Machine Learning for Image Synthesis

Julie Digne

Master ID3D
LIRIS - CNRS
Équipe Origami

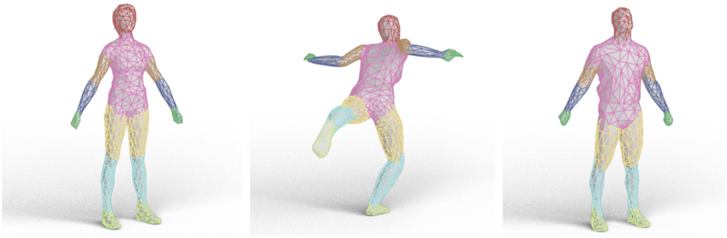
16/11/2023

Teaser 1

```
  File "/usr/lib/python3.9/multiprocessing/pool.py", line 771, in get
    raise self._value
AssertionError: daemonic processes are not allowed to have children
```

Stable Diffusion [Rombach et al.2022], Cody Blakeney (@code_star)

Teaser 2



MeshCNN [Hanocka et al. 2019]

Outline

Classical Vision Algorithms

- Line detection:

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example one rectangle and two triangles?

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example one rectangle and two triangles?
- How about when it's more complicated?

Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example one rectangle and two triangles?
- How about when it's more complicated?

Problem

MS-COCO : 91 categories of objects. [Biederman 87]: around 10000 to 30000 common objects to model: a model for each of them Not doable in practice.

Machine learning and vision

- Recognition/detection

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.
  - Locate an object in an image/video

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video
- Synthesis

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video
- Synthesis
  - Generate an image that looks like a set of examples

Machine learning and vision

- Recognition/detection
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video
- Synthesis
  - Generate an image that looks like a set of examples
  - Generate an image from a sketch given by a user.

Supervised and Unsupervised learning

- Supervised Learning a set of data $(x_i)_i$ and associated labels (ex: cat, car, house...) $(l_i)_i$, learn a function $\hat{f}$ such that $\hat{f}(x_i) = l_i$.

Supervised and Unsupervised learning

- Supervised Learning a set of data $(x_i)_i$ and associated labels (ex: cat, car, house...) $(l_i)_i$, learn a function $\hat{f}$ such that $\hat{f}(x_i) = l_i$.
- Unsupervised Learning a set of data $(x_i)_i$ without any label and learns from similarities between data.

Some examples from previous classes

- Meanshift

Some examples from previous classes

- Meanshift
- K-means

Some examples from previous classes

- Meanshift
- K-means
- Expectation-Maximization

Some examples from previous classes

- Meanshift
- K-means
- Expectation-Maximization

Grouping problems

Unsupervised learning: no label provided for learning the classes.

Is this object in the image?

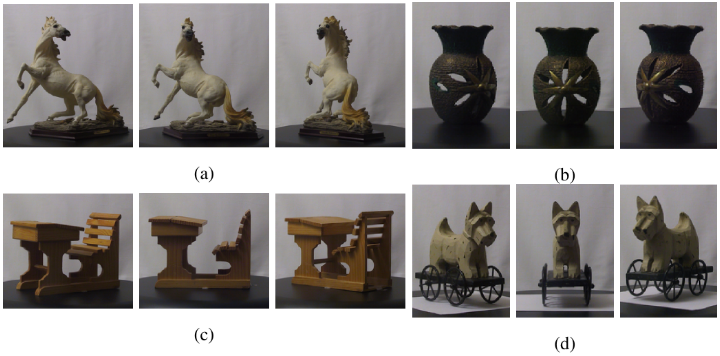
From Wikimedia Commons - user Mikefairbanks

From Wikimedia Commons - user Colonel_Warden

- Recognition/Classification: Is there a bicycle in this image?
- Detection: Where is the bicycle in the image if any?

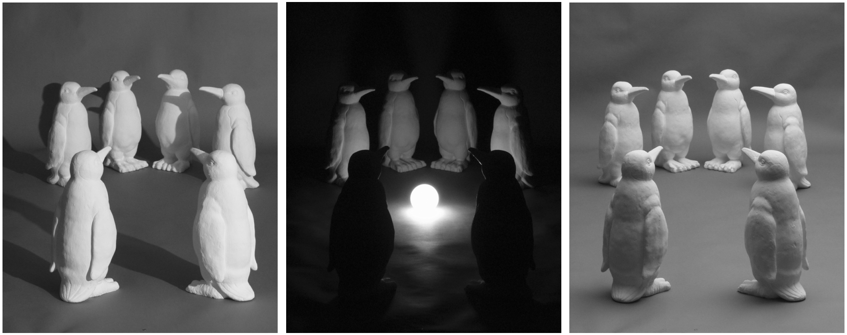
For now: supervised learning setting.

Why is object recognition/detection difficult?



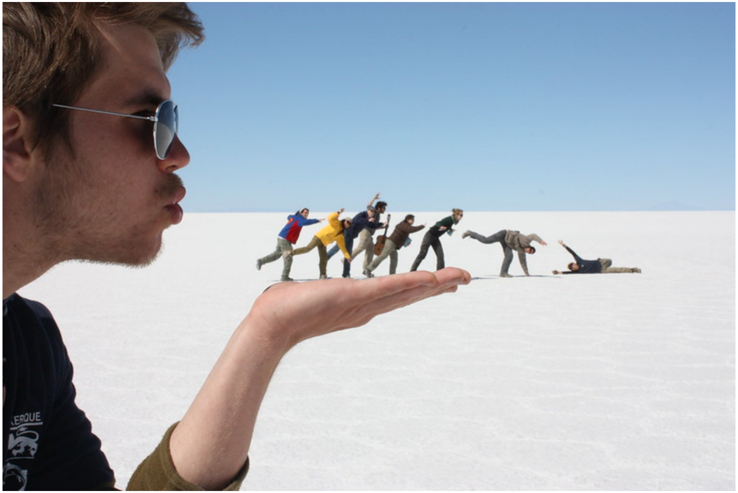
[Özuysal et al. 2010]

Why is object recognition/detection difficult?



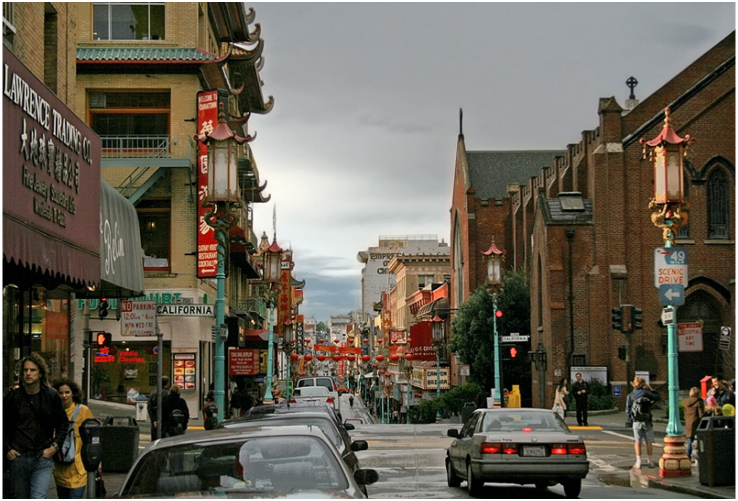
[Koenderink et al. 2007]

Why is object recognition/detection difficult?



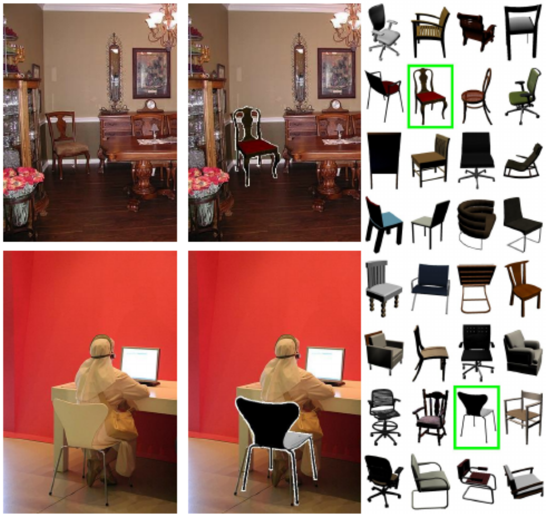
©Jeppe Olsen https://www.flickr.com/photos/jeppeolsen/4802301439/

Why is object recognition/detection difficult?



©Obeck https://www.flickr.com/photos/obeck/144795625/

Why is object recognition/detection difficult?



[Aubry et al. 2014]

Outline

General setup of a supervised machine learning problem

- Data: split into:
  - Training data
  - Evaluation data
  - Test data
- Given data and labels $(x_i, l_i)_i$, find f minimizing an objective function:

$$\sum_i (f(x_i) - l_i)^2$$

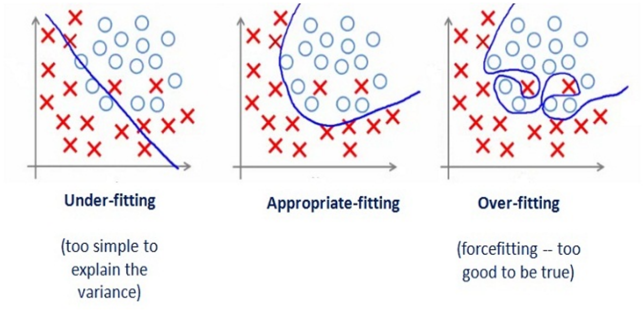
General setup of a supervised machine learning problem

- Data: split into:
  - Training data
  - Evaluation data
  - Test data
- Given data and labels $(x_i, l_i)_i$, find f minimizing an objective function:

$$\sum_i (f(x_i) - l_i)^2$$

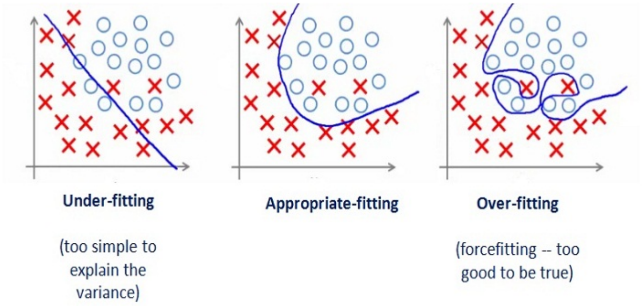
- This is the ℓ^2*loss* but several objective functions exist (also called loss)

Underfitting and Overfitting


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

From medium.com

Underfitting and Overfitting


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

From medium.com

https://data-mining.philippe-fournier-viger.com

Precision and Recall

- **Precision**: how accurate is the classifier in detecting a positive example and not misclassifying.

$$\frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False positives}}$$

- **Recall**: how accurate is the classifier in correctly detecting a positive example.

$$\frac{\#\text{True positives}}{\#\text{positive examples}}$$

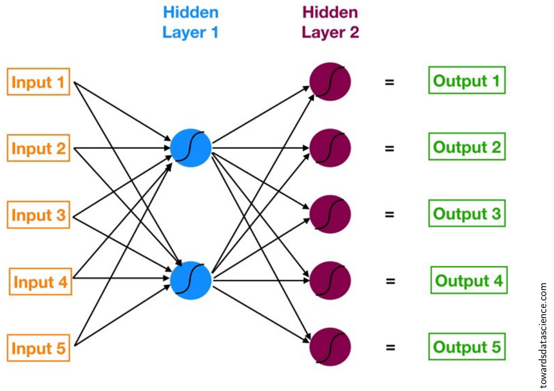
Precision and recall curves are usually drawn with respect to the number of training iterations.

Other indicators

Bias, variance, confusion matrix...

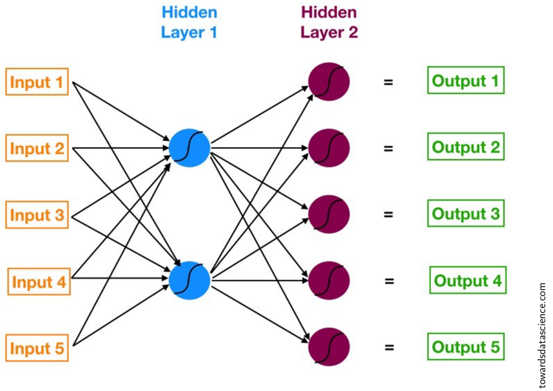
Outline

Neural Network



- Each connection has a weight w
- Each neuron has a bias b and an activation function s (e.g. sigmoid).
- Output of a neuron $s(wx + b)$

Neural Network



- Each connection has a weight w
- Each neuron has a bias b and an activation function s (e.g. sigmoid).
- Output of a neuron $s(wx + b)$

For images

Each pixel is an input to the net.

Training a neural network

Output

In classification cases, the neural network outputs a class the input image supposedly belongs to.

Training a neural network

Output

In classification cases, the neural network outputs a class the input image supposedly belongs to.

Cost function

For training samples, we evaluate how well the neural net performed via a cost function C: Mean Square error,Cross-Entropy...

Training a neural network

To optimize the cost C

Gradient descent with respect to weight w_i and bias b_i for each neuron i.

Training a neural network

To optimize the cost C

Gradient descent with respect to weight w_i and bias b_i for each neuron i.

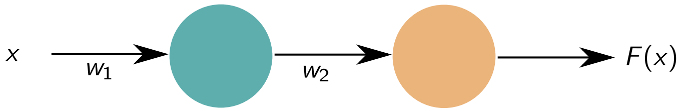
Back-Propagation

The gradient can be propagated back from the output to the input(chain rule).

Back-propagation example

Toy model

Compute the gradient of the cost with respect to each parameter.

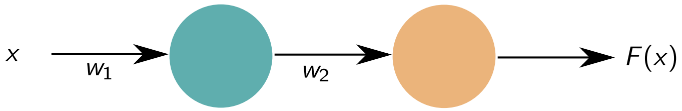


$a^1 = s_1(w_1 x + b_1) \quad a^2 = s_2(w_2 x + b_2)$

Back-propagation example

Toy model

Compute the gradient of the cost with respect to each parameter.



$$a^1 = s_1(w_1 x + b_1) \quad a^2 = s_2(w_2 x + b_2)$$

- In practice start with random weights and bias.

Convolutional Neural Networks

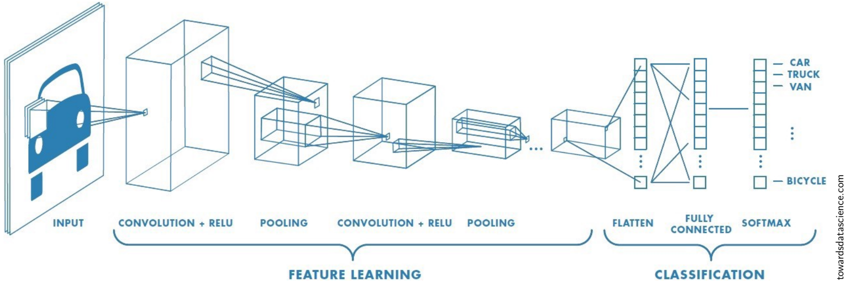
Shared parameters

Dropping fully connected layers, CNN use convolutions by kernels with weights independent of the image location. These weights are optimized during training.

Convolutional Neural Networks

Shared parameters

Dropping fully connected layers, CNN use convolutions by kernels with weights independent of the image location. These weights are optimized during training.



Convolution layer parameters

- Kernel size: controls the locality of the kernel
- Padding: increases the size of the input
- Dilatation: aggregates values from every *n* pixels where *n* is the dilatation. (eq to set some weights in the kernel to 0).
- Stride: performs the convolution centered every *n* pixels where *n* is the stride.

Visualization

`https://ezyang.github.io/convolution-visualizer/index.html`

Outline

Deep Image Prior [Ulyanov et al 2018]

Inverse Problems

We know only a degraded version of an image, we want to recover the original one.

Deep Image Prior [Ulyanov et al 2018]

Inverse Problems

We know only a degraded version of an image, we want to recover the original one.

ML + Inverse Problems

Learn the inverse transform from example

Deep Image Prior [Ulyanov et al 2018]

Inverse Problems

We know only a degraded version of an image, we want to recover the original one.

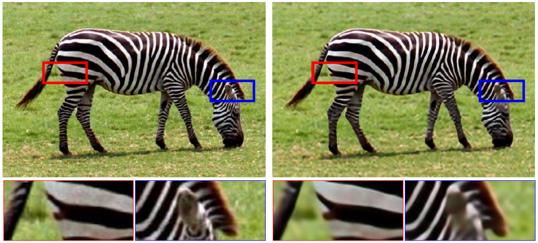
ML + Inverse Problems

Learn the inverse transform from example

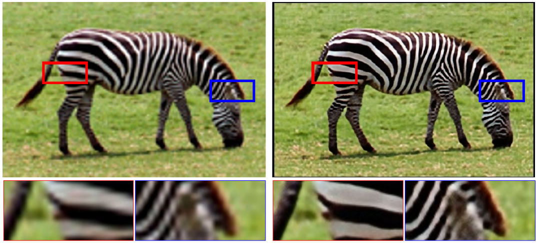
Deep Image Prior

Not all statistics need to be learn from databases, a lot is captured by the structure of generative convolutional nets.

Inverse problem: Super-resolution



(a) Ground truth (b) SRResNet [36], **Trained**



(c) Bicubic, **Not trained** (d) Deep prior, **Not trained**

Deep generator formulation

Deep generator

A network parametered by θ which maps an input code vector z to an image x:

$$x = f_\theta(z)$$

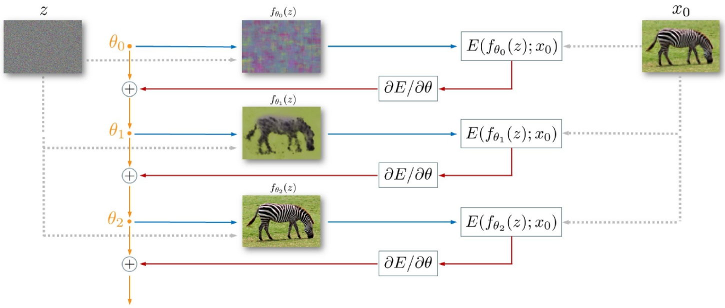
Usual DL approach

Knowledge about the distribution $p(x)$ of x is encoded in θ which should be optimized wrt to a database.

Deep prior approach [Ulyanov 2018]

A significant amount of information is encoded in the structure of the generator network, even without training its parameters.

Principle



[Ulyanov 2018]

Energy formulation

Inverse problem

x_0 is observed, one wants to find x close to x_0 but *better*:

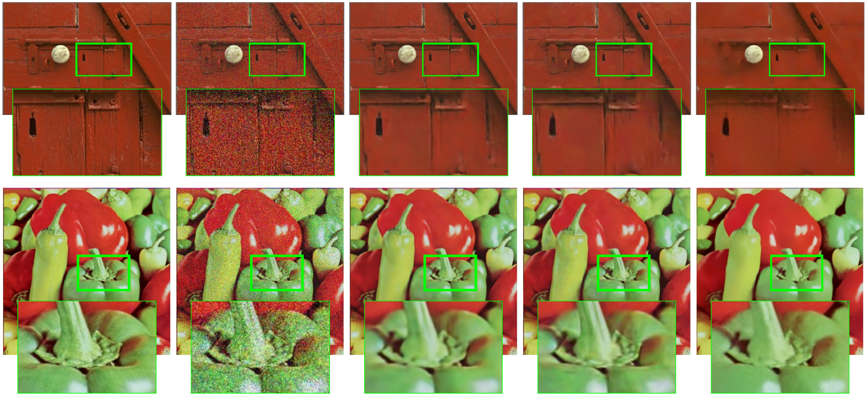
$$x^* = argmin_x E(x, x_0) + R(x)$$

- E is an energy (e.g. $E(x, x_0) = \|x - x_0\|^2$)
- R is a regularization term (e.g. $R(x) = \|x\|_2$, $R(x) = TV(x)$).

Deep Prior = regularizer

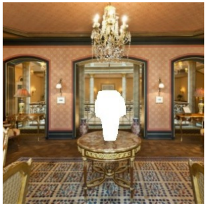
$$\theta^* = argmin_\theta E(x, f_\theta(z) \text{ with } x^* = f_{\theta^*}(z)$$

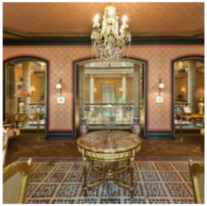
Denoising

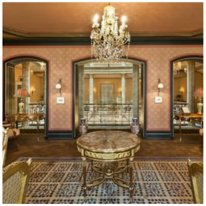

(a) GT (b) Input (c) Ours (d) CBM3D (e) NLM

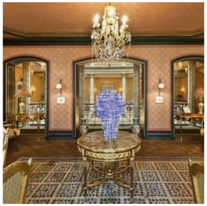
[Ulyanov 2018]

Inpainting


(a) Corrupted image

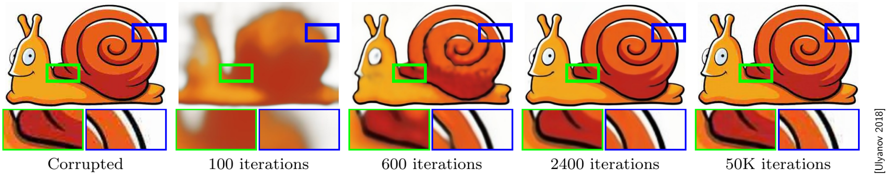

(b) Global-Local GAN [28]


(c) Ours, LR $= 0.01$


(d) Ours, LR $= 10^{-4}$

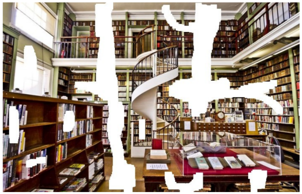
[Ulyanov 2018]

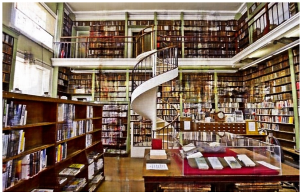
JPEG artefact removal

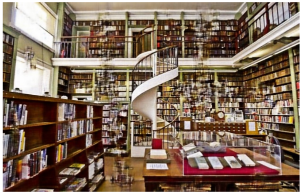


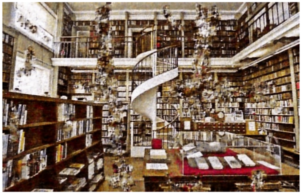
- Stop before overfitting!

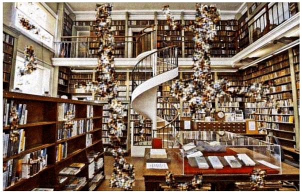
More complex inpainting

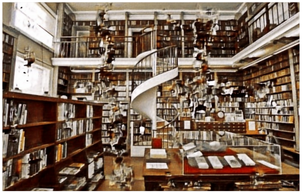

(a) Input (white=masked)


(b) Encoder-decoder, depth=6


(c) Encoder-decoder, depth=4


(d) Encoder-decoder, depth=2


(e) ResNet, depth=8


(f) U-net, depth=5

[Ulyanov 2018]

Lab work

- Train your own deep prior network
- Encoder-decoder architecture with two skip connections
- First application: plain image reconstruction
- Applications: denoising and inpainting

Outline

Generative problems

https://sh-tsang.medium.com/review-dcgan-deep
-convolutional-generative-adversarial-network-gan-ec390cded63c

Generative Problems

Goal

Given a set of samples $x_1, x_2, \cdots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$.

Generative Problems

Goal

Given a set of samples $x_1, x_2, \cdots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$.

- In practice, we use some prior knowledge of the problem to model p_θ.

Generative Problems

Goal

Given a set of samples $x_1, x_2, \cdots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$.

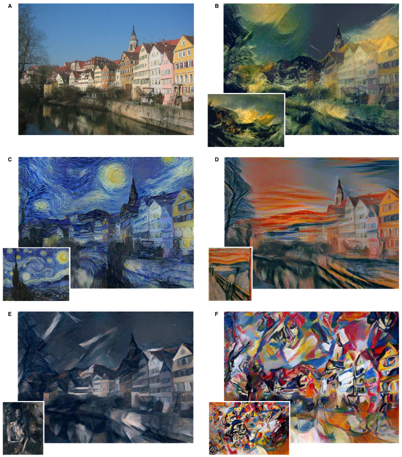
- In practice, we use some prior knowledge of the problem to model p_θ.
- Optimize θ, to minimize the difference between p and p_θ.

An almost-training-free approach

Idea

Use a pretrained CNN (ImageNet) and make the features resemble those of the target image (using gradient descent)

- Texture synthesis [Gatys et al. 2015]
- Style tranfer [Gatys et al. 2016].



[Gatys et al. 2016]

Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \cdots x_n$

Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \cdots x_n$
- $p_\theta(x_i) = \prod_i p(x_i|x_{i-1}, \cdots x_n)$

Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \cdots x_n$
- $p_\theta(x_i) = \prod_i p(x_i | x_{i-1}, \cdots x_n)$

Process

Generate pixels sequentially starting from a corner. Dependency on the previous pixels modeled by a Recurrent Neural Network (PixelRNN) or a Convolutional Neural Network (PixelCNN).

PixelRNN

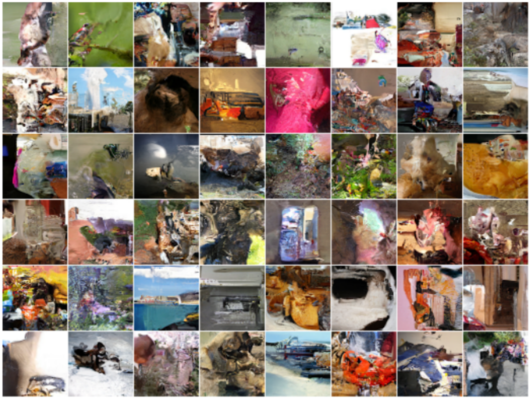


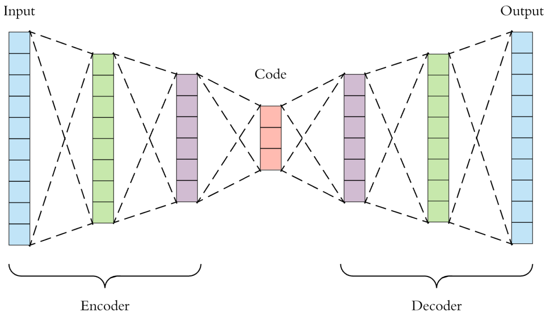
Image from [Van der Oord et al. 2016]

Samples trained on ImageNet, 64x64 images.

Pros and Cons

Pros: explicit model of p_θ, Good evaluation metric
Cons: slow because of sequential generation

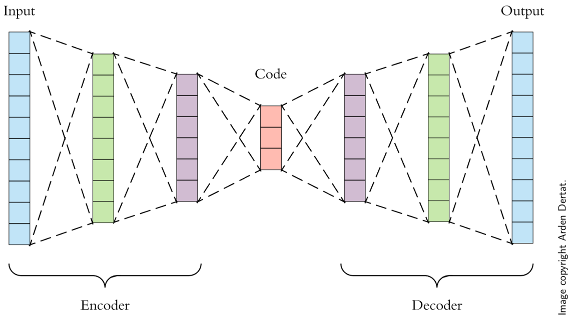
Auto-encoders



Goal

Given input data x produce z smaller than x that sums up x.

Auto-encoders

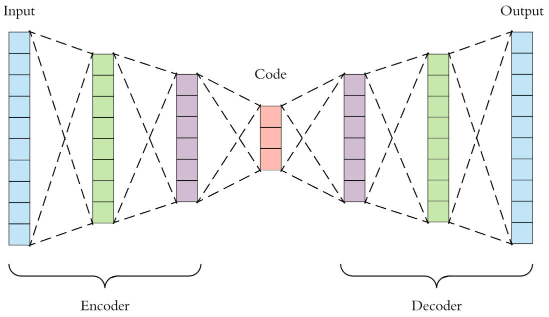


Goal

Given input data x produce z smaller than x that sums up x.

- Training done by encoding x into z, decoding z into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.

Auto-encoders

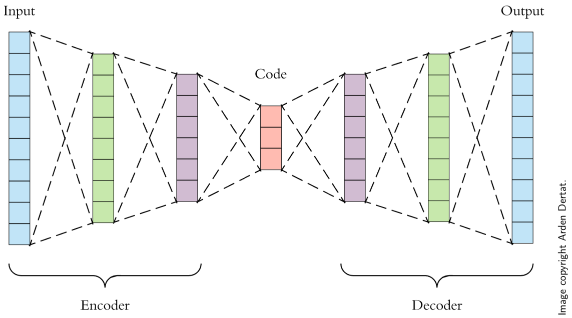


Goal

Given input data x produce z smaller than x that sums up x.

- Training done by encoding x into z, decoding z into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a sample in the latent space

Auto-encoders

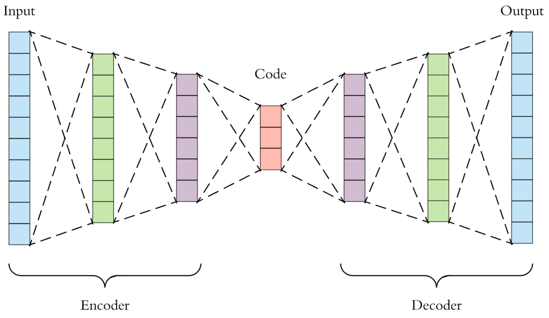


Goal

Given input data x produce z smaller than x that sums up x.

- Training done by encoding x into z, decoding z into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a sample in the latent space

Auto-encoders

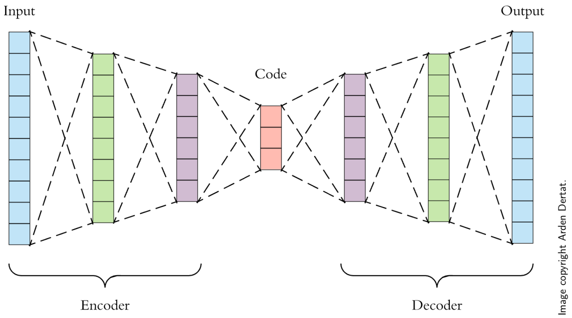


Goal

Given input data x produce z smaller than x that sums up x.

- Training done by encoding x into z, decoding z into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a sample in the latent space
- If the goal is to compute an embedding: after training throw away the decoder For generating new data, we keep only the decoder.

Auto-encoders

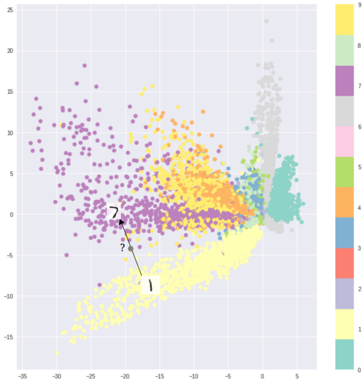


Goal

Given input data x produce z smaller than x that sums up x.

- Training done by encoding x into z, decoding z into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a sample in the latent space
- If the goal is to compute an embedding: after training throw away the decoder For generating new data, we keep only the decoder.

Latent space

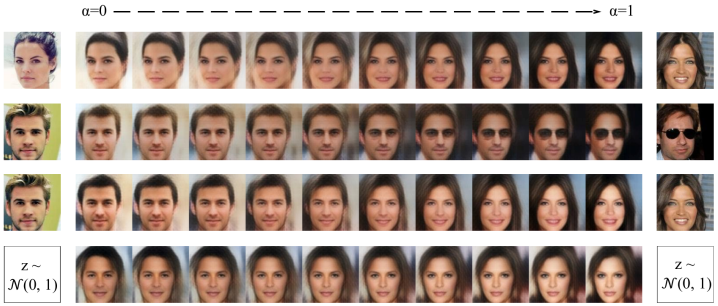


https://towardsdatascience.com

Problem

The data are not spread in the latent space and well clustered.

Variational Auto-encoder [Kingma and Welling 2016]

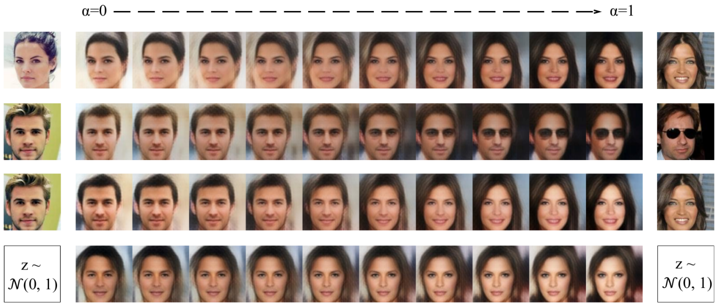


[Kingma and Welling 2016]

Idea

Ensure that the data spreads well in the latent space.

Variational Auto-encoder [Kingma and Welling 2016]



[Kingma and Welling 2016]

Idea

Ensure that the data spreads well in the latent space. Add some noise to embeddings in latent space and decode: the output should still be “valid”.

VAE

In practice

Instead of learning a vector embedding, the encoder outputs a covariance and mean.

- To decode, we sample from a Gaussian distribution with predicted covariance and mean and compare the distributions using Kullback-Leibler divergence.

Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$

Variational Auto-Encoder (VAE)

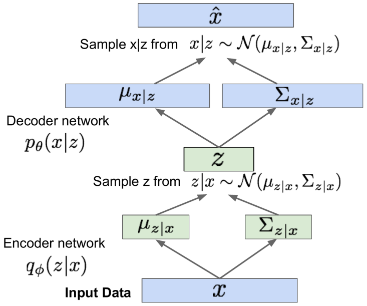
- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).

Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).
- Decoder Network: $p_\theta(x|z)$, gaussian model: $\mu_{x|z}, \Sigma_{x|z}$, can sample $x|z$ (probabilistic decoder).

Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).
- Decoder Network: $p_\theta(x|z)$, gaussian model: $\mu_{x|z}, \Sigma_{x|z}$, can sample $x|z$ (probabilistic decoder).



[Li, Johnson, Yeung, 2017]

Objective function in a VAE

Minimization

Computing parameters θ, ϕ maximizing:
$\mathcal{L}(x_i, \theta, \phi) = \log p_\theta(x_i) \geq E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)||p_\theta(z))$

Objective function in a VAE

Minimization

Computing parameters θ, ϕ maximizing:
$\mathcal{L}(x_i, \theta, \phi) = \log p_\theta(x_i) \geq E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)||p_\theta(z))$

- $\mathcal{L}(x_i, \theta, \phi)$ is a lower bound of $p_\theta(x_i)$

Image generation using VAE

- Sample z from gaussian prior (diagonal covariance).

Image generation using VAE

- Sample z from gaussian prior (diagonal covariance).
- Run z through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.

Image generation using VAE

- Sample z from gaussian prior (diagonal covariance).
- Run z through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x|z, \Sigma_x|z)$.

Image generation using VAE

- Sample z from gaussian prior (diagonal covariance).
- Run z through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x|z, \Sigma_x|z)$.

Remark

Diagonal covariance for z yields independent latent variables corresponding to interpretable factors of variation.

Image generation using VAE

- Sample z from gaussian prior (diagonal covariance).
- Run z through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x|z, \Sigma_x|z)$.

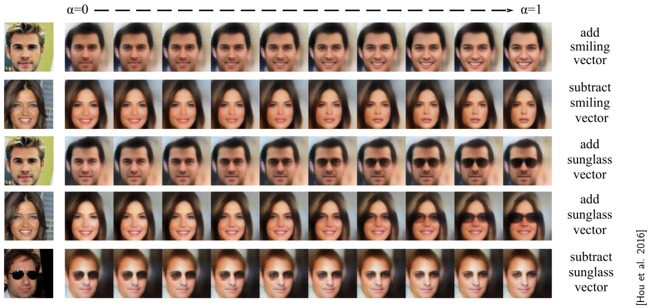
Remark

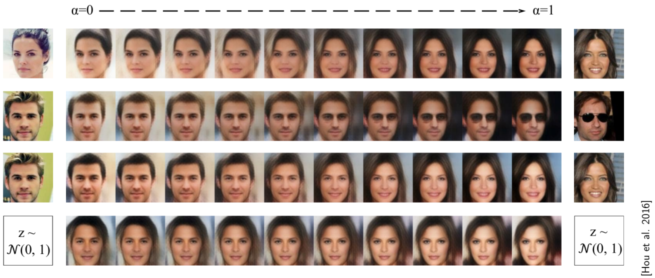
Diagonal covariance for z yields independent latent variables corresponding to interpretable factors of variation.

Pros & Cons

Pros: Interpolation possible in latent space. Latent variables can be interpretable.
Cons: Maximizes a lower bound of the likelihood, blurry results.

VAE Applications





Outline

GAN Principle

- We are not going to model explicitly the density $p_\theta(x)$
- But we will be able to sample from it!
- Sample from a simple distribution and learn the transform to the training distribution.

GAN Principle

- We are not going to model explicitly the density $p_\theta(x)$
- But we will be able to sample from it!
- Sample from a simple distribution and learn the transform to the training distribution.

Generative Adversarial training

Admit you have an oracle $\mathcal{D}$ that rates if an image I looks real ($\mathcal{D}(I) = 1$) or unreal ($\mathcal{D}(I) = 0$). If you want to synthesize an image, you want this oracle to judge the synthesized image as real.

GAN Principle

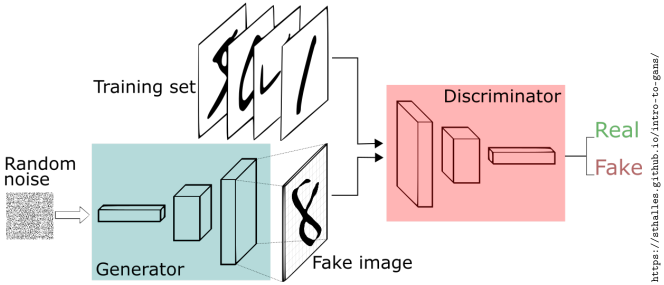
- We are not going to model explicitly the density $p_\theta(x)$
- But we will be able to sample from it!
- Sample from a simple distribution and learn the transform to the training distribution.

Generative Adversarial training

Admit you have an oracle $\mathcal{D}$ that rates if an image I looks <u>real</u> ($\mathcal{D}(I) = 1$) or <u>unreal</u> ($\mathcal{D}(I) = 0$). If you want to synthesize an image, you want this oracle to judge the synthesized image as <u>real</u>.

- Saddly, we have no oracle D available.

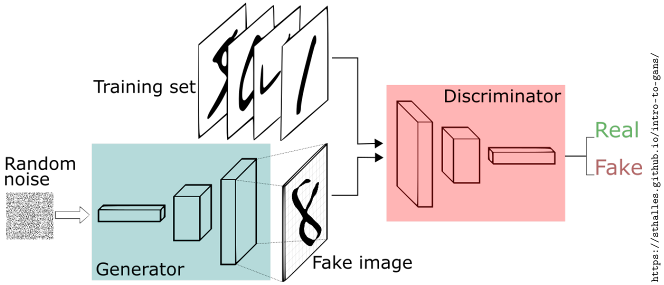
GAN



2 players Game

G tries to synthesize images that will fool D and D tries to distinguish between real images and fake images synthesized by G.

GAN



2 players Game

G tries to synthesize images that will fool D and D tries to distinguish between real images and fake images synthesized by G.

Objective Function

$$\min_{\theta_G} \max_{\theta_D} \mathrm{E}_{x \sim p_{data}(x)}[\log D_{\theta_D}(x)] + \mathrm{E}_{z \sim p_{prior}(z)}[\log(1 - D_{\theta_D}(G_{\theta_G}(z)))]$$

Where θ_D (resp. θ_G) are the parameters of the discriminator (resp. generator).

GAN training

Alternate optimization

Alternate between

1. Optimize parameters θ_D by gradient ascent (θ_G fixed).
2. Optimize parameters θ_G by gradient descent (θ_D fixed).

GAN training

Alternate optimization

Alternate between

1. Optimize parameters θ_D by gradient ascent (θ_G fixed).
2. Optimize parameters θ_G by gradient descent (θ_D fixed).

Do we need all the terms of the objective functions for the two steps?

GAN training

Alternate optimization

Alternate between

1. Optimize parameters θ_D by gradient ascent (θ_G fixed).
2. Optimize parameters θ_G by gradient descent (θ_D fixed).

Do we need all the terms of the objective functions for the two steps?

Problem

In practice hard to optimize! Alternative:

1. Optimize parameters θ_D by gradient ascent (θ_G fixed).
2. Optimize parameters θ_G by gradient ascent (θ_D fixed) with objective:

$$\max_{\theta_G} \mathrm{E}_{z\sim p_{prior}(z)} \log D_{\theta_D}(G_{\theta_G}(z))$$

Training Algorithm

Algorithm 1: Training

```
1 for $j = 1 \cdots N$ do
2     for $k = 1 \cdots K$ do
3         Sample a minibatch of $m$ samples $z_i$;
4         Sample a minibatch of $m$ real samples $x_i$;
5         Update $\theta_D$:
              $$\theta_D = \theta_D + \nu \nabla_{\theta_D} \left( \sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i)) \right)$$
6     Sample a minibatch of $m$ samples $z_i$;
7     Update $\theta_G$:
          $$\theta_G = \theta_G + \nu \nabla_{\theta_G} \left( \sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i)) \right)$$
```

Training Algorithm

Algorithm 2: Training

```
1 for $j = 1 \cdots N$ do
2     for $k = 1 \cdots K$ do
3         Sample a minibatch of $m$ samples $z_i$;
4         Sample a minibatch of $m$ real samples $x_i$;
5         Update $\theta_D$:
              $$\theta_D = \theta_D + \nu \nabla_{\theta_D} (\sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i)))$$
6     Sample a minibatch of $m$ samples $z_i$;
7     Update $\theta_G$:
          $$\theta_G = \theta_G + \nu \nabla_{\theta_G} (\sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i)))$$
```

Generation

Sample z and generate $\hat{x} = G(z)$.

Training Algorithm

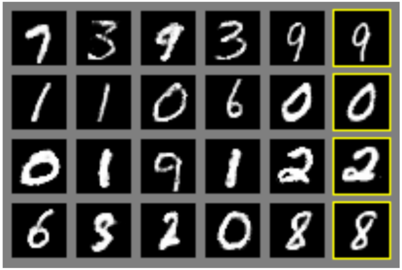
Algorithm 3: Training

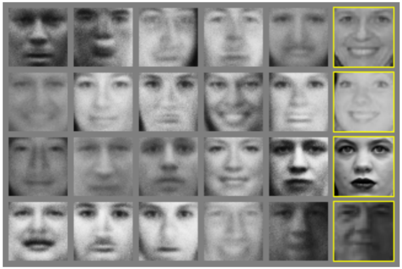
```
1 for $j = 1 \cdots N$ do
2     for $k = 1 \cdots K$ do
3         Sample a minibatch of $m$ samples $z_i$;
4         Sample a minibatch of $m$ real samples $x_i$;
5         Update $\theta_D$:
              $$\theta_D = \theta_D + \nu \nabla_{\theta_D} (\sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i)))$$
6     Sample a minibatch of $m$ samples $z_i$;
7     Update $\theta_G$:
          $$\theta_G = \theta_G + \nu \nabla_{\theta_G} (\sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i)))$$
```

Generation

Sample z and generate $\hat{x} = G(z)$. D is not needed.

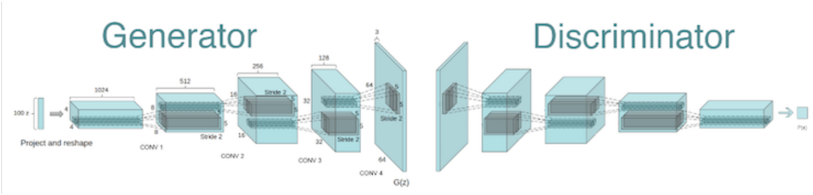
Results





[Goodfellow et al. 2014]

What are D and G?



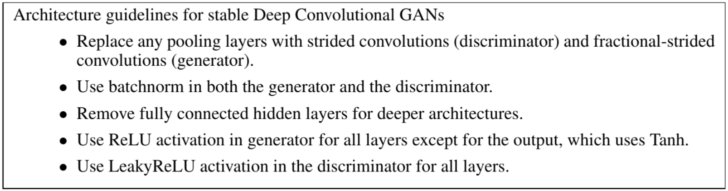
[Radford et al. 2016]

Deep convolutional GANs

GAN analysis

Pros and Cons

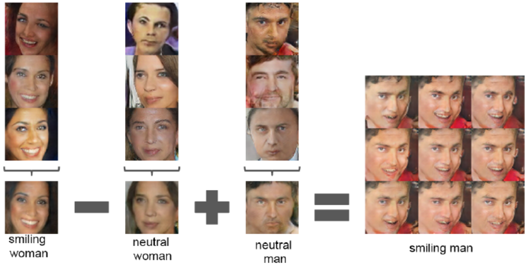
Pros: State-of-the-art results, difficult to quantify the quality of the results.
Cons: Difficult to train, cannot produce the explicit density.


Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

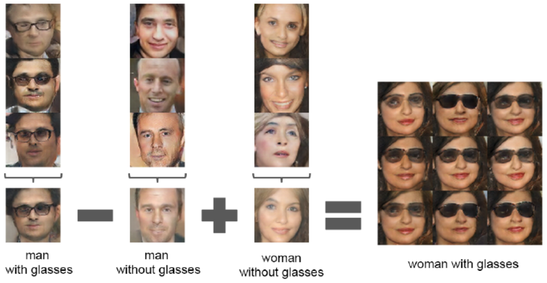
[Radford et al. 2016]

Latent space arithmetic



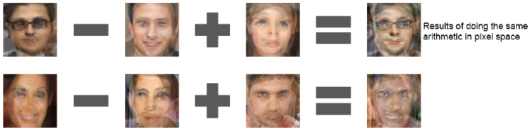
[Radford et al. 2016]

Latent space arithmetic



[Radford et al. 2016]

Comparison: pixel space arithmetic

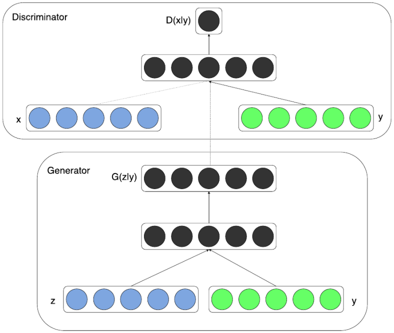


[Radford et al. 2016]

Conditional GANs

cGAN idea

Condition G and D on some additional variable y. Feed y to both G and D.

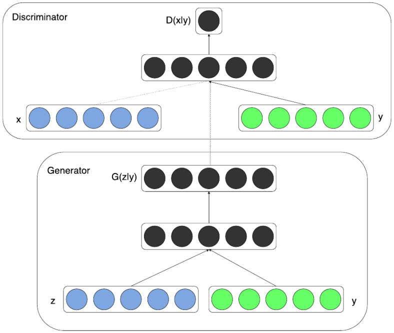


[Mirza et al. 2014]

Conditional GANs

cGAN idea

Condition G and D on some additional variable y. Feed y to both G and D.

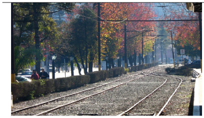
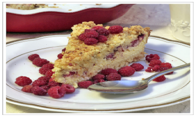
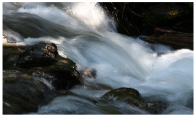
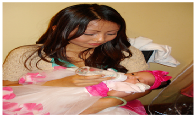


[Mirza et al. 2014]

Objective Function

$$\min_G \max_D \mathrm{E}_{x\sim p_{data}}[\log D(x|y)] + \mathrm{E}_{z\sim p_{prior}}[(1 - \log D(G(z)|y))]$$

Results of conditional GAN

	User tags + annotations	Generated tags
	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

[Mirza et al. 2014]

Conditioning on images

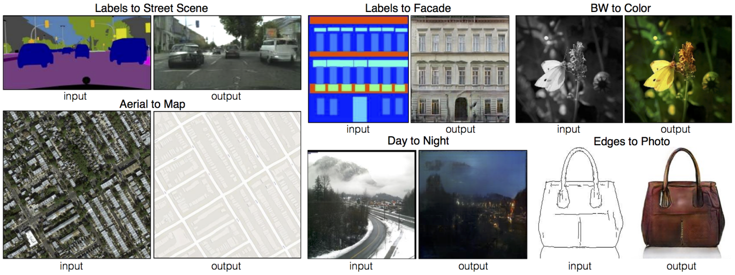
Image-to-Image Translation [Isola et al. 2017]

y is now an image we want to transform (sketch to object, day to night, B/W to color...). Other formulation:

$$\min_G \max_D \mathrm{E}_{(x,y)\sim p_{data}}[\log D(x,y)] + \mathrm{E}_{y\sim p_{data}, z\sim p_{prior}}[(1-\log D(G(z,y)|y))]$$
$$+\lambda\, \mathrm{E}_{x,y,z}[\|x - G(z,y)\|_1]$$

- Additional term favors resemblence to true result and produces better results [Pathak et al. 2014]

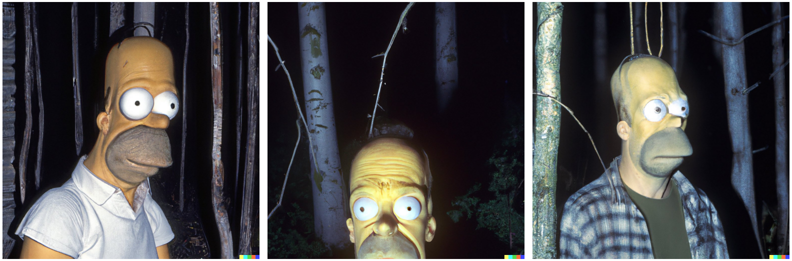
Conditioning on images



[Isola et al. 2017]

Outline

Denoising Diffusion for Image synthesis: **Dall-E 2**



A still of Homer Simpson in The Blair Witch Project

Growing field

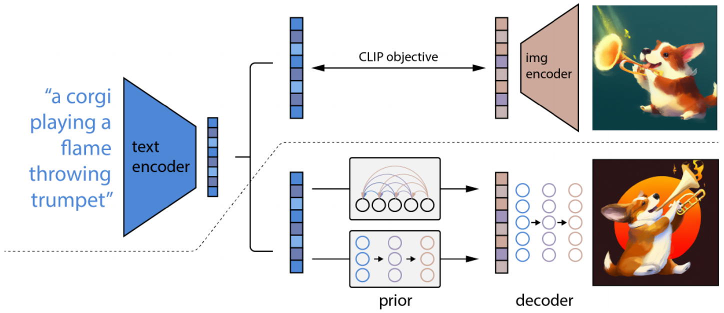
Dall-E 2 [Ramesh et al. 2022] - $\sim$ April; Stable Diffusion [Rombach et al. 2022] $\sim$ September (but also: Imagen...).
Dates back to: [Sohl-Dickstein et al. 2015] [Ho et al. 2020]

Example of Midjourner Result (Sept 2023)

a group of crazy french students attending a lecture on artificial intelligence

a crazy professor teaching artificial intelligence to his students in France ; surrealist

Principle

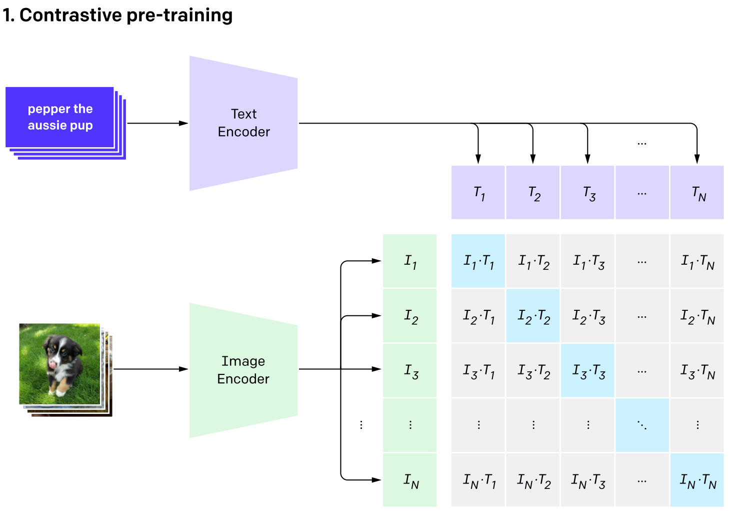


[Ramesh et al. 2022]

2 stages:

- Learn a CLIP (text+image) embedding for a caption
- Generate an image from the image embedding

CLIP [Radford et al. 2021]



1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 … T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ … $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ … $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ … $I_3 \cdot T_N$
⋮ ⋮ ⋮ ⋱ ⋮
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ … $I_N \cdot T_N$

[Radford et al. 2021]

Learns which caption goes with which image.

Back to Dall-E

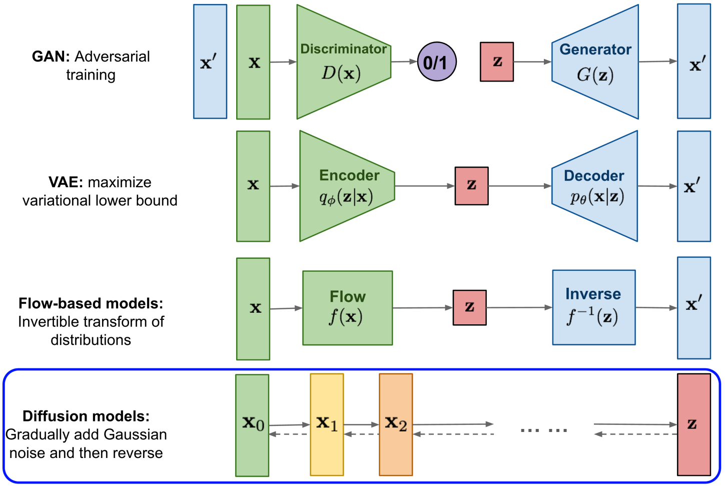
To build $P(x|y)$

- Learns a prior $P(z_i|y)$ that produces CLIP image embeddings z_i conditioned on captions y.
- Learns a decoder $P(x|z_i)$ or $P(x|z_i, y)$

Key Ingredient

Diffusion-based data generation

Diffusion-based data generation



GAN: Adversarial training
x'
x
Discriminator $D(x)$
0/1
z
Generator $G(z)$
x'
VAE: maximize variational lower bound
x
Encoder $q_\phi(z|x)$
z
Decoder $p_\theta(x|z)$
x'
Flow-based models: Invertible transform of distributions
x
Flow $f(x)$
z
Inverse $f^{-1}(z)$
x'
Diffusion models: Gradually add Gaussian noise and then reverse
x_0
x_1
x_2
... ...
z

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Blur an image until you get a noisy image, learn the reverse process

Diffusion Process

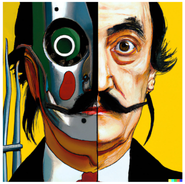
- Given $x_O \sim q(x_0)$, generate a Markov chain by adding noise $p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \sqrt{\alpha_t})I)$
- If the noise is large enough x_T can be sampled using $\mathcal{N}(0, I)$
- Iteratively remove the noise by learning a model $\mathcal{N}(\mu(x_t), \Sigma(x_t))$ approximating the true posterior $p(x_{t-1}|x_t)$
- Better: predict the added noise minimizing

$$L_{simple} = \mathbb{E}_{t\sim[0,T],x_{\mathbf{0}}\sim q(x_{\mathbf{0}}),\varepsilon\sim\mathcal{N}(0,I)}[\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2]$$

Some more details

- Generate 64x64 images
- Upsampling through two Diffusion-based upsampler models (256x256, 1024x1024).

Results


vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

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

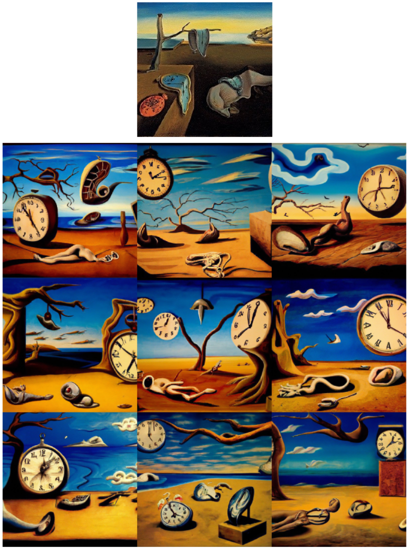
a dolphin in an astronaut suit on saturn, artstation

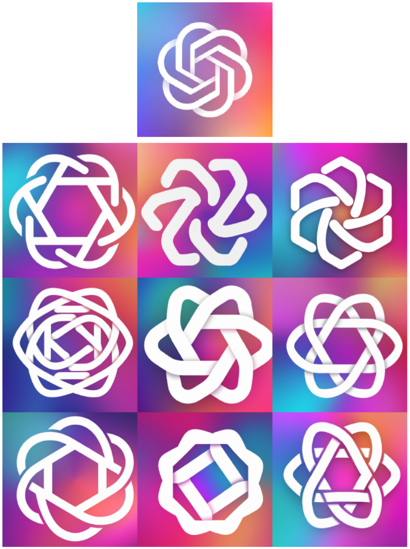
a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

a teddy bear on a skateboard in times square

[Ramesh et al. 2022]

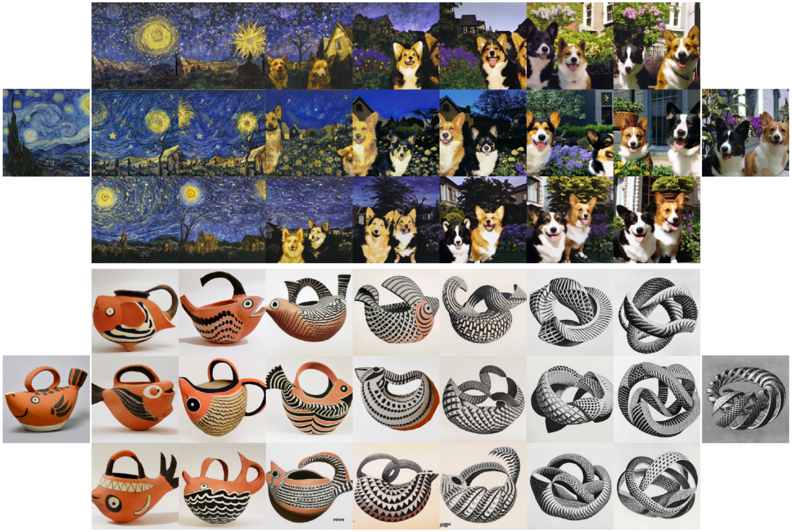
Decoding an input image





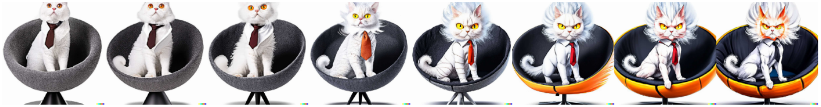
[Ramesh et al. 2022]

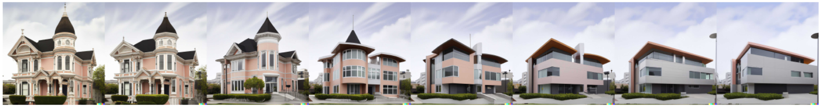
Interpolation

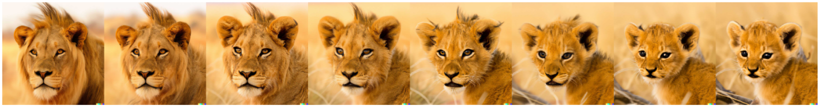


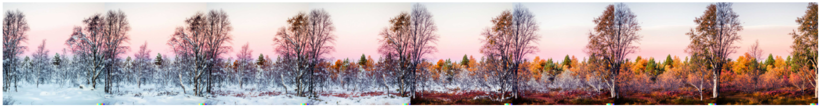
[Ramesh et al. 2022]

Text differences


a photo of a cat $\longrightarrow$ an anime drawing of a super saiyan cat, artstation


a photo of a victorian house $\longrightarrow$ a photo of a modern house


a photo of an adult lion $\longrightarrow$ a photo of lion cub


a photo of a landscape in winter $\longrightarrow$ a photo of a landscape in fall

[Ramesh et al. 2022]

Some reading

- The Elements of Statistical Learning, Trevor Hastie, Robert Tibshirani, Jerome Friedman, Springer, 2009.
- Sparse Modeling for Image and Vision Processing, Mairal, Bach, Ponce, Foundations and Trends in Computer Graphics and Vision, 2014.
- Deep Learning, Goodfellow et al., MIT Press, 2016. `http://www.deeplearningbook.org/`
- Deep Learning, a visual approach, Andrew Glassner, 2021.

Outline

A different architecture: Transformers [Vaswani 2017]

- State of the art technique in Natural Language Processing
- Extended to Vision
- Extended to Geometry
- Extended to multi-modalities

A different architecture: Transformers [Vaswani 2017]

- State of the art technique in Natural Language Processing
- Extended to Vision
- Extended to Geometry
- Extended to multi-modalities

A new and non local way to encode information

Build a summary representation (output) of a set of signals (values) relative to a specific signal (query).

Attention

- The output o is a weighted sum of the values v_i, weights depending on the query q

$$o = \sum_i w_i v_i \text{ and } w_i = softmax(q^T v_i)$$

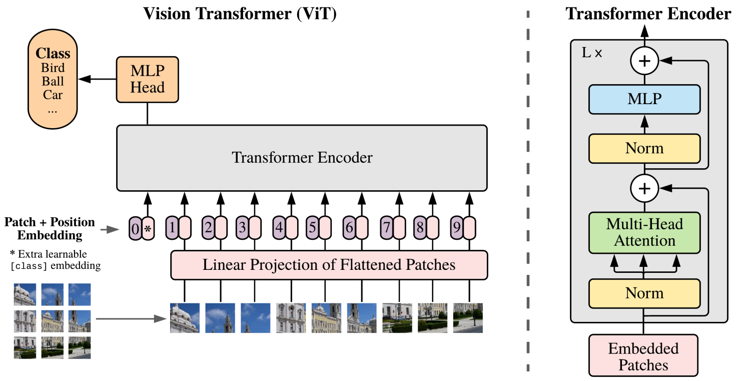
- More generic differentiate between *values* to aggregate and *keys* to compare:

$$o = \sum_i w_i v_i \text{ and } w_i = softmax(q^T k_i)$$

- Self-attention: Q=V
- Transformer [vaswani 2017] based on encoding and decoding (+ multihead attention + positional encoding).

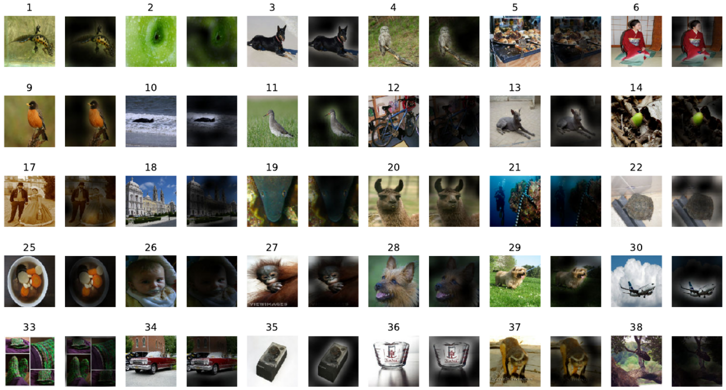
Looks familiar?

Vision Transformer



[Dosovitskyi 2017]

Vision Transformer - attentions



[Dosovitskyi 2017]