



Generative Adversarial Networks

Josnick Chayña, Kelvin Puchó, y Angelo Pérez

Universidad Nacional de San Agustín

July 22, 2022

① Introducción

② La Arquitectura de las GANs

③ La Funcion Objetiva

④ Variantes

⑤ Entrenamiento de las GANS

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

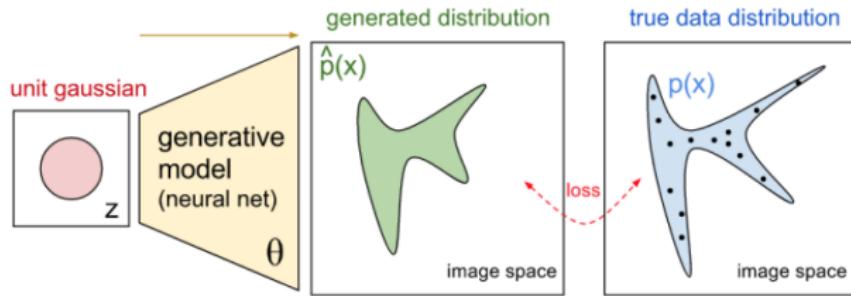
5 Entrenamiento de las GANS

¿Por qué los modelos generativos?

- Hasta ahora sólo hemos visto modelos discriminativos
 - Estima $P(Y|X)$
 - Dada una imagen X , predice una etiqueta Y
- Los modelos discriminativos tienen varias limitaciones importantes
 - No pueden modelar $P(X)$, es decir, la probabilidad de ver una determinada imagen
 - Por lo tanto, no pueden tomar muestras de $P(X)$, es decir, no pueden generar nuevas imágenes
- Los modelos generativos (en general) hacen frente a todo lo anterior
 - Pueden modelar $P(X)$
 - Pueden generar nuevas imágenes

Modelos generativos implícitos

- Los modelos generativos implícitos definen implícitamente una distribución de probabilidad
- Se empieza por muestrear el vector de códigos z a partir de una distribución fija y simple (gaussian distribution)
- La red generadora calcula una función diferenciable G que asigna z a una x en el espacio de datos.



Idea General

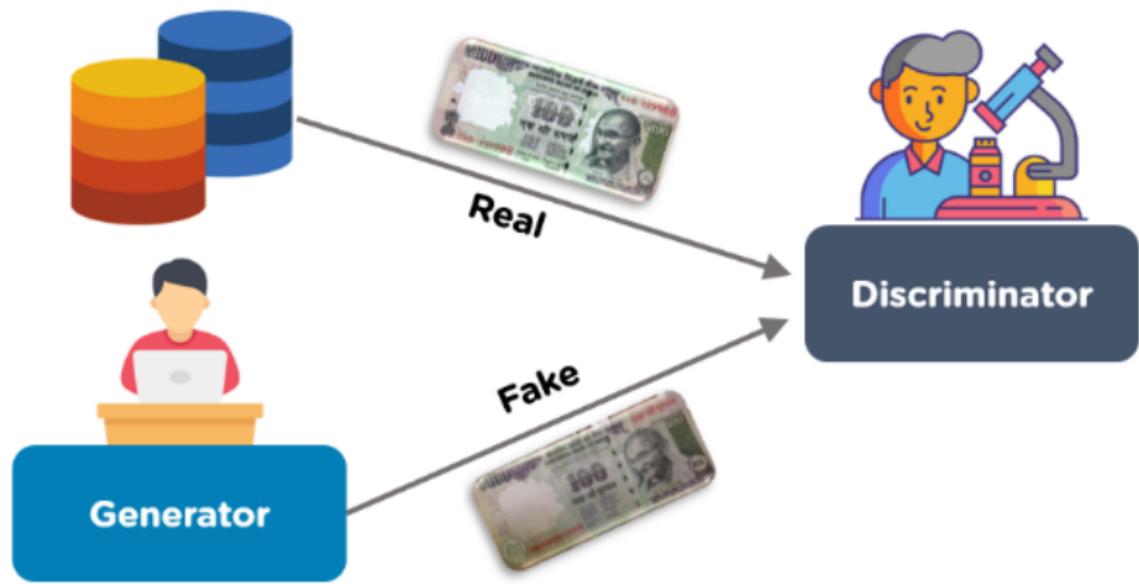


Figure 1: Esquema de funcionamiento de una red GAN

Generador

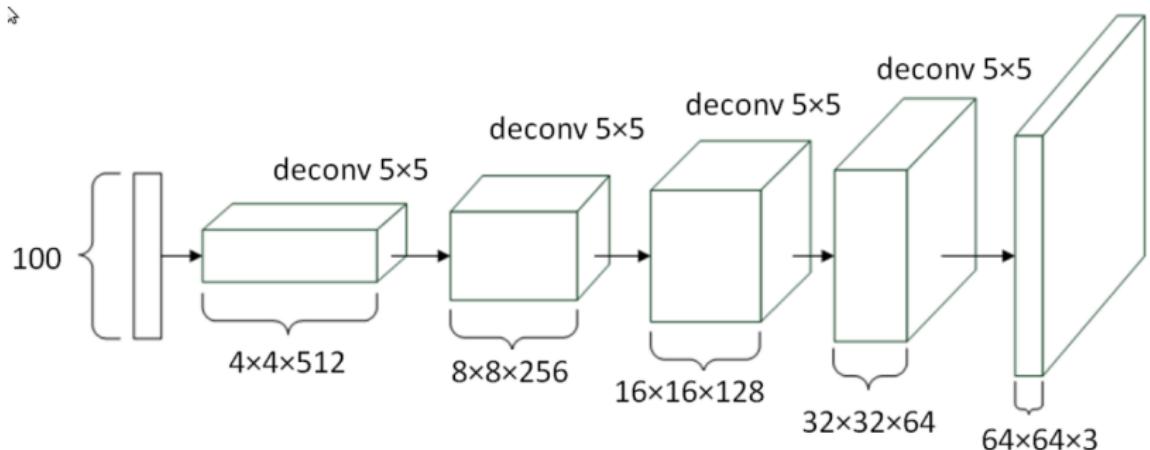


Figure 2: Zhu, Z.J. et al. (2018). Generator. [Figura 4]. Recuperado de [10.4236/jcsp.2018.93013](https://doi.org/10.4236/jcsp.2018.93013)

Discriminador

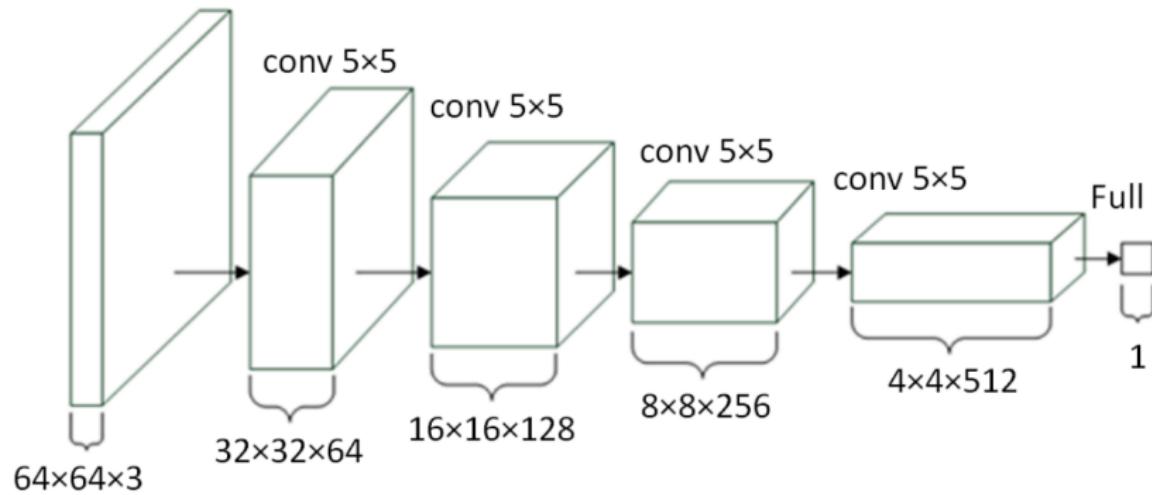


Figure 3: Zhu, Z.J. et al. (2018). Structure of discriminator. [Figura 5]. Recuperado de 10.4236/jcip.2018.93013

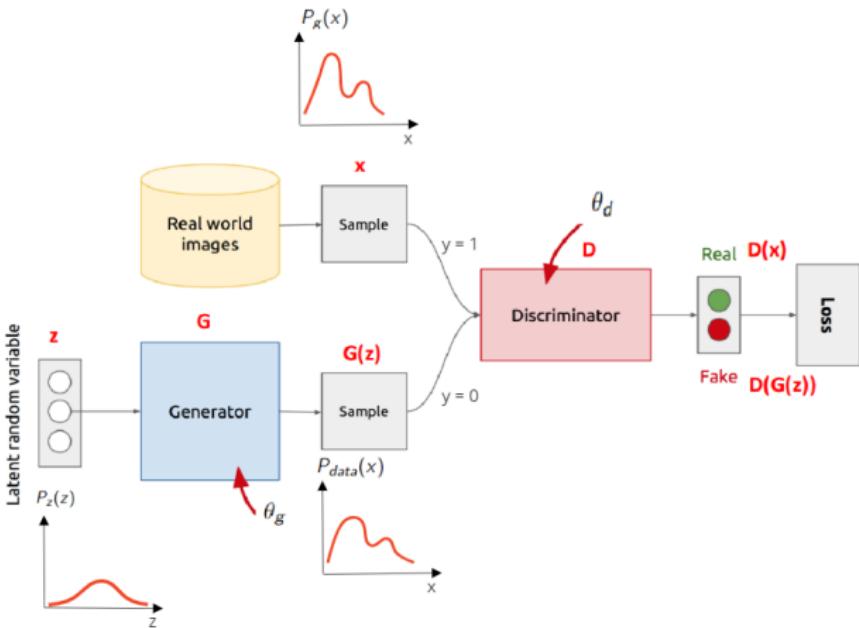
1 Introducción

2 La Arquitectura de las GANs

3 La Funcion Objetiva

4 Variantes

5 Entrenamiento de las GANS



- Z es un ruido aleatorio (Gaussian/Uniform).
- Z puede considerarse como la representación latente de la imagen.

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

Maximum likelihood game

Comparación

Problemas

4 Variantes

5 Entrenamiento de las GANS

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

Maximum likelihood game

Comparación

Problemas

4 Variantes

5 Entrenamiento de las GANS

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$

- Predicción del discriminador sobre los datos reales

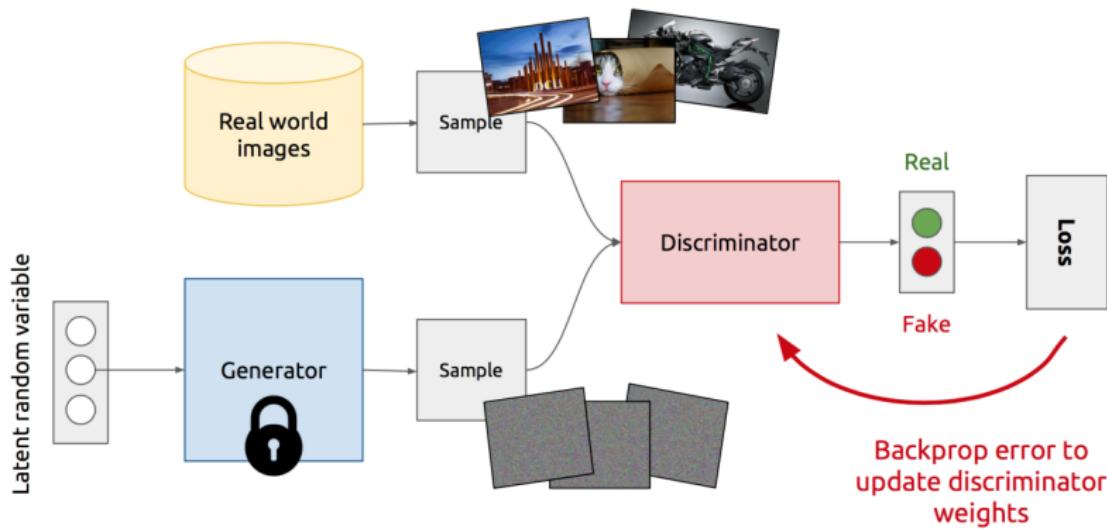
$$V(D, G) = \boxed{\mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)]} + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$

- Predicción del discriminador sobre los datos falsos

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \boxed{\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]}.$$

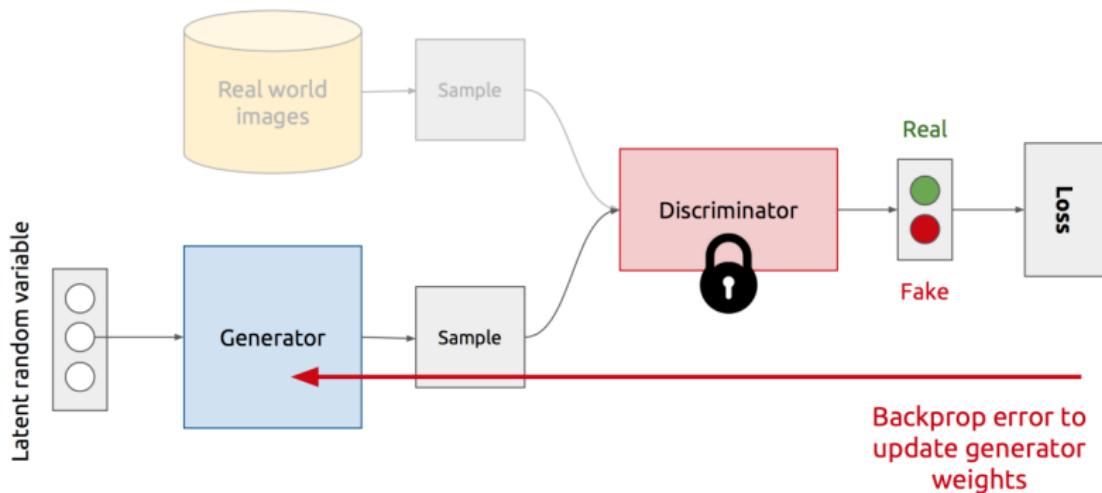
- Entrenando el Discriminador

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$



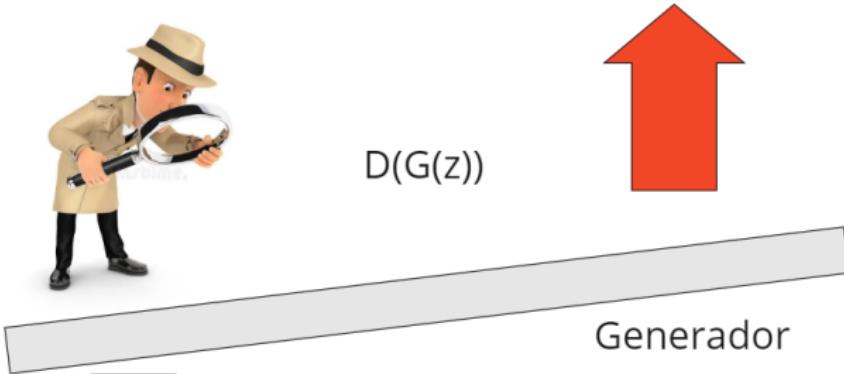
- Entrenando el Generador

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$



$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Discriminador



Generador



$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$

Generador

 $1 - D(G(z))$ 

Discriminador



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

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

end for

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

Discriminator
updates

Generator
updates

Figure 4: Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets.

Which one is Computer generated?

$$\min_G \max_D V(D, G)$$



donde: Para un generador fijo, el máximo de la función de costo con respecto al discriminador es una constante

$$V(D^*, G) = -\log(4)$$

Para un G fijo:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

$$V(G, D) = \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) d\mathbf{z}$$

$$= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x}$$



$$a \log(y) + b \log(1 - y)$$

$$\frac{\partial}{\partial y} \ a \log(y) + b \log(1 - y) = 0$$

$$\frac{a}{y} + \frac{b}{1-y}(-1) = 0 \quad \Rightarrow \quad \frac{a}{y} = \frac{b}{1-y}$$

$$a - ay = by \quad \Rightarrow \quad a = ay + by = (a + b)y$$

$$\Rightarrow \quad y = \frac{a}{a+b}$$

$$\therefore D(x)^* = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\log \left(\frac{1}{2} \right) \right] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \left[\log \left(\frac{1}{2} \right) \right]$$

$$= -\log(2) - \log(2)$$

$$= -\log(4)$$

$$D(x)^* = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

for $p_{data}(x) = p_g(x)$

$$D(x)^* = \frac{p_{data}(x)}{2p_{data}(x)} = \frac{1}{2}$$

Después de fijar $D(x)$:

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D_G^*(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D_G^*(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{P_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g} \left[\log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] \end{aligned}$$

La definición de la divergencia KullbackLeibler (KL) y la divergencia Jensen-Shannon (JS) entre dos distribuciones probabilísticas $p(x)$ y $q(x)$ se definen como:

$$KL(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx,$$

$$JS(p\|q) = \frac{1}{2}KL(p\| \frac{p+q}{2}) + \frac{1}{2}KL(q\| \frac{p+q}{2}).$$

Entonces reemplazamos la ecuación:

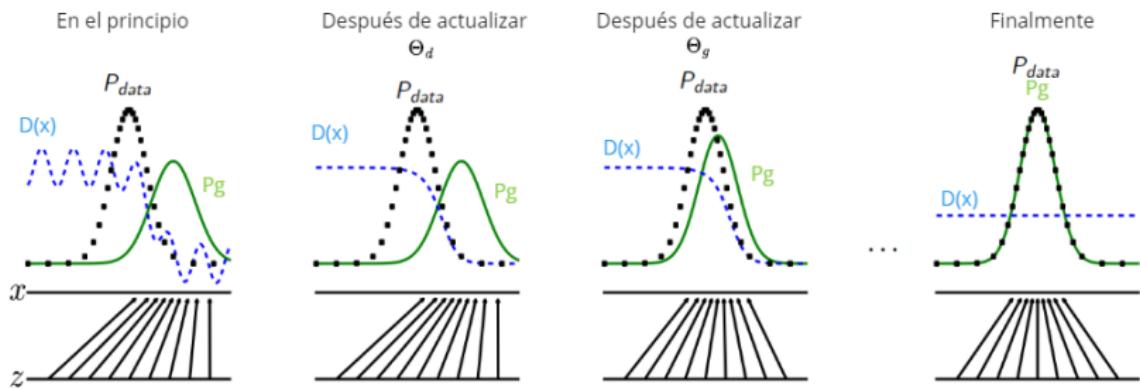
$$\begin{aligned} C(G) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D_G^*(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{P_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g} \left[\log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] \\ &= E_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{\frac{1}{2}(p_{\text{data}}(x) + p_g(x))} \right] \\ &\quad + E_{x \sim p_g} \left[\frac{p_g(x)}{\frac{1}{2}(p_{\text{data}}(x) + p_g(x))} \right] - 2 \log 2. \\ &= KL(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2}) + KL(p_g \parallel \frac{p_{\text{data}} + p_g}{2}) \\ &\quad - 2 \log 2 \\ &= 2JS(p_{\text{data}} \parallel p_g) - 2 \log 2. \end{aligned}$$

$$\min_G V(D, G) = 2JS(p_{data} \| p_g) - 2 \log 2.$$

$\Rightarrow p_{data}(x) = p_g(x)$

$$\min_G V(D, G) = -\log(4)$$

- Ciclo de entrenamiento



- Enfoque del Mini-max Game

Nash Equilibrium / Saddle Point

$$\frac{\partial J^D}{\partial D(X)} = 0 \rightarrow D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{model}(x)}$$

$$p_{data(x)} = P_{model}(x) \quad \forall x$$

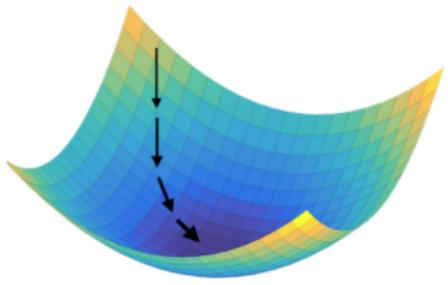
$$D^*(x) = \frac{1}{2} \quad \forall x$$

- Enfoque del Mini-max Game

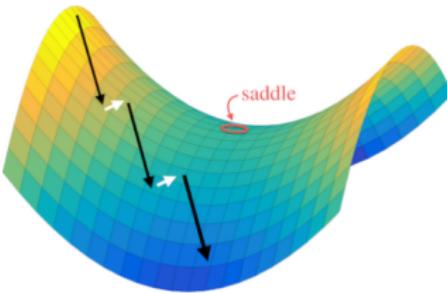
- Las redes neuronales adversarias resolvieron muchos problemas a lo largo de su existencia,
- Son notoriamente difíciles de entrenar.
- Estas dificultades provienen del hecho de que los pesos óptimos para las redes adversariales corresponden a puntos de silla, y no a minimizadores, de la función de pérdida.
- Los métodos de gradiente estocástico alterno que se suelen utilizar para este tipo de problemas no convergen de forma fiable a los puntos de equilibrio
- Y si convergen suelen ser muy sensibles a las tasas de aprendizaje.

- Enfoque del Mini-max Game

Standard Neural Net



Adversarial Net



- Las redes neuronales estándar son estables porque la ruta de entrenamiento se "atasca" en un minimizador de la función de pérdida.
- Los métodos de entrenamiento adversarios alternan entre pasos de minimización y maximización para converger a un punto de silla de la función de pérdida. Esto es potencialmente inestable porque los iterados se "deslizarán" por el lado de la silla si la minimización supera a la maximización (o viceversa).

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

Maximum likelihood game

Comparación

Problemas

4 Variantes

5 Entrenamiento de las GANS

- Es posible que la ecuación original del minimax game no pueda proporcionar suficiente gradiente para que G aprenda bien en la práctica.
- En general, G es pobre en el aprendizaje temprano y las muestras son claramente diferentes de los datos de entrenamiento. Por lo tanto, D puede rechazar las muestras generadas con alta confianza.
- En esta situación, $\log(1 - D(G(z)))$ se satura. Podemos entrenar a G para maximizar $\log(D(G(z)))$ en lugar de minimizar $\log(1 - D(G(z)))$.

$$\begin{aligned} J^{(G)} &= E_{z \sim p_z(z)} [-\log(D(G(z)))] \\ &= E_{x \sim p_g} [-\log(D(x))]. \end{aligned}$$

$$\begin{aligned} &E_{x \sim p_g} [-\log(D_G^*(x))] \\ &= KL(p_g \| p_{data}) - 2JS(p_{data} \| p_g) + \\ &E_{x \sim p_{data}} [\log D_G^*(x)] + 2 \log 2. \end{aligned}$$

- Esta función de pérdida es contradictoria porque el primer término pretende que la divergencia entre la distribución generada y la distribución real sea lo más pequeña posible mientras que el segundo término pretende que la divergencia entre estas dos distribuciones sea lo más grande posible debido al signo negativo.
- Esto traerá un gradiente numérico inestable para el entrenamiento de G .
- La divergencia KL no es una cantidad simétrica, lo que se refleja en los siguientes dos ejemplos
 - If $p_{data}(x) \rightarrow 0$ and $p_g(x) \rightarrow 1$, we have $KL(p_g \| p_{data}) \rightarrow +\infty$.
 - If $p_{data}(x) \rightarrow 1$ and $p_g(x) \rightarrow 0$, we have $KL(p_g \| p_{data}) \rightarrow 0$.

- Las penalizaciones para dos errores cometidos por G son completamente diferentes.
- El primer error es que G produce muestras casi reales que son difícil de que el D caiga y la penalización es bastante grande.
- El segundo error es que G no produce muestras reales y la penalización es bastante pequeña.
- El primer error es que las muestras generadas son inexactas, mientras que el segundo error es que las muestras generadas no son lo suficientemente diversas.

Por lo tanto G prefiere producir muestras repetidas pero seguras en lugar de arriesgarse a producir muestras diferentes pero inseguras, lo que tiene el problema del colapso de modo.



1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

Maximum likelihood game

Comparación

Problemas

4 Variantes

5 Entrenamiento de las GANS

Hay muchos métodos para aproximar en GANs. Bajo el supuesto de que el discriminador es óptimo, minimizar.

$$\begin{aligned} J^{(G)} &= E_{z \sim p_z(z)} [-\exp(\sigma^{-1}(D(G(z)))]) \\ &= E_{z \sim p_z(z)} [-D(G(z))/(1 - D(G(z)))] , \end{aligned}$$

donde es la función sigmoidea logística,



1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

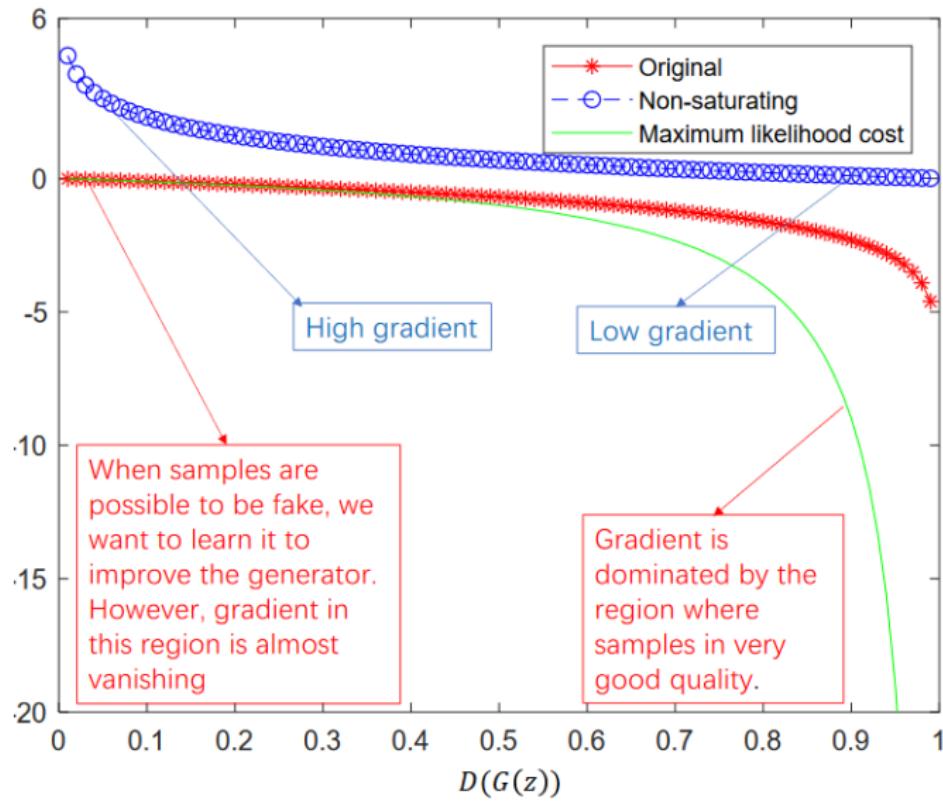
Maximum likelihood game

Comparación

Problemas

4 Variantes

5 Entrenamiento de las GANS



- Cuando la muestra puede ser falsa, es decir, el extremo izquierdo de la figura, tanto maximum likelihood game and the original minimax game sufren la desaparición del gradiente. El Non-saturating game no tiene este problema.
- El maximum likelihood game también tiene el problema de que casi todo el gradiente procede del extremo derecho de la curva, lo que significa que un número bastante pequeño de muestras en cada minibatch domina el cálculo del gradiente.
- El non-saturating game tiene una menor varianza de las muestras, que es la posible razón por la que tienes más éxito en las aplicaciones reales,



1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

Original minimax game

Non-saturating game

Maximum likelihood game

Comparación

Problemas

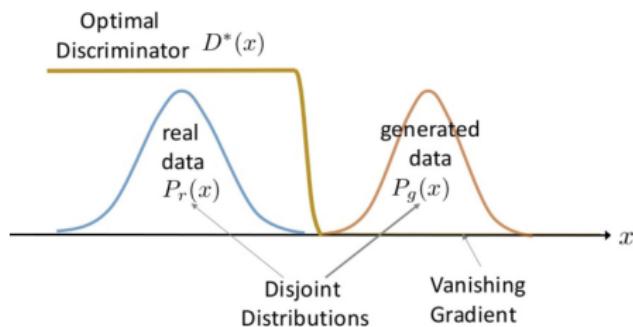
4 Variantes

5 Entrenamiento de las GANS

Problema del desvanecimiento de la gradiente con generador

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \boxed{\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]}.$$

Surge si el gradiente va a 0 si D es seguro, es decir, $D(G(z)) \rightarrow 0$



- Siempre que el discriminador se vuelve muy seguro, el valor de valor de la pérdida será cero.
- Nada que mejorar para Generador

¿Por qué los GAN son difíciles de entrenar?

-Non-Convergence

- A medida que el generador mejora con el entrenamiento, el rendimiento del discriminador empeora porque éste no puede distinguir fácilmente entre lo real y lo falso.
- Si el generador acierta perfectamente, el discriminador tiene una precisión del 50%. En efecto, el discriminador lanza una moneda al aire para hacer su predicción.
- Esta progresión plantea un problema para la convergencia de la GAN en su conjunto: la retroalimentación del discriminador se vuelve menos significativa con el tiempo. Si el GAN continúa entrenando más allá del punto en el que el discriminador está dando una respuesta completamente aleatoria, entonces el generador comienza a entrenar con una respuesta basura, y su propia calidad puede colapsar.

- Mode Collapse

- Si un generador produce una salida especialmente plausible, el generador puede aprender a producir sólo esa salida. De hecho, el generador siempre está intentando encontrar la salida que parece más plausible para el discriminador.
- Si el generador empieza a producir la misma salida (o un pequeño conjunto de salidas) una y otra vez, la mejor estrategia del discriminador es aprender a rechazar siempre esa salida. Pero si la siguiente generación del discriminador se atasca en un mínimo local y no encuentra la mejor estrategia, entonces es muy fácil que la siguiente iteración del generador encuentre la salida más plausible para el discriminador actual.
- Cada iteración del generador sobreoptimiza para un discriminador particular, y el discriminador nunca consigue aprender a salir de la trampa. Como resultado, los generadores rotan a través de un pequeño conjunto de tipos de salida.

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

GAN basadas en la optimización de la arquitectura

GAN basadas en la optimización de funciones objetivas

Style GAN

5 Entrenamiento de las GANS

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

GAN basadas en la optimización de la arquitectura

GAN basadas en la optimización de funciones objetivas

Style GAN

5 Entrenamiento de las GANS

Convolution Based GANs

Las GAN originales adoptan el perceptrón multicapa (MLP) para que funcione. Debido al hecho de que CNN es mejor que MLP en la extracción de características de imagen, en 2016 propusieron una red antagónica generativa convolucional profunda (DCGAN). Este enfoque reemplazó de manera innovadora la capa full connected con la capa de deconvolución en el generador, que logró un gran rendimiento en las tareas de generación de imágenes.

Condition Based GANs

Conditionals GAN (CGANs)

Introdujo la variable condicional c (la variable c puede ser etiquetas, texto u otros datos) tanto en el generador como en el discriminador para agregar condiciones al modelo usando información adicional para afectar el proceso de generación de datos.

InfoGAN

Al introducir información mutua, InfoGAN hace que el proceso de generación sea más controlable y los resultados se pueden interpretar mejor. Además del discriminador de las GAN originales, InfoGAN tiene una red Q adicional para generar las variables condicionales $Q(c|x)$

Condition Based GANs

Auxiliary Classifier GAN (ACGAN)

La variable de condición c no se agregará y se usará otro clasificador para mostrar la probabilidad sobre las etiquetas de clase. A continuación, se modifica la función de pérdida para aumentar la probabilidad de una predicción de clase correcta.

Condition Based GANs

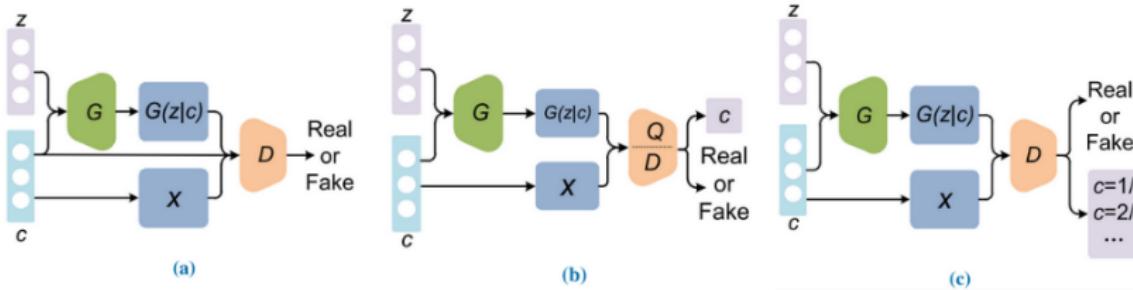


Figure 5:

Autoencoder Based GANs

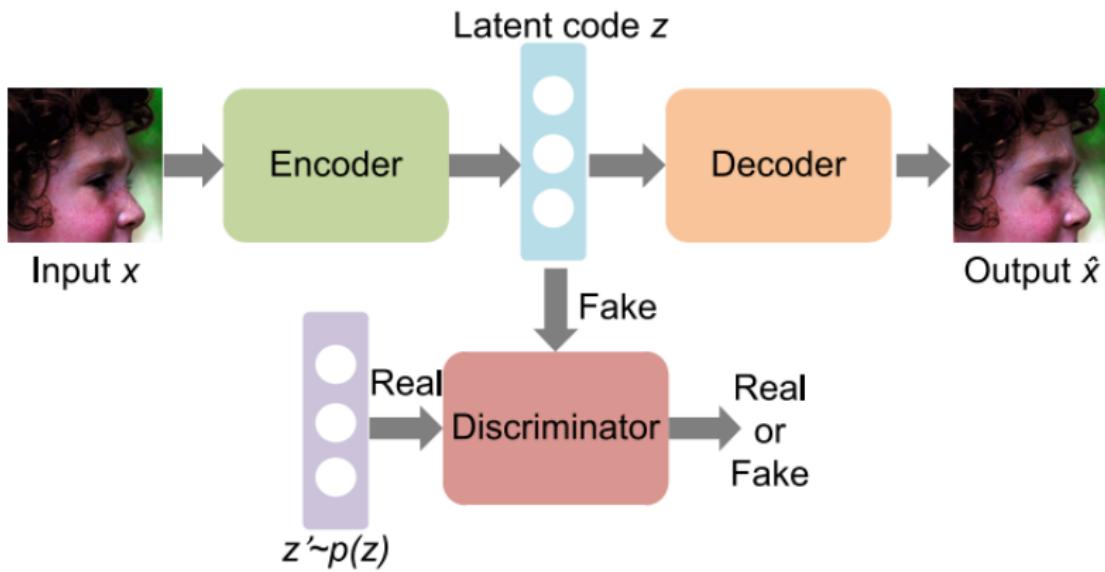


Figure 6:

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

GAN basadas en la optimización de la arquitectura

GAN basadas en la optimización de funciones objetivas

Style GAN

5 Entrenamiento de las GANS

Unrolled GANs

Utiliza una función de pérdida basada en gradientes para mejorar el generador. Y las GAN originales se lograron minimizando la divergencia de Jensen-Shannon (JS) para minimizar la función de pérdida del generador.

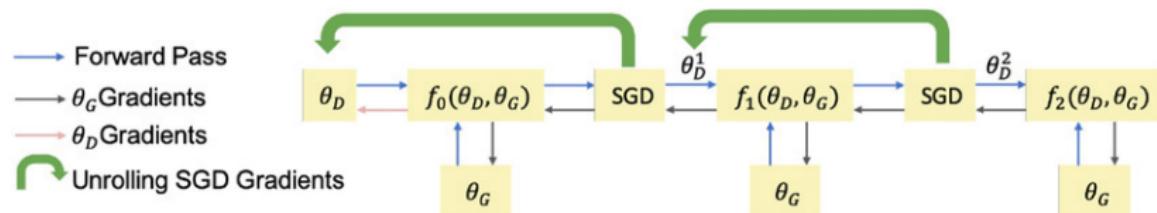


Figure 7:

Loss-Sensitive GAN

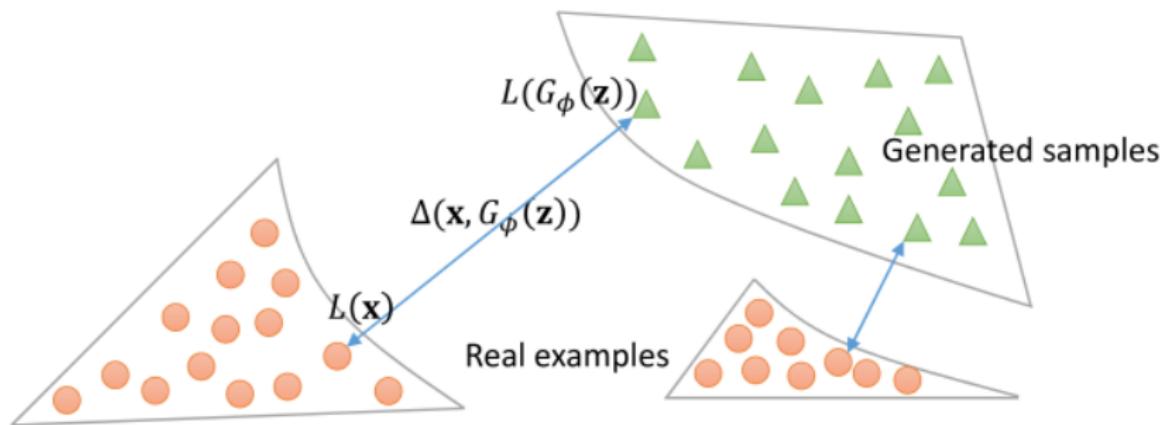


Figure 8:

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

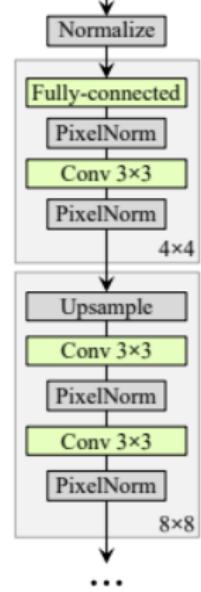
GAN basadas en la optimización de la arquitectura

GAN basadas en la optimización de funciones objetivas

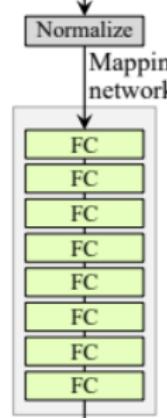
Style GAN

5 Entrenamiento de las GANS

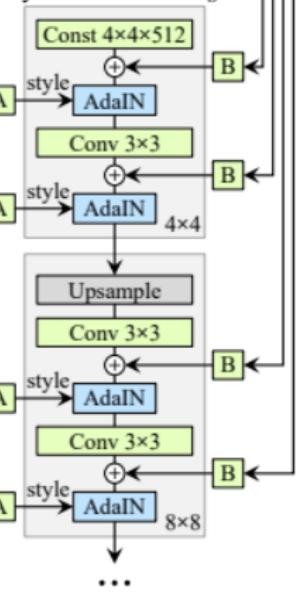
Style GAN - Arquitectura

Latent $\mathbf{z} \in \mathcal{Z}$ 

(a) Traditional

Latent $\mathbf{z} \in \mathcal{Z}$ 

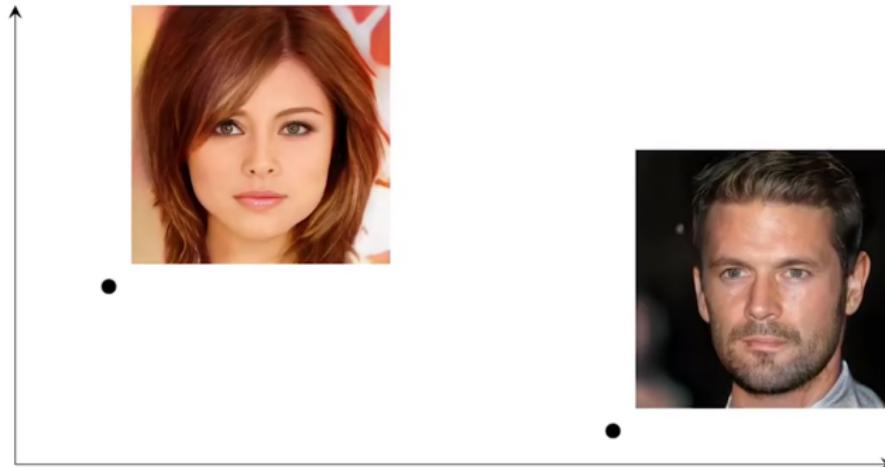
(b) Style-based generator

Synthesis network g 

Style GAN - Mapping



Latent space interpolation

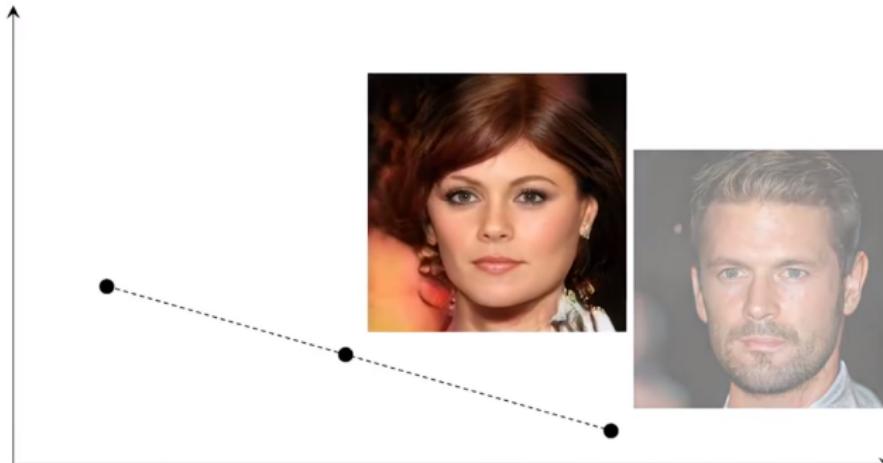


44

Style GAN - Mapping



Latent space interpolation



44

Style GAN - Resultado



1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

5 Entrenamiento de las GANS

Función Objetiva

Skills

Estructura

Aplicaciones

A pesar de la existencia teórica de soluciones únicas, el entrenamiento de GAN es duro y, a menudo, inestable por varias razones. Una dificultad proviene del hecho de que los pesos óptimos para las GAN corresponden a los puntos de silla, y no a los minimizadores, de la función de pérdida

- Yadav estabilizar una GAN con métodos de predicción Mediante el uso de tasas de aprendizaje independientes.
- Regla de actualización de dos escalas de tiempo (TTUR) tanto para el discriminador como para el generador para garantizar que el modelo pueda converger a un equilibrio de Nash local estable

- Un método para mejorar el entrenamiento de GAN es evaluar los "síntomas" empíricos que pueden ocurrir en el entrenamiento.
 - Colapso del modelo generativo para producir muestras muy similares para diversas entradas
 - La pérdida del discriminador converge rápidamente a cero
 - Dificultades para hacer converger el par de modelos

Por eso se presentan tres perspectivas: función objetiva, habilidades y estructura.

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

5 Entrenamiento de las GANS

Función Objetiva

Least squares generative adversarial networks (LSGANs)

Energy-based generative adversarial network (EBGAN)

Boundary equilibrium generative adversarial networks (BEGAN):

Mode regularized generative adversarial networks (MDGAN):

Spectrally normalized GANs (SN-GANs):

Skills

Estructura

- Desaparición del gradiente para entrenar G
- En un juego sin saturación obtendrá el problema del colapso del modo

Estos problemas son causados por la función objetivo y no pueden resolverse cambiando las estructuras de las GAN.

Rediseñar la función objetivo es una solución natural para mitigar estos problemas. Sobre la base de las fallas teóricas de las GAN, se han propuesto muchas variantes basadas en la función objetivo para cambiar la función objetivo de las GAN en función de análisis teóricos como las redes antagónicas generativas de mínimos cuadrados.

Least squares generative adversarial networks (LSGANs)

Se proponen LSGAN, para superar el problema del gradiente de desaparición en las GAN originales. Los LSGAN adoptan la pérdida por mínimos cuadrados en lugar de la pérdida de entropía cruzada en los GAN originales. Supongamos que se utiliza la codificación a-b para el discriminador de LSGAN, donde a y b son las etiquetas para la muestra generada y la muestra real, respectivamente. La pérdida del discriminador VLSGAN (D) y la pérdida del generador VLSGAN (G) de los LSGAN se definen como:

$$\begin{aligned} \min_D V_{LSGAN}(D) = & E_{x \sim p_{data}(x)} [(D(x) - b)^2] \\ & + E_{z \sim p_z(z)} [(D(G(z)) - a)^2], \end{aligned}$$

$$\min_G V_{LSGAN}(G) = E_{z \sim p_z(z)} [(D(G(z)) - c)^2],$$

donde c es el valor que G espera que D crea para las muestras generadas. Hay dos ventajas de las LSGAN en comparación con las GAN originales:

- El nuevo límite de decisión producido por D penaliza un gran error en aquellas muestras generadas que están lejos del límite de decisión, lo que hace que esas muestras generadas de "baja calidad" se muevan hacia el límite de decisión. Esto es bueno para generar muestras de mayor calidad.
- La penalización de las muestras generadas lejos del límite de decisión puede proporcionar más gradiente al actualizar la G , lo que supera los problemas de gradiente de desaparición en las GAN originales

Energy-based generative adversarial network (EBGAN):

El discriminador de EBGAN se ve como una función de energía, que otorga alta energía a las muestras falsas ("generadas") y menor energía a las muestras reales. Dado un margen m positivo, las funciones de pérdida para EBGAN se pueden definir de la siguiente manera:

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+,$$

$$\mathcal{L}_G(z) = D(G(z)),$$

donde $[y]_+ = \max(0, y)$ es la función de la unidad lineal rectificada (ReLU). Tenga en cuenta que en las GAN originales, el discriminador D otorga una puntuación alta a las muestras reales y una puntuación baja a las muestras generadas ("falsas"). Sin embargo, el discriminador en EBGAN atribuye baja energía (puntuación) a las muestras reales y mayor energía a las generadas. EBGAN tiene un comportamiento más estable que las GAN originales durante el entrenamiento

Boundary equilibrium generative adversarial networks

(BEGAN): BEGAN también usa un autoencoder para discriminar. Utilizando la teoría de control proporcional, BEGAN propone un método de equilibrio novedoso para equilibrar el generador y el discriminador en el entrenamiento, que es rápido, estable y resistente a los cambios de parámetros.

Mode regularized generative adversarial networks (MDGAN):

El entrenamiento inestable y el colapso del modelo de las GAN se debe a la forma funcional muy especial de los discriminadores entrenados en espacios de alta dimensión, lo que puede hacer que el entrenamiento se atasque o empuje la masa de probabilidad en la dirección equivocada, hacia una concentración más alta que esa. de la distribución de datos reales.

La idea clave de MDGAN es utilizar un encoder $E(x)$: $x \rightarrow z$ para producir la variable latente z para el generador G en lugar de utilizar ruido. Este procedimiento tiene dos ventajas:

- El encoder garantiza la correspondencia entre $z (E(x))$ y x , lo que hace que G sea capaz de cubrir diversos modos en el espacio de datos. Por lo tanto, previene el problema del colapso del modo.
- Debido a que la reconstrucción del codificador puede agregar más información al generador G , no es fácil para el discriminador D distinguir entre las muestras reales y las generadas.

La función de pérdida para el generador y el encoder de MDGAN es:

$$\mathcal{L}_G = -E_{z \sim p_z(z)} [\log (D (G (z)))] \\ + E_{x \sim p_{data}(x)} \left[\begin{array}{l} \lambda_1 d (x, G \circ E (x)) \\ + \lambda_2 \log D (G \circ E (x)) \end{array} \right],$$

$$\mathcal{L}_E = E_{x \sim p_{data}(x)} \left[\begin{array}{l} \lambda_1 d (x, G \circ E (x)) \\ + \lambda_2 \log D (G \circ E (x)) \end{array} \right],$$

donde tanto λ_1 como λ_2 son parámetros de ajuste libre, d es la métrica de distancia, como la distancia euclídea, y $G \circ E (x) = G (E (x))$

Spectrally normalized GANs (SN-GANs):

Las SN-GAN proponen un nuevo método de normalización de peso llamado normalización espectral para hacer estable el entrenamiento del discriminador. Esta nueva técnica de normalización es computacionalmente eficiente y fácil de integrar en los métodos existentes. La normalización espectral utiliza un método simple para hacer que la matriz de peso W satisfaga la restricción de Lipschitz $\sigma(W) = 1$.

$$\bar{W}_{SN}(W) := W/\sigma(W),$$

donde W es la matriz de peso de cada capa en D y $\sigma(W)$ es la norma espectral de W . Se demuestra que [23] las SN-GAN pueden generar imágenes de igual o mejor calidad en comparación con los métodos de estabilización de entrenamiento anteriores. En teoría, la normalización espectral se puede aplicar a todas las variantes de GAN. Tanto BigGANs como SAGAN utilizan la normalización espectral y tienen buenos rendimientos en Imagenet

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

5 Entrenamiento de las GANS

Función Objetiva

Skills

Estructura

Aplicaciones

Técnicas muy útiles y mejoradas para entrenar GANs:

- Coincidencia de características
- Discriminación de minibatches
- Promedio histórico
- Suavizado de etiquetas unilaterales

1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

5 Entrenamiento de las GANS

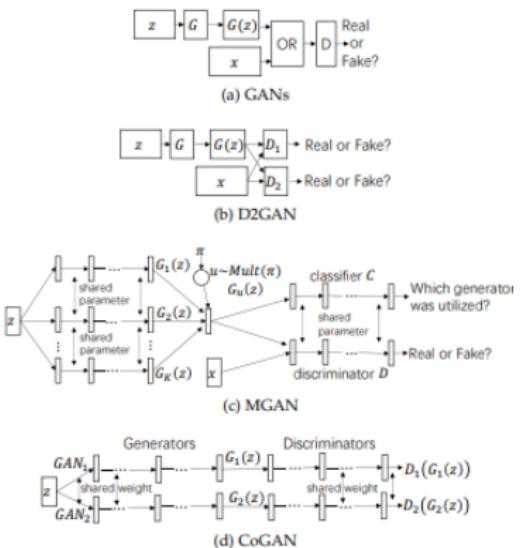
Función Objetiva

Skills

Estructura

Aplicaciones

- GANs originales
Utilizaban perceptrón multicapa (MLP)
 - Red neuronal recurrente (RNN)
 - Red neuronal convolucional (CNN)
- Redes adversariales generativas laplacianas (LAPGAN) y SinGAN:
 - LAPGAN se propone para producir imágenes de mayor resolución
 - SinGAN aprende un modelo generativo a partir de una sola imagen natural
- Multidiscrimination Learning
- Multi-generator learning
- Multi-GAN learning



1 Introducción

2 La Arquitectura de las GANs

3 La Función Objetiva

4 Variantes

5 Entrenamiento de las GANS

Función Objetiva

Skills

Estructura

Aplicaciones



Text to Image Synthesis



Face Aging

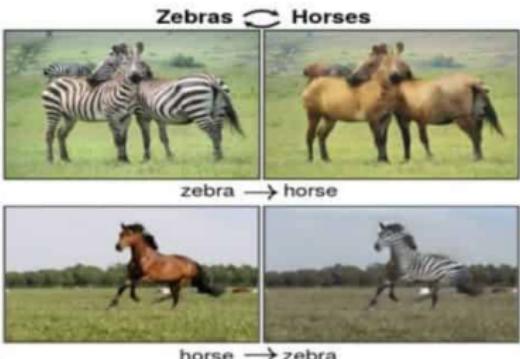


Image to Image Translation



Super Resolution



GRACIAS!!