# Visión Computacional

Ivan Sipiran

# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc

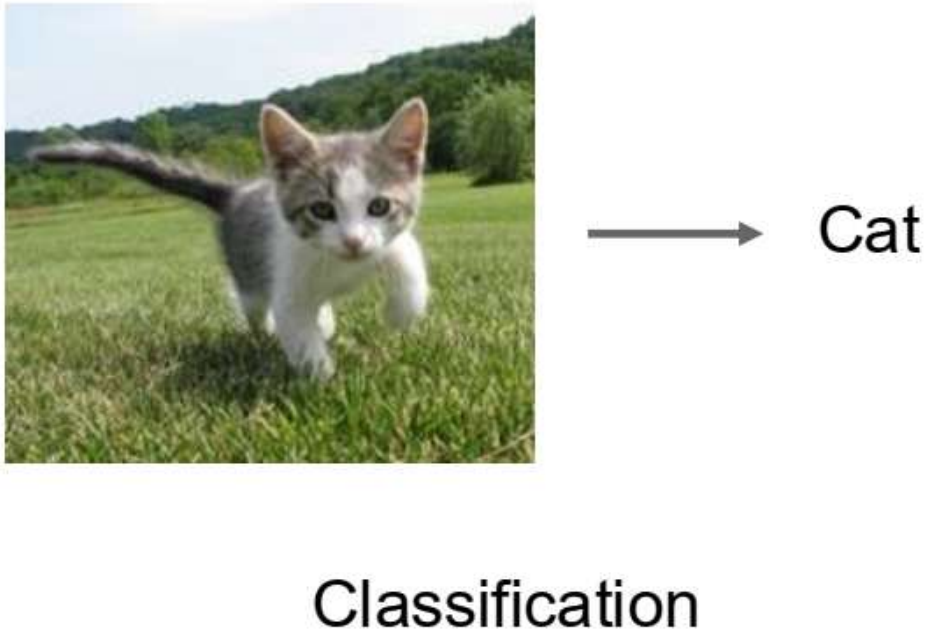
# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc



# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc

A cat sitting on a suitcase on the floor

Image captioning

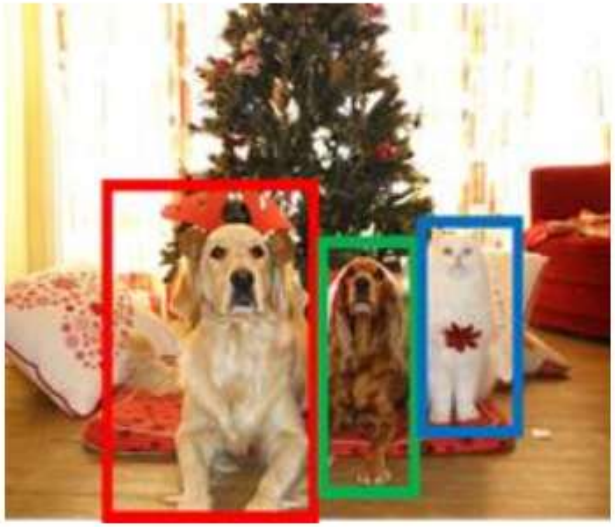
# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc



DOG, DOG, CAT

Object Detection

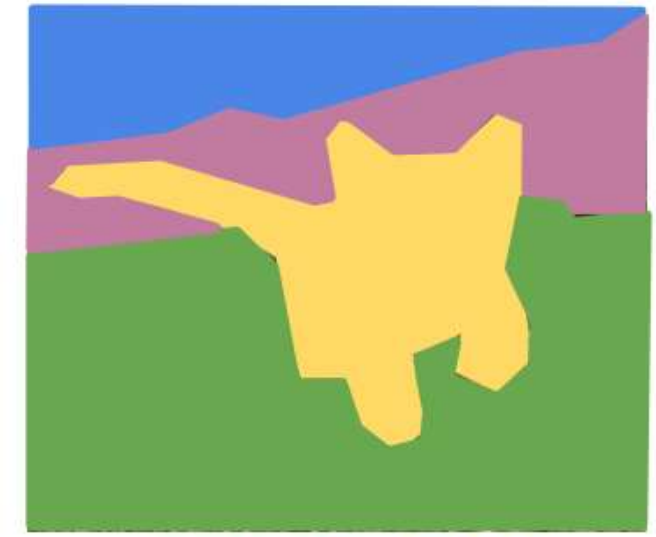
# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc




Semantic Segmentation

# Aprendizaje supervisado vs no supervisado

**Aprendizaje supervisado**

**Data**: (x, y)
X es dato, y es etiqueta

**Meta**: Aprender una función que mapee x→y

**Ejemplos**: clasificación, regresión, detección de objetos, segmentación semántica, captioning de imágenes, etc

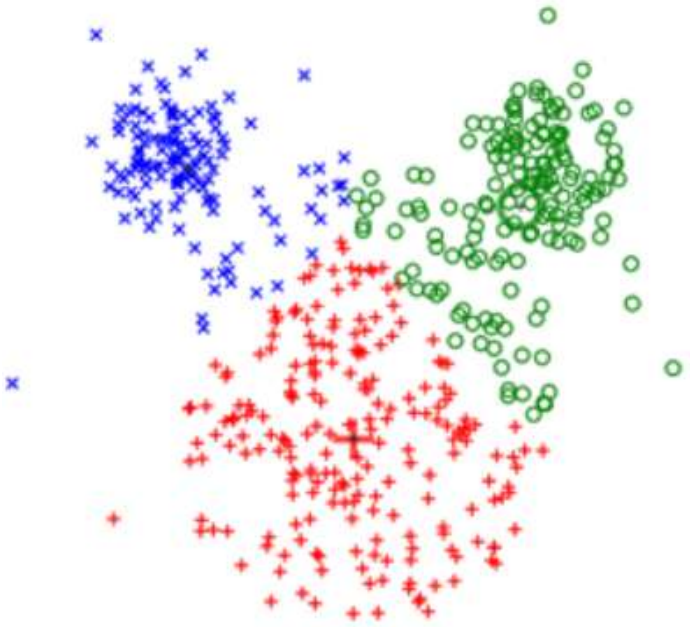
**Aprendizaje no supervisado**

**Data**: x
Solo datos, no etiquetas

**Meta**: Aprender la estructura interna de la data

**Ejemplos**: clustering, reducción de dimensionalidad, estimación de densidad, etc.

# Aprendizaje supervisado vs no supervisado



K-means clustering

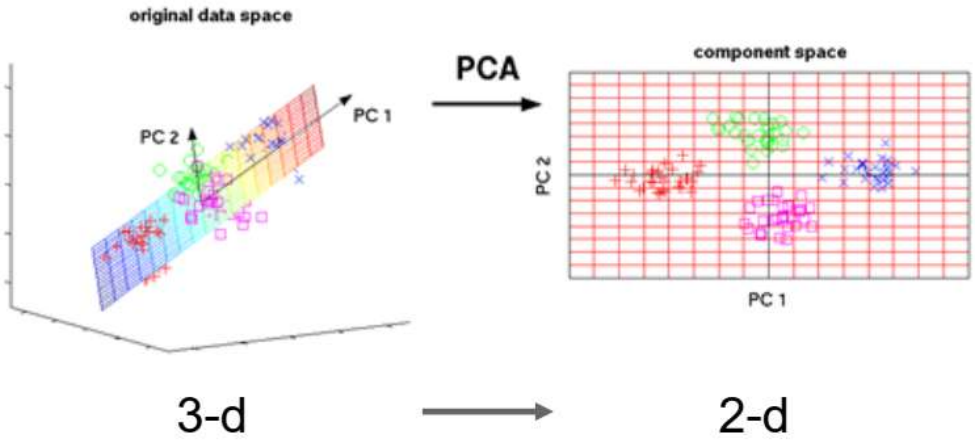
**Aprendizaje no supervisado**

**Data**: x
Solo datos, no etiquetas

**Meta**: Aprender la estructura interna de la data

**Ejemplos**: clustering, reducción de dimensionalidad, estimación de densidad, etc.

# Aprendizaje supervisado vs no supervisado




Principal Component Analysis
(Dimensionality reduction)

**Aprendizaje no supervisado**

**Data**: x
Solo datos, no etiquetas

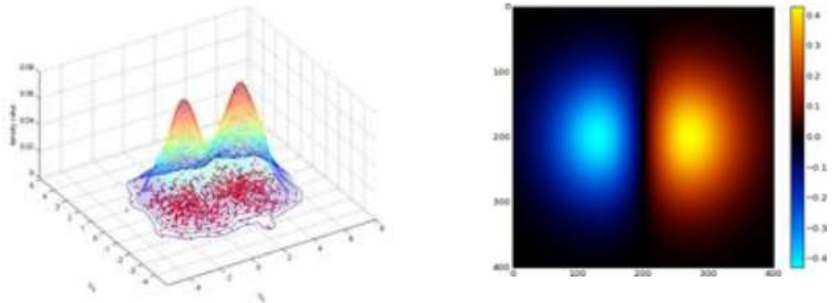
**Meta**: Aprender la estructura interna de la data

**Ejemplos**: clustering, reducción de dimensionalidad, estimación de densidad, etc.

# Aprendizaje supervisado vs no supervisado



1-d density estimation



2-d density estimation

Modeling p(x)

**Aprendizaje no supervisado**

**Data**: x
Solo datos, no etiquetas

**Meta**: Aprender la estructura interna de la data

**Ejemplos**: clustering, reducción de dimensionalidad, estimación de densidad, etc.

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

**Data:** x

**Label:** y

Cat

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

**Data:** x

**Label:** y

Cat

**Probabilidades**

**Función de densidad**

$p(x)$ asigna un número positivo a cada posible $x$; números más altos significan que $x$ es más probable.

Funciones de densidad son **normalizadas**

$$\int_X p(x)dx = 1$$

Diferentes valores de $x$ **compiten** por densidad

# Modelos generativos vs discriminativos

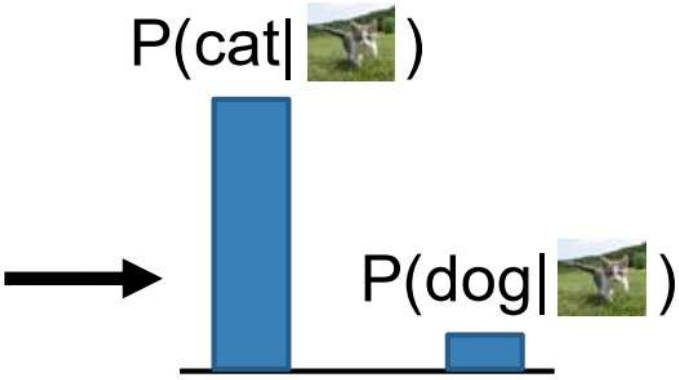
**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

**Data:** x

**Label:** y

Cat



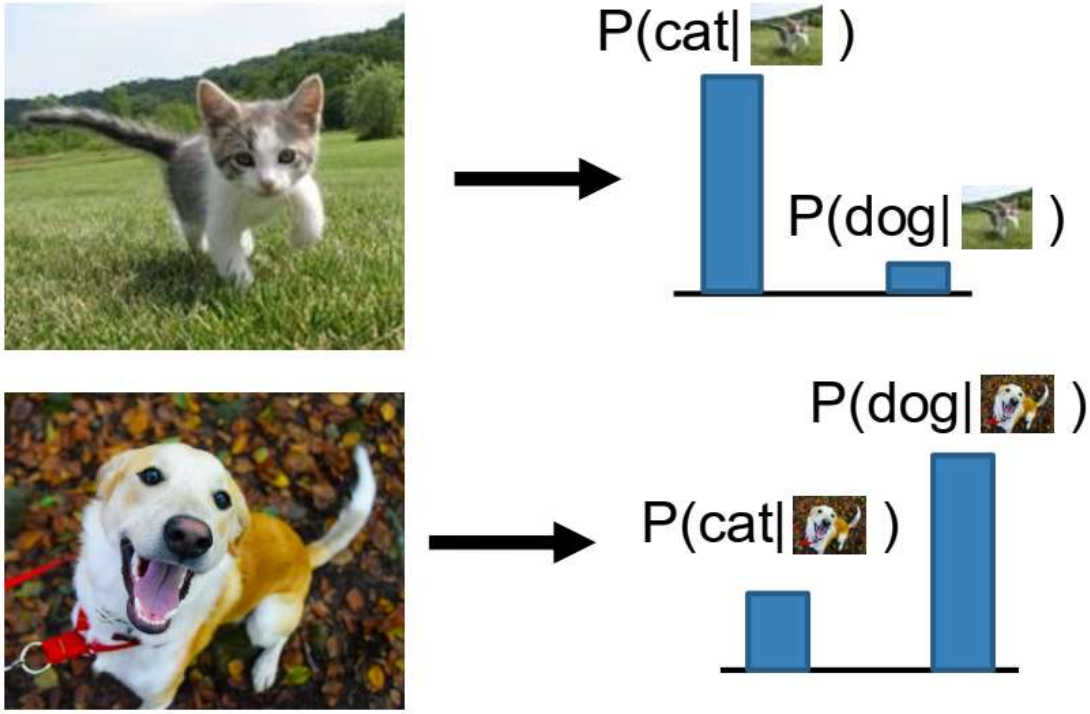

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

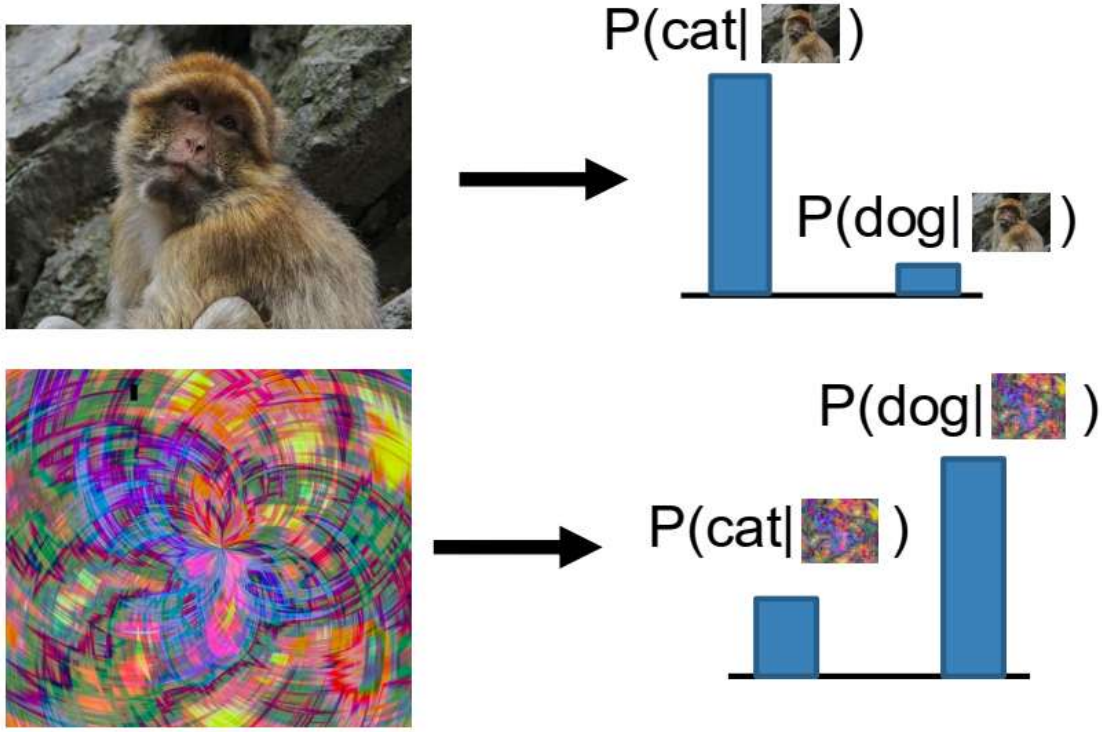



Posibles etiquetas para cada imagen compiten por probabilidad. No existe competencia entre imágenes

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

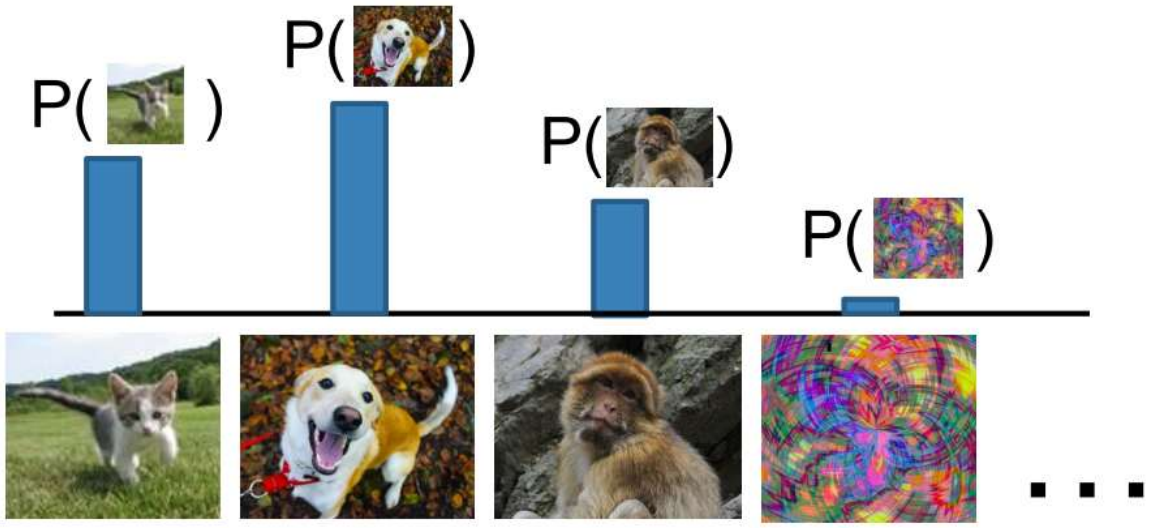



No hay forma de manejar inputs no razonables; deben entregar una distribución de etiquetas para cualquier input

# Modelos generativos vs discriminativos

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

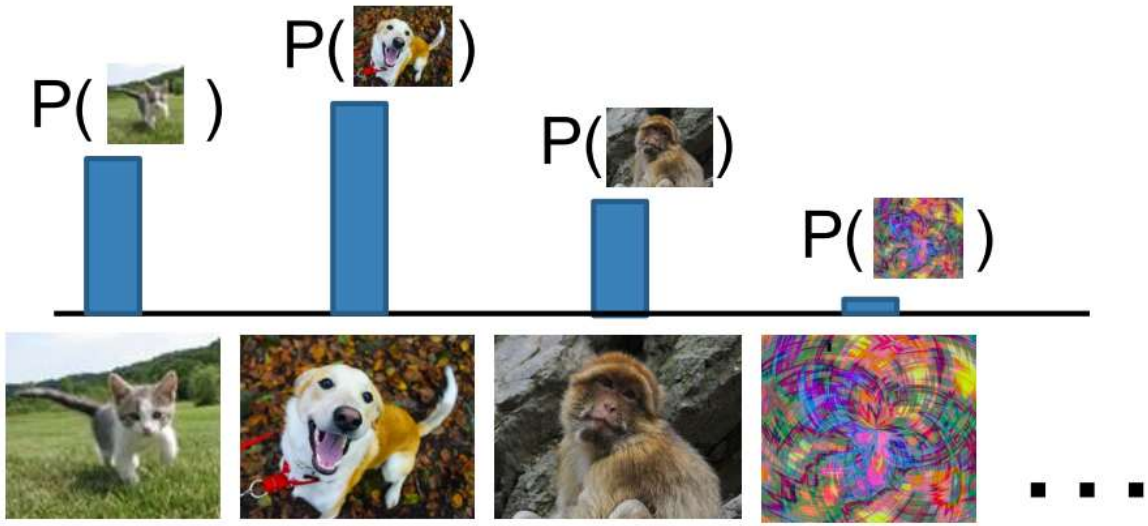



Todas las posibles imágenes compiten por masa de probabilidad

# Modelos generativos vs discriminativos

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$



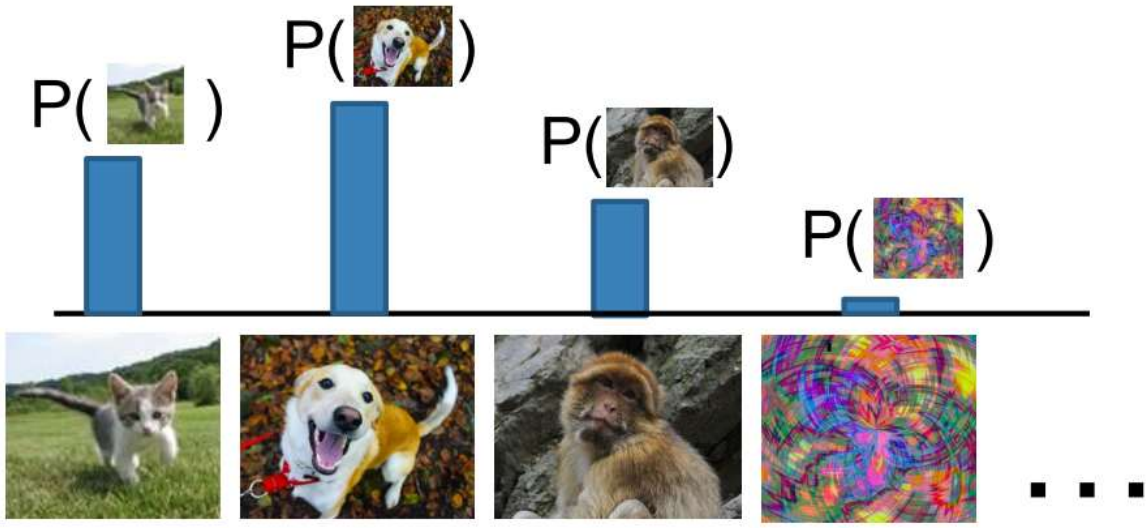

Todas las posibles imágenes compiten por masa de probabilidad

Requiere una comprensión profunda: es más probable que un perro esté sentado o parado?

# Modelos generativos vs discriminativos

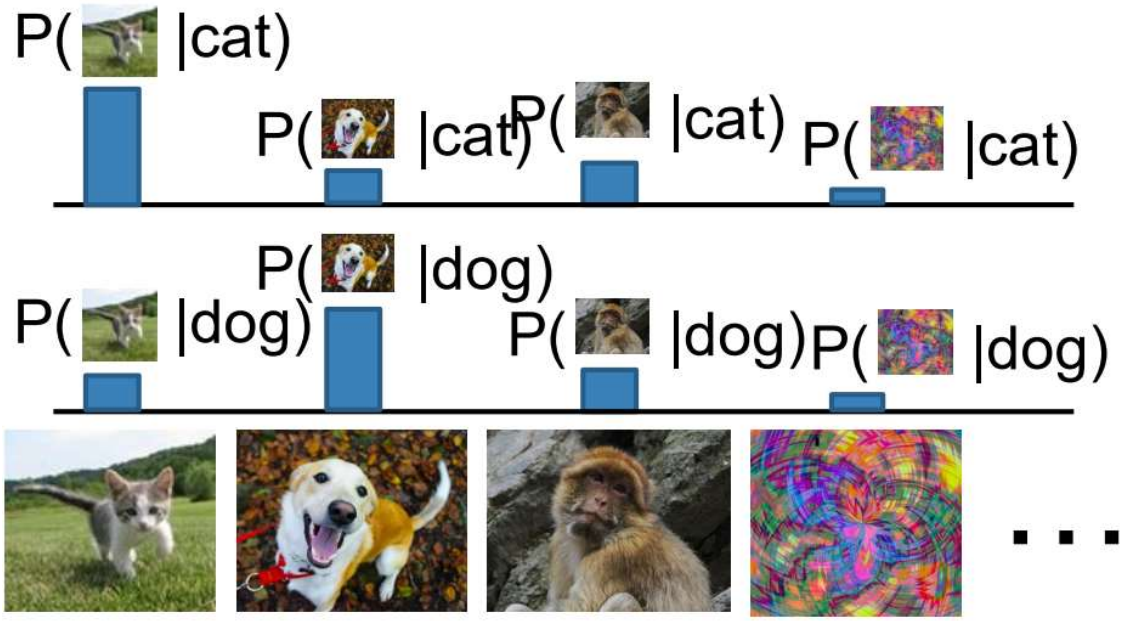
**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$




El modelo puede rechazar entradas no razonables dándoles masa de probabilidad pequeña

# Modelos generativos vs discriminativos




**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

Cada posible etiqueta induce una competición entre todas las posibles imágenes

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

Recordemos la regla de Bayes:

$$P(x \mid y) = \frac{P(y \mid x)}{P(y)} P(x)$$

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

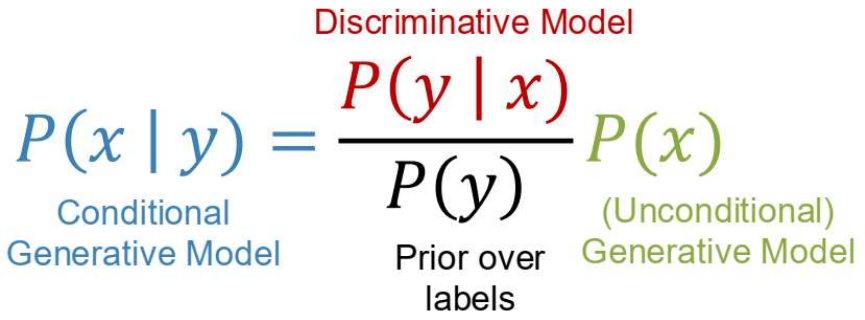
**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

Recordemos la regla de Bayes:




Podemos construir un modelo generativo condicional desde otros componentes….pero no es común en la práctica

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

Asignar etiquetas a datos
Feature learning (con etiquetas)

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

→ Asignar etiquetas a datos
Feature learning (con etiquetas)

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

→ Detectar outliers
Feature learning (sin etiquetas)
Sampling para generar nuevos datos

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

→ Asignar etiquetas a datos
Feature learning (con etiquetas)

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

→ Detectar outliers
Feature learning (sin etiquetas)
Sampling para generar nuevos datos

**Modelo generativo condicional**:

Aprender una distribución de probabilidad $p(x|y)$

→ Asignar labels mientras se rechazan Outliers
Sampling para generar datos desde labels

# Modelos generativos vs discriminativos

**Modelo discriminativo**:

Aprender una distribución de probabilidad $p(y|x)$

**Modelo generativo**:

Aprender una distribución de probabilidad $p(x)$

**Modelo generativo condicional**:

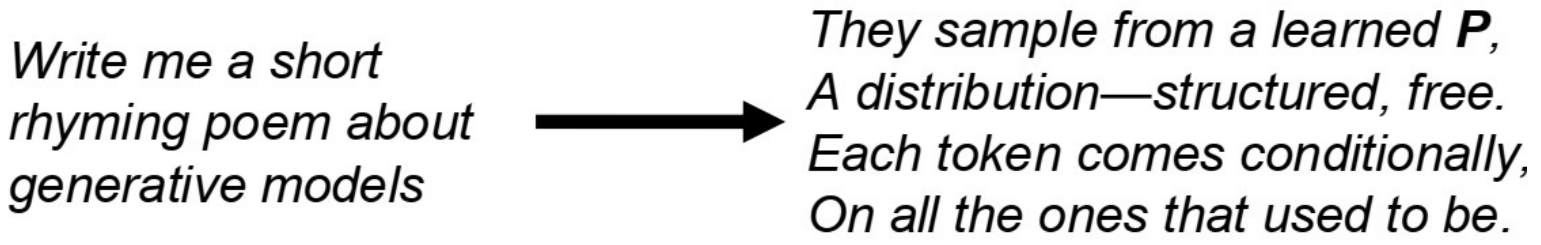
Aprender una distribución de probabilidad $p(x|y)$

**Modelos generativos** es cualquiera de estos dos; los modelos condicionales son más comunes en la práctica

# ¿Porqué modelos generativos?

**Ambigüedad:** Si existen muchas posibles salidas $x$ para una etiqueta $y$, queremos modelar $P(x|y)$

**Modelamiento de lenguaje:** Produce texto de salida $x$ desde texto de entrada $y$

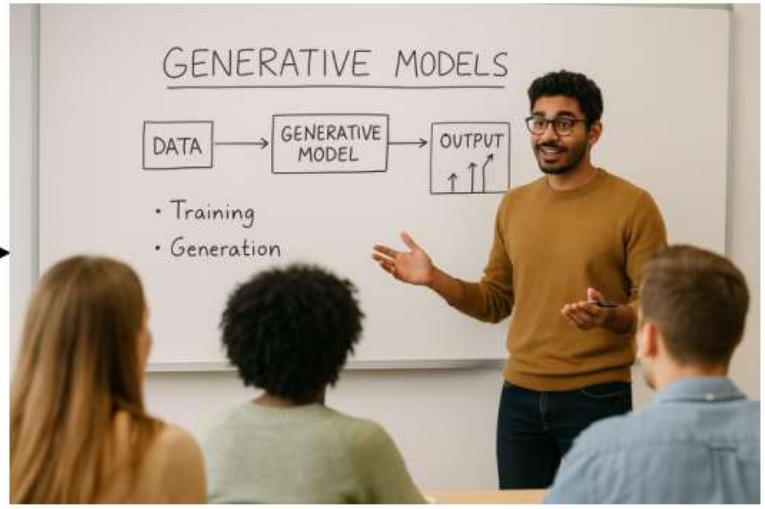


# ¿Porqué modelos generativos?

**Ambigüedad:** Si existen muchas posibles salidas $x$ para una etiqueta $y$, queremos modelar $P(x|y)$

**Texto a imagen:** Produce imagen de salida $x$ desde texto de entrada $y$

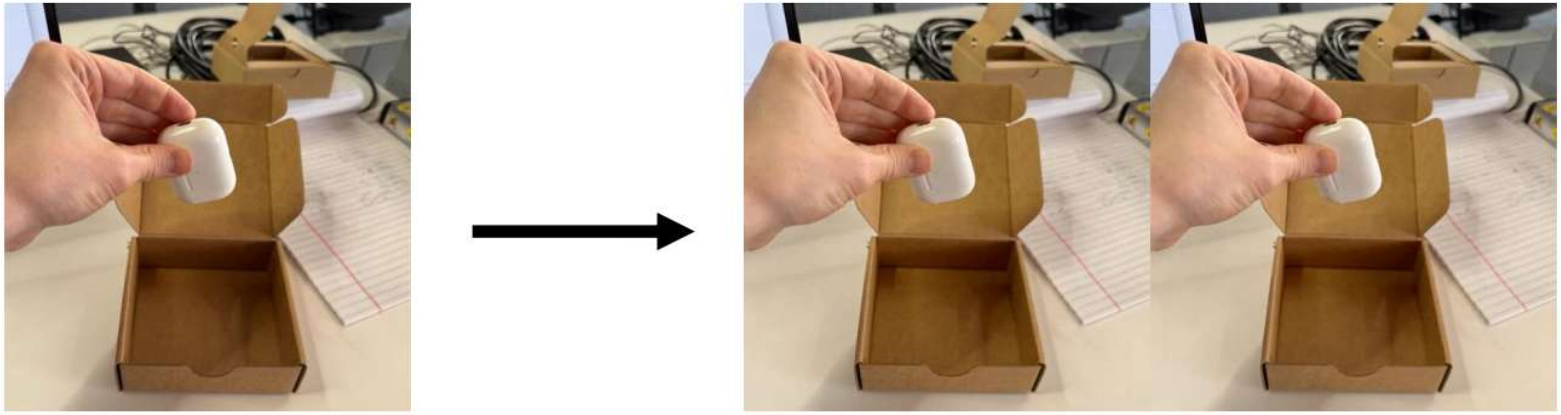
*Make me an image showing a person teaching a class on generative models in front of a whiteboard*



# ¿Porqué modelos generativos?

**Ambigüedad:** Si existen muchas posibles salidas $x$ para una etiqueta $y$, queremos modelar $P(x|y)$

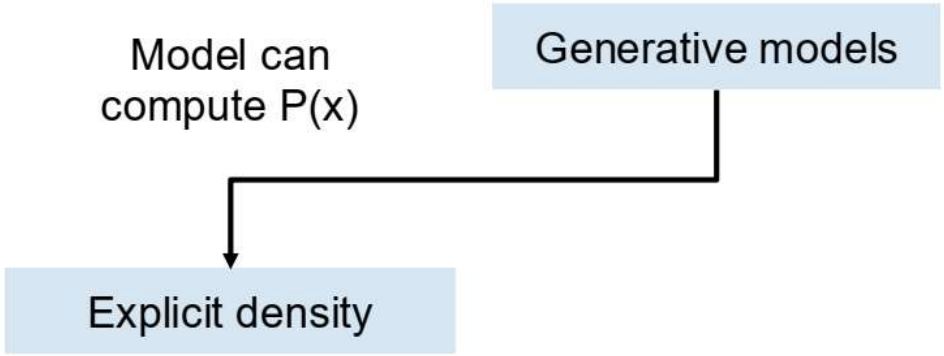
**Imagen a video:** Qué sigue a continuación?



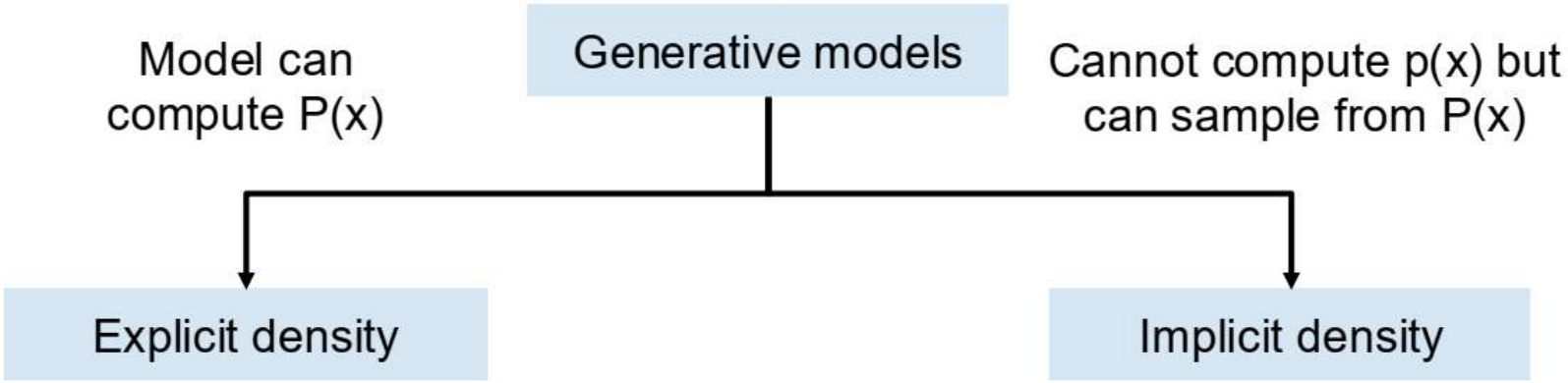
# Taxonomía

Generative models

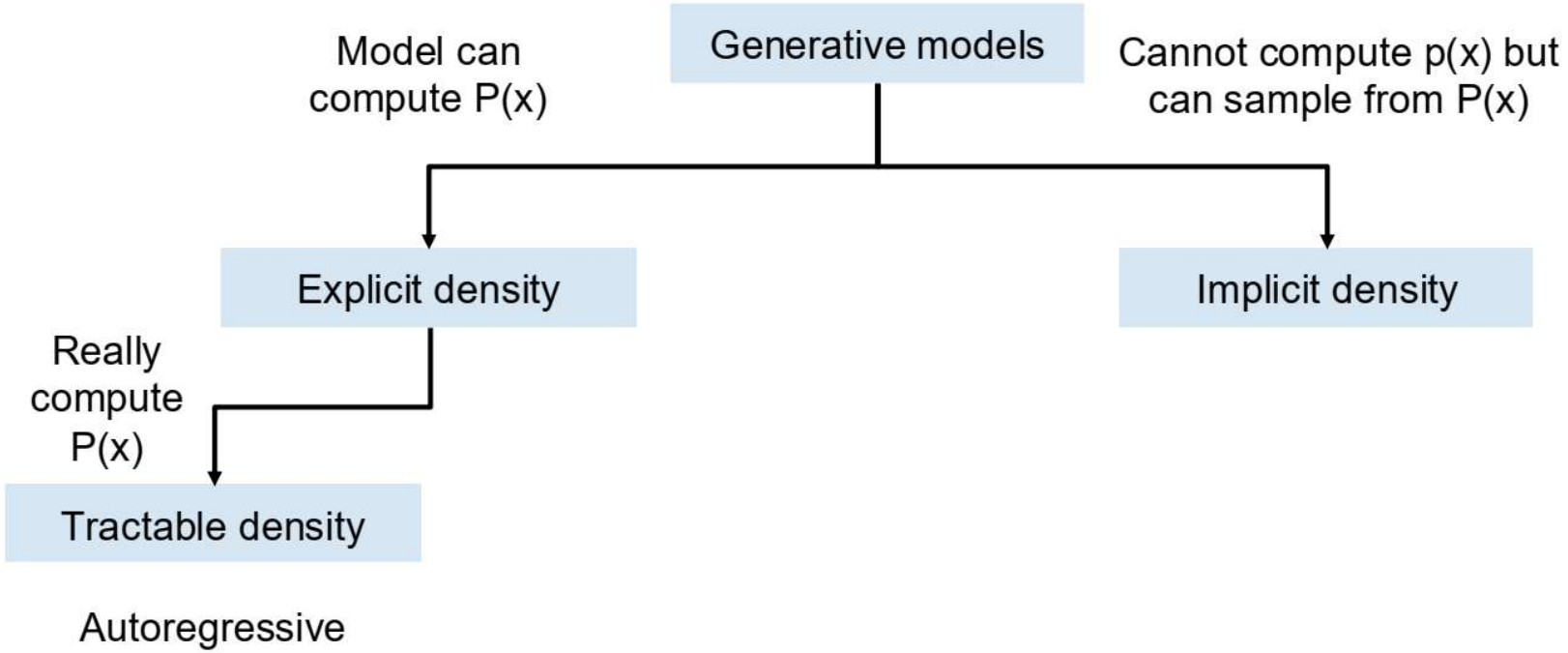
# Taxonomía



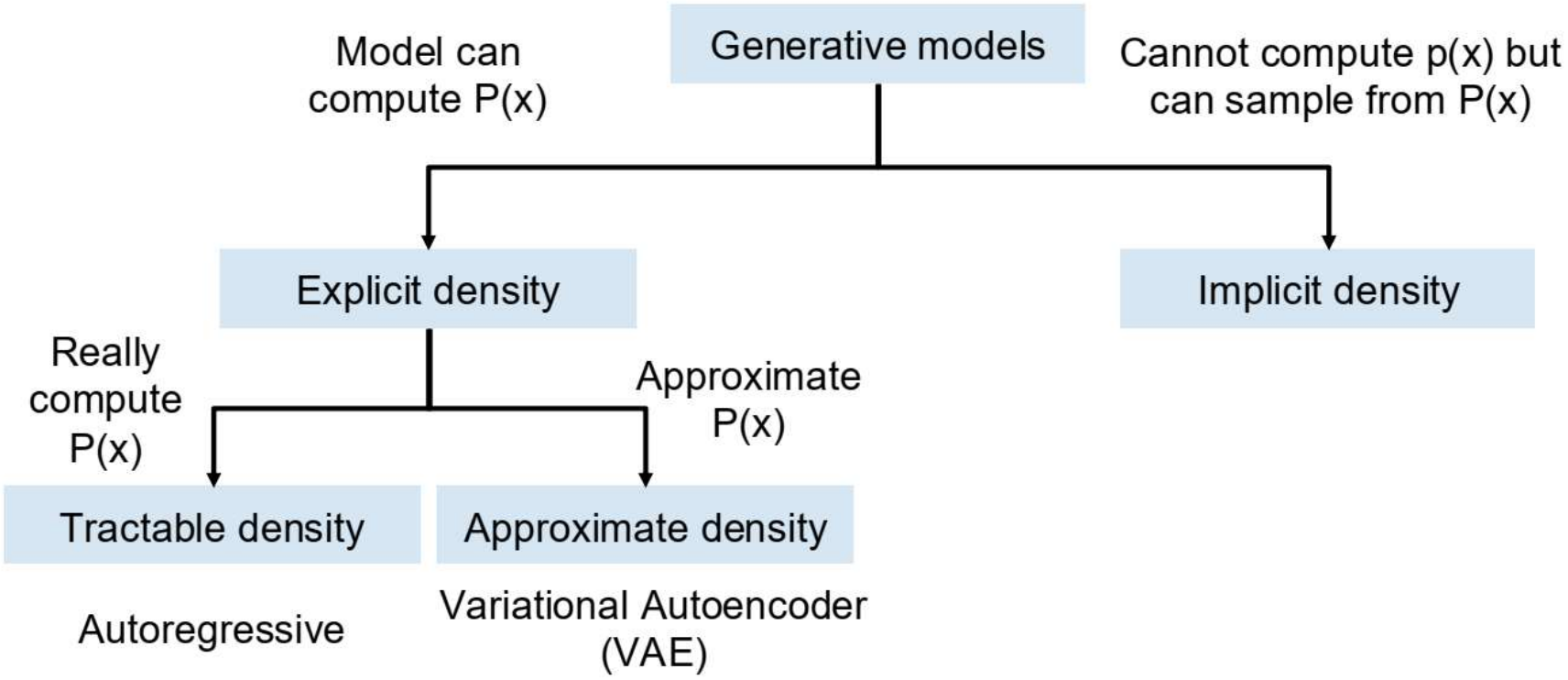
# Taxonomía


Generative models
Model can compute P(x)
Cannot compute p(x) but can sample from P(x)
Explicit density
Implicit density

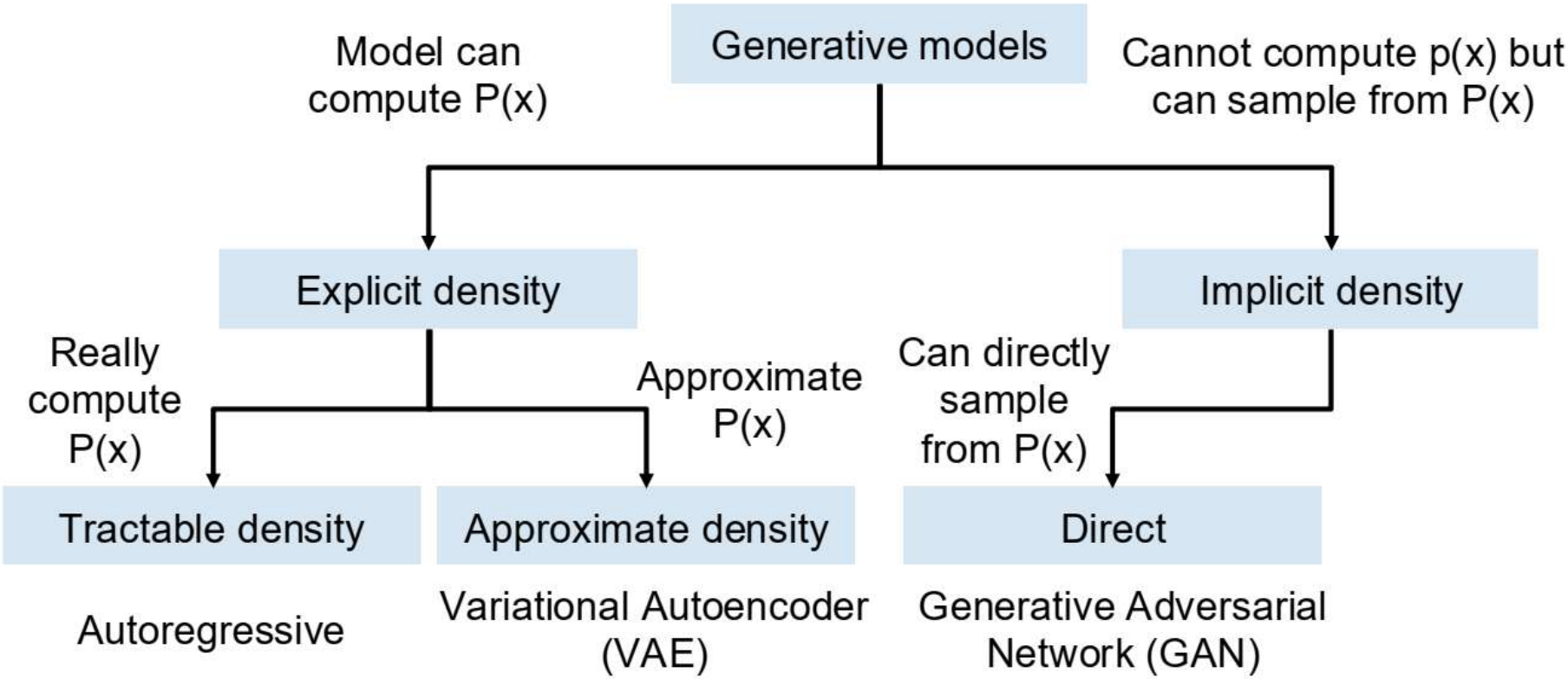
# Taxonomía


Generative models
Model can compute P(x)
Cannot compute p(x) but can sample from P(x)
Explicit density
Implicit density
Really compute P(x)
Tractable density
Autoregressive

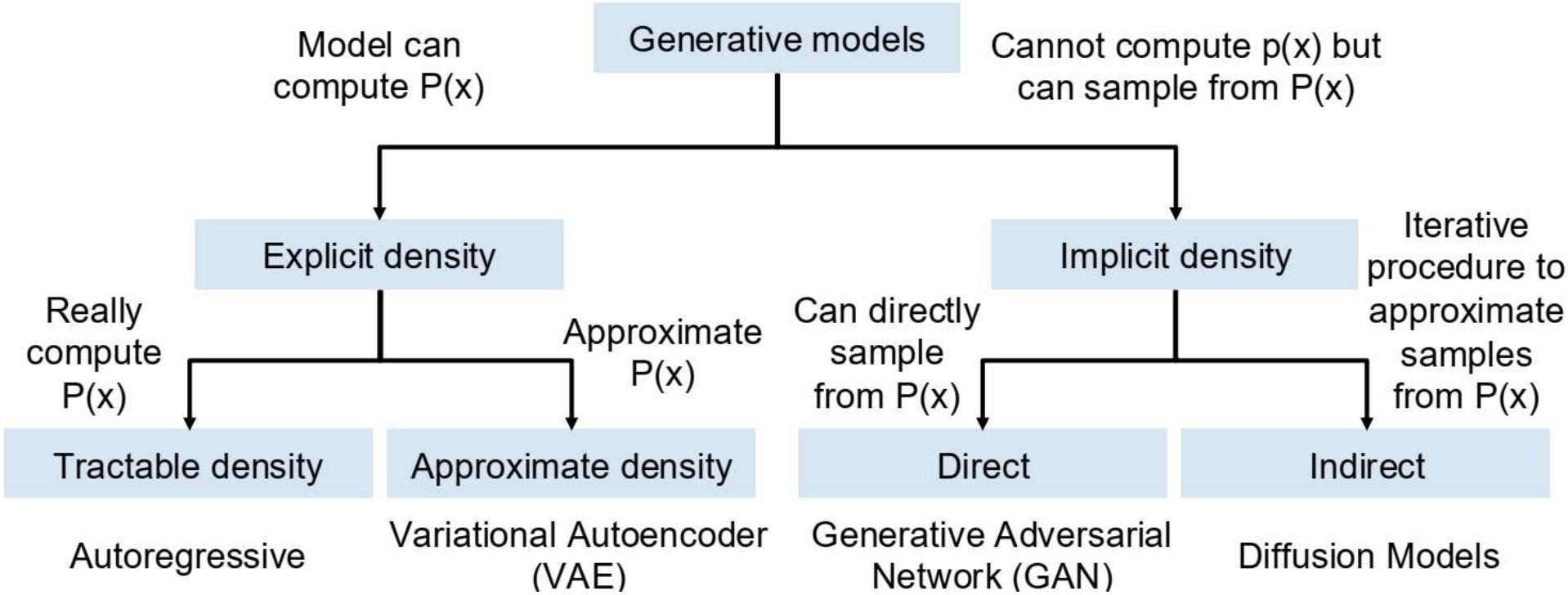
# Taxonomía


Generative models
Model can compute P(x)
Cannot compute p(x) but can sample from P(x)
Explicit density
Implicit density
Really compute P(x)
Approximate P(x)
Tractable density
Approximate density
Autoregressive
Variational Autoencoder (VAE)

# Taxonomía


Generative models
Model can compute P(x)
Cannot compute p(x) but can sample from P(x)
Explicit density
Implicit density
Really compute P(x)
Approximate P(x)
Can directly sample from P(x)
Tractable density
Approximate density
Direct
Autoregressive
Variational Autoencoder (VAE)
Generative Adversarial Network (GAN)

# Taxonomía


Generative models
Model can compute P(x)
Cannot compute p(x) but can sample from P(x)
Explicit density
Implicit density
Really compute P(x)
Approximate P(x)
Can directly sample from P(x)
Iterative procedure to approximate samples from P(x)
Tractable density
Approximate density
Direct
Indirect
Autoregressive
Variational Autoencoder (VAE)
Generative Adversarial Network (GAN)
Diffusion Models

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

Dado un dataset $x^{(1)}, x^{(2)}, \cdots, x^{(N)}$, entrenar el modelo para resolver

$$W^* = \arg\max_{\mathrm{W}} \prod_i p(x^{(i)})$$

Maximizar la probabilidad de entrenar datos
(Maximum likelihood estimation)

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

Dado un dataset $x^{(1)}, x^{(2)}, \cdots, x^{(N)}$, entrenar el modelo para resolver

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

$$= \arg\max_W \sum_i \log p(x^{(i)})$$

Maximizar la probabilidad de entrenar datos (Maximum likelihood estimation)

Truco del Log: Cambiar productos y sumas

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

Dado un dataset $x^{(1)}, x^{(2)}, \cdots, x^{(N)}$, entrenar el modelo para resolver

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

Maximizar la probabilidad de entrenar datos (Maximum likelihood estimation)

$$= \arg\max_W \sum_i \log p(x^{(i)})$$

Truco del Log: Cambiar productos y sumas

$$= \arg\max_W \sum_i \log f(x^{(i)}, W)$$

Función de Loss, optimizarla con SGD

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

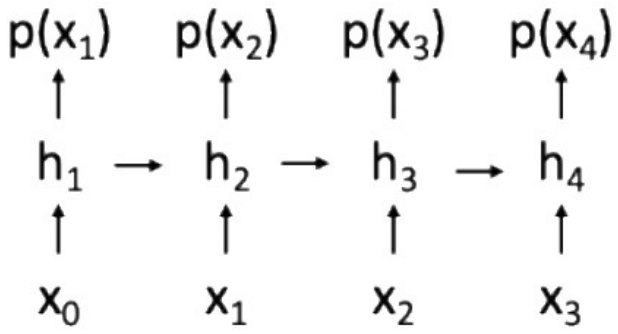
Asumir que $x$ es una secuencia $x = (x_1, x_2, \cdots, x_T)$, usar la regla de la cadena de probabilidades

$$\begin{aligned} p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\ &= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \end{aligned}$$

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

Asumir que $x$ es una secuencia $x = (x_1, x_2, \cdots, x_T)$, usar la regla de la cadena de probabilidades




$$
\begin{aligned}
p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\
&= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\
&= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})
\end{aligned}
$$

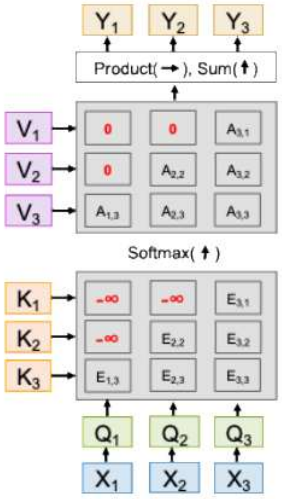
Modelado como RNN!

# Modelos autoregresivos

**Objetivo:** Definir una función explícita para $p(x) = f(x, W)$

Asumir que $x$ es una secuencia $x = (x_1, x_2, \cdots, x_T)$, usar la regla de la cadena de probabilidades

Language modeling with masked Transformer



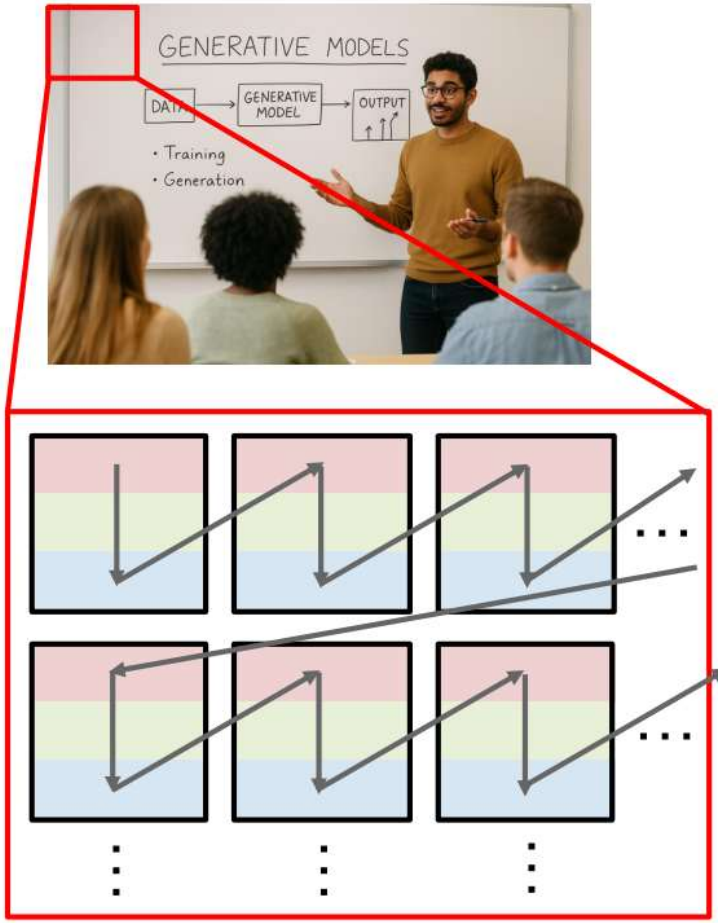

$$
\begin{aligned}
p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\
&= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\
&= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})
\end{aligned}
$$

# Modelos autoregresivos sobre imágenes

Tratar las imágenes como una secuencia de valores de 8 bits

Predecir cada píxel como una clasificación entre 256 posibles valores [0,255]

Modelar con una RNN o un Transformer



# Modelos autoregresivos sobre imágenes

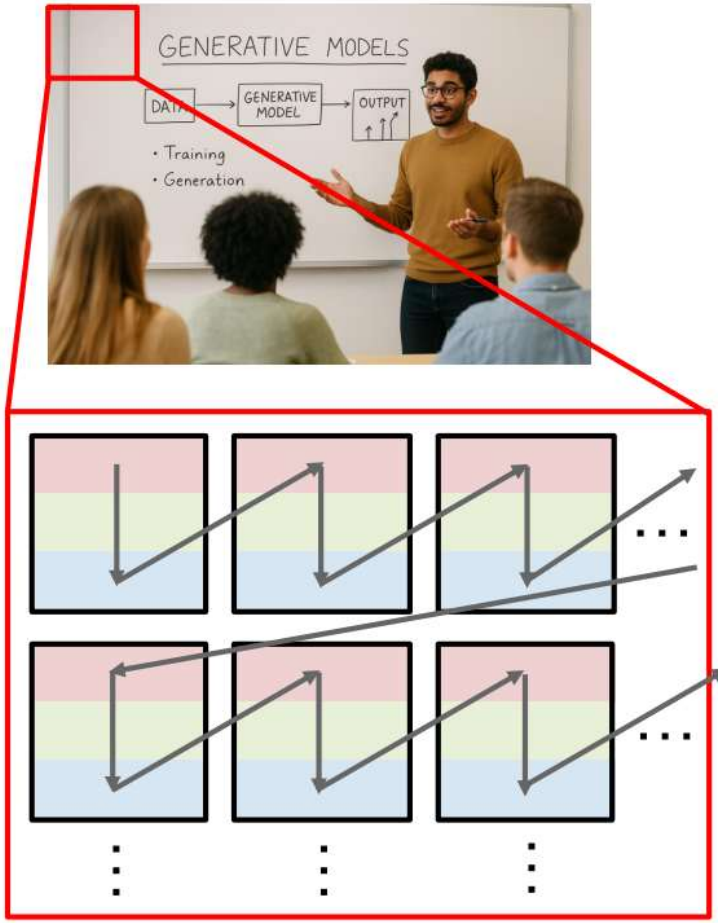
Tratar las imágenes como una secuencia de valores de 8 bits

Predecir cada píxel como una clasificación entre 256 posibles valores [0,255]

Modelar con una RNN o un Transformer

Problema: Muy costoso. Imagen de 1024x1024 tiene 3M de píxeles

Solución: Modelar como secuencia de parches



# Variational Autoencoders

PixelRNN / PixelCNN explícitamente parametriza la función de densidad con una red neuronal, así que se puede entrenar para maximizar el likelihood de la data de entrenamiento

$$\mathrm{p}_W(x) = \prod_{t=1}^{T} p_W(x_t \mid x_1, \ldots, x_{t-1})$$

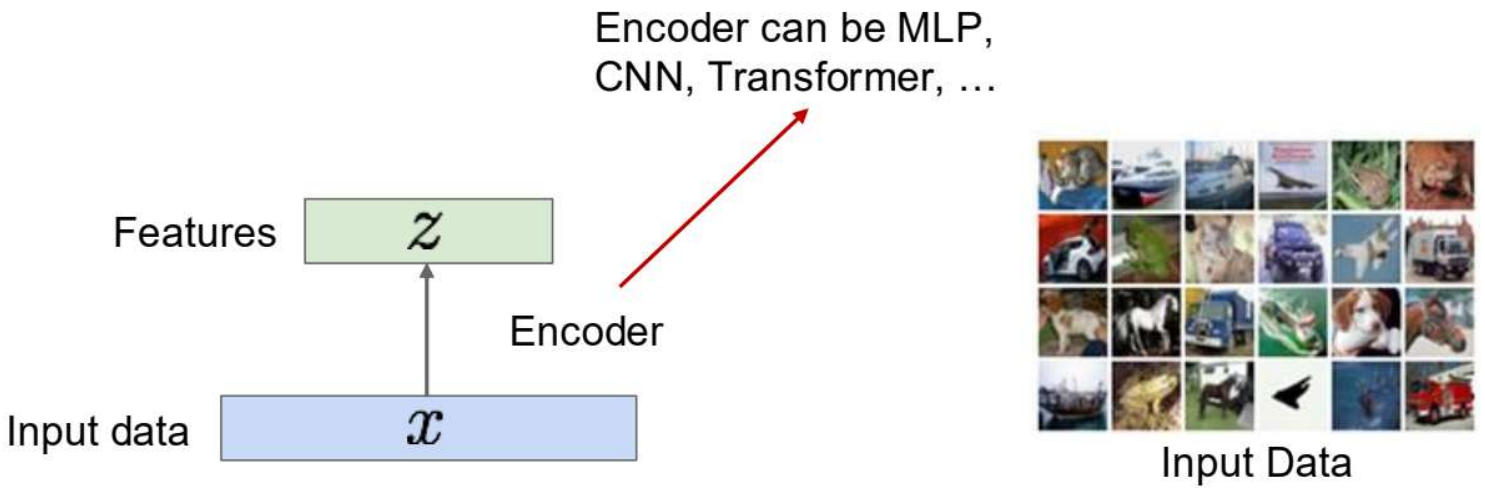
Variational Autoencoders (VAE) define una **densidad intratable** que no se puede computar explícitamente o optimizar.

Pero podemos optimizar directamente una **cota inferior** de la densidad

# Autoencoders

**Idea**: Método no supervisado para aprender a extraer features $z$ desde entradas $x$, sin etiquetas

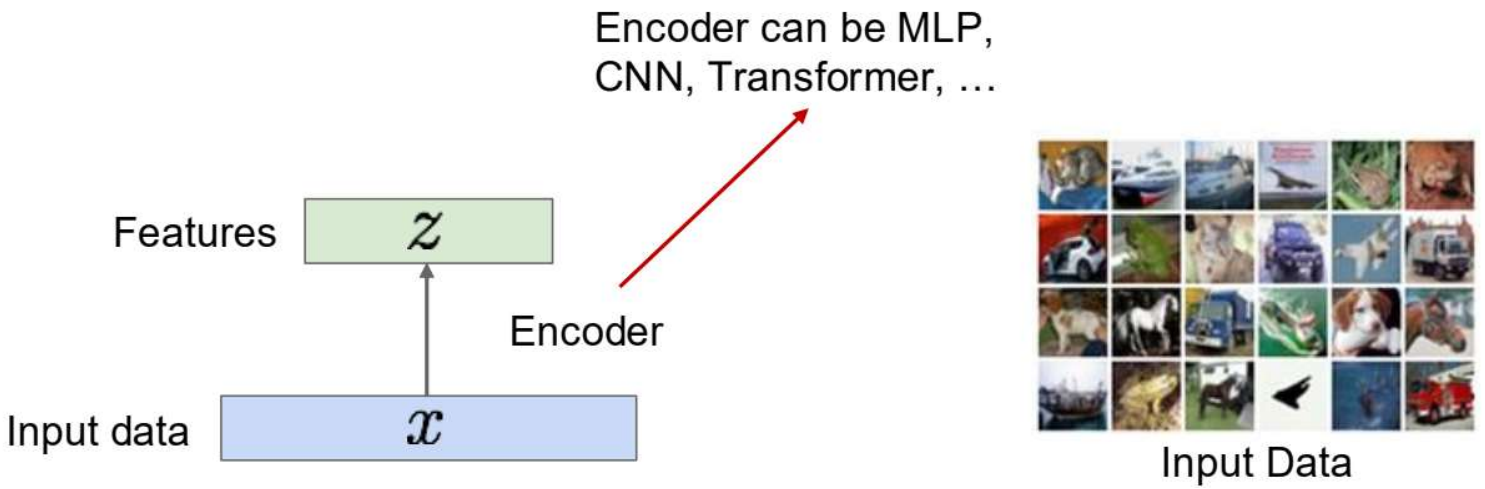
Features deberían extraer información útil (identidad, apariencia, estilo, etc.) que puedan ser usados en downstream tasks.



# Autoencoders

Problema: cómo podemos aprender sin etiquetas?

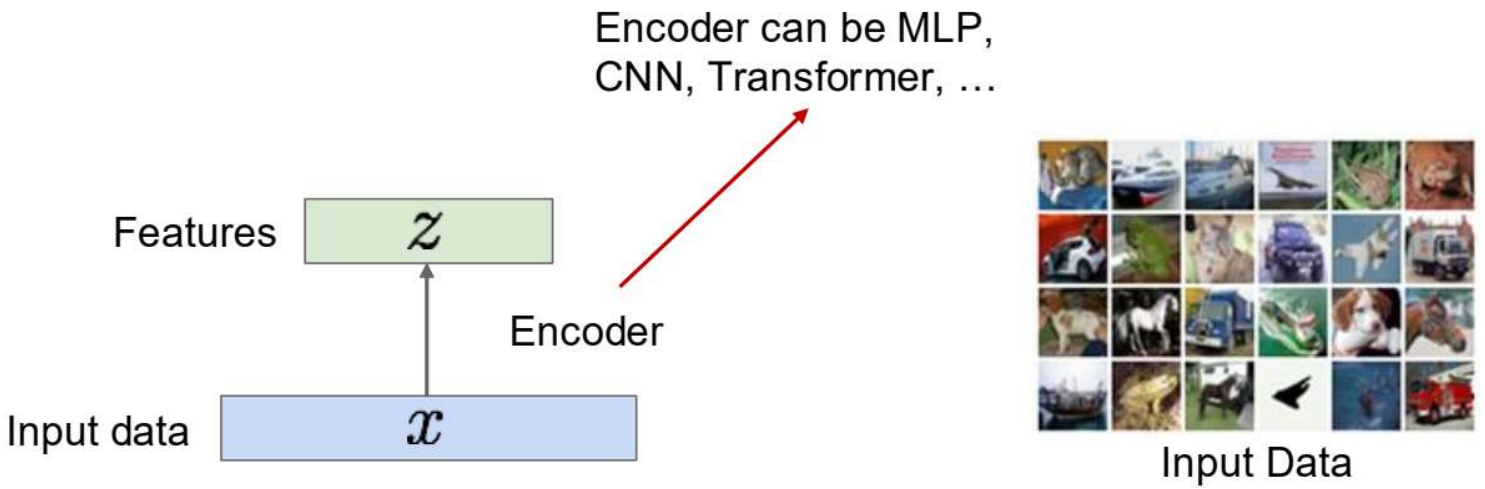
Features deberían extraer información útil (identidad, apariencia, estilo, etc.) que puedan ser usados en downstream tasks.



# Autoencoders

Problema: cómo podemos aprender sin etiquetas?

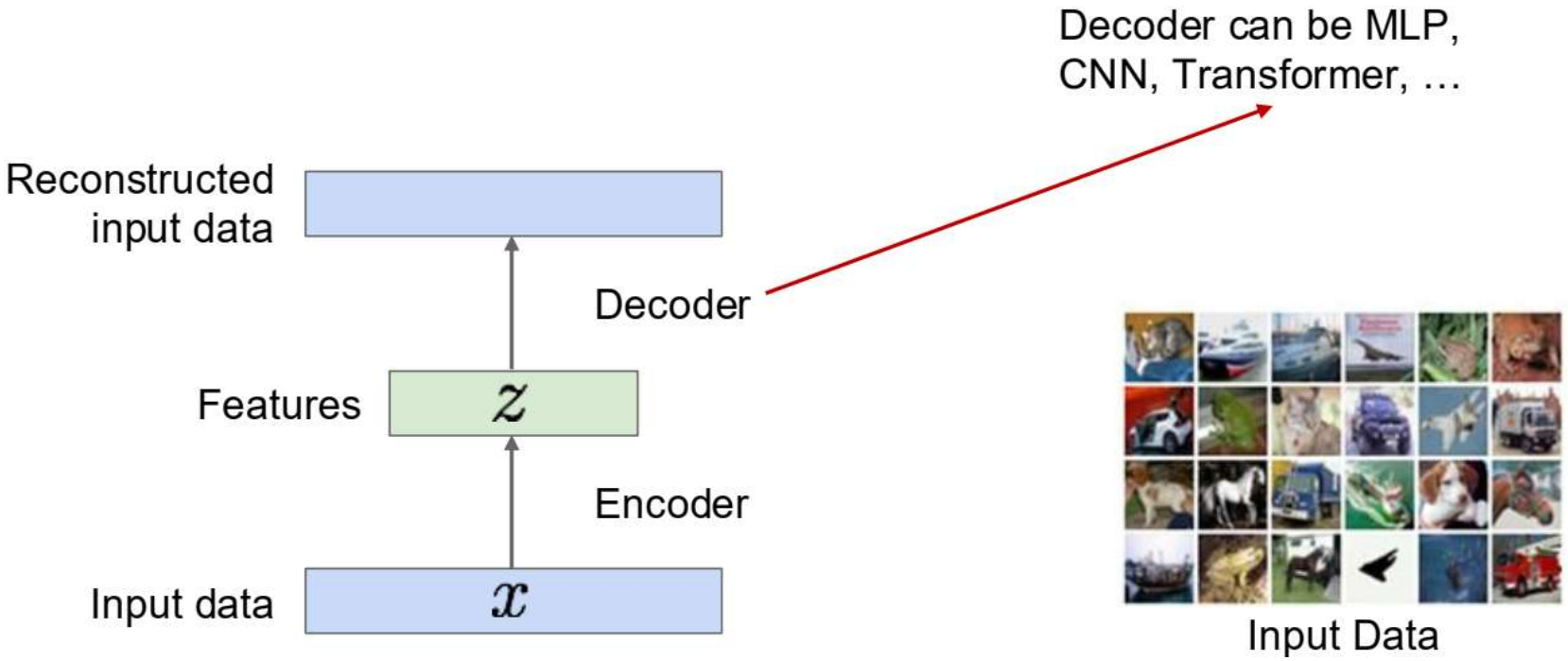
Solución: Reconstruir la data de entrada con un decoder



# Autoencoders

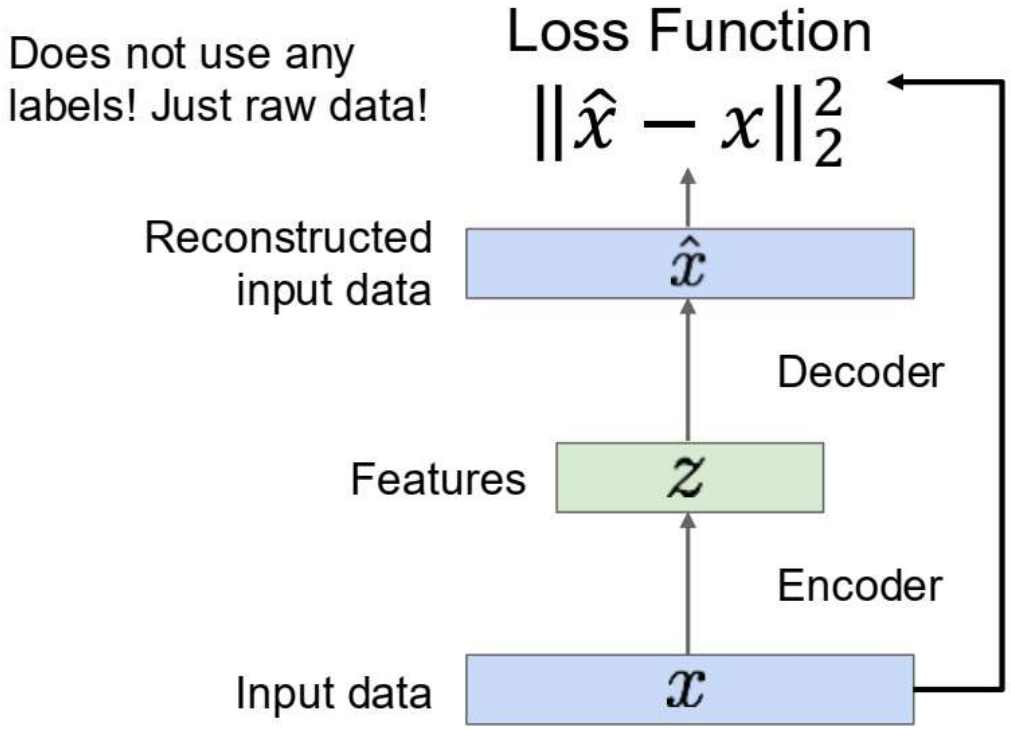
Problema: cómo podemos aprender sin etiquetas?

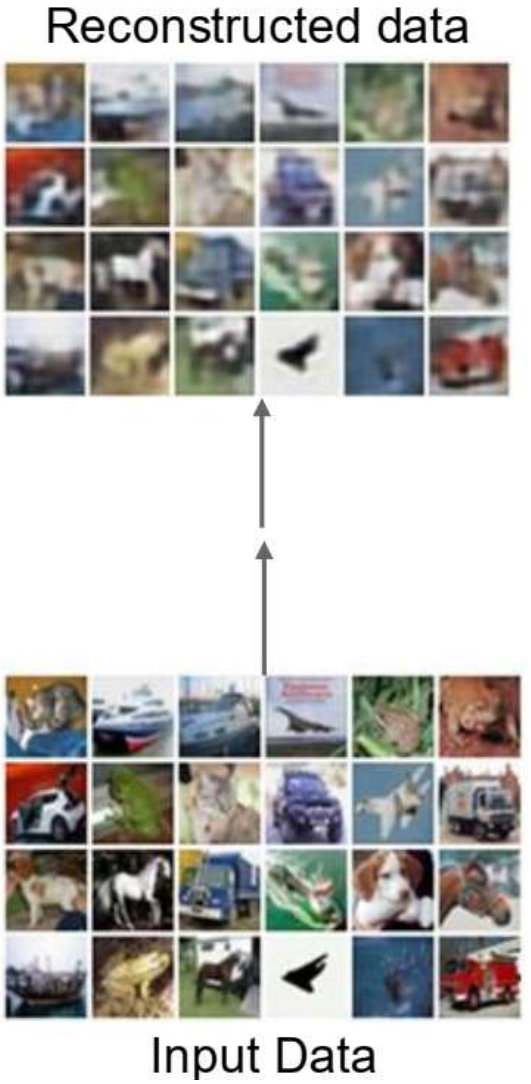
Solución: Reconstruir la data de entrada con un decoder



# Autoencoders

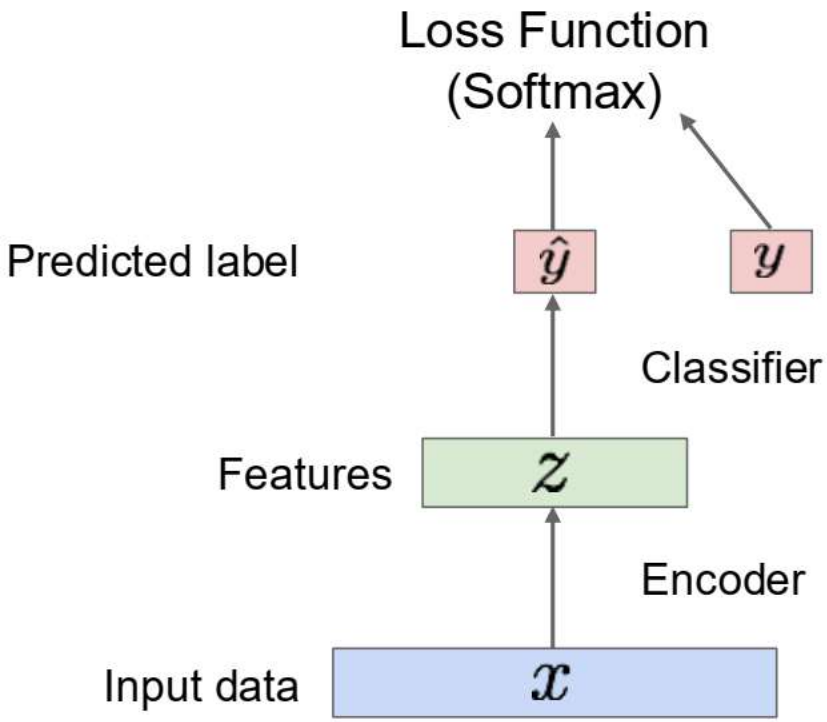
Loss: distancia L2 entre input y dato reconstruido

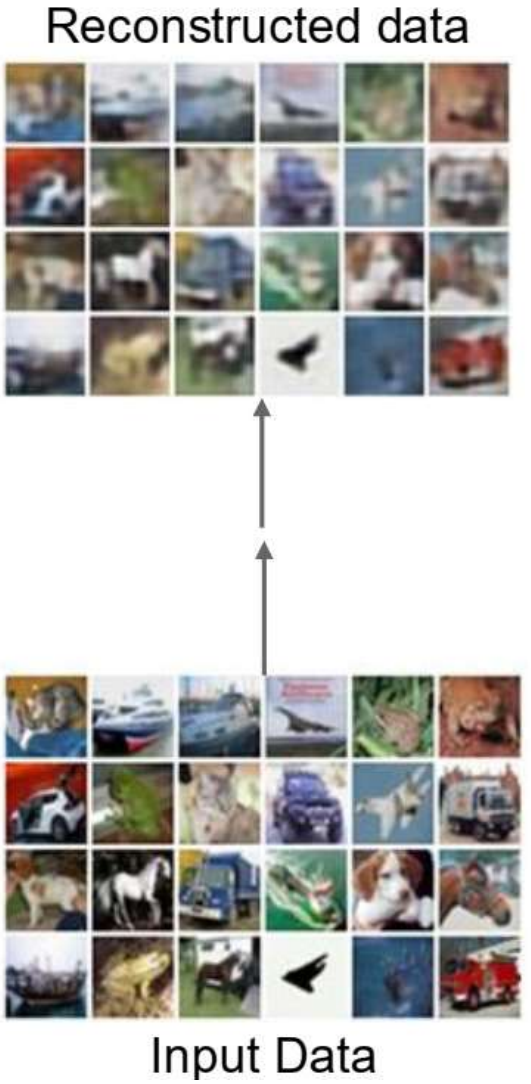


# Autoencoders

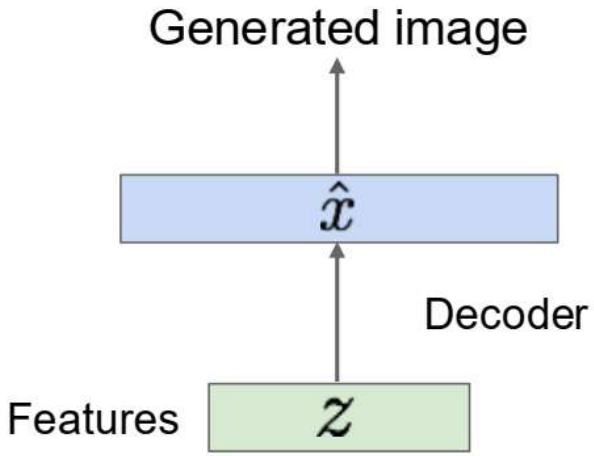
Después de entrenar, se usa el encoder para
Tareas downstream



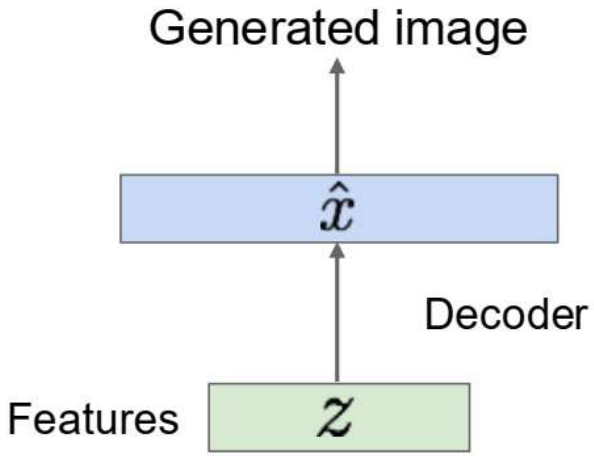
# Autoencoders

Si quieres generar un nuevo dato, podemos usar el decoder



# Autoencoders

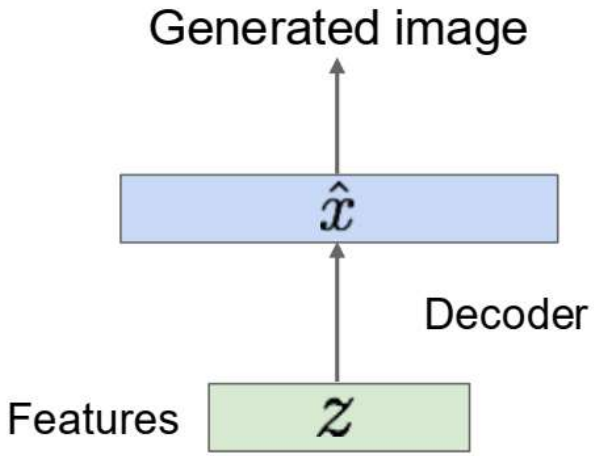
Si quieres generar un nuevo dato, podemos usar el decoder




Problema: Generar un nuevo $z$ no es fácil

# Autoencoders

Si quieres generar un nuevo dato, podemos usar el decoder




Problema: Generar un nuevo $z$ no es fácil

Solución: qué tal si forzamos que $z$ salga de una distribución conocida?

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

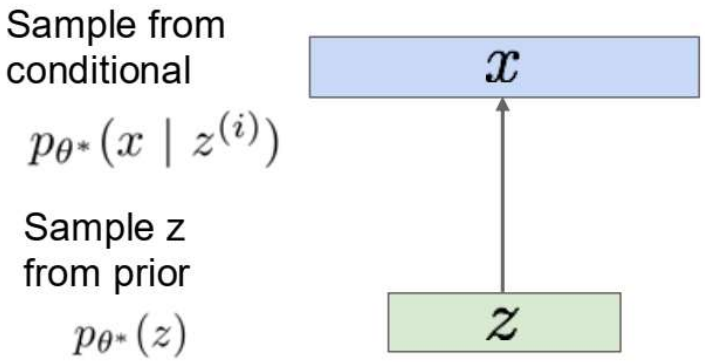
Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

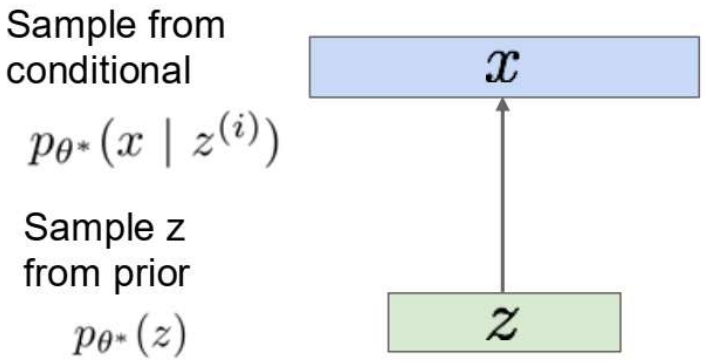
Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

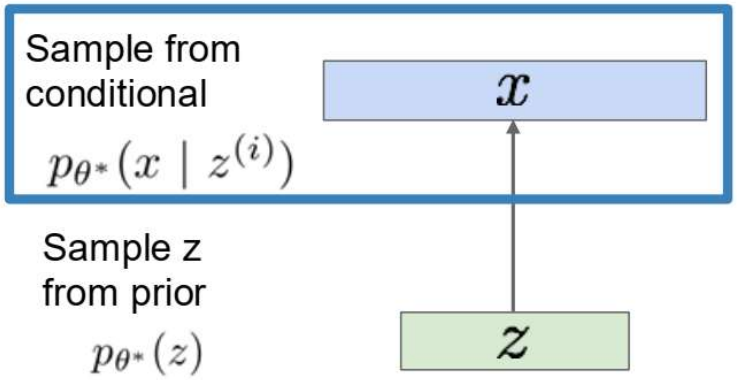
Asumir prior simple $p(z)$, Gaussiano por ejemplo

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

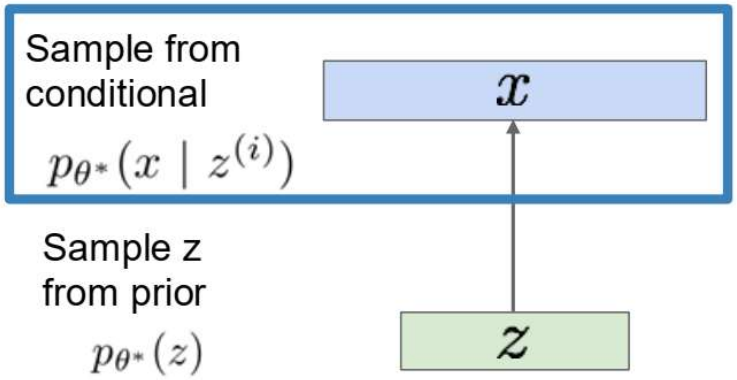
Si tuviéramos dataset de $(x, z)$ podríamos entrenar el modelo condicional

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Pero como no observamos $z$, lo marginalizamos

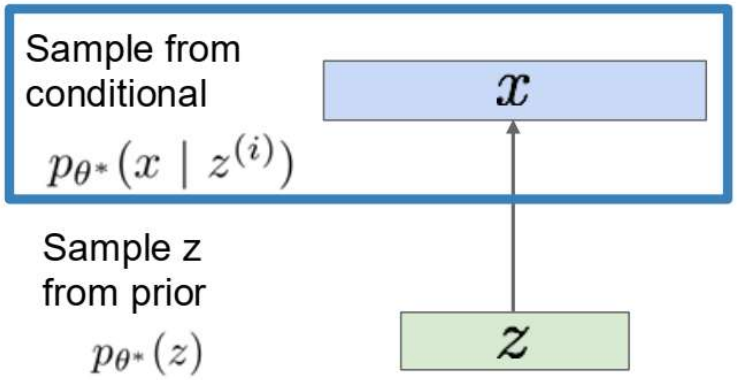
$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Pero como no observamos $z$, lo marginalizamos

$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

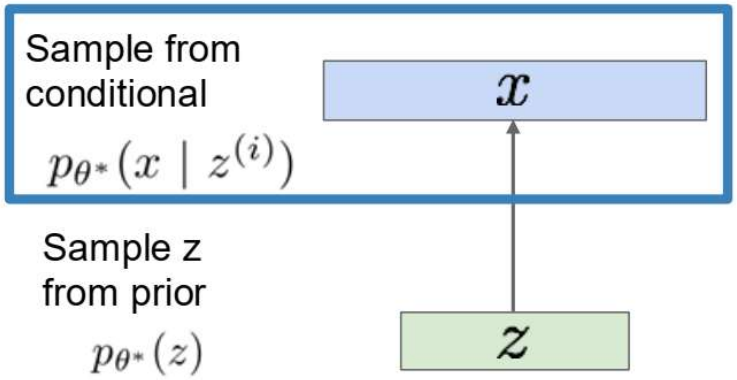
Podemos computar esto con el decoder

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Pero como no observamos $z$, lo marginalizamos

$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

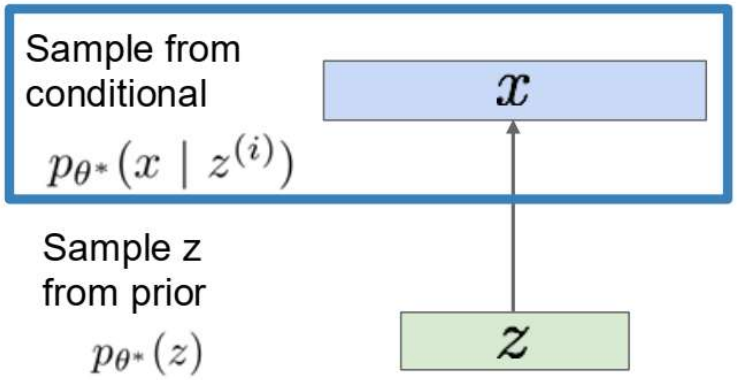
Asumiendo prior Gaussiano

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Pero como no observamos $z$, lo marginalizamos

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

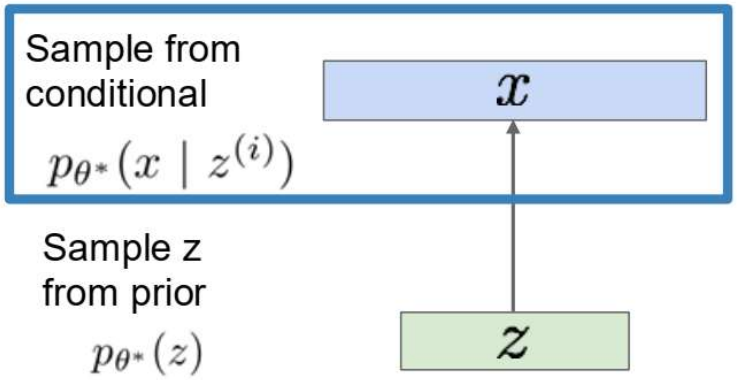
Problema: no podemos integrar sobre todo $z$

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

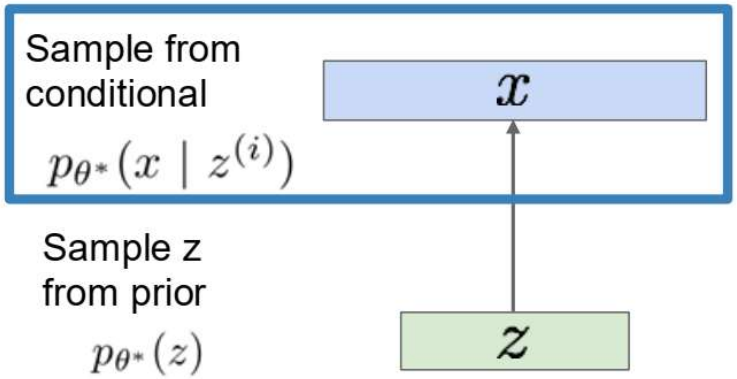
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

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

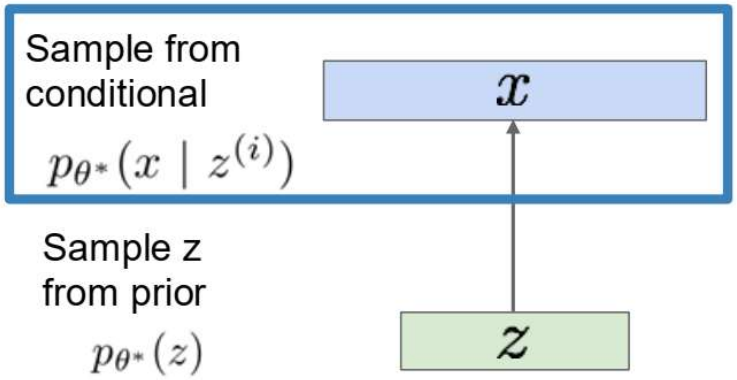
Computar con el decoder

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

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

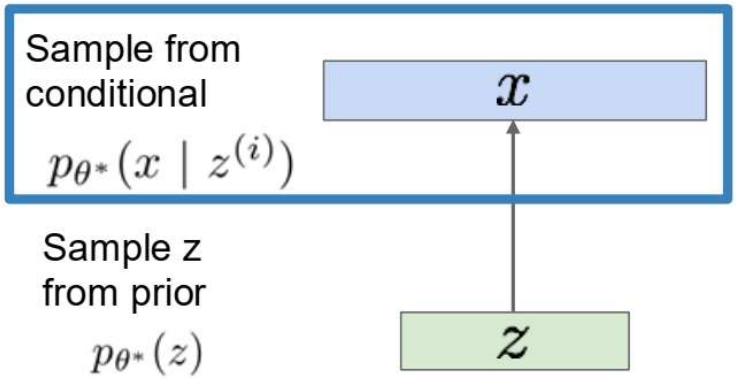
Asumiendo prior Gaussiano

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

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

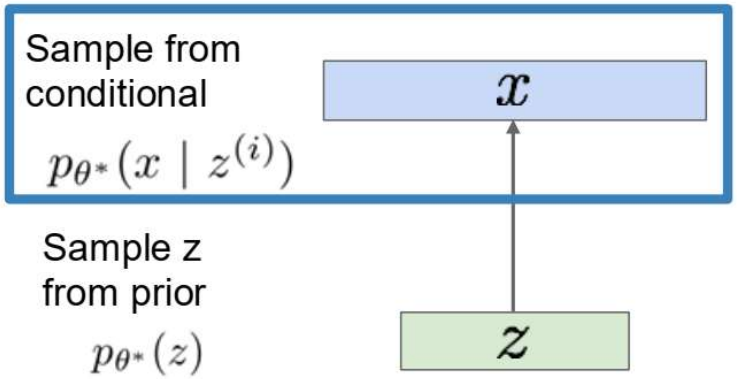
Pero no hay forma de computar esto

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

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

Pero no hay forma de computar esto

Salvo que se entrene otra red que aprenda

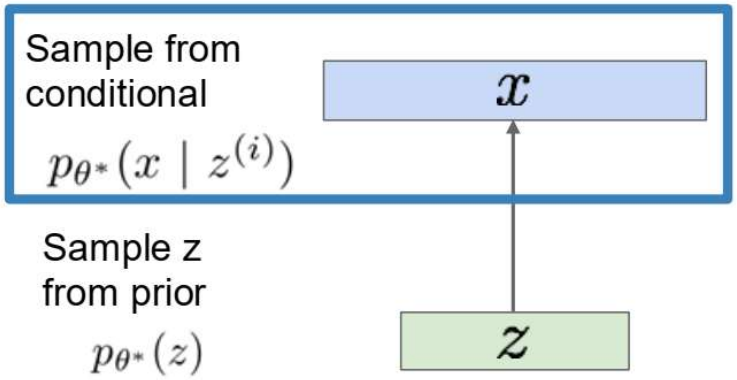
$$q_\phi(z|x) = p_\theta(z|x)$$

# Variational Autoencoders

Forma probabilística de un autoencoder

1. Aprender features latentes $z$ desde datos
2. Samplear desde el modelo para generar nueva data

Después de entrenar:




Asumir que training data $\{x^{(i)}\}_{i=1}^{N}$ es generada desde vector latente $z$

Intuición: $x$ es una imagen, $z$ es factores latentes usados para generar $x$:atributos, orientación, etc.

Idea básica: maximizar likelihood

Otra idea: usar Bayes

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)} \approx \frac{p_\theta(x \mid z)p_\theta(z)}{q_\phi(z \mid x)}$$

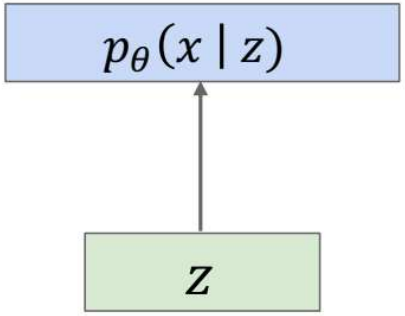
# Variational Autoencoders

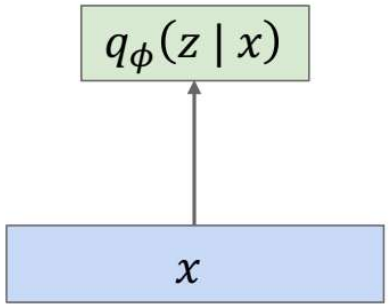
**Decoder Network**
Input $z$ latente, salida distribución sobre data $x$

**Encoder Network**
Input data $x$, salida distribución sobre latentes $z$

Si podemos asegurar que $q_\phi(z|x) \approx p_\theta(z|x)$, entonces podemos aproximar

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$




**Idea**
Entrenar conjuntamente el encoder y decoder

Cómo computar distribuciones de probabilidad?

# Variational Autoencoders

**Decoder Network**
Input $z$ latente, salida distribución sobre data $x$

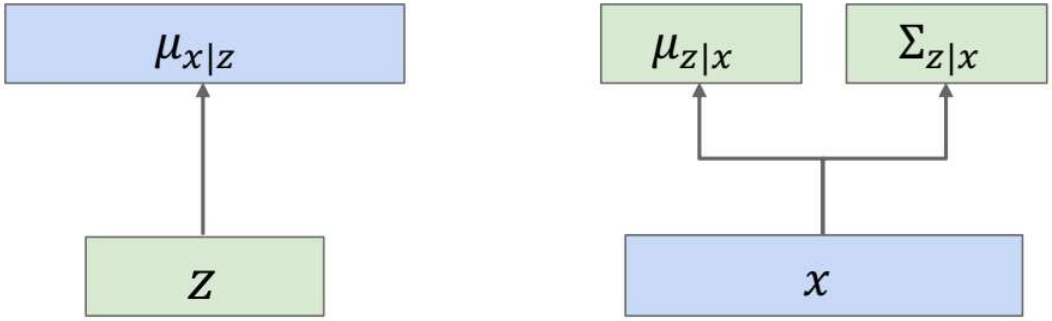
**Encoder Network**
Input data $x$, salida distribución sobre latentes $z$

Si podemos asegurar que $q_\phi(z|x) \approx p_\theta(z|x)$, entonces podemos aproximar

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

$$p_\theta(x \mid z) = N(\mu_{x|z}, \sigma^2) \qquad q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$

$$\log p_\theta(x \mid z) = -\frac{1}{2\sigma^2}\|x - \mu\|_2^2 + C_2$$




**Idea**
Entrenar conjuntamente el encoder y decoder

Maximizar $\log p_\theta(x|z)$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$

Usamos el expectation sobre todo $z$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

Término de reconstrucción: $x$→encoder→decoder debería reconstruir $x$
Se puede computar para Gaussianos

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

Salida del encoder debería corresponder con prior sobre $z$
Se puede computar para Gaussianos

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

Aproximación posterior: Salida del encoder debería aproximarse a $p_\theta(z|x)$

No se puede computar para Gaussianos

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

Aproximación posterior: Salida del encoder debería aproximarse a $p_\theta(z|x)$

KL es $\geq 0$, así que podemos sacar ese término y tener una cota inferior del likelihood

# Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

This is our VAE training objective

# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

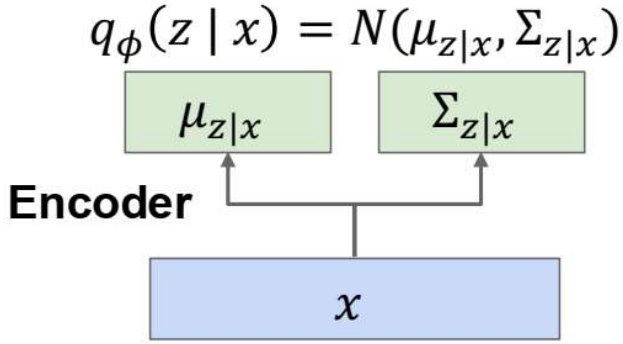
$$E_{z\sim q_\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

$$E_{z\sim q_\phi(z|x)}[log\ p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Pasar la data input por el encoder para obtener distribución sobre $z$

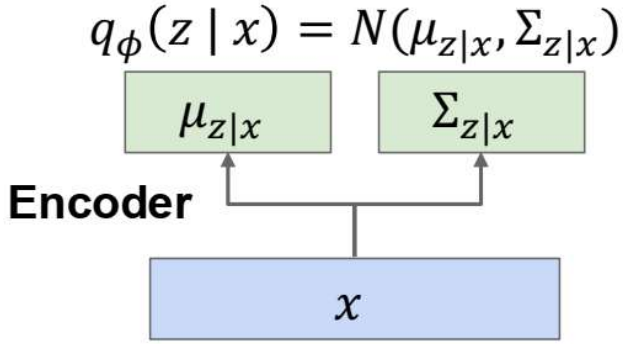


# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

$$E_{z\sim q_\phi(z|x)}[log\ p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Pasar la data input por el encoder para obtener distribución sobre $z$
2. Prior loss: Salida del encoder debería ser Gaussiano

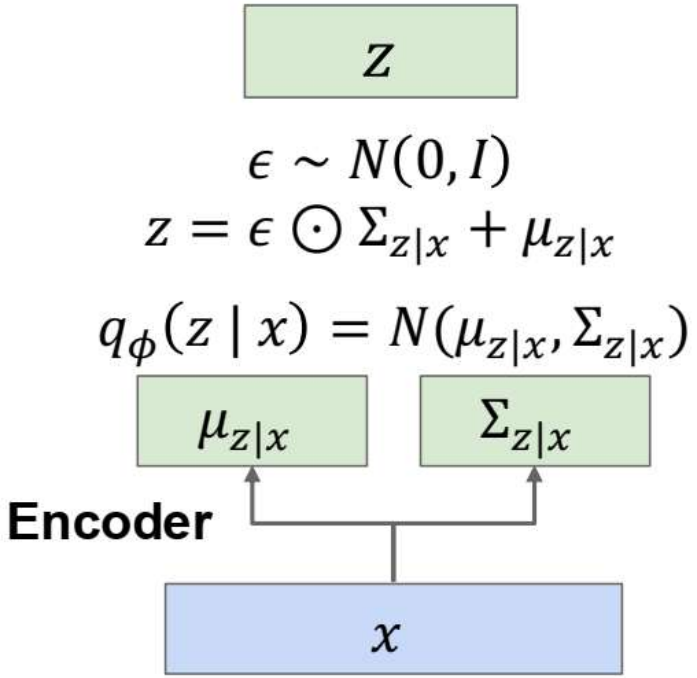


# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

$$E_{z\sim q_\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Pasar la data input por el encoder para obtener distribución sobre $z$
2. Prior loss: Salida del encoder debería ser Gaussiano
3. Samplear $z$ desde la salida del encoder $q_\phi(z \mid x)$

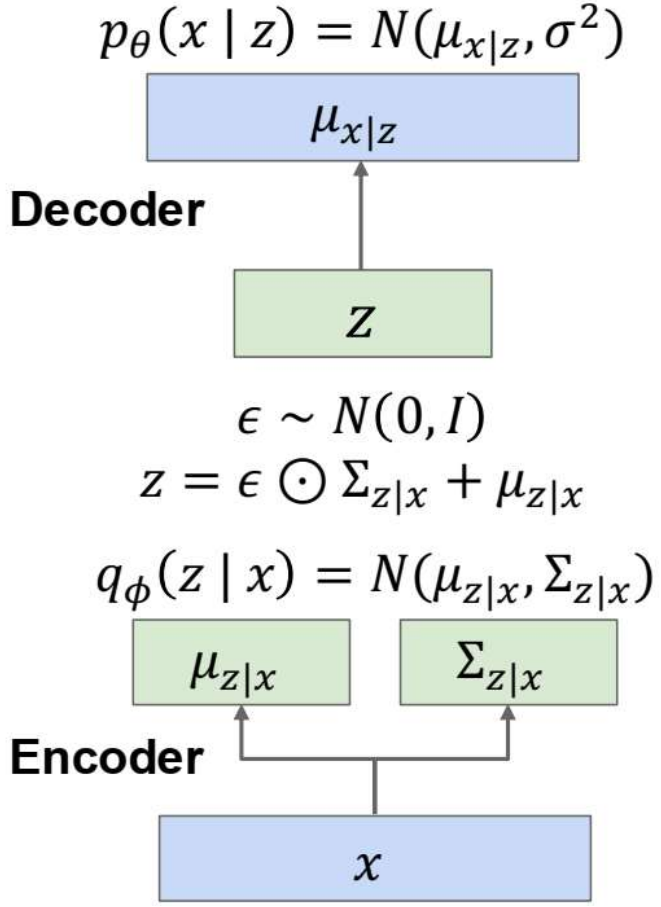


# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

$$E_{z\sim q_\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Pasar la data input por el encoder para obtener distribución sobre $z$
2. Prior loss: Salida del encoder debería ser Gaussiano
3. Samplear $z$ desde la salida del encoder $q_\phi(z \mid x)$
4. Pasar $z$ por el decoder para obtener media de datos

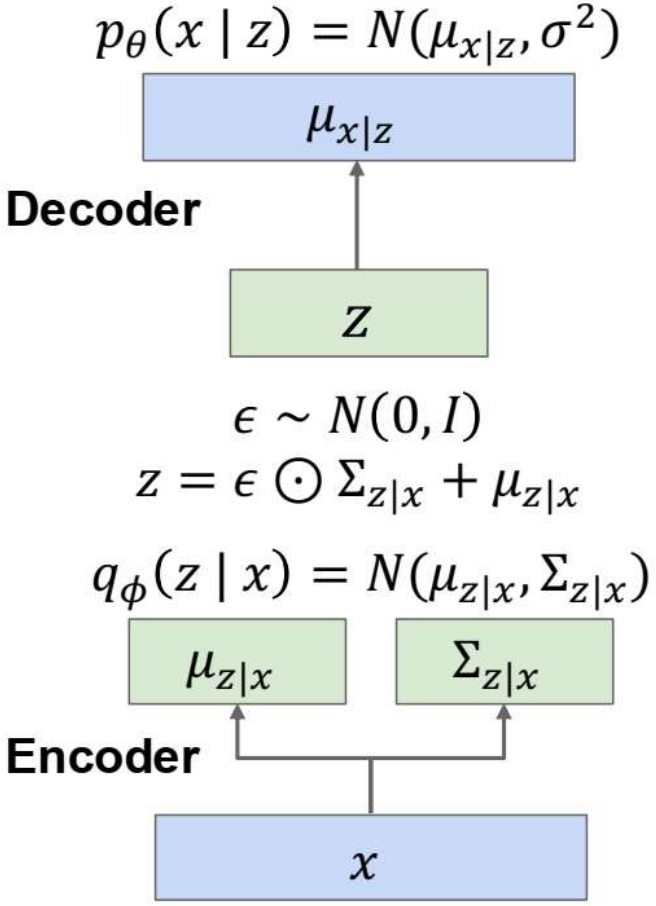


# Variational Autoencoders - Training

Entrenar por maximizar la cota inferior

$$E_{z\sim q_\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Pasar la data input por el encoder para obtener distribución sobre $z$
2. Prior loss: Salida del encoder debería ser Gaussiano
3. Samplear $z$ desde la salida del encoder $q_\phi(z \mid x)$
4. Pasar $z$ por el decoder para obtener media de datos
5. Loss de reconstrucción



# Modelos de difusión - Intuición

Escoger una **distribución de ruido**
$z \sim p_{noise}$ (usualmente Gaussiano)

# Modelos de difusión - Intuición

Escoger una **distribución de ruido** $z \sim p_{noise}$ (usualmente Gaussiano)

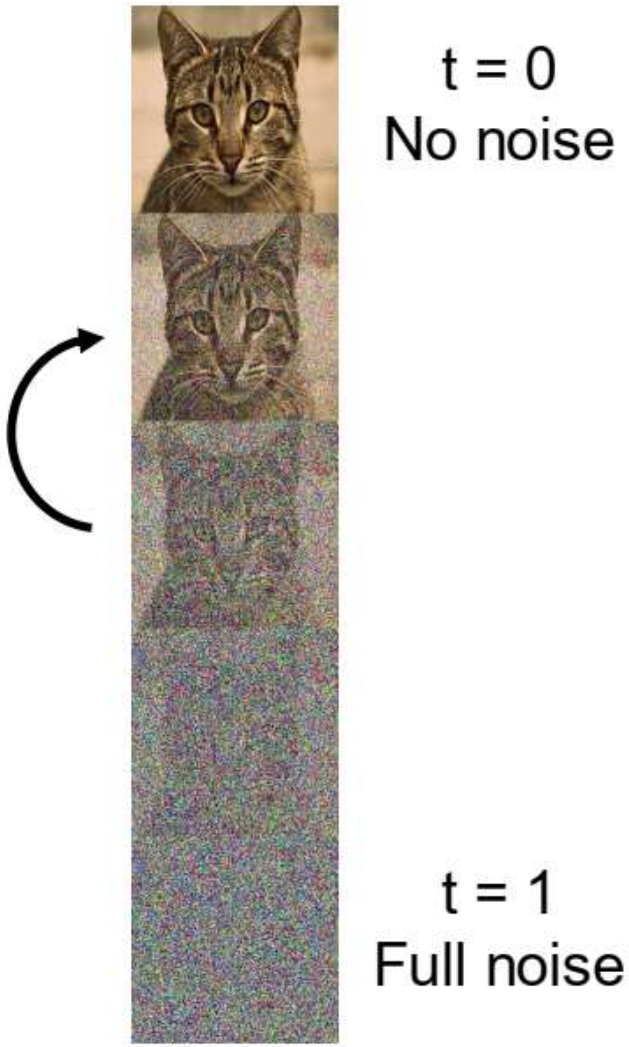
Considerar data corrupta $x$ bajo diferentes niveles de ruido $t$ hasta obtener $x_t$

# Modelos de difusión - Intuición

Escoger una **distribución de ruido** $z \sim p_{noise}$ (usualmente Gaussiano)

Considerar data corrupta $x$ bajo diferentes niveles de ruido $t$ hasta obtener $x_t$

Entrenar una red neuronal para remover un poco de ruido $f_\theta(x_t, t)$



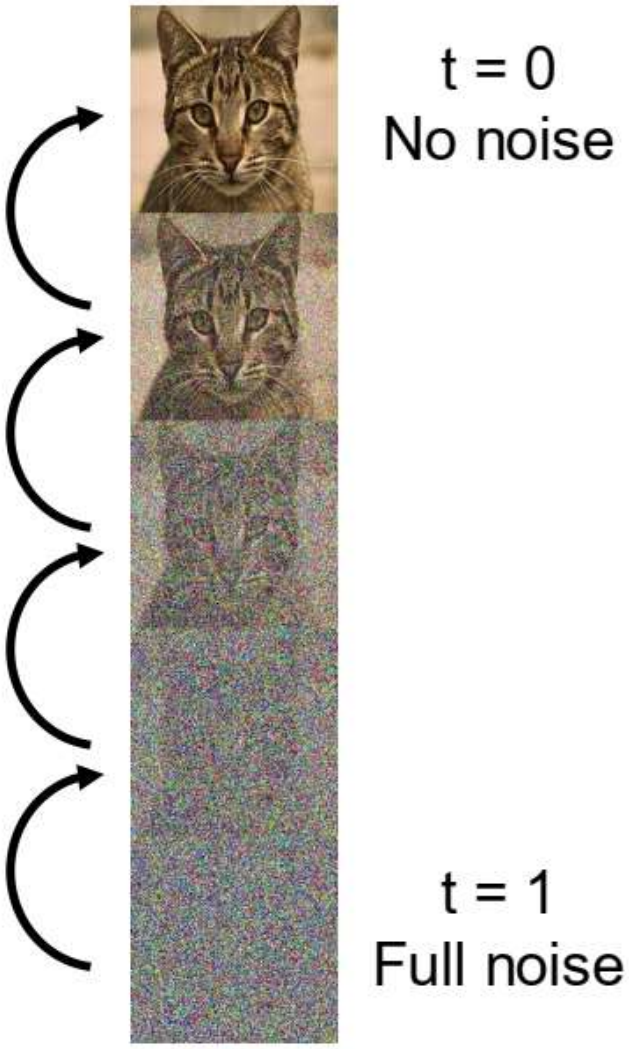
# Modelos de difusión - Intuición

Escoger una **distribución de ruido** $z \sim p_{noise}$ (usualmente Gaussiano)

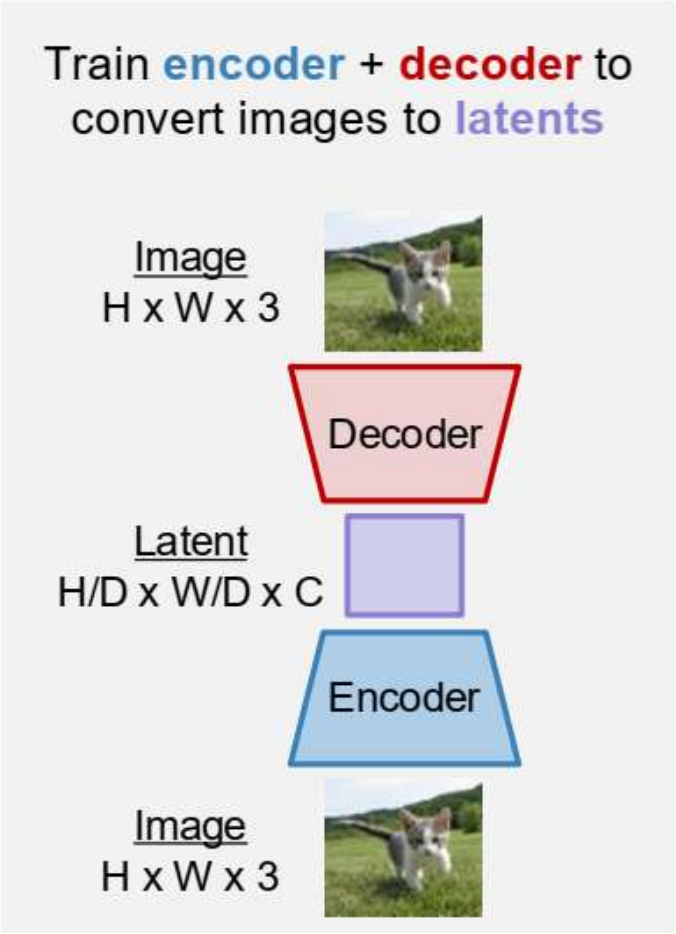
Considerar data corrupta $x$ bajo diferentes niveles de ruido $t$ hasta obtener $x_t$

Entrenar una red neuronal para remover un poco de ruido $f_\theta(x_t, t)$

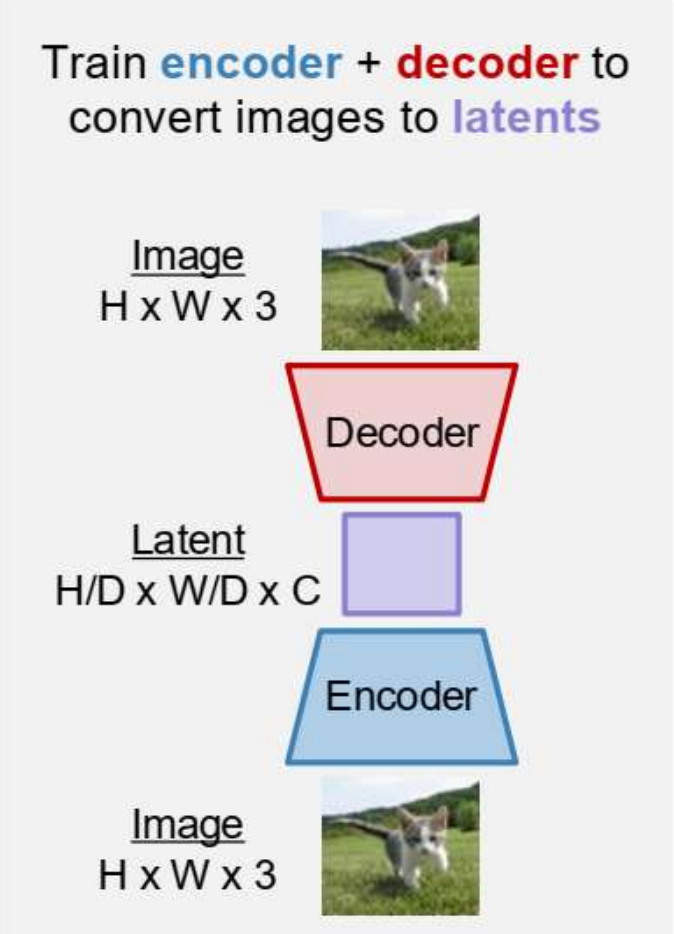
En inferencia, samplear $x_1 \sim p_{noise}$ y aplicar $f_\theta$ muchas veces en secuencia para generar $x_0$

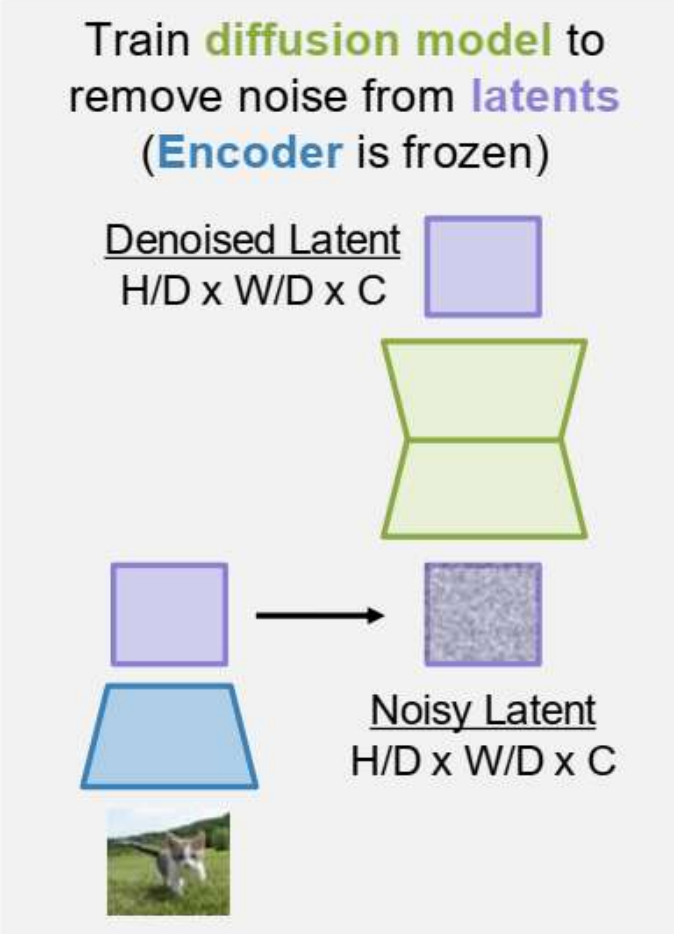


# Modelos de difusión latentes

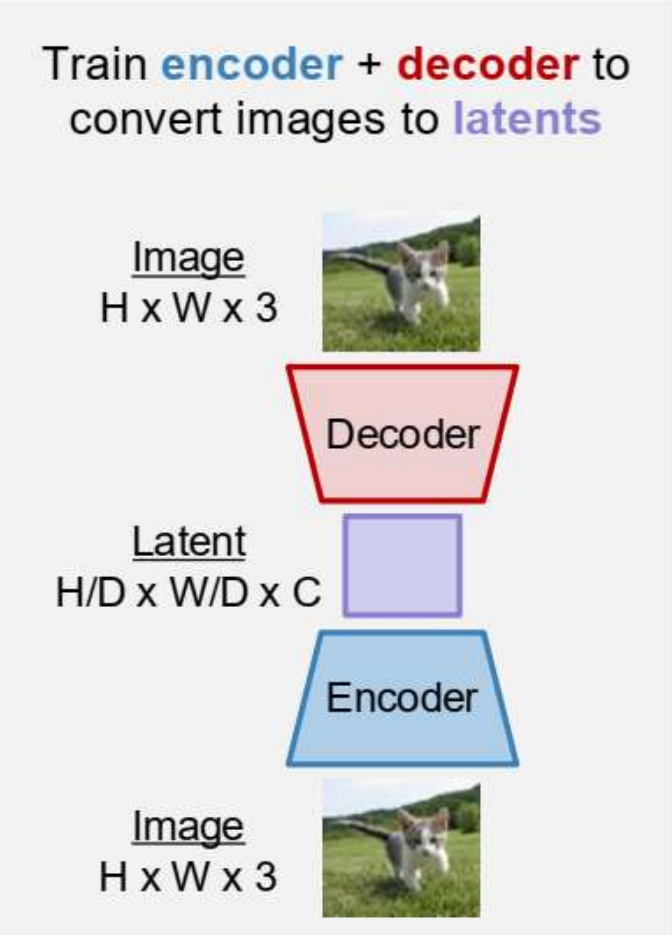


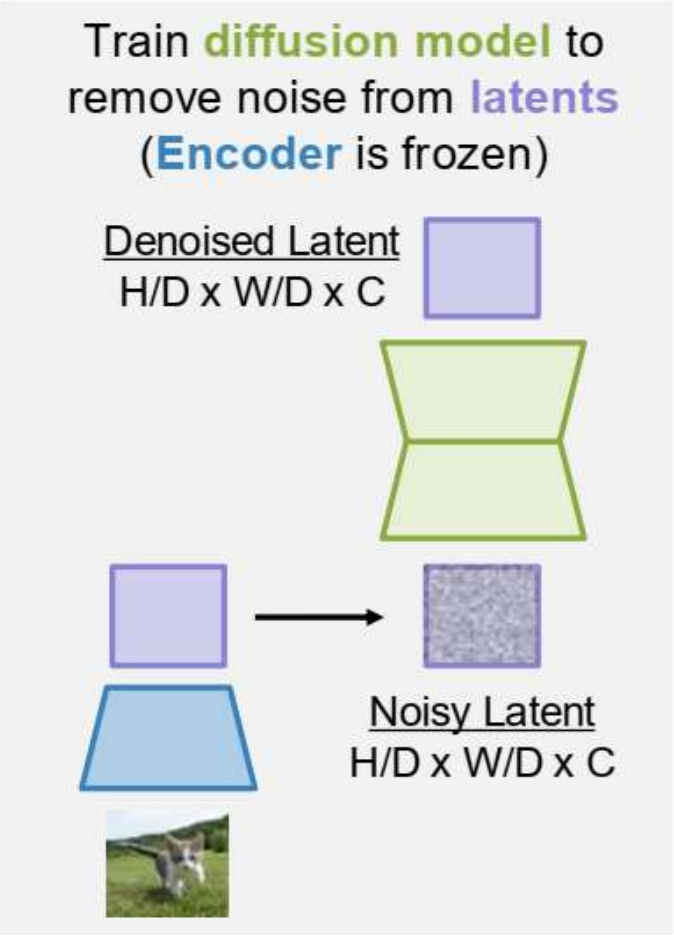
# Modelos de difusión latentes

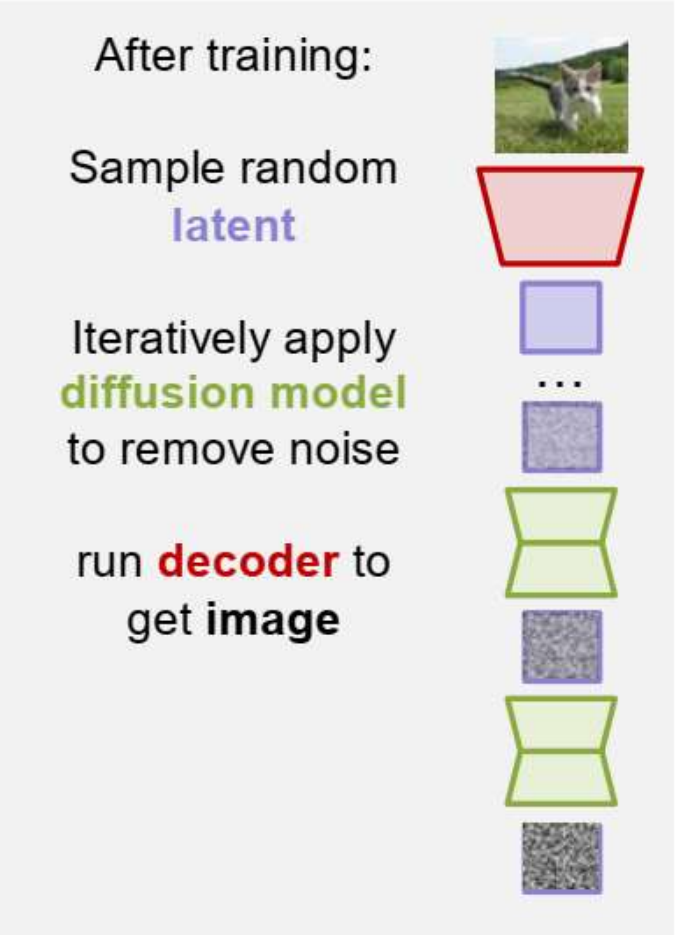


# Modelos de difusión latentes



# Modelos de difusión latentes