

# GAN: GENERATIVE ADVERSARIAL NETWORKS

FÍSICA COMPUTACIONAL II

PH.D. SANTIAGO ECHEVERRI ARTEAGA



## GENERATIVE ADVERSARIAL NETWORKS

- Los investigadores iban a realizar un concurso de síntesis de habla para ver cuál era la red que podía generar voces más realistas.
- Decidieron cancelarlo porque una red neuronal “el discriminador” iba a ser el juez del concurso y podría ser falseado por redes que en lugar de ser realistas se aprovecharan de las debilidades de la época en términos de la forma de entrenar.
- Esto llevo a Ian Goodfellow y su equipo a desarrollar un modelo en que el discriminador continuaría mejorando para reconocer voces falsas de verdaderas a medida que se entrenaba la red que las generaba.

THE GENERATIVE MODEL CAN BE THOUGHT OF AS ANALOGOUS TO A TEAM OF COUNTERFEITERS, TRYING TO PRODUCE FAKE CURRENCY AND USE IT WITHOUT DETECTION, WHILE THE DISCRIMINATIVE MODEL IS ANALOGOUS TO THE POLICE, TRYING TO DETECT THE COUNTERFEIT CURRENCY. COMPETITION IN THIS GAME DRIVES BOTH TEAMS TO IMPROVE THEIR METHODS UNTIL THE COUNTERFEITS ARE INDISTINGUISHABLE FROM THE GENUINE ARTICLES.

Ian Godfellow et. al.



Generator



Real Money



Fake Money

Discriminator

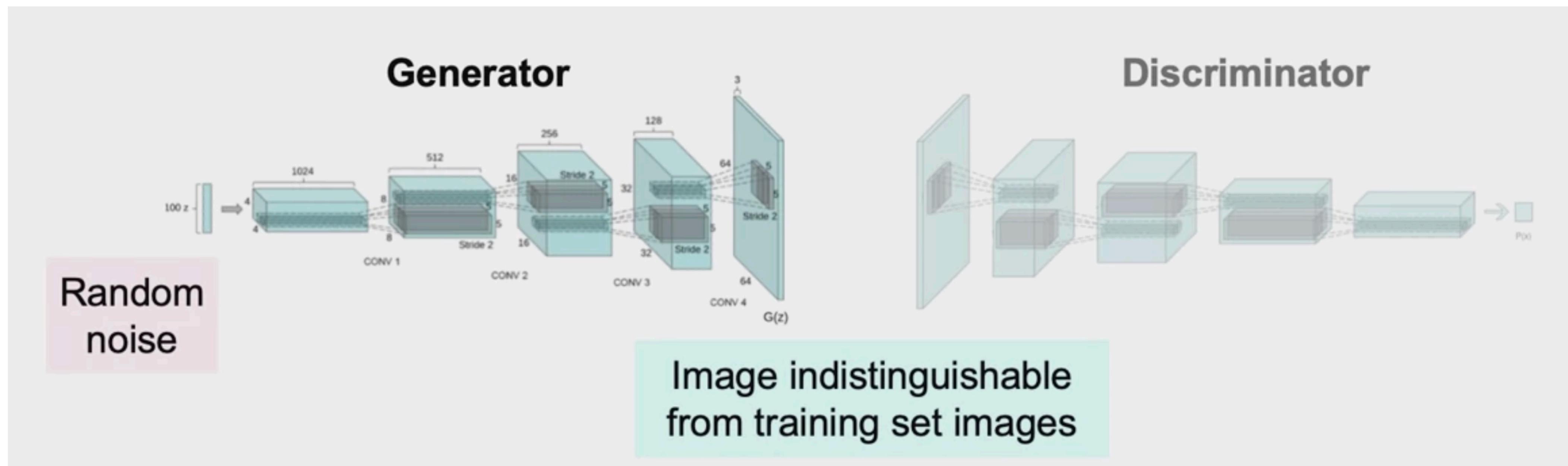


Counterfeiter prints fake money. It is labeled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labeling the fake money as real.

The police are trained to distinguish between. Sometimes, the police give feedback to the counterfeiter about why the money is fake.

## GENERATIVE ADVERSARIAL NETWORKS

- Para ello se alimentaría al discriminador tanto con voces reales como falsas y debía distinguir una de otra.
- Al pasar el gradiente del Loss del discriminador respecto al input al modelo generativo se podía mejorar su eficacia en generar voces más realistas.
- El generador genera a partir de ruido aleatorio una imagen que sea indistinguible a las del dataset original



¿CÓMO FUNCIONAN?

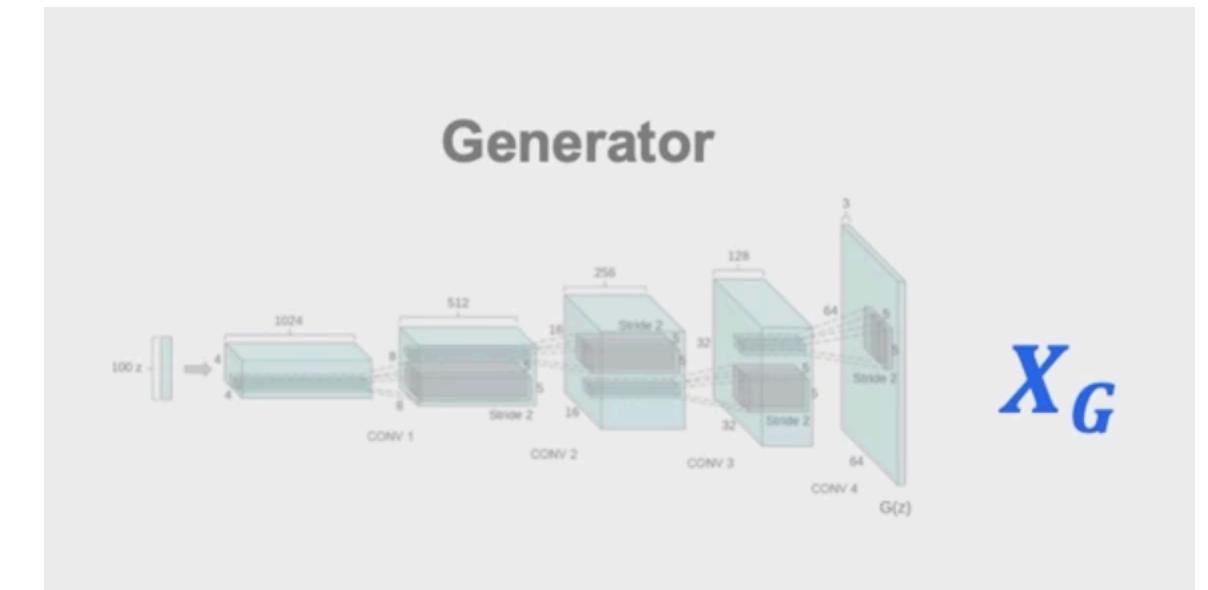
## GANS: PROCEDIMIENTO

1. Inicializar los pesos del generador y del discriminador aleatoriamente
2. Inicializar el vector de ruido y generar la imagen usando el generador  $X_G$  (Del mismo tamaño que el training set)
3. Predecir con el discriminador la probabilidad  $P_{real}^{X_G}$  de que la imagen  $X_G$  sea real
4. Calcular los Loss asumiendo que  $X_G$  fue real  $L_1^{X_G} = f(P_{real}^{X_g}, 1)$  (Comparando  $P_{real}^{X_G}$  con 1) y asumiendo que fue falso  $L_0^{X_G} = f(P_{real}^{X_g}, 0)$  (Comparando  $P_{real}^{X_G}$  con 0)
5. Use  $L_0^{X_G}$  para entrenar el discriminador. **Éste solo se interesa por predecir que la imagen no es real** Haciendo backpropagation **solo para el discriminador**
6. Calcule  $\frac{\partial L_{real}^{X_G}}{\partial X_G}$  (¿La imagen generada se ve realista?) con backpropagation pero NO actualice con esta los pesos del dicriminador
7. Use  $\frac{\partial L_{real}^{X_G}}{\partial X_G}$  para entrenar el generador (Para que las imágenes se vean realistas hay que fijarse en los errores del discriminador)
8. Pase imágenes reales  $X_R$  a través del discriminador y calcule la probabilidad de que sean reales  $P_{real}^{X_R}$
9. Calcule  $P_{real}^{X_R}$  y con esto  $L_1^{X_R} = f(P_{real}^{X_R}, 1)$
10. Use  $L_1^{X_R} = f(P_{real}^{X_R}, 1)$  para entrenar el discriminador
11. Repita el procedimiento con random noise generado para el generador **hasta que las imágenes del generador parezcan reales**

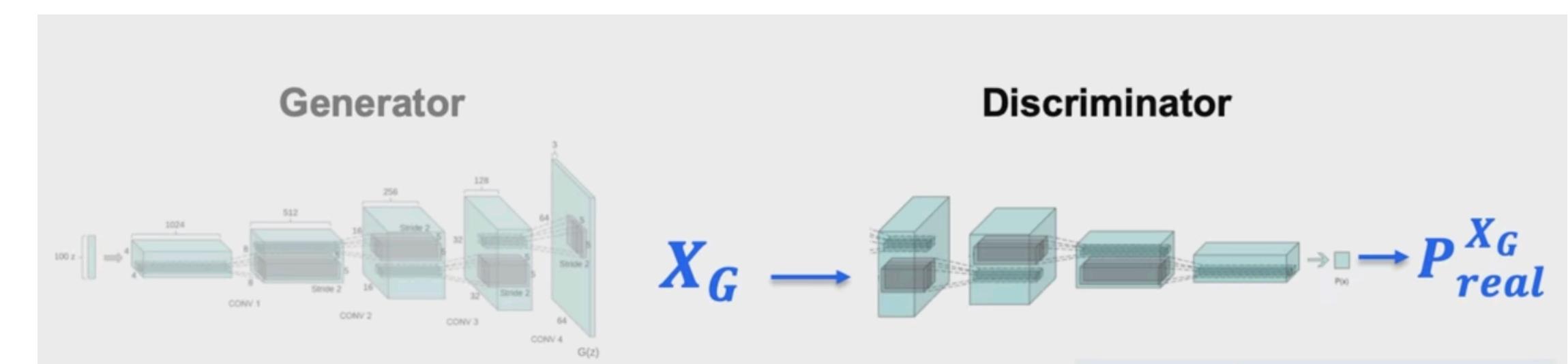
¿CÓMO FUNCIONAN?

## GANS: PROCEDIMIENTO

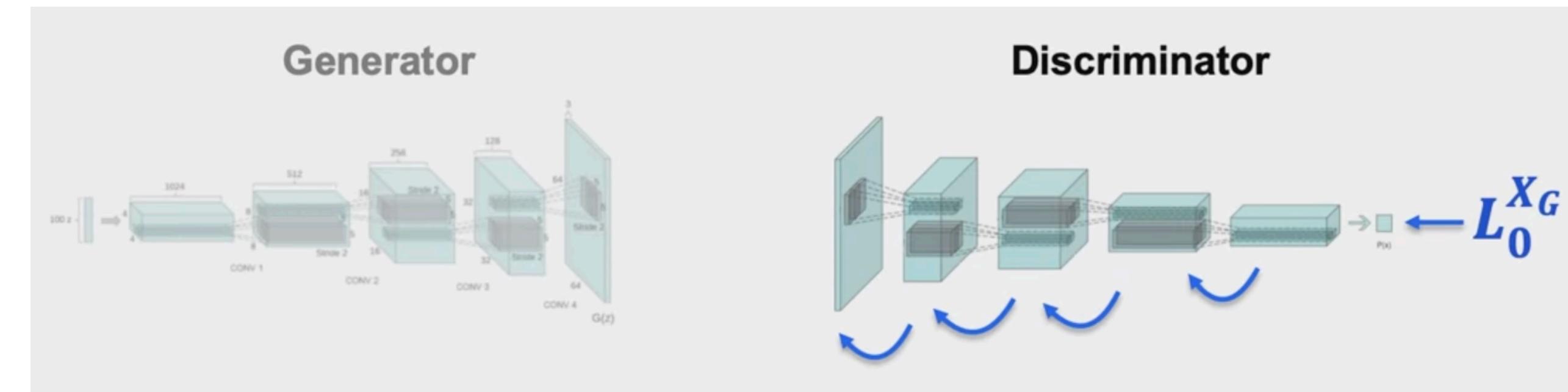
1. Inicializar los pesos del generador y del discriminador aleatoriamente
2. Inicializar el vector de ruido y generar la imagen usando el generador  $X_G$  (Del mismo tamaño que el training set)



3. Predecir con el discriminador la probabilidad  $P_{real}^{X_G}$  de que la imagen  $X_G$  sea real

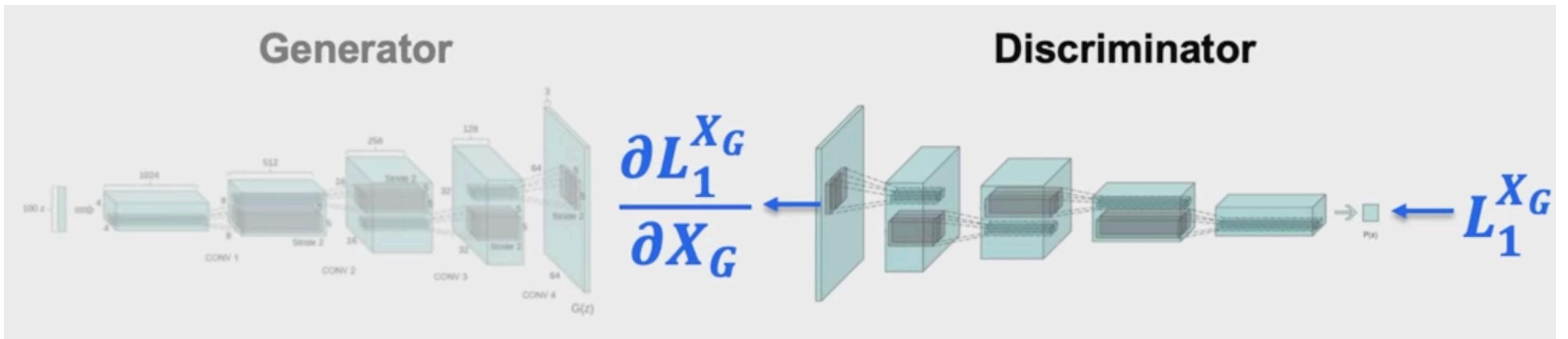


4. Calcular los Loss asumiendo que  $X_G$  fue real  $L_1^{X_G} = f(P_{real}^{X_g}, 1)$  (Comparando  $P_{real}^{X_G}$  con 1) y asumiendo que fue falso  $L_0^{X_G} = f(P_{real}^{X_g}, 0)$  (Comparando  $P_{real}^{X_G}$  con 0)
5. Use  $L_0^{X_G}$  para entrenar el discriminador. **Este solo se interesa por predecir que la imagen no es real** Haciendo backpropagation **solo para el discriminador**

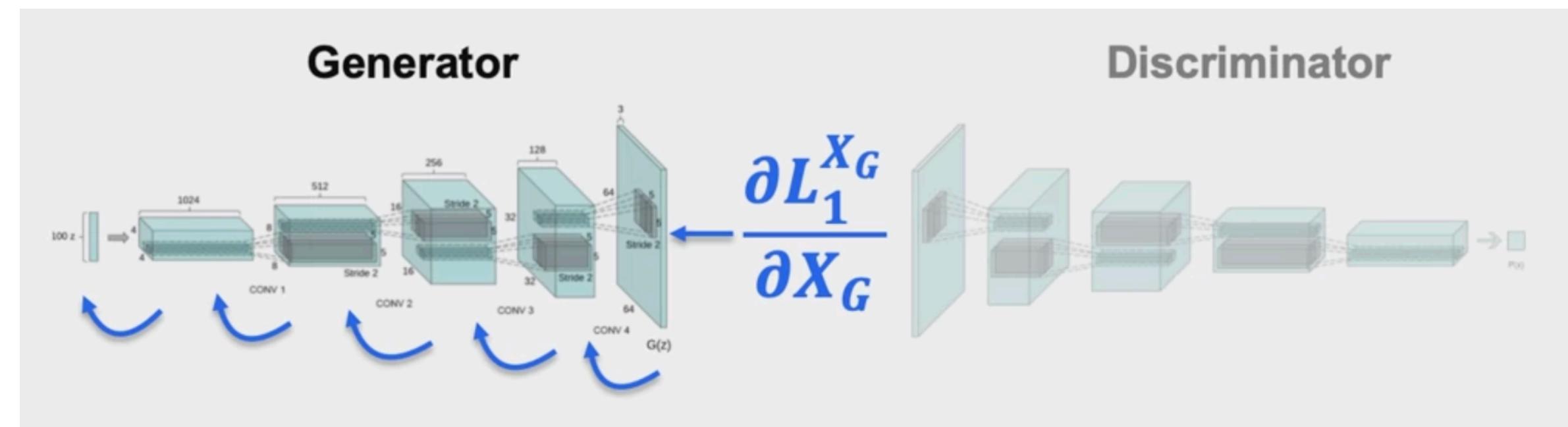


# GANS: PROCEDIMIENTO

6. Calcule  $\frac{\partial L_{real}^{X_G}}{\partial X_G}$  con backpropagation (¿La imagen generada se ve realista?) pero NO actualice con esta los pesos del discriminador



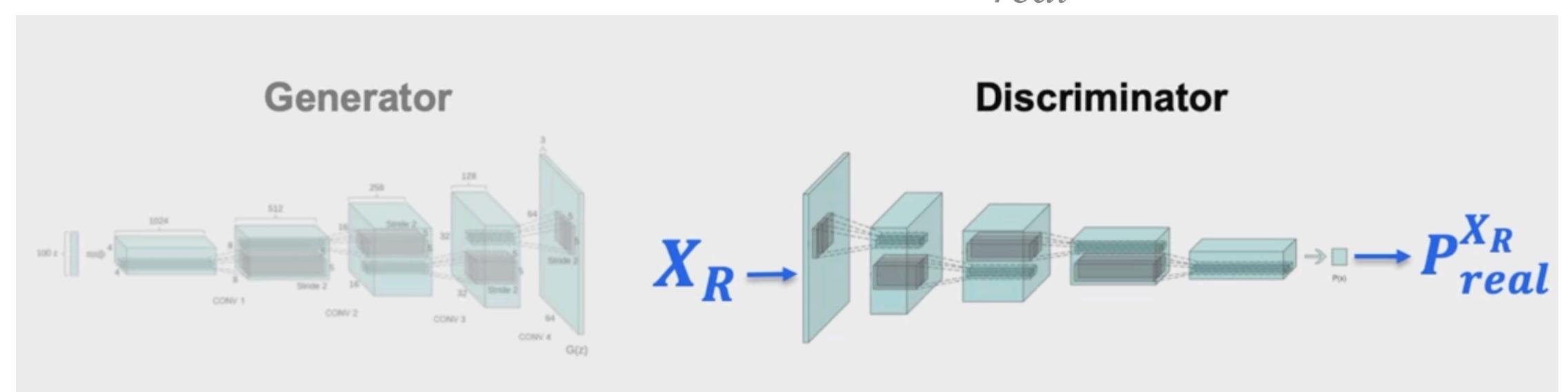
7. Use  $\frac{\partial L_{real}^{X_G}}{\partial X_G}$  para entrenar el generador (Para que las imágenes se vean realistas hay que fijarse en los errores del discriminador)



8. Pase imágenes reales  $X_R$  a través del discriminador y calcule la probabilidad de que sean reales  $P_{real}^{X_R}$

9. Calcule  $P_{real}^{X_R}$  y con esto  $L_1^{X_R} = f(P_{real}^{X_R}, 1)$

10. Use  $L_1^{X_R} = f(P_{real}^{X_R}, 1)$  para entrenar el discriminador

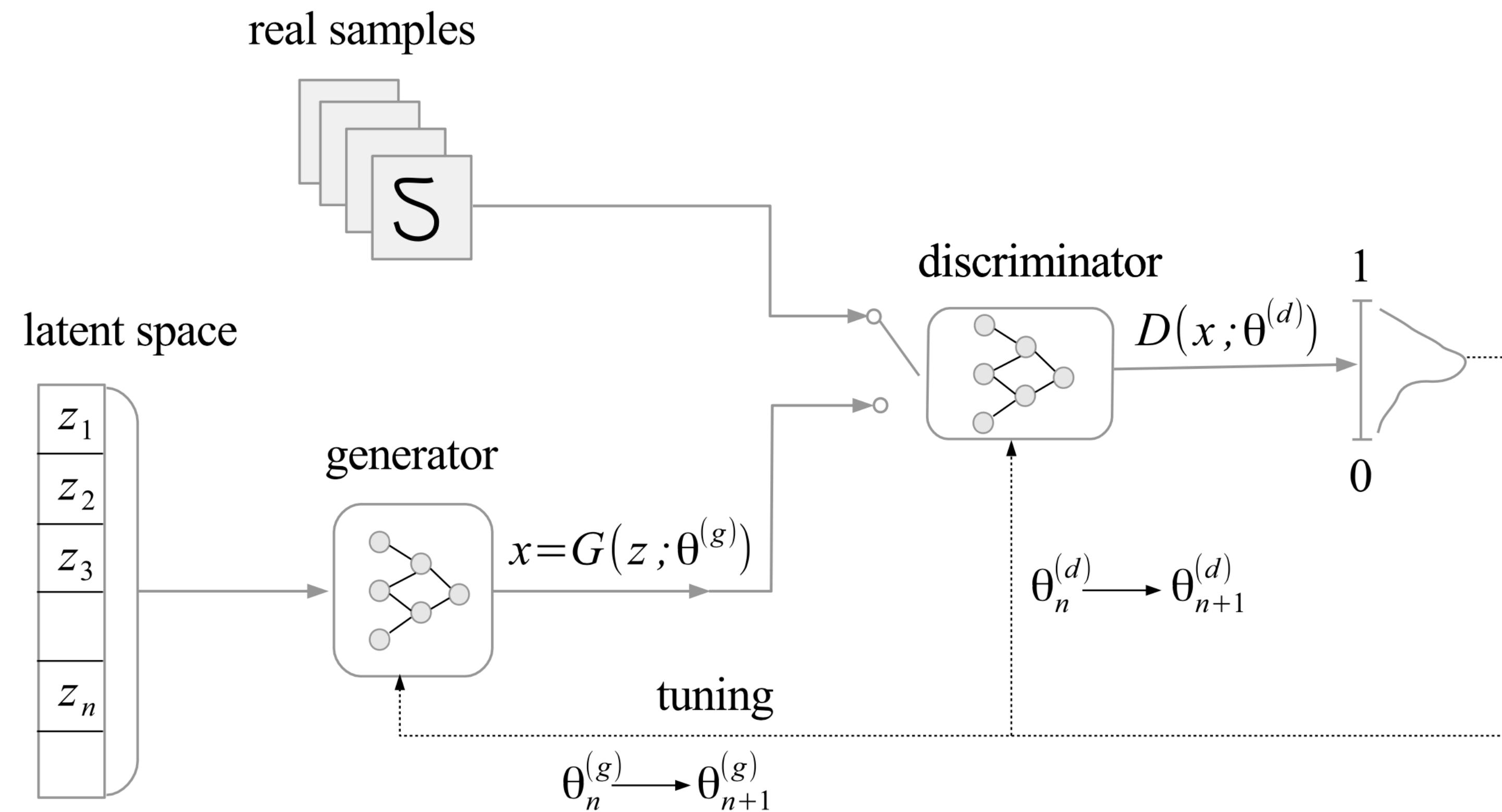


# FUNDAMENTO MATEMÁTICO



## GANS... GANS EVERYWHERE!

Sean  $D$  el discriminador,  $G$  el generador,  $\theta_d$  los parámetros del discriminador,  $\theta_g$  los parámetros del generador,  $P_z(z)$  la distribución de ruido de entrada,  $P_{data}(x)$  la distribución original de los datos y  $P_g(x)$  la distribución generada



## GANS... GANS EVERYWHERE!

---

Sean  $D$  el discriminador,  $G$  el generador,  $\theta_d$  los parámetros del discriminador,  $\theta_g$  los parámetros del generador,  $P_z(z)$  la distribución de ruido de entrada,  $P_{data}(x)$  la distribución original de los datos y  $P_g(x)$  la distribución generada.

Tomando como base la Binary cross-entropy  $L(\hat{y}, y) = [y \log(\hat{y}) + (1 - y)\log(1 - \hat{y})]$  ([la cuál se relaciona con la entropía relativa de  \$p\$  respecto a  \$q\$  o también llamada Kullback-Leibler divergence  \$D\_{KL}\(p\|q\)\$](#) ) como  $H(p, q) = H(p) + D_{KL}(p\|q)$  con  $H(q)$  la entropía de  $q$ .

Cuando los datos provienen de  $P_{data}(x)$ ,  $y = 1 \therefore L(D(x), 1) = \log(D(x))$

Cuando los datos provienen del generador,  $y = 0 \therefore L(D(G(z)), 0) = \log(1 - D(G(z)))$

Así, el como el objetivo del discriminador es clasificar correctamente los datos falsos y reales del dataset, su Loss es  $L^{(D)} = \max [\log(D(x)) + \log(1 - D(G(z)))]$

Al mismo tiempo, el generador está compitiendo contra el discriminador, por lo que su Loss será  $L^{(G)} = \min [\log(D(x)) + \log(1 - D(G(z)))]$

La función Loss combinada sería  $L = \min_G \max_D [\log(D(x)) + \log(1 - D(G(z)))]$

Sean  $D$  el discriminador,  $G$  el generador,  $\theta_d$  los parámetros del discriminador,  $\theta_g$  los parámetros del generador,  $P_z(z)$  la distribución de ruido de entrada,  $P_{data}(x)$  la distribución original de los datos y  $P_g(x)$  la distribución generada.

La función Loss dada por  $L = \min_G \max_D \left[ \log(D(x)) + \log(1 - D(G(z))) \right]$ , es válida solo para un punto, por lo que hay que tomar los valores esperados

$$\min_G \max_D V(D, G) = \min_G \max_D \left[ E_{x \sim P_{data}(x)} [\log(D(x))] + E_{z \sim P_z(z)} [\log(1 - D(G(z)))] \right]$$

Que también se puede escribir como una integral tomada sobre el espacio  $\mathcal{X}$

$$V(D, G) = \int_{\mathcal{X}} P_{data}(x) \ln(D(x)) + P_z(x) \ln(1 - D(G(x))) dx$$

## GANS... GANS EVERYWHERE!

---

$$V(D, G) = \int_{\mathcal{X}} P_{data}(x) \ln(D(x)) + P_z(x) \ln(1 - D(G(x))) dx$$

Para un generador  $G$  fijo se puede considerar el Loss como un funcional  $F$  de  $D(x)$

como  $F(D(x)) = \int_{\mathcal{X}} L(D(x)) dx$  con el Lagrangiano

$$L = P_{data}(x) \ln(D(x)) + P_z(x) \ln(1 - D(G(x)))$$

En donde los puntos críticos satisfacen  $\frac{\partial L((D))}{\partial D} = 0$ , es decir:  $\frac{P_{data}(x)}{D(x)} - \frac{P_z(x)}{1 - D(x)} = 0$

Lo cual al resolverlo para  $D(x)$  se obtiene el discriminador óptimo (segunda derivada

negativa) dado un generador fijo  $D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_z(x)}$

Sin embargo  $P_{data}(x)$  no es dado 😞

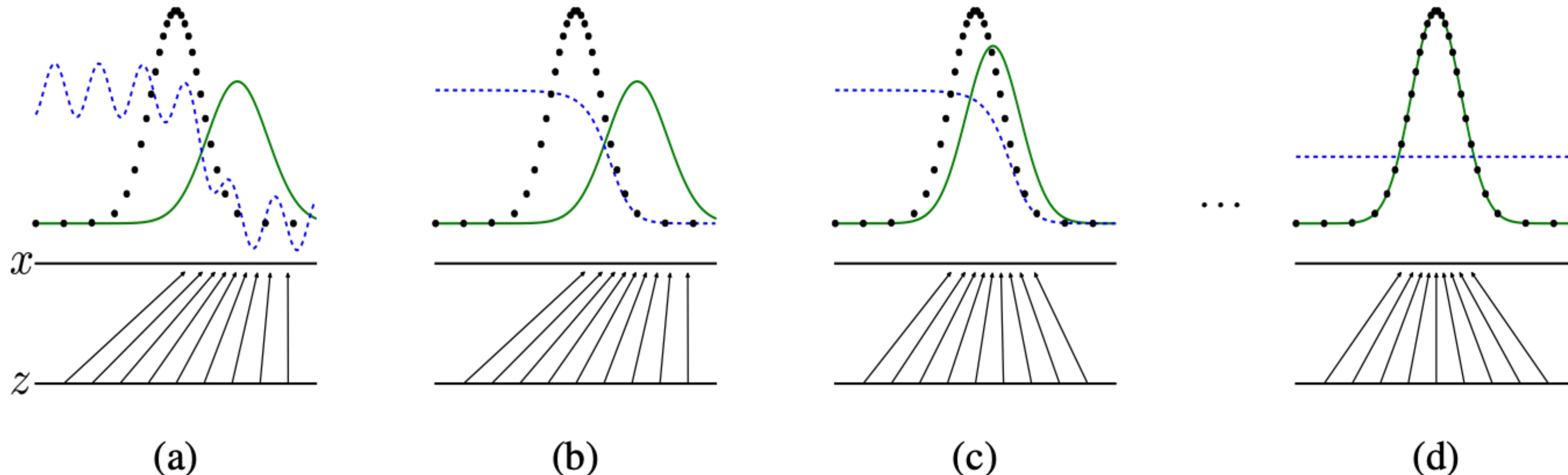
Se puede mostrar que  $V(G, D_G^*) = 2D_{JS} \left( P_z(x) \| P_{data}(x) \right) - \ln 4$  donde  $D_{JS}$  es la divergencia de Jensen-Shanon, también conocido como radio de información

Además, se puede mostrar que el mínimo global  $G^* = \arg \min_G V(G, D_G^*)$  se consigue si y solo si  $P_{data}(x) = P_z(x)$

Y se acaba de mostrar que esta solución EXISTE Y ES ÚNICA!!!!

---

# ¿Y QUÉ SIGNIFICA ÉSTE RESULTADO?



**Figure 1:** Generative adversarial nets are trained by simultaneously updating the **discriminative distribution** ( $D$ , blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line)  $p_x$  from those of the **generative distribution**  $p_g$  ( $G$ ) (green, solid line). The lower horizontal line is the domain from which  $z$  is sampled, in this case uniformly. The horizontal line above is part of the domain of  $x$ . The upward arrows show how the mapping  $x = G(z)$  imposes the non-uniform distribution  $p_g$  on transformed samples.  $G$  contracts in regions of high density and expands in regions of low density of  $p_g$ . (a) Consider an adversarial pair near convergence:  $p_g$  is similar to  $p_{\text{data}}$  and  $D$  is a partially accurate classifier. (b) In the inner loop of the algorithm  $D$  is trained to discriminate samples from data, converging to  $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$ . (c) After an update to  $G$ , gradient of  $D$  has guided  $G(z)$  to flow to regions that are more likely to be classified as data. (d) After several steps of training, if  $G$  and  $D$  have enough capacity, they will reach a point at which both cannot improve because  $p_g = p_{\text{data}}$ . The discriminator is unable to differentiate between the two distributions, i.e.  $D(x) = \frac{1}{2}$ .

---

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

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

**end for**

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

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

**end for**

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

---

ONE DOES NOT SIMPLY

TRAIN A GAN

imgflip.com

GANS .....



GANS EVERYWHERE

memycreator.net

## PROBLEMAS AL MOMENTO DE ENTRENAR

- ▶ Las Losses del generador y discriminador van a estar fluctuando, por lo que NO son un buen sensor para detener el entrenamiento
- ▶ Hay formas para cuantificar la calidad de una imagen (Inception score y otros) **TAREA**
- ▶ Son más sensibles que otras redes a
  - ▶ Arquitectura de ambas redes
  - ▶ Learning rates
  - ▶ Loss functions
  - ▶ Técnicas de optimización
- ▶ Ejemplos: Deepfakes, Interpolación de edad, texto a imágenes
- ▶ REVISAR PAPERS
- ▶ Vanishing gradients
- ▶ Colapso de modos: El algoritmo puede llegar a un punto de estancamiento en donde no hay mejora



## MNIST CONDITIONAL GAN EN KERAS

<https://colab.research.google.com/drive/1IUuNRCSwmfofa7J7I0-TqSW62ydai9Mn?usp=sharing>

## PyTorch Example

[https://pytorch.org/tutorials/beginner/dcgan\\_faces\\_tutorial.html](https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html)