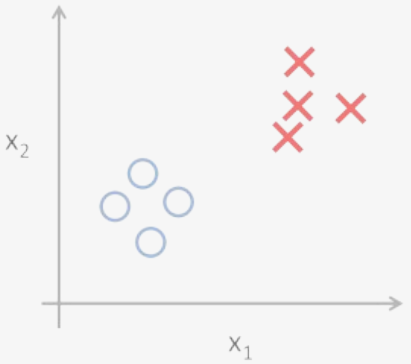
Reinforcement Learning

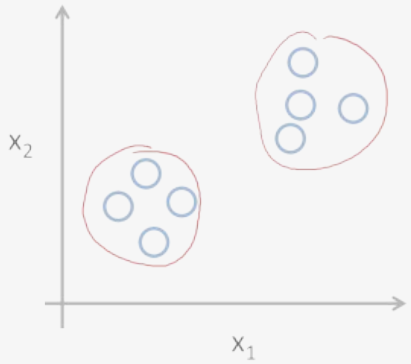
Supervised

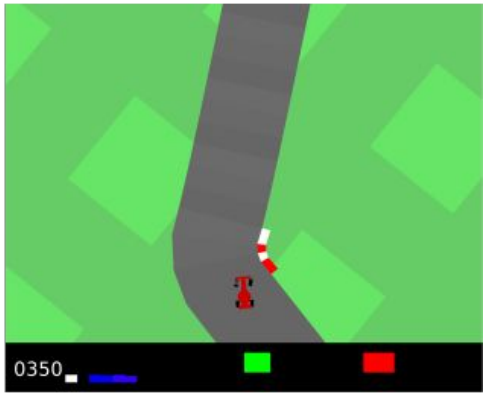
Unsupervised

Reinforcement

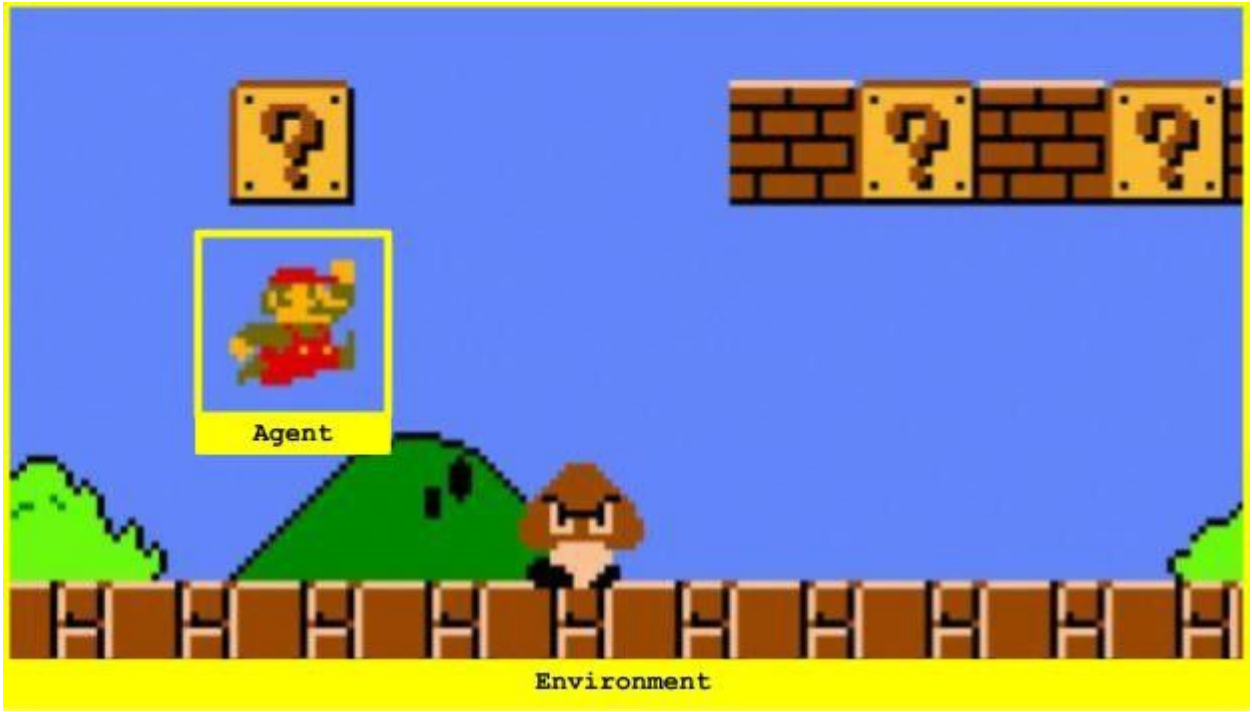
Learn from mistakes







Agent & Environment

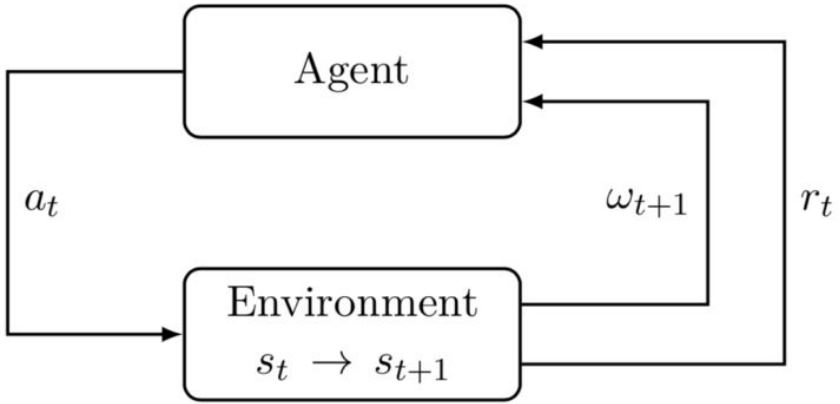


Agent & Environment

Subset of ML that deals with **sequential decision-making**

Maximize cumulative reward given sequences of actions in an environment

Learn good behaviour through experience

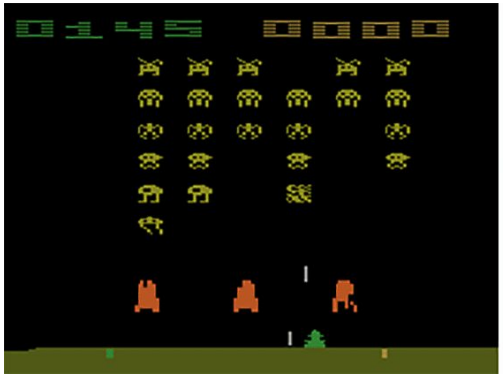


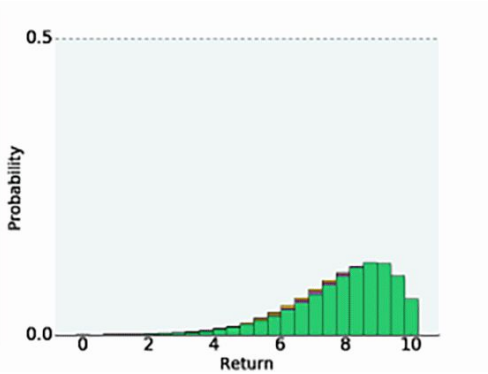
State $s_t \in S$

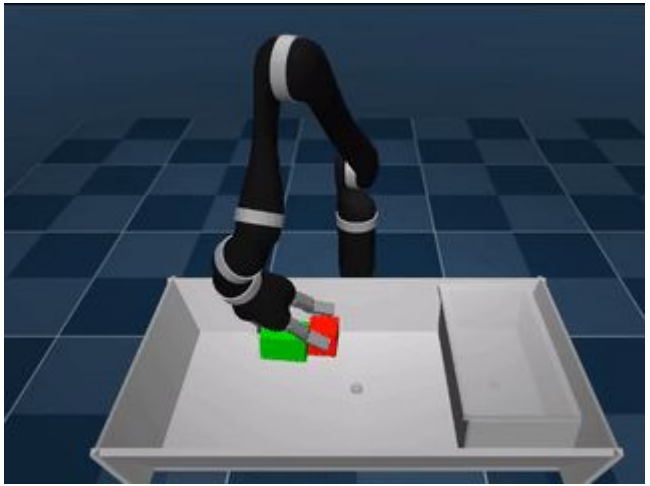
Action $a_t \in A$

Observation $\omega_t \in \Omega$

Agent & Environment







Agent & Environment

Our World is way too complicated

Reinforcement learning is very promising but has a long way to go