Deep Learning for Vision (DLV)

Generative models

Denis Coquenet

2024-2025

Université de Rennes

istic Informatique Électronique

Knowledge

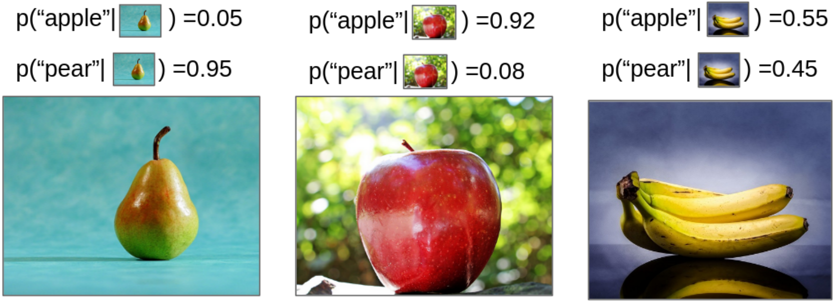
- Key principles of GAN and diffusion models
- Advantages/drawbacks of both approaches
- Be aware of ethical issues
- Limitations of the evaluation approaches

Skills and know-how

- Distinguish discriminative and generative tasks
- Use off-the-shelf models (practical session)

Discriminative model

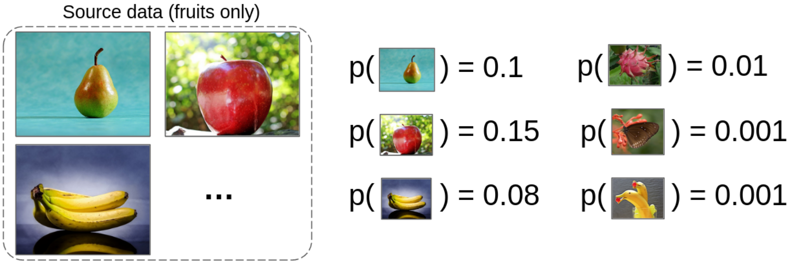
Learn a probability distribution $p(c|\boldsymbol{x})$ of a given set of classes $c \in \mathcal{C}$
$=$ what is the probability that image $\boldsymbol{x}$ belongs to class c
$=$ competition between classes



➤ Impossible to handle unknown classes

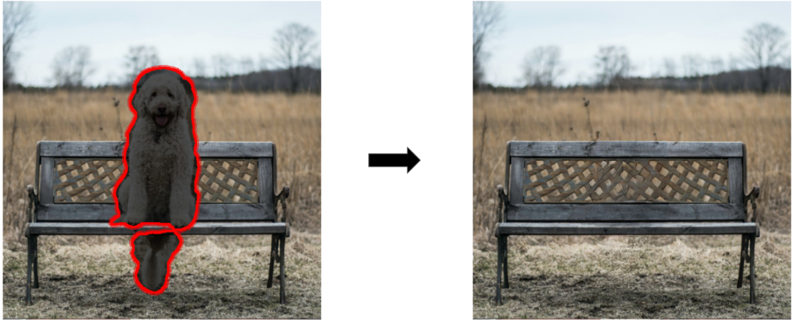
Generative model

Learn a probability distribution of the images $p(x)$
$=$ what is the probability that image x belongs to the distribution ?
$=$ competition between all images



➤ Needs a high-level image understanding

- Image generation from scratch
- Style transfering
- Text-to-image generation
- Image edition



Generative tasks

Why?

- Tools for artists / designers
- Image upscaling
- Social networks (face swapping, filters)

Challenges

- The generated images must be various but coherent
- Images must reflect the user's wishes
- Images can be of several nature: photorealistic, cartoon, painting

Hard to evaluate

What is a good generated image?

➤ Depends on the goal

General goals:

- Realism/Creative (given context)
- Diversity

Approaches

- Human evaluation (subjective, costly, biased)
- Automatic evaluation (limited by model capacity)
  - Task-driven (*e.g.*, result of classification model)
  - Distribution comparison between real/generated images

Goal

- Compute the distance between two distributions $X \sim \mathcal{N}(\mu_X, \sigma_X)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y)$

Fréchet distance

$$d(X,Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

$$d(X,Y) = 0 \Leftrightarrow \mu_X = \mu_Y \text{ and } \sigma_x = \sigma_y$$

- The lower the better

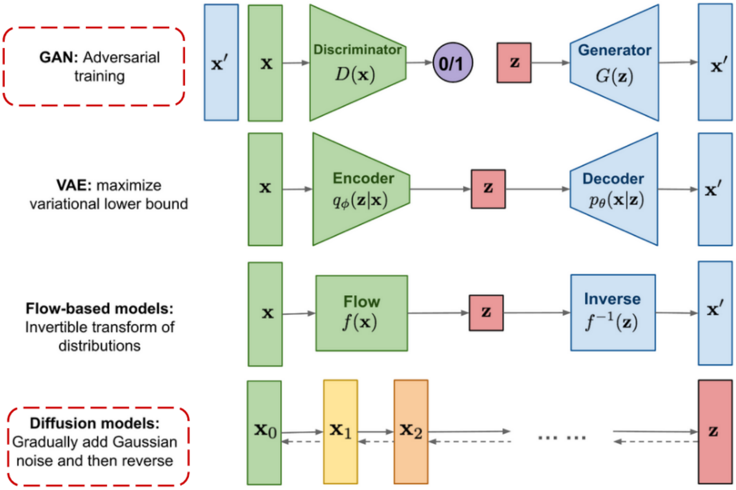
Fréchet Inception Distance (FID)

Idea: compute distance between distributions of real and generated images
X: a set of real images
Y: a set of generated images
The distance is computed in the feature space using an Inception model pre-trained on ImageNet, without the classification layer (vector of 2048)

$$\mathsf{FID}(X, Y) = ||\mu_X - \mu_Y||_2^2 + \mathsf{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{\frac{1}{2}})$$

with μ_X, μ_Y the means, Σ_X, Σ_Y the covariance matrices and Tr the trace function (sum of diagonal values)

- Requires enough data to be representative (>10,000)
- Can be long to compute



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

Table of contents

GAN (2014) [1]

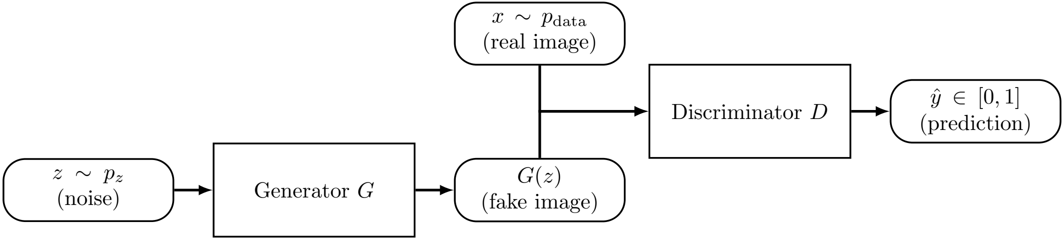
- Generative Adversarial Networks

Idea

- Generate artificial images that look like a target domain
- A noise-to-image process to generate many different images

How

- Unsupervised representation learning
- Capture data distribution through discrimination between real/generated data



A minimax two-player game approach

A generative model G:

- Generate samples as plausible as possible (w.r.t. the problem domain)

A discriminative model D:

- Classify samples as real (1=from domain) or fake (0=generated by D)

G tries to fool D, and D tries not to be fooled

GAN (2014) [1]

Discriminator objective

➤ Maximize classification between real and generated examples

$$\max_{\theta_d} [\mathbb{E}_{x \sim p_{\textsf{data}}(x)} \log \underbrace{D_{\theta_d}(x)}_{\substack{\text{descriminator output}\\ \text{for real examples}}} + \mathbb{E}_{z \sim p_z(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z)))}_{\substack{\text{descriminator output}\\ \text{for generated examples}}}]$$

$D_{\theta_d}(x) \to 1$ and $D_{\theta_d}(G_{\theta_g}(z)) \to 0$

Generator objective

➤ Minimize classification performance = improve generation

$$\min_{\theta_g} [\mathbb{E}_{z \sim p_z(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

$D_{\theta_d}(G_{\theta_g}(z)) \to 1$

GAN (2014) [1]

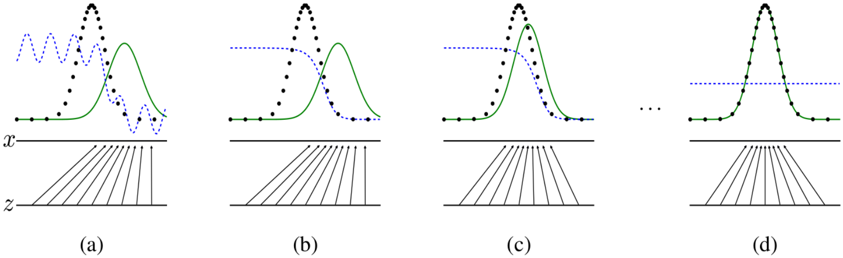
Global objective function

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{\textsf{data}}(x)} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

- Opposite goals

Training approach

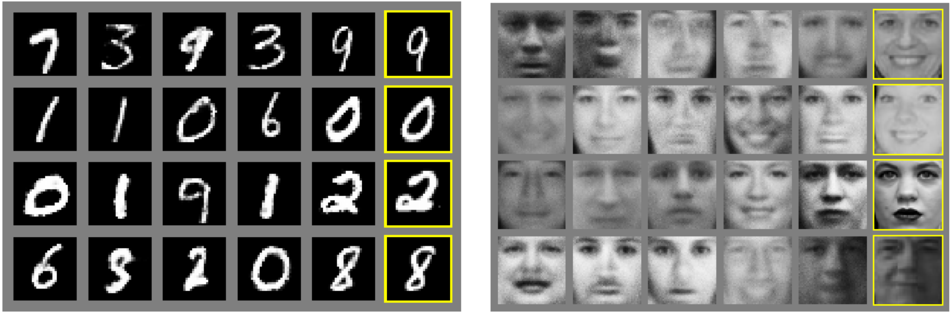
- Alternate training between generator and discriminator
- Generator and discriminator implemented as MLP



Domain (black), discriminative (blue), generative (green) distributions

(a) D partially accurate, G differs from domain distribution
(b) D is further trained
(c) G is trained to fool D, which is frozen
➤ (b) and (c) repeated until:
(d) Cannot improve more: $G(\text{z}) = p_{\text{data}}$ and $D(x) = D(G(z)) = \frac{1}{2}$.

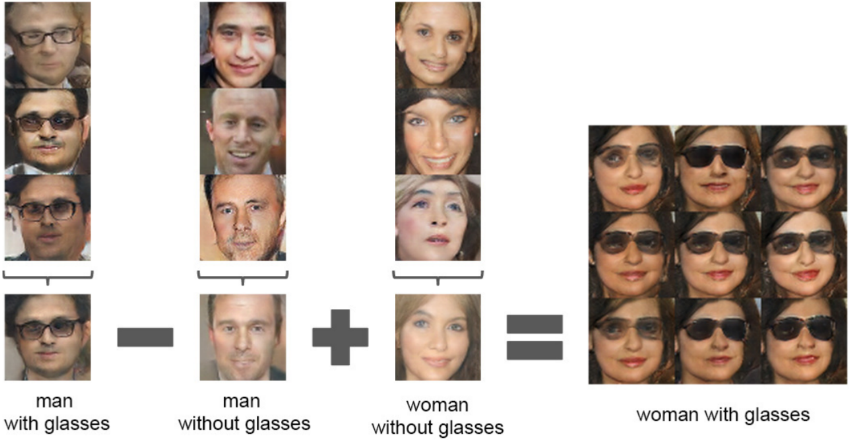
- Predictions



Real samples are framed in yellow

- Hard to train
- Noisy predictions

- Arithmetic properties on noise space

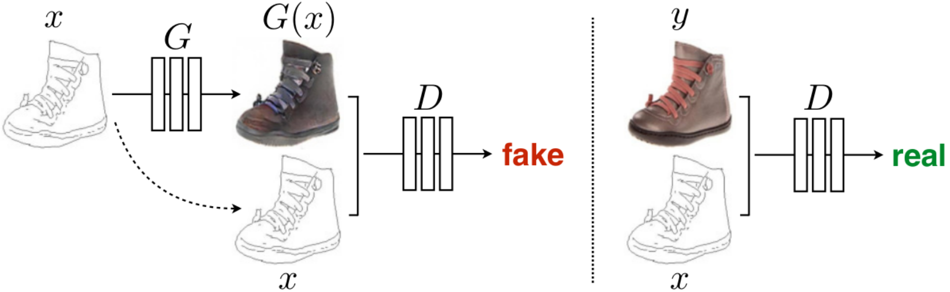


Input vectors are averaged for three examples

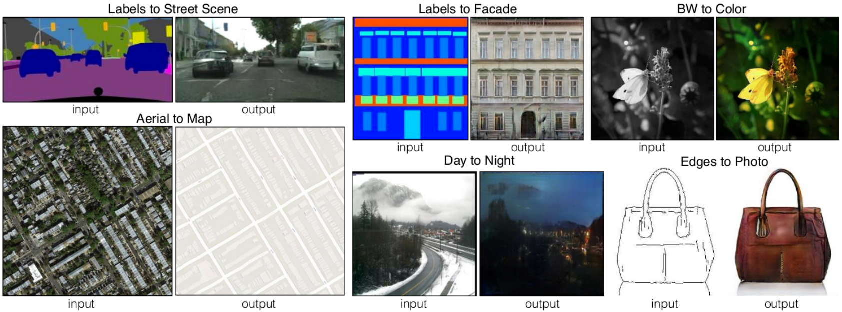
Deep Convolutional GAN (2015) [2]

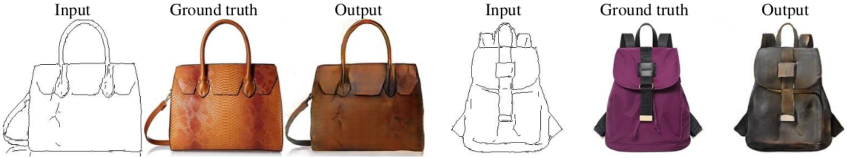
A turn vector is computed from faces turning right or left

➤ Condition the generation process with an input image
Noise modeled as dropout in generative model
$\rightarrow$ weak modifications for same input



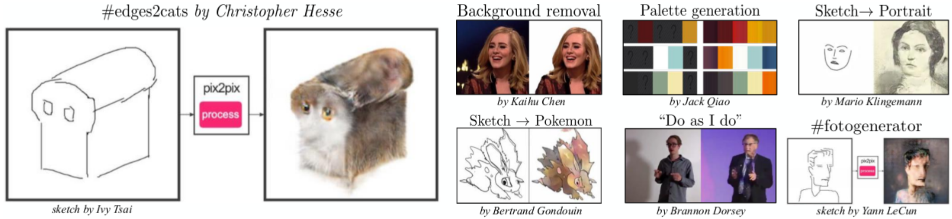
- ➤ Can handle several image-to-image translation tasks





- Day to night

- Examples from the community



- Upsample (x4) low-resolution image

Y = original image (high resolution), X = degraded image

Which one is the original image ?

Metrics

Peak Signal-to-Noise (PSNR)

$$\text{PSNR} = 20\log_{10}\left(\frac{\max(y)}{\sqrt{\text{MSE}(x,y)}}\right)$$

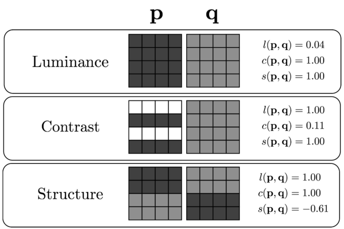
Structural Similarity Index Measure (SSIM)

$$\text{SSIM}(x,y) = l(x,y)^{\alpha}c(x,y)^{\beta}s(x,y)^{\gamma}$$

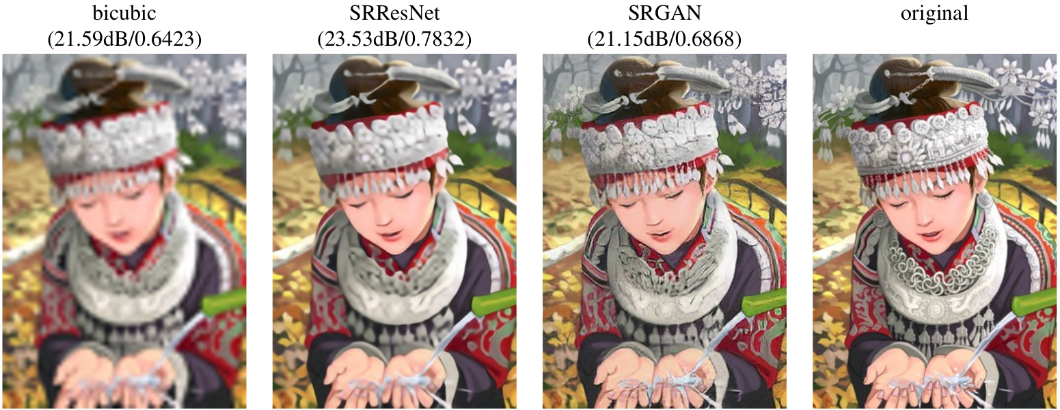
Luminance: $l(x,y) = \dfrac{2\mu_x\mu_y + c_l}{\mu_x^2 + \mu_y^2 + c_l}$

Contrast: $c(x,y) = \dfrac{2\sigma_x\sigma_y + c_c}{\sigma_x^2 + \sigma_y^2 + c_c}$

Structure: $s(x,y) = \dfrac{2\sigma_{xy} + c_s}{\sigma_x + \sigma_y + c_s}$



- Be careful with metrics!

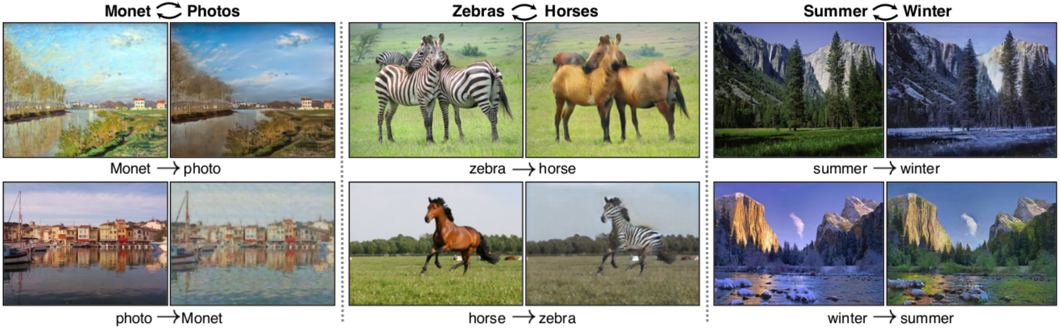


Bicubic interpolation vs MSE-based training vs GAN-based training (PSNR/SSIM)

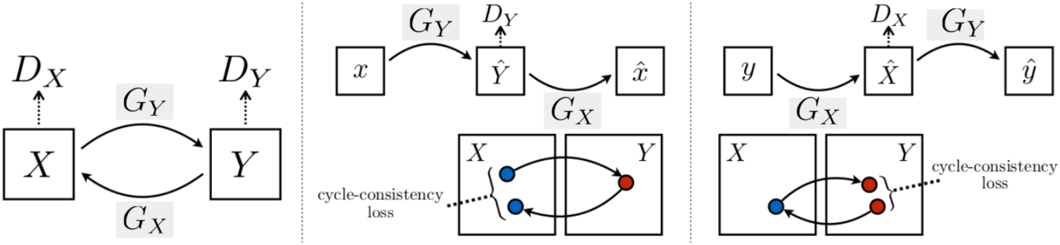
"PSNR and SSIM fail to capture image quality with respect to human the visual system"

Cycle GAN (2017) [5]

- Translation between two domains, in both directions



Cycle GAN (2017) [5]



Two domains: X, Y
Two generators: $G_X : Y \to X$, $G_Y : X \to Y$
Two discriminators: $D_X : X \to [0, 1]$, $D_Y : Y \to [0, 1]$

➤ No need for paired data: only two sets of unlabeled data

Cycle GAN (2017) [5]

GAN loss for $X \to Y$

$$\mathcal{L}_{\mathsf{GAN}}(G_Y, D_Y, X, Y) = \mathbb{E}_{y\sim p_{\mathsf{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x\sim p_{\mathsf{data}}(x)}[\log(1 - D_Y(G_Y(x)))]$$

Cycle consistency loss

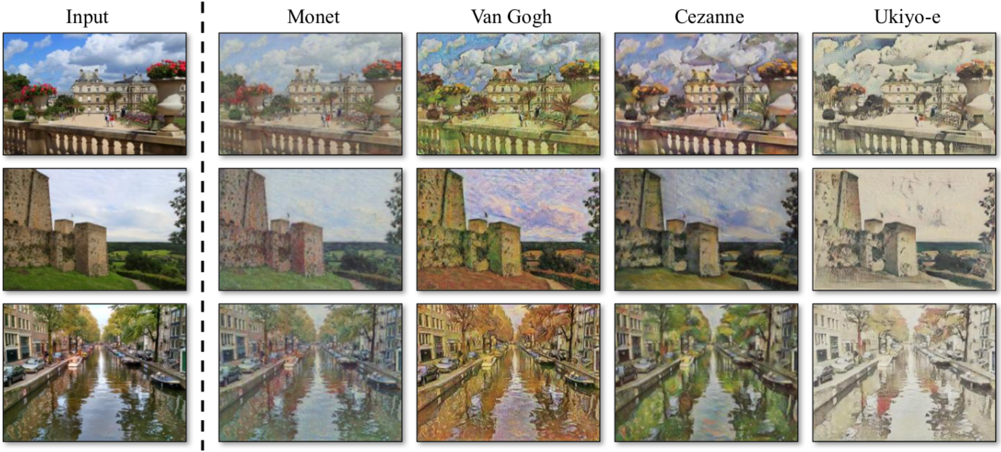
$$\mathcal{L}_{\mathsf{cycle}}(G_X, G_Y) = \mathbb{E}_{x\sim p_{\mathsf{data}}(x)}[||G_X(G_Y(x)) - x||_1] + \mathbb{E}_{y\sim p_{\mathsf{data}}(y)}[||G_Y(G_X(y)) - y||_1]$$

Global loss

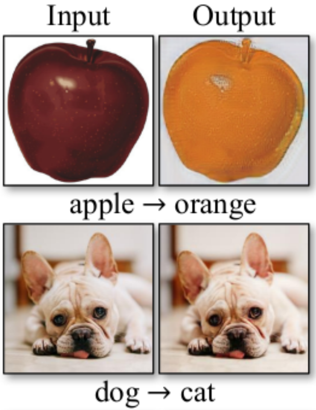
$$\mathcal{L} = \mathcal{L}_{\mathsf{GAN}}(G_Y, D_Y, X, Y) + \mathcal{L}_{\mathsf{GAN}}(G_X, D_X, Y, X) + \mathcal{L}_{\mathsf{cycle}}(G_X, G_Y)$$

Cycle GAN (2017) [5]

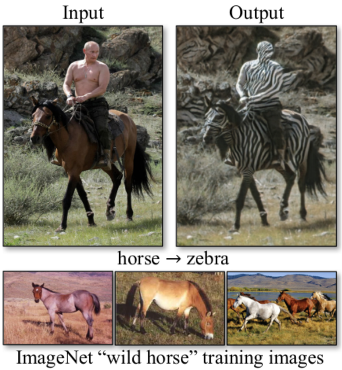
➤ Works well for texture/color changes



- Failure cases:

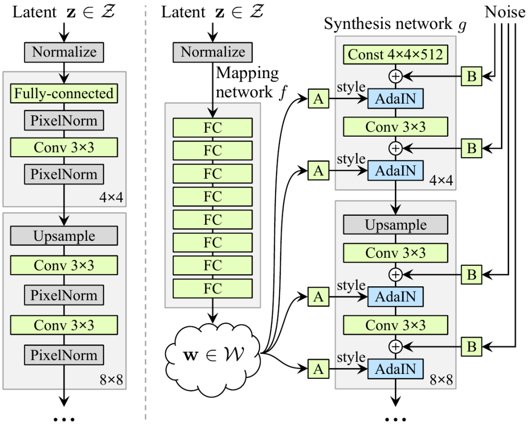


Geometric modifications



Out-of-domain data

- Enable control of the synthesis
- Include stochastic variation



(a) Traditional (b) Style-based generator

Style GAN (2019) [6]

- Use mapping network for disentanglement

> "There are various definitions for disentanglement, but a common goal is a latent space that consists of linear subspaces, each of which controls one factor of variation."

Adaptive Instance Normalization

- Representation w is used to extract several styles $y = (y_s, y_b)$

$$\text{AdaIN}(x_i, y) = y_{s,i} \frac{x_i - \mu_{x_i}}{\sigma_{x_i}} + y_{b,i}$$

Styles are applied per channel i, on whole 2D latent representations

➤ Mixing styles from two samples (z_A, z_B)
(using w_B for specific layers and w_A for the others)

From B:

Coarse
(4^2 - 8^2)

Middle
(16^2 - 32^2)

Fine
(64^2 - 1024^2)

→ pose, eyeglasses

→ hairstyle

→ color scheme

➤ Noise is added at each layer at pixel level, enabling low-level variation while preserving style

(a) Generated image (b) Stochastic variation (c) Standard deviation

GAN: conclusion

- GAN: a revolution in generative models (quality, resolution) but hard to train

Vanishing gradient

Discriminator too good: loss becomes very low, gradient too: no feedback for generator

Convergence issues

Generator sufficiently good (discrimator: 50% accuracy): cannot improve more, receive junk feedback from discriminator

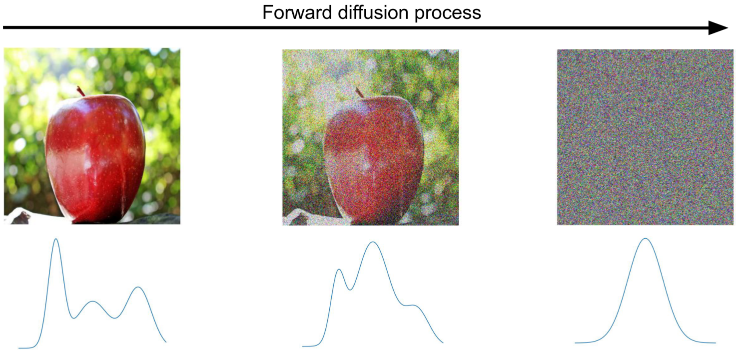
Mode collapse

The generator find an example which fool very well the discriminator and start producing always the same outputs, while discriminator stuck in local minima

Table of contents

Diffusion [7]

"The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an **iterative forward diffusion process**. We then **learn a reverse diffusion process** that restores structure in data, yielding a highly flexible and tractable generative model of the data"



Goal: from a complex, unknown data distribution to a gaussian distribution

Let $\boldsymbol{x}_0 \sim q(x)$ be the input image from the real data distribution and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ a gaussian noise.
We want $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ after t iterations

Naive iterative noising process

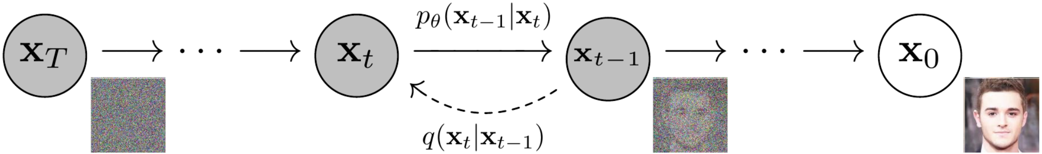
$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_{t-1}$$

$X \sim \mathcal{N}(\mu, \sigma^2) \Leftrightarrow X = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$
$\implies \boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_{t-1}, \boldsymbol{I})$
$\implies \boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_0, t\boldsymbol{I})$

Issues

Var$(\boldsymbol{x}_t)$ keeps increasing with t
Mean remains the same

Denoising Diffusion Probabilistic Models (DDPM, 2020) [8]



Forward diffusion process: Markov chain

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}$$

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \mu = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \sigma^2 = \beta_t \boldsymbol{I})$$

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

β_t: variance schedule $(0 < \beta_t < 1)$

➤ A step-by-step process from original image $\boldsymbol{x}_0$ to pure noise $\boldsymbol{x}_T$

Property

If $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ and $Z = X + Y$
then $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$

By defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \underbrace{\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}}_{\sim\mathcal{N}(0,\alpha_t(1-\alpha_{t-1}))} + \underbrace{\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}}_{\sim\mathcal{N}(0,1-\alpha_t)} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned}
$$

$\bar{\alpha}_t$ can be precomputed $\rightarrow$ $\boldsymbol{x}_t$ can be calculated for any t directly.

Forward process properties

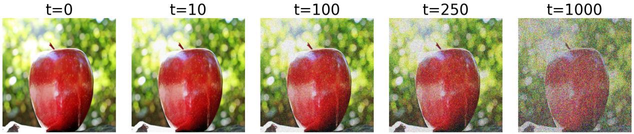
Recall

$\boldsymbol{x}_t \sim \mathcal{N}(\sqrt{\bar{\alpha}}\boldsymbol{x}_0, \sqrt{1-\bar{\alpha}}\boldsymbol{I})$
$\alpha_t = 1 - \beta_t$
$\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

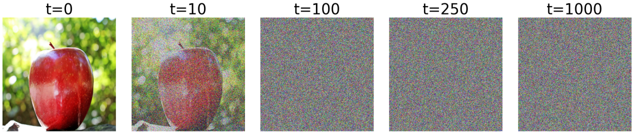
Converge to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Choosing $0 < \beta_t < 1$
$\Rightarrow 0 < \alpha_t < 1$
$\Rightarrow \lim\limits_{t \to +\infty} \bar{\alpha}_t = 0$
$\Rightarrow \boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if T high enough

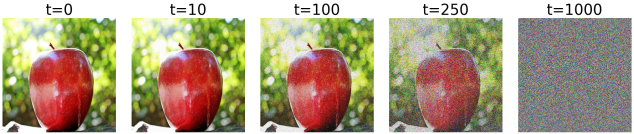
Choice of β_t



Too low ($\beta_t = 0.001 \; \forall t$): too many iterations



Too high ($\beta_t = 0.1 \; \forall t$): too much noise, difficult to learn transition



Linear from $\beta_0 = 0.0001$ to $\beta_T = 0.2$

- From noise to image

Initial state

$p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \boldsymbol{I})$

We need $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to reverse the process, but intractable!

- Learn it with neural networks instead: $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$

Iterative denoising process

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_\theta^2(\boldsymbol{x}_t, t))$$

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$$

Loss function

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}_0,t,\boldsymbol{\epsilon}}[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}_{\boldsymbol{x}_t}, t)||^2]$$

- Difference between added noise $\boldsymbol{\epsilon}$ and predicted noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$
- $\boldsymbol{\epsilon}_\theta$ implemented as U-Net

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Sampling

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_\theta^2(\boldsymbol{x}_t, t))$$

$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t))$

$\sigma_\theta^2(\boldsymbol{x}_t, t) = \sigma_t^2 = \beta_t$

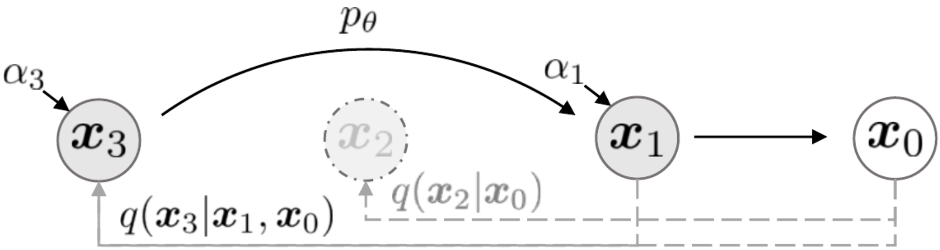
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

- Long sampling time: T steps to generate an image

Idea

Use a non-markovian process to skip steps when sampling



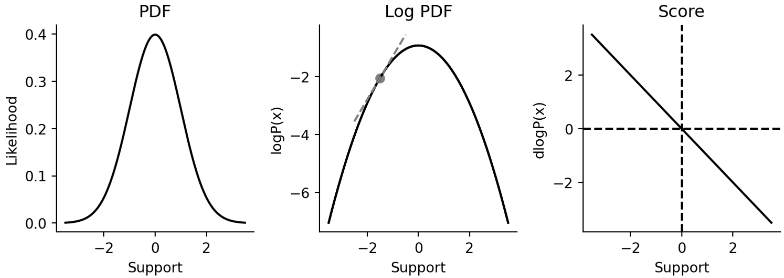
- 10-50 times faster

What is a score function?

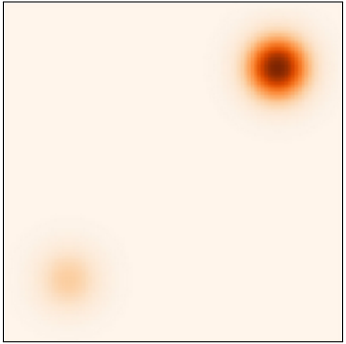
Let $p(x)$ be a Probability Density Function (PDF)
The score function of p is defined as:

$$s(x) = \nabla_x \log p(x)$$

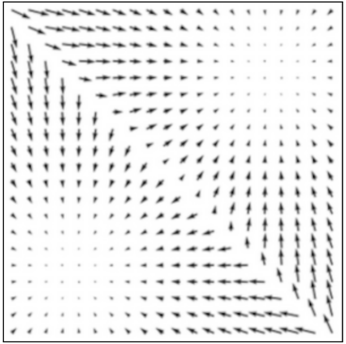
$=$ direction vector to maximize probability



2D example



Density



Score

Goal

Approximate the score function with a neural network $s_\theta(x)$

$$\min_\theta \mathbb{E}_{p(x)}[||s(x) - s_\theta(x)||_2^2]$$

- But s(x) is unknown!

Equivalence

$$\min_\theta \mathbb{E}_{p(x)}\left[\mathrm{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}||s_\theta(x)||_2^2)\right]$$

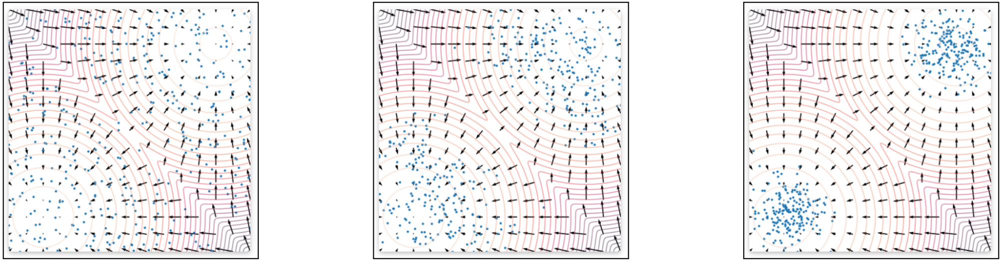
Only depends on $s_\theta(x)$

Sampling (denoising) with Langevin dynamics

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t + \epsilon s_\theta(\boldsymbol{x}_t) + \sqrt{2\epsilon}\boldsymbol{z}_{t-1}$$

$\boldsymbol{z}_t \sim \mathcal{N}(0, 1)$
ϵ: a fixed step size

➤ Repeat T times from $x_T \sim \mathcal{N}(0, 1)$ to x_0



x_T x_t x_0

Goal

Guiding the generation process

Conditioned generation

Add instruction as input to generate a specific item

Guiding the generation process

Use $\nabla_x \log p(x|y)$ instead of $\nabla_x \log p(x)$
where y is an additional input which specifies what we want to generate

How to compute $\nabla_x \log p(x|y)$?

Bayes' rule

$$p(x|y) = \frac{p(y|x) \cdot p(x)}{p(y)}$$

$$\log p(x|y) = \log p(y|x) + \log p(x) - \underbrace{\log p(y)}_{\nabla_x \log p(y) = 0}$$

$$\nabla_x \log p(x|y) = \nabla_x \log p(y|x) + \nabla_x \log p(x)$$

- $\nabla_x \log p(y|x)$ can be obtained using a classifier

Classification task

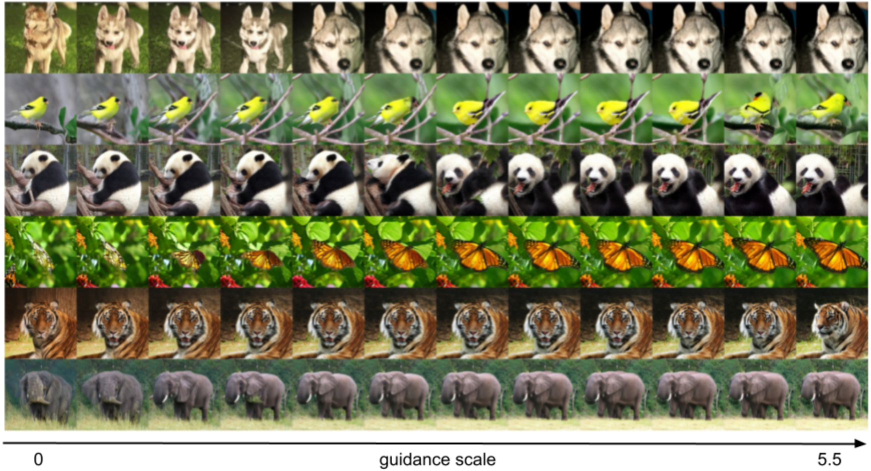
Learn $p_\theta(y|x)$
x: input image
y: class

Tuning the guidance impact

Introduction of a guidance scale γ:

$$\nabla_x \log p_\gamma(x|y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y|x)$$

- Needs to train a classifer on noisy images


0
guidance scale
5.5

Classifier-free guidance

$$\nabla_x \log p_\gamma(x|y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y|x)$$

Bayes' rule

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

$$\log p(y|x) = \log p(x|y) + \underbrace{\log p(y)}_{\nabla_x \log p(y) = 0} - \log p(x)$$

$$\nabla_x \log p(y|x) = \nabla_x \log p(x|y) - \nabla_x \log p(x)$$

$$\Rightarrow \nabla_x \log p_\gamma(x|y) = (1-\gamma) \underbrace{\nabla_x \log p(x)}_{\text{unconditional}} + \gamma \underbrace{\nabla_x \log p(x|y)}_{\text{conditional}}$$

➤ can be jointly train with a single diffusion model by dropping-out the conditional term (10%-20% of the time)

➤ Guided Language to Image Diffusion for generation and Editing (GLIDE)

Compare (classifier-free) text guidance:

$$\nabla_x \log p_\gamma(x|y) = (1-\gamma)\nabla_x \log p(x) + \gamma \nabla_x \log p(x|y)$$

with CLIP guidance:

$$\nabla_x \log p_\gamma(x|y) = \nabla_x \log p(x) + \gamma \nabla_x \left(f(x) \cdot g(y)\right)$$

+ Super Resolution

CLIP (2021) [13]

➤ Contrastive Language-Image Pre-training (CLIP)

Idea

Jointly train two encoders to avoid human annotation effort:

- An image encoder f: ResNet/ViT
- A text encoder g: transformer

➤ Both encoders are trained to generate a fixed-length latent representation from text/image input sharing the same feature space

How?

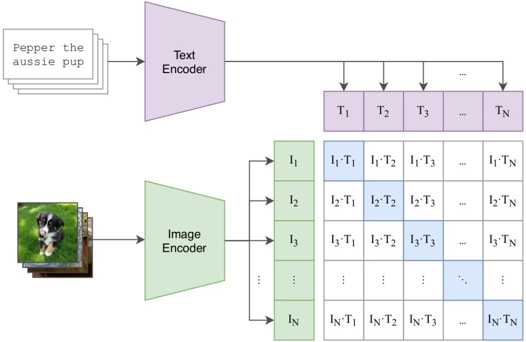
Constrative learning between images and captions

➤ 400 million pairs (image, text) collected from the web

x: image
y: caption

Image encoder: $f(x) = I$

Text encoder: $g(y) = T$



Symmetric loss

$$\mathcal{L} = \frac{1}{2}\Big(\sum_{i=1}^{N} \underbrace{\mathcal{L}_{\text{CE}}(\hat{y}_i^I, y_i^I)}_{\text{image } i \text{ compared to all texts}} + \sum_{t=1}^{N} \underbrace{\mathcal{L}_{\text{CE}}(\hat{y}_t^T, y_t^T)}_{\text{text } t \text{ compared to all images}}\Big)$$

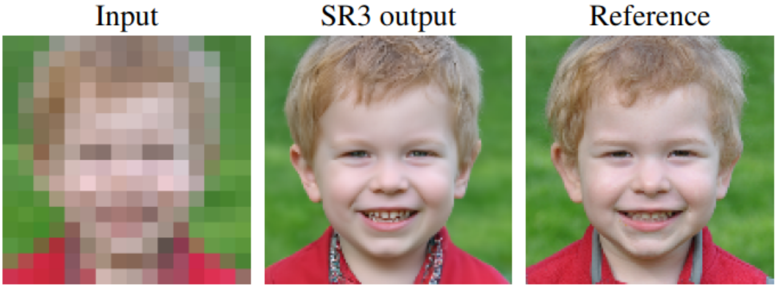
$$\hat{y}_{i,j}^I = I_i \cdot T_j$$

$$\hat{y}_{t,j}^T = I_j \cdot T_t$$

- Image-conditioned guidance

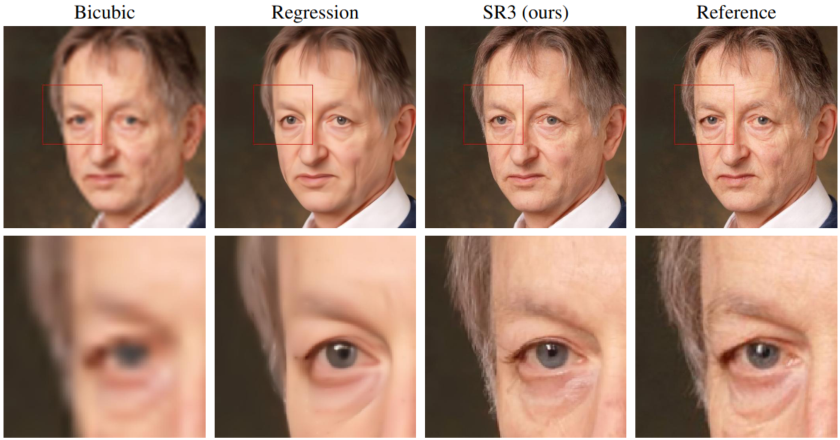
Diffusion-based upscaling task

Generate high-resolution image from noise, conditioned on low-resolution image



$16 \times 16 \rightarrow 128 \times 128$ pixels

➤ Comparison with other approaches



$64 \times 64 \rightarrow 512 \times 512$ pixels

GLIDE (2022) [12]

- Classifier-free guidance better than CLIP guidance

"a green train is coming down the tracks" | "a group of skiers are preparing to ski down a mountain." | "a small kitchen with a low ceiling" | "a group of elephants walking in muddy water." | "a living area with a television and a table"

- Image inpainting = image edition based on text and mask

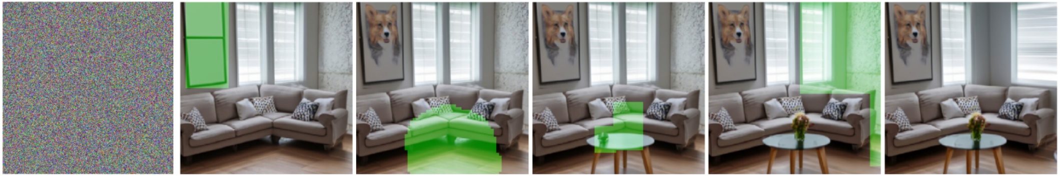
Image inpainting task

Generate image from noise, conditioned on text and masked original image
= text-guided and image-guided

"zebras roaming in the field"

"a girl hugging a corgi on a pedestal"

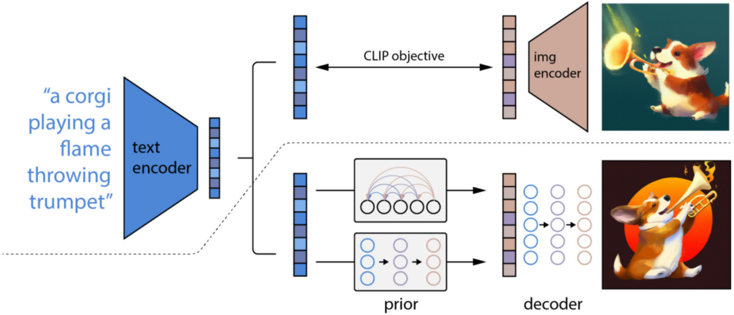
➤ Combining image generation and image inpainting



"a cozy living room" | "a painting of a corgi on the wall above a couch" | "a round coffee table in front of a couch" | "a vase of flowers on a coffee table" | "a couch in the corner of a room"

1) Generate first image from noise, conditioned by text
2) Update specific part of image from image, conditioned by masked image and text

DALL-E 2 (2022) [15]



- CLIP: align image/text representations
- Prior $P(z_i|y)$: produces CLIP image embeddings z_i conditioned on caption y = diffusion
- Decoder $P(x|z_i, y)$: produces image x conditioned on CLIP image embedding z_i (and optionally caption y) = diffusion

DALL-E 2 (2022) [15]

CLIP with image encoder f and text encoder g

Given an input couple (image x, caption y):

$z_i = f(x)$

$z_t = g(y)$

Prior p

Generate $\tilde{z}_i$ with diffusion model conditioned on:

- Transformer-encoded caption
- CLIP-encoded caption (optionally)

$$\mathcal{L}_{\text{prior}} = \mathbb{E}_{t,z_i}[||\tilde{z}_i^t - z_i||]$$

DALL-E 2 (2022) [15]

Decoder

Generate 64×64 image $\tilde{x}$ with diffusion model conditioned on:

- Prior output $\tilde{z}_i$ (z_i at training time)
- Caption

Both conditions are randomly dropped to boost performance

Upscaling

Two upscaling stages:

- $64 \times 64 \rightarrow 256 \times 256$ diffusion model
- $256 \times 256 \rightarrow 1024 \times 1024$ diffusion model

Text conditioning useless from experiments

- Examples

Input: "panda mad scientist mixing sparkling chemicals, artstation"

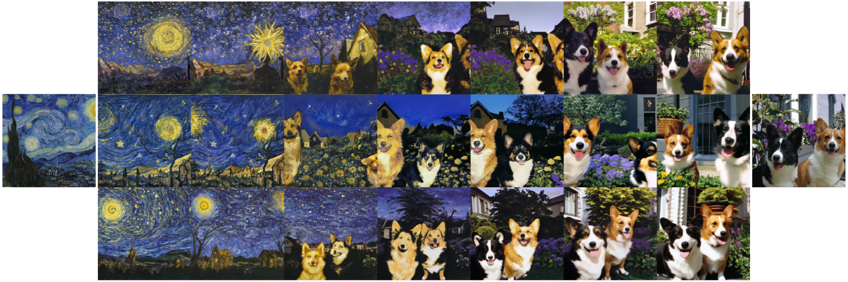
Input: "a dolphin in an astronaut suit on saturn, artstation"

➤ Diffusion model stochasticity

Input: image x

- Compute CLIP image embedding $z_i = g(x)$
- Decoder forward process with z_i

➤ Image interpolation



Input: images x_1 and x_2

- Compute CLIP image embedding $z_i^1 = g(x_1)$ and $z_i^2 = g(x_2)$
- Compute interpolation embedding z_i from z_i^1 and z_i^2
- Decoder forward process with z_i

➤ CLIP-based image edition trick

a photo of an adult lion → a photo of lion cub

a photo of a landscape in winter → a photo of a landscape in fall

Input: couple (image x, caption y) + goal caption y^*

- Compute difference vector z_d between CLIP-encoded texts $f(y)$ and $f(y^*)$
- Gradually modify CLIP image embedding $g(x)$ with respect to z_d
- Generate image from this altered image embedding

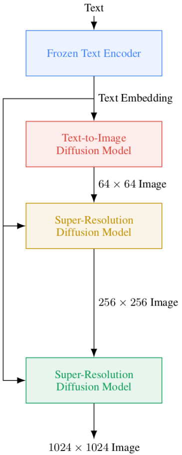
- Another text-to-image diffusion model

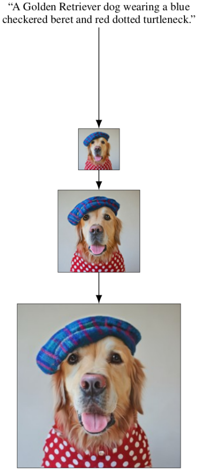
Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

Imagen (2022) [16]





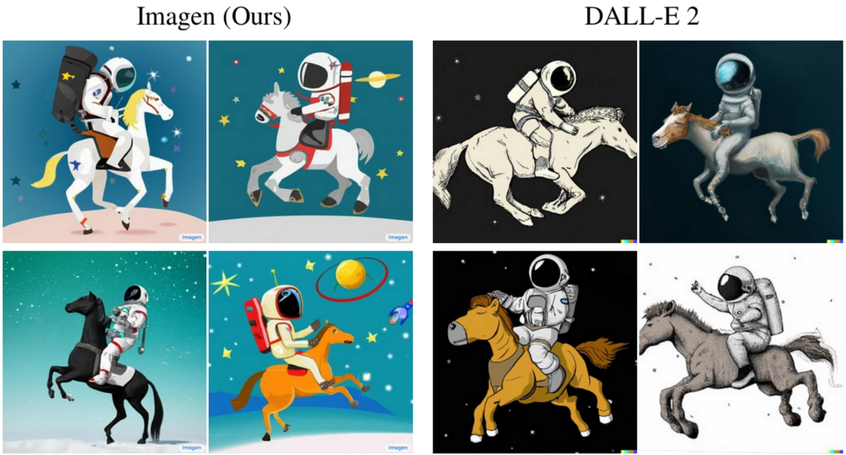
➤ Text-only encoder (T5) with very large dataset is better than image-text encoder (CLIP) with less data for text-to-image generation

➤ Scaling text encoder more efficient than scaling diffusion part

- Improvements for some cases

A black apple and a green backpack.

➤ Still some failure cases



A horse riding an astronaut.

Diffusion results comparison (from Imagen paper)

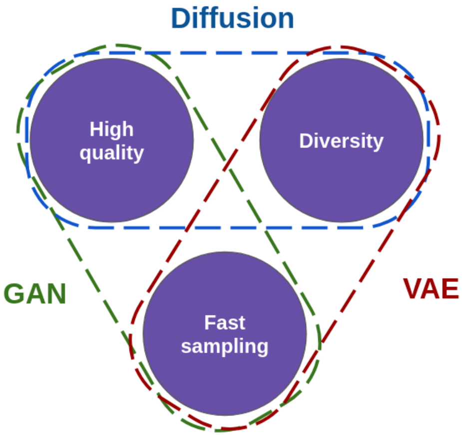
➤ Evaluation on the MS COCO validation set

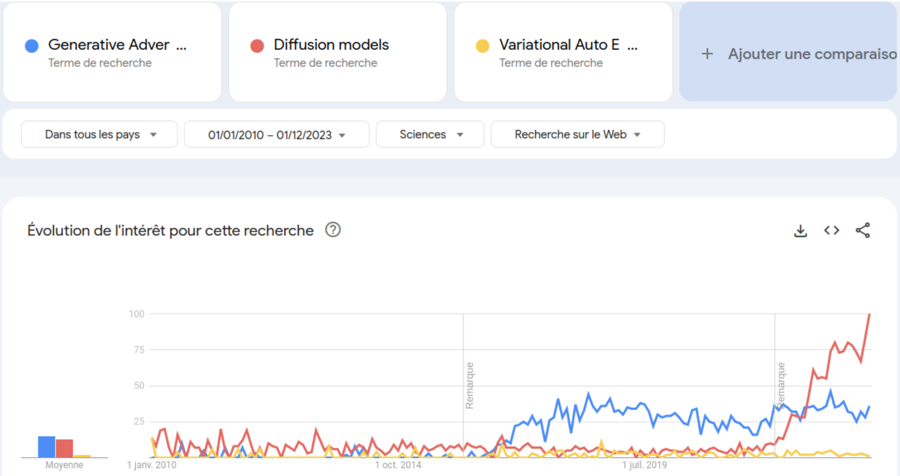
Model	**FID-30K**	**Zero-shot FID-30K**
AttnGAN [76]	35.49	
DM-GAN [83]	32.64	
DF-GAN [69]	21.42	
DM-GAN + CL [78]	20.79	
XMC-GAN [81]	9.33	
LAFITE [82]	8.12	
Make-A-Scene [22]	7.55	
DALL-E [53]		17.89
LAFITE [82]		26.94
GLIDE [41]		12.24
DALL-E 2 [54]		10.39
Imagen (Our Work)		**7.27**

Plenty of models

- GLIDE, DALL-E (OpenAI)
- Imagen (Google)
- CM3leon (Meta)
- MidJourney (Independent)
- Stable Diffusion (Stability AI, open source)

➤ What about training data? Intellectual property?


Diffusion
High quality
Diversity
GAN
VAE
Fast sampling



from www.trends.google.fr

Two approaches for a same goal

- GAN: generator vs discriminator
- Diffusion: denoising process

Realism

Diffusion models mark a new stage in the generation of ultra-realistic photos

➤ Adaptation to video
➤ Be careful with deepfakes
➤ The beginning of a new era in cinema?

➤ Next time: practical session!

References I

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. 2014, pp. 2672–2680.

[2] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations*. 2016.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5967–5976.

[4] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 105–114.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision*. 2017, pp. 2242–2251.

[6] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2019, pp. 4401–4410.

[7] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. JMLR Workshop and Conference Proceedings. 2015, pp. 2256–2265.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*. 2020.

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *9th International Conference on Learning Representations*. 2021.

[10] Yang Song and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 2019, pp. 11895–11907.

[11] Prafulla Dhariwal and Alexander Quinn Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*. 2021, pp. 8780–8794.

[12] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models". In: *International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 16784–16804.

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 8748–8763.

[14] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. "Image Super-Resolution via Iterative Refinement". In: *IEEE Transactions on Pattern Anaysis and Machine Intelligence* 45.4 (2023), pp. 4713–4726.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: (2022). URL: https://doi.org/10.48550/arXiv.2204.06125.

[16] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *NeurIPS*. 2022.