



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Módulo 6 IA Generativa

Diploma en Inteligencia Artificial



informatica.usm.cl
@informaticausm





Acerca del relator

- Profesor Diploma Inteligencia Artificial de la UTFSM
- Socio Director Sistemas Openb SpA
- Ingeniero Informático de la USACH, Master en Finanzas Aplicadas de la UDD, Diplomado en Data Analytics de la Universidad de Chile/MIT Sloan y Master en Inteligencia Artificial de la UAI.
- Más de 15 años en roles gerenciales , supervisando la implementación de soluciones personalizadas de Infraestructura TI y proyectos de analítica de datos e Inteligencia Artificial.
- Fundador de una ONG enfocada en analítica de datos para impacto social.
- Durante este año ha impartido cursos de "IA Generativa con Python" y "Programación de LLM usando Python".

Pablo Álvarez González





- Modelos Probabilísticos Clásicos
- Mezcla Gausiana (GMM)
- Modelo oculto de Markov (HMM)
- Autoencoder Variacional (VAE)
- Redes GAN

Agenda del Módulo

01

Fundamentos

Modelos discriminativos versus generativos

03

Autoencoders Variacionales

De autoencoders básicos a VAE

05

Evolución y Aplicaciones

Variantes modernas y casos de uso

02

Modelos Clásicos

Mezcla Gaussiana y Modelos Ocultos de Markov

04

Redes GAN

Arquitectura adversaria y entrenamiento

06

Desafíos y Futuro

Problemas éticos y próximas tecnologías

La Pregunta Fundamental en IA

Modelos Discriminativos

¿A qué clase Y pertenece este dato X?

Modelamos la probabilidad condicional:

$$P(Y | X)$$

Se enfoca en aprender la **frontera de decisión** entre clases para realizar clasificación o predicción.

Modelos Generativos

¿Cómo se creó este dato X?

Modelamos la distribución de probabilidad:

$$P(X) \text{ o } P(X, Y)$$

Se enfoca en aprender la **distribución de los datos** para generar nuevas muestras realistas.



**Objetivo:
Modelar $P(x)$
para generar
nuevas
muestras**

El Desafío de la Alta Dimensionalidad

El Problema

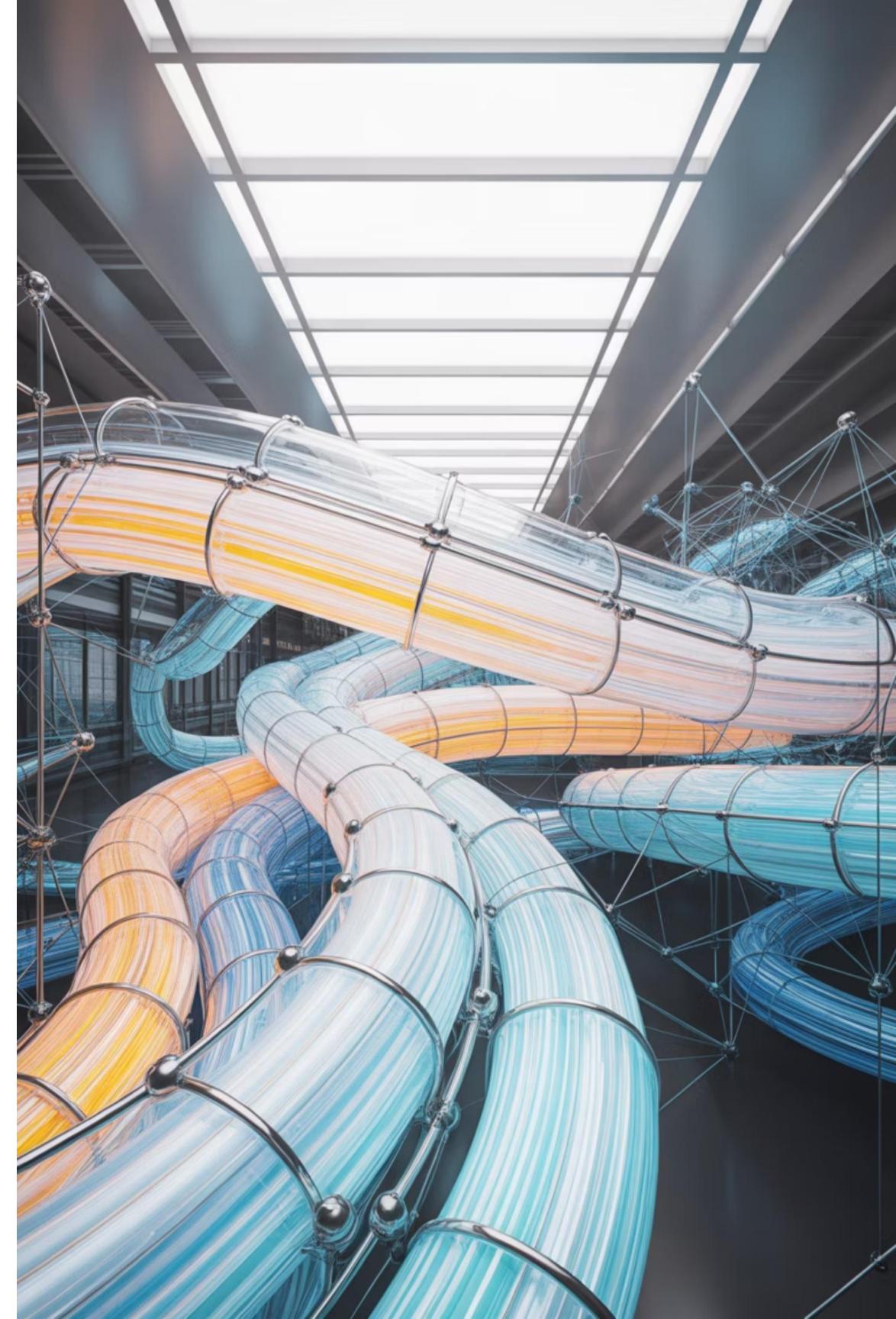
Una imagen de 1024×1024 píxeles tiene **más de 1 millón de dimensiones** que modelar simultáneamente.

Complejidad Computacional

Modelar la distribución de probabilidad conjunta de todos estos píxeles es **computacionalmente intratable**.

La Solución

Necesitamos modelos que simplifiquen mediante **variables latentes** o suposiciones de independencia.





Parte 1: Modelos de Mezcla Gaussiana (GMM)

Mezcla Gaussiana



Figura 1: Segmentación de Imágenes

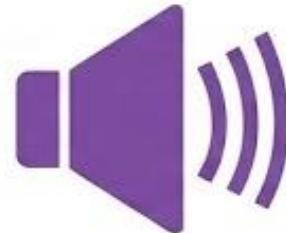


Figura 2: Clasificación de sonidos



Figura 3: Clasificación de documentos

Concepto de Mezcla Gaussiana

Los Modelos de Mezcla Gaussiana (GMM) proponen que **cualquier distribución de datos compleja puede aproximarse** mediante la suma ponderada de múltiples distribuciones Gaussianas simples.

No es solo "agrupar datos", es **modelar la densidad de probabilidad** subyacente.

$$P(x) = \sum_{k=1}^K \pi_k N(x | \mu_k, \Sigma_k)$$

Parámetros

- π_k : Peso de la mezcla
- μ_k : Centro del cluster
- Σ_k : Matriz de covarianza

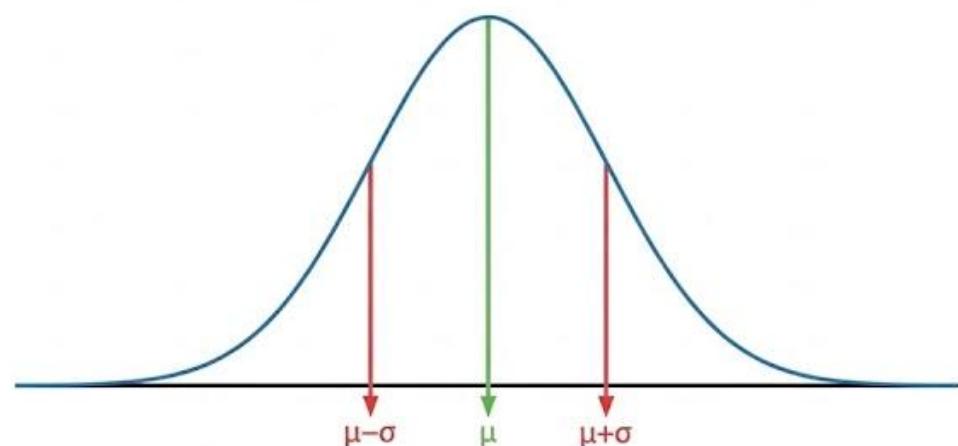
Concepto de Mezcla Gaussiana

Una mezcla Gaussiana es una función de distribución de probabilidad formada como una suma ponderada de K distribuciones de probabilidad Gaussiana

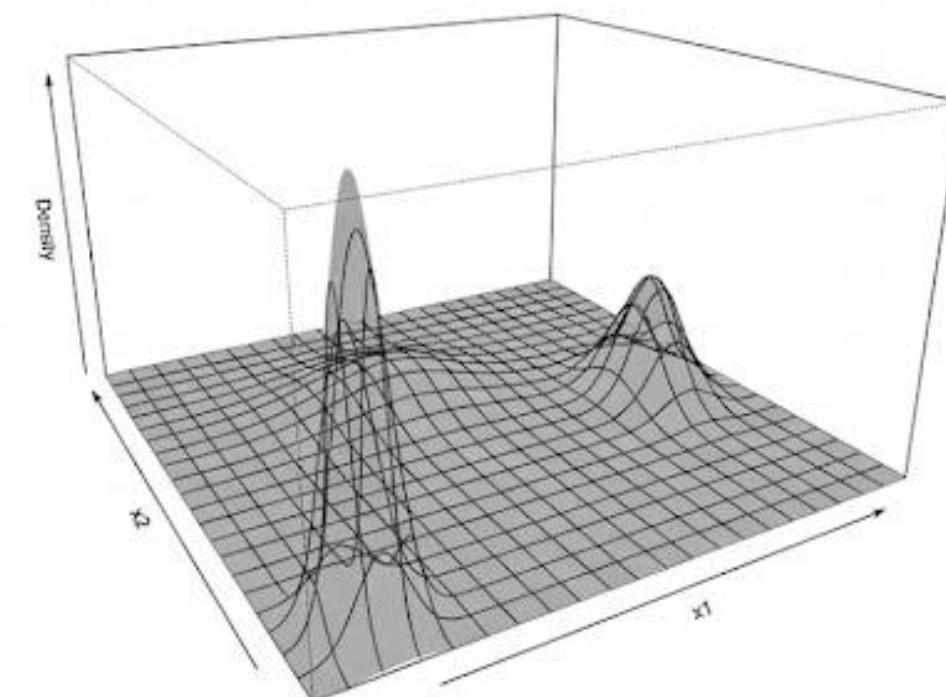
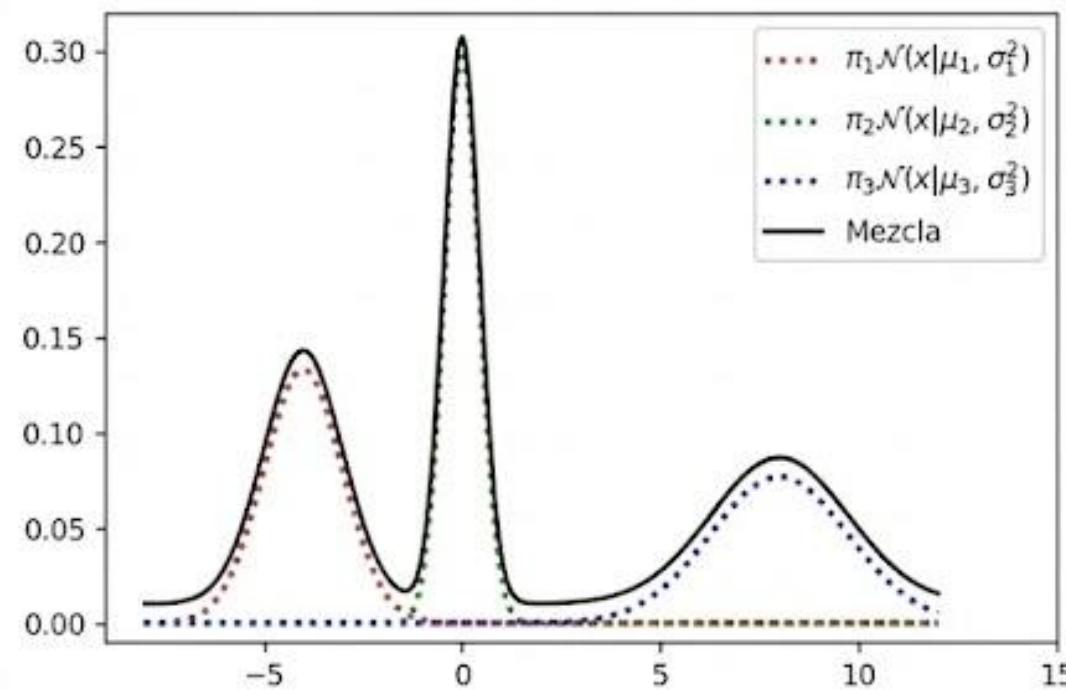
$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

Pesos Componentes

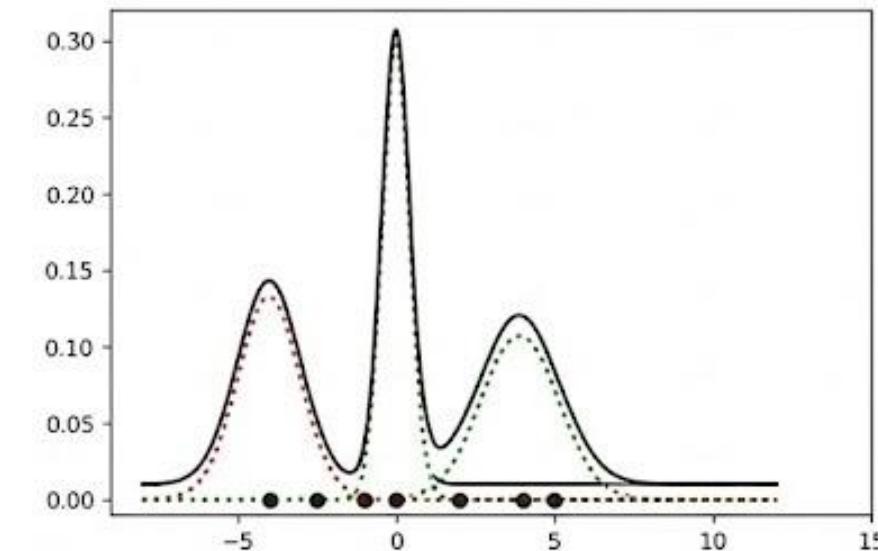
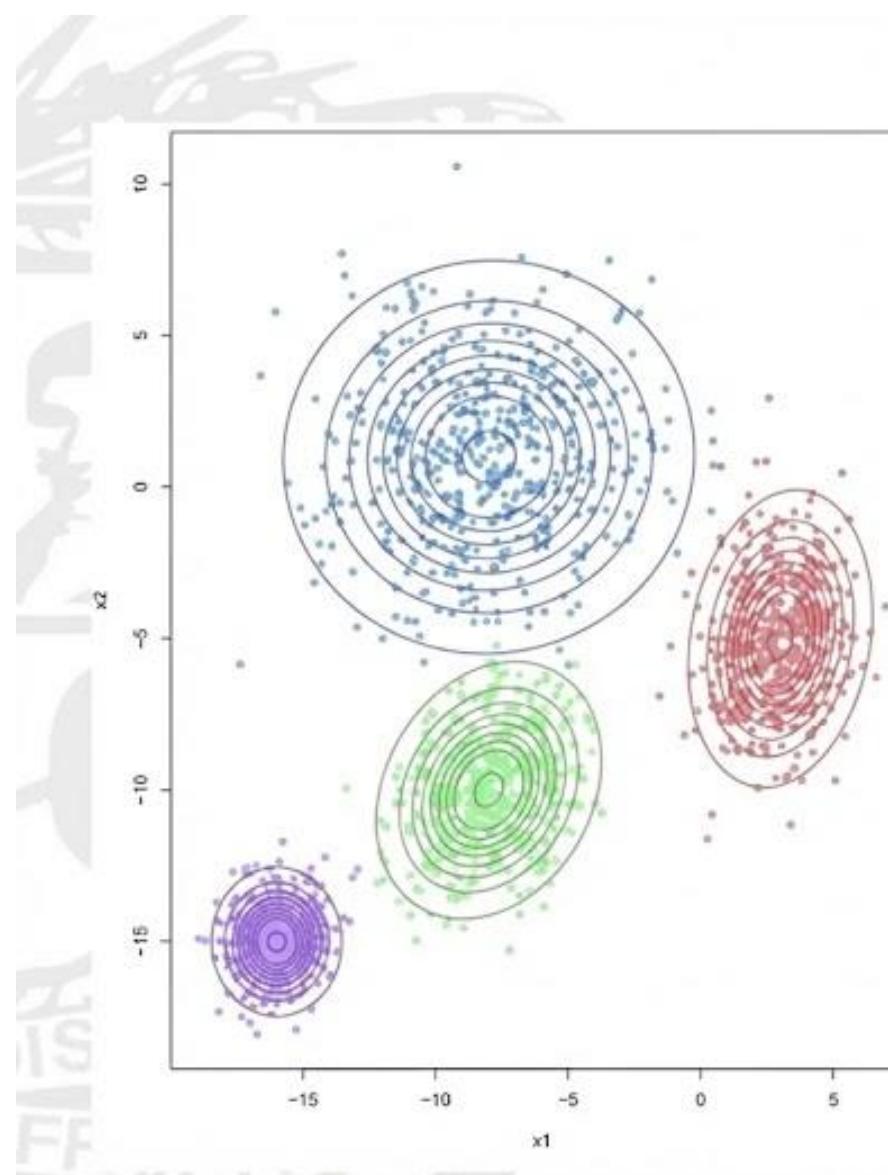
tal que $\sum_{k=1}^K \pi_k = 1$



Concepto de Mezcla Gaussiana



Concepto de Mezcla Gaussiana



Un Modelo de Mezclas Gaussianas (GMM por sus siglas en inglés) es un modelo probabilístico en el que se ajusta una mezcla Gaussiana a un conjunto de datos, es decir que representamos a los datos como una suma de distribuciones Gaussianas.



GMM vs K-Means

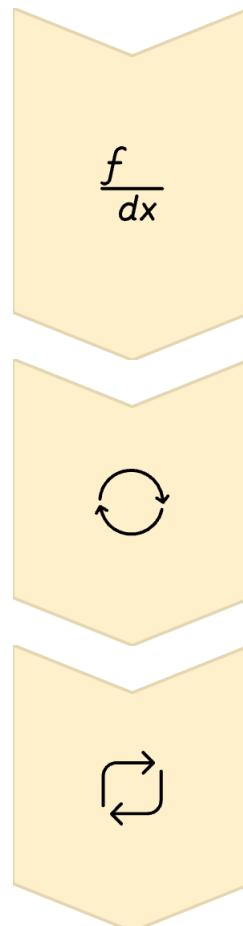
K-Means (Rígido)

Cada punto pertenece a **un solo cluster** de forma determinística. Asignación binaria sin incertidumbre.

GMM (Probabilístico)

Cada punto tiene una **probabilidad de pertenecer a cada cluster**. Permite modelar incertidumbre y solapamiento.

Algoritmo EM para GMM



Paso E (Expectation)

Calculamos la probabilidad de que cada dato pertenezca a cada componente Gaussiana, dadas las estimaciones actuales de parámetros.

Paso M (Maximization)

Recalculamos los parámetros μ_k y σ_k basándonos en las probabilidades calculadas en el paso E.

Iteración

Repetimos estos pasos alternadamente hasta que el modelo converja a una solución óptima local.

Limitaciones de GMM



¿Dónde funciona bien?

- Datos tabulares de baja dimensión
- Problemas de clustering suave
- Estimación de densidad en espacios simples

¿Dónde falla?

- Imágenes de alta resolución
- Estructuras complejas no lineales
- Cuando hay miles de dimensiones

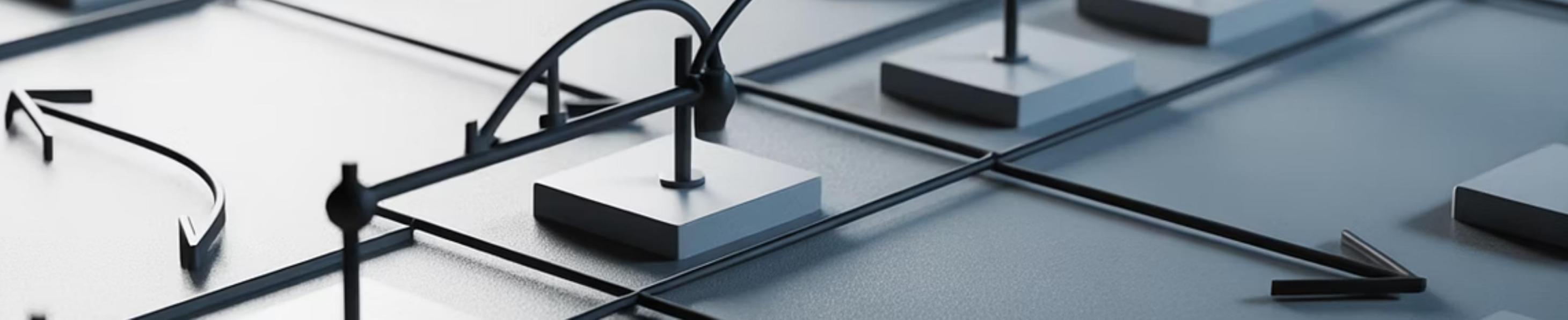
Ejemplo de GMM

Para aplicar los GMM, tomaremos el conjunto de datos *Rice_MSC_Dataset* que contiene 75000 observaciones para cinco clases de arroz, de modo que para cada observación se tienen 107 características, donde las primeras 106 corresponden a medidas tomadas a cada muestra de arroz y la última que nos indica la clase de la observación.

	AREA	PERIMETER	MAJOR-AXIS	MINOR-AXIS	CLASS
1	7805.00	437.92	209.82	48.02	Basmati
2	7503.00	340.76	138.34	69.84	Arborio
3	5124.00	314.62	141.98	46.58	Jasmine
4	6781.00	307.023	116.2443	74.8093	Karacadag
5	11648.00	445.53	178.47	84.93	Ipsala

Tabla 1: Previsualización de *Rice_MSC_Dataset*.





Parte 2: Modelos Ocultos de Markov (HMM)

El Problema del Tiempo

GMM asume que los datos son **independientes e idénticamente distribuidos** (i.i.d.). Pero, ¿qué sucede cuando trabajamos con secuencias temporales?



Texto

Las palabras dependen de las anteriores en la oración.



Audio

Los fonemas se conectan en secuencias temporales.



Secuencias Biológicas

Los nucleótidos en ADN tienen dependencias secuenciales.

Necesitamos modelos que capturen dependencias temporales.

Arquitectura de HMM

Los Modelos Ocultos de Markov modelan secuencias asumiendo que existen **estados ocultos** (no observables) que generan las **observaciones** visibles.



Componentes de un HMM



Matriz de Transición (A)

Define la probabilidad de pasar del estado oculto i al estado j :

$$P(S_t \mid S_{t-1})$$



Matriz de Emisión (B)

Define la probabilidad de generar una observación visible dado un estado oculto:

$$P(O_t \mid S_t)$$



Estado Inicial (π)

Define las probabilidades de comenzar en cada estado posible al inicio de la secuencia:

$$P(S_0)$$

Ejemplo Clásico: Reconocimiento de Voz



Antes de las redes neuronales profundas, los HMM eran el estándar para reconocimiento de voz.

Estados Ocultos

Fonemas o unidades lingüísticas que no vemos directamente.

Observaciones

Características extraídas del audio (MFCC, espectrogramas).

El modelo aprende qué fonemas probablemente generaron qué señales de audio.

Aplicaciones de HMM

Procesamiento de Voz

Reconocimiento automático de habla y síntesis de voz.

Bioinformática

Predicción de estructuras de genes y alineamiento de secuencias.

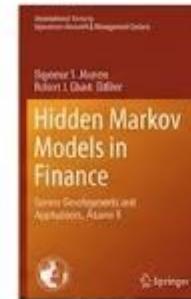
Finanzas

Modelado de series temporales y predicción de regímenes de mercado.

Aplicaciones de HMM

Finanzas

regímenes de
mercado



Ecología

movimiento animal
captura-recaptura

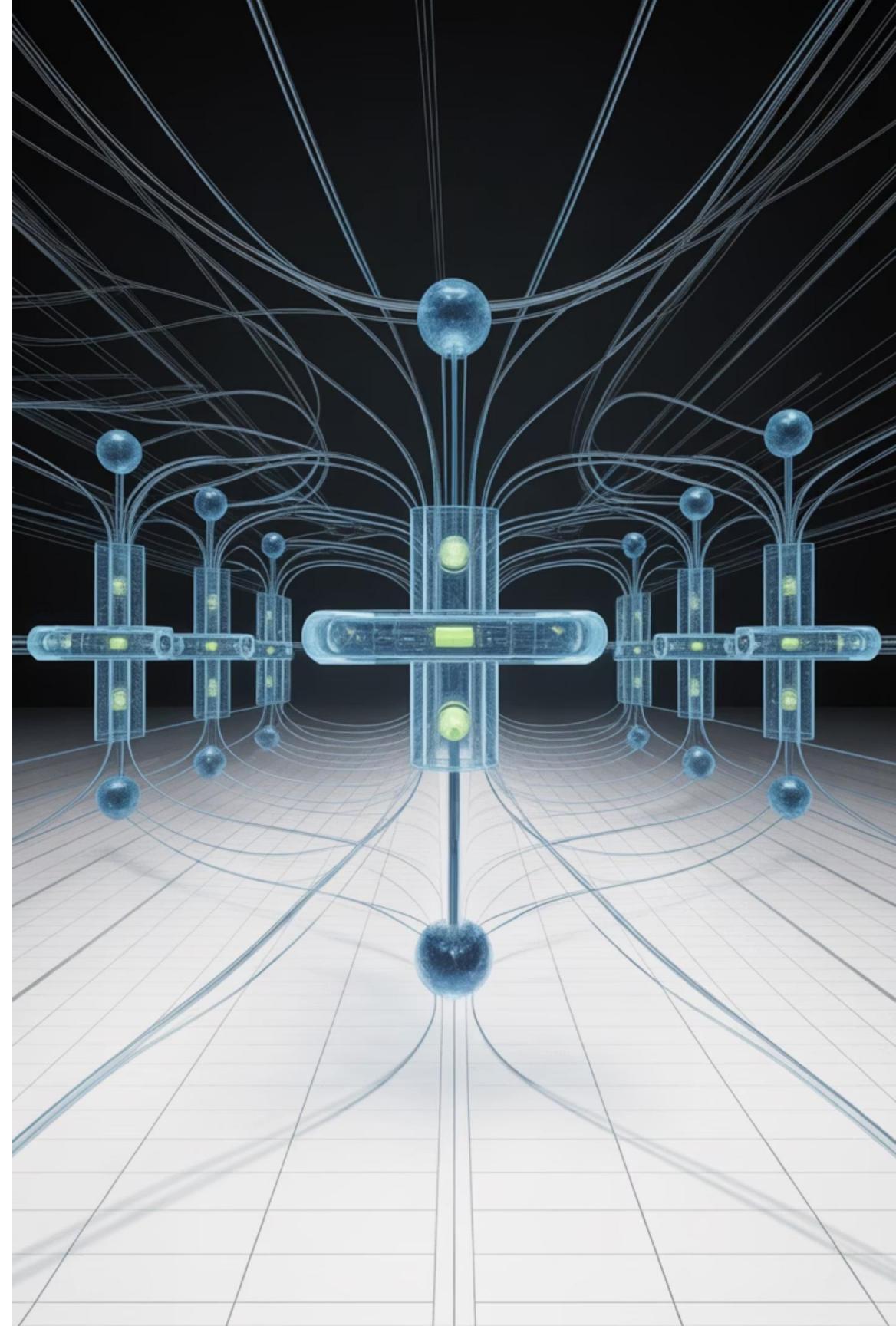
Medicina

progresión de la enfermedad

Reconocimiento de Patrones Generales

reconocimiento de voz
movimiento humano

Parte 3: Autoencoders Variacionales (VAE)

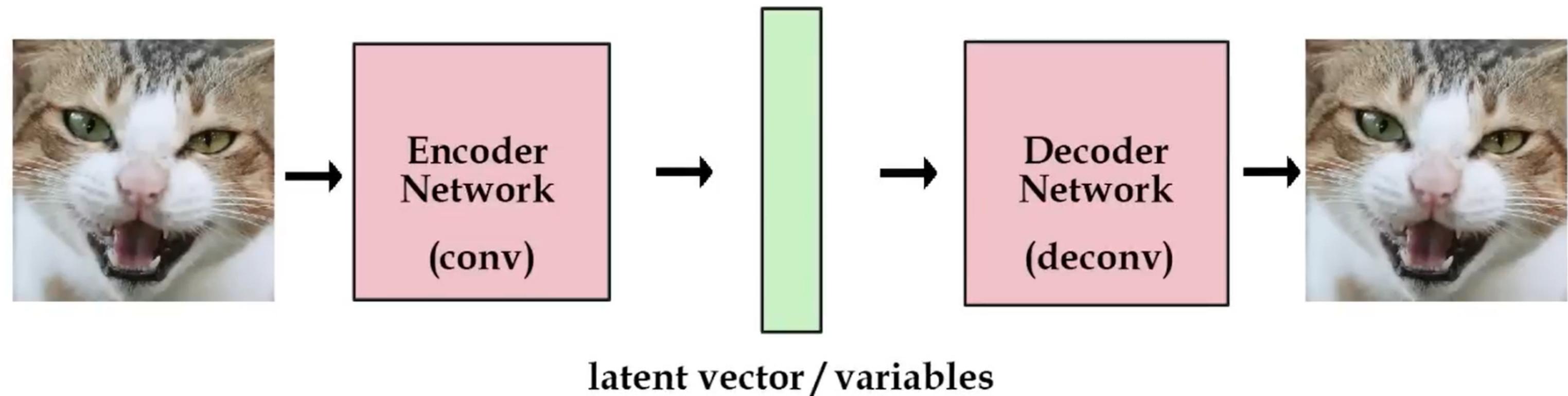


Del Autoencoder Básico al VAE

Antes de entender VAE, revisemos el **autoencoder estándar**:



Autoencoders



Componentes del Autoencoder

1

Encoder

Red que comprime la entrada X a un vector latente de menor dimensión Z .

2

Bottleneck

Cuello de botella que obliga a la red a aprender solo las características más esenciales de los datos.

3

Decoder

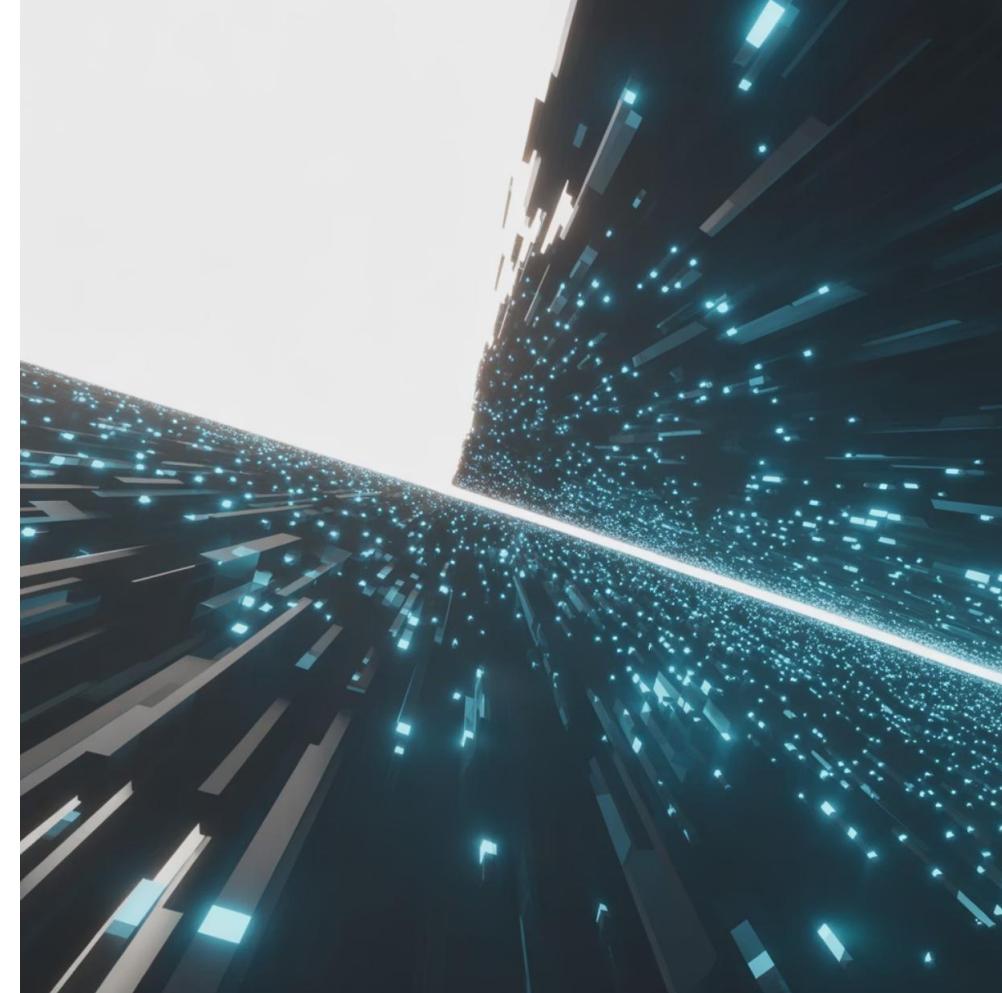
Red que reconstruye la entrada original X a partir del vector latente Z .

El Problema del Autoencoder Estándar

Espacio Latente Discontinuo

El encoder aprende a mapear cada entrada a un **punto específico** en el espacio latente. Este espacio es **discontinuo y disperso**.

Si tomamos un punto aleatorio en Z, el decoder genera ruido sin sentido.



Consecuencia

No podemos usar autoencoders estándar para **generación** de nuevos datos, solo para compresión.

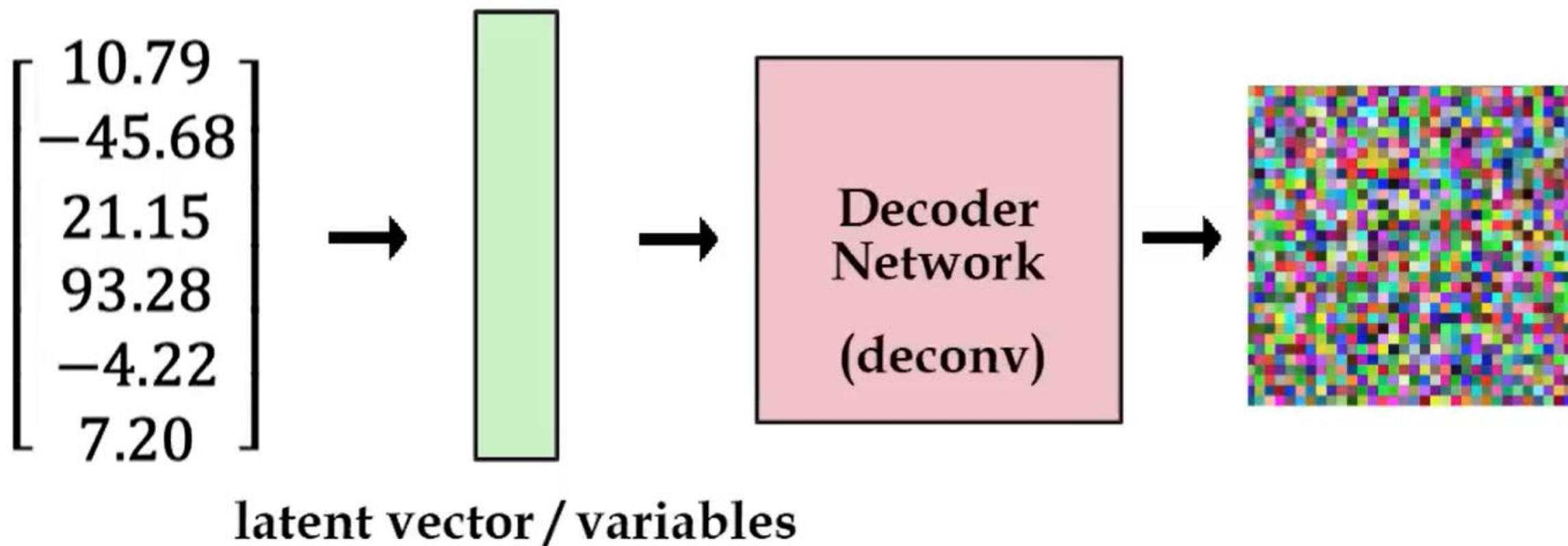
Autoencoders

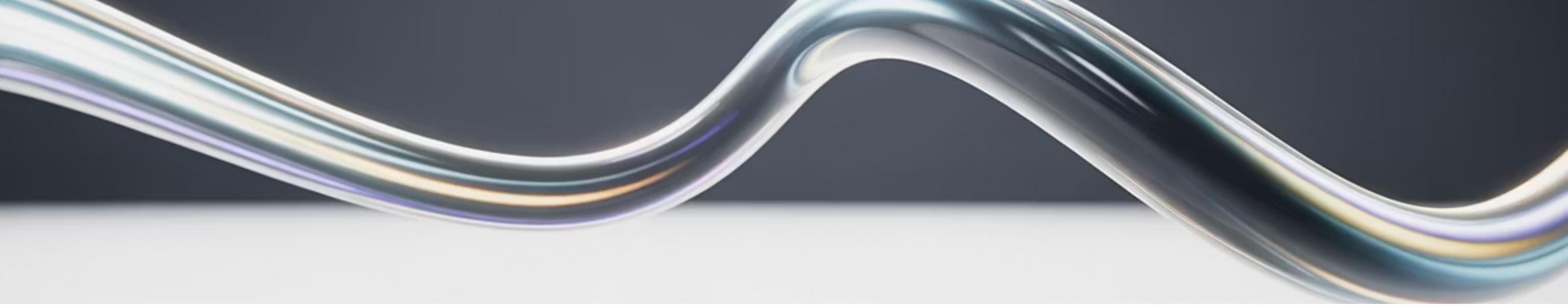
AutoEncoder



Autoencoders

¿Es posible sacar datos aleatorios?



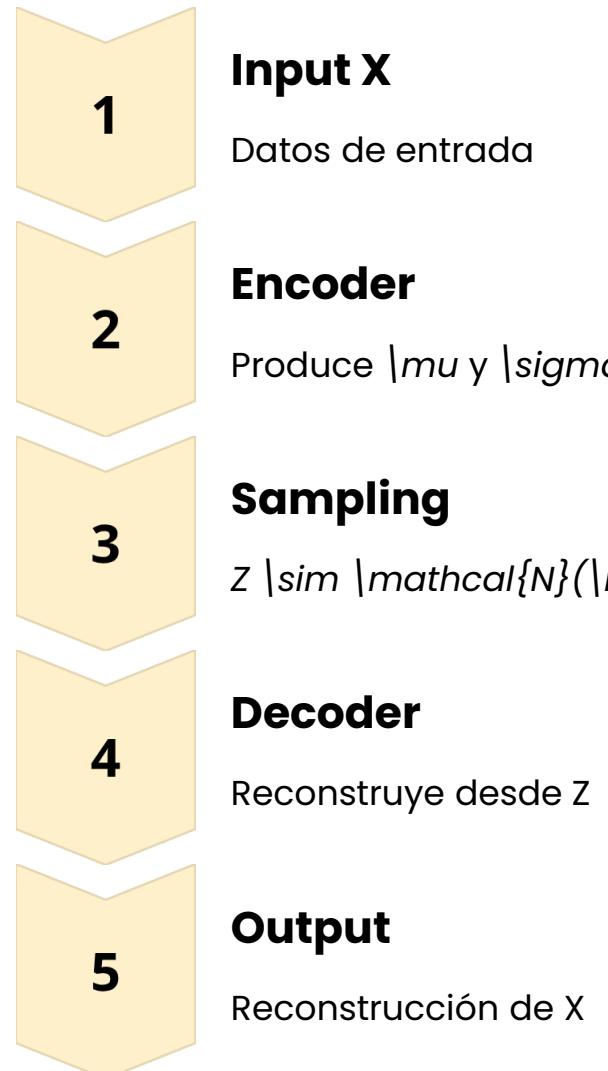


La Innovación de VAE

En lugar de codificar un punto fijo, codificamos una distribución de probabilidad

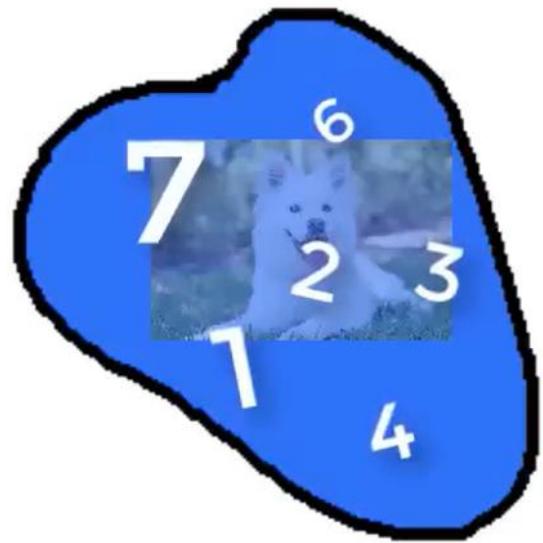
Arquitectura del VAE

El Autoencoder Variacional introduce un cambio fundamental: el encoder no produce un punto Z, sino los parámetros de una distribución Gaussiana.



Sampling en VAEs

Dog
Distribution



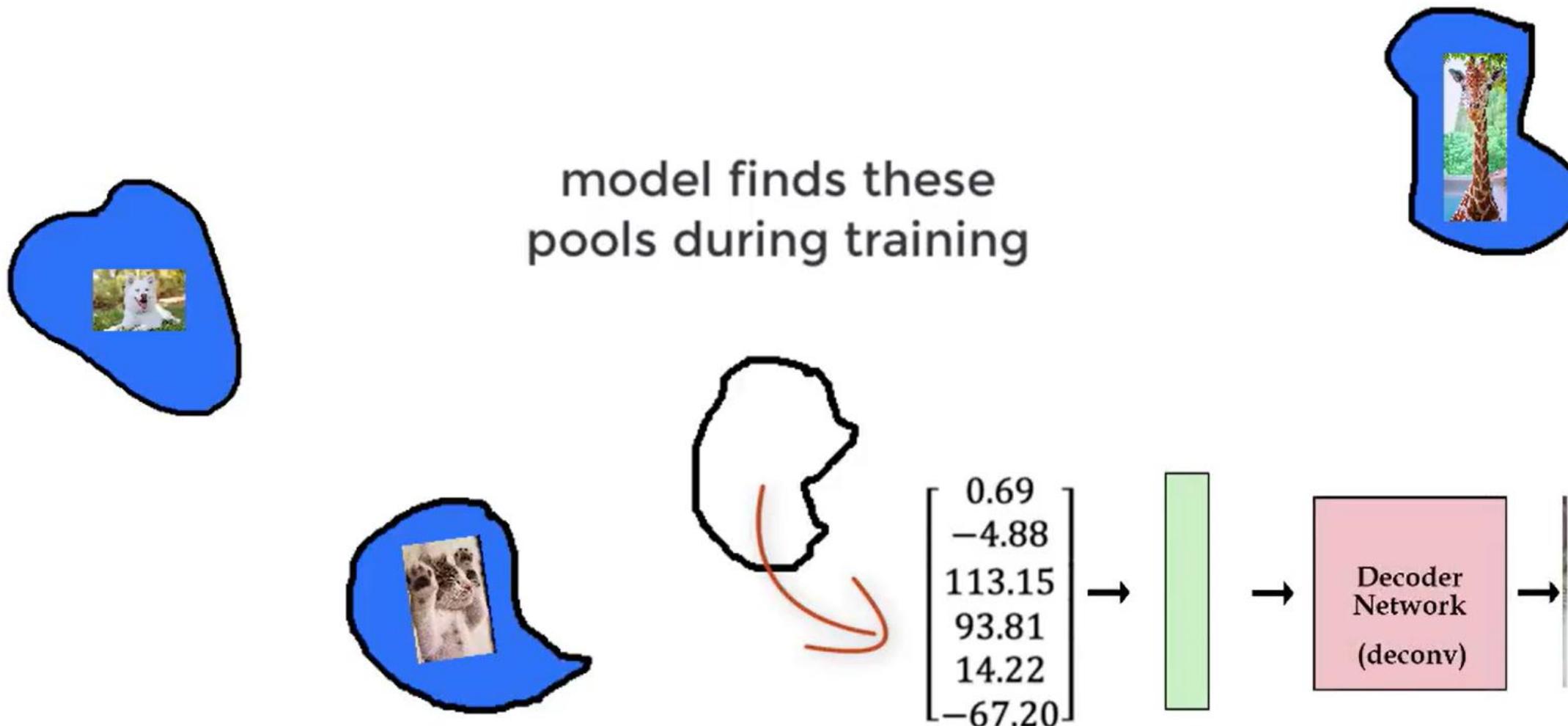
Giraffe
Distribution



Cat
Distribution

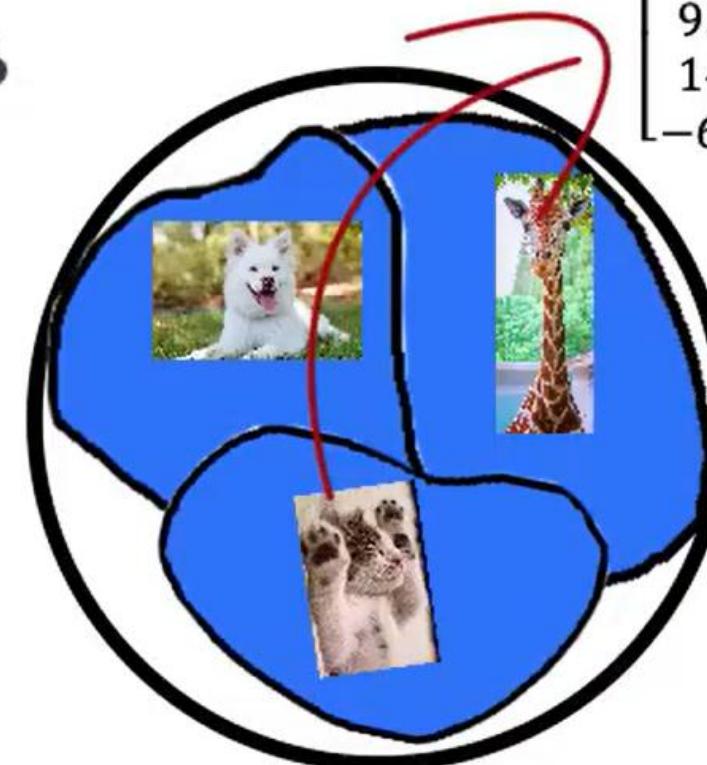


Training VAEs

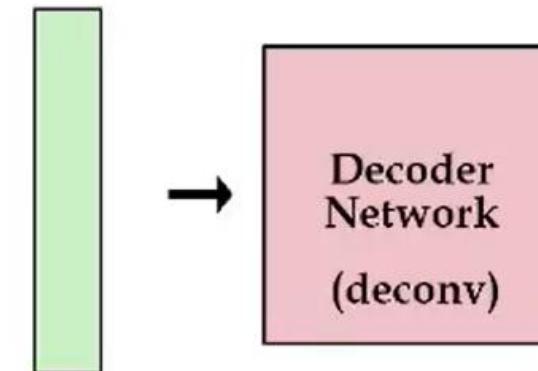


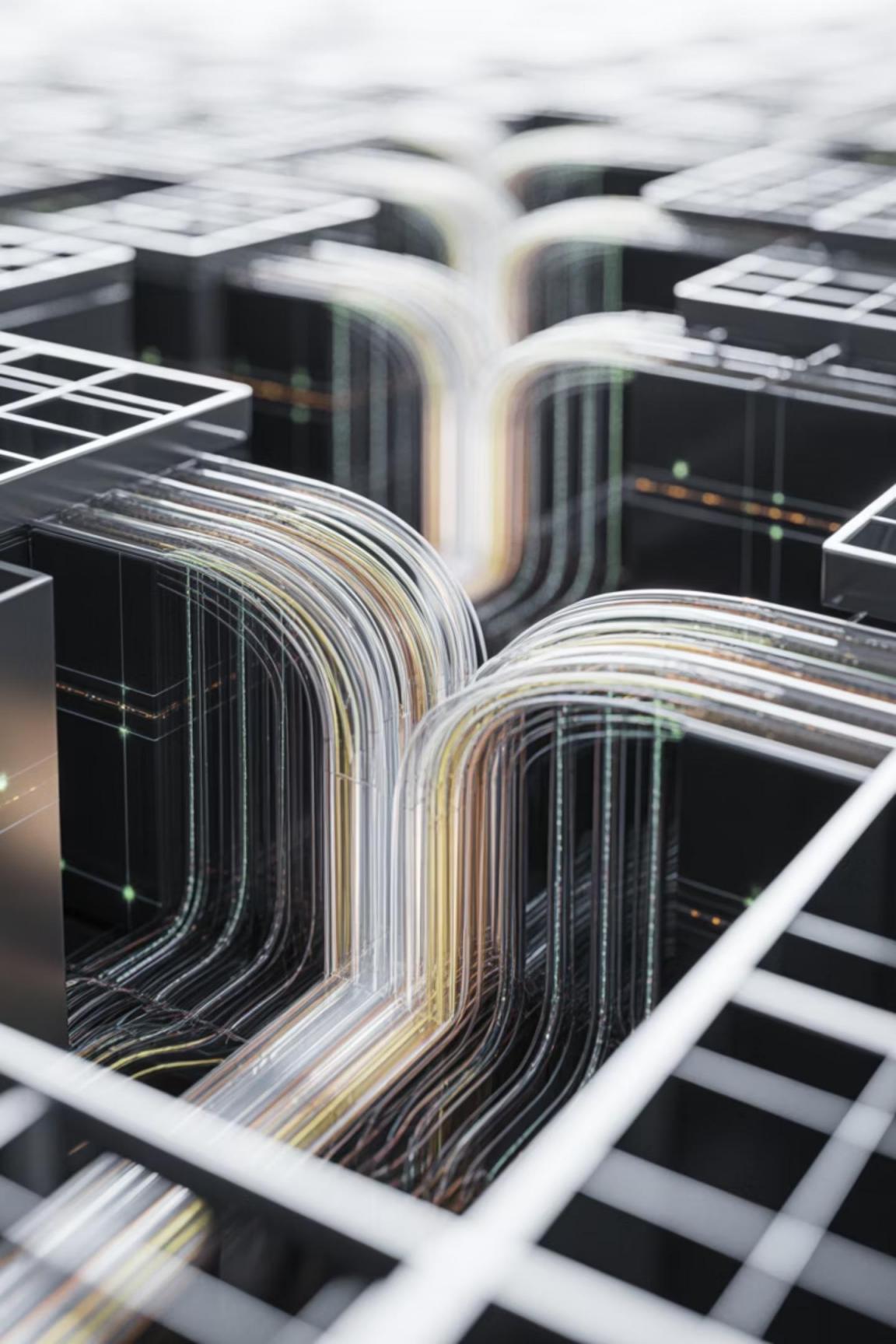
Testing VAEs

continuous
region



$$\begin{bmatrix} 0.69 \\ -4.88 \\ 113.15 \\ 93.81 \\ 14.22 \\ -67.20 \end{bmatrix}$$





El Truco de la Reparametrización

Problema Técnico

No se puede hacer backpropagation a través de un **muestreo aleatorio** porque introduce no-diferenciabilidad.

Solución Elegante

Reparameterization Trick:

$$Z = \mu + \sigma \epsilon$$

Donde $\epsilon \sim \mathcal{N}(0, 1)$ es ruido externo muestreado de una normal estándar. Esto permite que los gradientes fluyan hacia μ y σ .

Función de Pérdida del VAE

El entrenamiento de VAE balancea dos objetivos en conflicto:

$$L = \| X - \hat{X} \| + D_{KL}(N(\mu, \sigma) \parallel N(0, 1))$$



Término de Reconstrucción

Penaliza la diferencia entre la entrada original y la reconstrucción. Asegura que el modelo aprenda a reproducir los datos.



Divergencia KL

Obliga a que el espacio latente se parezca a una normal estándar, haciéndolo suave y continuo para permitir generación.

Ventajas y Desventajas de VAE

✓ Ventajas

- **Espacio latente continuo y ordenado:** ideal para interpolación
- **Entrenamiento estable:** sin el problema de equilibrio de GANs
- **Interpretabilidad:** las dimensiones latentes pueden tener significado
- **Generación controlada:** podemos navegar el espacio latente

✗ Desventajas

- **Imágenes borrosas:** pérdida MSE promedia píxeles
- **Falta de detalles finos:** pierde alta frecuencia
- **Menor realismo:** comparado con GANs





¿Por qué VAE genera imágenes borrosas?

La función de pérdida de **Error Cuadrático Medio (MSE)** promedia los píxeles, eliminando detalles de alta frecuencia como bordes nítidos y texturas finas.

Aquí es donde entran las Redes Generativas Adversarias...

Autoencoders vs Variable Autoencoders

Normal AutoEncoder	Variational AutoEncoder
<p>Why does this exist?</p> <ul style="list-style-type: none">- Learn a hidden representation of input (that “vector”)- Can NOT generate new data	<ul style="list-style-type: none">- Learns to generate new data
<p>What does this optimize?</p> <ul style="list-style-type: none">- Minimize reconstruction loss	<p>What does this optimize?</p> <ul style="list-style-type: none">- Minimize reconstruction loss + latent loss- Latent vectors are sampled from Gaussian Mixture



Parte 4: Redes Generativas Adversarias (GAN)

La Idea Revolucionaria de Ian Goodfellow (2014)

Sustituir la función de pérdida fija por una red neuronal que aprende a ser la función de pérdida

Los Dos Jugadores del Juego

Generador (G)

"El Falsificador"

Intenta crear datos falsos que parezcan reales para engañar al discriminador. Aprende la distribución de los datos reales.

Discriminador (D)

"El Detective"

Intenta distinguir entre datos reales del dataset y datos falsos generados. Proporciona señal de aprendizaje al generador.

Arquitectura GAN: Flujo de Datos



El Juego Minimax

El entrenamiento de GAN es un **juego de suma cero** donde un jugador gana lo que el otro pierde:

$$\min_G \max_D V(D, G) = \mathbb{E}_x p_{data}(x) [\log D(x)] + \mathbb{E}_z p_z(z) [\log(1 - D(G(z)))]$$

Esta es la **fórmula fundamental** que define el entrenamiento adversarial.

Desglosando la Función Objetivo

Término 1: $\log D(x)$

El discriminador quiere **maximizar** esto: clasificar correctamente las muestras reales como reales ($D(x) \rightarrow 1$).

Término 2: $\log(1 - D(G(z)))$

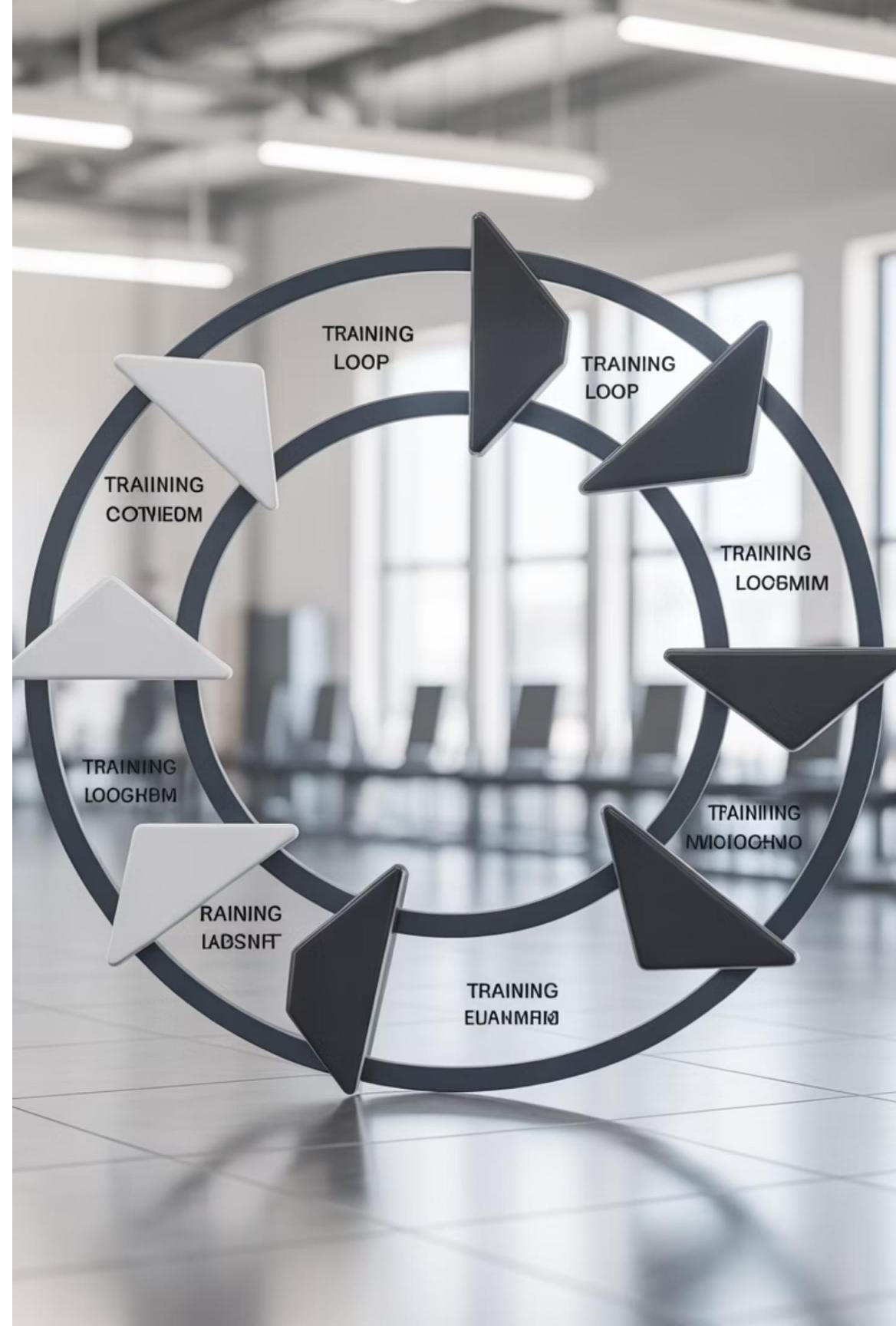
El discriminador quiere **maximizar** esto: clasificar las muestras falsas como falsas ($D(G(z)) \rightarrow 0$).

Objetivo del Generador

El generador quiere **minimizar** el segundo término: hacer que $D(G(z)) \rightarrow 1$, engañando al discriminador.

Algoritmo de Entrenamiento GAN

En cada iteración (epoch), alternamos entre entrenar el Discriminador y el Generador:

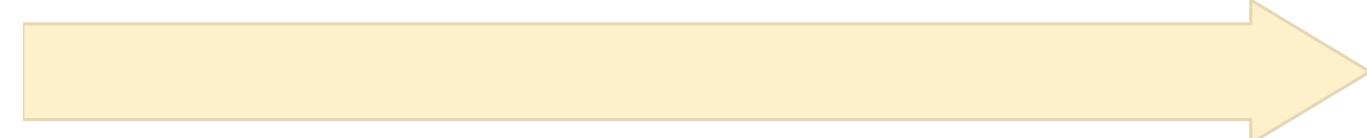


Paso 1: Entrenar el Discriminador



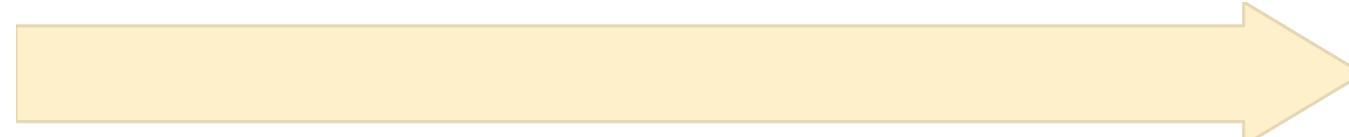
Muestras Reales

Tomar un lote de imágenes reales x del dataset de entrenamiento.



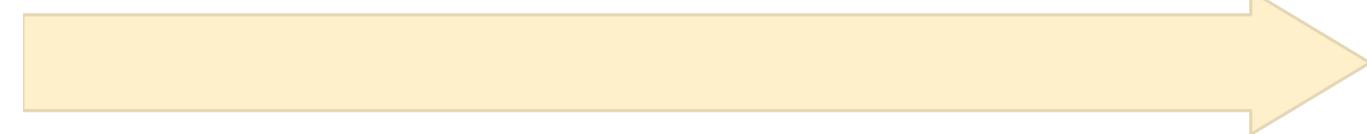
Muestras Falsas

Generar un lote de imágenes falsas $G(z)$ usando el generador con ruido aleatorio.



Actualización

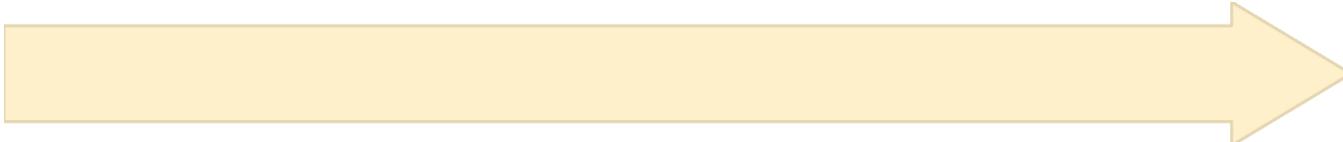
Ajustar pesos de D para clasificar x |to 1 y $G(z)$ |to 0.



Importante

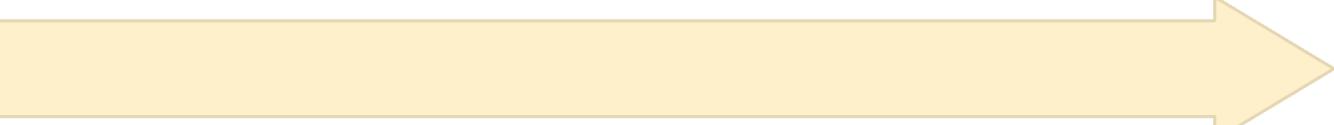
Congelar los pesos del Generador durante este paso.

Paso 2: Entrenar el Generador



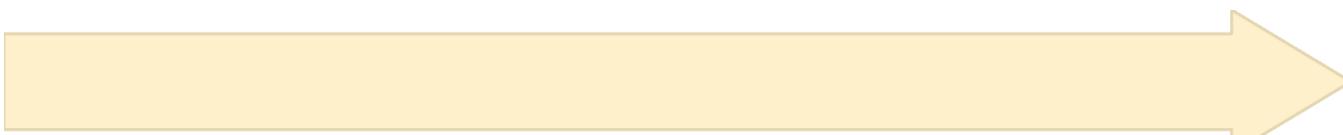
Generar Falsas

Generar nuevas imágenes falsas $G(z)$ con ruido aleatorio.



Pasar por D

Pasar las imágenes falsas por el discriminador, pero con etiquetas "reales" (queremos engañarlo).



Actualización

Ajustar pesos de G usando el error proporcionado por D para mejorar la calidad.



Importante

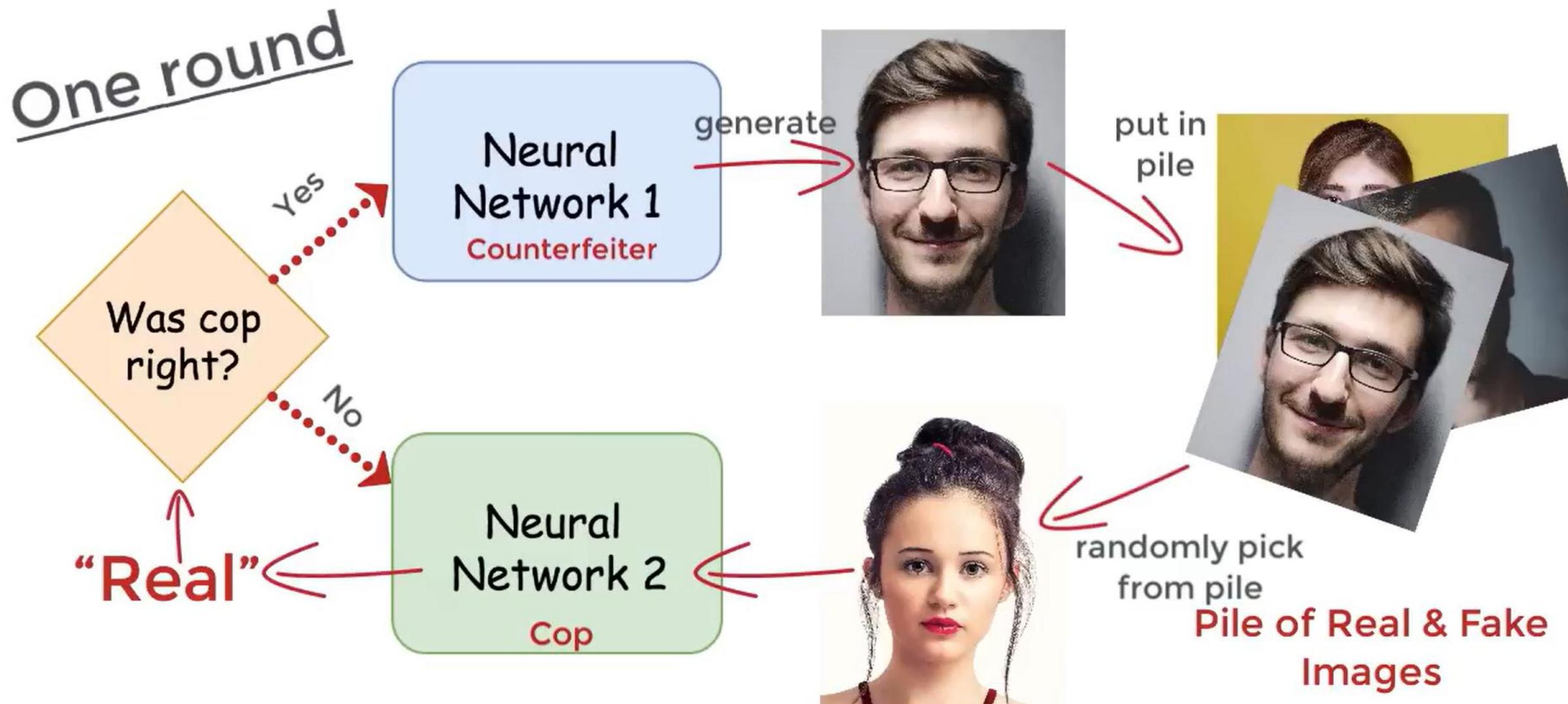
Congelar los pesos del Discriminador durante este paso.



Visualizando el Entrenamiento

A medida que avanza el entrenamiento, el Generador aprende a producir muestras cada vez más realistas que engañan al Discriminador.

Entrenamiento red GAN



Parte 5: Desafíos en el Entrenamiento de GANs



Desafío 1: Mode Collapse

El Problema

El Generador encuentra **una sola imagen** (o pocas imágenes) que engaña al Discriminador y la produce repetidamente, ignorando la diversidad del dataset.

Síntoma

Pides "gatos" y el modelo genera siempre el mismo gato blanco en la misma posición.



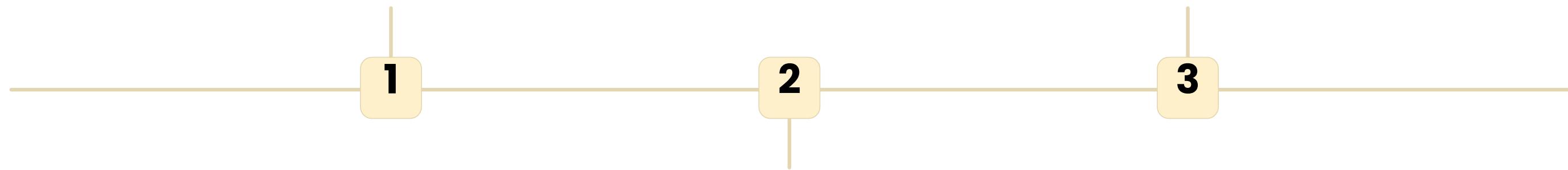
Soluciones

- Mini-batch discrimination
- Unrolled GANs
- Wasserstein Loss
- Múltiples discriminadores

Desafío 2: Desvanecimiento de Gradientes

Discriminador Perfecto

Si D es demasiado bueno (100% precisión), su función de pérdida se vuelve plana.



G deja de aprender

El entrenamiento se estanca y el Generador no mejora.

Gradientes Nulos

El Generador no recibe señal útil de aprendizaje. Los gradientes tienden a cero.

Solución: Wasserstein GAN (WGAN)

Introducida en 2017, WGAN usa la **Distancia de Wasserstein** (Earth Mover's Distance) en lugar de la divergencia Jensen-Shannon est\'andar.

Ventajas

- Proporciona gradientes m\'as suaves y continuos
- Estabiliza el entrenamiento significativamente
- La funci\'on de p\'erdida correlaciona con la calidad de las im\'agenes
- Reduce mode collapse





Parte 6: Evolución de las GANs

DCGAN (2015): La Primera GAN Estable

Deep Convolutional GAN introdujo mejores prácticas arquitectónicas que permitieron entrenar GANs de forma confiable.

Capas Convolucionales

Uso extensivo de CNNs en lugar de redes completamente conectadas.

Strided Convolutions

Reemplazó pooling con convoluciones con stride para aprendizaje jerárquico.

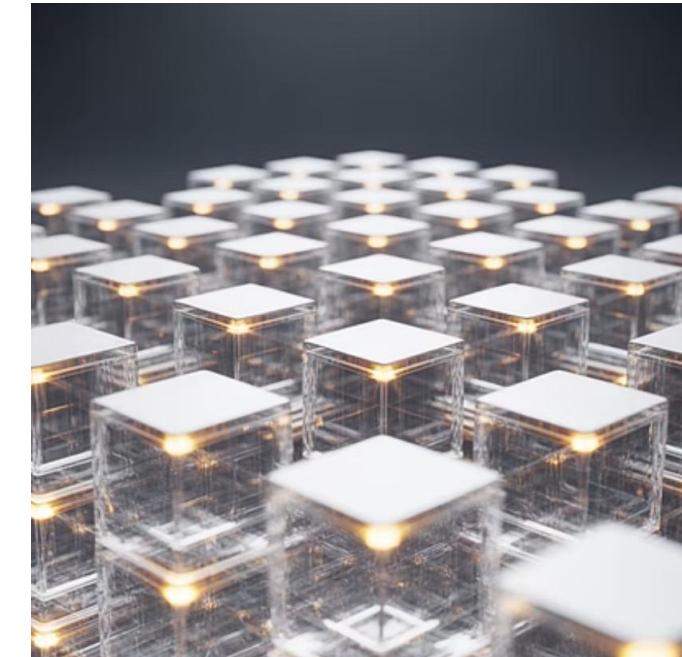
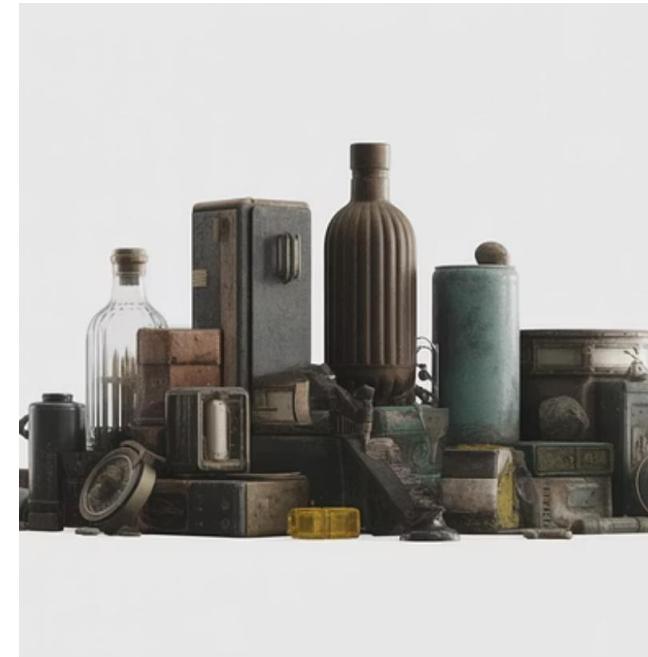
Batch Normalization

Normalización por lotes para estabilizar el entrenamiento.

ReLU y Tanh

Funciones de activación apropiadas en generador y discriminador.

Imágenes Generadas por DCGAN



DCGAN permitió por primera vez generar imágenes estables de 64×64 píxeles con calidad aceptable.



StyleGAN: El Estado del Arte en Rostros

Desarrollada por NVIDIA en 2018, StyleGAN representa el **estado del arte** en generación de rostros fotorrealistas.

Innovaciones de StyleGAN



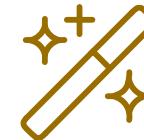
Mapping Network

Transforma el ruido latente z a un espacio intermedio w más desenredado y controlable.



Estilos por Capa

Permite controlar estilos "gruesos" (forma, pose) y "finos" (color de pelo, textura) independientemente.



Inyección de Ruido

Introduce ruido estocástico a nivel de capa para detalles finos como pecas o cabello suelto.

Control Jerárquico en StyleGAN

Estilos Gruesos

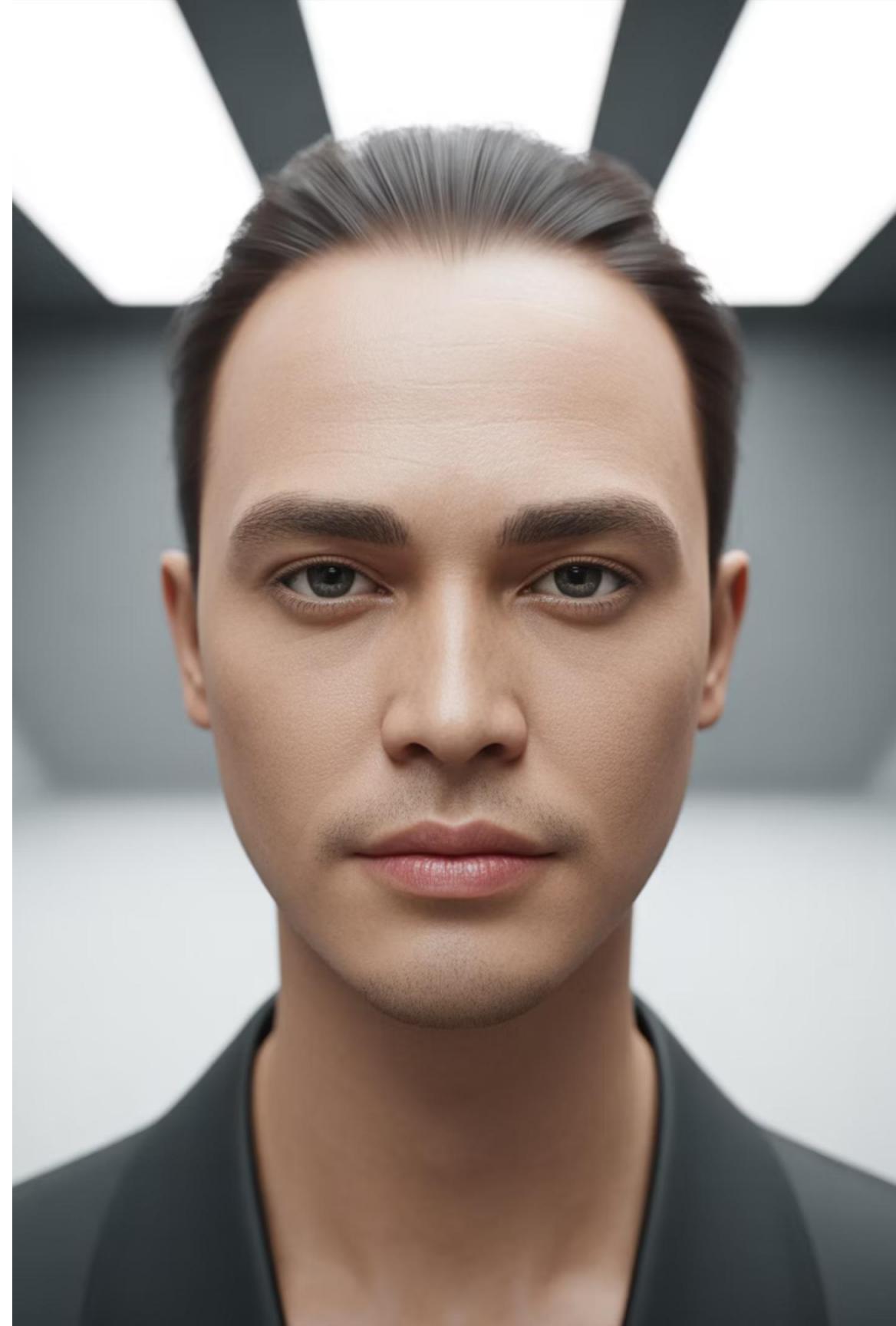
Controlados en las **capas tempranas**:

- Forma de la cara
- Pose y orientación
- Estructura general
- Género y edad

Estilos Finos

Controlados en las **capas finales**:

- Color de cabello
- Iluminación
- Microestructuras
- Textura de la piel



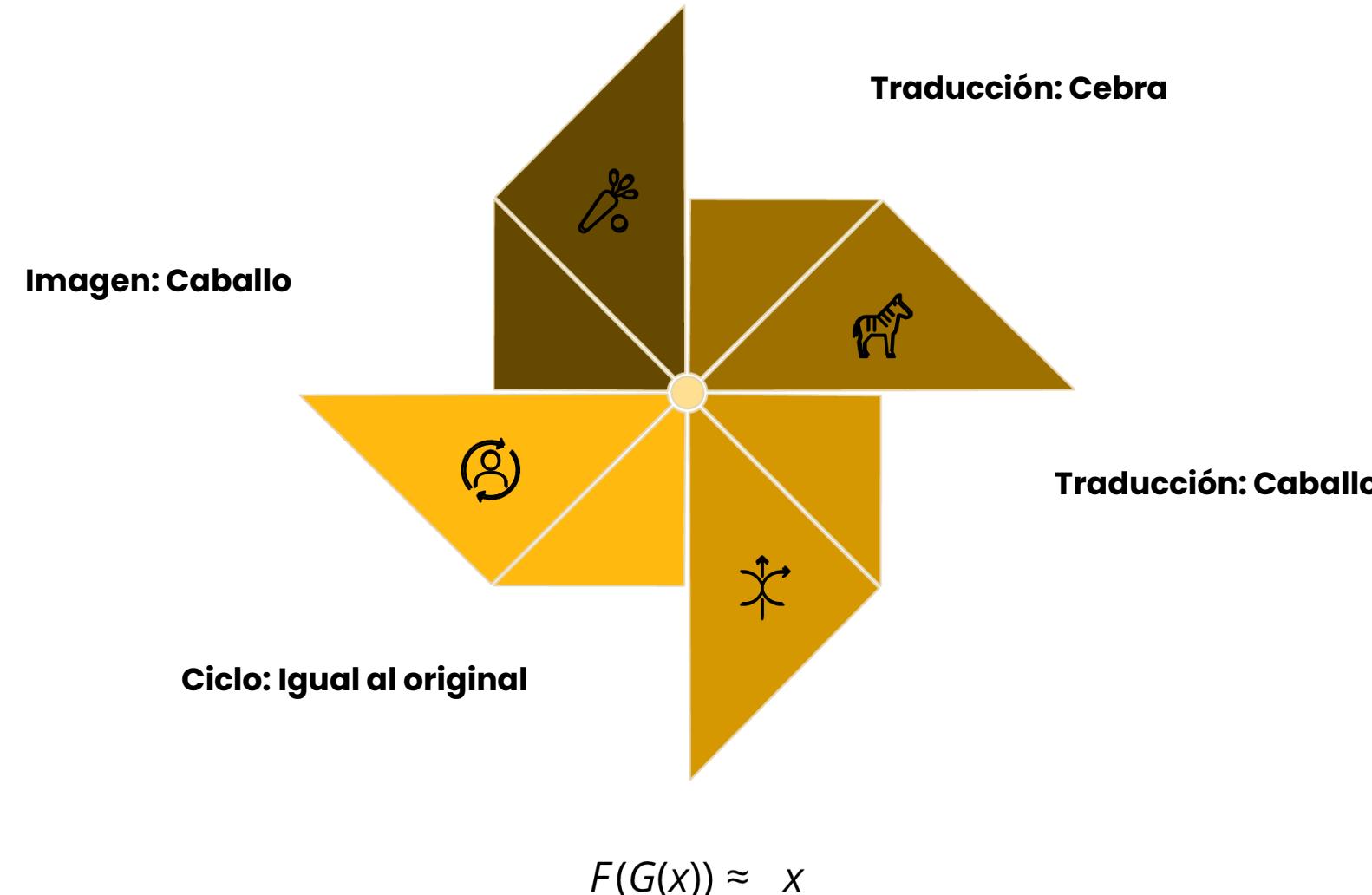


CycleGAN: Traducción sin Pares

CycleGAN permite **traducción imagen-a-imagen sin datos emparejados**. ¿Cómo convertir caballos en cebras sin fotos del mismo caballo como cebra?

Cycle Consistency Loss

La innovación clave: si traduzco de un dominio a otro y de vuelta, debería obtener la imagen original.



Esta restricción permite entrenar sin pares de datos correspondientes.

Aplicaciones de CycleGAN

Transferencia de Estilo

Verano a invierno, día a noche

Foto a Arte

Convertir fotos en pinturas estilo
Monet

Mapas

Vista satelital a mapa de calles



Parte 7: Aplicaciones Reales de GANs

Aplicaciones de GANs en la Industria



Super-Resolución

Convertir videos HD a 4K o 8K. Usado en streaming y upscaling (DLSS de NVIDIA).



Inpainting

Restaurar partes dañadas de fotografías antiguas o remover objetos de imágenes.



Diseño de Moda

Generación de nuevos diseños de ropa, avatares virtuales y modelado 3D.



Datos Sintéticos

Generar datos médicos (rayos X, MRI) preservando privacidad de pacientes.

GANs en Medicina



Las GANs permiten generar **imágenes médicas sintéticas** que preservan la privacidad del paciente pero mantienen características clínicas relevantes.

Beneficios

- Aumentar datasets pequeños de enfermedades raras
- Entrenar modelos sin exponer datos sensibles
- Balancear clases desbalanceadas
- Simular progresiones de enfermedades



Deepfakes: La Cara Oscura

Las GANs permitieron el **intercambio de rostros** (face swapping) con fidelidad sin precedentes, creando videos falsos extremadamente convincentes.

Desafíos Éticos de los Deepfakes

Desinformación Política

Videos falsos de figuras públicas diciendo cosas que nunca dijeron, manipulando opinión pública.

Suplantación de Identidad

Bypass de sistemas biométricos de seguridad facial. Fraudes y extorsiones.

Necesidad de Detección

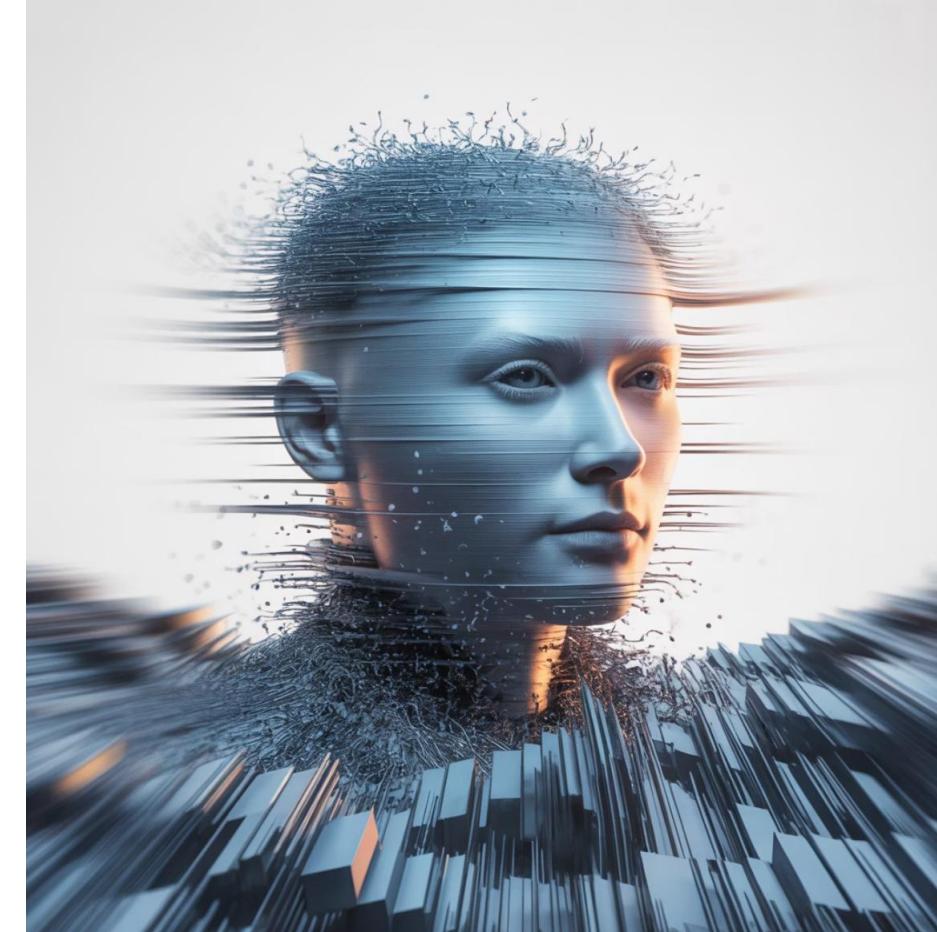
Desarrollo de "discriminadores forenses" para detectar contenido generado artificialmente.

La tecnología requiere marcos éticos y legales urgentes.

¿Por qué nos Movemos hacia Difusión?

Problemas Persistentes de GANs

- Entrenamiento muy **inestable**
- Mode collapse recurrente
- Dificultad para equilibrio Nash
- Problemas de escalabilidad a texto-a-imagen
- Difícil de condicionar con prompts complejos



Modelos de Difusión

Aunque las GANs son rápidas (un solo paso forward), los modelos de difusión ofrecen **mayor estabilidad y diversidad**.

GANs vs Variable Autoencoders

Generative Adversarial Nets	Variational AutoEncoder
<p>How does this learn to generate data?</p> <ul style="list-style-type: none">- Generator and Discriminator play a minimax game- Consists of a Generator and Discriminator networks	<p>How does this learn to generate data?</p> <ul style="list-style-type: none">- Minimize reconstruction loss, latent loss- Consists of an Encoder and Decoder
<p>How stable is training?</p> <ul style="list-style-type: none">- Requires finding a “Nash Equilibrium” during training.	<p>How stable is training?</p> <ul style="list-style-type: none">- Closed form solution to determine “end-of-train” phase

GANs vs Variable Autoencoders

Generative Adversarial Nets	Variational AutoEncoder
<p>How good are the generated images?</p>  <ul style="list-style-type: none">- Sharper images generated compared to VAEs	 <p>Blurry</p> <ul style="list-style-type: none">- Reconstruction Loss: make sure output is similar to input image- Latent loss: Vector takes fixed range of values

Comparación: VAE vs GAN

VAE

- ✓ Entrenamiento estable
- ✓ Espacio latente continuo
- ✓ Interpolación suave
- ✗ Imágenes borrosas
- ✗ Menor realismo

GAN

- ✓ Imágenes ultra nítidas
- ✓ Fotorrealismo extremo
- ✓ Detalles de alta frecuencia
- ✗ Entrenamiento inestable
- ✗ Mode collapse

Resumen: Evolución de Modelos Generativos



Conceptos Clave para Recordar

Modelos Generativos

Aprenden $P(X)$ para generar nuevas muestras, no solo clasificar.

Variables Latentes

Simplificación clave para manejar alta dimensionalidad.

Entrenamiento Adversarial

Juego minimax entre generador y discriminador.

Aplicaciones y Ética

Poder tecnológico requiere responsabilidad social.

Recursos Adicionales



Papers Fundamentales

- Goodfellow et al. (2014): Generative Adversarial Networks
- Kingma & Welling (2013): Auto-Encoding Variational Bayes
- Karras et al. (2019): StyleGAN - A Style-Based Generator



Implementaciones

- TensorFlow GAN library
- PyTorch Lightning GAN examples
- StyleGAN2-ADA (NVIDIA official)

Ejemplo Gan

En este ejercicio implementaremos una **Conditional DCGAN** (Deep Convolutional GAN) utilizando el dataset **Fashion-MNIST**.

A diferencia de una GAN tradicional que genera imágenes aleatorias del espacio latente, una **cGAN** recibe una **etiqueta (condición)** tanto en el Generador como en el Discriminador.





¿Preguntas?

Módulo 6.2 - Modelos Probabilísticos Clásicos y GANs

Universidad Técnica Federico Santa María

Departamento de Informática