

# Aprendizaje profundo

## REDES NEURONALES GENERATIVAS

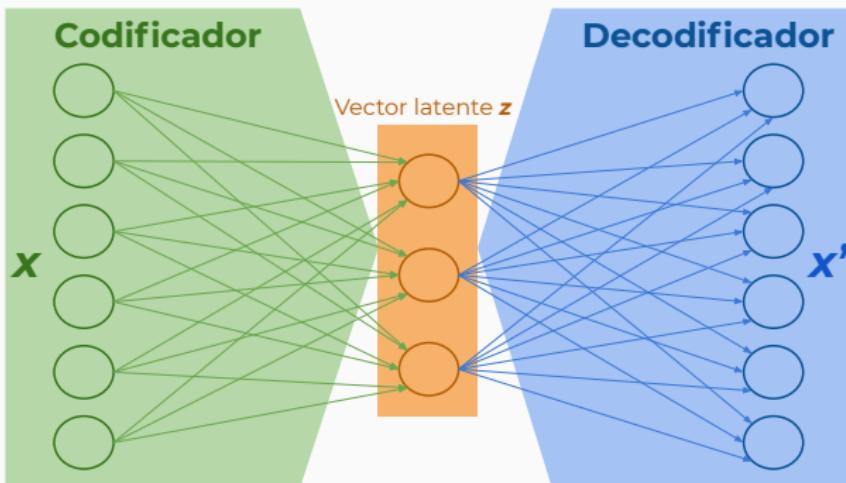
---

Gibran Fuentes-Pineda

Enero 2021

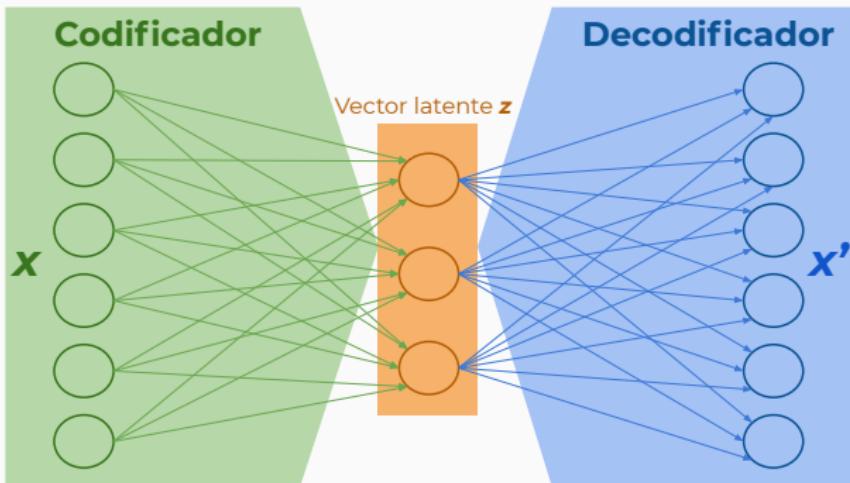
# Esquema codificador-decodificador

- Red entrenada generando sus mismas entradas
  - Codificador:  $z = F_{W_c, b_c}(x)$
  - Decodificador:  $x' = G_{W_d, b_d}(z)$
  - $W_c, b_c, W_d, b_d = \arg \min_{W_c, b_c, W_d, b_d} \|x - x'\|_2^2$

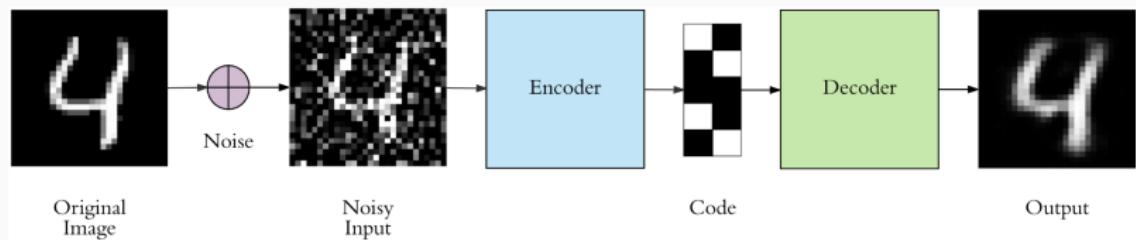


# Autocodificadores contractivos

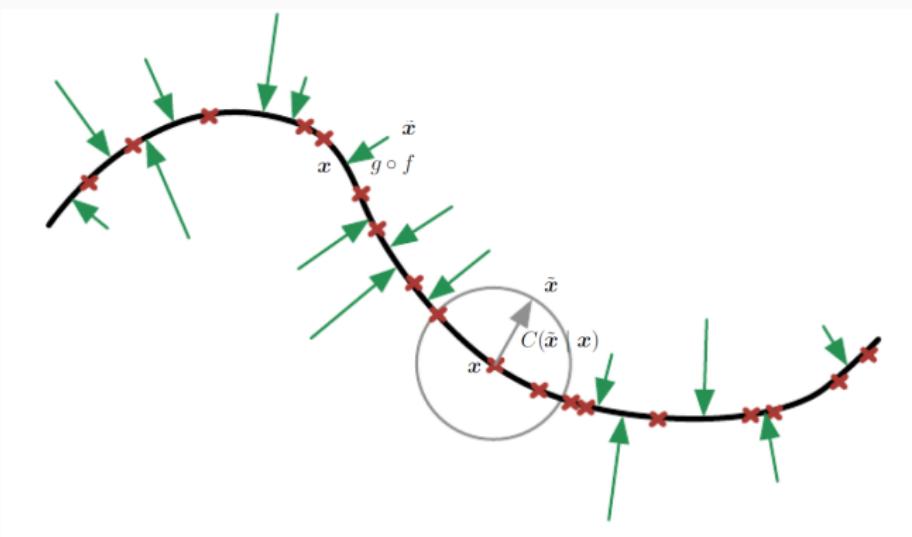
- Hace  $z$  de menor dimensionalidad que  $x$



# Autocodificadores quita ruido



# Autocodificadores quita ruido



# Autocodificadores dispersos

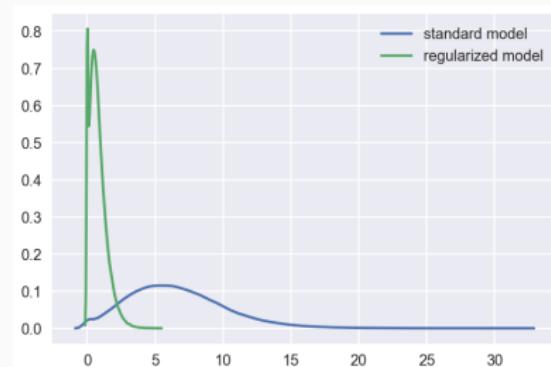


Figura tomada de <https://medium.com/towards-data-science/applied-deep-learning-part-3-autoencoders-1c083af4d798>

# Tipos de autocodificadores

---

- Estrategias para evitar aprender la función identidad
  - Hacer  $z$  de menor dimensionalidad que  $x$
  - Añadiendo ruido a entrada y tratando de reconstruir entrada original (*Denosing Autoencoders*)
  - Forzando  $z$  a ser disperso (*Sparse Autoencoders*)
  - Maximizando probabilidad de datos (*Variational Autoencoders*)

# Autocodificadores como estrategia de inicialización

- Usado como estrategia de inicialización de pesos por capa (Restricted Boltzman Machines)

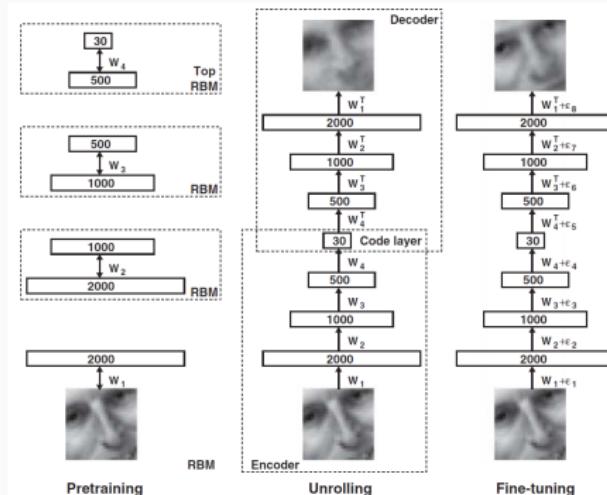


Figura tomada de Hinton y Salakhutdinov 2006

# Inferencia variacional

- Hacen presuposiciones sobre distribución a posteriori para hacerla más tratable computacionalmente

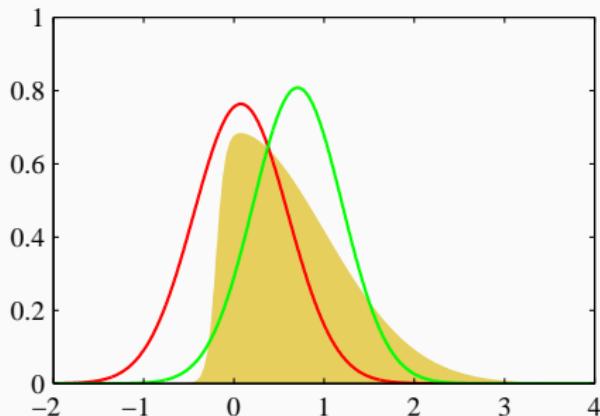


Imagen tomada de C. Bishop. *Pattern Recognition and Machine Learning*, 2006

# Divergencia de Kullback–Leibler

- Medida de cómo una distribución diverge de otra distribución esperada
  - 0 – similar
  - 1 – diferente

$$D_{KL}(P|Q) = \sum_{x \in X} P(x) \log \left( \frac{P(x)}{Q(x)} \right) = - \sum_{x \in X} P(x) \log \left( \frac{Q(x)}{P(x)} \right)$$

# Autocodificadores variacionales (ACV)

- Hacen presuposiciones sobre distribución de  $z$ 
  - $z \sim P_\theta(Z)$
  - $x \sim P_\theta(X|Z)$
- Proceso generativo
  1. Se obtiene muestra  $z^{(i)}$  de distribución a priori  $z^{(i)} \sim P_\theta(Z)$
  2. Se muestrea  $x^{(i)}$  de distribución condicional  
 $x^{(i)} \sim P_\theta(x^{(i)}|z^{(i)})$
- Problema: estimar  $\theta$  sin conocer valores de  $z$

- $P_{\theta}(Z) \sim \mathcal{N}(0, I)$
- $P_{\theta}(X|Z) \sim \mathcal{N}(\mu, \sigma^2)$
- Inferencia de distribución a posterior es intratable

$$P_{\theta}(Z|X) = \frac{P_{\theta}(X|Z)P_{\theta}(Z)}{P_{\theta}(X)}$$

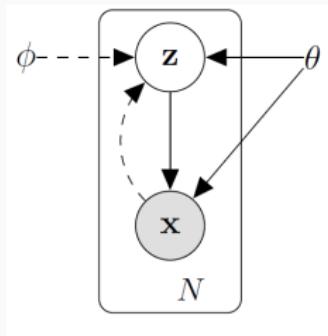


Imagen tomada de

## ELBO: Evidence Lower BOund

- En el entrenamiento de los autocodificadores variacionales se maximiza la cota inferior de evidencia o ELBO:

$$ELBO(x) = \sum_z Q(z|x) \log(P(x|z)) + \sum_z Q(z|x) \log\left(\frac{Q(z|x)}{P(z)}\right)$$

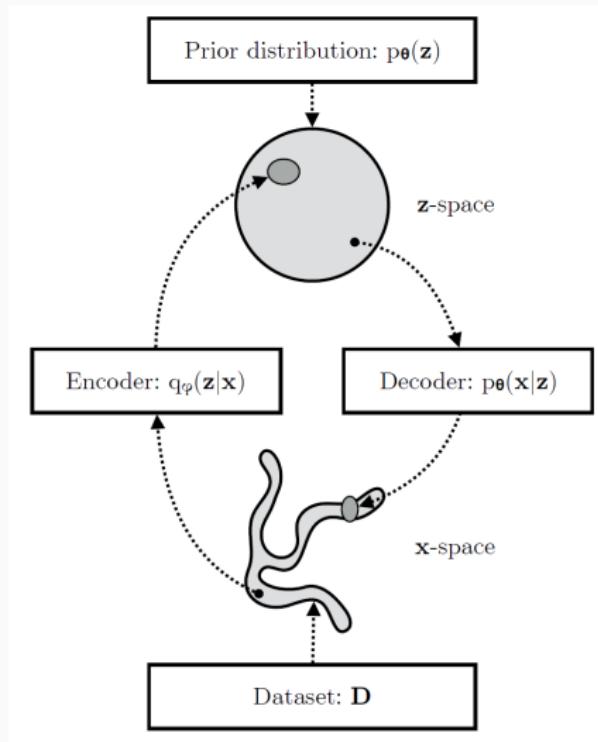
- ELBO es una cota inferior variacional de  $\log(P(X))$

- Aproxima distribución a posteriori con red codificadora  $Q_\phi(Z|X)$
- Cota inferior de  $\log P_\theta(X)$  (aproxima máxima verosimilitud):  $\phi$  y  $\theta$  optimizadas conjuntamente
  - Equivalente a minimizar la divergencia de Kullback Leibler (KL)
- SGD
  1. Muestreo en mini-lotes
  2. Muestreo de distribución a posteriori aproximada

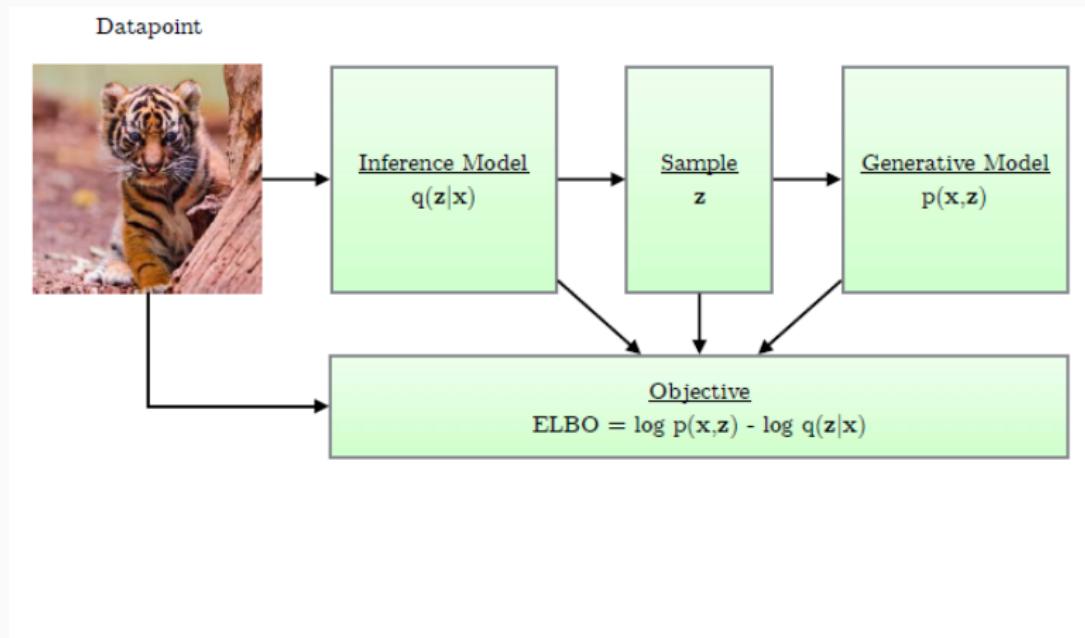
## Codificador-decodificador variacional (1)

- **Red codificadora** genera media  $\mu_Z$  y covarianza diagonal  $\Sigma_Z$  de  $Q_\phi(Z|X)$
- **Red decodificadora** genera media  $\mu_X$  y covarianza diagonal  $\Sigma_X$  de  $P_\theta(X|Z)$
- $x'^{(i)}$  se obtiene
  1. Muestreando  $z^{(i)}$  de  $Q_\phi(Z|X)$
  2. Muestreando  $x^{(i)}$  de  $P_\theta(X|Z)$

## Codificador-decodificador variacional (2)

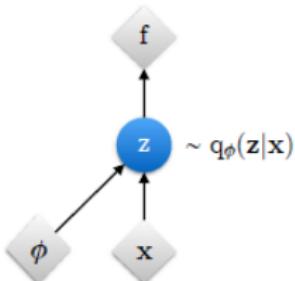


# Entrenamiento del autocodificador variacional

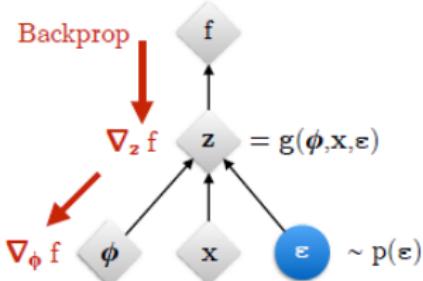


# Truco de reparametrización

Original form



Reparameterized form



: Deterministic node



: Evaluation of  $f$



: Random node



: Differentiation of  $f$

# Esquema general del modelo de autocodificador variacional

- Busca optimizar divergencia de Kullback–Leibler esperada

$$\mathcal{D}_{KL}(Q_\phi(Z|X) || P_\theta(X|Z))$$

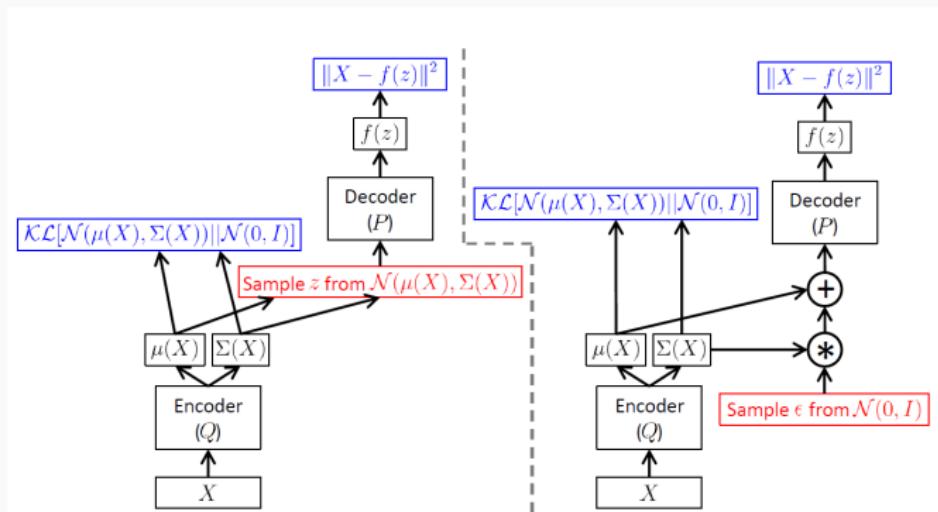
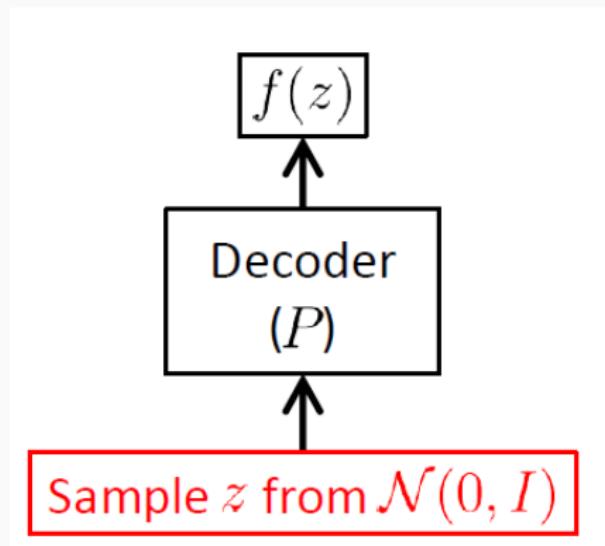


Imagen tomada de Carl Doersch. *Tutorial on Variational Autoencoders*, arXiv:1606.05908, 2016

## Generación de datos

- Para generar nuevos datos se muestrea  $\mathbf{z}$  de  $\mathcal{N}(0, \mathbb{I})$



## Aplicaciones: generación de dígitos

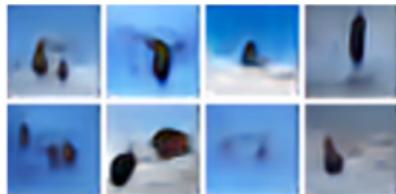
## Aplicaciones: generación de imágenes



## Aplicaciones: generación de rostros



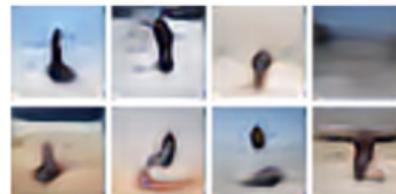
A stop sign is flying in blue skies.



A herd of elephants flying in the blue skies.



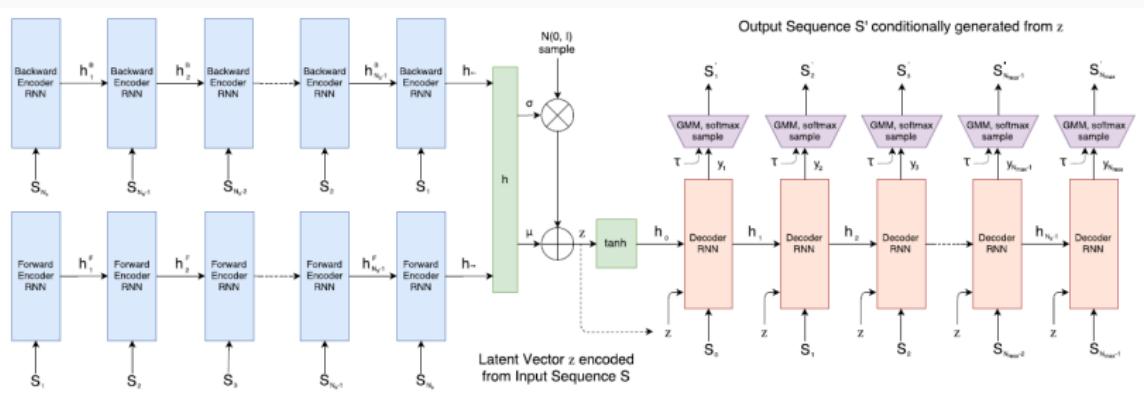
A toilet seat sits open in the grass field.



A person skiing on sand clad vast desert.

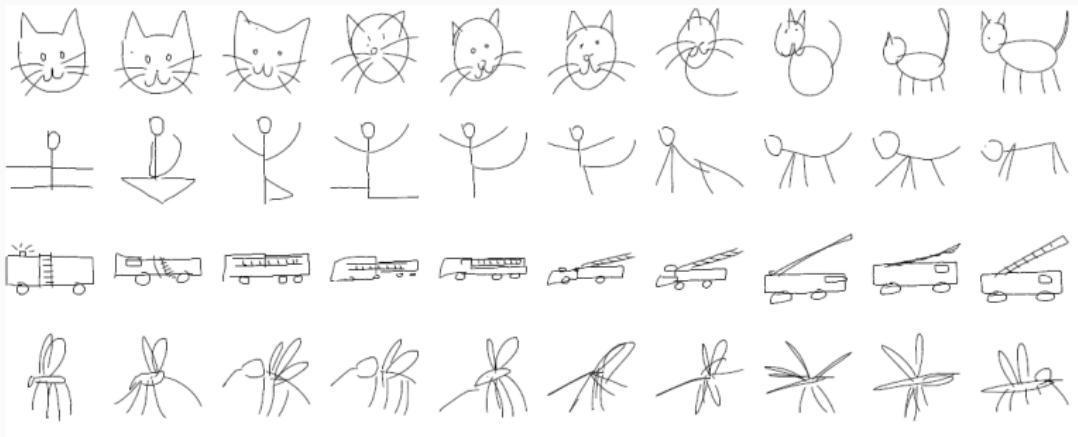
# Aplicaciones: generación de dibujos

- sketch-rnn



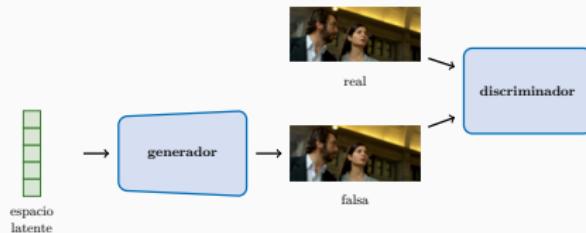
# Aplicaciones: generación de dibujos

- sketch-rnn



# Redes generativas antagónicas (GANs)

- Dos redes en competencia
  - Generadora trata de engañar a discriminadora con ejemplos falsos
  - *Discriminadora* trata de identificar si un ejemplo es real o falso



## Generador vs discriminador (1)

---

- Un juego tipo minimax

$$\mathcal{L}_{\mathcal{D}} = -\frac{1}{2} \mathbb{E}_{x \sim data} [\log(\mathcal{D}(x))] - \frac{1}{2} \mathbb{E}_z [\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

$$\mathcal{L}_{\mathcal{G}} = -\mathcal{L}_{\mathcal{D}}$$

- Equilibrio es un punto singular de la pérdida del discriminador  $\mathcal{D}$
- Generador  $\mathcal{G}$  minimiza la probabilidad logarítmica de acierto del discriminador  $\mathcal{D}$

## Generador vs discriminador (2)

- Versión no saturada

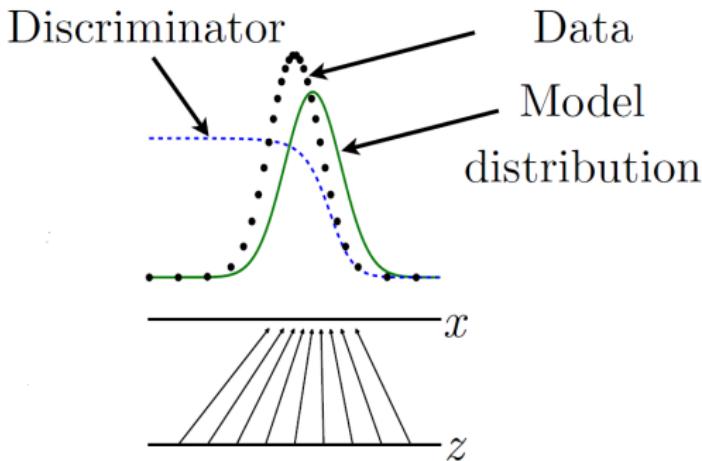
$$\begin{aligned}\mathcal{L}_{\mathcal{D}} &= -\frac{1}{2} \mathbb{E}_{x \sim data} [\log(\mathcal{D}(x))] - \frac{1}{2} \mathbb{E}_z [\log(1 - \mathcal{D}(\mathcal{G}(z)))] \\ \mathcal{L}_{\mathcal{G}} &= -\frac{1}{2} \mathbb{E}_z [\log(\mathcal{D}(\mathcal{G}(z)))]\end{aligned}$$

- Equilibrio no se describe con una sola pérdida
- Generador  $\mathcal{G}$  maximiza la probabilidad logarítmica de fallo del discriminador  $\mathcal{D}$

# Estrategia del discriminador $\mathcal{D}$

- Solución óptima del discriminador

$$\mathcal{D}(x) = \frac{P_{datos}(x)}{P_{datos}(x) + P_{modelo}(x)}$$



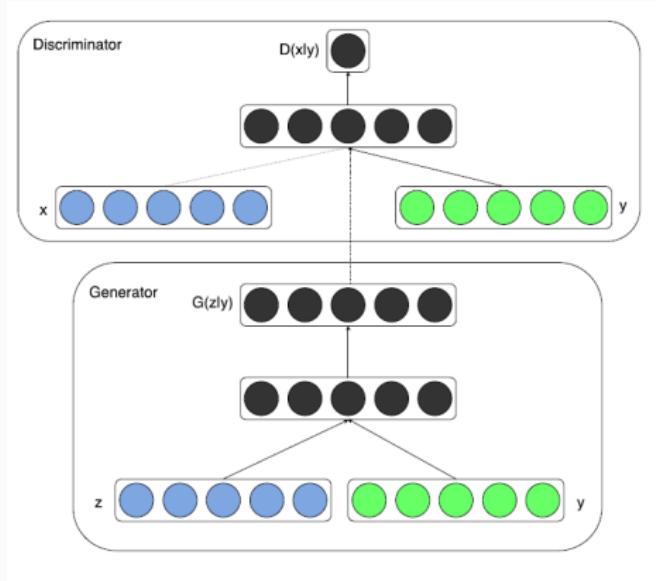
# Entrenamiento de GANs

- Descenso por el gradiente simultáneamente en:
  1. Mini-lote de ejemplos de entrenamiento
  2. Mini-lote de ejemplos generados
- Función de pérdida minimax

$$\min_{W_G} \max_{W_D} = [\mathbb{E}_{x \sim P_{\text{datos}}} [\log(D_{W_G}(x))] + \mathbb{E}_{z \sim P_z} [\log(1 - D_{W_G}(G_{W_G}(z)))]]$$

# GAN condicional

$$\min_{W_G} \max_{W_D} = [\mathbb{E}_{x \sim P_{\text{datos}}} [\log(\mathcal{D}_{W_G}(x|y))] + \mathbb{E}_{z \sim P_z} [\log(1 - \mathcal{D}_{W_G}(\mathcal{G}_{W_G}(z|y)))]]$$



# Red convolucional adversaria (DCGAN)

- Red con capas convolucionales transpuestas como  $\mathcal{G}$ 
  - Entrada: vector latente  $z \in \mathbb{R}^{100}$  muestreado de distribución uniforme
  - Salida: imagen generada a partir de  $z$

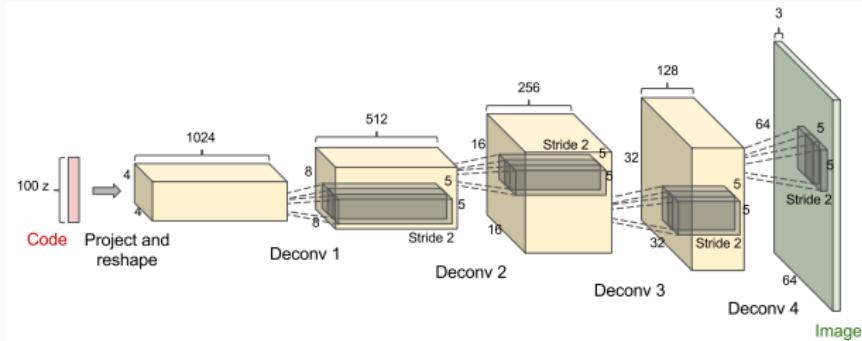


Figura tomada de <https://blog.openai.com/generative-models/>

# Red convolucional adversaria (DCGAN)

- Red con capas convolucionales transpuestas como  $\mathcal{G}$ 
  - Entrada: vector latente  $z \in \mathbb{R}^{100}$  muestreado de distribución uniforme
  - Salida: imagen generada a partir de  $z$

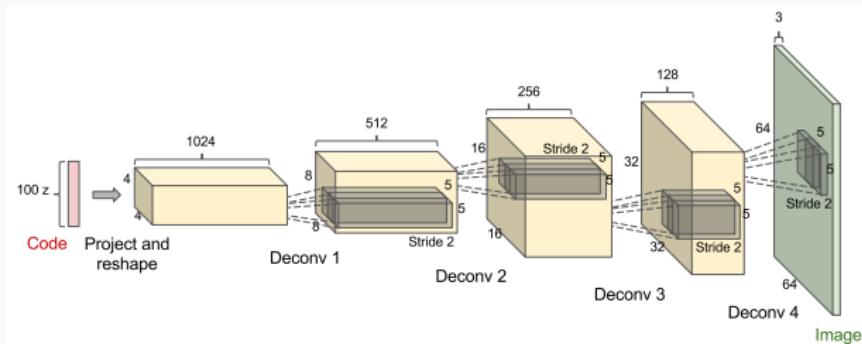


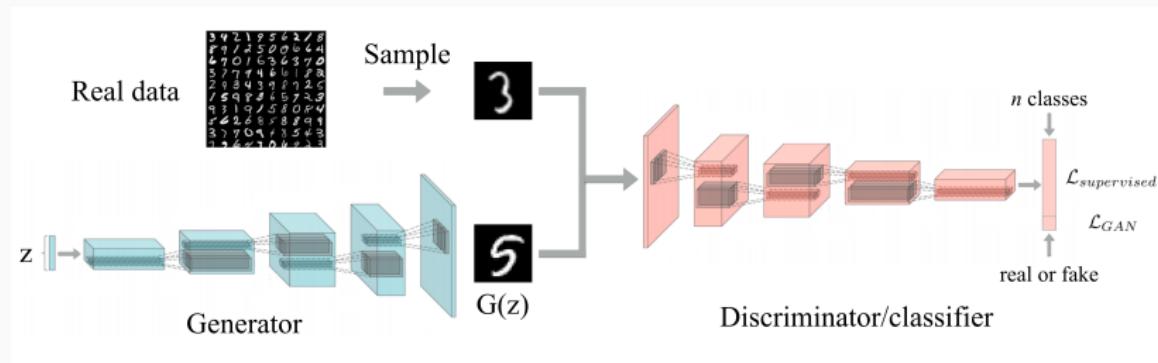
Figura tomada de <https://blog.openai.com/generative-models/>

- Objetivo: aprender a generar imágenes similares a las reales ( $P_{\text{modelo}}$ )

# GAN semi-supervisadas (1)

- Clasificación multi-clase con  $K$  clases
- Se asignan imágenes generadas a clase extra ( $K + 1$  clases)
- Pérdida

$$\begin{aligned}\mathcal{L} &= -\mathbb{E}_{\mathbf{x}, y \sim P_{\text{datos}}} [\log P_{\text{modelo}}(y|\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \mathcal{G}} [\log P_{\text{modelo}}(y = K+1|\mathbf{x})] \\ &= \mathcal{L}_{\text{supervisado}} - \mathcal{L}_{\text{nosupervisado}}\end{aligned}$$



## GAN semi-supervisadas (2)

- Mejora la calidad de las imágenes generadas



## GANs: trucos

- Usar etiquetas (aprendizaje semisupervisado)
- Etiquetas suaves y ruidosas (voltearlas algunas veces durante el entrenamiento)
- Normalizar valores y usar normalización por lotes
- Adam como optimizador
- Muestrear de distribución gaussiana  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
- Alternar entre datos reales y datos generados
- Leaky ReLU en lugar de ReLU
- Tangente hiperbólica en salida en generación de imágenes
- Evitar submuestreo y capas densas
- Usar Dropout

# Aplicaciones: traducción imagen a imagen



# Aplicaciones: traducción texto a imagen

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



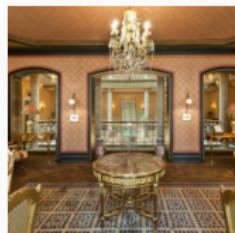
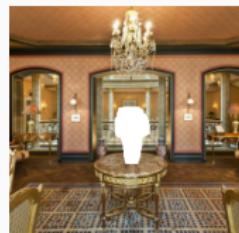
the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



# Aplicaciones: Edición de imágenes



# Aplicaciones: Generación de poses 3D

