Generative Models: old and new.

Etat de l'art (11/05/2023)

Jean-Yves Franceschi, Mike Gartrell, and Ugo Tanielian.

CRITEO

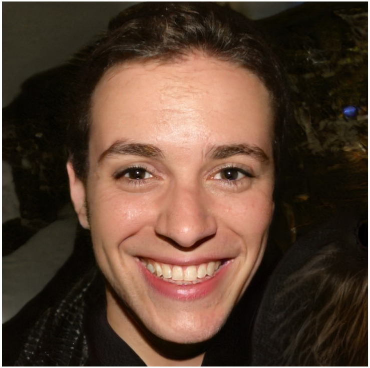
The Future is Wide Open

Outline

Outline

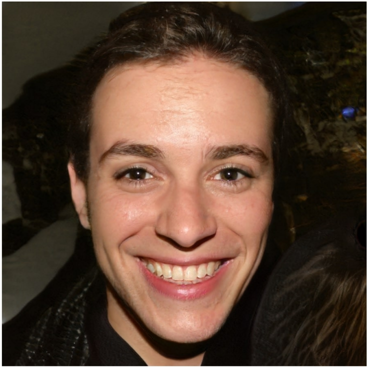
Outline

Let's Play a Game...




Which face is real?

Let's Play a Game...




Which face is real?
This one!

Generative Modeling

- A discriminative model is a way to model the conditional probability of a target Y (low-dimension) given some covariates X (high-dimension).
- Conversely, a generative model tries to model the conditional probability of X, possibly conditionally to Y.

Figure: Sampling from $P(X \mid Y)$ on MNIST using a ConditionalGan (Mirza et al., 2014).

Many applications: art, image manipulation, robustness, physics & finance, etc.

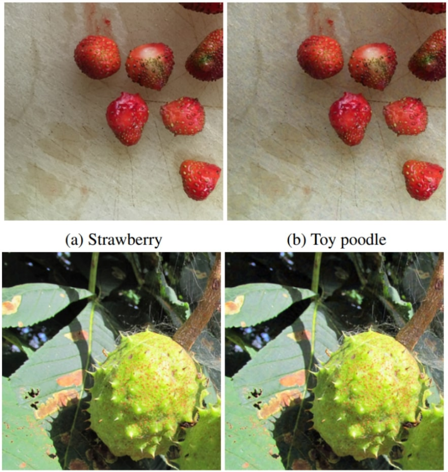
Art: Edmond de Belamy

Figure: Edmond de Belamoy

Merchandising: Virtual Try-on Problem

Figure: vue.ai

Robustness: Attacking Classifiers with GANs



(a) Strawberry (b) Toy poodle

(c) Buckeye (d) Toy poodle

Figure: C. Xiao et al., 2018

Robustness: Defending Classifiers with GANs

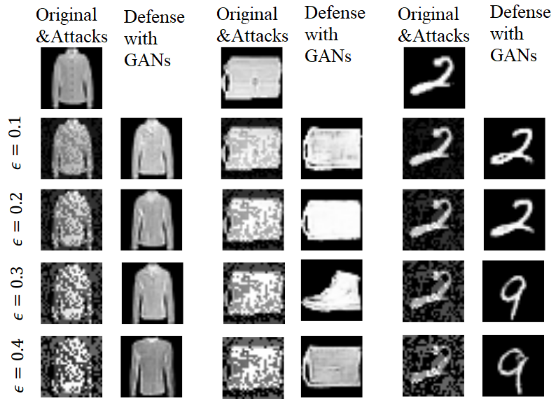
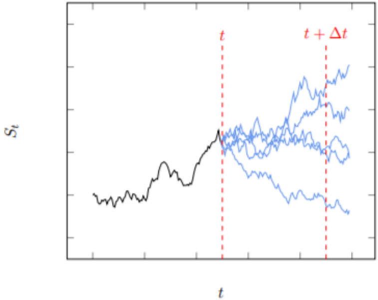


Figure: Samangouei et al., 2018

Physics & Finance

- Using GANs to solve SDEs (L. Yang et al., 2018).
- Synthetic data generation (Takahashi et al., 2019) and Monte Carlo simulation of SDEs using GANs (Rhijn et al., 2021).



(a) Simulating MCMC with GANs: C. Xiao et al., 2018.

- Market prediction (Xingyu et al., 2018): a model that learns the properties of data without explicit assumptions or mathematical formulations.
- Pricing options with GANs.

Many Types of Deep Generative Models

In this course, we will focus on the three main types of models that have been used for image generation.

- **GANs** frame generative modeling, an unsupervised learning problem, as a supervised one.
- **VAEs** indirectly optimize the log-likelihood of the data by maximizing the evidence lower bound (ELBO).
- **Diffusion models** recover the data from pure noise by learning how to invert destructive image perturbations.

Many Types of Deep Generative Models

In this course, we will focus on the three main types of models that have been used for image generation.

- **GANs** frame generative modeling, an unsupervised learning problem, as a supervised one.
- **VAEs** indirectly optimize the log-likelihood of the data by maximizing the evidence lower bound (ELBO).
- **Diffusion models** recover the data from pure noise by learning how to invert destructive image perturbations.

Other types of models exist.

- **Normalizing flows** are defined as sequences of provably invertible transformations.
- **Autoregressive models** generate samples dimension by dimension (pixel by pixel, word by word).

Both directly optimize the log-likelihood.

Outline

Generating Data from Noise

Generator Network

A generator g_θ is a network mapping samples z from a known distribution p_z over $\mathbb{R}^d$ to parameters of a higher-dimensional distribution $\mathcal{G}$ over $\mathbb{R}^D$:

$$p_\theta : \qquad z \sim p_z, \quad x \sim \mathcal{G}(g_\theta(z)).$$

- g_θ, $\mathcal{G}$ and p_z define a generated distribution p_θ:

$$p_\theta(x) = \int_z p_\theta(x \mid z) p(z) \, \mathrm{d}z.$$

- The modeled distribution is defined as a push-forward distribution.
- Examples:
  - For GANs, $\mathcal{G}$ is a Dirac: $x = g_\theta(z)$.
  - For VAEs, $\mathcal{G}$ is usually a Gaussian: $x \sim \mathcal{N}(g_\theta^\mu(z), g_\theta^\sigma(z))$.
- Usually, $d \ll D$.

How to Train a Generator?

Objective

Choose a distance/divergence $\mathcal{D}$. $p_\theta \approx p_{\mathrm{data}}$ by having $\mathcal{D}(p_{\mathrm{data}} \parallel p_\theta) \approx 0$.

In practice

We can choose $\mathcal{D} = \mathbb{D}_{\mathrm{KL}}$ (the KL divergence).

How to Train a Generator?

Objective

Choose a distance/divergence $\mathcal{D}$. $p_\theta \approx p_{\text{data}}$ by having $\mathcal{D}(p_{\text{data}} \parallel p_\theta) \approx 0$.

In practice

We can choose $\mathcal{D} = \mathbb{D}_{\text{KL}}$ (the KL divergence).

$$\begin{aligned}
\mathbb{D}_{\text{KL}}(p_{\text{data}} \parallel p_\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\
&= -\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x) - H(p_{\text{data}}).
\end{aligned}$$

How to Train a Generator?

Objective

Choose a distance/divergence $\mathcal{D}$. $p_\theta \approx p_{\text{data}}$ by having $\mathcal{D}(p_{\text{data}} \parallel p_\theta) \approx 0$.

In practice

We can choose $\mathcal{D} = \mathbb{D}_{\text{KL}}$ (the KL divergence).

$$\begin{aligned}\mathbb{D}_{\text{KL}}(p_{\text{data}} \parallel p_\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\ &= -\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x) - H(p_{\text{data}}).\end{aligned}$$

- We would like to maximize the log-likelihood $\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x)$.

How to Train a Generator?

Objective

Choose a distance/divergence $\mathcal{D}$. $p_\theta \approx p_{\text{data}}$ by having $\mathcal{D}(p_{\text{data}} \parallel p_\theta) \approx 0$.

In practice

We can choose $\mathcal{D} = \mathbb{D}_{\text{KL}}$ (the KL divergence).

$$
\begin{aligned}
\mathbb{D}_{\text{KL}}(p_{\text{data}} \parallel p_\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\
&= -\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x) - H(p_{\text{data}}).
\end{aligned}
$$

- We would like to maximize the log-likelihood $\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x)$.
- Intractable in the general case (hard to calculate or impossible):

$$
p_\theta(x) = \int_z p_\theta(x \mid z) p(z) \, \mathrm{d}z.
$$

How to Train a Generator?

Objective

Choose a distance/divergence $\mathcal{D}$. $p_\theta \approx p_{\text{data}}$ by having $\mathcal{D}(p_{\text{data}} \parallel p_\theta) \approx 0$.

In practice

We can choose $\mathcal{D} = \mathbb{D}_{\text{KL}}$ (the KL divergence).

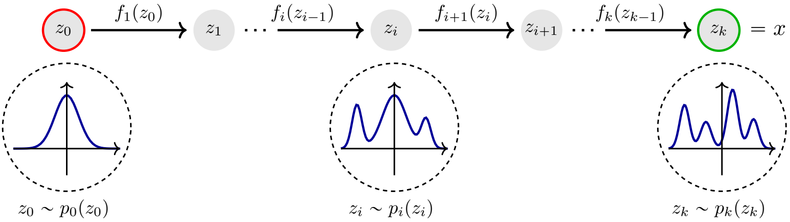
$$\begin{aligned}\mathbb{D}_{\text{KL}}(p_{\text{data}} \parallel p_\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\ &= -\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x) - H(p_{\text{data}}).\end{aligned}$$

- We would like to maximize the log-likelihood $\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x)$.
- Intractable in the general case (hard to calculate or impossible):

$$p_\theta(x) = \int_z p_\theta(x \mid z) p(z) \,\mathrm{d}z.$$

- Let's see two tricks to tackle this problem.

Trick 1: Normalizing Flows



Janosh Riebesell. MIT license.

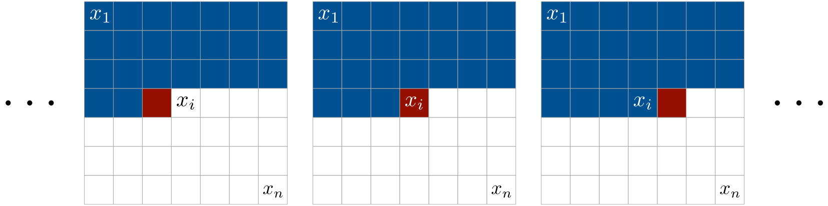
Principle

If $g_\theta = f_k \circ \ldots \circ f_1$ with invertible f_is ($d = D$), then:

$$p_\theta(x) = p_z(g_\theta^{-1}(x)) \left| \det \mathrm{Jac}_x(g_\theta^{-1}) \right| .$$

→ *Architecture difficult to design; used for specific applications.*

Trick 2: Autoregressive Models



Oord et al. (2016).

Principle

Generate x dimension by dimension, component by component:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i}).$$

→ *Computationally heavy and more adapted to NLP.*

Model Comparisons

Model	Sampling	Training	Stability	Results	Efficiency
Flows	Generator	Exact NLL	OK	Insufficient for images	Fast
AR	Autoregressive generator	Exact NLL	OK	SOTA	Prohibitive

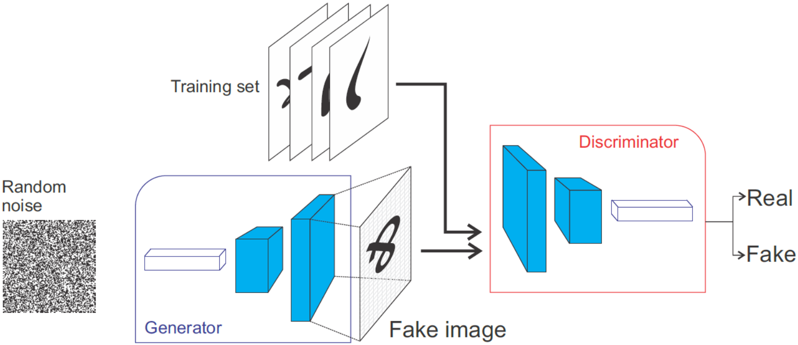
Outline

Moving away from likelihood (1)

GANs circumvent the problem with adversarial training.

Outline

GANs (Goodfellow et al., 2014)



Source: medium.

Motivation: Generating Artificial Contents.

Pros

- Simple generation.
- Work extremely well with high-dimensional data.
- Allow manifold discovering: image interpolation.

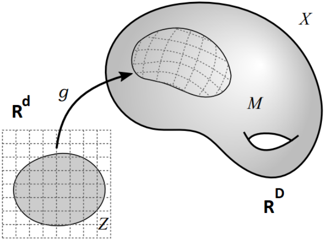
Abdal et al., 2019.

Cons

- Unknown probability density function: we cannot easily check low density areas.
- Tricky training.

The Data

- **Data**:
  - ▻ Target distribution: probability measure $\mu^\star$ on $\mathbb{R}^D$.
  - ▻ Finite-samples: $X_1, \ldots, X_n$ i.i.d. as $\mu^\star$. μ_n: empirical measure.
  - ▻ Objective: how can we sample from $\mu^\star$?
- Latent variable:
  - ▻ Z defined on $\mathbb{R}^d$.
  - ▻ Z is typically uniform or Gaussian.
  - ▻ $d \ll D$: the manifold hypothesis.



Shao et al., 2018.

Generator & Discriminator

Generator: a parametric family of functions from $\mathbb{R}^d$ to $\mathbb{R}^D$.

- Each G_θ is a neural network.
- Definition: $G_\theta(Z) \overset{\mathcal{L}}{\sim} \mu_\theta$.
- Notation: $\mathcal{G} = \{G_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^P$.
- Associated family of **distributions**: $\mathcal{P} = \{\mu_\theta : \theta \in \Theta\}$.
- Each μ_θ is a **candidate** to represent $\mu^\star$.

Discriminator: a parametric family of functions from $\mathbb{R}^D$ to $\mathbb{R}$.

- **Notation**: $\mathcal{D} = \{D_\alpha : \alpha \in \Lambda\}$, $\Lambda \subseteq \mathbb{R}^Q$.
- In GANs algorithms, each D_α is a **neural network**.
- D_α is trained to distinguish between real and fake samples.

Adversarial Principle

- **Objective**: two-player game, looking for a Nash equilibrium to

$$\inf_{\theta \in \Theta} \sup_{\alpha \in \Lambda} \Big[\mathbb{E} \log(D_\alpha(X)) + \mathbb{E} \log(1 - D_\alpha(G_\theta(Z))) \Big].$$

  - ▻ The higher $D(x)$, the higher the probability that x is drawn from $\mu^\star$.
  - ▻ The generator and the discriminator have opposite objectives.
  - ▻ Forget: estimation by maximum likelihood.
  - ▻ Forget: a strategy based on nonparametric density estimation.
- **Empirical version**:

$$\inf_{\theta \in \Theta} \sup_{\alpha \in \Lambda} \Big[\frac{1}{n} \sum_{i=1}^{n} \log(D_\alpha(X_i)) + \mathbb{E} \log(1 - D_\alpha(G_\theta(Z))) \Big].$$

- The min-max optimum is found by alternating stochastic gradient descent.
- **Generative principle**: $\hat{\theta}_n \to G_{\hat{\theta}_n} \to G_{\hat{\theta}_n}(Z_1), G_{\hat{\theta}_n}(Z_2) \ldots \to$ new images.

GANs: (A Bit of) Theory

- Let us denote
  - μ the density of the true data
  - $\mu_G = G(\mu_{\text{noise}})$ the density of the data generated by a generator G
- Our main goal is to find G that minimizes a well-chosen distance between μ and μ_G
- **Intuition**: the performance of the best discriminator mesures this gap between μ and μ_G (the bigger the gap, the better the optimal discriminator).

Can we formalize this intuition ?

GANs: (A Bit of) Theory

Solving the Inner Optimization Problem

The optimal discriminator (without regularization) D_G^* is

$$x \mapsto \frac{\mu(x)}{\mu(x) + \mu_G(x)} \quad .$$

The corresponding loss at this point is

$$\mathcal{L}_G(D_G^*) = 2\mathbb{D}_{\mathrm{JS}}(\mu, \mu_G) - \log 4 \quad ,$$

where $\mathbb{D}_{\mathrm{JS}}$ is the Jensen-Shannon divergence (symmetric variant of the KL-divergence).

Training the GAN $\equiv$ finding G that minimizes $\mathbb{D}_{\mathrm{JS}}(\mu, \mu_G)$

The Role of the Discriminator

In practice, one has always $\mathcal{D} = \{D_\alpha : \alpha \in \Lambda\}$

$$\sup_{\alpha \in \Lambda} \Big[\mathbb{E} \log(D_\alpha(X)) + \mathbb{E} \log(1 - D_\alpha(G_\theta(Z))) \Big]$$

acts like a divergence between the distributions μ_θ and the empirical distribution μ_n.

- **Neural net divergence** (Arora et al., 2017).
- **Adversarial divergence** (Liu et al., 2017).

Other Variants of GANs

- **Least squares GANs** (Mao et al., 2017), related to the Pearson-ξ^2 div.:

$$\text{(discr. objective)} \qquad \sup_{\alpha \in \Lambda} \sum_{i=1}^{n} (D_\alpha(X_i) - 1)^2 + \sum_{i=1}^{n} D_\alpha(G_\theta(Z_i))^2,$$

$$\text{(gen. objective)} \qquad \inf_{\theta \in \Theta} \sum_{i=1}^{n} (D_\alpha(G_\theta(Z_i)) - 1)^2.$$

- Nowozin et al. (2016) proposed **f-GANs** and showed that any f-divergence can be used for training GANs:

$$\inf_{\theta \in \Theta} \sup_{\alpha \in \Lambda} \mathbb{E} D_\alpha(X) - \mathbb{E}(f^\star \circ D_\alpha)(G_\theta(Z)), \quad f^\star \text{ convex conjugate.}$$

- **WGANs**.

GANs: Alchemy?

Lots of "hacks" to stabilize the training:

- Normalize the inputs
- $\min \log(1 - D)$ (saturating) vs $\max \log(D)$ (non-saturating) for the generator
- Choose the noise prior wisely
- BatchNorm on full real / fake images
- Avoid Sparse Gradients (ReLu $\rightarrow$ LeakyReLu)
- Use soft / noisy labels
- Choose the optimizers wisely (e.g. Adam for G, SGD for D), decay rates for Adam are important
- Exponential moving average on the generator's parameters
- ...

(e.g. https://github.com/soumith/ganhacks)

GANs: Pathological behaviors

- **Oscillation / bad convergence**

Due to alternating optimization to solve the minimax game / find a Nash equilibrium

- **Unstability** / divergence

- **Mode collapse**

Happens when the training data is multi-modal (which is usually the case in practice): can be a good strategy for the generator to target the easiest mode of the target distribution (pullover in the example below)

- **Breathrough:** the theoretical study by Arjovsky, Chintala, and Bottou (2017)

- **The Jensen-Shannon** divergence does not allow to take into account the metric structure of the space.

- The authors propose **WGANs** which have become a standard in machine learning.

GANs training problems

From Roth et al., 2017, there are three different challenges for learning the model distribution:

- **empirical estimation**: the model family may contain the true distribution, but one has to identify it based on a finite training sample.
- **density misspecification**: there exists no parameter for which these densities are similar.
- **dimensional misspecification**: the model distribution and the true distribution do not have a density function wrt the same base measure (supp(P) $\bigcap$ supp(Q) may be negligible).

GANs vs Flows: an interesting comparison

Setting

- Let $\mathbb{R}^d$ be the latent space with latent variable Z.
- Let $\mathcal{G} = \{G_\theta, \theta \in \Theta\}$ be a class of invertible functions.

Pros

- Simpler architecture $\&$ simpler loss: likelihood.
- Less prone to mode collapse. Especially, when compared to cGANs (known to be nearly deterministic).
- Super Resolution image generation Lugmayr et al., 2020.

Cons

- The input and output dimensions must be the same.
- The transformation must be invertible.
- The latent space is still high dimensional, so it's harder to play with it.

Visual comparisons Flow vs GANs

Figure: **Left:**StyleGAN. **Right:** Glow.

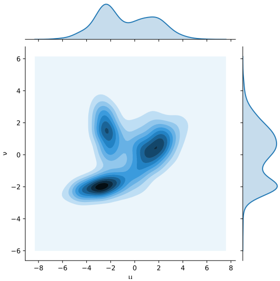
Outline

Wasserstein GANs

- They propose to go with the Wasserstein distance $\mathbb{D}_{W_1}$.

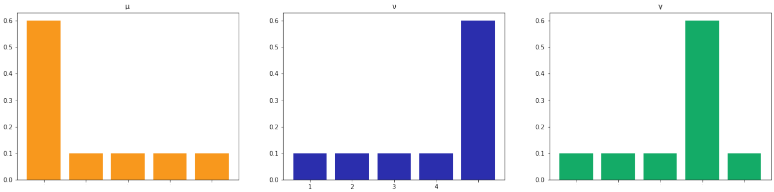
$$\mathbb{D}_{W_1}(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int d(x, y) \, \mathrm{d}\gamma(x, y)$$

- Continuous "earth moving distance"



Wasserstein GANs (cted)

Advantages of $\mathbb{D}_{W_1}$ over $\mathbb{D}_{\text{JS}}$?



$$\mathbb{D}_{W_1}(\mu, \nu) = 2 > \mathbb{D}_{W_1}(\mu, \gamma) = 1.5$$

$$\mathbb{D}_{\text{JS}}(\mu, \nu) = 0.20 < \mathbb{D}_{\text{JS}}(\mu, \gamma) = 0.25$$

Problem: How to compute $\arg\min_G \mathbb{D}_{W_1}(\mu, \mu_G)$?

Wasserstein GANs (cted)

- Using Kantorovich-Rubinstein duality theorem, WGANs aim to solve:

$$\mathbb{D}_{W_1}(\mu, \mu_G) = \max_{\|D|_L \leqslant 1} \Big[\mathbb{E}_{X\sim\mu}\left[D(X)\right] - \mathbb{E}_{X\sim\mu_G}\left[D(X)\right]\Big] \quad ,$$

where $\|D|_L$ is the Lipschitz semi-norm equal to

$$\max_{x,y} \frac{\|D(x) - D(y)\|}{\|x - y|} \quad .$$

- We get a **new loss** for the discriminator !
- **WGANs**: in practice, one **always** has a parametric $\mathcal{D} = \{D_\alpha : \alpha \in \Lambda\}$:

$$\inf_{\theta\in\Theta} \sup_{\alpha\in\Lambda} |\mathbb{E}_{\mu^\star} D_\alpha - \mathbb{E}_{\mu_\theta} D_\alpha| = ??$$

- **Empirical WGANs**:

$$\inf_{\theta\in\Theta} \sup_{\alpha\in\Lambda} \Big[\frac{1}{n}\sum_{i=1}^{n} D_\alpha(X_i) - \mathbb{E}D_\alpha(G_\theta(Z))\Big] = ??$$

Understanding the benefits of WGANs

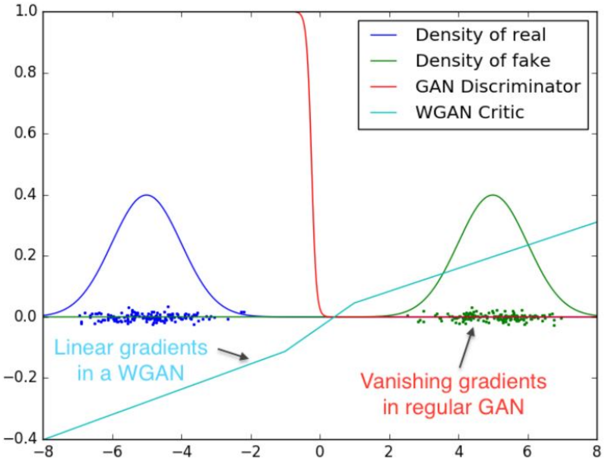


Figure: From Arjovsky and Bottou, 2017

Controlling the Gradient of the Discriminator?

The **compactness** requirement is **classical** when parameterizing GANs.

- Weight clipping (Arjovsky, Chintala, et al., 2017).
- Gradient penalty (Gulrajani et al., 2017).
- Spectral normalization (Miyato et al., 2018).
- Block orthonormalization.
- This opens a whole new focus for the study of 1-Lipschitz neural networks (Béthune et al., 2022; Tanielian and Biau, 2021).

Advantages of WGANs

- Wasserstein Distance is continuous and almost differentiable everywhere, which allows us to train the model to optimality.
- JS Divergence locally saturates as the discriminator gets better, thus the gradients becomes zero and vanishes.
- Wasserstein distance is a meaningful metric, i.e, it converges to 0 as the distributions get close to each other and diverges as they get farther away.
- The mode collapse problem is also mitigated when using Wasserstein distance as the objective function.

Understanding the performance of WGANs

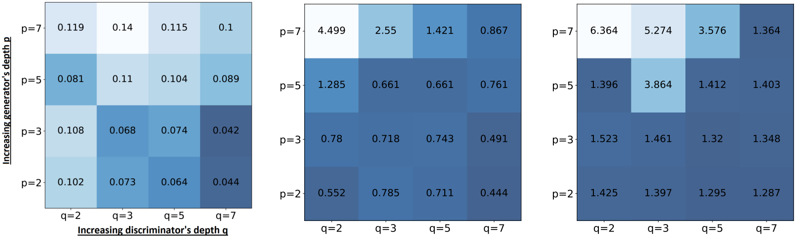
$$d_{\mathsf{Lip}_1}(\mu^\star, \mu_{\hat{\theta}_n}) \leqslant \varepsilon_{\mathsf{estim}} + \varepsilon_{\mathsf{optim}} + \inf_{\theta\in\Theta} d_{\mathsf{Lip}_1}(\mu^\star, \mu_\theta)$$
$$= \varepsilon_{\mathsf{estim}} + \varepsilon_{\mathsf{optim}} + \varepsilon_{\mathsf{approx}}$$

- $\varepsilon_{\mathsf{estim}} = \sup_{\theta_n\in\hat{\Theta}_n} \left[d_{\mathsf{Lip}_1}(\mu^\star, \mu_{\theta_n}) - d_{\mathsf{Lip}_1}(\mu^\star, \mu_{\bar{\theta}_n})\right]$ (data)
- $\varepsilon_{\mathsf{optim}} = \sup_{\bar{\theta}\in\bar{\Theta}} d_{\mathsf{Lip}_1}(\mu^\star, \mu_{\bar{\theta}}) - \inf_{\theta\in\Theta} d_{\mathsf{Lip}_1}(\mu^\star, \mu_\theta)$ (metric discrepancy)
- $\varepsilon_{\mathsf{approx}} = \inf_{\theta\in\Theta} d_{\mathsf{Lip}_1}(\mu^\star, \mu_\theta)$ (model)

We can now decompose the performance in **three distinct losses** compared to the classic **bias/variance trade off**.

Synthetic Experiments

- **Setting**: $\mu^\star$ is a mixture of Gaussian densities with 2, 4 or 9 **components**.
- A **family** of generators: $\{\mathcal{G}_p : p = 2, 3, 5, 7\}$.
- A **family** of discriminators: $\{\mathcal{D}_q : q = 2, 3, 5, 7\}$.
- We draw $X_1, \ldots, X_n$ drawn from $\mu^\star$ with $n = 5000$.
- We plot the **performance**: $\sup_{\theta_n \in \hat{\Theta}_n} \; d_{\mathrm{Lip}_1}(\mu^\star, \mu_{\hat{\theta}_n}) \leqslant \varepsilon_{\mathrm{estim}} + \varepsilon_{\mathrm{optim}} + \varepsilon_{\mathrm{approx}}$.



(a) $K = 2$. (b) $K = 4$. (c) $K = 9$.

Figure: $d_{\mathrm{Lip}_1}(\mu^\star, \mu_{\theta_n})$ for different generator's and discriminator's capacity.

A too small discriminator facilitate instability and mode collapse

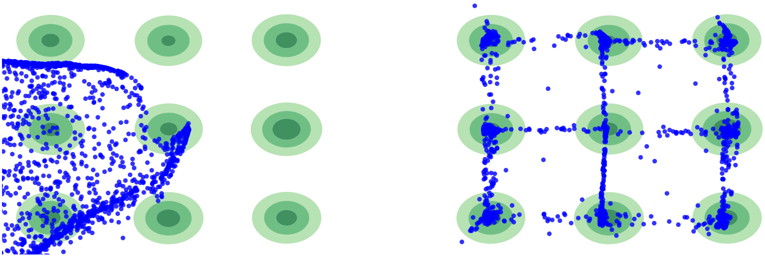


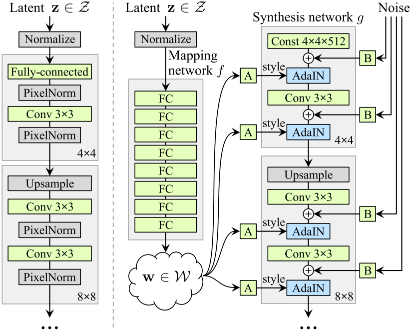
Figure: Left: Discriminator's depth=2, Generator's=4. Right: Discriminator's depth=5, Generator's=4

Scaling WGAN and Gradient Penalties

- The gradient penalty can be used with many other losses.
- Many SOTA models have been trained with hinge or vanilla losses combined with a gradient penalty.

- BigGAN (Brock et al., 2019): large-scale analysis for training on ImageNet with many insights and hacks.

- StyleGAN (Karras et al., 2019): introduced new style-based generator for hierarchical generation.

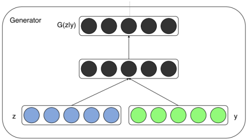


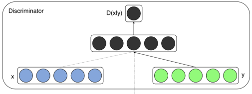
(a) Traditional (b) Style-based generator

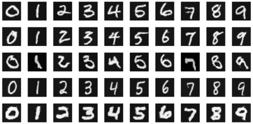
Outline

Conditional GANs

- Introduced by Mirza et al. (2014): use of a conditioning input into your GAN.
- The conditionning input is given both to the generator and the discriminator







When dealing with **disconnected manifolds**:

- Khayatkhoei et al. (2018) use of multiple generators.
- Tanielian, Issenhuth, et al. (2020): gradient's based truncation.
- cGANs solve easily this problem by adding disconnectedness in the latent space.

Conditional GANs and paired datasets

Formalization

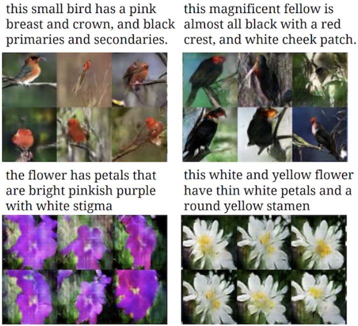
We have a dataset $\mathcal{D}$ made a paired items $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with both conditioning item and a target image. The conditioning can be anything (text, image, ...). By using both elements, we can add a reconstruction term to the GAN loss.

This enables us to solve the following tasks.

- Text-to-Image translation: StackGANs (Zhang et al., 2017).
- Image-to-Image translation: Pix2Pix (Isola et al., 2017).
- Semantic-Image-to-Photo translation (T.-C. Wang et al., 2018).
- Image inpainting (Wu et al., 2019).
- Super resolution (Ledig et al., 2017).

The unpaired case can also be tackled (Zhu et al., 2017).

Examples


this small bird has a pink breast and crown, and black primaries and secondaries.
this magnificent fellow is almost all black with a red crest, and white cheek patch.
the flower has petals that are bright pinkish purple with white stigma
this white and yellow flower have thin white petals and a round yellow stamen

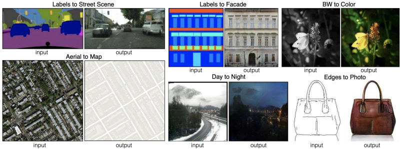
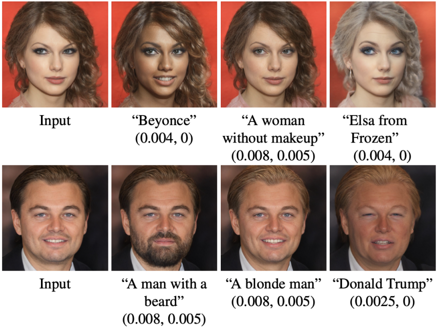

Labels to Street Scene
input
output
Aerial to Map
input
output
Labels to Facade
input
output
Day to Night
input
output
BW to Color
input
output
Edges to Photo
input
output

Image manipulation with GANs & CLIP

- CLIP jointly trains an image encoder and text encoder.
- Trained on 400 million (image, text).
- The training objective: given a batch of N (image, text) pairs, predicting which of the N $\times$ N possible (image, text) pairings across a batch actually occurred.

StyleCLIP and **VQGAN-CLIP** Crowson et al., 2022 both combine GANs and CLIP, and trains a latent optimization in $\mathcal{W}^+$ (but it requires a few minutes of optimization) .



Model Comparisons

Model	Sampling	Training	Stability	Results	Efficiency
GANs	Generator	Min-max	Nash equilibrium	Sharp but mode collapse	Fast
Flows	Generator	Exact NLL	OK	Insuffi-cient for images	Fast
AR	Auto-regressive generator	Exact NLL	OK	SOTA	Pro-hibitive

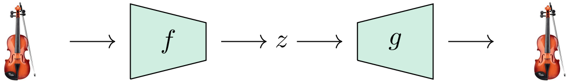
Outline

VAEs circumvent the problem with the ELBO (evidence lower bound).

Outline

Autoencoders

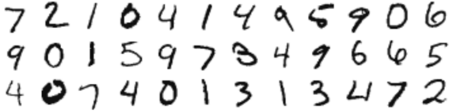
- Main idea: force a self-supervised network to compress the original representation in a low-dimensional latent space.



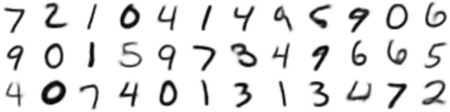
- The goal is to learn an encoder f and a decoder g such that $g \circ f$ is close to identity.
- If f and g are linear, the optimal solution is given by a PCA.
- Otherwise, we can achieve better performance with deep networks.

Deep Autoencoders

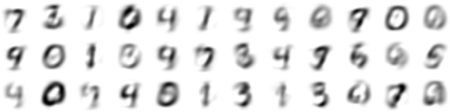
X (original samples)



$g \circ f(X)$ (CNN, $d = 8$)



$g \circ f(X)$ (PCA, $d = 8$)



(by courtesy of François Fleuret)

How to sample from autoencoders ?

- Simple answer: sample z in the latent space and feed it into the decoder
- However it is very likely that the encoded inputs lies in a low-dimensional manifold inside the latent space

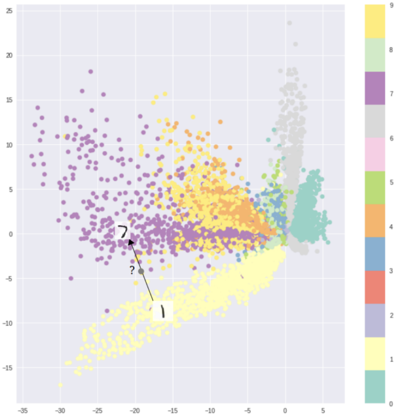
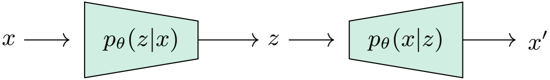


Figure: From https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs as probablistic autoencoders

- AEs can lead to severe **overfitting**: some points of the latent space will give meaningless content once decoded.
- How can we regularize autoencoders?
- **Let us constraint the latent variable z to follow a fixed distribution from which we can sample easily.**
- Let's rewrite everything with probabilities!



- $p_\theta(z \mid x)$ is intractable since we do not know the distribution of the true data so we approximate it by the variational distribution $q_\phi(z \mid x)$ that should minimize:

$$\mathbb{D}_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p(z \mid x)).$$

VAEs' principles

VAEs can be defined as being an **autoencoders** whose training are **regularised** to avoid overfitting and ensure that the latent space has good properties that enable generative process.

- First, the input X is encoded as distribution over the latent space.
- Second, a point z from the latent space is sampled from that distribution p.
- Third, the sampled point is decoded and the reconstruction error can be computed.
- Finally, the reconstruction error is backpropagated through the network.

VAEs and log-likelihood

Lemma

For any variational distribution q_ϕ, the (true) marginal log-likelihood $\log p_\theta(x)$ can be written as

$$\log p_\theta(x) = \mathbb{D}_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p_\theta(z \mid x)) + \mathcal{L}_{\theta,\phi}.$$

Note that:

- $\mathcal{L}_{\theta,\phi}$ is called the **variational lower bound** since $\log p_\theta(x) \geqslant \mathcal{L}_{\theta,\phi}$.
- For a fixed θ, minimizing the KL-divergence wrt ϕ is similar to **maximize** $\mathcal{L}_{\theta,\phi}$.
- For a fixed ϕ, maximizing $\mathcal{L}_{\theta,\phi}$ wrt θ, maximizes the expected log-likelihood of the data.

VAEs loss function

- Let's summarize!

The loss function to minimize is $-\mathcal{L}_{\theta,\phi}$ and can be rewritten as:

$$-\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] + \mathbb{D}_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p_\theta(z)) \ .$$

- The first term is called the **reconstruction loss**: if $p_\theta(x \mid z)$ is Gaussian, then it becomes an MSE.
- The second term can be seen as a **regularizer** toward the prior distribution of the latent variable p_θ.

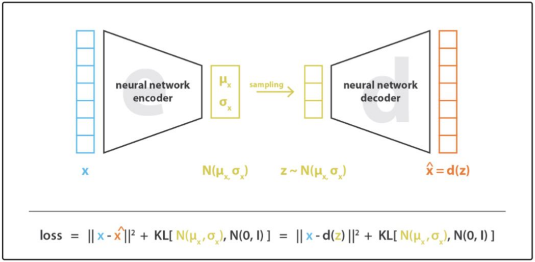
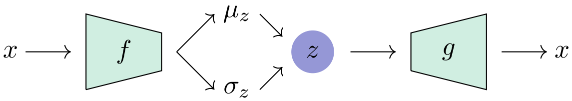


Figure: From towardsdatascience

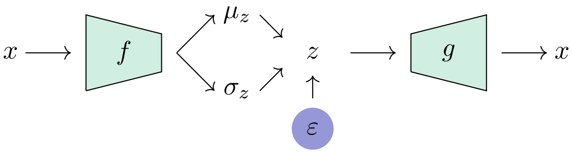
One last problem! How to backprop?

We need to be very careful about the way we sample from the distribution returned by the encoder during the training.

- **Problem:** Impossible to backpropagate through a **stochastic node** like z.



- **Solution (ex. for a Gaussian posterior):** Let's write $z = \mu_z + \sigma_z \odot \epsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$ to have a differentiable path end-to-end.



Reparametrization trick

Outline

From VAEs to VQGANs

Van Den Oord et al. (2017) propose a Vector Quantized VAE. It differs from VAEs in two key ways:

- The encoder network outputs discrete, rather than continuous, codes.
- The prior is learnt rather than static.

The consequences are:

- It allows the model to circumvent issues of "posterior collapse" (i.e. when the signal from the encoder is either too weak or too noisy, and as a result, decoder starts ignoring samples drawn from the posterior).
- The discretization is done via a step of quantization.
- The prior distribution over the discrete latents $p(z)$ is a categorical distribution.
- It and can be made autoregressive by depending on other z in the feature map.
- Esser et al. (2021) propose to add an adversarial loss to VQVAE to obtain VQGAN.

Vector-quantized latent space

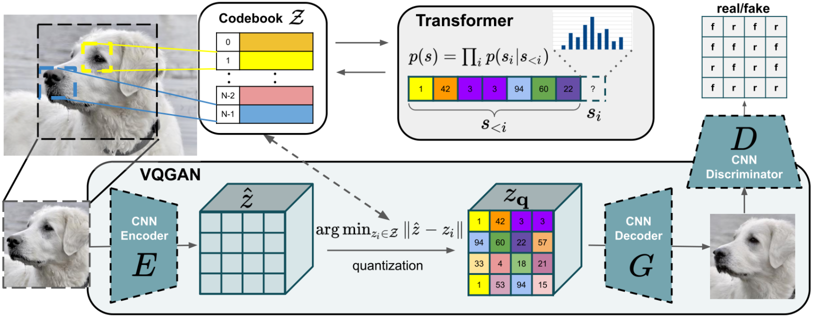
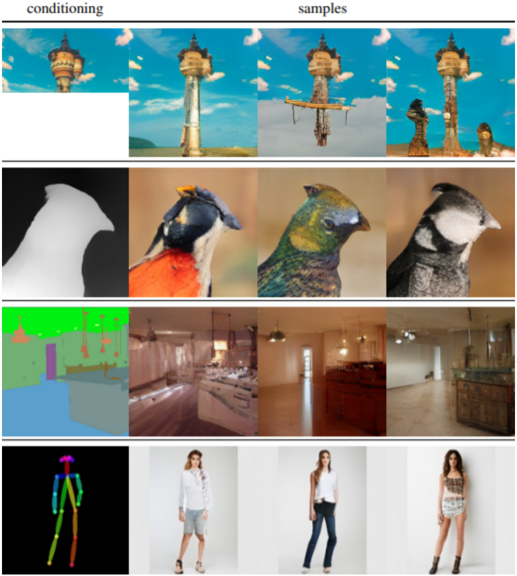


Figure: Introduction of VQGANs from Esser et al., 2021.

Unifying NLP and Computer Vision

- VQGAN aims at training a triplet (E, D, C) (encoder, decoder, codebook).
- With a trained encoder E, for any dataset of images $\mathcal{D}$, one can create a dataset of sequences $\mathcal{D}_S$.
- One can train any language model on $\mathcal{D}_S$ (Transformers, RNN, etc...), to be able to generate likely sequences.
- After that, use the decoder to decode them into images.
- **TLTR,** all the known language models (including the newest LLMs) can now be used to generate images.

Generating images with Transformers



Properties of this VQ latent space (1)

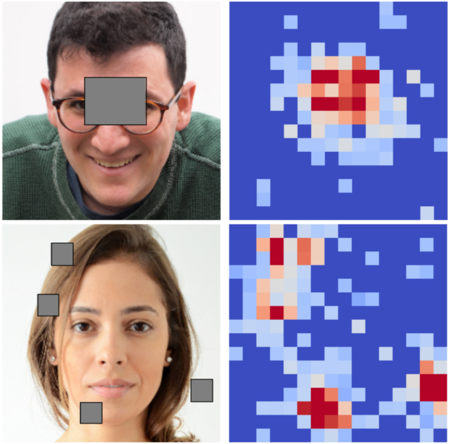


Figure: Each VQGAN token is strongly tied to a small spatial area in the image space. Perturbed images lead to variations of tokens in the latent space.

Properties of this VQ latent space (2)

Figure: Each VQGAN token is strongly tied to a small spatial area in the image space. Collages of images can easily be done with collages of latent representations.

Properties of this VQ latent space (3)

Original.	Difference between original and VQGAN reconstruction.	VQGAN reconstruction.	Difference between original and VQGAN plus latent optimization reconstruction.	VQGAN plus latent optimization reconstruction.

Figure: From Issenhuth et al., 2021: one of the PBS from the VQGAN discrete space is that it can have a lower ability to reconstruct images.

Image manipulation with VQGANs

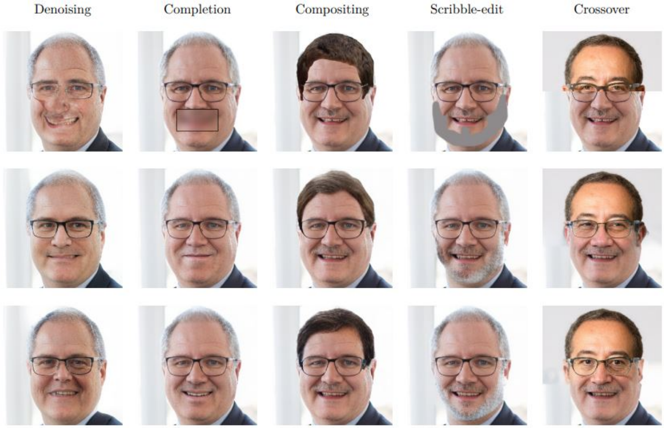


Figure: From Issenhuth et al., 2021: one can train a discrete space model to manipulate images.

Text-to-Image models with a VQ latent space

Two famous large scale models are based on this reconstruction

- DALLE (Ramesh, Pavlov, et al., 2021):
- Pathways Autoregressive Text-to-Image model (PARTI, Yu et al., 2022)

Some numbers:

- Detailed comparisons of four scales of Parti models – 350M, 750M, 3B and 20B – and observe consistent improvements.

Model Comparisons

Model	Sampling	Training	Stability	Results	Efficiency
GANs	Generator	Min-max	Nash equilibrium	Sharp but mode collapse	Fast
VAEs	Generator	ELBO (AE + KL)	Collapsing	Blurry images	Fast
Flows	Generator	Exact NLL	OK	Insuffi-cient for images	Fast
AR	Auto-regressive generator	Exact NLL	OK	SOTA	Pro-hibitive

Outline

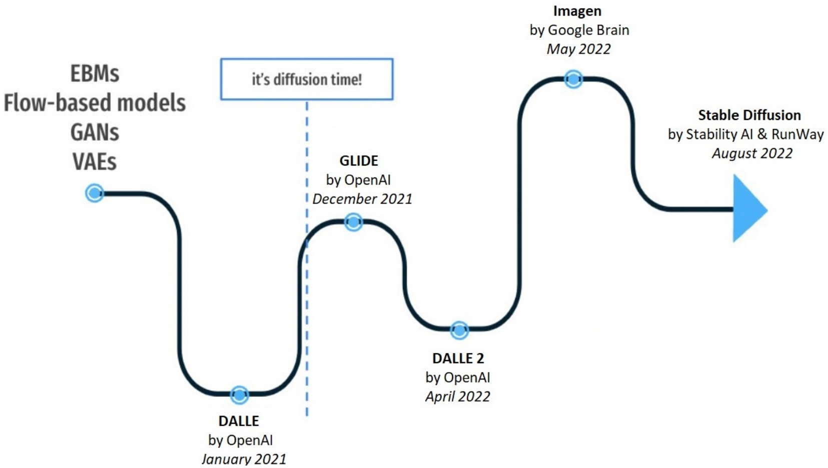
Outline

Moving away from likelihood with Diff. models

Diffusion models replace the generator and instead estimate the score function

The rise of Diffusion models

- Stable Diffusion (Rombach et al., 2022): 330 citations in a year (only 95 papers cited Goodfellow et al., 2020 in 2014-2015).



Intuition

easy!

Generating by Inverting Noise

- We know how to get noise from an image.

Intuition

easy!

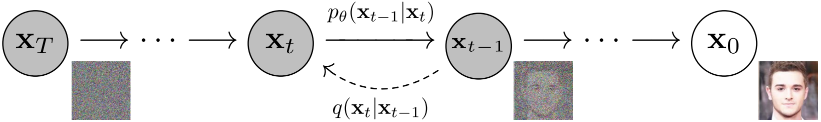
hard!

Generating by Inverting Noise

- We know how to get noise from an image.
- Diffusion models learn the reverse process to generate images from noise.

Outline

Variational Approach



Ho et al. (2020).

- $q(x_t \mid x_{t-1})$: noise process, usually additive Gaussian noise:

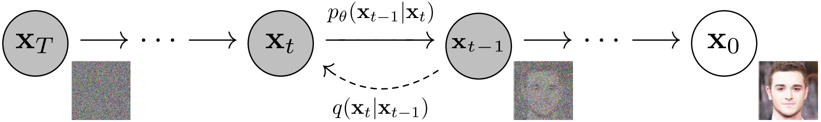
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}x_{t-1}, \beta_t I\right),$$

$$\text{hence } q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\alpha_t}x_0, (1-\alpha_t)I\right).$$

- $q(x_{t-1} \mid x_t)$: true unknown denoising process.
- $p_\theta(x_{t-1} \mid x_t)$: parameterized, approximate denoising process.

- Similar to a hierarchical VAE, we have only access an ELBO can be derived; difference: roles of the encoder and decoder switched.

Variational Approach



Ho et al. (2020).

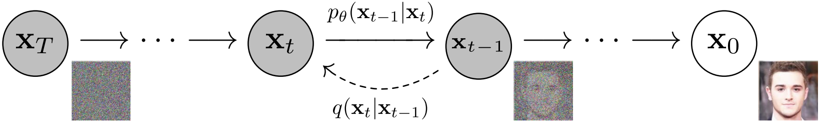
- $q(x_t \mid x_{t-1})$: noise process, usually additive Gaussian noise:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}x_{t-1}, \beta_t I\right),$$

$$\text{hence } q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\alpha_t}x_0, (1-\alpha_t)I\right).$$

- $q(x_{t-1} \mid x_t)$: true unknown denoising process.
- $p_\theta(x_{t-1} \mid x_t)$: parameterized, approximate denoising process.
- For large enough T, $q(x_T) \approx \mathcal{N}\left(0, \sigma^2 I\right)$, so we choose $p_\theta(x_T) = \mathcal{N}\left(0, \sigma^2 I\right)$.
- Similar to a hierarchical VAE, we have only access an ELBO can be derived; difference: roles of the encoder and decoder switched.

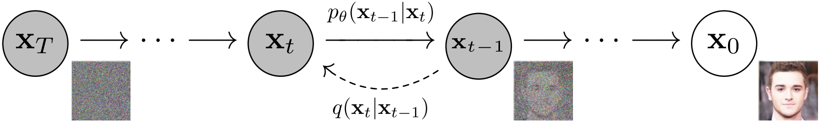
Parameterization & ELBO (Informal)



- The variational lower bound gives:

$$\log p_\theta(x) \geqslant \mathbb{E}_{x_{1:T} \sim q(x_{1:T}|x_0)} \log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} \triangleq \mathcal{L}_\theta .$$

Parameterization & ELBO (Informal)



- The variational lower bound gives:

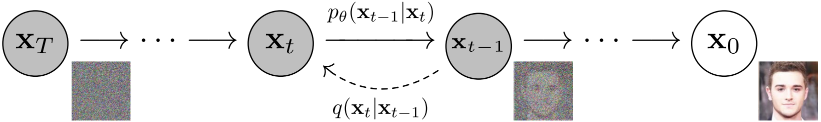
$$\log p_\theta(x) \geqslant \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)} \log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} \triangleq \mathcal{L}_\theta.$$

- To keep all KLs between Gaussians and knowing $q(x_t \mid x_0)$, we choose:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(\mu_\theta(x_t, t), \sigma_t^2 I\right),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\varepsilon_\theta(x_t, t)\right).$$

Parameterization & ELBO (Informal)



- The variational lower bound gives the following loss function:

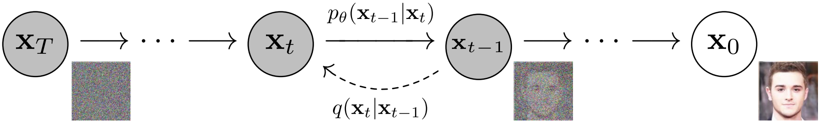
$$\log p_\theta(x) \geqslant \mathbb{E}_{x_{1:T} \sim q(x_{1:T} | x_0)} \log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} \triangleq \mathcal{L}_\theta.$$

- To keep all KLs between Gaussians and knowing $q(x_t \mid x_0)$, we choose:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(\mu_\theta(x_t, t), \sigma_t^2 I\right),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \varepsilon_\theta(x_t, t)\right).$$

Parameterization & ELBO (Informal)



- The variational lower bound gives the following loss function:

$$\log p_\theta(x) \geqslant \mathbb{E}_{x_{1:T} \sim q(x_{1:T}|x_0)} \log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} \triangleq \mathcal{L}_\theta.$$

- To keep all KLs between Gaussians and knowing $q(x_t \mid x_0)$, we choose:

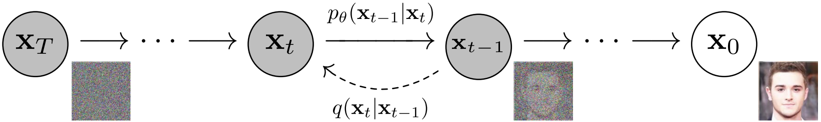
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(\mu_\theta(x_t, t), \sigma_t^2 I\right),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\varepsilon_\theta(x_t, t)\right).$$

- Intuitively, $\varepsilon_\theta(x_t, t)$ aims at reconstructing the noise $\varepsilon \sim \mathcal{N}(0, I)$ used to perturb x_0 into $x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon$.

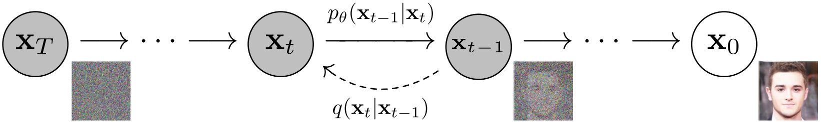
$$\mathcal{L}_\theta = \sum_t \mathbb{E}_{x_0,\varepsilon}\ f(\alpha_t, \beta_t, \sigma_t)\left\|\varepsilon - \varepsilon_\theta(x_t, t)\right\|_2^2, \quad x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\varepsilon.$$

In Practice



- Inference:
  - sample $x_T \sim p_\theta(x_T) = \mathcal{N}\left(0, \sigma^2 I\right)$,
  - sequentially generate x_ts from $p_\theta(x_{t-1} \mid x_t)$ until the final sample x_0.
- ε_θ is a large U-Net, or a transformer;

In Practice

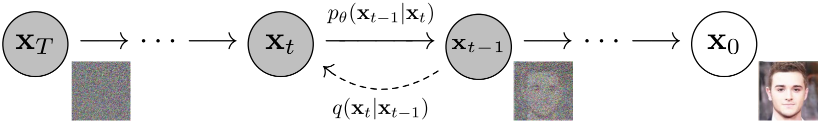


- Inference:
  - sample $x_T \sim p_\theta(x_T) = \mathcal{N}\left(0, \sigma^2 I\right)$,
  - sequentially generate x_ts from $p_\theta(x_{t-1} \mid x_t)$ until the final sample x_0.
- ε_θ is a large U-Net, or a transformer;
- The loss function is usually simplified:

$$\mathcal{L}_\theta = \sum_t \mathbb{E}_{x_0,\varepsilon} \cancel{f(\alpha_t, \beta_t, \sigma_t)} \left\| \varepsilon - \varepsilon_\theta \left(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\varepsilon, t \right) \right\|_2^2 .$$

- Hundreds of diffusion steps are used to generate images: we would like to skip some of them.

In Practice



- Inference:
  - sample $x_T \sim p_\theta(x_T) = \mathcal{N}\left(0, \sigma^2 I\right)$,
  - sequentially generate x_ts from $p_\theta(x_{t-1} \mid x_t)$ until the final sample x_0.
- ε_θ is a large U-Net, or a transformer;
- The loss function is usually simplified:

$$\mathcal{L}_\theta = \sum_t \mathbb{E}_{x_0,\varepsilon} \cancel{f(\alpha_t, \beta_t, \sigma_t)} \left\| \varepsilon - \varepsilon_\theta\left(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\varepsilon, t\right) \right\|_2^2 .$$

- Hundreds of diffusion steps are used to generate images: we would like to skip some of them.

Change of Paradigm

The variational approach is not adapted!

Outline

Score Function

- The whole goal of score-based models is to train a neural network $s_\theta(x)$ to learn $\nabla \log p(x)$, called the score function:

$$\nabla_x \log p_\theta(x) \approx s_\theta(x).$$

- We can represent the score model as a neural network, trained by minimizing the Fisher Divergence with the ground truth score function: $\mathbb{E}_{p(x)}\left[\left\|s_\theta(x) - \nabla p(x)\right\|_2^2\right]$.
- We don't have access to the ground truth score function for real data, so we use score matching approaches to minimize the Fisher divergence without the ground truth.

- How do we learn a function $s_\theta : \mathbb{R}^D \to \mathbb{R}^D$ such that $s_\theta(x) \approx \nabla_x \log p(x)$. We estimate the score from a kernel density estimator $q_\sigma(\tilde{x}) = \frac{1}{N}\sum_i q_\sigma(\tilde{x}|x_i)$, such that $s_\theta(x) \approx \nabla_x \log q_\sigma(x) \approx \nabla_x \log p(x)$.

Score Function: Sampling

- Score function is the gradient of the log likelihood with respect to data x
- This gradient tells us what direction in data space to move in order to increase the likelihood of x
- Score function defines a vector field over the data space, pointing toward the modes
- We can generate samples by starting at any point and following the score until we reach a mode, using Langevin dynamics:

$$x_{i+1} \leftarrow x_i + c\nabla \log p(x_i) + \sqrt{2c}\epsilon, \qquad i = 0, 1, \ldots, K$$

  - x_0 is sampled from a prior
  - $\epsilon \sim \mathcal{N}(0, I)$; ensures that we hover around a mode without collapsing into it, and allows for stochastic trajectories

Score Function: Sampling Trajectories

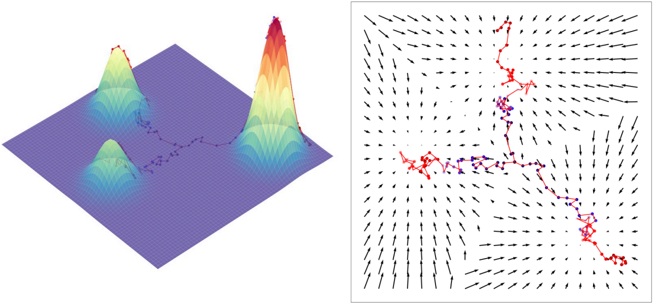


Figure 6: Visualization of three random sampling trajectories generated with Langevin dynamics, all starting from the same initialization point, for a Mixture of Gaussians. The left figure plots these sampling trajectories on a three-dimensional contour, while the right figure plots the sampling trajectories against the ground-truth score function. From the same initialization point, we are able to generate samples from different modes due to the stochastic noise term in the Langevin dynamics sampling procedure; without it, sampling from a fixed point would always deterministically follow the score to the same mode every trial.

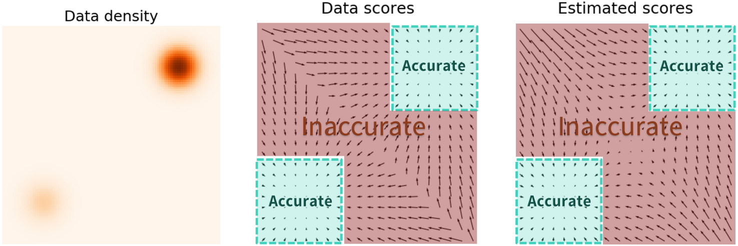
Figure: Figure from Luo, 2022

Score Function Training: Noise Levels

Using multiple levels of Gaussian noise addresses the following issues with score matching: $p_t(x_t) = \int p(x)\mathcal{N}(x, \sigma_t^2 I)\,\mathrm{d}x$ (Song and Ermon, 2019).

Problem 1

Score function is ill-defined when x lies on a low-dimensional manifold in a high-dimensional space



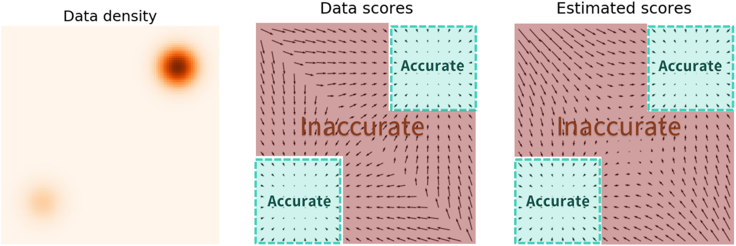
Score Function Training: Noise Levels

Using multiple levels of Gaussian noise addresses the following issues with score matching: $p_t(x_t) = \int p(x)\mathcal{N}(x, \sigma_t^2 I)\,\mathrm{d}x$ (Song and Ermon, 2019).

Problem 2

Learned score function estimate obtained from score matching is not accurate in low density regions

Adding Gaussian noise will increase the area each mode covers in the data distribution, which adds more signal for training in low-density regions



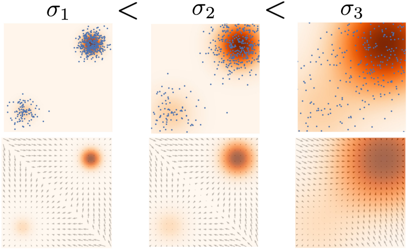
Score Function Training: Noise Levels

Using multiple levels of Gaussian noise addresses the following issues with score matching: $p_t(x_t) = \int p(x)\mathcal{N}(x, \sigma_t^2 I)\,\mathrm{d}x$ (Song and Ermon, 2019).

Problem 3

Langevin dynamics may not mix in the case where the true data is a mixture of disjoint distributions and might hinder the accuracy of score estimation.

Adding multiple levels of Gaussian noise with increasing variance will result in intermediate distributions that respect the ground truth mixing coefficients.



Slow mixing of Langevin dynamics

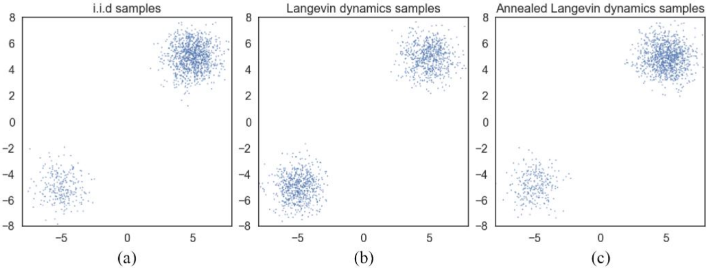


Figure 3: Samples from a mixture of Gaussian with different methods. (a) Exact sampling. (b) Sampling using Langevin dynamics with the exact scores. (c) Sampling using annealed Langevin dynamics with the exact scores. Clearly Langevin dynamics estimate the relative weights between the two modes incorrectly, while annealed Langevin dynamics recover the relative weights faithfully.

Score Function: a recap

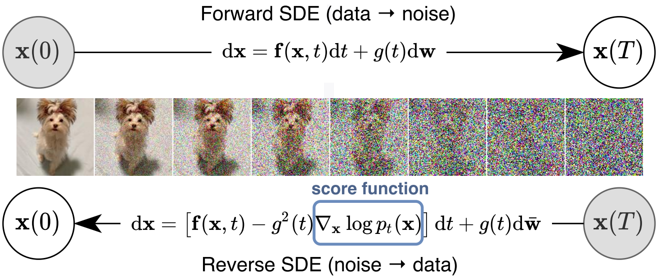
- We then get back the same $\hat{\mathcal{L}}_\theta$ as before:

$$\hat{\mathcal{L}}_\theta = \sum_{t=1}^{T} \lambda(t) \mathbb{E}_{x_0 \sim p_{\mathrm{data}}, x_t \sim p_{\sigma_t}(x_t|x_0)} \left[\left\| s_\theta(x,t) - \nabla \log p_t(x_t \mid x_0) \right\|_2^2 \right]$$

  - $\lambda(t)$: Positive weighting function that is conditioned on noise level t
- We use annealed Langevin dynamics to generate samples
  - Run Langevin dynamics for $t = T, T-1, \ldots, 2, 1$ sequentially
  - Initialize from some fixed prior
  - Each sampling step starts from the final sample of the previous time step
  - Noise levels steadily decrease over time t, and we reduce the step size over time, so the samples converge to the true mode
  - Similar to sampling a variational diffusion model

Unifying diffusion & score-based models

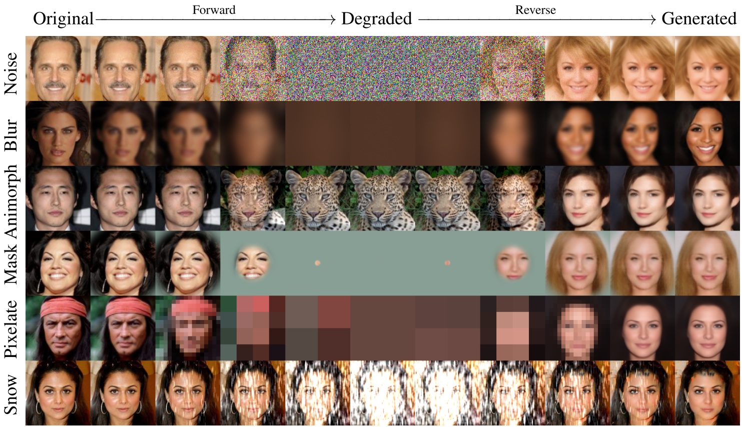
- We can generalize diffusion models to an infinite number of time steps or noise scales → continuous time.
- From a score-based perspective, perturbation of the data can be represented as **a stochastic process**, described by a stochastic differential equation (SDE).
- Sampling requires reversing the SDE → requires estimating the score function at each continuous noise level.



Song, Sohl-Dickstein, et al. (2021).

Beyond White Noise

Perturbations other than Gaussians are possible!

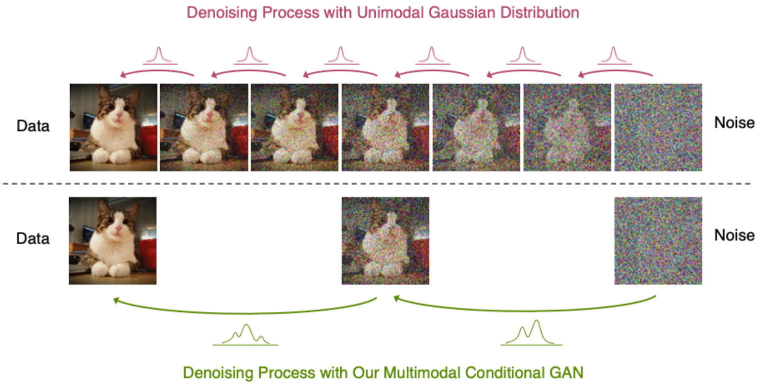


Bansal et al. (2022).

Outline

Combining GANs and Diffusion

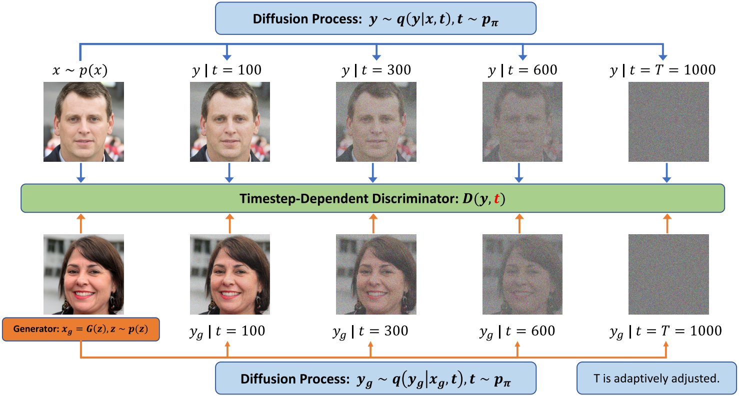
Replacing score by GANs in diffusion...



Denoising Diffusion GANs (Z. Xiao et al., 2022).

Combining GANs and Diffusion

... training a noise-perturbed GAN (Sønderby et al., 2017)!



Diffusion GANs (Zhendong Wang et al., 2022).

To facilitate GANs training, Diffusion GANs use a diffusion process to generate Gaussian-mixture distributed instance noise.

Model Comparisons

Model	Sampling	Training	Stability	Results	Efficiency
GANs	Generator	Min-max	Nash equilibrium	Sharp but mode collapse	Fast
VAEs	Generator	ELBO (AE + KL)	Collapsing	Blurry images	Fast
Diffusion	Differential equation	Denoising	OK	SOTA	Slow
Flows	Generator	Exact NLL	OK	Insuffi-cient for images	Fast
AR	Auto-regressive generator	Exact NLL	OK	SOTA	Pro-hibitive

Outline

Outline

Text2Image model with diffusion

Diffusion models have been largely used for text2image generation. Among the most famous works, we can name:

- **DALLE2** Ramesh, Dhariwal, et al., 2022: diffusion-based decoder conditionned on a CLIP embedding. A two step learning: a prior that produces CLIP image embeddings, and a decoder that generates images.
- **Imagen** Saharia et al., 2022: use of pre-trainded large language models for text encoder. Increasing the size of the language model in Imagen boosts both sample fidelity and image-text alignment much more than increasing the size of the image diffusion model.
- **Stable Diffusion 1 &2** Rombach et al., 2022: to increase both the inference/training, and reduce the compute power, the diffusion is done in latent space of powerful pretrained autoencoders.

Important note: only large scale text2image ***open-source*** *generative model.*

Using CLIP as a conditioning

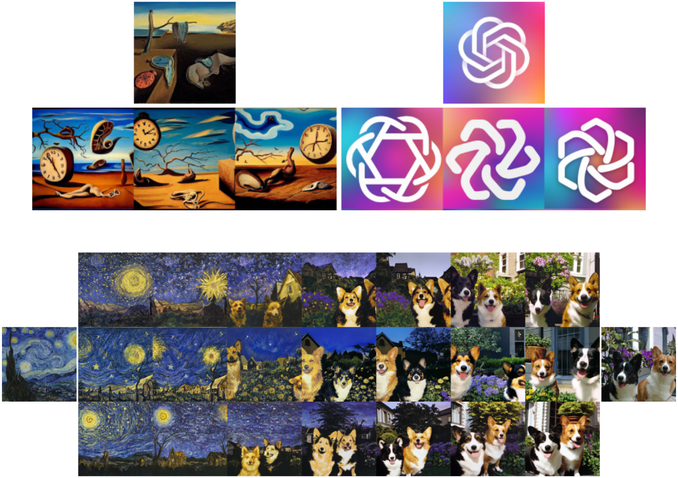


Figure: From Ramesh, Dhariwal, et al., 2022: reconstruction and interpolation with CLIP diffusion.

Challenge 1: problems when generating humans

Figure: From Ramesh, Dhariwal, et al., 2022: all large scale generative models still have issues with generating humans: especially faces, hands.

Challenge 2: spelling issues & hidden language

(a) Image generated with the prompt: "`Two farmers talking about vegetables, with subtitles.`"

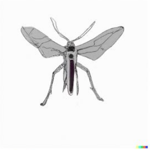
(b) Image generated with the prompt: "`Vicootes.`"

(c) Image generated with the prompt: "`Apoploe vesrreaitais.`"

(a) Prompt: "`Painting of Apoploe vesrreaitais`"

(b) Prompt: "`cartoon, Apoploe vesrreaitais`"

(c) Prompt: "`3-D rendering of Apoploe vesrreaitais`"



(d) Prompt: "`line art, Apoploe vesrreaitais`"

Figure: From Daras et al., 2022.

Challenge 3: Customizing & Control

- Finetuning the textual encoder: Textual inversion from Gal et al., 2022.
- Finetuning the visual decoder with Dreambooth from Ruiz et al., 2022 + details here.

Figure: Dreambooth example from Ruiz et al., 2022.

Challenge 4: multi-modality

Failures: incorrect relative scaling, physically impossible configuration.

Failure: improper handling of negation or indication of absence.

Figure: From Yu et al., 2022

Left: A shiny robot wearing a race car suit and black visor stands proudly in front of an F1 race car. The sun is setting on a cityscape in the background. comic book illustration.

Right: A plate that has no bananas on it. there is a glass without orange juice next to it.

Challenge 5: Filtering the data

- The main goal is to reduce graphic and explicit training data.
- But also to prevent image regurgitation.
- Stable Diffusion: unfortunately, the filter dramatically cut down on the number of people in the dataset and that meant folks had to work harder to get similar results generating people.

Unfiltered

Filtered

Figure: From From DALLE's blog: generations for the prompt "military protest" from our unfiltered model (left) and filtered model (right). Notably, the filtered model almost never produces images of guns.

Interesting links and open-source models

- LAION project: largest open source dataset and CLIP model here.
- Stable Diffusion project Stability.ai.
- A large list of examples for DALLE2 here.

Latest developments with Midjourney

AI wins art competitions

Figure: *Jason M. Allen via Midjourney*

Context: Mr. Allen submited one of his Midjourney creations to the Colorado State Fair, which had a division for "digital art/digitally manipulated photography."

Artists strike back

ARTIFICIAL INTELLIGENCE / TECH / CREATORS

AI art tools Stable Diffusion and Midjourney targeted with copyright lawsuit

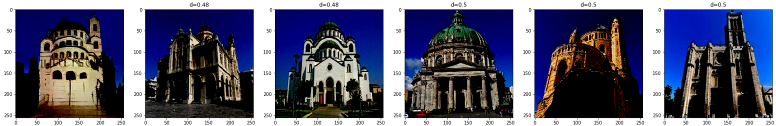
Home | Art in America | Features

Artists Are Suing Artificial Intelligence Companies and the Lawsuit Could Upend Legal Precedents Around Art

Artists and Illustrators Are Suing Three A.I. Art Generators for Scraping and 'Collaging' Their Work Without Consent

Outline

Are GANs memorizing the dataset?



Generated sample   Nearest neighbors in the training dataset using L2 distance

- Few shot learning regime: memorization doable...
- Huge dataset. $K \to \infty \implies$ underfitting, impossible to memorize ?

However, here is a recent observation:

Figure: Left: prompt "a beautiful green forest with a lake and snow -capped mountains in the background". Right: Banff, Alberta, Canada.

Are WGANs working because they fail ?

Figure: Left: $W(\mu_n, \tilde{\mu}_n) = 51.40$, Right: $W(\mu_n, \mu_n^k) = 40.15$ (k-means) (Stanczuk et al., 2021)

- Interesting properties of convolutional networks ??

$$\underset{\theta \in \Theta}{\operatorname{argmin}}\, d_{\mathscr{D}}(\mu_n, \mu_\theta) \neq \underset{\theta \in \Theta}{\operatorname{argmin}}\, d_{\mathsf{Lip}_1}(\mu_n, \mu_\theta).$$

- The discriminator punishes more samples out of the target manifold...
- Failure of the L_2 distance as a perceptual distance.

Are difusion models better than GANs?

2022 has seen **the rise** of Diffusion models.

- Better empirical results (FID). Is it fair?
- First, **large** difference in the datasets they were trained on.
- ImageNet (15 million) vs LAION (2-3 billions).
- GANs are **unstable**: there is a significant difference in the losses used. DMs might just be smoother.
- DMs are **easier to scale** than GANs.
- GANs may be still less understood than other models (room for improvement?).

No free lunch theorem ? Generative Trilemna

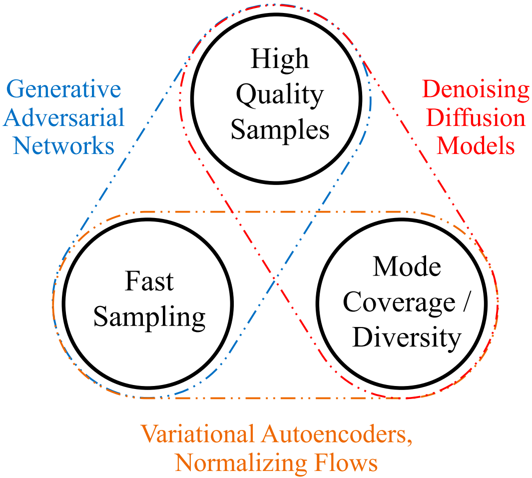


Figure: There is a trade-off between performance, inference speed, and the diversity Z. Xiao et al. (2022).

What generative models can be applied to text?

Large Language Models have also risen these past few years.

- OpenAI **GPT1**, **GPT2**, and **GPT3**.
- OpenAI **ChatGPT**
- All these models are **autoregressive likelihood-based** auto-regressive training objectives.
- Are latent spaces useful/necessary in this setting?

Some weak examples

- GANs are ill-posed to deal with discrete outputs: calls for reinforcement learning.
- MALIGANs Che et al., 2017, discriminative search (Scialom et al., 2020).
- Continuous (X. L. Li et al., 2022) or discrete (Reid et al., 2022) diffusion for text.

Thank you!

Any questions?

References I

Abdal, Rameen, Yipeng Qin, and Peter Wonka (2019). "Image2stylegan: How to embed images into the stylegan latent space?" In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4432–4441.

Arjovsky, Martin and Léon Bottou (2017). Towards Principled Methods for Training Generative Adversarial Networks. arXiv: 1701.04862 [stat.ML].

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein gan". In: arXiv preprint arXiv:1701.07875.

Arora, Sanjeev et al. (2017). "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". In: CoRR abs/1703.00573. arXiv: 1703.00573. URL: http://arxiv.org/abs/1703.00573.

Bansal, Arpit et al. (2022). "Cold diffusion: Inverting arbitrary image transforms without noise". In: arXiv preprint arXiv:2208.09392.

Béthune, Louis et al. (2022). "Pay attention to your loss: understanding misconceptions about Lipschitz neural networks". In: Advances in Neural Information Processing Systems.

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2019). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: International Conference on Learning Representations.

Che, Tong et al. (2017). "Maximum-likelihood augmented discrete generative adversarial networks". In: arXiv preprint arXiv:1702.07983.

References II

Crowson, Katherine et al. (2022). "Vqgan-clip: Open domain image generation and editing with natural language guidance". In: European Conference on Computer Vision. Springer, pp. 88–105.

Daras, Giannis and Alexandros G Dimakis (2022). "Discovering the Hidden Vocabulary of DALLE-2". In: arXiv preprint arXiv:2206.00169.

Esser, Patrick, Robin Rombach, and Bjorn Ommer (2021). "Taming transformers for high-resolution image synthesis". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883.

Gal, Rinon et al. (2022). "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: arXiv preprint arXiv:2208.01618.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: Advances in neural information processing systems, pp. 2672–2680.

— (2020). "Generative adversarial networks". In: Communications of the ACM 63.11, pp. 139–144.

Gulrajani, Ishaan et al. (2017). "Improved training of wasserstein gans". In: Advances in Neural Information Processing Systems, pp. 5767–5777.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: Advances in Neural Information Processing Systems. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 6840–6851.

References III

Isola, Phillip et al. (2017). "Image-to-image translation with conditional adversarial networks". In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134.

Issenhuth, Thibaut et al. (2021). "EdiBERT, a generative model for image editing". In: arXiv preprint arXiv:2111.15264.

Karras, Tero, Samuli Laine, and Timo Aila (2019). "A style-based generator architecture for generative adversarial networks". In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401–4410.

Khayatkhoei, Mahyar, Maneesh K Singh, and Ahmed Elgammal (2018). "Disconnected manifold learning for generative adversarial networks". In: Advances in Neural Information Processing Systems 31.

Ledig, Christian et al. (2017). "Photo-realistic single image super-resolution using a generative adversarial network". In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4681–4690.

Li, Xiang Lisa et al. (2022). "Diffusion-LM Improves Controllable Text Generation". In: Advances in Neural Information Processing Systems. Ed. by Alice H. Oh et al.

Liu, S., O. Bousquet, and K. Chaudhuri (2017). "Approximation and convergence properties of generative adversarial learning". In: Advances in Neural Information Processing Systems 30. Ed. by I. Guyon et al. Red Hook: Curran Associates, Inc., pp. 5551–5559.

References IV

Lugmayr, Andreas et al. (2020). "Srflow: Learning the super-resolution space with normalizing flow". In: European Conference on Computer Vision. Springer, pp. 715–732.

Luo, Calvin (2022). "Understanding diffusion models: A unified perspective". In: arXiv preprint arXiv:2208.11970.

Mao, X. et al. (2017). "Least Squares Generative Adversarial Networks". In: IEEE International Conference on Computer Vision.

Mirza, Mehdi and Simon Osindero (2014). "Conditional Generative Adversarial Nets". In: CoRR abs/1411.1784. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

Miyato, Takeru et al. (2018). "Spectral Normalization for Generative Adversarial Networks". In: International Conference on Learning Representations.

Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka (2016). "f-gan: Training generative neural samplers using variational divergence minimization". In: Advances in neural information processing systems, pp. 271–279.

Oord, Aäron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (June 2016). "Pixel Recurrent Neural Networks". In: Proceedings of The 33rd International Conference on Machine Learning. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1747–1756.

Ramesh, Aditya, Prafulla Dhariwal, et al. (2022). "Hierarchical text-conditional image generation with clip latents". In: arXiv preprint arXiv:2204.06125.

References V

Ramesh, Aditya, Mikhail Pavlov, et al. (2021). "Zero-shot text-to-image generation". In: International Conference on Machine Learning. PMLR, pp. 8821–8831.

Reid, Machel, Vincent J. Hellendoorn, and Graham Neubig (2022). "DiffusER: Discrete diffusion via edit-based reconstruction". In: arXiv preprint arXiv:2210.16886.

Rhijn, Jorino van et al. (2021). "Monte Carlo Simulation of SDEs using GANs". In: arXiv preprint arXiv:2104.01437.

Rombach, Robin et al. (2022). "High-resolution image synthesis with latent diffusion models". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695.

Roth, Kevin et al. (2017). "Stabilizing training of generative adversarial networks through regularization". In: Advances in neural information processing systems 30.

Ruiz, Nataniel et al. (2022). "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: arXiv preprint arXiv:2208.12242.

Saharia, Chitwan et al. (2022). "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: arXiv preprint arXiv:2205.11487.

Samangouei, Pouya, Maya Kabkab, and Rama Chellappa (2018). "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models". In: International Conference on Learning Representations.

References VI

Scialom, Thomas et al. (July 2020). "Discriminative Adversarial Search for Abstractive Summarization". In: Proceedings of the 37th International Conference on Machine Learning. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 8555–8564.

Shao, Hang, Abhishek Kumar, and P Thomas Fletcher (2018). "The riemannian geometry of deep generative models". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Worksh pp. 315–323.

Sønderby, Casper Kaae et al. (2017). "Amortised MAP Inference for Image Super-resolution". In: International Conference on Learning Representations.

Song, Yang and Stefano Ermon (2019). "Generative Modeling by Estimating Gradients of the Data Distribution". In: Advances in Neural Information Processing Systems. Vol. 32. Curran Associates, Inc.

Song, Yang, Jascha Sohl-Dickstein, et al. (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: International Conference on Learning Representations.

Stanczuk, Jan et al. (2021). "Wasserstein GANs work because they fail (to approximate the Wasserstein distance)". In: arXiv preprint arXiv:2103.01678.

Takahashi, Shuntaro, Yu Chen, and Kumiko Tanaka-Ishii (2019). "Modeling financial time-series with generative adversarial networks". In: Physica A: Statistical Mechanics and its Applications 527, p. 121261. ISSN: 0378-4371.

References VII

Tanielian, Ugo and Gerard Biau (2021). "Approximating Lipschitz continuous functions with GroupSort neural networks". In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 442–450.

Tanielian, Ugo, Thibaut Issenhuth, et al. (2020). "Learning disconnected manifolds: a no gan's land". In: International Conference on Machine Learning. PMLR, pp. 9418–9427.

Van Den Oord, Aaron, Oriol Vinyals, et al. (2017). "Neural discrete representation learning". In: Advances in neural information processing systems 30.

Wang, Ting-Chun et al. (2018). "High-resolution image synthesis and semantic manipulation with conditional gans". In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8798–8807.

Wang, Zhendong et al. (2022). "Diffusion-GAN: Training GANs with Diffusion". In: arXiv preprint arXiv:2206.02262.

Wu, Huikai et al. (2019). "Gp-gan: Towards realistic high-resolution image blending". In: Proceedings of the 27th ACM international conference on multimedia, pp. 2487–2495.

Xiao, Chaowei et al. (2018). "Generating adversarial examples with adversarial networks". In: arXiv preprint arXiv:1801.02610.

Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat (2022). "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs". In: International Conference on Learning Representations.

References VIII

Xingyu, Zhou et al. (2018). "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets". In: Mathematical Problems in Engineering 2018, p. 11.

Yang, Liu, Dongkun Zhang, and George Em Karniadakis (2018). "Physics-informed generative adversarial networks for stochastic differential equations". In: arXiv preprint arXiv:1811.02033.

Yu, Jiahui et al. (2022). "Scaling autoregressive models for content-rich text-to-image generation". In: arXiv preprint arXiv:2206.10789.

Zhang, Han et al. (2017). "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: Proceedings of the IEEE international conference on computer vision, pp. 5907–5915.

Zhu, Jun-Yan et al. (Oct. 2017). "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks". In: Proceedings of the IEEE International Conference on Computer Vision (ICCV).