

# Deep Generative Models

Generative Adversial Networks



Marco Teran



UNIVERSIDAD  
SERGIO ARBOLEDA

Noviembre 2020- Bogotá

## ¿Cuál rostro es real?



## Aprendizaje supervisado vs. No supervisado

## Aprendizaje supervisado

- **Datos:**  $(x, y)$   
donde  $x$  es un dato,  $y$  es una etiqueta
  - **Objetivo:** Aprender la función de mapeo  $x \rightarrow y$
  - **Aplicaciones:** Clasificación, regresión, detección de objetos, segmentación semántica, etc.

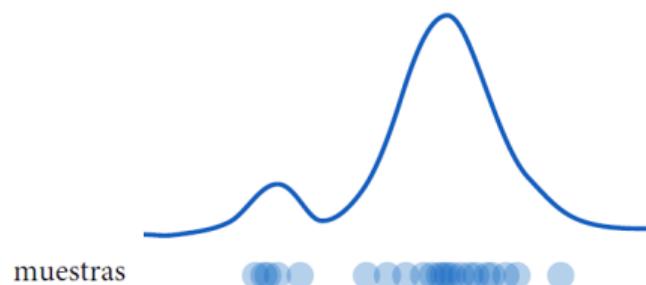
## Aprendizaje no supervisado

- **Datos:**  $x$   
 $x$  es un dato, no hay etiquetas
  - **Objetivo:** Aprender alguna estructura oculta o subyacente de los datos  
 $x \rightarrow y$
  - **Aplicaciones:** Agrupación, reducción de características o dimensionalidad, etc.

## Modelado generativo

**Objetivo:** Tomar como entrada muestras de entrenamiento de alguna distribución y aprender un modelo que represente esa distribución Estimación de la densidad

## Estimación de la densidad



## Generación de muestras



## Input samples

Training data  $\sim P_{data}(x)$

### Generated samples

Generated  $\sim P_{model}(x)$

¿Cómo podemos aprender la  $P_{model}(x)$  similar a la  $P_{data}(x)$ ?

## ¿Por qué modelos generativos? Debiasing

Capacidad de descubrir las variables latentes subyacentes en un conjunto de datos



Color de piel homogéneo, postura

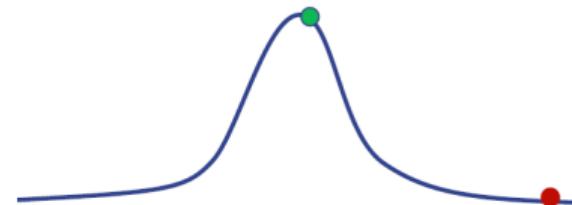


Color de piel diverso, postura, iluminación

¿Cómo podemos usar las distribuciones latentes para crear conjuntos de datos justos y representativos?

# ¿Por qué modelos generativos? Detección de valores atípicos (outliers)

- **Problema:** ¿Cómo podemos detectar cuando nos encontramos con algo nuevo o raro?
- **Estrategia:** Apalancar la generación de modelos, detectan valores atípicos en la distribución
- Utilizan valores atípicos durante el entrenamiento para mejorar aún más



El 95% de los datos de conducción:

(1) soleado, (2) autopista, (3) carretera recta



Detectar valores atípicos para evitar comportamientos impredecibles durante el entrenamiento



Casos extremos



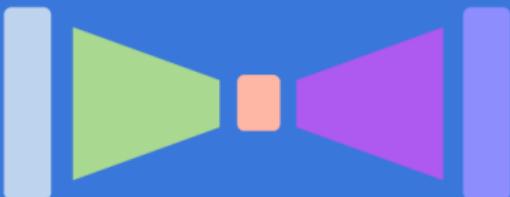
Mal clima



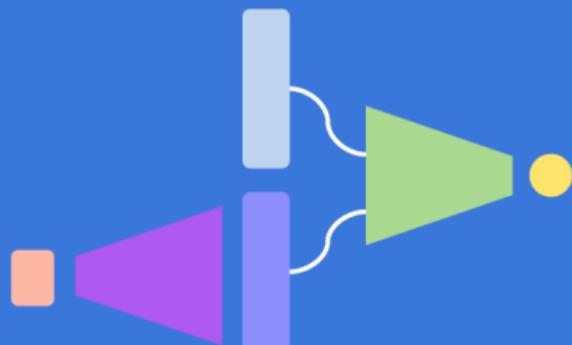
Peatones

# Modelos variables latentes

Autoencoders and Variational  
Autoencoders (VAEs)



Generative Adversarial  
Networks (GANs)



# ¿Qué es una variable latente?

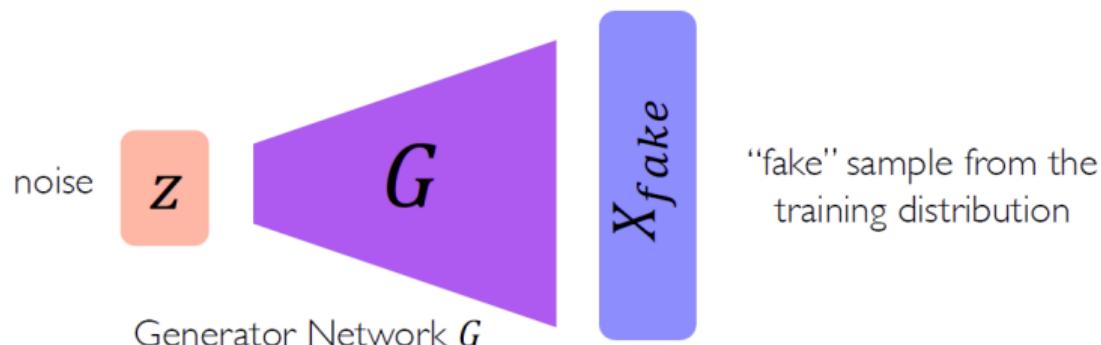


¿Podemos aprender **verdaderos factores explicativos**, por ejemplo, las variables latentes, sólo a partir de los datos observados?

# **Generative Adversarial Networks (GANs)**

# ¿Y si sólo queremos tomar una muestra?

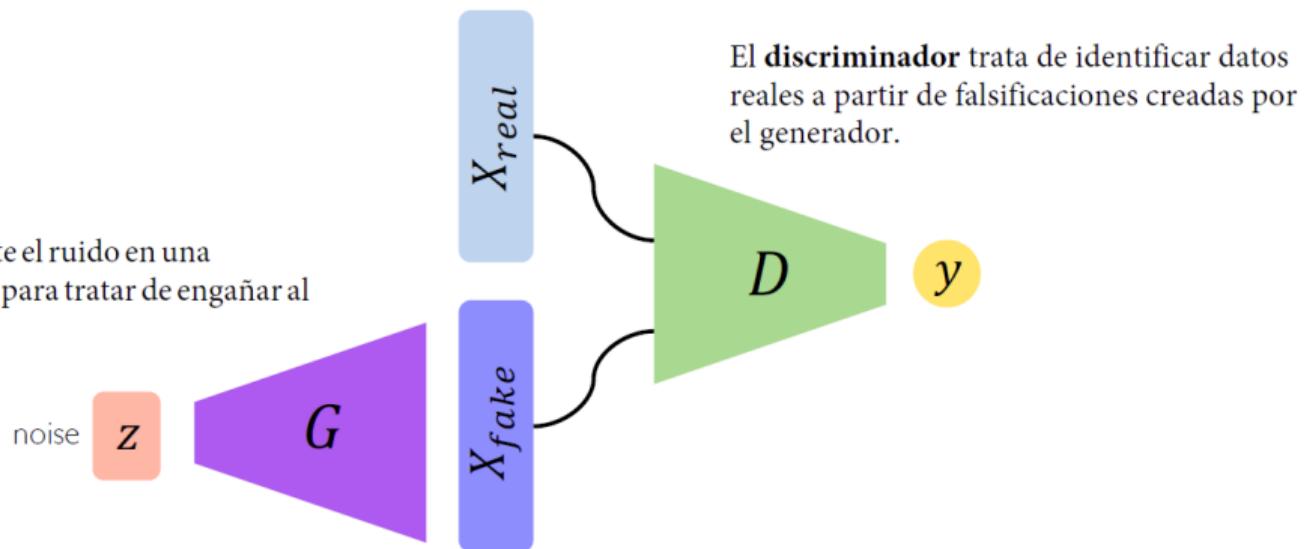
- **Idea:** no modelar explícitamente la densidad, y en su lugar sólo muestrear para generar nuevas instancias.
- **Problema:** querer tomar muestras de una distribución compleja... ¡no se puede hacer esto directamente!
- **Solución:** muestrear a partir de algo simple (ruido), aprender una transformación a la distribución de la formación.



# Generative Adversarial Networks (GANs)

Las **Redes Generativas Adversas (GANs)** son una forma de hacer un modelo generativo haciendo que dos redes neuronales compitan entre sí.

El **generador** convierte el ruido en una imitación de los datos para tratar de engañar al discriminador.



# Intuición detrás de las GANs

El **generador** parte del ruido para intentar crear una imitación de los datos.

Generator



# Intuición detrás de las GANs

El **discriminador** mira tanto los datos reales como los falsos creados por el generador.

Discriminator



Generator



## Intuición detrás de las GANs

El **discriminador** mira tanto los datos reales como los falsos creados por el generador.

Discriminator



Generator



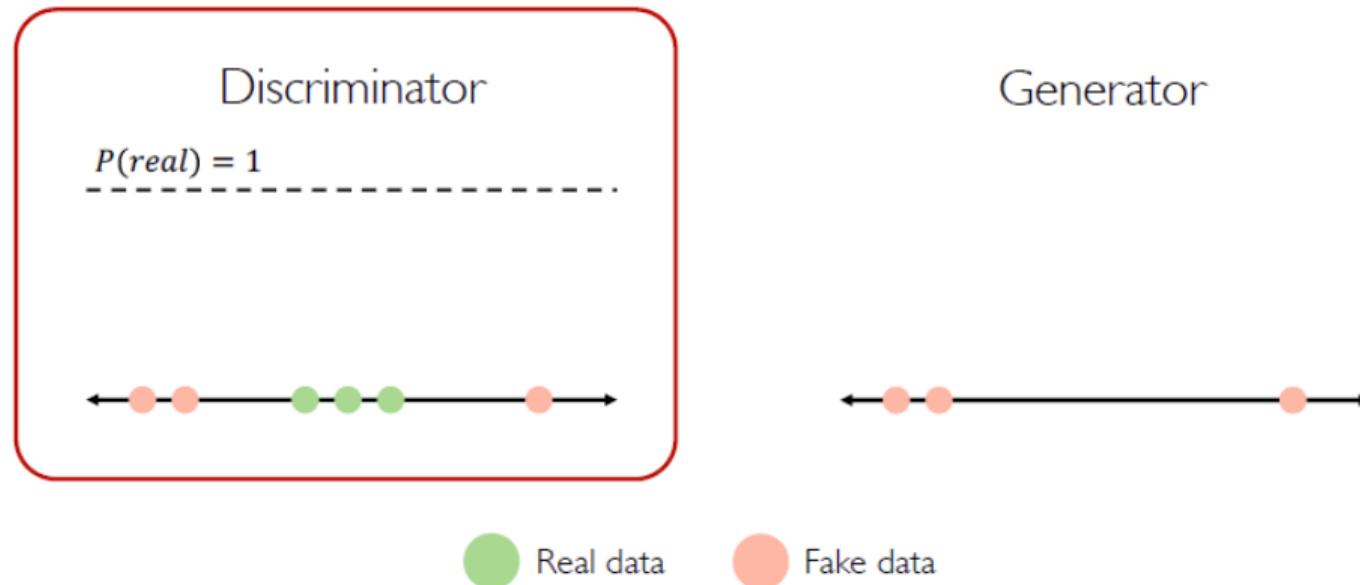
Real data



Fake data

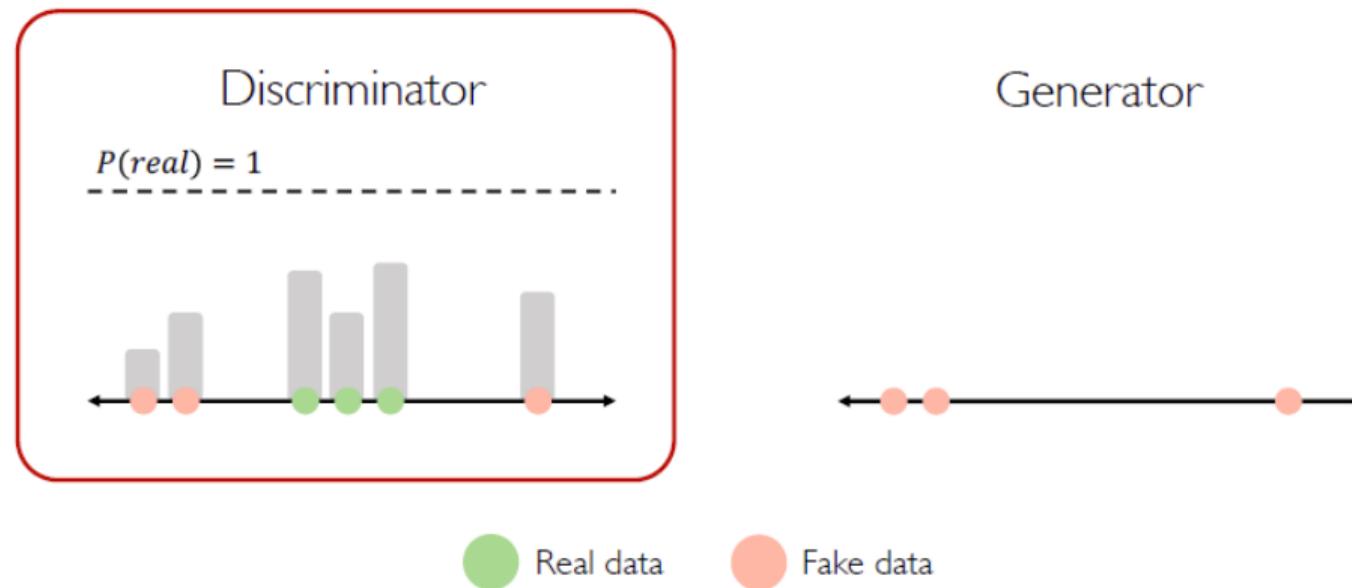
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



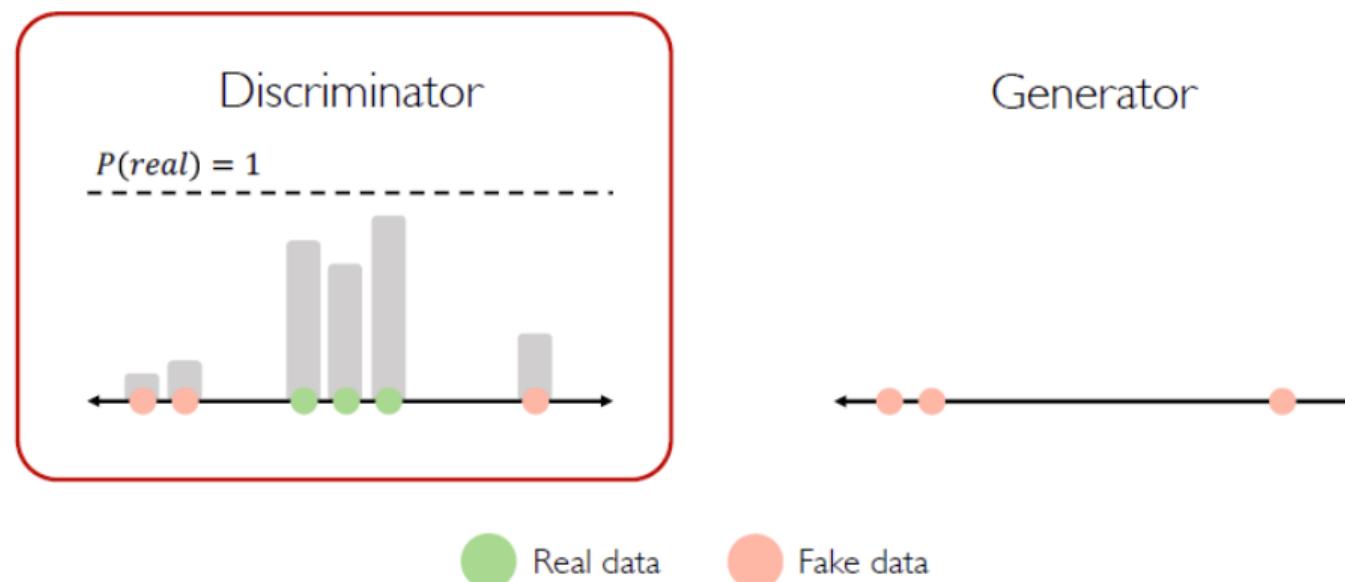
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



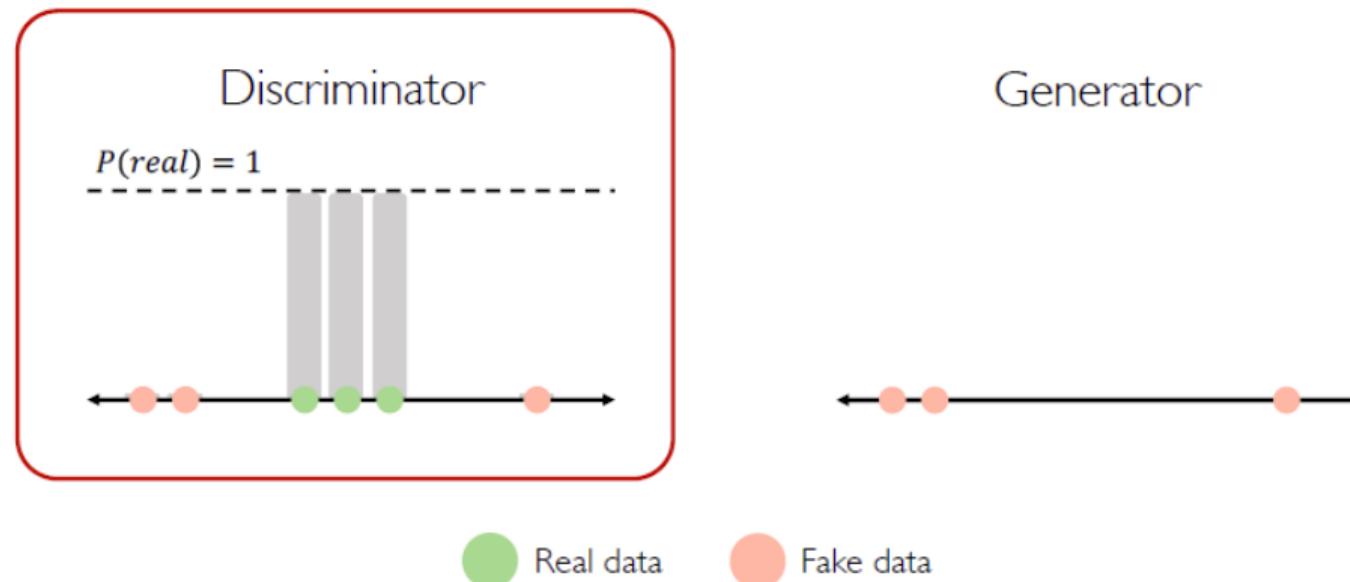
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



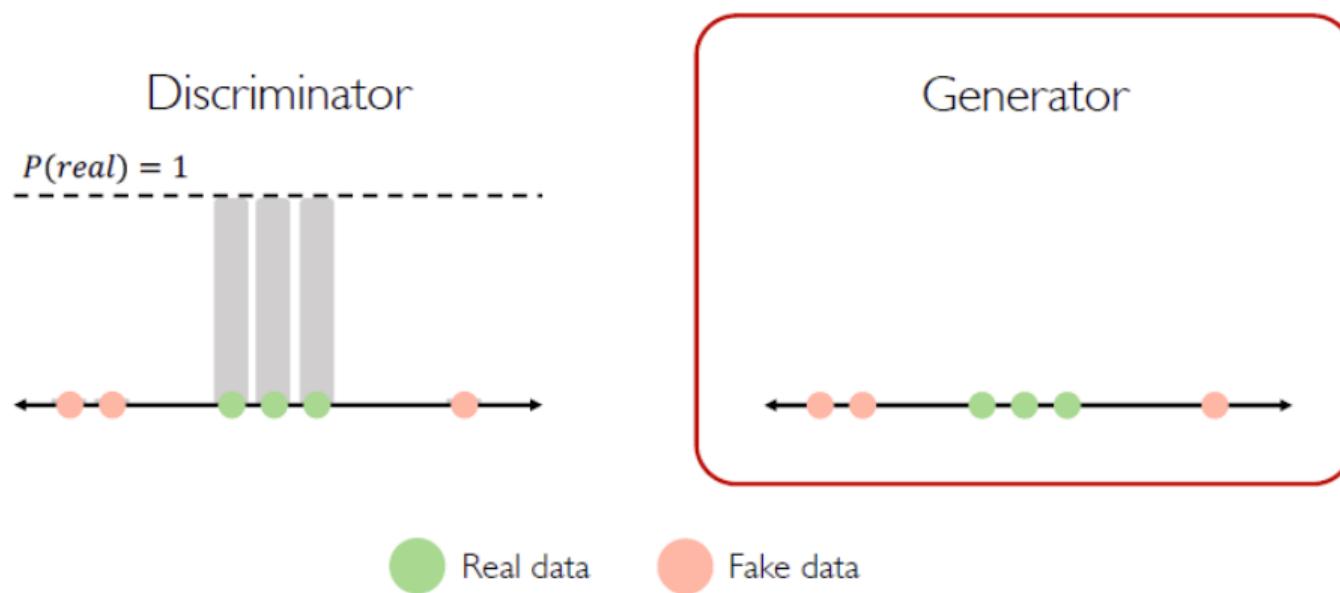
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



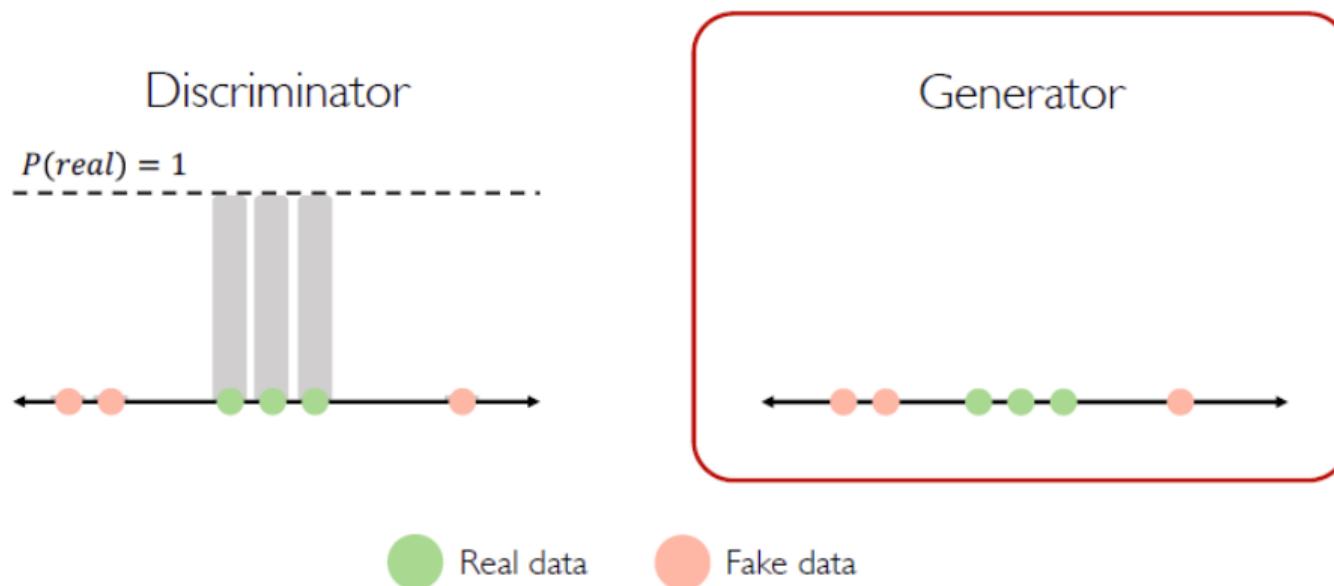
# Intuición detrás de las GANs

El **generador** trata de mejorar su imitación de los datos.



# Intuición detrás de las GANs

El **generador** trata de mejorar su imitación de los datos.



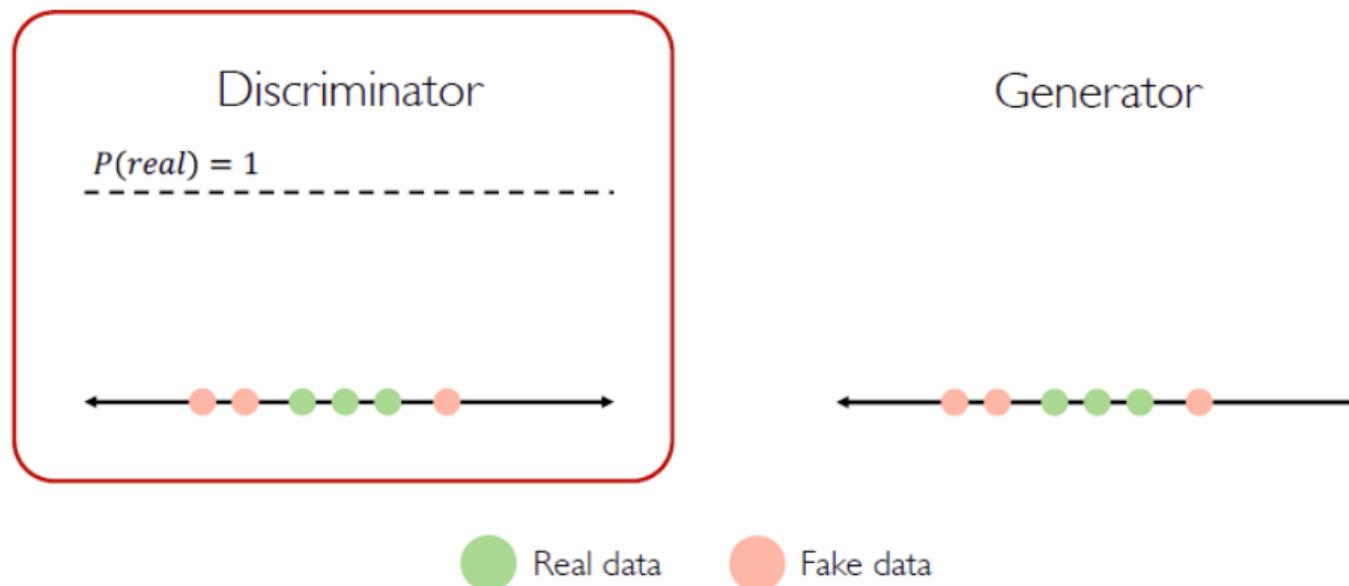
# Intuición detrás de las GANs

El **generador** trata de mejorar su imitación de los datos.



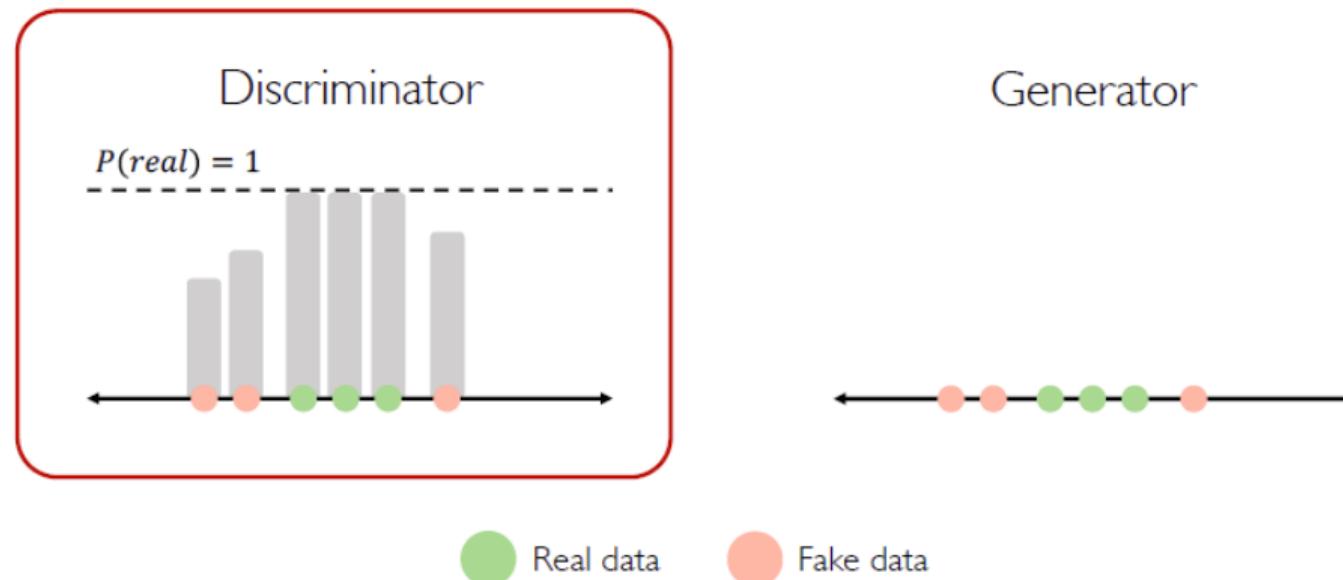
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



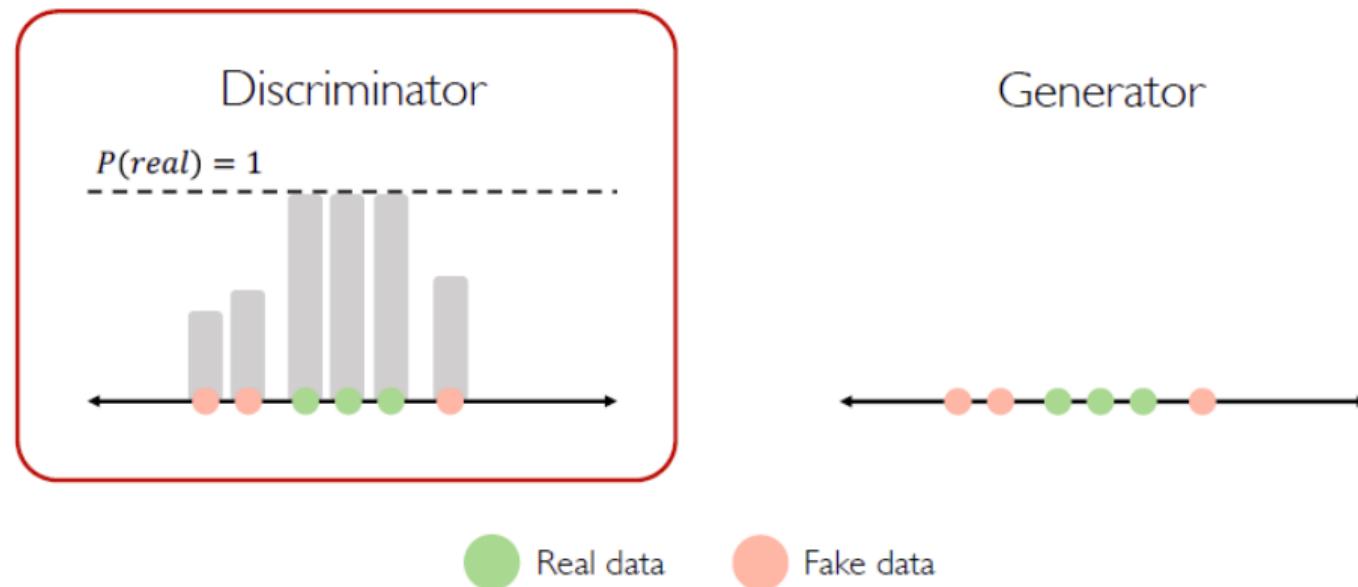
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



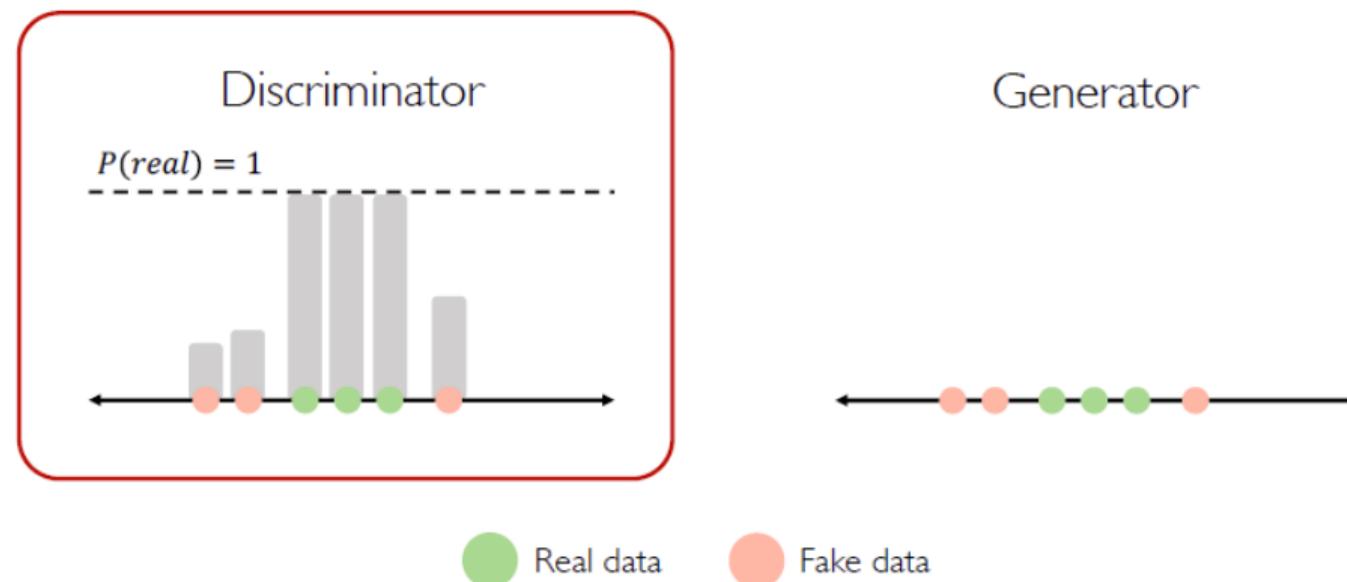
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



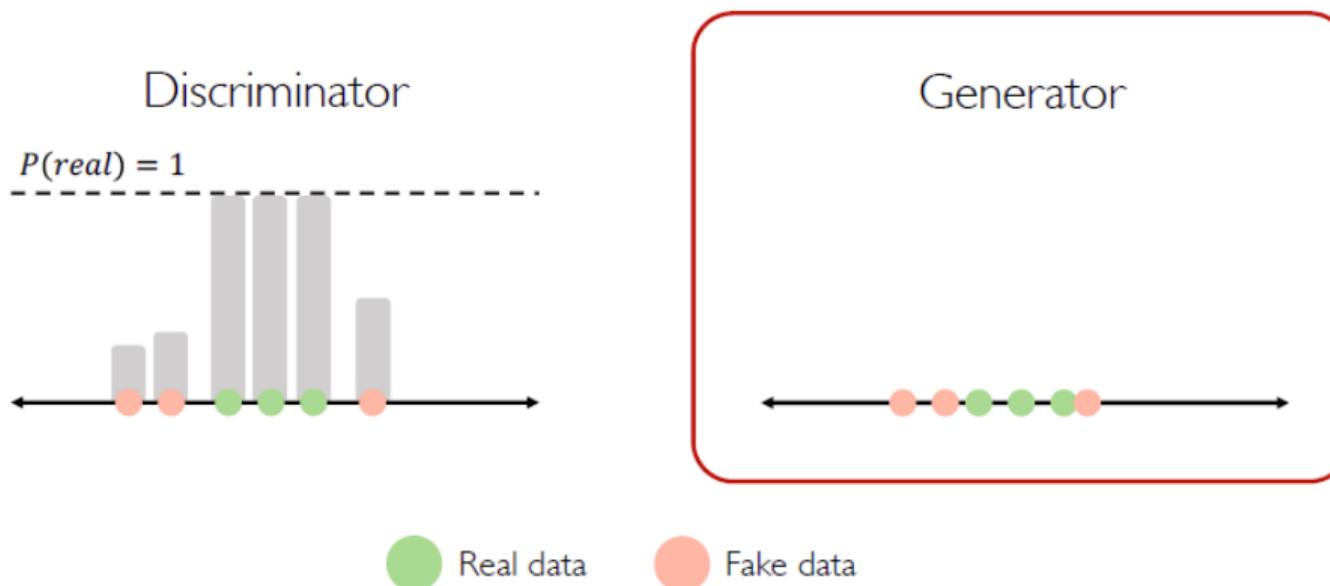
# Intuición detrás de las GANs

El **discriminador** trata de predecir lo que es real y lo que es falso.



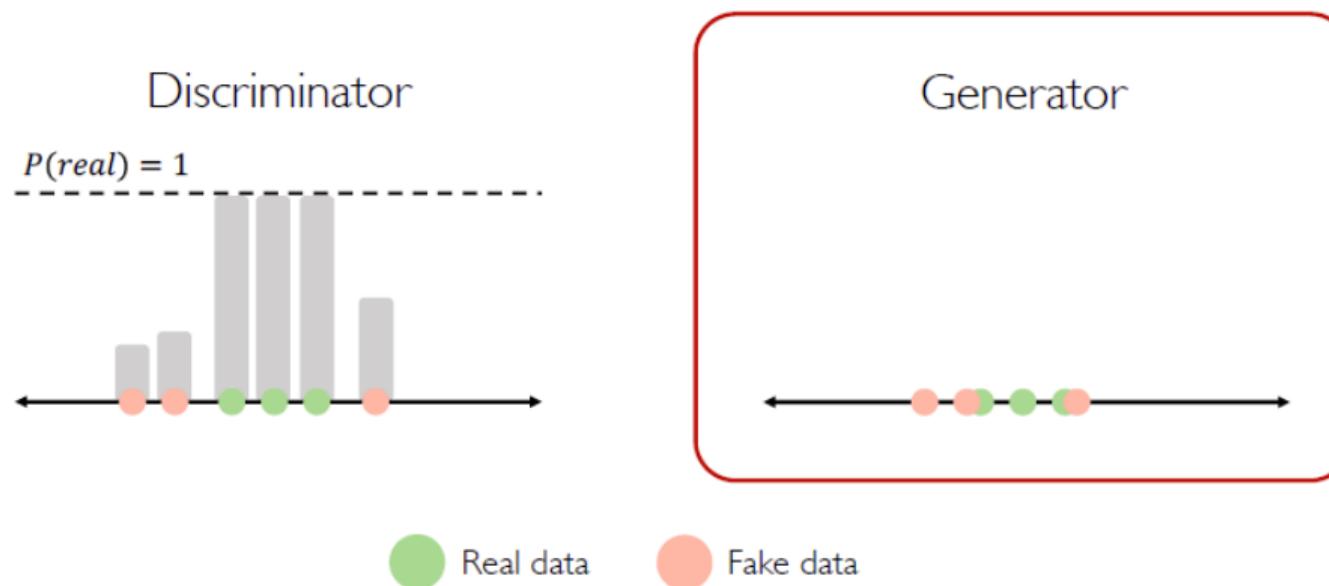
# Intuición detrás de las GANs

El **generador** trata de mejorar su imitación de los datos.



# Intuición detrás de las GANs

El **generador** trata de mejorar su imitación de los datos.



# Intuición detrás de las GANs

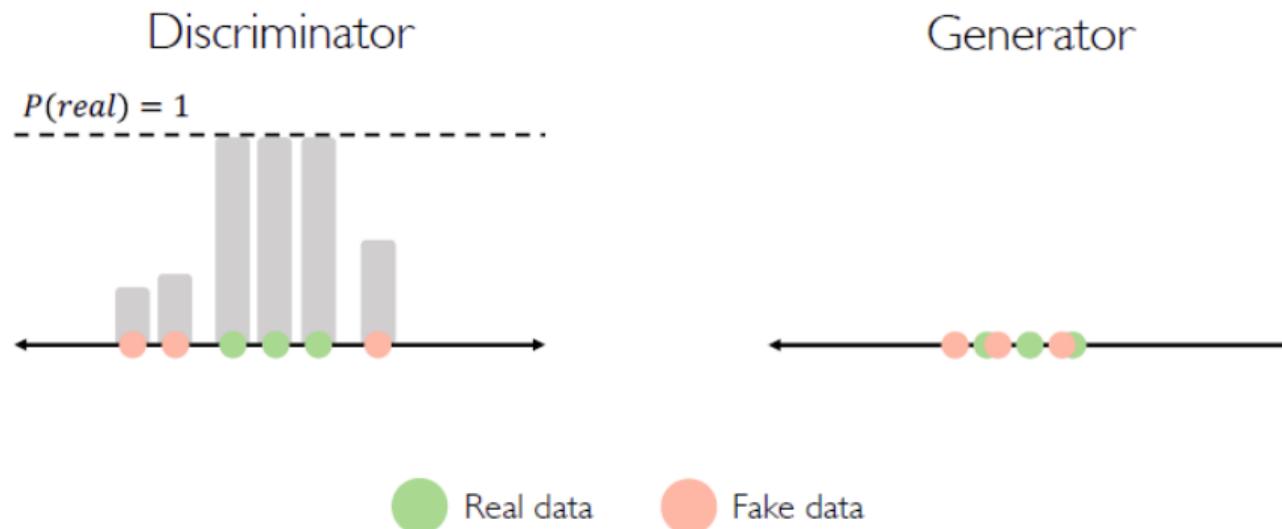
El **generador** trata de mejorar su imitación de los datos.



# Intuición detrás de las GANs

El **discriminador** trata de identificar los datos reales de las falsificaciones creadas por el generador.

El **generador** trata de crear imitaciones de datos para engañar al discriminador.



## Entrenando GANs

El **discriminador** trata de identificar los datos reales de las falsificaciones creadas por el generador.

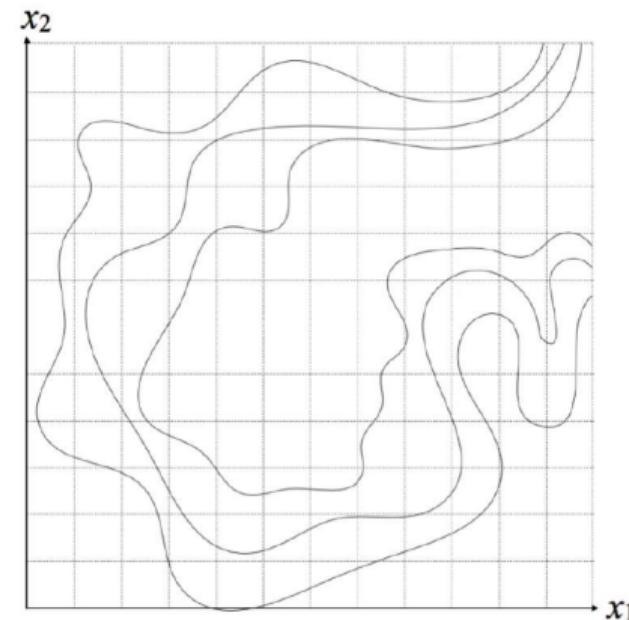
El **generador** trata de crear imitaciones de datos para engañar al discriminador.

**Entrenar** una GAN se realiza a través de un juego de **minimax**:

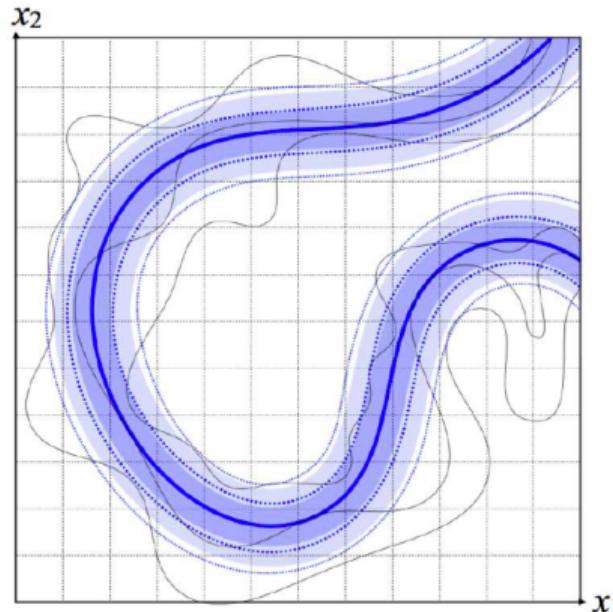
$$\min_{\Theta_g} \max_{\Theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\Theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\Theta_d}(G_{\Theta_g}(z))) \right]$$

- El **discriminador** quiere maximizar el objetivo  $D(x)$  cerca de 1,  $D(G(z))$  cerca de 0.
  - El **generador** quiere minimizar el objetivo  $D(G(z))$  cerca de 1.

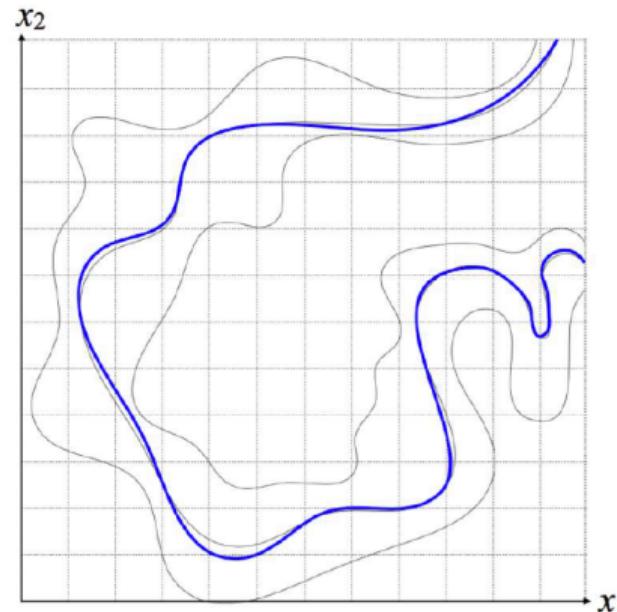
## ¿Por qué GSNs



## ¿Por qué GSNs



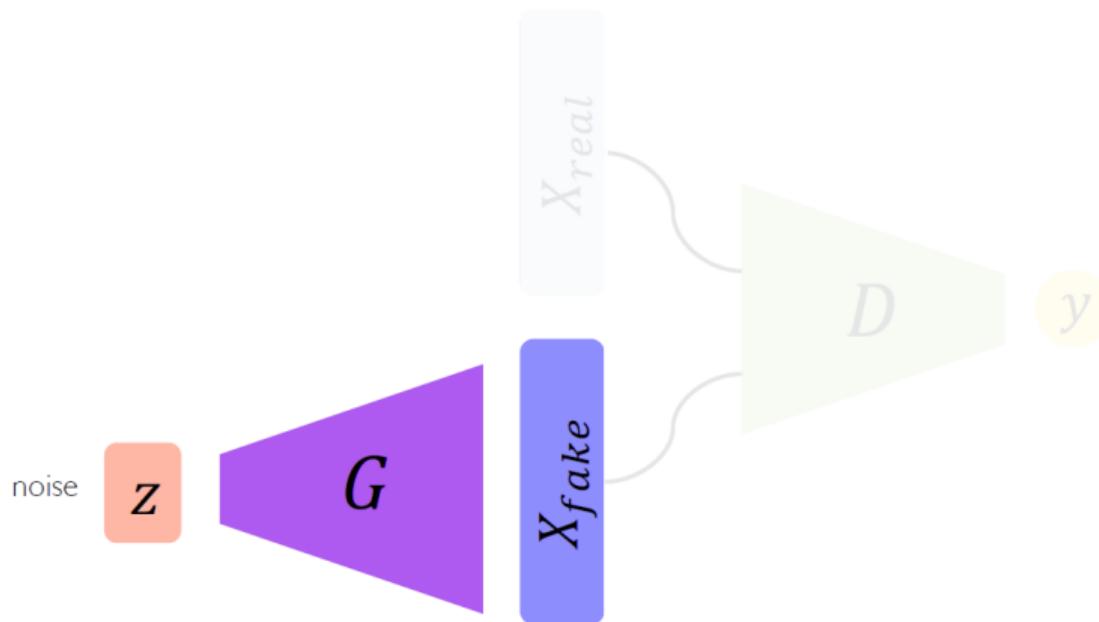
more traditional max-likelihood approach



GAN

# Generando nuevos datos con GANs

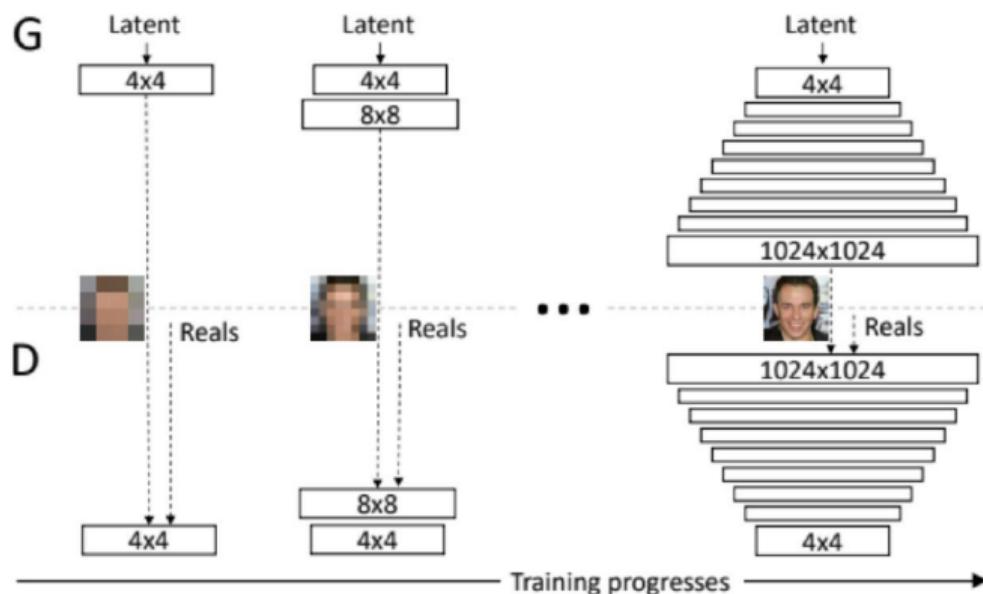
Después del entrenamiento, usa la red de generadores para crear nuevos datos que nunca se han visto antes.



# GANs: avances recientes

# Progressive growing of GANs (NVIDIA)

Crecimiento progresivo de las GAN



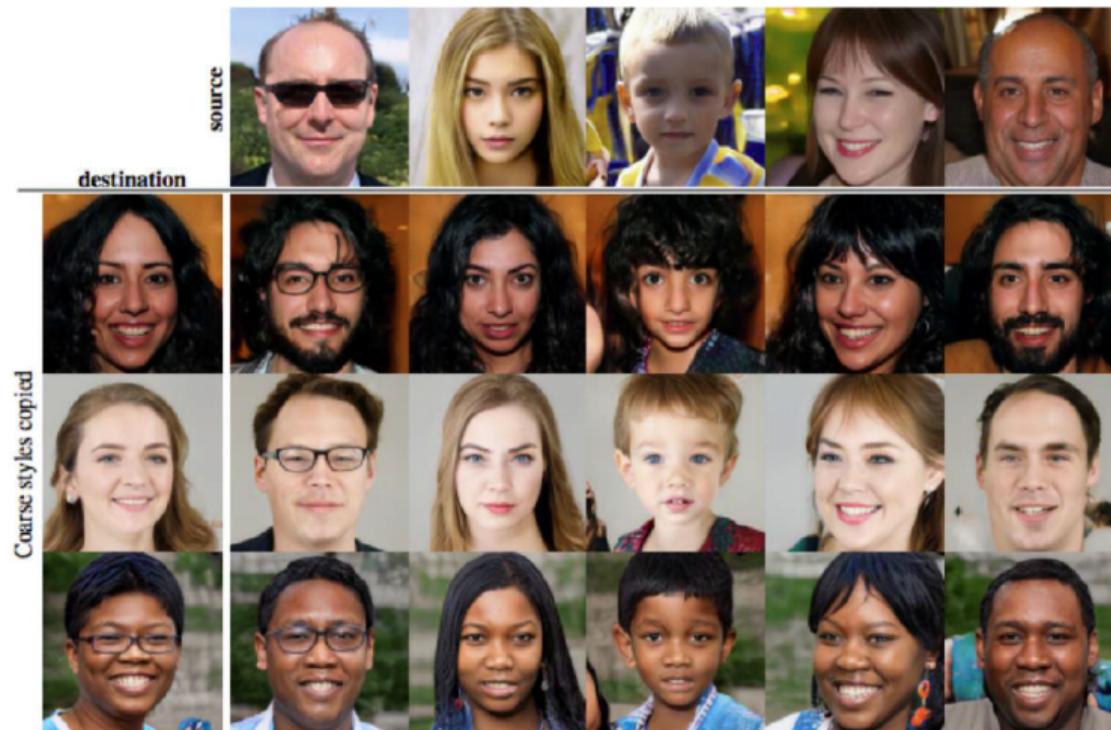
# Progressive growing of GANs: results



# Generador basado en el estilo: resultados

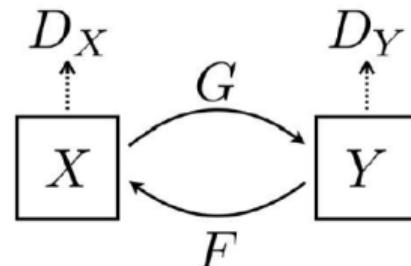


# Transferencia basada en el estilo: resultados



# CycleGAN: transformación del dominio

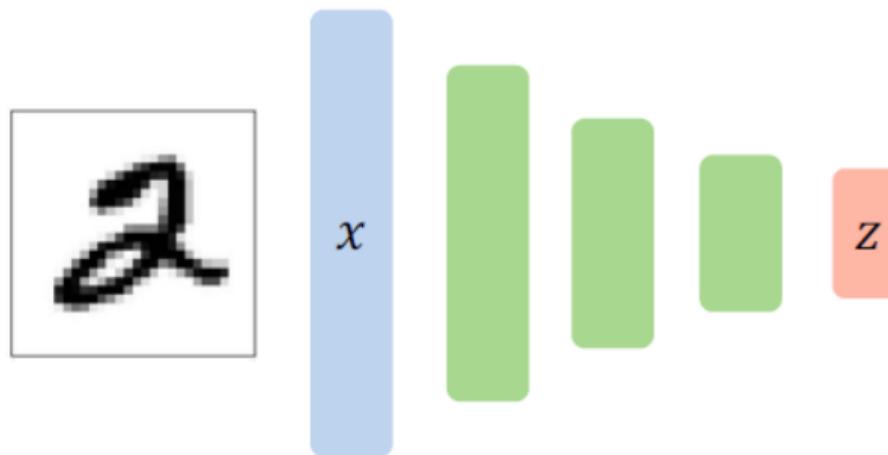
**CycleGAN** aprende las transformaciones a través de los dominios con datos no emparejados.



# Autoencoders

## Autoencoders: background

Enfoque no supervisado para aprender una representación de características de menor dimensión a partir de datos de entrenamiento no etiquetados

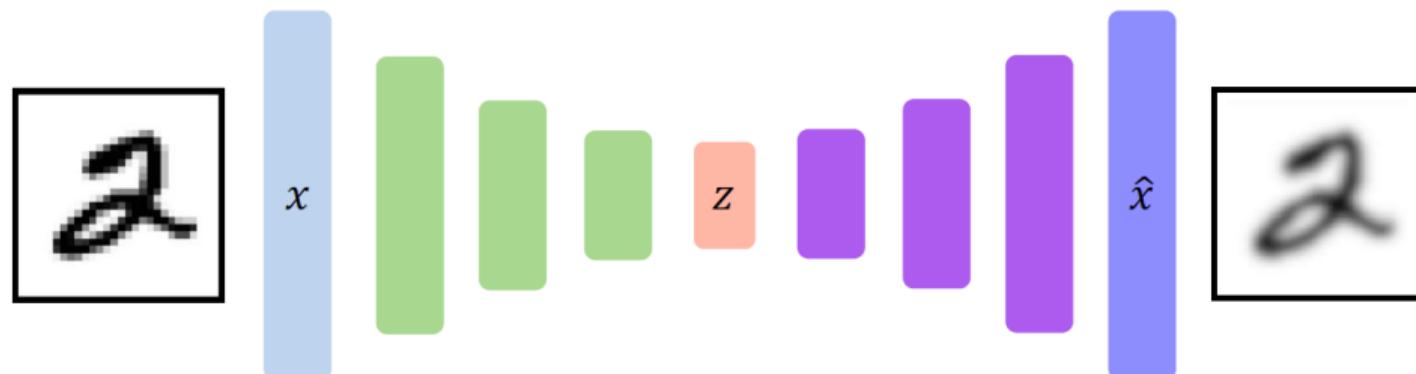


El "Codificador" (*encoder*) aprende el mapeo de los datos,  $x$ , a un espacio latente de baja dimensión,  $z$

## Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**

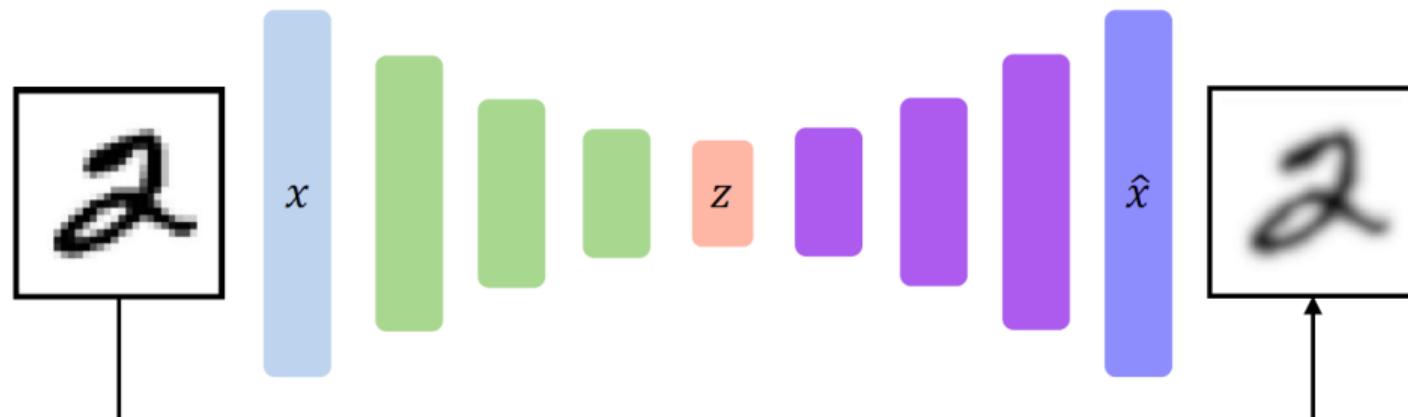


El "Decodificador" (*decoder*) aprende a mapear desde la latente,  $z$ , hasta una observación reconstruida,  $\hat{x}$

## Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**



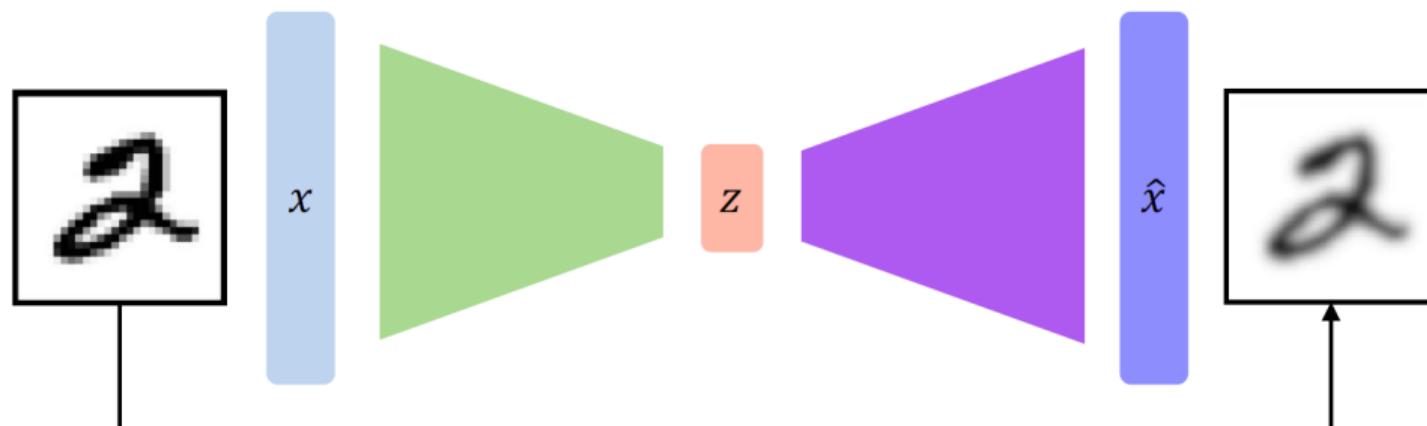
$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

La función de pérdida no usa  
ninguna etiqueta!!

## Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

La función de pérdida no usa  
ninguna etiqueta!!

# La dimensionalidad del espacio latente → calidad de la reconstrucción

- ¡La autocodificación es una forma de compresión!
- Un espacio latente más pequeño forzará un mayor cuello de botella de entrenamiento

2D latent space



5D latent space



Ground Truth

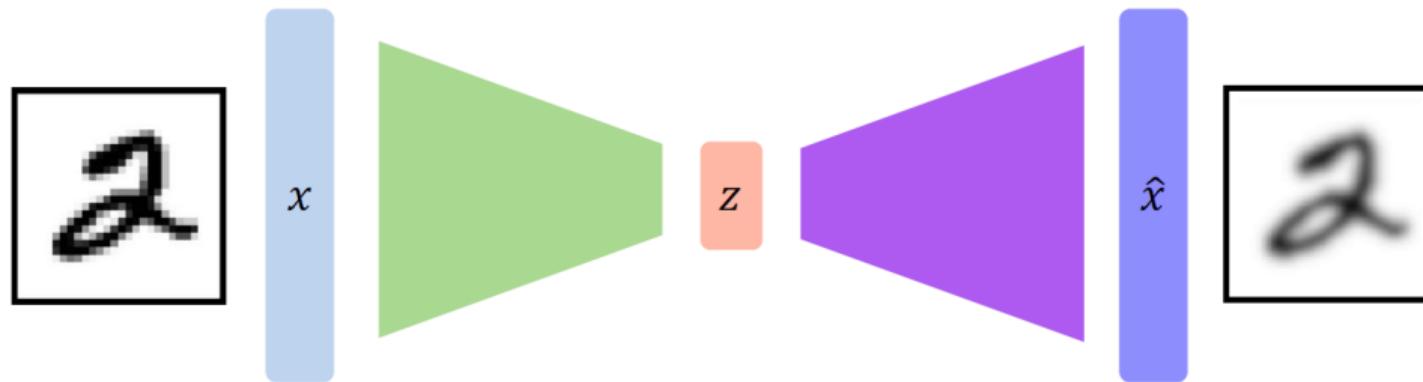


# Autocodificadores para el aprendizaje por representación

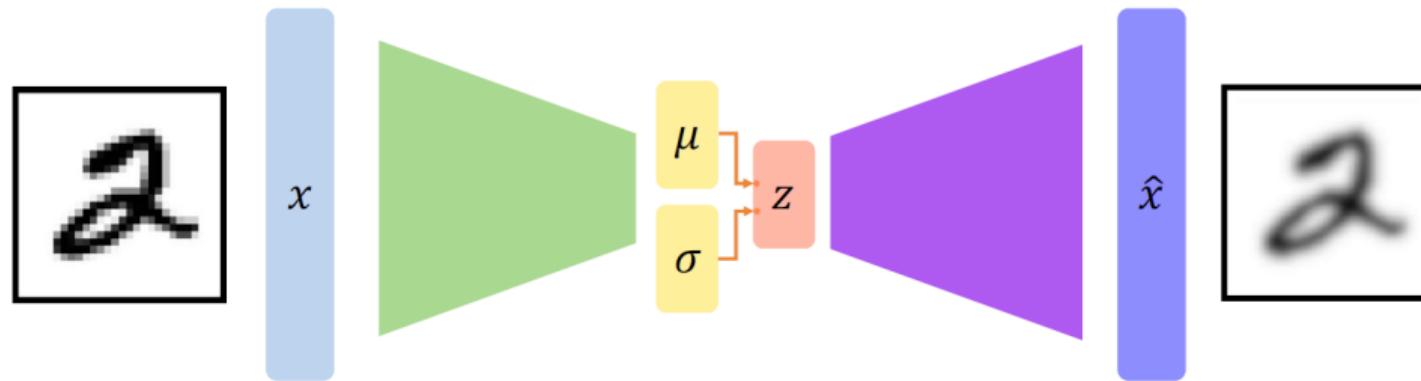
- La **capa oculta cuello de botella** obliga a la red a aprender una comprimida representación latente
- La **pérdida por reconstrucción** obliga a la representación latente a capturar (*o codificar*) tanta "información" sobre los datos como sea posible
- Autocodificación = **Codificación automática** de datos

# Variational Autoencoders (VAEs)

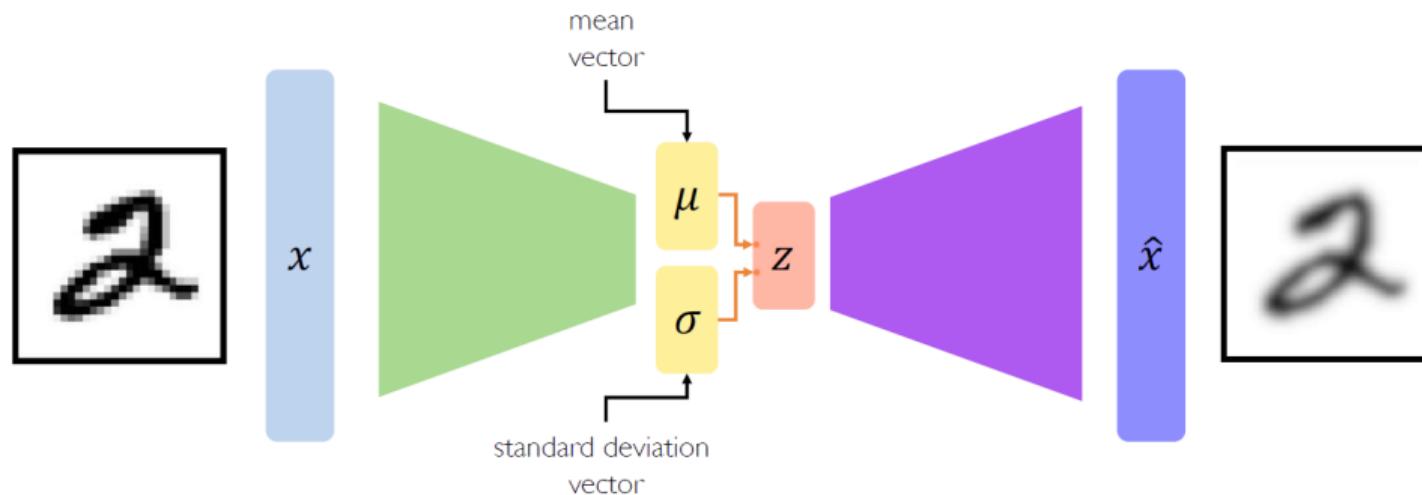
# VAEs: diferencia clave con el autoencoder tradicional



# VAEs: diferencia clave con el autoencoder tradicional

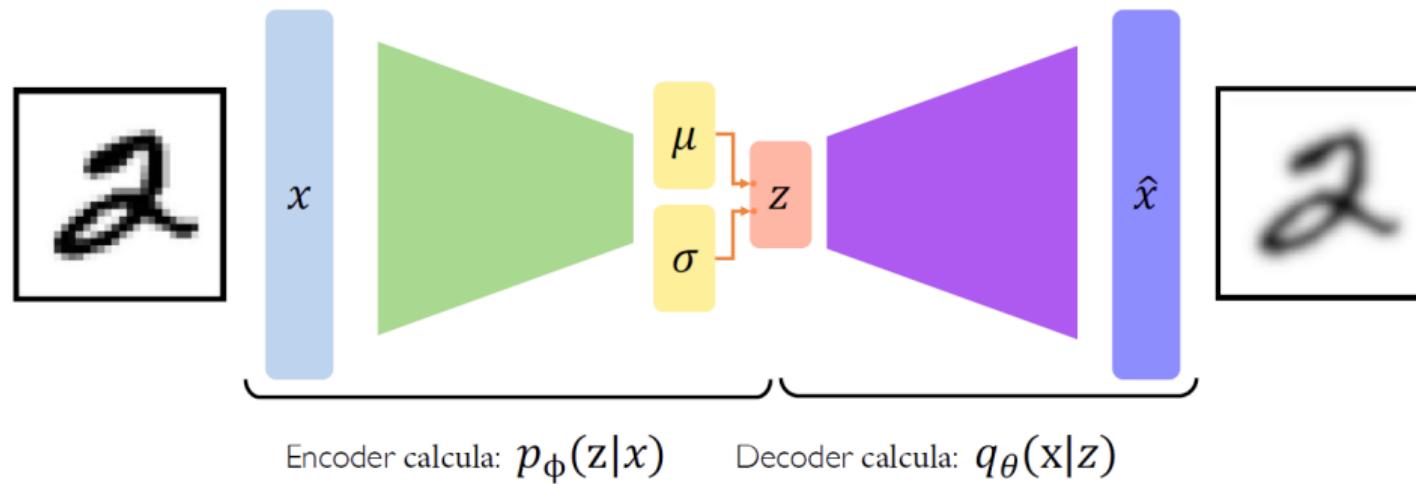


# VAEs: diferencia clave con el autoencoder tradicional

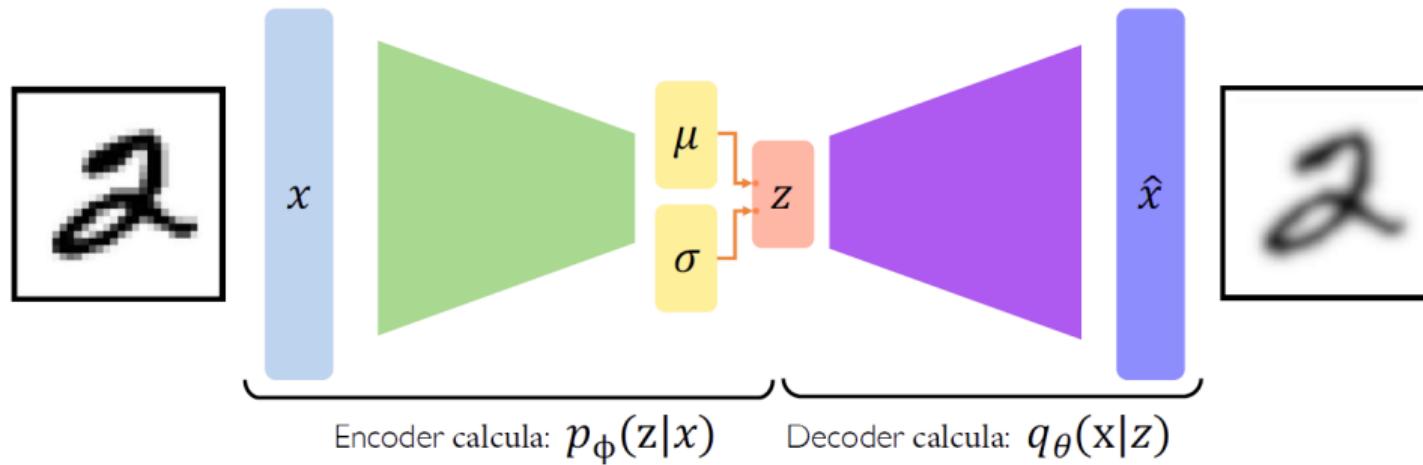


- ¡Los autoencoders variables son un giro probabilístico de los autocodificadores!
- Muestra de la media y la desviación estándar para calcular la muestra latente

# Optimización de la VAE

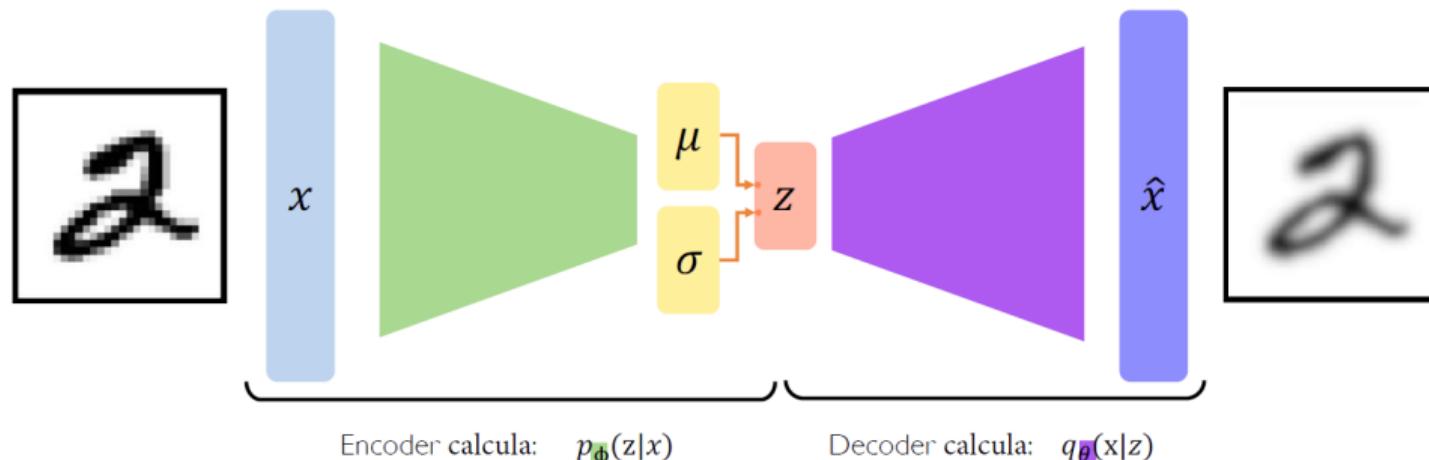


# Optimización de la VAE



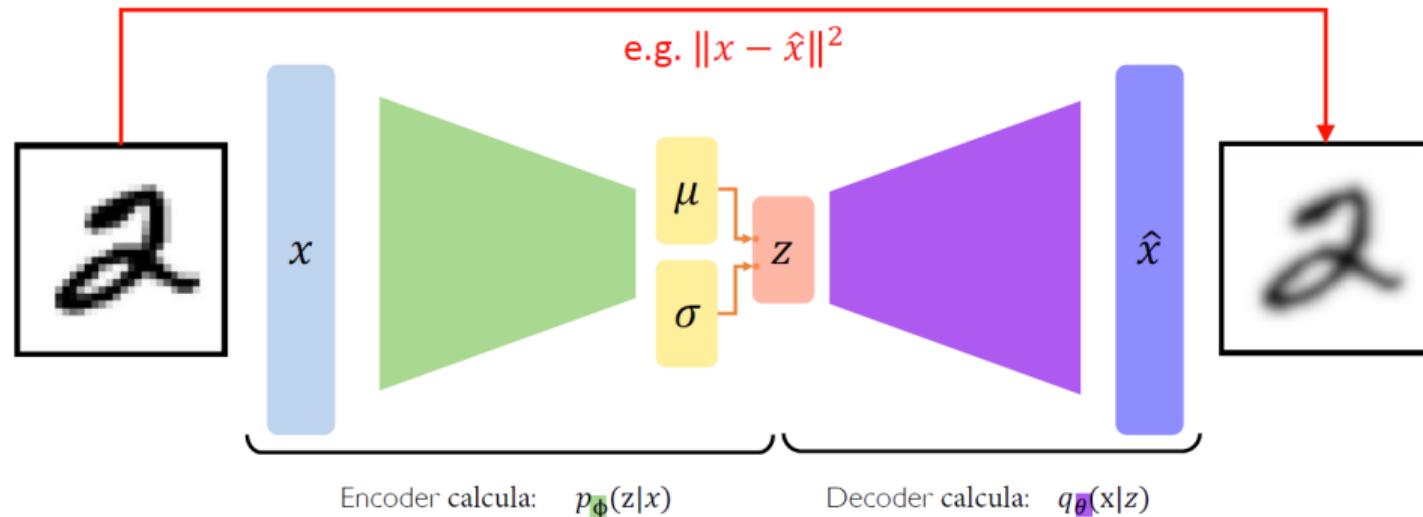
$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

# Optimización de la VAE



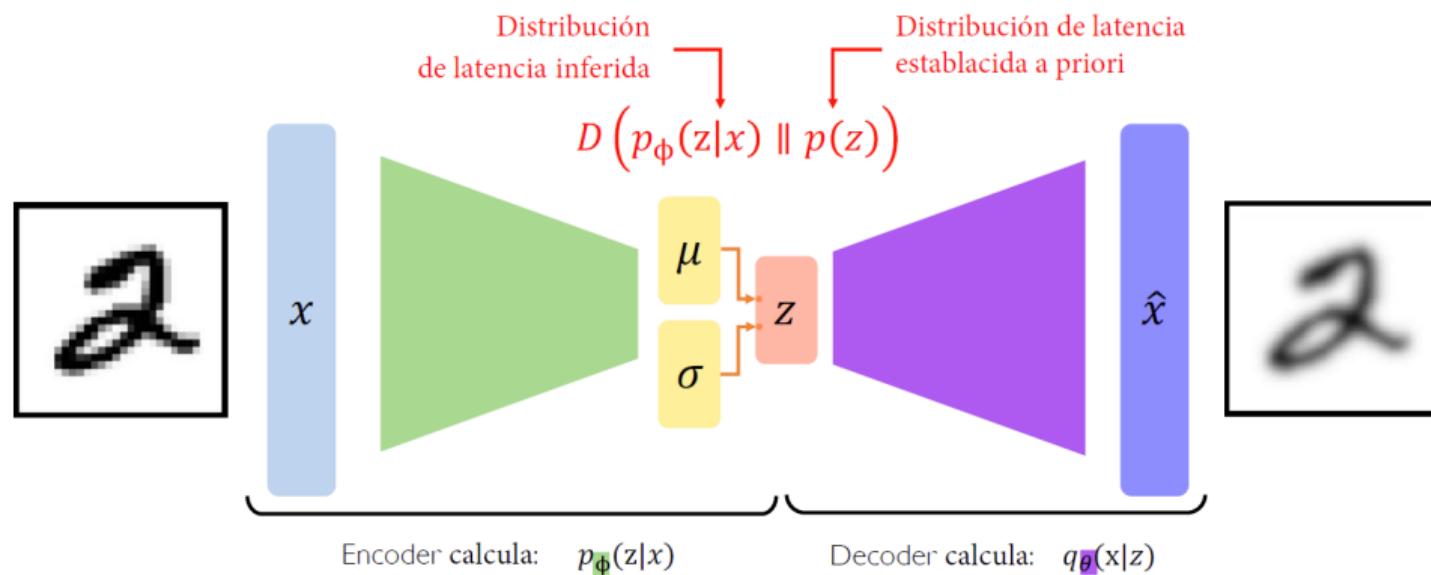
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

# Optimización de la VAE



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(termino de regularización)}$$

# Optimización de la VAE



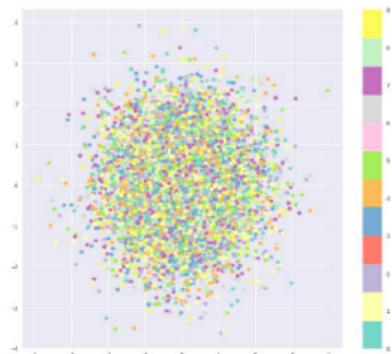
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(termino de regularización)}$$

# Los *a priors* de la distribución latente

$$D(p_{\phi}(z|x) \parallel p(z))$$

↑                              ↑

Distribución de latencia inferida      Distribución de latencia establecida a priori



## La elección común del prior:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

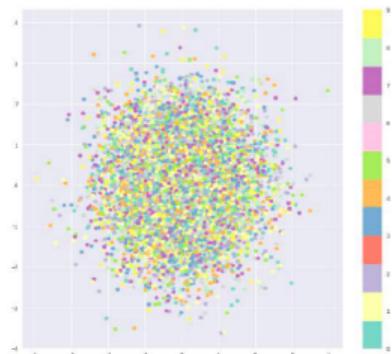
- Ajena a las codificaciones a distribuirlas uniformemente alrededor del centro del espacio latente
- Penaliza a la red cuando intente "engaños" agrupando puntos en regiones específicas (es decir, memorizando los datos)

# Los *a priors* de la distribución latente

$$\begin{aligned} D(p_{\phi}(z|x) \parallel p(z)) \\ = -\frac{1}{2} \sum_{j=0}^{k-1} (\sigma_j + \mu_j^2 - 1 - \log \sigma_j) \end{aligned}$$

KL-divergencia entre las dos distribuciones

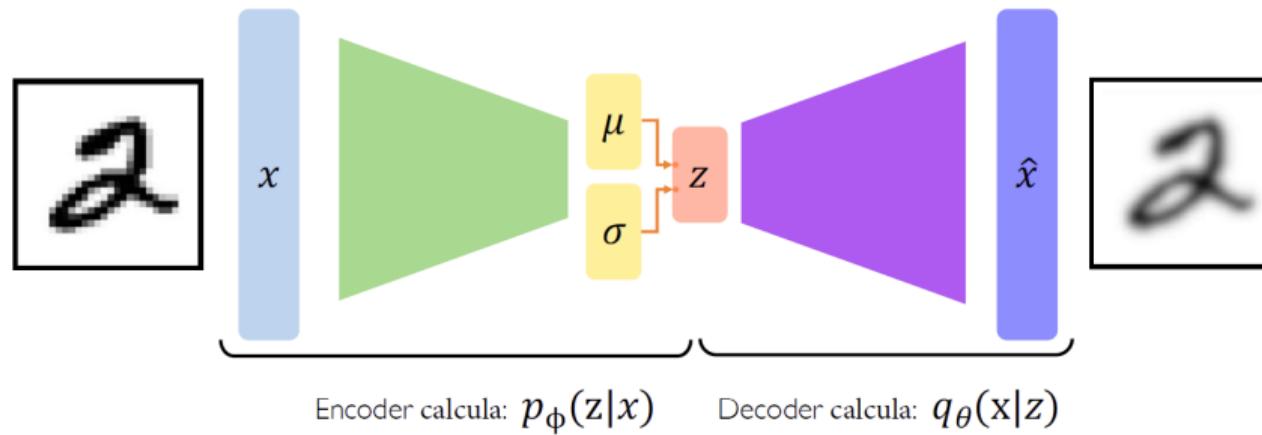
La elección común del prior:



$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Alienta a las codificaciones a distribuirse uniformemente alrededor del centro del espacio latente
- Penaliza a la red cuando intente "engañosamente" agrupar puntos en regiones específicas (es decir, memorizando los datos)

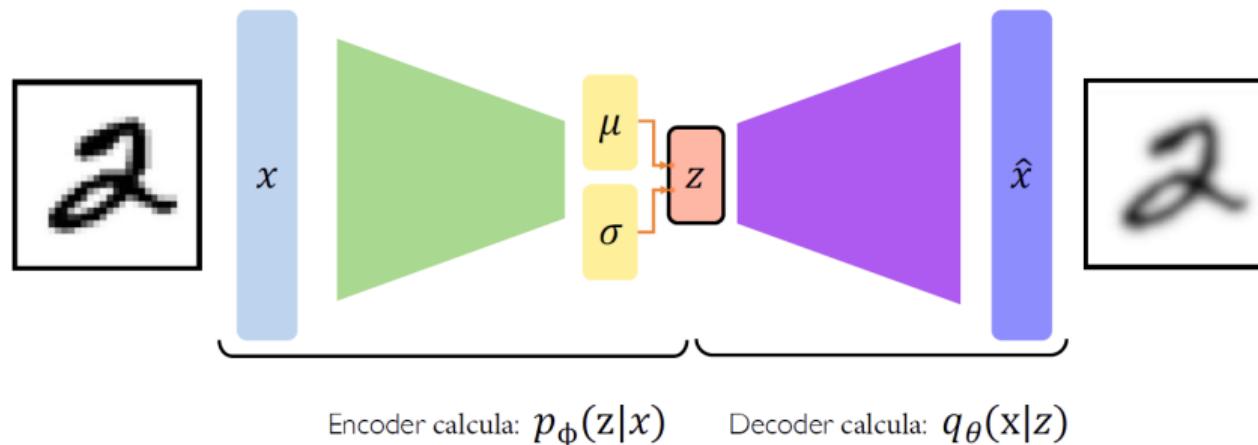
# Gráfo de cálculo de la VAE



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

# Gráfo de cálculo de la VAE

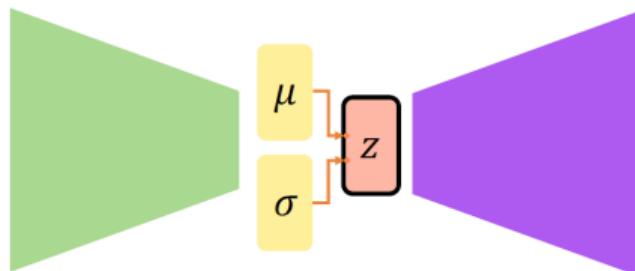
**Problema:** No podemos retropropagar los gradientes a través de las capas de muestreo



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

# Reparametrizando la capa de muestreo

Idea clave:



$$z \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

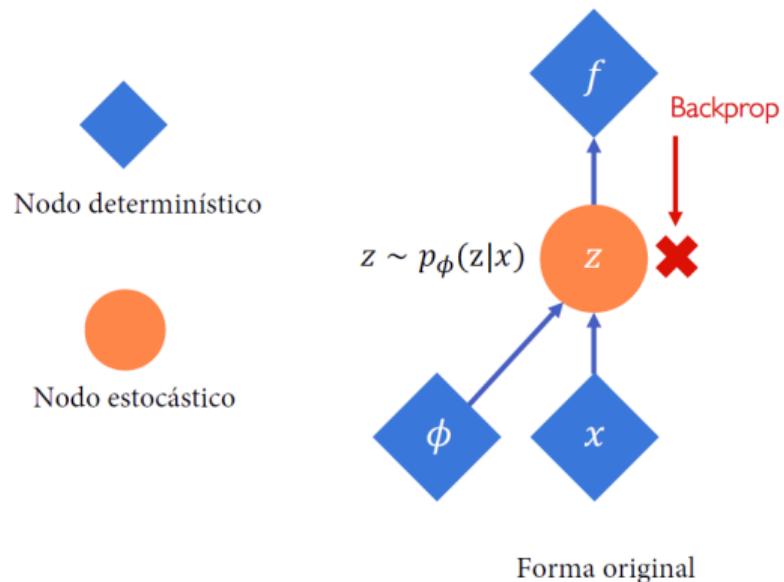
Considerar el vector latente muestreado como una suma de

- un vector  $\mu$  fijo
- y un vector  $\sigma$  fijo, escalado por constantes aleatorias extraídas de la distribución *a prior*

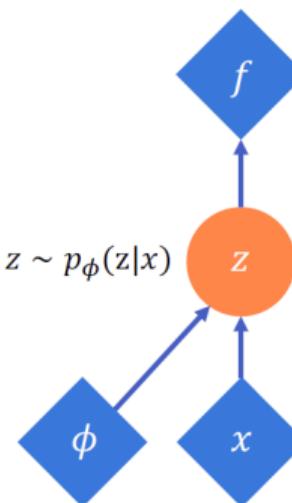
$$z = \mu + \sigma \odot \epsilon$$

donde  $\epsilon \sim \mathcal{N}(0, 1)$

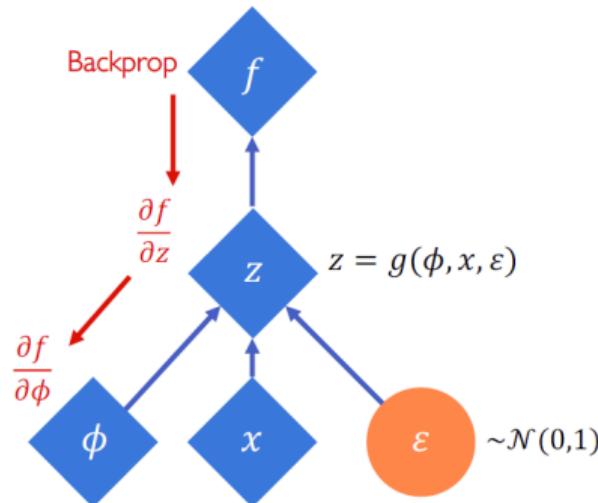
# Reparametrizando la capa de muestreo



# Reparametrizando la capa de muestreo



Forma original



Forma reparametrizada

## VAEs: Perturbación latente

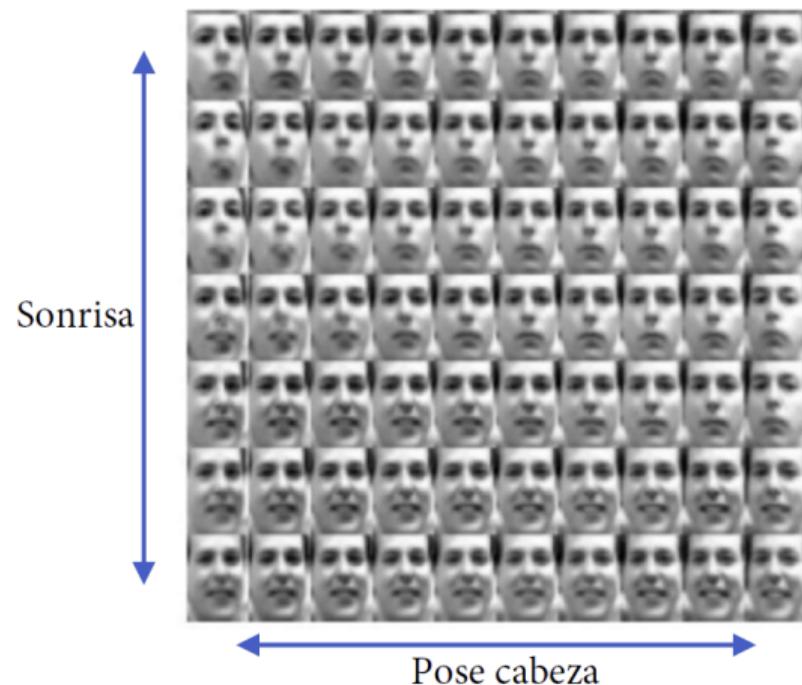
- Aumentar o disminuir lentamente una **sola variable latente**
- Mantener todas las demás variables fijas



Pose de la cabeza

Diferentes dimensiones de  $z$  codifica **diferentes características latentes interpretables**

# VAEs: Perturbación latente



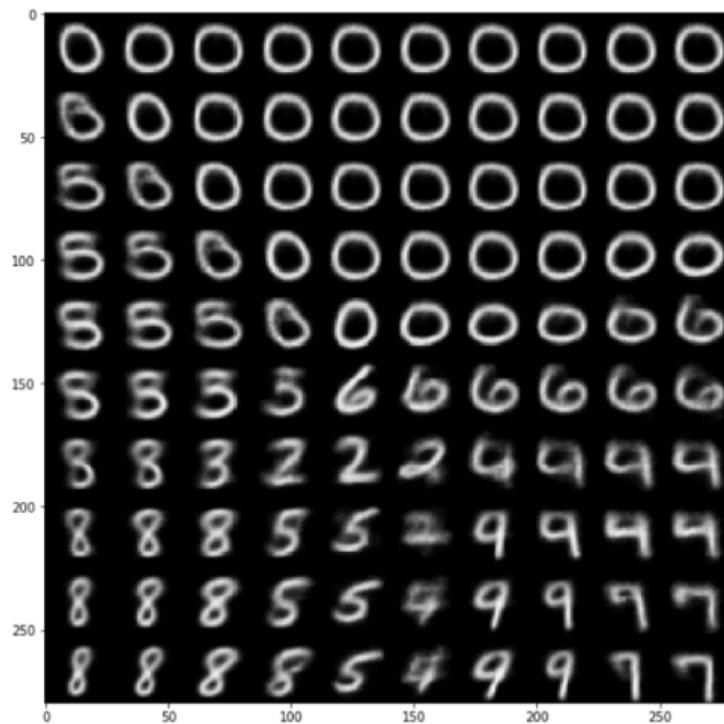
- Idealmente, queremos variables latentes que no estén correlacionadas entre sí
- Aplicar el priorato diagonal a las variables latentes para fomentar independencia
- **Desenredo** (*Disentanglement*)

# VAEs: Perturbación latente



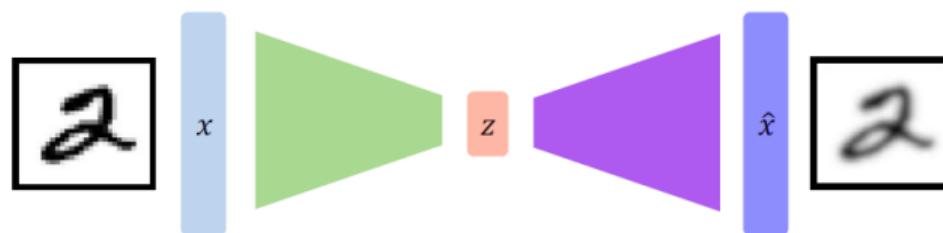
Google BeatBlender

## VAEs: Perturbación latente



# Resumen de las VAEs

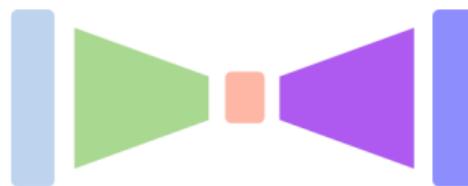
- 1 Comprimir la representación del mundo a algo que podamos usar para aprender
- 2 La reconstrucción permite un aprendizaje no supervisado (¡sin etiquetas!)
- 3 El truco de la reparameterización para entrenar de extremo a extremo
- 4 Interpretar las variables latentes ocultas utilizando la perturbación
- 5 Generar nuevos ejemplos



# Modelado Generativo Profundo: Resumen

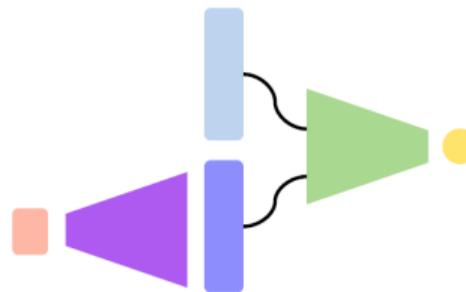
## Autoencoders and Variational Autoencoders (VAEs)

- Aprende el espacio latente de **menor dimensión** y toma **muestras** para generar reconstrucciones de entrada



## Generative Adversarial Networks (GANs)

- Redes de generadores y discriminadores en competencia



# Muchas gracias por su atención

*¿Preguntas?*

**Contact:** Marco Tulio Teran De La Hoz  
**e-mail:** marco.teran@usa.edu.co



UNIVERSIDAD  
SERGIO ARBOLEDA