



UNIVERSIDAD DE CHILE

Inteligencia Artificial Generativa

Let's talk about hype stuff

Valentin Barriere // Clemente Henriquez

Universidad de Chile – DCC

Diplomado de Postítulo en Inteligencia Artificial, Primavera 2025

Introducción y Modelos de Visión generativos

Outline : Modelos Generativos

Modelos Generativos

Introducción

Tipos de Modelos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

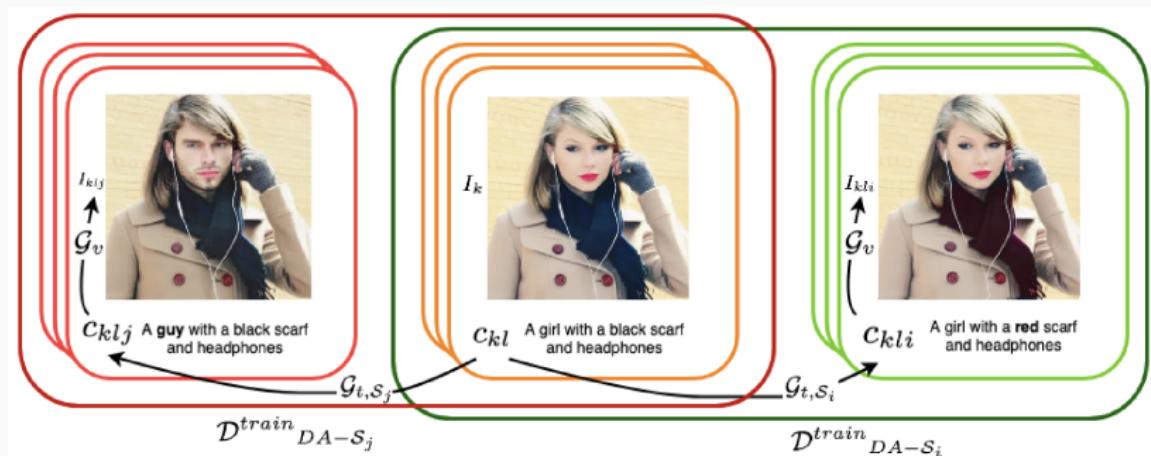
Otros

Conclusión

¿Por qué necesitamos modelos generativos?

La IA no es solo toma de decisiones — también es creación.

- Los modelos generativos permiten que las máquines **creén** datos realistas: imágenes, texto, audio, etc...: aquí [un ejemplo con StyleGan2](#)
- Permiten:
 - Aumento de datos
 - Generación creativa de contenido
 - Simulación y Stress test de IA usando contrafácticos



¿Por qué necesitamos modelos generativos?

- **Idea crítica:** Un modelo generativo puede engañar a uno discriminativo.
- Ejemplo: los GANs generan imágenes falsas que los clasificadores etiquetan con confianza — exponiendo vulnerabilidades en los sistemas de decisión.
- Esta interrelación impulsa avances en robustez, seguridad y evaluación para **construir IA confiable**.



$p(y = \text{cat}|\mathbf{x}) = 0.90$
 $p(y = \text{dog}|\mathbf{x}) = 0.05$
 $p(y = \text{horse}|\mathbf{x}) = 0.05$



noise



$p(y = \text{cat}|\mathbf{x}) = 0.05$
 $p(y = \text{dog}|\mathbf{x}) = 0.05$
 $p(y = \text{horse}|\mathbf{x}) = 0.90$

Figure 2: Agregar ruido resulta en un cambio de la etiqueta predicha.

Modelos Generativos vs. Discriminativos

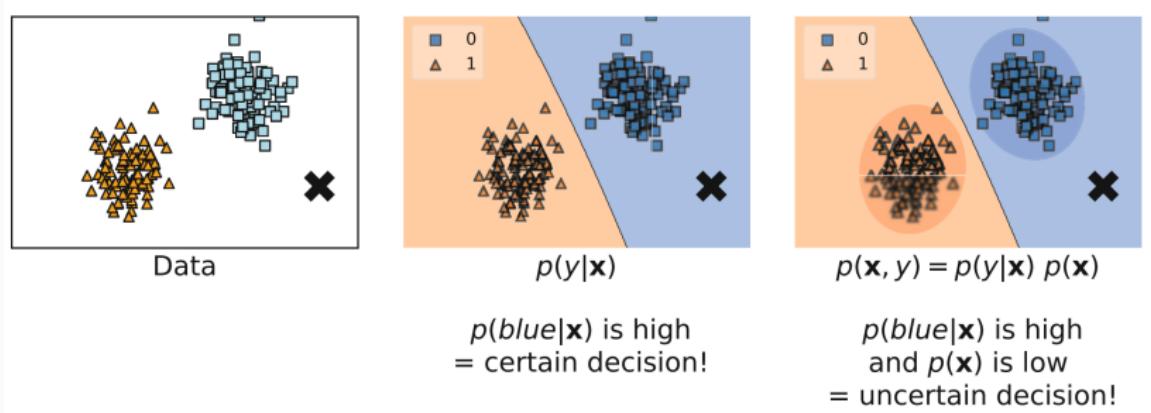


Figure 3: Ejemplo de datos y dos enfoques para la toma de decisiones: un enfoque discriminativo y un enfoque generativo.

- **Discriminativo:** Aprenden $p(y|x)$: “Dado el input, ¿qué etiqueta?”
- **Generativo:** Aprenden $p(x,y)$ o $p(x)$: “¿Cómo se genera la data?”

Modelos Generativos vs. Discriminativos

Modelos Discriminativos	Modelos Generativos
Enfoque en la frontera de decisión	Modelan la distribución completa de los datos
Ejemplos: Regresión Logística, clasificadores CNN	Ejemplos: Naive Bayes, VAEs, GANs, modelos AR
Alta precisión para clasificación	Pueden generar muestras, manejar datos faltantes, detectar anomalías
No pueden generar nuevos datos	Más complejos de entrenar

Idea clave:

Los modelos generativos preguntan: “*¿Cómo se ve el mundo?*”

Los modelos discriminativos preguntan: “*¿Qué es esta cosa específica?*”

Modelos Generativos vs. Discriminativos

Modelos Discriminativos	Modelos Generativos
Enfoque en la frontera de decisión	Modelan la distribución completa de los datos
Ejemplos: Regresión Logística, clasificadores CNN	Ejemplos: Naive Bayes, VAEs, GANs, modelos AR
Alta precisión para clasificación	Pueden generar muestras, manejar datos faltantes, detectar anomalías
No pueden generar nuevos datos	Más complejos de entrenar

Idea clave:

Los modelos generativos preguntan: “*¿Cómo se ve el mundo?*”

Los modelos discriminativos preguntan: “*¿Qué es esta cosa específica?*”

Los modelos generativos son más ambiciosos — modelan el proceso completo de creación de datos.

Aplicaciones de Modelos Generativos Profundos

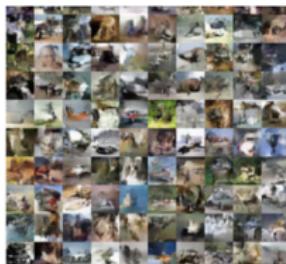
¿Dónde se usan? Impacto en el mundo real

Los modelos generativos profundos están transformando múltiples dominios:

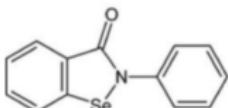
“ i want to talk to you . ”
“i want to be with you . ”
“i do n't want to be with you . ”
i do n't want to be with you .
she did n't want to be with him .

he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

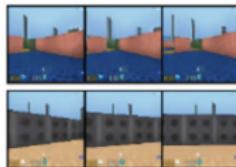
Text



Images



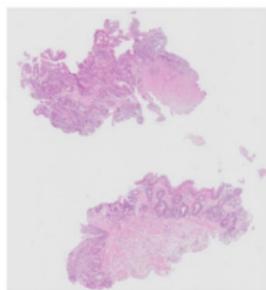
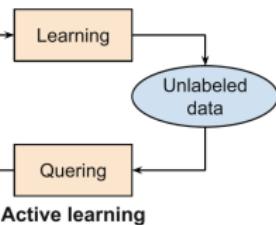
Graphs



Reinforcement learning



Audio



Medical imaging

Figure 4: Descubrimiento de fármacos, DeepFake, Contrafácticos, Agentes conversacionales, Generación de música.

Aplicaciones de Modelos Generativos Profundos

¿Dónde se usan? Impacto en el mundo real

Los modelos generativos profundos están transformando múltiples dominios:

- **Imagen:** Super-resolución, inpainting, transferencia de estilo (p. ej., DALL-E, Stable Diffusion)
- **Texto:** Generación de historias, sistemas de diálogo, síntesis de código (p. ej., GPT)
- **Audio:** Clonación de voz, generación de música (p. ej., Jukebox)
- **Video:** Predicción, animación, deepfakes
- **Descubrimiento Científico:**
 - Generar nuevas moléculas con potencial farmacéutico (graph VAEs, difusión en grafos)
 - Simular sistemas físicos
- **Detección de Anomalías:** Aprender la data normal y señalar outliers (p. ej., en imágenes médicas)

Aplicaciones de Modelos Generativos Profundos

¿Dónde se usan? Impacto en el mundo real

Los modelos generativos profundos están transformando múltiples dominios:

- **Imagen:** Super-resolución, inpainting, transferencia de estilo (p. ej., DALL-E, Stable Diffusion)
- **Texto:** Generación de historias, sistemas de diálogo, síntesis de código (p. ej., GPT)
- **Audio:** Clonación de voz, generación de música (p. ej., Jukebox)
- **Video:** Predicción, animación, deepfakes
- **Descubrimiento Científico:**
 - Generar nuevas moléculas con potencial farmacéutico (graph VAEs, difusión en grafos)
 - Simular sistemas físicos
- **Detección de Anomalías:** Aprender la data normal y señalar outliers (p. ej., en imágenes médicas)

La IA generativa no es solo arte — es una herramienta para la ciencia, la medicina y la ingeniería.

Modelos Autorregresivos (ARMs)

Construyendo los datos paso a paso

- Idea central: Factorizar la distribución conjunta secuencialmente:

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^n p(x_i | x_{<i})$$

- Predecir cada elemento condicionado a los anteriores — estrictamente secuencial.

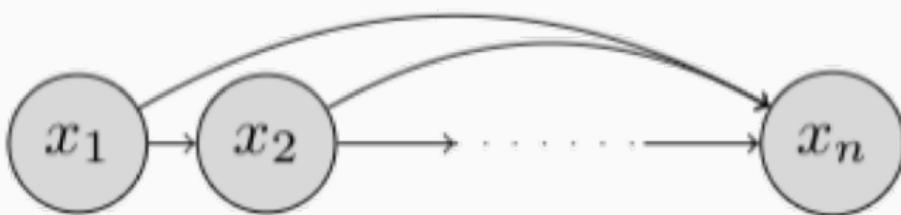


Figure 4: Factorización por la regla de la cadena expresada gráficamente como una red bayesiana.

Modelos Autorregresivos (ARMs)

Construyendo los datos paso a paso

- Idea central: Factorizar la distribución conjunta secuencialmente:

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^n p(x_i | x_{<i})$$

- Predecir cada elemento condicionado a los anteriores — estrictamente secuencial.

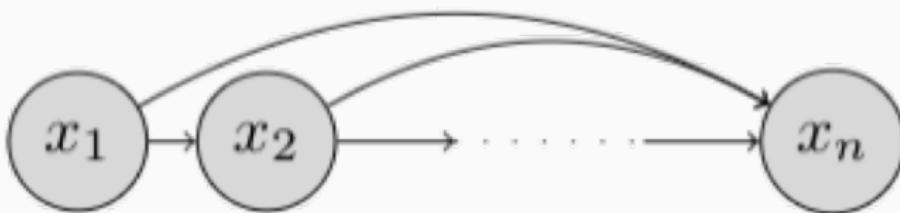


Figure 4: Factorización por la regla de la cadena expresada gráficamente como una red bayesiana.

Básicamente, usas los valores pasados para generar los futuros, y así sucesivamente.

Modelos Autorregresivos (ARMs)

Construyendo los datos paso a paso

- Idea central: Factorizar la distribución conjunta secuencialmente:

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^n p(x_i | x_{<i})$$

- Predecir cada elemento condicionado a los anteriores — estrictamente secuencial.
- Ejemplos:**

- Convoluciones causales en imágenes (PixelCNN, PixelRNN) o audio (Wavenet)
- Atención causal en Transformers (GPT)
- [Más información aquí](#)

Pros:

- Verosimilitud exacta
- Compresión sin pérdida
- Entrenamiento estable

Contras:

- Muestreo lento (sin paralelización)
- La calidad depende del orden de generación

Modelos de Variables Latentes: VAE, Difusión, GANs

Estos modelos asumen que los datos x se generan a partir de variables latentes z :

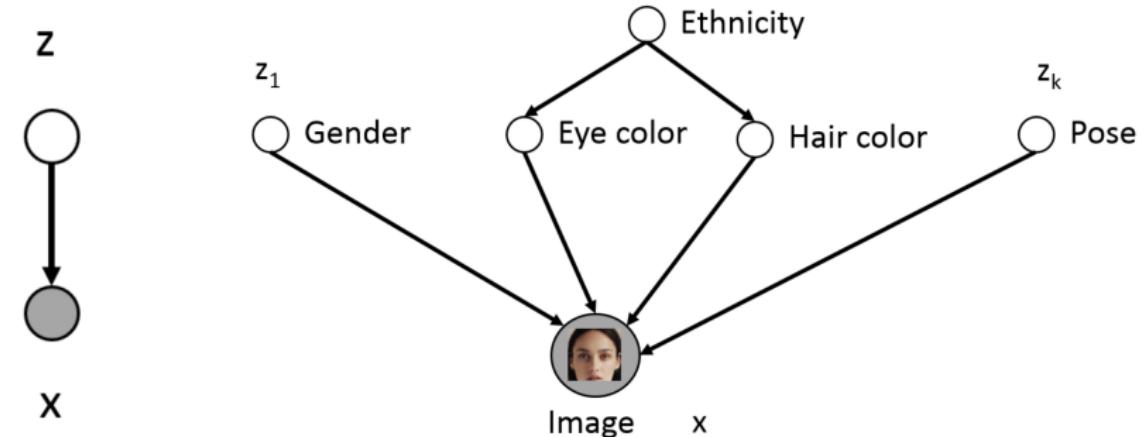
$$x \sim p(x|z), \quad z \sim p(z)$$

Motivación

- Gran variabilidad en las imágenes x debido a género, color de ojos, color de pelo, pose, etc. Sin embargo, a menos que las imágenes estén anotadas, estos factores de variación no están explícitamente disponibles (latentes).
- **Idea:** modelar explícitamente estos factores usando variables latentes z



Modelos de Variables Latentes: VAE, Difusión, GANs



- Solo las variables sombreadas x se observan en los datos (valores de píxeles). Las variables latentes z corresponden a características de alto nivel.
 - Si z se elige correctamente, $p(x|z)$ podría ser mucho más simple que $p(x)$.
 - Si hubiéramos entrenado este modelo, entonces podríamos identificar características vía $p(z|x)$, por ejemplo, $p(\text{ColorOjos} = \text{Azul}|x)$

Modelos de Variables Latentes: VAE, Difusión, GANs

VAE

Autoencoder Variacional:
aprende un codificador y
decodificador
probabilísticos. Maximiza
ELBO. Bueno para espacios
latentes estructurados.

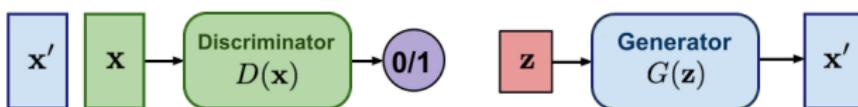
Difusión

Añade ruido gradualmente
y luego invierte el proceso.
Calidad de vanguardia
mediante score matching.

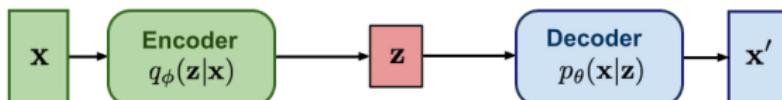
GAN

Entrenamiento adversarial:
generador contra
discriminador. Sin densidad
explícita. Muestreo rápido,
salidas nítidas.

GAN: Adversarial
training



VAE: maximize
variational lower bound



Diffusion models:

Gradually add Gaussian
noise and then reverse



Outline : Modelos Autorregresivos

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

Otros

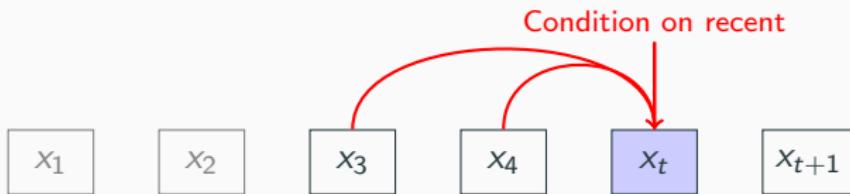
Conclusión

Modelos Autorregresivos: Memoria finita y Contexto

Un modelo autorregresivo (AR) genera datos un paso a la vez:

$$p(x) = \prod_{t=1}^T p(x_t | x_{<t}), \quad x_{<t} = (x_1, \dots, x_{t-1})$$

Pero en la práctica, los modelos tienen **memoria finita** — no pueden condicionar sobre todo el pasado. El pasado distante se ignora mediante una ventana de contexto.



El pasado distante se ignora: $p(x_t | x_{<t}) \approx p(x_t | x_{t-k}, \dots, x_{t-1})$

Ventana de contexto: Solo los últimos k tokens se usan en la memoria:

$$p(x) = p(x_1)p(x_2 | x_1) \prod_{d=3}^D p(x_d | x_{d-1}, x_{d-2})$$

Desafío: Cómo modelar dependencias largas de forma eficiente?

Memoria de Largo Alcance mediante RNNs

$$p(x_d | x_{<d}) = p(x_d | h_d) = p(x_d | \text{RNN}(x_{d-1}, h_{d-1}))$$

Las RNNs mantienen un **estado oculto** h_t que resume el pasado:

$$h_t = f_\theta(h_{t-1}, x_{t-1}), \quad \hat{x}_t = g_\theta(h_t)$$

h_d es un contexto oculto que actúa como memoria y permite aprender dependencias de largo alcance.

Pros:

- Naturalmente autorregresivas y causales
- Pueden, en teoría, recordar toda la historia

Contras:

- Gradientes que se desvanecen
→ difícil aprender dependencias a largo plazo
- Secuenciales: no se puede paralelizar el entrenamiento

Comparación Memoria Finita vs RNN

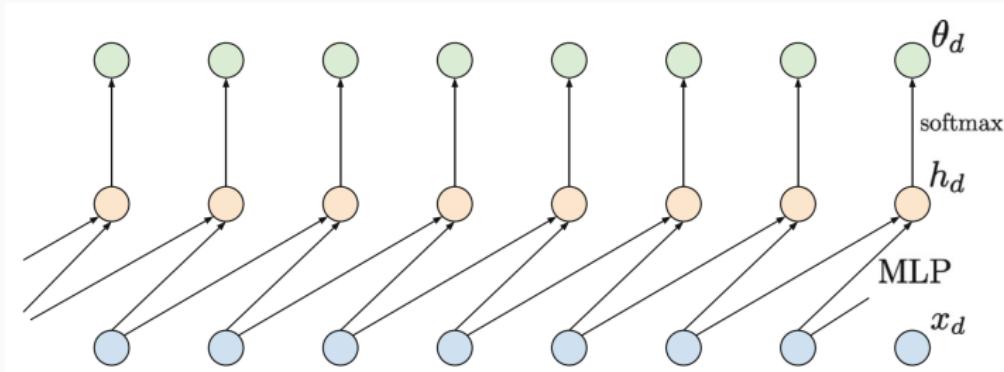


Figure 5: MLP dependiendo de 2 últimas entradas. Proba θ_d no depende de x_d .

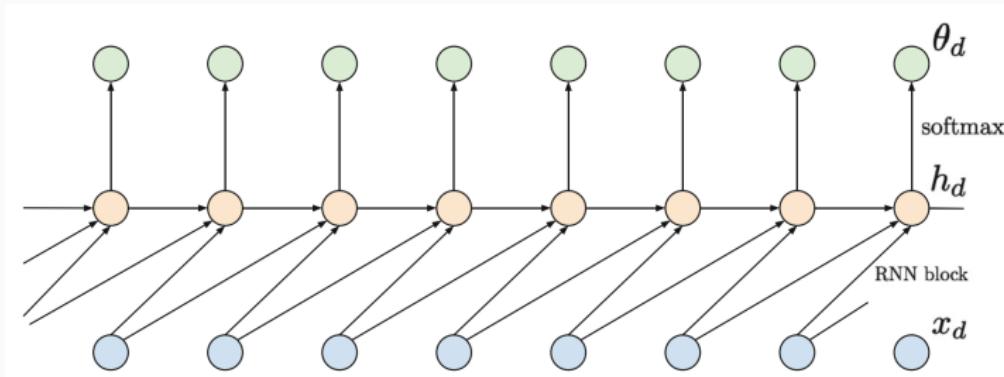


Figure 6: Igual con RNN, añadiendo dependencia adicional entre h_d .

Redes de Convolución Causales: Autorregresión Paralela

Las CNN causales aplican convoluciones **sólo sobre entradas pasadas**, preservando la autorregresión: $h_t = \sigma \left(\sum_{k=1}^K W_k x_{t-k} + b \right)$

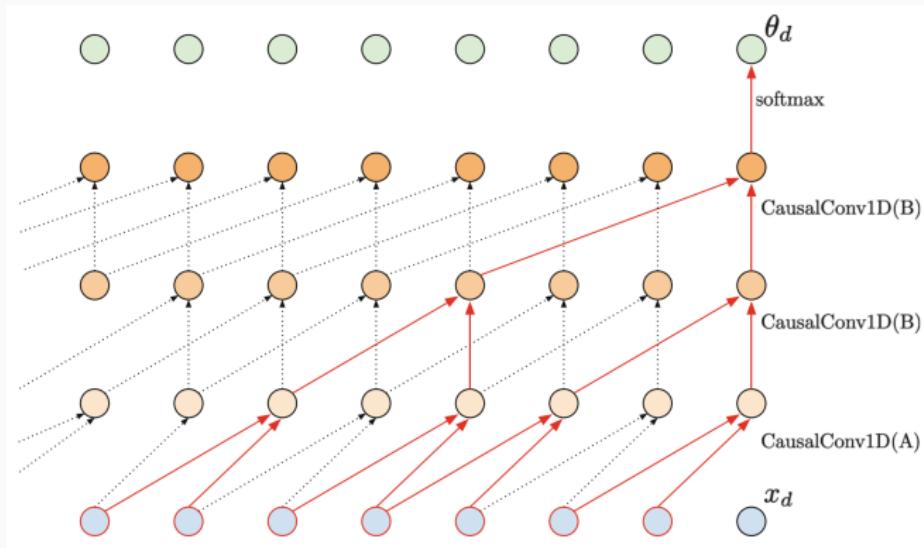


Figure 7: Convoluciones causales con tamaño de kernel 2, pero aplicando dilatación en capas superiores se puede procesar una entrada mucho mayor (bordes rojos); por tanto, se utiliza una memoria mayor.

Redes de Convolución Causales: Autorregresión Paralela

Ventajas de convoluciones causales apiladas

- **Convoluciones dilatadas:** campo receptivo crece exponencialmente
- **Paralelizables** en el tiempo: entrenamiento rápido
- **Causales:** sin filtración del futuro

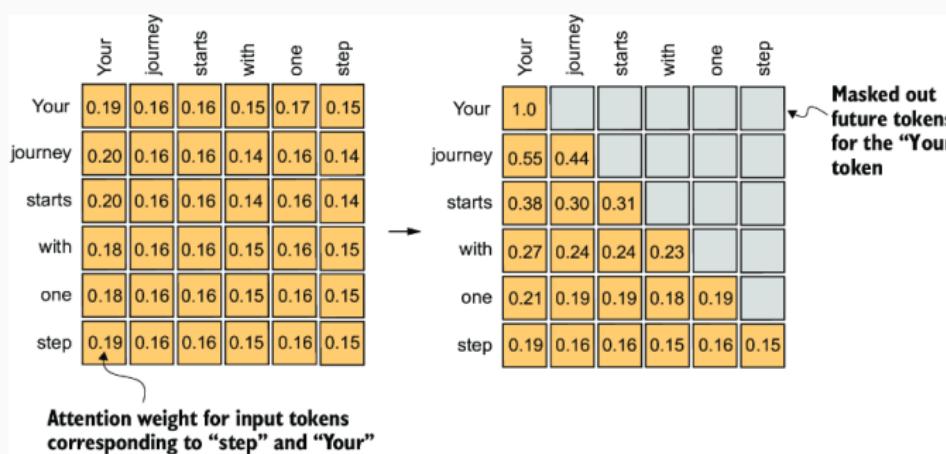
Atención Causal: Transformers para Modelado Autorregresivo

Los Transformers usan **self-attention**:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Pero para ser autorregresivos, aplicamos una **máscara causal**:

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \underbrace{M}_{\text{máscara}}\right), \quad M_{ij} = \begin{cases} 0 & i \geq j \\ -\infty & i < j \end{cases}$$



- **Contexto global:** atiende a todos los tokens previos
- **Entrenamiento paralelo:** a diferencia de las RNNs
- **Ponderación flexible:** aprende qué recordar

Outline : Autoencoders Variacionales

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

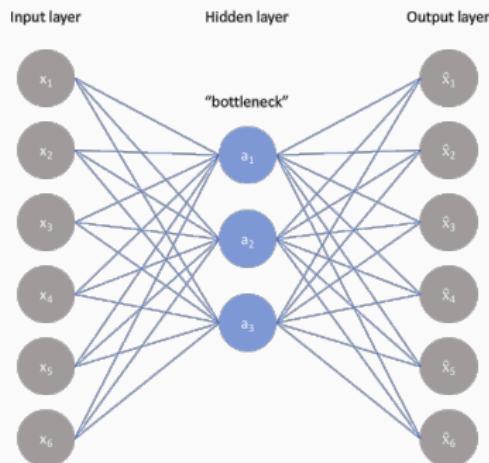
Otros

Conclusión

Autoencoder: Principio

Principio

El principio de un autoencoder es encontrar una representación distribuida de dimensión inferior **a** que permita reconstruir la entrada original **X**, conteniendo así **la mayor cantidad de información posible**.



Autoencoder: Entrenamiento

Entrenamiento no supervisado

El autoencoder se entrena de manera no supervisada usando una función de pérdida basada en la reconstrucción $\hat{\mathbf{X}}$ de la entrada original \mathbf{X} : $\ell(\mathbf{X}) = \|\mathbf{X} - \hat{\mathbf{X}}\|_2$.

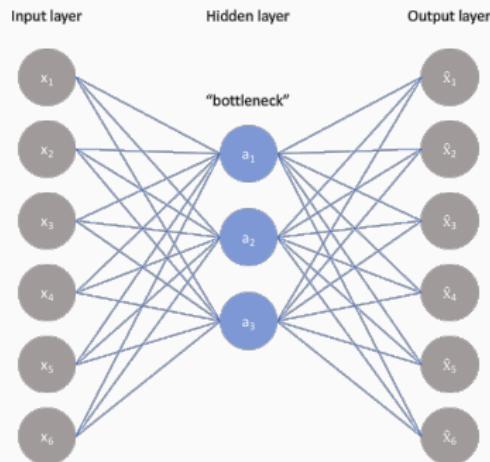
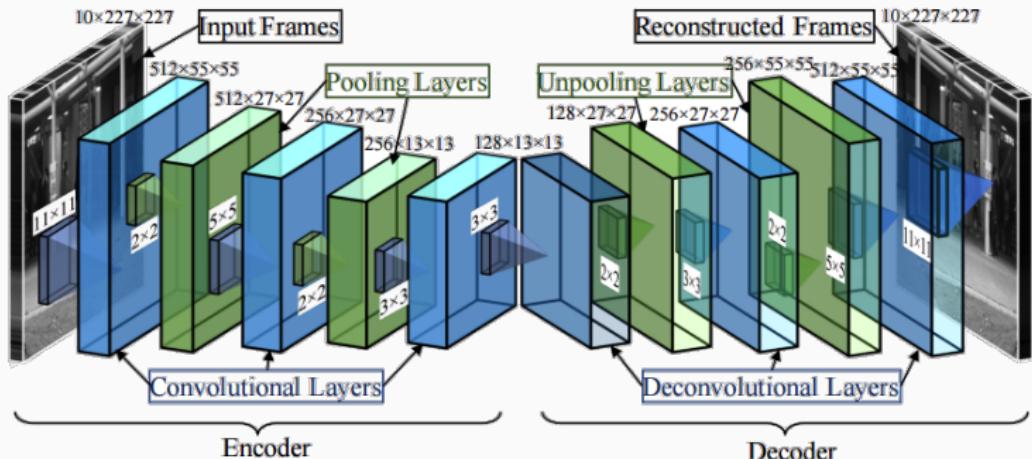


Figure 8: Ejemplo de un AE de una sola capa; se pueden usar cualquier número de capas

Autoencoder: Tipos de Codificadores

Cualquier tipo de codificador

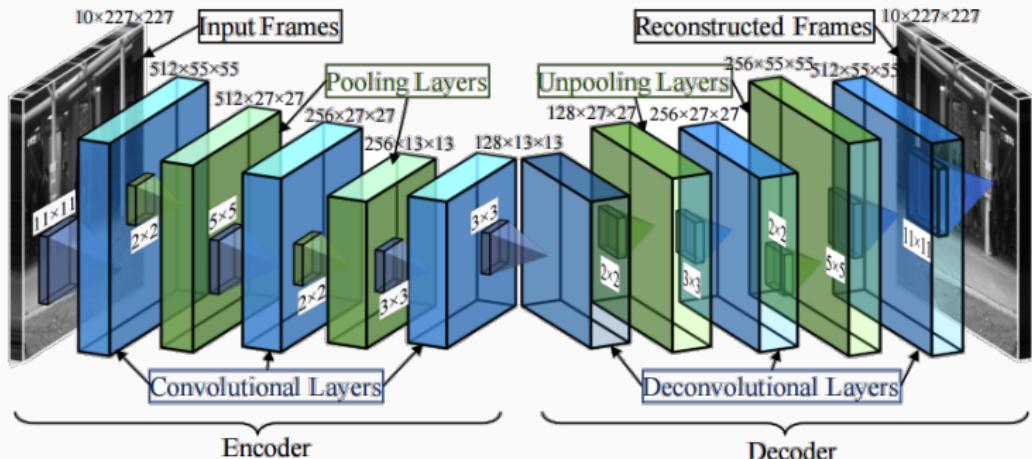
El codificador y el decodificador pueden ser más que perceptrones multicapa—también pueden ser CNNs o RNNs.



Autoencoder: Tipos de Codificadores

Cualquier tipo de codificador

El codificador y el decodificador pueden ser más que perceptrones multicapa—también pueden ser CNNs o RNNs.

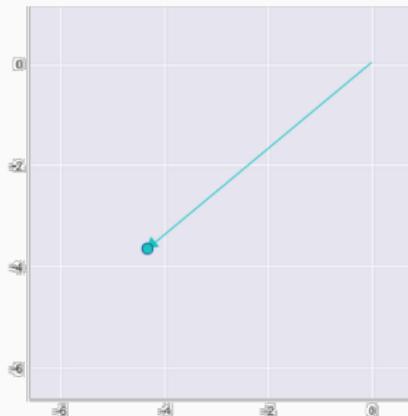


Problema: los AE son muy buenos en compresión pero malos para generación

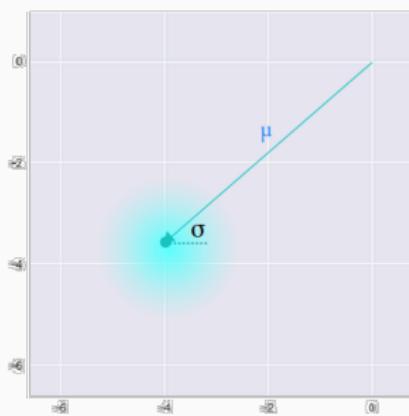
Autoencoder Variacional

Ventaja

Esto permite que el espacio latente esté "suavizado" entre diferentes puntos, posibilitando transiciones significativas de un punto a otro y generando nuevos datos coherentes.



Standard Autoencoder
(direct encoding coordinates)

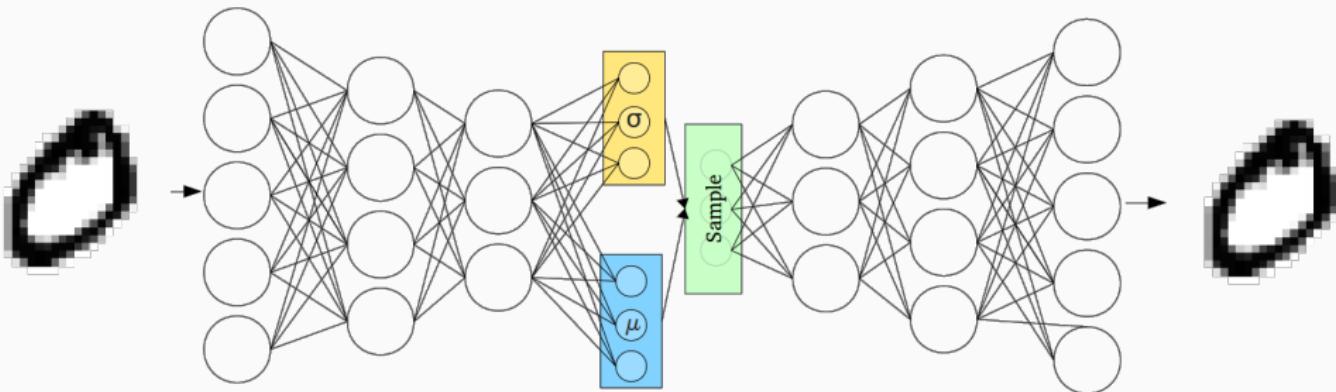


Variational Autoencoder
(μ and σ initialize a probability distribution)

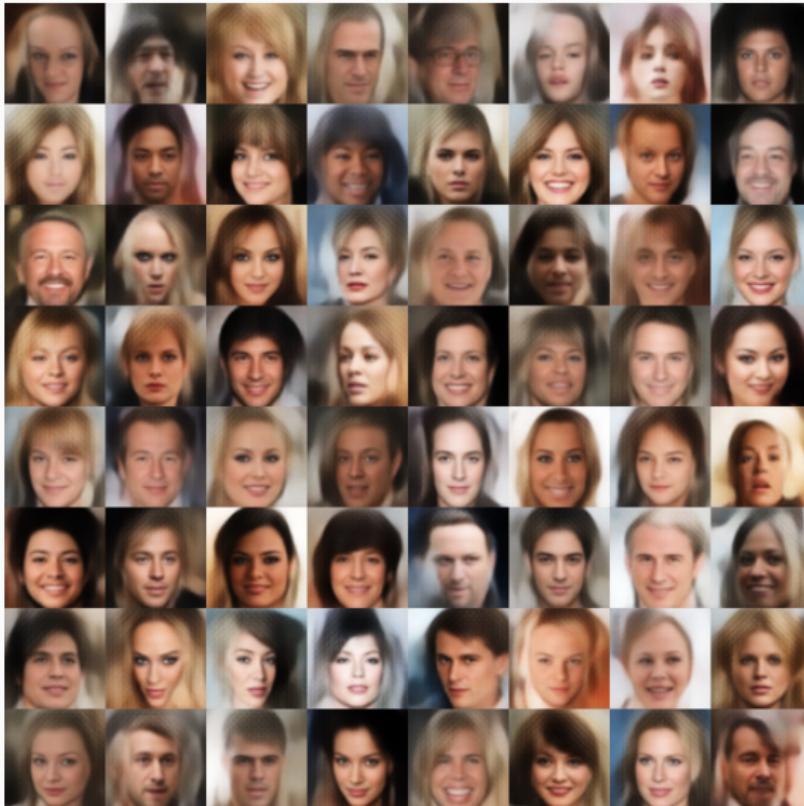
Autoencoder Variacional

Principio

El VAE ya no codifica los datos como un único punto en el espacio, sino como una distribución gaussiana con media μ y desviación estándar σ .

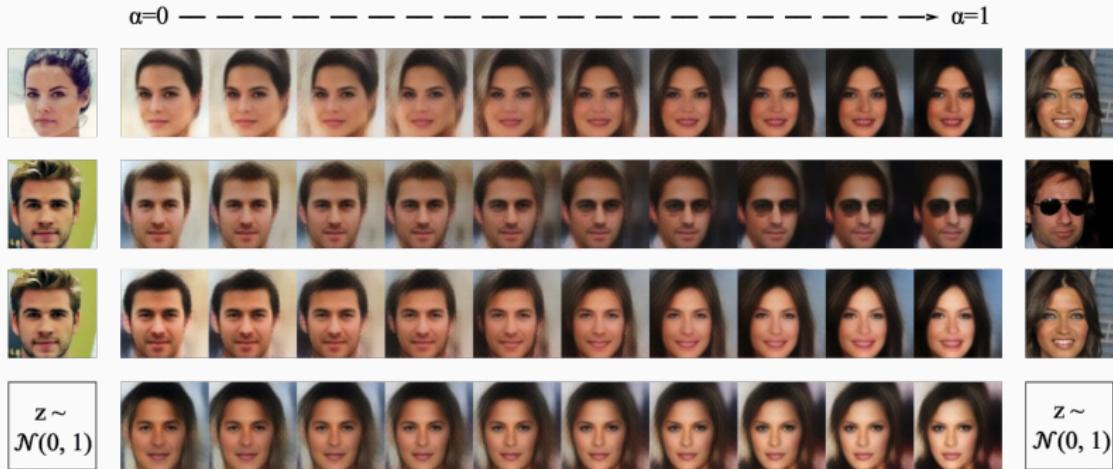


Autoencoder Variacional: Aplicación de ejemplo



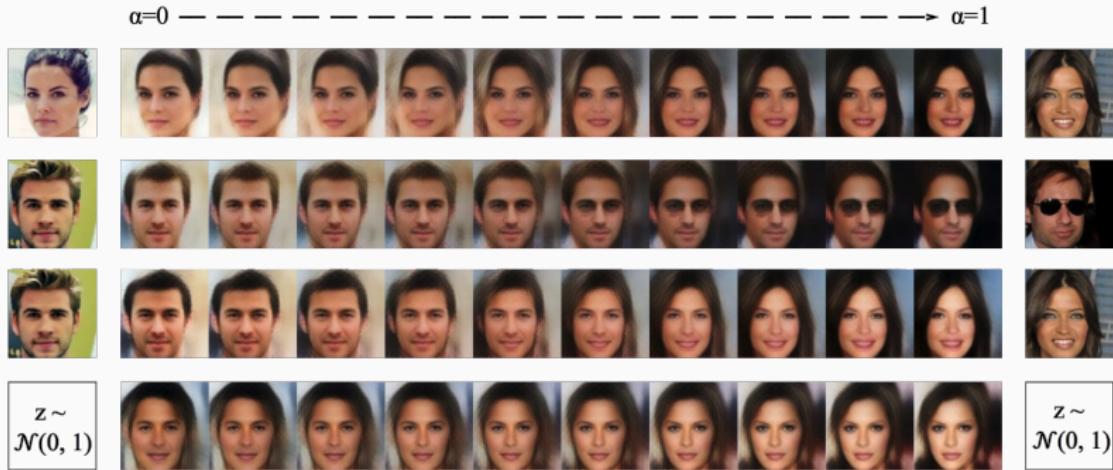
Autoencoder Variacional: Aplicación de ejemplo

Transición suave de una imagen a otra:



Autoencoder Variacional: Aplicación de ejemplo

Transición suave de una imagen a otra:



Excelente para tareas donde un espacio latente bien estructurado es importante, como: exploración de datos, detección de anomalías o generar muestras de datos diversas.

El problema: Posterior intratable

Queremos modelar datos x usando variables latentes z :

$$p_{\theta}(x) = \int p_{\theta}(x|z)p(z) dz$$

Pero para aprender buenas representaciones, también necesitamos el posterior:

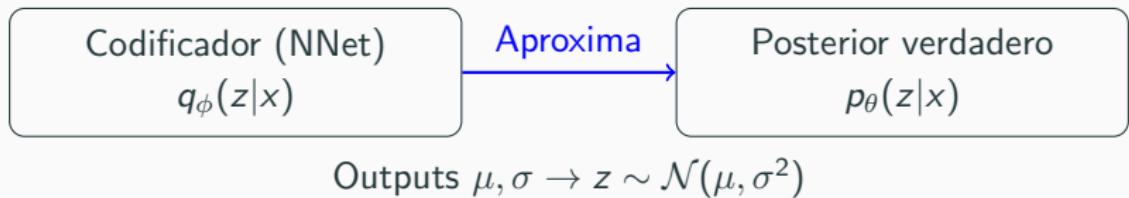
$$p_{\theta}(z|x) = \frac{p_{\theta}(x|z)p(z)}{p_{\theta}(x)}$$

Problema: $p_{\theta}(x)$ (**la verosimilitud marginal**) es intratable

- Requiere integrar sobre todos los posibles z
- Para z de alta dimensión y decodificadores complejos, esto es imposible.
- Necesitamos una aproximación

La solución: Inferencia variacional con $q_\phi(z|x)$

Introducimos una **distribución variacional** $q_\phi(z|x)$ — una aproximación al posterior verdadero.



Nuestro objetivo: hacer $q_\phi(z|x)$ cercano a $p_\theta(z|x)$ minimizando:

$$\text{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$$

Pero este KL todavía depende de $p_\theta(x)$... Así que lo reescribimos:

$$\log p_\theta(x) =$$

$$\underbrace{\mathbb{E}_q [\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) \parallel p(z))}_{\text{ELBO: lo que maximizamos}} + \underbrace{\text{KL}(q_\phi(z|x) \parallel p_\theta(z|x))}_{\text{siempre} > 0}$$

Sabiendo que $p_\theta(x) < 1 \Rightarrow \log p_\theta(x) < 0$, ELBO es el objetivo.

ELBO: Un objetivo computable

La **Evidencia Lower BOund (ELBO)** es:

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]}_{\text{Reconstrucción}} - \underbrace{\text{KL} (q_\phi(z|x) \parallel p(z))}_{\text{Regularización latente}}$$

Término de Reconstrucción

Maximizar esto asegura que el decodificador pueda recrear x desde z .

Término de Divergencia KL

Obliga a que $q_\phi(z|x)$ coincida con la prior $p(z) = \mathcal{N}(0, 1)$.

Idea clave: Maximizar ELBO:

- Mejora el ajuste a los datos
- Hace que $q_\phi(z|x)$ se acerque a $p_\theta(z|x)$
- Regulariza el espacio latente para un muestreo significativo

¿Por qué regularizar $q_\phi(z|x)$ hacia $\mathcal{N}(0, 1)$?

Alguien podría decir: "El codificador debe devolver $\mu = 0$, $\sigma = 1$ "

No: Puede devolver cualquier μ, σ — pero paga un **costo KL** si se desvían.

¿Por qué hacer esto?

- Asegura que el **decodificador** aprenda a interpretar $z \sim \mathcal{N}(0, 1)$
- Habilita el **muestreo**: generar nuevos datos con $z \sim \mathcal{N}(0, 1)$, luego $x = \text{decode}(z)$
- Previene el **colapso del posterior**: el codificador no ignora la prior
- Crea un **espacio latente suave y estructurado** para interpolación

¿Por qué regularizar $q_\phi(z|x)$ hacia $\mathcal{N}(0, 1)$?

Alguien podría decir: "El codificador debe devolver $\mu = 0$, $\sigma = 1$ "

No: Puede devolver cualquier μ, σ — pero paga un **costo KL** si se desvían.

¿Por qué hacer esto?

- Asegura que el **decodificador** aprenda a interpretar $z \sim \mathcal{N}(0, 1)$
- Habilita el **muestreo**: generar nuevos datos con $z \sim \mathcal{N}(0, 1)$, luego $x = \text{decode}(z)$
- Previene el **colapso del posterior**: el codificador no ignora la prior
- Crea un **espacio latente suave y estructurado** para interpolación

Compromiso: pérdida KL vs. reconstrucción — el ELBO los balancea.

Outline : Modelos de Difusión

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

Otros

Conclusión

Difusión: Nuevos avances en texto-a-imagen I

Modelos de difusión permiten generar imágenes condicionadas en texto

Demo [Stable Diffusion 3 Medium](#)

Learn more about the [Stable Diffusion 3 series](#). Try on [StabilityAI API](#), [Stable Assistant](#), or on Discord via [Stable Artisan](#). Run locally with [ComfyUI](#) or [diffusers](#)

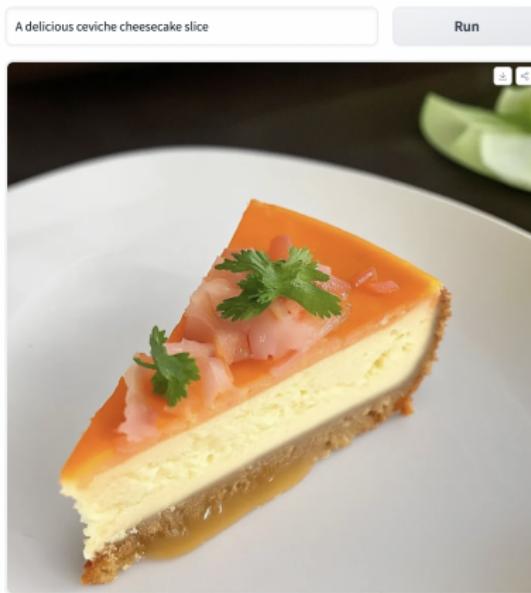


Figure 9: Ejemplos de imágenes divertidas obtenidas con modelos de difusión

Difusión: Nuevos avances en texto-a-imagen II

Definición

Modelos probabilísticos que aprenden la distribución de los datos modelando la reversa de un proceso de difusión (añadir ruido paso a paso) para generar nuevos puntos de datos.

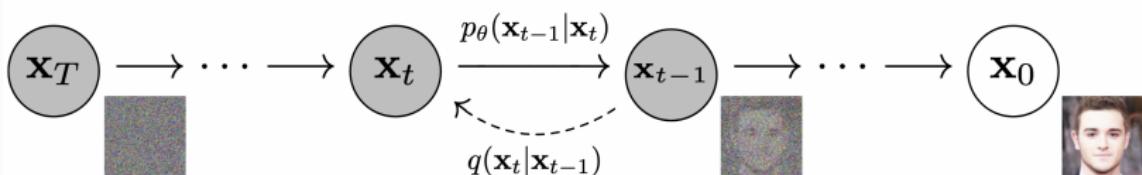


Figure 10: Eliminando ruido de una imagen [2]

Proceso

- **Forward (directo):** Añadir pequeñas cantidades de ruido a los datos, que gradualmente se vuelven más ruidosos con los pasos de tiempo, hasta parecer ruido aleatorio.
- **Reverse (inverso):** Aprender a eliminar el ruido gradualmente para recuperar los datos originales, empezando desde ruido puro y generando datos realistas.

Ejemplo de código de generación con un modelo de difusión



```
1 import torch
2 from diffusers import DDPMPipeline
3
4 # Establece el dispositivo para usar nuestra GPU o CPU
5 device = get_device()
6
7 # Carga la pipeline
8 image_pipe =
    → DDPMPipeline.from_pretrained("google/ddpm-celebahq-256")
9 image_pipe.to(device)
10
11 # Muestra una imagen
12 image_pipe().images[0]
```

Un ejemplo de código en un notebook de Colab:

[disponible aquí](#). Nótese que es mucho más rápido si se genera usando cuantización.



Encoder-Decoder U-net

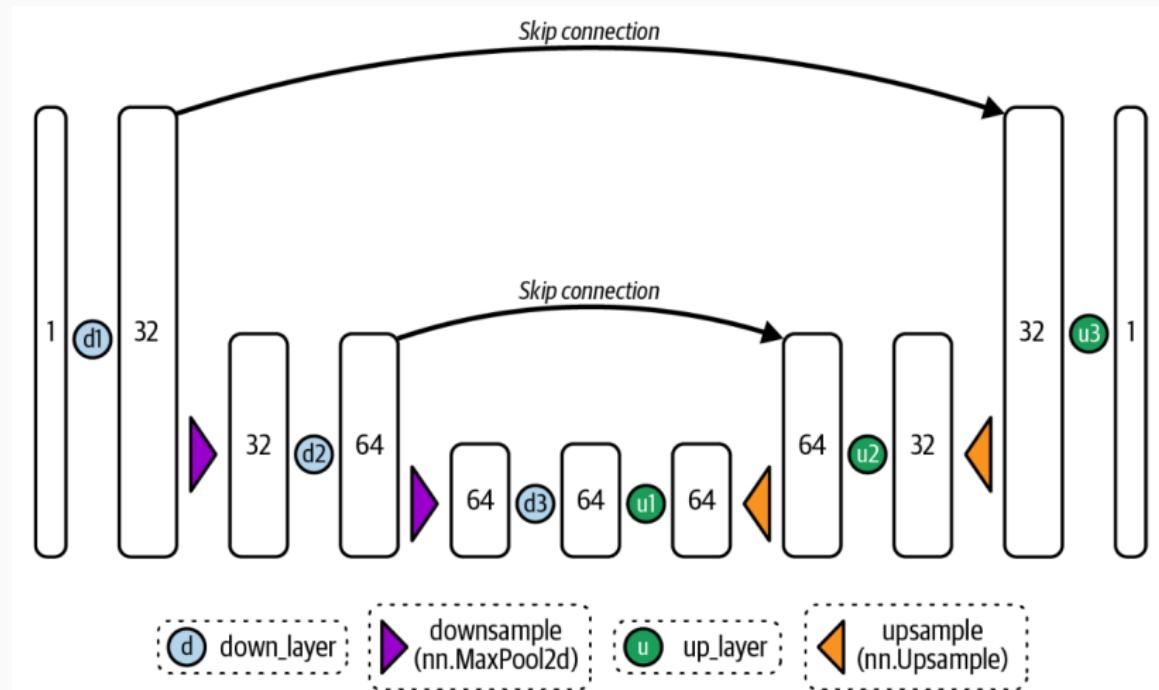


Figure 11: Arquitectura U-net [6]

Difusión: Stable Diffusion

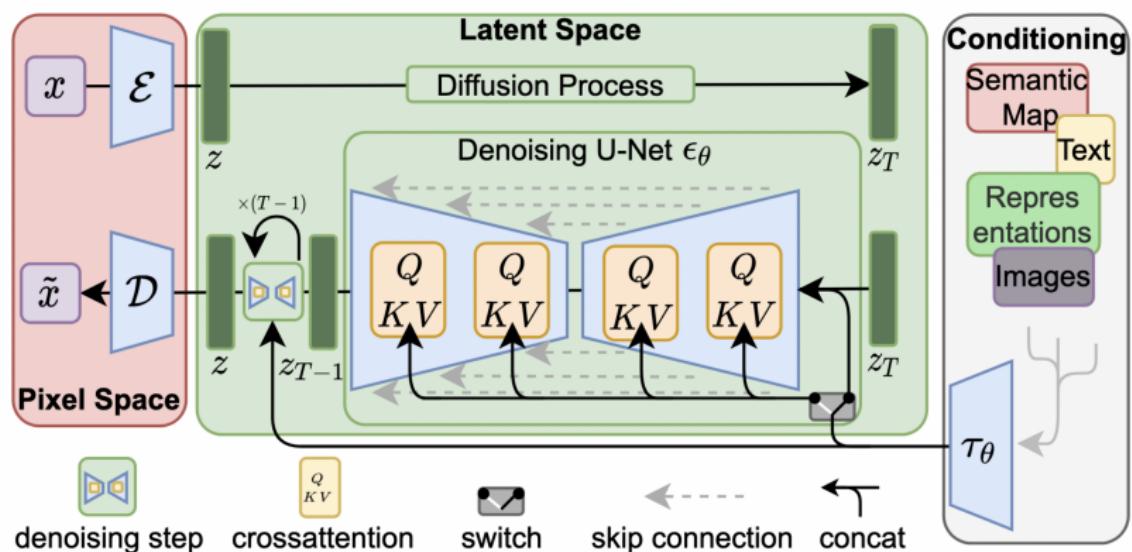


Figure 12: Arquitectura de Stable Diffusion [5]

- Usar una Red de Denoising [2] (p. ej., U-Net) para reducir el ruido en cada paso.
- El condicionamiento (p. ej., embedding de texto) ayuda a guiar al modelo para generar una salida específica usando el mecanismo x-attention.

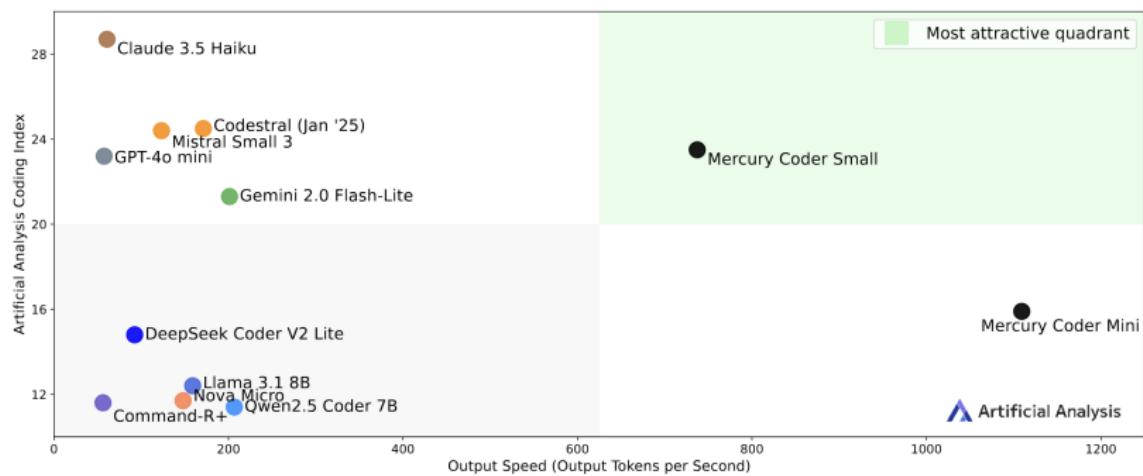
Modelos de difusión textuales

También existen para texto. Tienen muchas cualidades [3]:

- Más rápidos (5-10x)
- Computacionalmente más baratos (5-10x)
- Calidad similar

Coding Index vs. Output Speed: Smaller models

Artificial Analysis Coding Index (represents the average of LiveCodeBench & SciCode);
Output Speed: Output Tokens per Second; 1,000 Input Tokens; Coding focused workload



Outline : Redes Generativas Adversariales

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

Otros

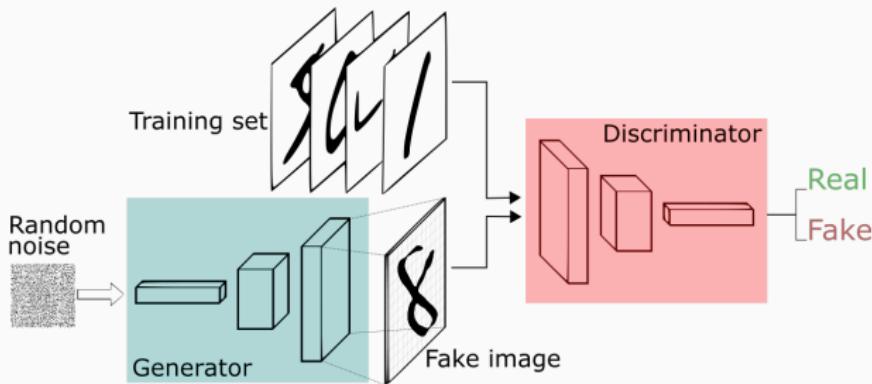
Conclusión

Red Generativa Adversarial (GAN)

Principio

- Entrena un modelo (**Generador**) para generar datos que otro modelo (**Discriminador**) no debería poder reconocer como artificial.
- El discriminador tarda en aprender

Al final, obtenemos datos generados (p. ej., imágenes) que se parecen a los de nuestro conjunto.

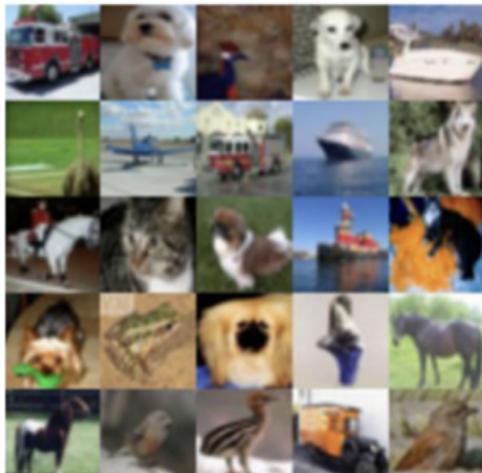


Generar imágenes de alta calidad

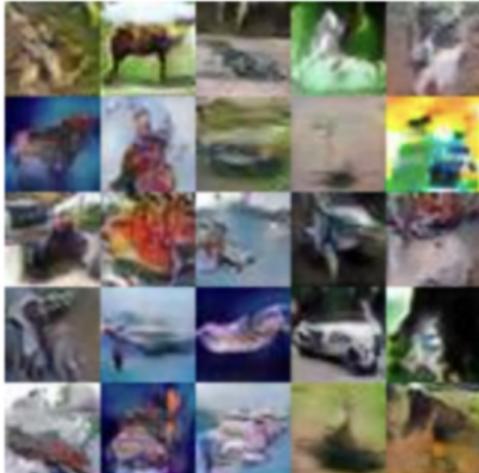
this-person-does-not-exist.com



Prueba de dos muestras: Planteamiento del problema



vs.



$$S_1 = \{\mathbf{x} \sim P\}$$

$$S_2 = \{\mathbf{x} \sim Q\}$$

Dado un conjunto finito de muestras de dos distribuciones:

$S_1 = \{x \sim P\}$ y $S_2 = \{x \sim Q\}$, ¿podemos determinar si las muestras provienen de la misma distribución (es decir, $P = Q$)?

Estadístico de prueba

Este es el problema de la **prueba de dos muestras**:

- **Hipótesis nula** $H_0: P = Q$
- **Hipótesis alternativa** $H_1: P \neq Q$

Un estadístico de prueba $T(S_1, S_2)$ cuantifica la diferencia entre S_1 y S_2 .

Ejemplo: Diferencia de medias

$$T(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} x - \frac{1}{|S_2|} \sum_{x \in S_2} x$$

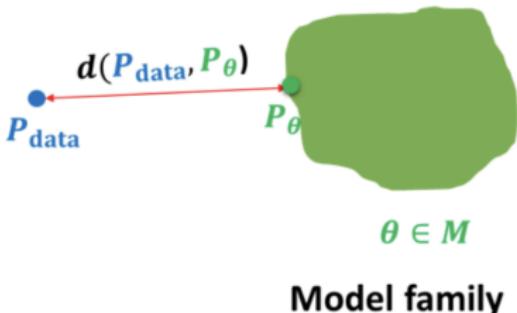
- Si $T > \alpha$ (umbral), **rechazar** H_0
- De lo contrario, H_0 es consistente con los datos

Observación clave: T es **sin verosimilitud** — usa solo muestras, no densidades P o Q .

Conexión con el modelado generativo



$$\mathbf{x}_i \sim P_{\text{data}} \\ i = 1, 2, \dots, n$$



Model family

- Tenemos muestras reales: $S_1 = \mathcal{D} = \{x \sim p_{\text{data}}\}$
- Un modelo con distribución p_θ , del cual podemos muestrear:
 $S_2 = \{x \sim p_\theta\}$

Idea: prueba de dos muestras para medir la discrepancia entre p_{data} y p_θ .

Definición alternativa de entrenamiento: Minimizar un objetivo de prueba de dos muestras:

$$\min_{\theta} \mathcal{L}(\theta) = T(S_1, S_2)$$

Objetivo: Hacer p_θ indistinguible de p_{data} .

Un modelo puede aprender este estadístico T

Encontrar un buen estadístico de prueba T en altas dimensiones es difícil.

Además, sabemos $S_1 \sim p_{\text{data}}$, $S_2 \sim p_\theta$, y inicialmente $p_{\text{data}} \neq p_\theta$.

Idea clave: Aprender el estadístico de prueba para detectar automáticamente diferencias entre S_1 y S_2 , entrenando un **clásificador** (llamado *discriminador*) para distinguirlos.

Un modelo puede aprender este estadístico T

Encontrar un buen estadístico de prueba T en altas dimensiones es difícil.

Además, sabemos $S_1 \sim p_{\text{data}}$, $S_2 \sim p_\theta$, y inicialmente $p_{\text{data}} \neq p_\theta$.

Idea clave: Aprender el estadístico de prueba para detectar automáticamente diferencias entre S_1 y S_2 , entrenando un **clasificador** (llamado *discriminador*) para distinguirlos.

El discriminador aprende un estadístico poderoso para la prueba de dos muestras

Clasificador binario D_ϕ (p. ej., red neuronal) para distinguir:

- Muestras reales $x \sim p_{\text{data}}$ (etiqueta $y = 1$)
- Muestras falsas $x \sim p_\theta$ (etiqueta $y = 0$)
- **Estadístico de prueba:** Pérdida de clasificación negativa.
- **Pérdida baja** \Rightarrow las muestras son fáciles de distinguir ($p_{\text{data}} \neq p_\theta$)
- **Pérdida alta** \Rightarrow las muestras son difíciles de distinguir ($p_{\text{data}} \approx p_\theta$)
- **Objetivo:** Maximizar el estadístico de prueba (apoyar $H_1 : p_{\text{data}} \neq p_\theta$), es decir, minimizar la pérdida de clasificación.

Objetivo de entrenamiento del discriminador

Maximizar lo siguiente respecto a D_ϕ :

$$\max_{D_\phi} V(p_\theta, D_\phi) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D_\phi(x)] + \mathbb{E}_{x \sim p_\theta} [\log(1 - D_\phi(x))]$$

Aproximado a partir de muestras:

$$V \approx \sum_{x \in S_1} \log D_\phi(x) + \sum_{x \in S_2} \log(1 - D_\phi(x))$$

Esto es entropía cruzada binaria:

- Predecir $D_\phi(x) \approx 1$ para $x \in S_1$ (real)
- Predecir $D_\phi(x) \approx 0$ para $x \in S_2$ (falso)

El discriminador óptimo es $D_\theta^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$. Lo que hace imposible discriminar cuando $p_\theta = p_{\text{data}}$: $D_\theta^*(x) = \frac{1}{2}$

Objetivo GAN: Pérdida Minimax

El objetivo de entrenamiento de GAN es:

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D_{\phi}(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D_{\phi}(G_{\theta}(z)))]$$

Objetivo GAN: Pérdida Minimax

El objetivo de entrenamiento de GAN es:

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D_{\phi}(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D_{\phi}(G_{\theta}(z)))]$$

El discriminador D_{ϕ} intenta:

- Maximizar la probabilidad de asignar etiquetas correctas:
- $\log D_{\phi}(x)$ para datos reales
- $\log(1 - D_{\phi}(G_{\theta}(z)))$ para datos falsos

El generador G_{θ} intenta:

- Minimizar $\log(1 - D_{\phi}(G_{\theta}(z)))$, es decir, (engañosamente) engañar al discriminador
- Hacer que $p_{\theta}(x)$ sea indistinguible de p_{data}

Objetivo GAN: Pérdida Minimax

El objetivo de entrenamiento de GAN es:

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D_{\phi}(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D_{\phi}(G_{\theta}(z)))]$$

El discriminador D_{ϕ} intenta:

- Maximizar la probabilidad de asignar etiquetas correctas:
- $\log D_{\phi}(x)$ para datos reales
- $\log(1 - D_{\phi}(G_{\theta}(z)))$ para datos falsos

El generador G_{θ} intenta:

- Minimizar $\log(1 - D_{\phi}(G_{\theta}(z)))$, es decir, (engañosamente) engañar al discriminador
- Hacer que $p_{\theta}(x)$ sea indistinguible de p_{data}

Sin embargo, es un equilibrio de Nash en un juego no cooperativo de 2 jugadores. **Actualizar el gradiente de ambos modelos simultáneamente (actualizar independientemente del otro jugador) no puede garantizar la convergencia.**

Objetivo GAN: Algoritmo

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Redes Generativas Adversariales



Explicaciones detalladas sobre GANs [aquí](#) o [allí](#) ; [Tutorial de código](#) ;
[Aplicaciones GAN](#) ;

Visualización de generación GAN en ImageNet

GANs: Creative Adversarial Network (CAN)

Mismo principio que GAN, pero con dos etapas de discriminación

- La primera etapa discrimina si una imagen es real o falsa: esto incentiva al generador a **crear imágenes que se parezcan a la distribución de datos del conjunto** (es decir, que luzcan realistas).
- La segunda etapa clasifica el estilo artístico de la imagen: incentiva al generador a **crear imágenes no asociadas a ninguna clase conocida**. Resuelta una imagen que parece plausible y realista, sin pertenecer a ningún movimiento artístico conocido.

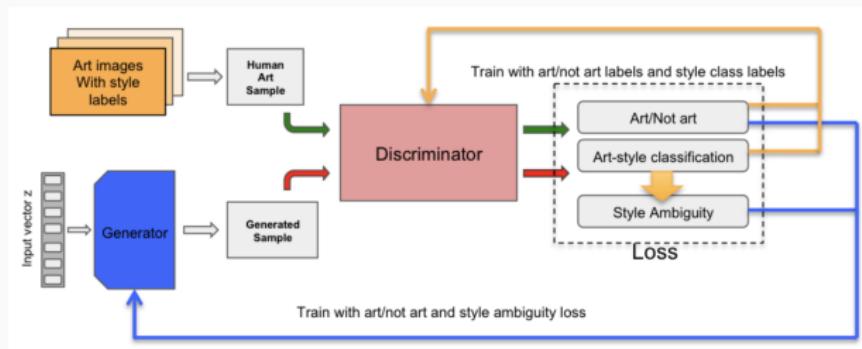


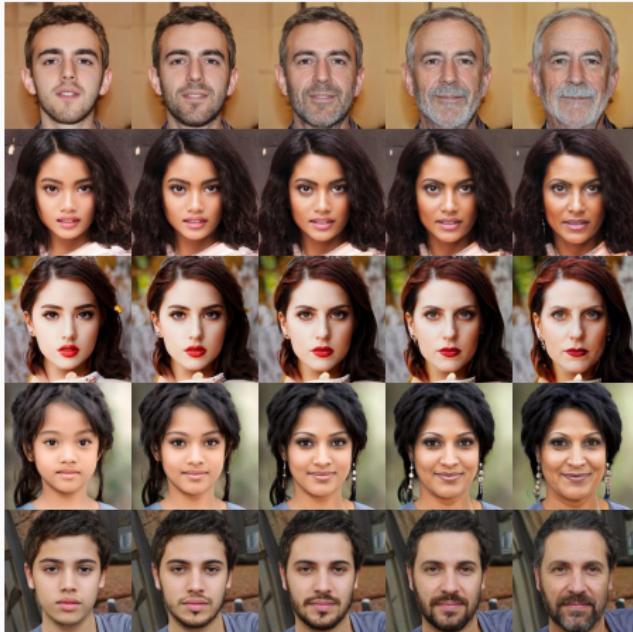
Figure 13: El discriminador tiene 2 salidas: 1 realismo y 2 estilo artístico.

GANs: Creative Adversarial Network (CAN)



Implementación en TensorFlow ; [Paper](#)

Ejemplo de edición de imagen



Manera simple de editar una foto real:

1. Encontrar el z que generaría una imagen representando esta cara
2. Mover ese vector en la "dirección de edad/sonrisa/género" en el espacio latente

Inversión de imagen en GANs

Finding faces inside StyleGAN's latent space:



 @xsteenbrugge

Figure 14: Cada cara tiene su latente z que puede generarla.

Inversión de imagen en GANs

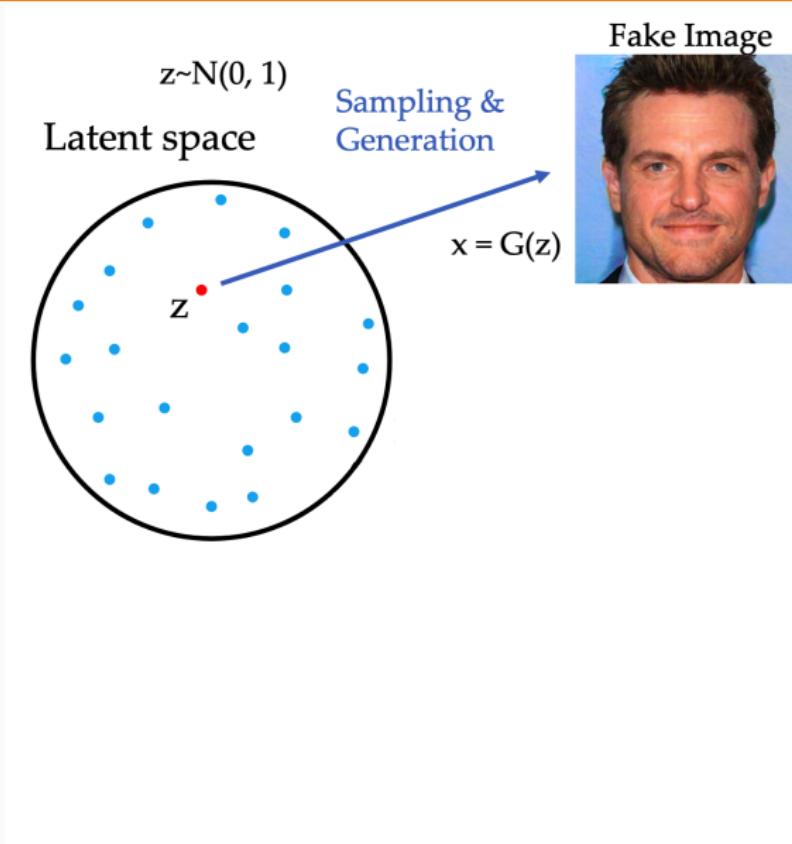
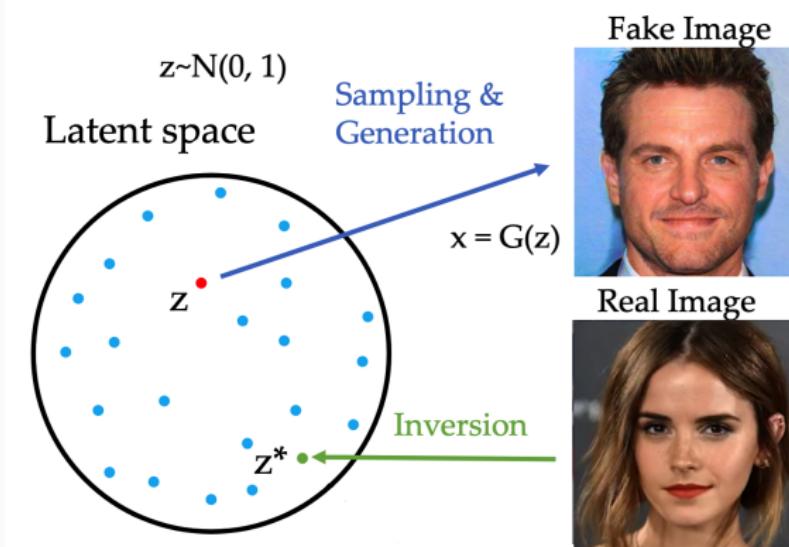


Figure 14: Se trata de encontrar z^* que genera la imagen

Inversión de imagen en GANs



invert real image into latent space

$$z^* = \arg \min_z (G(z), x)$$

Figure 14: Se trata de encontrar z^* que genera la imagen

Inversión de imagen en GANs

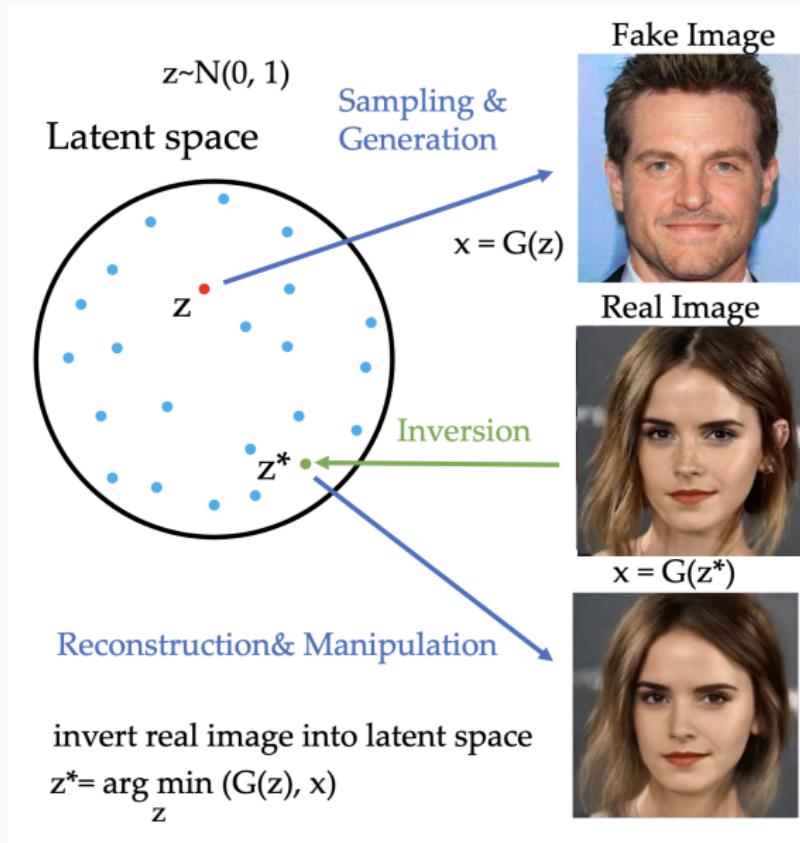


Figure 14: Se trata de encontrar z^* que genera la imagen

Edición de imagen con GANs

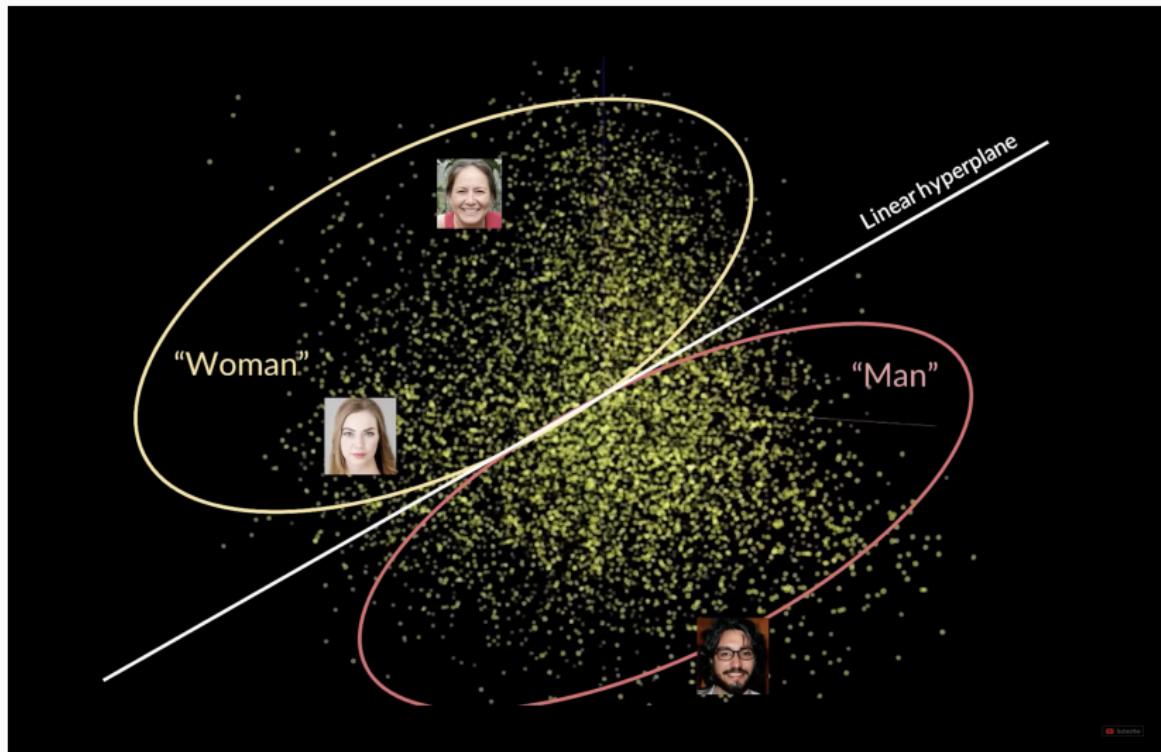


Figure 15: Clasificar z respecto al género

Edición de imagen con GANs

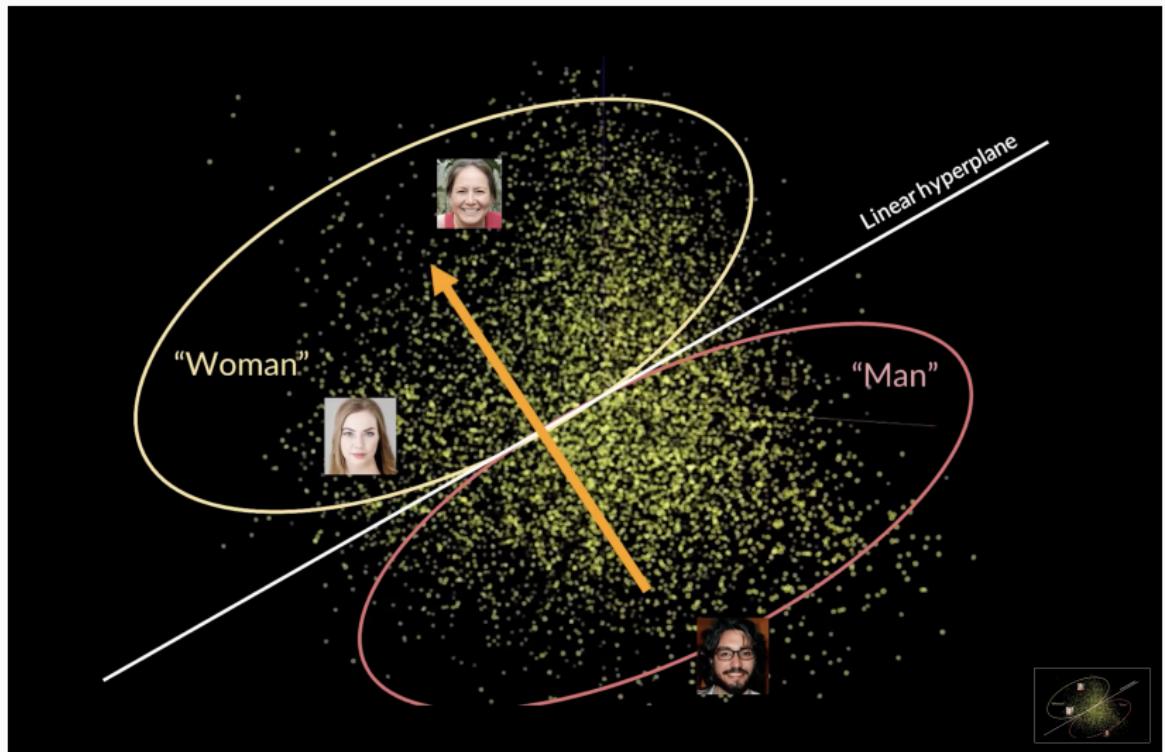


Figure 15: Tomar la normal del hiperplano

Edición de imagen con GANs

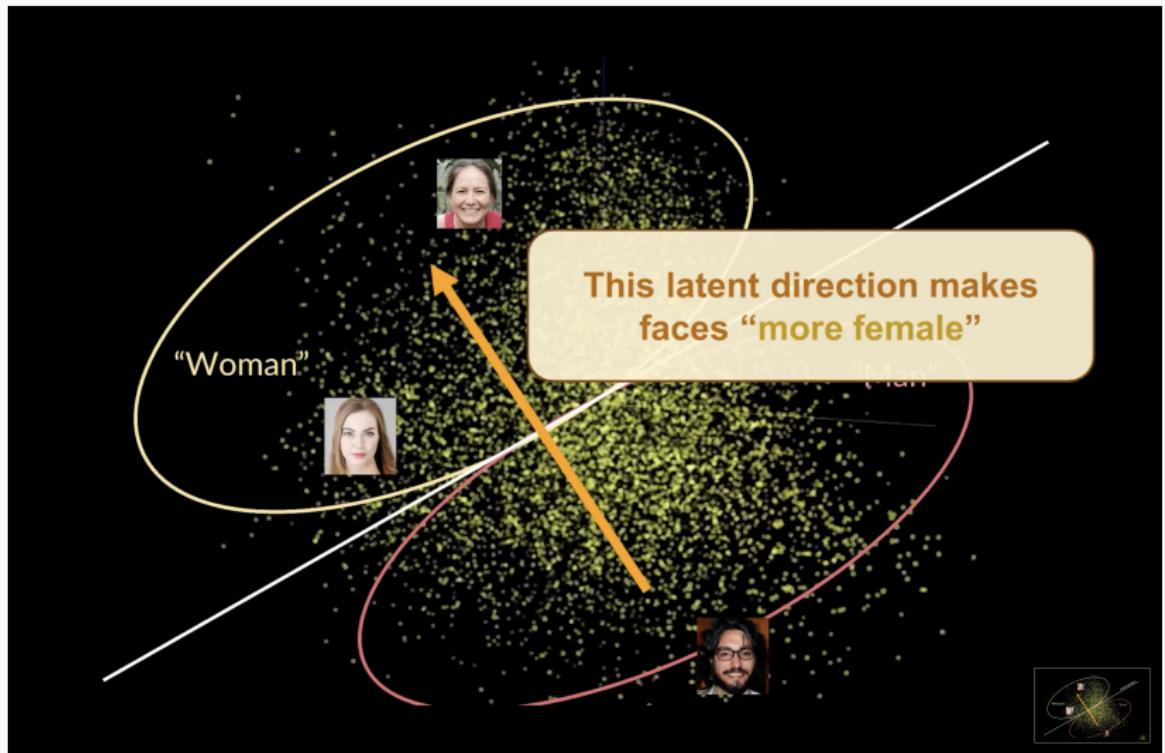


Figure 15: Editar imágenes con esta dirección

Outline : Otros

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

Otros

Conclusión

Transferencia de Estilo Neuronal

Principio

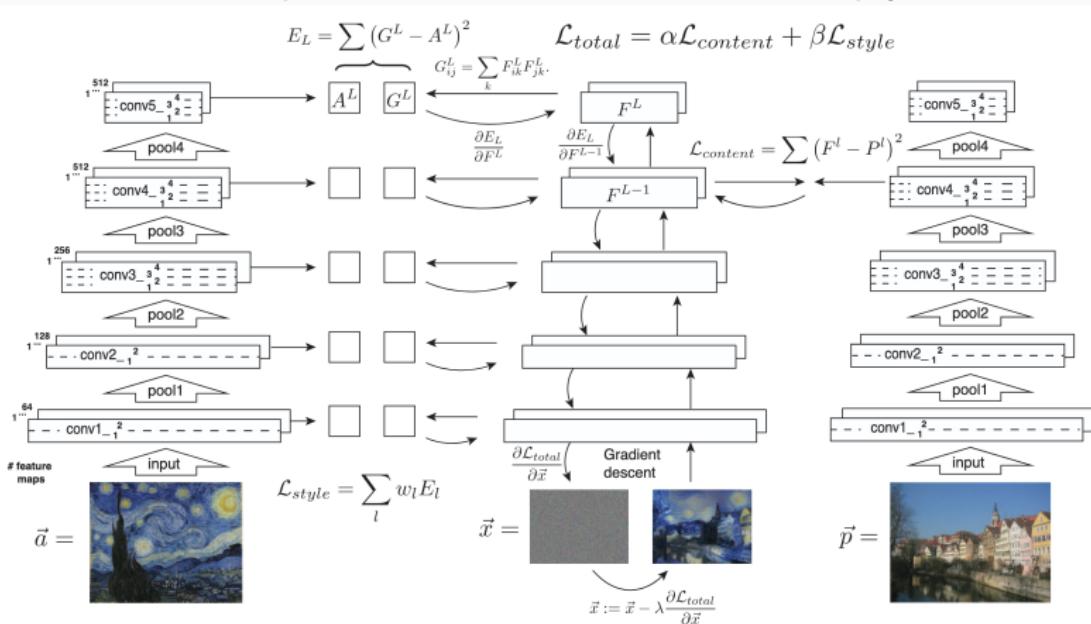
Aplica el estilo de una imagen a otra separando el contenido (representación de alto nivel) y el estilo (representación de bajo nivel).



[Tutorial en PyTorch](#); [Paper](#)

Transferencia de Estilo Neuronal

- Optimización:** Ruido blanco \vec{x} se optimiza iterativamente, vía sus características F_l (contenido) y G_l (estilo).
- Cálculo de la pérdida:** $\mathcal{L}_{content} = \|F_l - P_l\|^2$ y $\mathcal{L}_{style} = \sum_l \|G_l - A_l\|^2$ minimizadas vía backpropagacion.
- Salida:** La \vec{x} optimizada coincide con el contenido \vec{p} y el estilo \vec{a} .



Transferencia de Estilo Neuronal: resultado



Transferencia de Estilo Neuronal: resultado



Edición de imagen con GANs: transfiriendo estilo



Figure 16: Traducción imagen-a-imagen basada en StyleGan [4] permite la toonificación usando [este modelo](#)

Edición de imagen con GANs: transfiriendo estilo

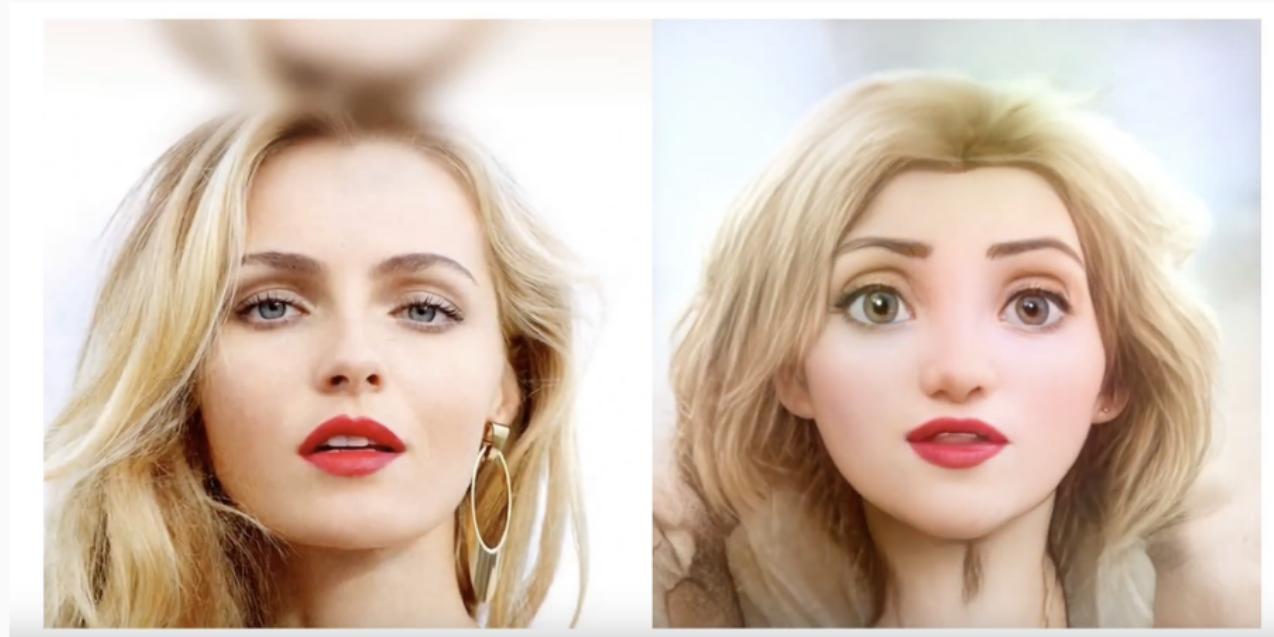


Figure 16: Traducción imagen-a-imagen basada en StyleGan [4] permite la toonificación usando [este modelo](#)

Outline : Conclusión

Modelos Generativos

Modelos Autorregresivos

Autoencoders Variacionales

Modelos de Difusión

Redes Generativas Adversariales

Otros

Conclusión

Tabla resumen: Comparación de modelos generativos

Característica	Autorregresivo	Implícito (GAN)	Prescrito (VAE, Difusión)
Entrenamiento	Estable	Inestable	Estable
Verosimilitud	Exacta	No	Aproximada
Muestreo	Lento	Rápido	Rápido
Compresión	Sin pérdida	No	Con pérdida
Representación	No	No	Sí

Verde = Ventaja, Rojo = Limitación, Amarillo = Compensación.

Explicación de la tabla

- **Entrenamiento:** Los GANs sufren inestabilidad y colapso de modos; AR, VAE y Difusión son más estables.
- **Verosimilitud:** Los modelos AR calculan probabilidades exactas (útil para detección de anomalías). Los GANs no tienen verosimilitud. VAE usa ELBO; Difusión usa score matching.
- **Muestreo:** Los modelos AR son lentos (secuenciales). Los GANs muestran en una pasada. Difusión es rápido por paso pero requiere muchos pasos.
- **Compresión:** Solo los modelos AR permiten compresión sin pérdida (p. ej., Bits-Back). Otros son inherentemente con pérdida.
- **Aprendizaje de representación:** VAEs y Difusión ofrecen espacios latentes significativos (p. ej., para edición). Los GANs requieren trabajo adicional para invertir.

Conclusiones

No existe **un modelo mejor en general** — solo el **mejor modelo para la tarea**:

- ¿Necesitas calidad? → GAN o Difusión
- ¿Necesitas espacio latente? → VAE o Difusión
- ¿Necesitas verosimilitud? → AR o VAE
- ¿Necesitas rapidez? → GAN
- ¿Necesitas compresión? → Autorregresivo

Recursos útiles

- Ermon, S. (2023). CS236: Deep Generative Models [1]
- Sanseviero, O. et al. (2025). Hands-On Generative AI with Transformers and Diffusion Models. [7]
- Tomczak, J. M. (2022). Deep Generative Modeling. [8]

Questions?

References i

-  S. Ermon.
CS236: Deep Generative Models, 2023.
-  J. Ho, A. Jain, and P. Abbeel.
Denoising diffusion probabilistic models.
In Advances in Neural Information Processing Systems, volume 2020-Decem, pages 1–25, 2020.
-  I. Labs, S. Khanna, S. Kharbanda, S. Li, H. Varma, E. Wang, S. Birnbaum, Z. Luo, Y. Miraoui, A. Palrecha, S. Ermon, A. Grover, and V. Kuleshov.
Mercury: Ultra-Fast Language Models Based on Diffusion.
pages 1–15, 2025.

References ii

-  E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or.
Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation.
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 2287–2296, 2021.
-  R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer.
High-Resolution Image Synthesis with Latent Diffusion Models.
CVPR, 2022.

References iii

-  O. Ronneberger, P. Fischer, and T. Brox.
U-net: Convolutional networks for biomedical image segmentation.
In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
-  O. Sanseviero, P. Cuenca, A. Passos, and J. Whitaker.
Hands-On Generative AI with Transformers and Diffusion Models.
2025.
-  J. M. Tomczak.
Deep Generative Modeling.
2022.