Redes generativas adversariales (GAN)

Aprendizaje profundo

Alberto Díaz Álvarez, Edgar Talavera Muñoz y Guillermo Iglesias Hernández
Departamento de Sistemas Informáticos - Universidad Politécnica de Madrid

17 de abril de 2023

*The **most interesting idea in the last ten years** in machine learning.*

- Yann LeCun - Sobre las redes GAN -

Limitaciones de los *variational autoencoders*

Anteriormente vimos que los variational autoencoders (VAE):

- Son capaces de generalizar gracias al uso de una función de densidad
- Pueden generar datos _indistinguibles de los de entrada
- Pueden generar nuevos datos sin copiar directamente los ejemplos

Pero, como cualquier otro modelo, tienen sus desventajas:

- Poca calidad en los datos generados, lo que limita su uso en aplicaciones reales
- Poca diversidad en los datos generados por depender, entre otros, sólo de la función de densidad aprendida

Estas desventajas son las que motivan la aparición de modelos para paliarlas

Retrospectiva

Durante un entrenamiento tradicional, tenemos el siguiente proceso:

1. Se infiere una salida $\hat{y}$ a partir de una entrada X
2. Se calcula el *loss* $\mathcal{L}$ entre la salida $\hat{y}$ y la salida deseada y
3. Usando retropropagación, se actualizan los pesos de la red para minimizar el *loss*

Este proceso es sencillo, ya que toda las funciones están fijadas previamente

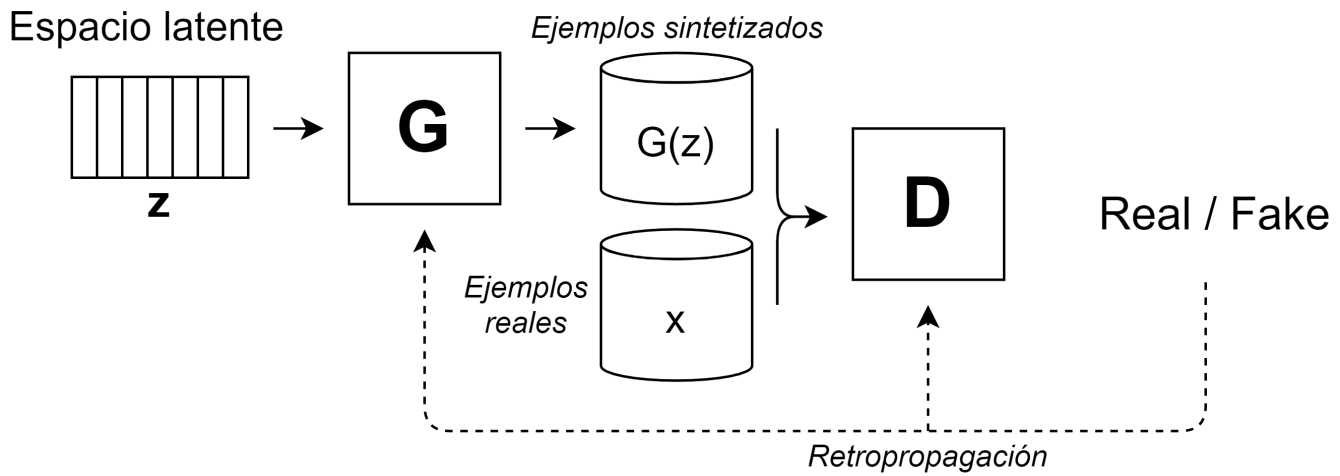
- Pero, ¿y si su loss es **inferido por la predicción de otro modelo**?

Pues eso son, a grandes rasgos, los modelos generativos adversariales

Esquema general

En los modelos generativos adversariales (GAN) se trabaja con un modelo compuesto de dos redes

- Complica un poco el modelo, pero los resultados producidos son asombrosos.



Durante su entrenamiento se produce una competición entre las redes **G** y **D**

- En ella ambos modelos mejoran progresivamente de manera simultánea

Ejemplo: Falsificador vs. policía (I)

- Se quiere hacer falsificaciones de billetes
- Se busca que la policía sea incapaz de distinguir entre billetes falsos y reales



Ejemplo: Falsificador vs. policía (II)

Falsificador

Quiere engañar con billetes falsos

- Es un excelente artista
- Es capaz de generar nuevos billetes
- No sabe qué es un billete

Policía

Quiere separar los billetes falsos

- Discrimina entre billetes verdaderos y falsos
- Tampoco sabe qué es un billete

Ejemplo: Falsificador vs. policía (III)

El falsificador:

- Intenta engañar al policía generando billetes que se parezcan a los reales
- Si engaña al policía sabe que ese billete es de buena calidad

Ejemplo: Falsificador vs. policía (IV)

El policía:

- Intenta no ser engañado discriminando las diferencias más sutiles entre los billetes
- Si es engañado aprende cuáles son esos detalles que hacen que el billete sea falso

Ejemplo: Falsificador vs. policía (y V)

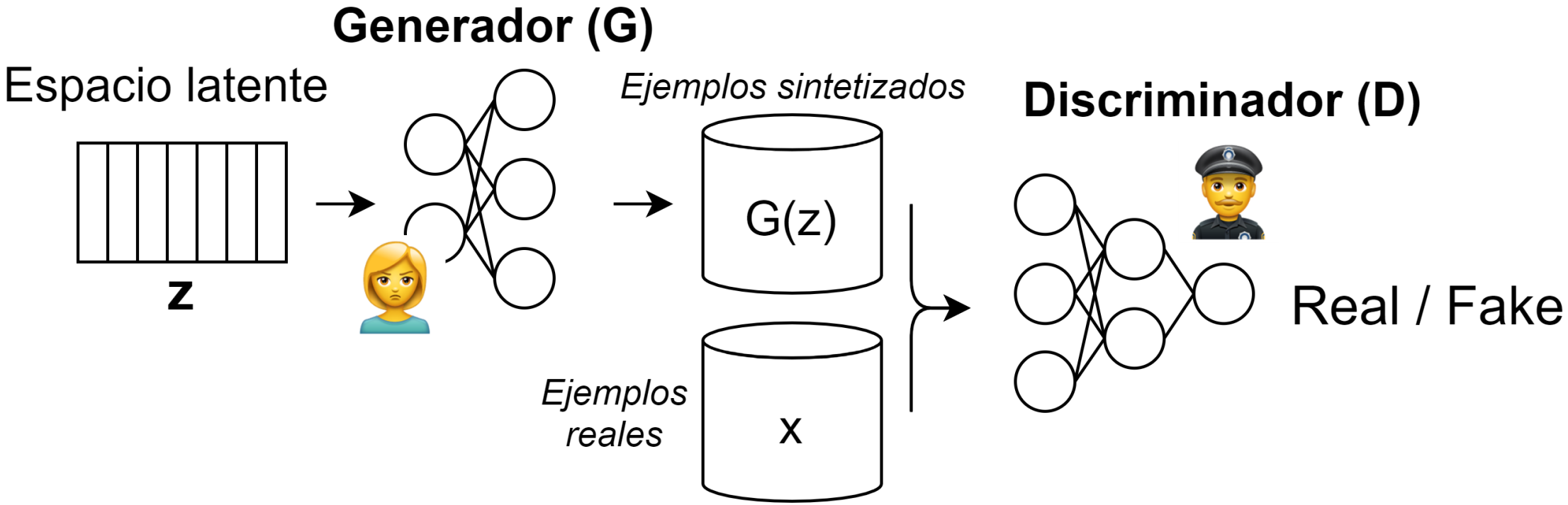
El policía y el falsificador **compiten** por quien consigue vencer al otro

- Según pasa el tiempo los billetes del falsificador son **más realistas**
- Eso sí, el policía aprende a **diferenciar mejor** los pequeños detalles

Esta competición hace que ambos mejoren de manera **simultánea y paralela**

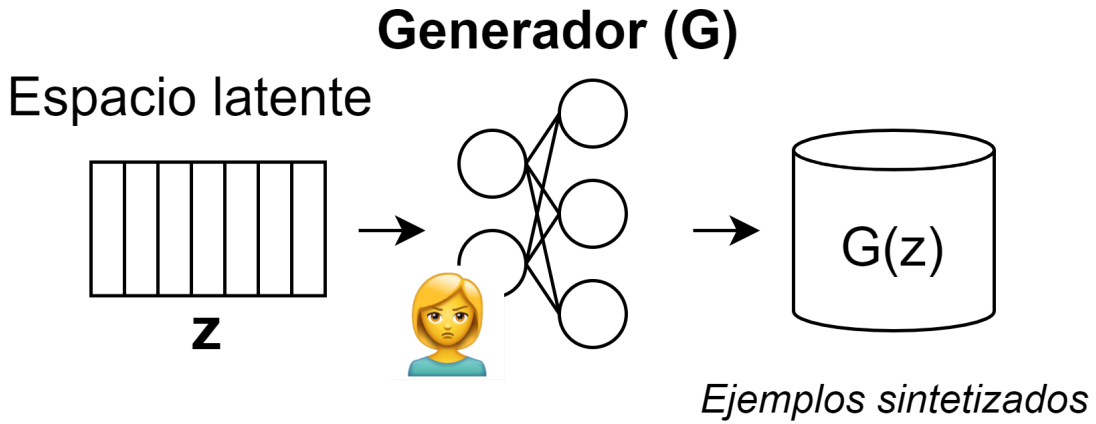


Esquema general de una red GAN



Red generadora

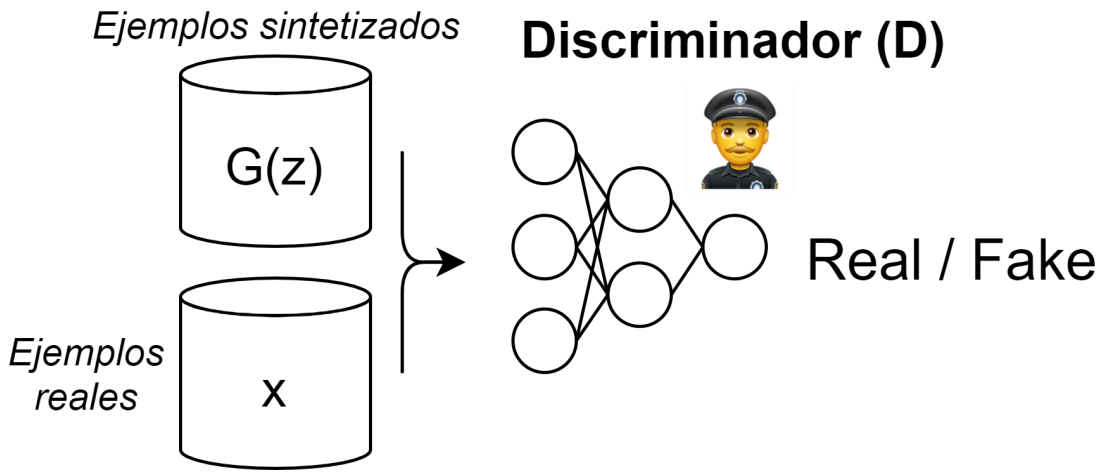
Genera datos a partir de un espacio latente que le proporciona aleatoriedad



El espacio latente es un vector de números aleatorios

Red discriminadora

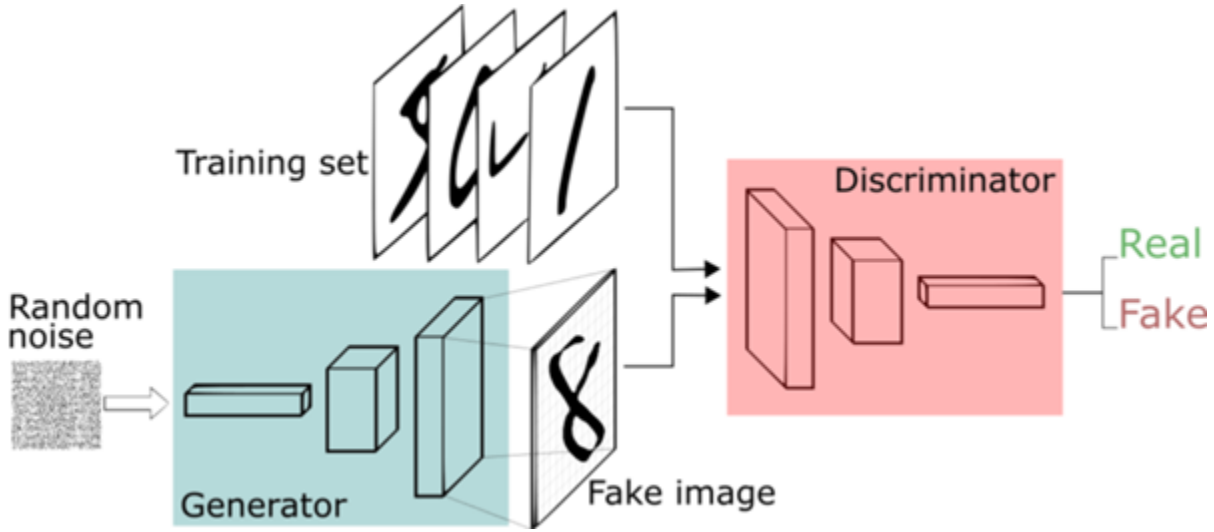
Discrimina los datos a partir de los reales y los sintéticos



Vanilla GAN

Arquitectura

La siguiente figura muestra el esquema general de una GAN *vanilla* (básica)



Su arquitectura se compone, además de generador y discriminador:

- **Ruido**: Vector aleatorio que se le pasa al generador para generar datos
- **Conjunto de entrenamiento**: Datos reales para entrenar el discriminador

En una *vanilla* GAN el discriminador se limita a un clasificador binario

Minimax game

Los entrenamientos de GANs se basan en un juego de suma cero

- Competición entre dos jugadores donde la ganancia de uno es la pérdida del otro

$$\min_G \max_D \mathcal{L}(D,G) = E_{x\sim p_r} log[D(x)] + E_{z\sim p_z} log[1 - D(G(x))]$$

- $x \sim p_r$: Distribución de los datos reales
- $z \sim p_z$: Distribución de probabilidad del espacio latente de G
  - Lo normal: Ruido gaussiano o uniforme utilizado para modelar nuevas muestras de datos

D busca diferenciar entre $D(x)$ (datos reales) y $D(G(x))$ (distribución sintética)

- El **generador** busca minimizar la expresión $\log[1 - D(G(x))]$
- El **discriminador** busca maximizar la expresión $\log[D(x)] + \log[1 - D(G(x))]$

Minimax game

Se converge cuando el generador y el discriminador llegan a un equilibrio de Nash

- Que es el punto óptimo para el juego minimax descrito anteriormente
- Los dos jugadores (modelos) tratan de engañarse mutuamente
- Por tanto **se alcanza** cuando uno de los jugadores **no cambia** su acción **independientemente** de lo que haga el **oponente**.

Las redes GAN son muy difíciles de entrenar en la práctica, con problemas como:

- **Colapso modal**: El generador genera siempre la misma muestra
- **Desvanecimiento del gradiente**
- **Explosión del gradiente**

Principales problemas de las GAN

Problemas de las GAN

Las GAN cuentan con una serie de problemas comunes

- Suelen tener que ver con las particularidades de su arquitectura

Revisaremos los siguientes problemas:

- Inestabilidad
- Problemas derivados del gradiente:
  - Desvanecimiento del gradiente
  - Explosión del gradiente
- Colapso modal
- Problemas de parada

El problema de la inestabilidad

El modelo GAN cuenta con dos redes neuronales

- Las redes neuronales de por sí son modelos inestables
- Al hacer que dos modelos interactúen entre sí se aumenta esta inestabilidad
- Mantener una coordinación entre ambas redes es un proceso delicado

Para intentar evitarla, existen técnicas para equilibrar el entrenamiento:

- Que las arquitecturas de la D y G tengan la misma forma, pero invertida
- Limitar la potencia de D cuando aprende más rápido (p.ej con regularización)

Para aprender, las dos redes necesitan que su antagónica falle de vez en cuando

Problemas del gradiente

Comunes a todo tipo de red y están relacionados con su número de capas

$$W_x' = W_x - \alpha \left(\frac{\partial \mathcal{L}}{\partial W_x} \right)$$

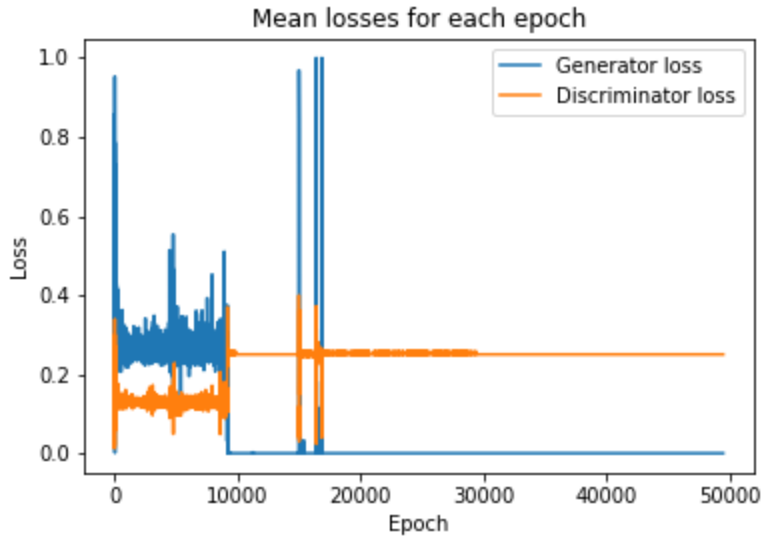
Al realizarse la retropropagación el error pasan de unas capas a otras

- Si el error es muy grande, se puede producir un explosión del gradiente
- Si el error es muy pequeño, se puede producir un desvanecimiento del gradiente

Gradient explosion

Cuando la actualización de pesos toma valores muy elevados

- También se conoce como *exploding gradients*

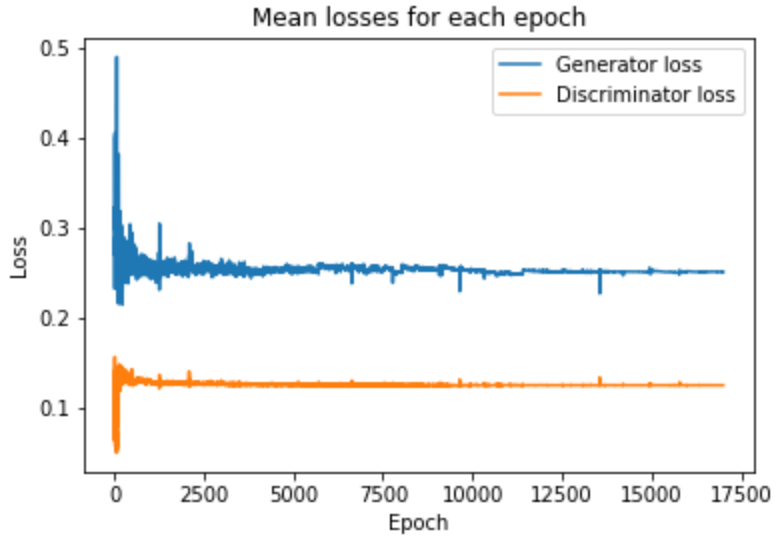


Se identifica con valores de pérdidas de `NaN` o muy exageradas

Gradient vanishing

Cuando la actualización de pesos se hace nula por ser valores muy pequeños

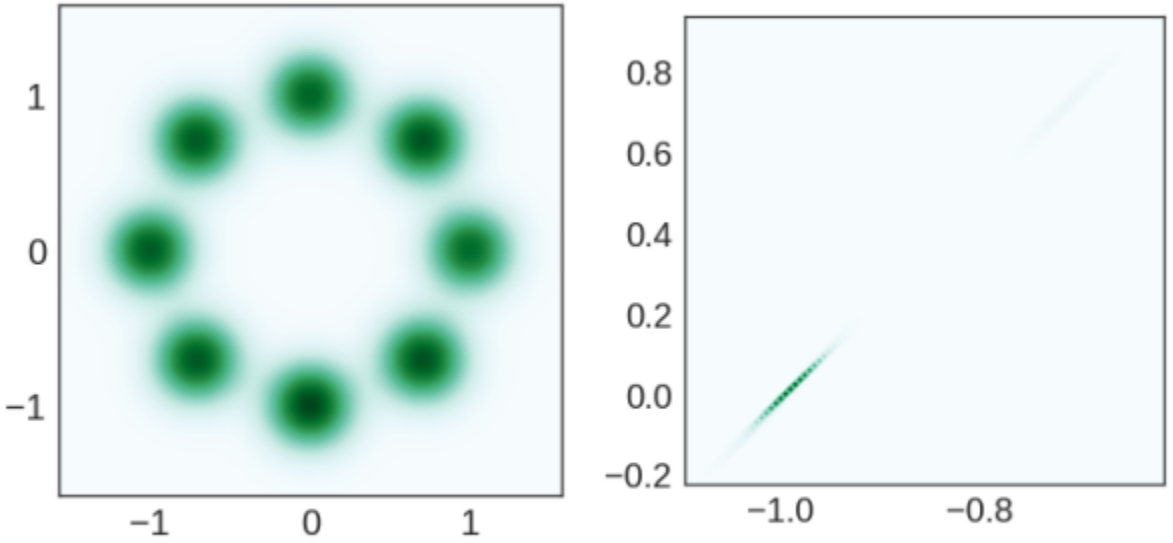
- También se conoce como *vanishing gradientes*



Se identifica cuando la pérdida es constante en el tiempo

Colapso modal (I)

La información generada sólo pertenece a un subgrupo de la total

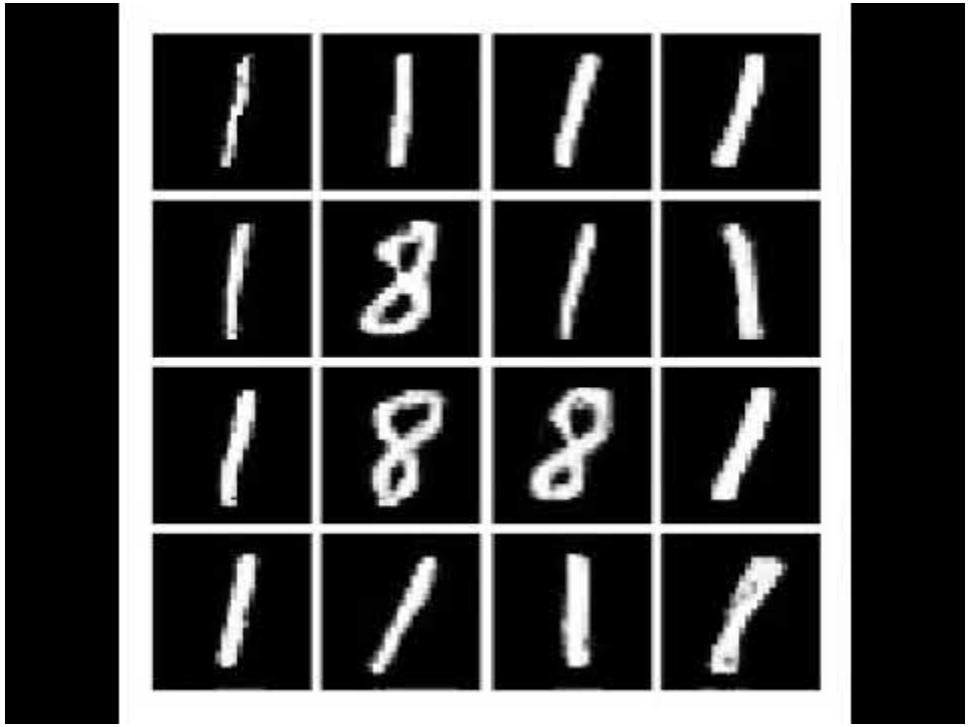


(a) True Data (b) GAN

Colapso modal (y II)

El ejemplo más clasico es el de producir siempre la misma imagen generadora

- Independientemente de la entrada de la red



El problema de la parada

En una red tradicional la optimización produce un descenso monótono del error

- De esta manera podemos determinar cuándo está optimizado y parar

En el entrenamiento de una GAN **la función de pérdida no sigue ningún patrón**

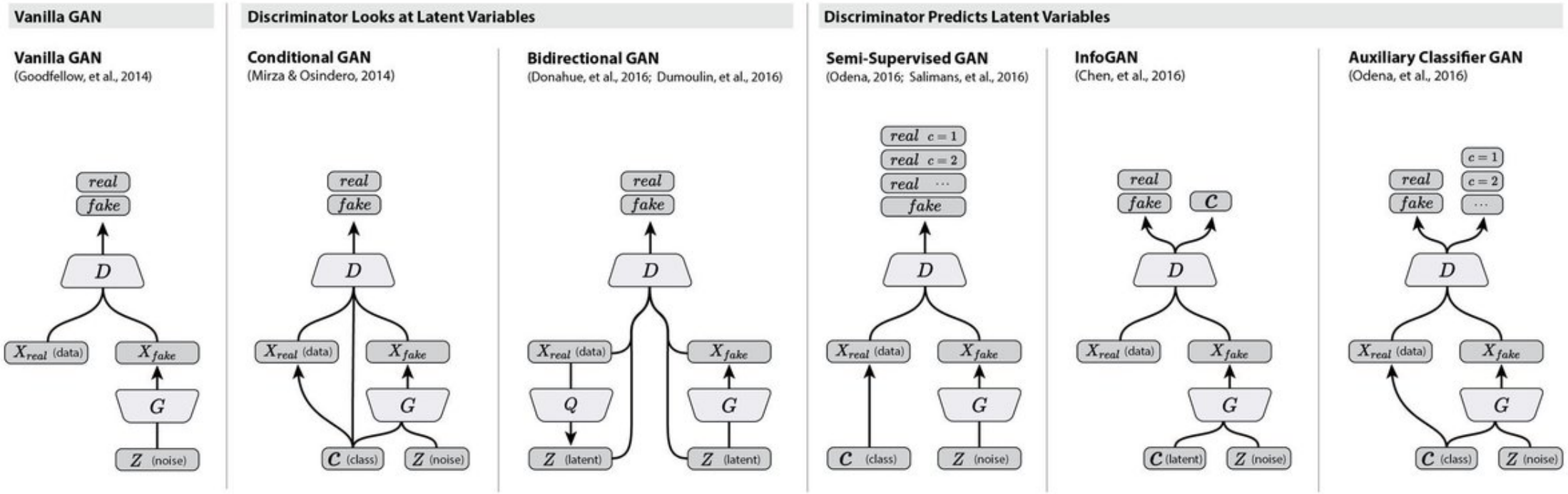
- Es imposible conocer el estado de las redes por su función de pérdida
- Por tanto, no podemos saber cuándo el modelo está suficientemente optimizado
  - No podemos saber con seguridad si es buen momento para parar el entrenamiento

Modelos avanzados

Modelos inciales

Existen muchas arquitecturas diferentes, algunas **muy** especializados

- Un buen punto de partida es aprender modelos iniciales y de alcance general
- A partir de ahí, podemos ir creciendo hacia modelos más concretos

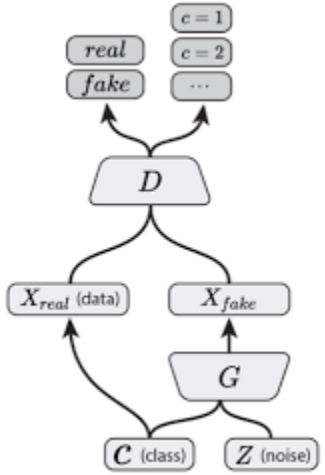


Un modelo inicial: ACGAN

Una GAN básica genera imaǵenes sin poder controlar qué se genera

- ¿Y si el añadimos la capacidad de añadir C clases?
- ¿Y si la red interna es una red convolucional en lugar de un perceptrón?

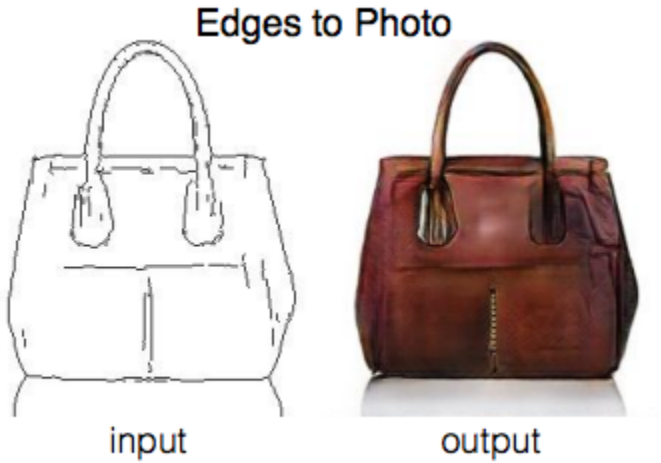
El resultado son las ACGAN: Auxiliary Classifier Generative Adversarial Network



Un modelo específico: Pix2Pix

Los modelos *Image-to-Image translation* (Pix2Pix) convierte una imagen en otra

- Son una forma de obtener imágenes «reales» a partir de imágenes sencillas



Podemos jugar con algunos ejemplos en pix2pix online

Aplicaciones de Pix2Pix

Algunas de las tareas donde se ha aplicado Pix2Pix pueden ser:

- Generación de [personajes de anime](Towards the Automatic Anime Characters Creation with Generative Adversarial Networks)
- Generación de imágenes a parir de pies de foto
- Térmica $\rightarrow$ fotos en color.
- Generación fotorealística de retratos de personas
- GauGAN Creación de imágenes fotorealísticas a partir de bocetos
- Modelos en tres dimensiones a partir de fotos
- GameGAN: Emulación de un juego completo sólo redes GAN

Gracias