





Redes Generativa Adversarias

PhD(e). MsC. Ing. Jonnatan Arias Garcia

- Ian Goodfellow et al.
- NeurIPS 2014

NeurIPS Proceedings  

Search

Search

Generative Adversarial Nets

Part of [Advances in Neural Information Processing Systems 27 \(NIPS 2014\)](#)

[Bibtex](#) [Metadata](#) [Paper](#) [Reviews](#)

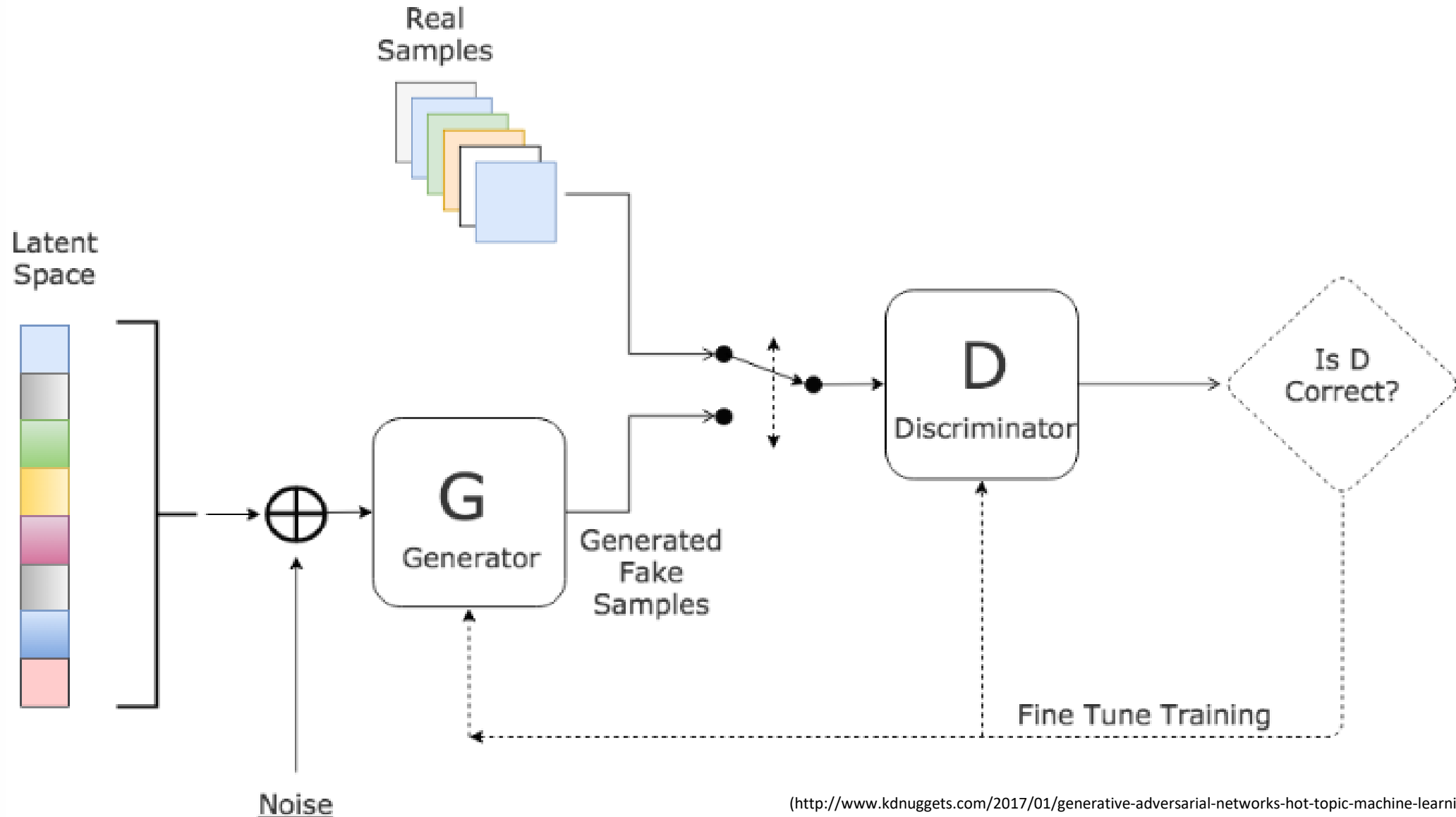
Authors

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

Abstract

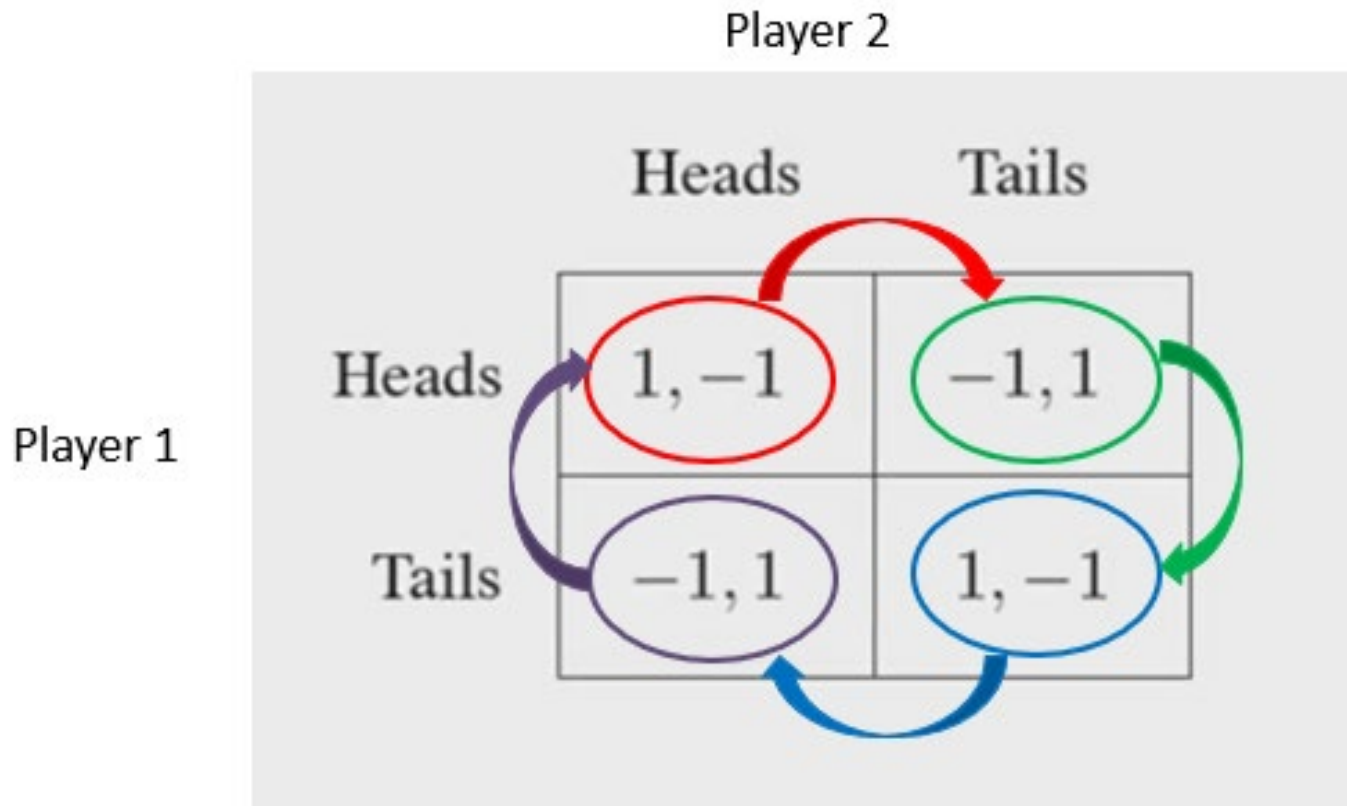
We propose a new framework for estimating generative models via adversarial nets, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G . The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D , a unique solution exists, with G recovering the training data distribution and D equal to $1/2$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

Generative Adversarial Network



Idea detrás de las GANs

- se basa en un enfoque de aprendizaje adversarial en el que dos redes neuronales compiten entre sí en un juego de suma cero

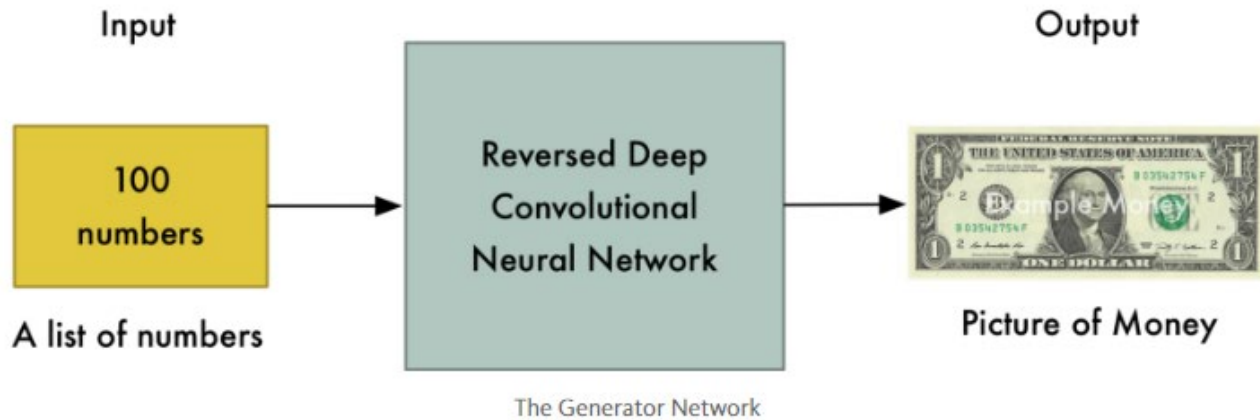


Suma cero: ganancias o pérdidas de un participante son exactamente iguales a las pérdidas o ganancias del otro participante.

Entrenamiento de la GAN

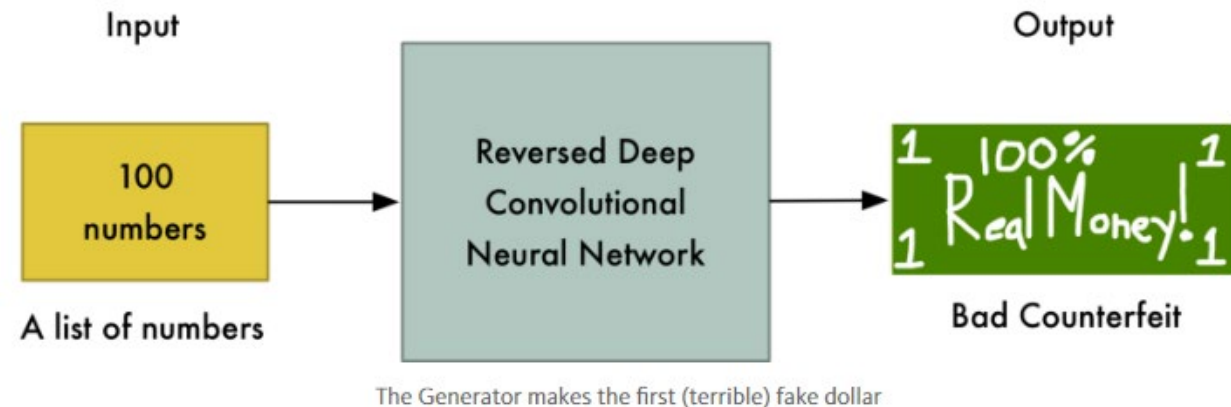
Generador (Generator, G):

- El generador toma una muestra de ruido aleatorio y la transforma en datos falsos que intentan imitar los datos reales del conjunto de entrenamiento.
- Su objetivo es generar datos que no puedan ser distinguidos de los datos reales por el discriminador.



- Update the generator by descending its stochastic gradient:

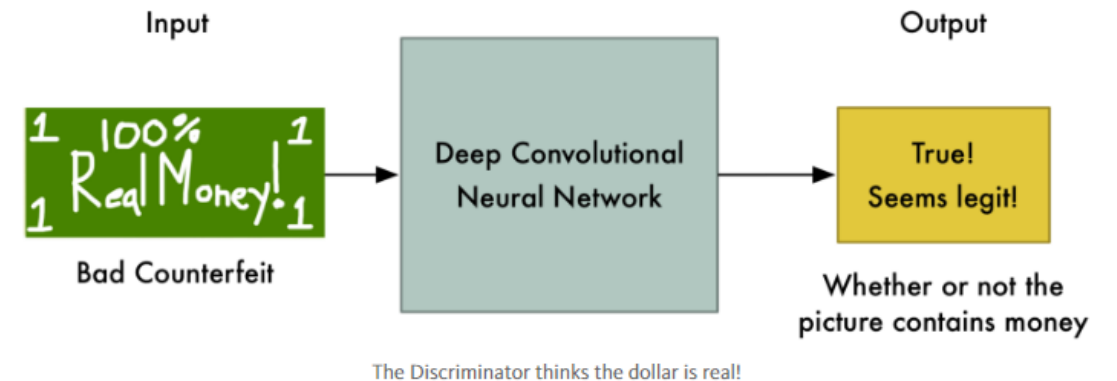
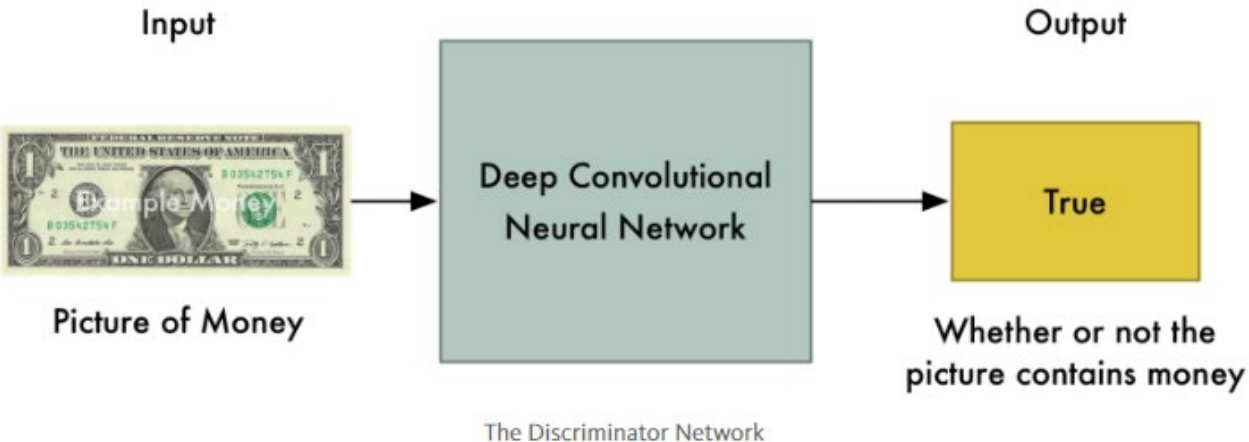
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(1 - D \left(G \left(z^{(i)} \right) \right) \right)$$



Entrenamiento de la GAN

Discriminador (Discriminator, D):

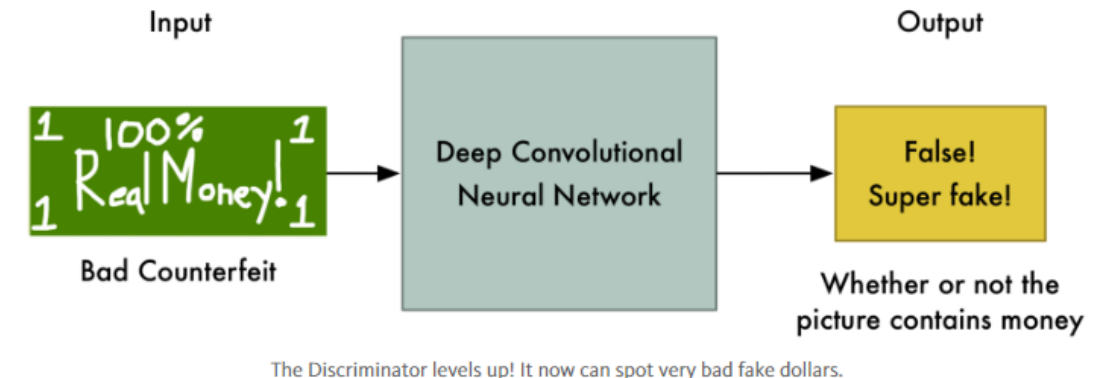
- El discriminador recibe tanto datos reales como datos generados (falsos) y trata de clasificarlos correctamente como reales o falsos.
- Su objetivo es maximizar la precisión en distinguir entre datos reales y generados.



- Update the discriminator by ascending its stochastic gradient:

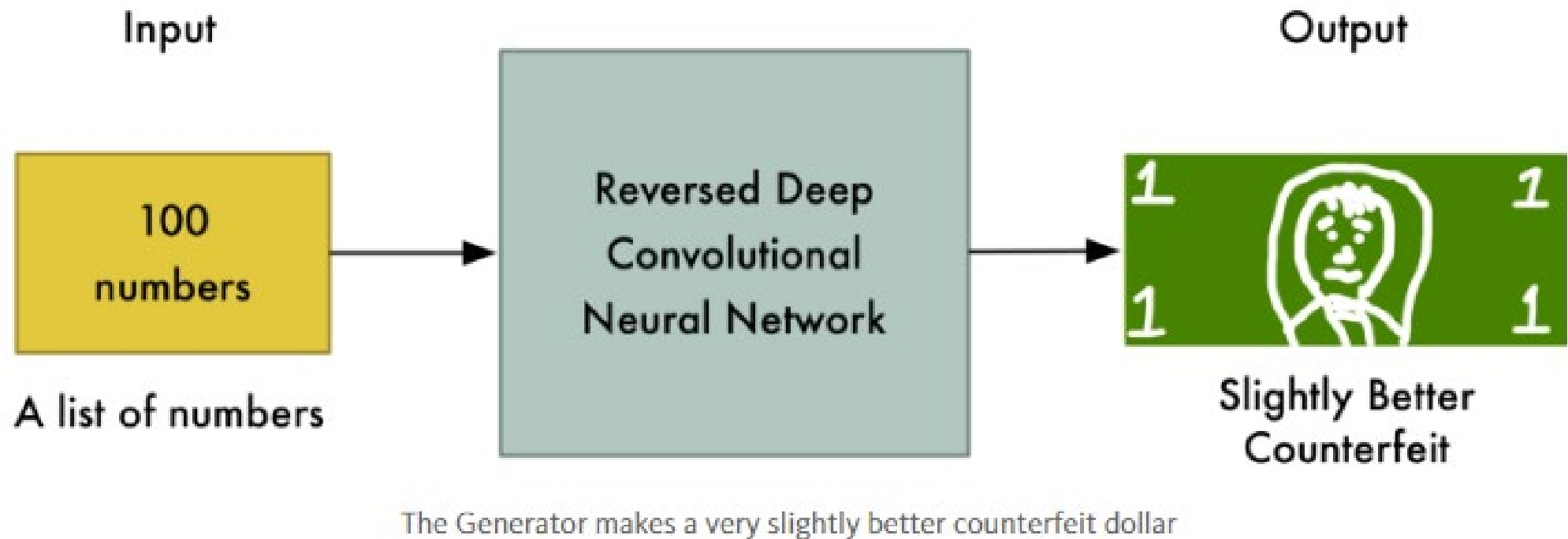
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right]$$

From minibatch of data From minibatch of noise



Entrenamiento de la GAN

El generador crea versiones (falsas) mejores



Funcionamiento de la GAN

1. Inicialización

- Comúnmente con pesos aleatorio tanto en el discriminador como el en generador

2. Entrenamiento Alternante

Se realiza alternando entre los pasos del generador y el discriminador:

- **Actualizar el Discriminador:**

- Se entrena el discriminador con un mini-lote de datos reales y un mini-lote de datos generados por el generador.
- El discriminador ajusta sus pesos para minimizar la pérdida en la clasificación de datos reales y generados.

- **Actualizar el Generador:**

- Se entrena el generador usando la retropropagación a través del discriminador.
- El generador ajusta sus pesos para maximizar la probabilidad de que el discriminador clasifique los datos generados como reales.
- Esto se hace tratando de "engañar" al discriminador.

Funcionamiento de la GAN

3. Objetivo de Minimización y Maximización:

- El objetivo del generador es minimizar la función de pérdida que representa cuán bien puede el discriminador identificar los datos falsos.
- El objetivo del discriminador es maximizar la función de pérdida que representa su capacidad para distinguir entre datos reales y generados.

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

Donde:

- $D(x)$ es la probabilidad de que x provenga de los datos reales.
- $G(z)$ es el dato generado a partir del ruido z .
- $p_{data}(x)$ es la distribución de los datos reales.
- $p_z(z)$ es la distribución del ruido.

Funcionamiento de la GAN

Actualización de discriminador y generador

k times

- Update the discriminator by ascending its stochastic gradient:

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



- Update the generator by descending its stochastic gradient:

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

En la practica

- $D1 = D(x)$ \longleftarrow (D wants it to be near 1)
- $D2 = D(G(z))$ \longleftarrow (D wants it to be near 0)

```
loss_d = tf.reduce_mean(-tf.log(D1) - tf.log(1 - D2))  
loss_g = tf.reduce_mean(-tf.log(D2))
```

Colapso de la moda

- También conocido como **escenario Helvetica**
- Problema donde el generador produce una variedad limitada de muestras y queda atrapado en pocas modas del conjunto de datos de train., por ende crea copias de una pequeña cantidad.
- El generador aprende a mapear muchas entrada diferentes del valor z para el mismo punto de salida
- No parece ser causado por ninguna función de costo en particular

Colapso de la moda



Figure 9: Comparison to Generative Adversarial What-Where Networks (Reed et al., 2016a). GAN samples have very low diversity, whereas our samples are all quite different.

Colapso de la moda

- Ocurre debido a:
 - Desequilibrio en el Train.
 - Discriminador mejora mas rápido que el generador.
 - Gradientes Faltantes
 - Gradiente que se retropropagan desde el discriminador al generador pueden ser insuficientes o débiles para impulsar al generador a explorar.
 - Inestabilidad del Train
 - Pequeñas modificaciones en el discriminador pueden hacer variar mucho al generador

Colapso de la moda

- Mejoras para evitar el colapso
 - **Wasserstein GAN (WGAN):** Usa métricas para diferencias entre reales y generadas
 - **Minibatch Discrimination:** Mini batches
 - **Feature Matching:** Se entrena el generador para igualar estadísticas de una capa intermedia de discriminador en lugar de engañarlo.
 - **Unrolled GANs:** extender en train el generador previo a actualizar el discriminador
 - **Latent Space Regularization:** regularización de z
 - **Ensembles of Discriminators:** múltiples discriminadores

Aplicaciones

Generación de Imágenes: Crear imágenes realistas a partir de ruido, estilo de transferencia, superresolución de imágenes, etc.

Generación de Datos Sintéticos: Crear datos sintéticos para entrenar otros modelos de aprendizaje automático.

Video y Animación: Generar videos realistas o animaciones a partir de un conjunto de imágenes estáticas.

Música y Arte: Generar música, pinturas y otros tipos de arte digital.

Modelado 3D: Crear modelos tridimensionales realistas de objetos.

Desafíos y Mejoras

Estabilidad del Entrenamiento: El entrenamiento puede ser inestable y puede llevar a problemas como el colapso del modo, donde el generador produce una variedad limitada de muestras.

Equilibrio entre G y D: Mantener un equilibrio adecuado entre las capacidades del generador y el discriminador es crucial para el éxito del entrenamiento.