

GANs, WGANs y extensiones

Diego Fonseca

30 de junio de 2021

Kullback–Leibler y Jensen–Shannon Divergence

Generative Adverarial Networks GAN

Wasserstein GAN (WGAN)

CycleGAN

Kullback–Leibler y Jensen–Shannon Divergence

KL (Kullback–Leibler) divergence mide como una distribución de probabilidad p difiere de una segunda distribución de probabilidad q .

$$D_{KL}(p \parallel q) := \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx.$$

Note que D_{KL} no es simétrica. En casos donde $p(x)$ es cercano a cero pero $q(x)$ no es cero podría causar problema sobretodo cuando la intención es medir la similitud de las dos distribuciones.

Jensen–Shannon Divergence es otra medida de similitud entre dos distribuciones de probabilidad, acotada por $[0, 1]$. JS si es simétrica.

$$D_{JS}(p \parallel q) := \frac{1}{2} D_{KL}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \parallel \frac{p+q}{2}\right).$$

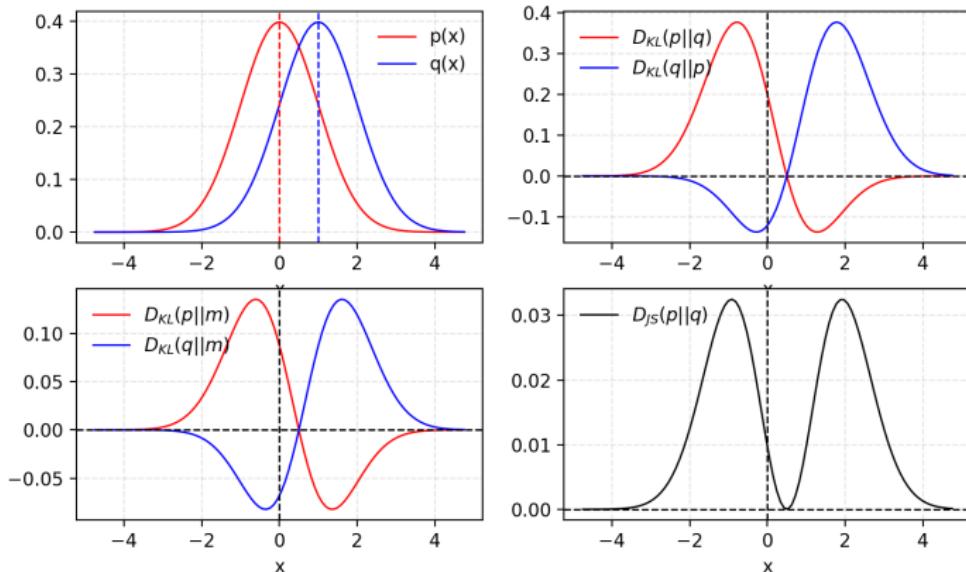


Figura: Dos distribuciones gaussianas, p con $\text{media} = 0$ y $\text{std} = 1$ y q with $\text{media} = 1$ y $\text{std} = 1$. el promedio de las dos distribuciones es notado por $m = (p + q)/2$. D_{KL} es asimétrica pero D_{JS} es simétrica.

Generative Adversarial Networks GAN (Goodfellow, 2014)

Un **GAN** consiste de dos modelos:

- Un **discriminador** D estima la probabilidad de que un ejemplo dado provenga de la base de datos real. Esta trabaja como un critico el cual es optimizado para distinguir los ejemplos falsos de los reales.
- Un **generador** G genera ejemplos sintéticos a partir de una variable de ruido z la cual tiene como input. Esta es entrenada para capturar la distribución real de los datos lo mas preciso posible, en otra palabras, poder engañar al discriminador con alta probabilidad.

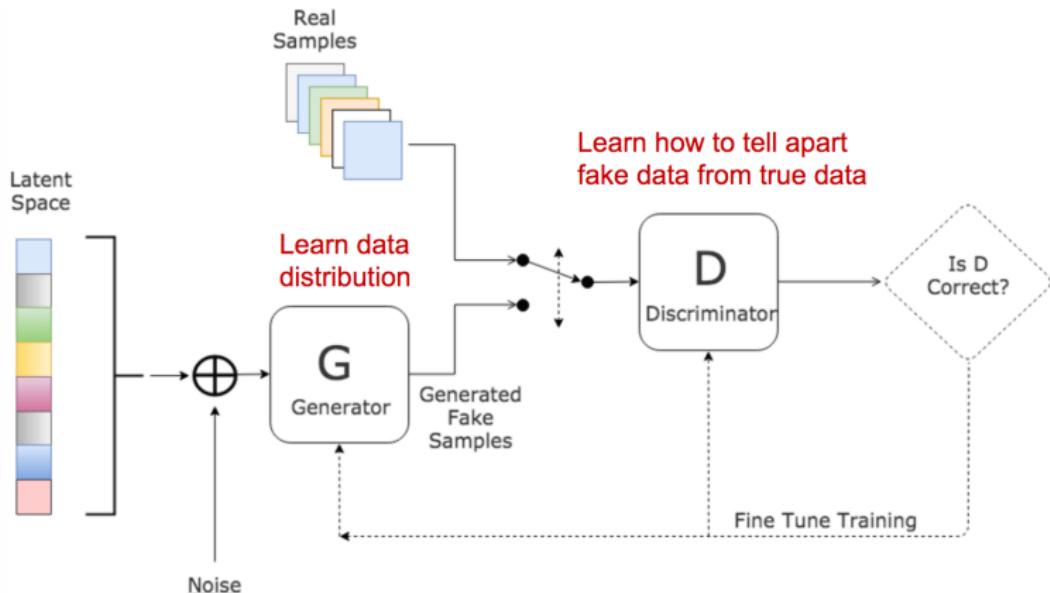


Figura: Arquitectura de un GAN. Imagen tomada de <https://www.kdnuggets.com/2017/01/generative-...-learning.html>.

Estos dos modelos compiten el uno contra el otro durante el proceso de entrenamiento: El generador G está tratando de engañar al discriminador, mientras que el modelo crítico D (discriminador) está tratando de no ser engañado. Este interesante **juego de suma cero** entre dos modelos motiva a ambos a mejorar sus funcionalidades.

Simbolo significado

p_z Distribución de datos sobre entrada de ruido z

p_g Distribución de los datos generados.

p_r Distribución de los datos reales x

Formalmente:

- ▶ Se debe asegurar que la decisión del discriminador D sobre un dato real sea lo mas precisa posible, eso se logra maximizando $\mathbb{E}_{x \sim p_r(x)}[\log D(x)]$. Al mismo tiempo, dado un dato falso $D(z), z \sim p_z(z)$, se espera que el discriminador arroje una probabilidad, $D(G(z))$, cercana a cero, esto se logra maximizando $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$.
- ▶ Se debe asegurar que el generador G sea entrenado para incrementar las posibilidades de que D produzca una alta probabilidad para una imagen falsa, esto se logra encontrando G que maximice $\mathbb{E}_{z \sim p_z(z)}[\log[1 - D(G(z))]]$.

Combinando ambos objetivos se concluye que D y G están jugando un **minimax game** el cual consiste en solucionar el siguiente problema de optimización:

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

donde

$$\begin{aligned} L(D, G) &= \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D(x))]. \end{aligned}$$

¿Cuál es el valor óptimo para D ?

Note que

$$L(D, G) = \int_X (p_r(x) \log(D(x)) + p_g(x) \log(D(x))) dx.$$

Almanado

$$y = D(x), \quad A = p_r(x), \quad B = p_g(x)$$

entonces los que está dentro de la integral es $f(y) = A \log(y) + B \log(1 - y)$ (podemos ignorar de forma segura la integral porque se muestrea x sobre todos los valores posibles), luego, ya que se sabe que para cualquier $(A, B) \in \mathbb{R}^2 \setminus (0, 0)$ la función $y \mapsto A \log(y) + B \log(1 - y)$ alcanza su máximo en $[0, 1]$ en $y = \frac{A}{A+B}$. Por lo tanto, el mejor valor del discriminador es

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}.$$

Una vez el generador es entrenado hasta su óptimo, p_g estará muy cerca de P_r . Cuando $p_g = p_r$, $D^*(x)$ se convierte en $\frac{1}{2}$

¿Cuál es el valor óptimo global?

Cuando ambos G y D están en sus valores óptimos se tiene $p_g = p_r$ y $D^*(x) = \frac{1}{2}$ y la función de perdida se convierte en:

$$\begin{aligned} L(D^*, G) &= \int_x (p_r(x) \log(D^*(x)) + p_g(x) \log(1 - D^*(x))) dx \\ &= \log\left(\frac{1}{2}\right) \int_x p_r(x) dx + \log\left(\frac{1}{2}\right) \int_x p_g(x) dx \\ &= -2 \log(2) \end{aligned}$$

¿Qué representa la función de perdida L ?

De acuerdo a lo anterior la **divergencia JS** entre p_g y p_r puede ser calculada como

$$\begin{aligned}
 D_{JS}(p_r \| p_g) &= \frac{1}{2} D_{KL}\left(p_r \| \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \| \frac{p_r + p_g}{2}\right) \\
 &= \frac{1}{2} \left(\log 2 \int_x p_r(x) \frac{p_r(x)}{p_r(x) + p_g(x)} dx \right) \\
 &\quad + \frac{1}{2} \left(\log 2 + \int_x p_g(x) \frac{p_g(x)}{p_r(x) + p_g(x)} dx \right) \\
 &= \frac{1}{2} (\log 4 + L(D^*, G)) .
 \end{aligned}$$

Así se tiene

$$L(D^*, G) = 2D_{JS}(p_r \| p_g) - 2\log 2$$

En conclusión, la función de perdida de un GAN cuantifica la similitud entre p_g y p_r por medio de D_{JS} cuando el discriminador es óptimo. El mejor G^* que replica la distribución de los datos reales alcanza el mínimo $L(G^*, D^*) = -2\log 2$.

El algoritmo

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:

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

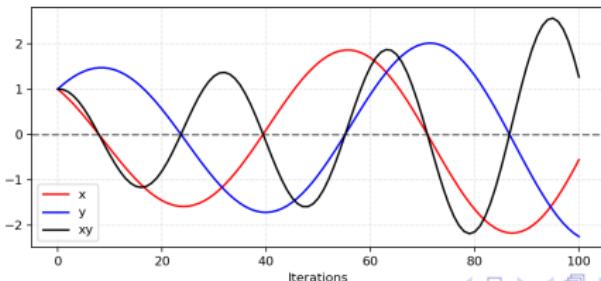
end for

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

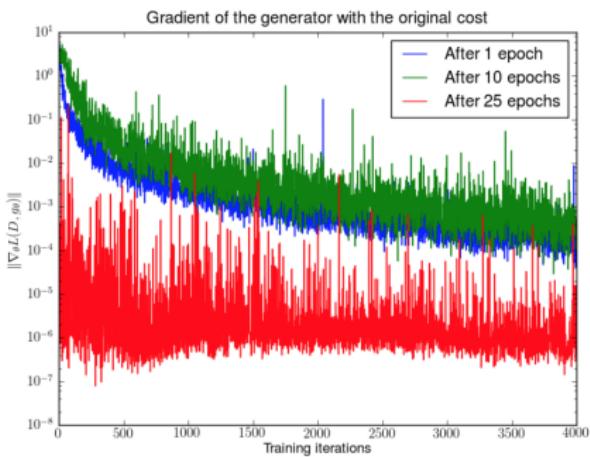
Algunas dificultades de los GAN's

- ▶ Es difícil alcanzar el equilibrio de Nash (Salimans et al. (2016)): El problema radica en que se emplea método del gradiente descendente como procedimiento de entrenamiento. Por ejemplo, considere $f(x, y) = xy$ y el problema $\min_x \max_y f(x, y)$. Un jugador toma el control de x para minimizar $f_1(x, y) = xy$ mientras otro toma el control de y para minimizar $f_2(x, y) = -xy$.

Ya que $\frac{df_1}{dx} = y$ y $\frac{df_1}{dy} = -x$, se actualiza x con $x - \eta y$ y y con $y = y + \eta x x$ simultáneamente en una iteración. Este procedimiento no permite llegar a un equilibrio.



- **Desvanecimiento del gradiente:** Si se deja el generador fijo y se optimiza solo el discriminador se presenta lo siguiente:



- Si el discriminador se comporta mal, el generador no tiene retroalimentación precisa y la función de pérdida no puede representar la realidad.
- Si el discriminador hace un gran trabajo, el gradiente de la función de pérdida se reduce a casi cero y el aprendizaje se vuelve muy lento o incluso atascado.

Este dilema es claramente capaz de hacer que el entrenamiento GAN sea muy difícil.

Wasserstein GAN (WGAN) (Arjovsky, 2017)

Consiste en cambiar la función de perdida GAN por la distancia de Wasserstein entre p_r y p_g .

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y) \sim \gamma} (\|x - y\|).$$

Por la dualidad de Kantorovich-Rubinstein se tiene:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r} [f(x)] - \mathbb{E}_{x \sim p_g} [f(x)]$$

para cualquier $K \geq 0$.

Suponga que estas funciones f vienen de una familia parametrizada de funciones K -Lipschitz, $\{f_w\}_{w \in W}$. Entes modificado Wasserstein-GAN, el modelo “*discriminador*” es usado para aprender w y asi encontrar una buena f_w y la función de perdida es configurada como:

$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

donde g_θ es el generador (una red neuronal con parámetro θ). En realidad f_w también es una red neuronal con parámetro w que es K -Lipschitz respecto a su input.

El gran problema es **mantener la continuidad K -Lipschitz de f_w** durante el entrenamiento para que todo funcione. Para solucionarlo se usa un truco simple pero muy práctico: después de cada actualización de gradiente, fije los pesos w a una ventana pequeña, como $[-0.01, 0.01]$, lo que da como resultado un espacio de parámetros compacto W .

El algoritmo

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

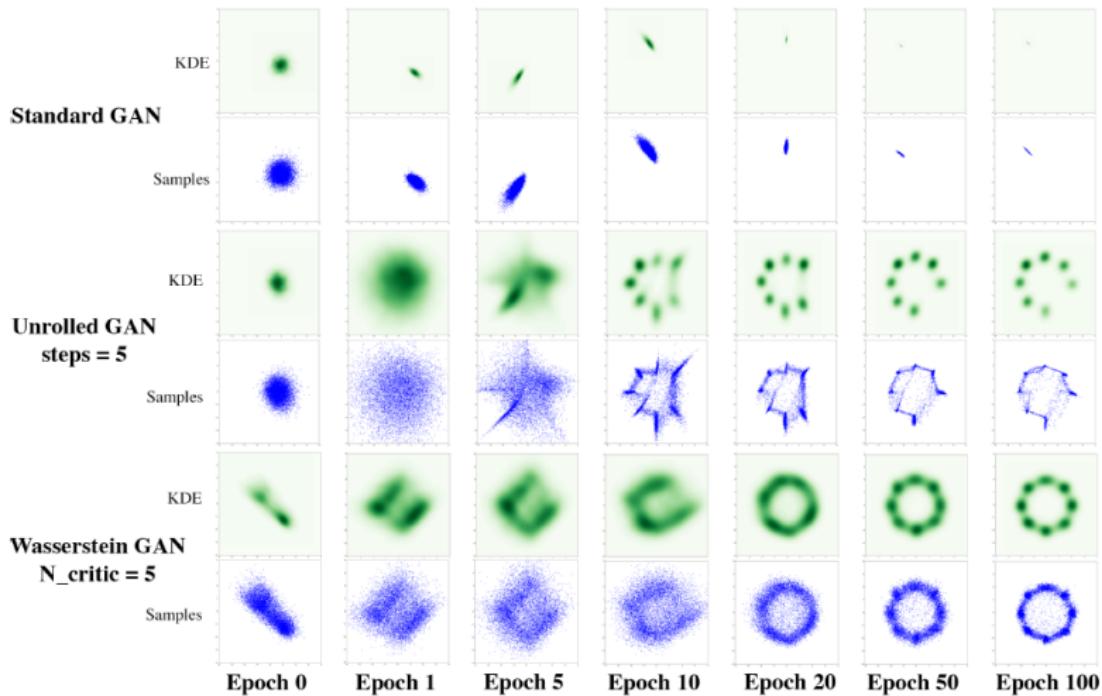
Require: : α , the learning rate. c , the clipping parameter. m , the batch size.
 n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```

Ejemplo: Comparando diferentes métodos de aprender una mezcla de 8 distribuciones gaussianas esparcidas en un círculo.



Dificultades

El Wasserstein GAN no es perfecto. El problema del algoritmo radica en el ajuste de los pesos w (weight clipping) para asegurar la K -Lipschitz continuidad, la respecto los mismos autores del artículo original (Arjovsky, 2017) mencionan:

“Weight clipping is a clearly terrible way to enforce a Lipschitz constraint”

WGAN aun sufre de un entrenamiento inestable, convergencia lenta después de ajustar los pesos w (cuando la ventana $[-c, c]$ es muy grande), y desvanecimiento del gradiente (cuando la ventana es muy pequeña).

¿Comó solucionar el problema de la ventana de ajuste?

Para responder la pregunta se requieren los siguiente resultados

Proposición 1

Sean p_r y p_g dos distribuciones en espacio métrico compacto \mathcal{X} . Entonces existe una 1-Lipschitz función f^ la cual es la solución óptima de $\max_{\|f\|_L \leq 1} \mathbb{E}_{y \sim p_r}[f(y)] - \mathbb{E}_{x \sim p_g}[f(x)]$. Sea π el coupling óptimo entre p_r y p_g definido como el maximizador de $W(p_r, p_g) = \inf_{\pi \in \Pi(p_r, p_g)} \mathbb{E}_{(x,y) \sim \pi} [\|x - y\|]$ donde $\Pi(p_r, p_g)$ es el conjunto de distribuciones conjuntas cuyas marginales son p_g y p_g . Entonces, si f^* es diferenciable, $\pi(x = y) = 0$ y $x_t = tx + (1 - t)y$ con $0 \leq t \leq 1$, se cumple que $\mathbb{P}_{(x,y) \sim \pi} \left[\nabla f^*(x_t) = \frac{y - x_t}{\|y - x_t\|} \right] = 1$.*

Corolario 2

f^* tiene gradiente con norma 1 en casi todas parte bajo p_r y p_g .

Teniendo en cuenta lo anterior, una solución es propuesta en [4] (Gulrajani, 2017) la cual consiste en imponer una penalidad en el gradiente lo cual genera la siguiente función de costo

$$L = \underbrace{\mathbb{E}_{\tilde{x} \sim p_g}[D(\tilde{x})] - \mathbb{E}_{x \sim p_r}[D(x)]}_{\text{costo original}} + \underbrace{\lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]}_{\text{penalidad en gradiente}}$$

donde $p_{\hat{x}}$ define una distribución uniforme a lo largo de la linea recta entre pares de puntos muestreados de la distribución p_r y p_g , es decir, \hat{x} es muestreado (sampleado) de \tilde{x} y x con t uniformemente muestreado entre 0 y 1, $\hat{x} = t\tilde{x} + (1 - t)x$.

Algoritmo de WGAN-GP

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ , the number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyperparameters α, β_1, β_2 .

Require: initial critic parameters w_0 , initial generator parameters θ_0 .

```

1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $x \sim \mathbb{P}_r$ , latent variable  $z \sim p(z)$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{x} \leftarrow G_\theta(z)$ 
6:        $\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\tilde{x}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{z^{(i)}\}_{i=1}^m \sim p(z)$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

CycleGAN (Zhu, 2018)

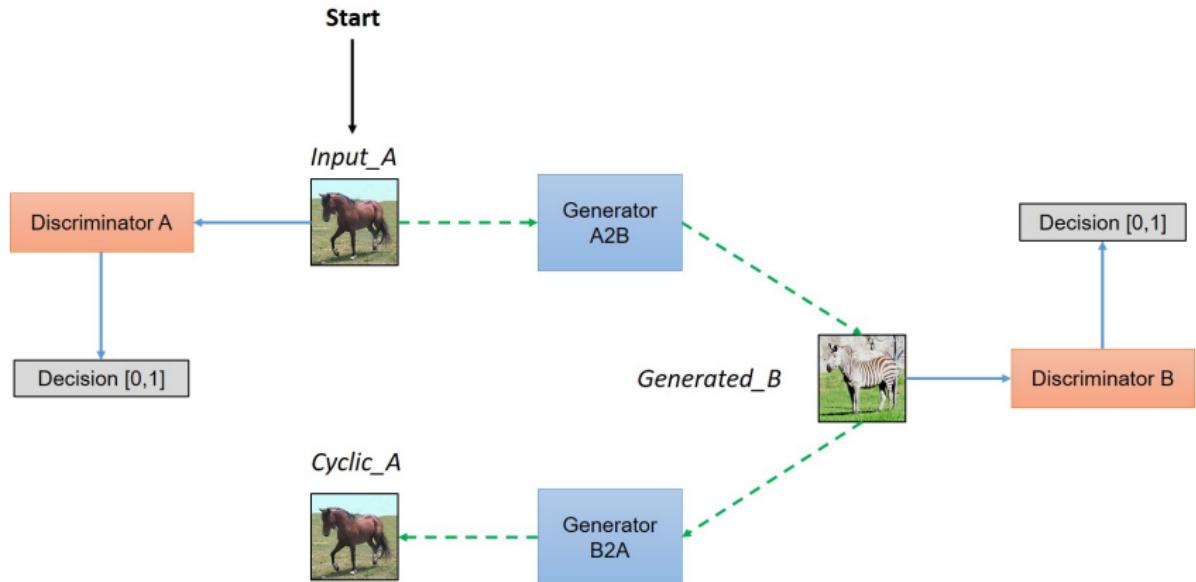
El objetivo es generar un mapa entre dos dominios dados X y Y (por ejemplo, X imágenes de caballos, Y imágenes de cebras) dado muestras de entrenamiento $\{x_i\}_{i=1}^N$ donde $x_i \in X$ y $\{y_j\}_{j=1}^M$ donde $y_j \in Y$. Denotamos la distribución real de los datos x como p_{r_x} y la de los datos y como p_{r_y} .

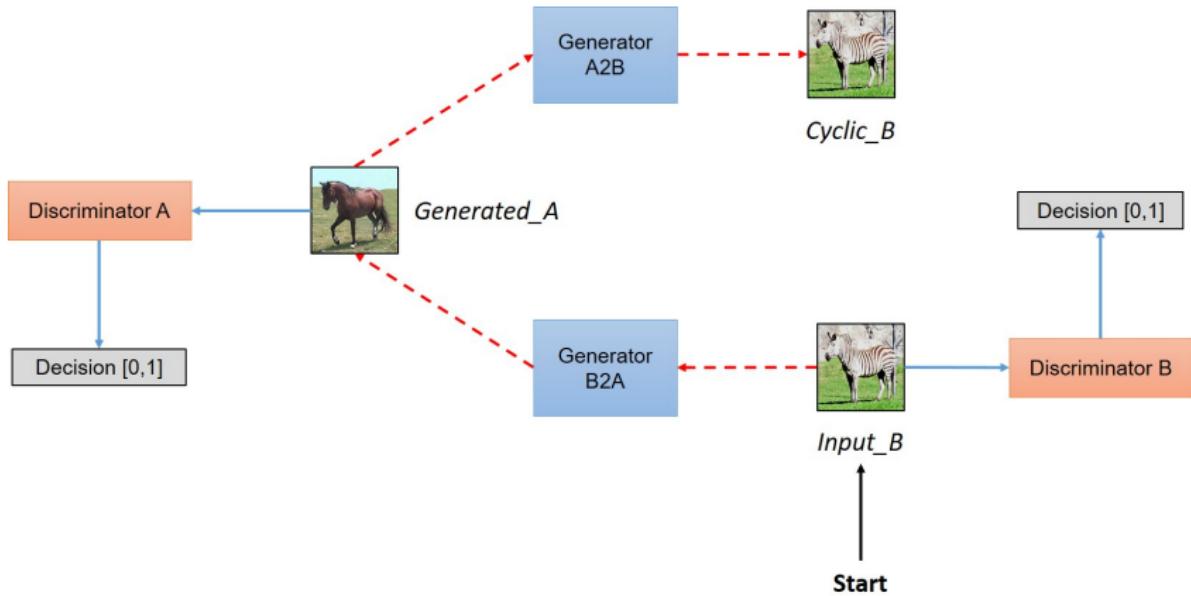
El modelo que se propone incluye dos mapas $G : X \rightarrow Y$ y $F : Y \rightarrow X$. Adicionalmente se incluyen discriminadores D_X y D_Y , donde D_X intenta distinguir entre imágenes x e imágenes trasladadas $F(y)$, simultáneamente D_Y intenta distinguir entre imágenes y e imágenes trasladadas $G(x)$.

El objetivo se resume en dos términos:

- ▶ **Adversarial losses:** Hacer coincidir la distribución de las imágenes generadas con la distribución de datos en el dominio objetivo.
- ▶ **Cycle consistency losses:** Evitar que los mapas G y F aprendidos en el proceso se contradigan entre sí.

Esquema





La arquitectura de un CycleGAN se compone de las dos imágenes anteriores.

¿Como es la función de perdida para un CycleGAN?

Es la suma de tres funciones de perdida:

- ▶ **Adversarial losses:** De estas se tiene dos funciones:

$$L_{\text{GAN}}(G, D_Y) = \mathbb{E}_{y \sim p_{r_y}} [\log(D_Y(y))] + \mathbb{E}_{x \sim p_{r_x}} [\log(1 - D_Y(G(x)))] ,$$

$$L_{\text{GAN}}(F, D_X) = \mathbb{E}_{y \sim p_{r_y}} [\log(D_X(y))] + \mathbb{E}_{x \sim p_{r_x}} [\log(1 - D_X(G(x)))] .$$

- ▶ **Cycle consistency losses:**

$$L_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{r_x}} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{r_y}} [\|G(F(y)) - y\|_1] .$$

La **función de perdida completa** es

$$L(G, F, D_X, D_Y) = L_{\text{GAN}}(G, D_Y) + L_{\text{GAN}}(F, D_X) + \lambda L_{\text{cyc}}(G, F) .$$

El problema de optimización a solucionar es

$$\min_{G, F} \max_{D_X, D_Y} L(G, F, D_X, D_Y) .$$

Monet **Photos**

Monet → photo

Zebras **Horses**

zebra → horse

Summer **Winter**

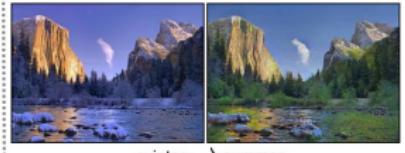
summer → winter



photo → Monet



horse → zebra



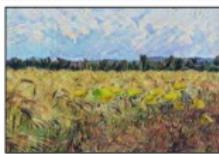
winter → summer



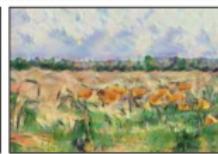
Photograph



Monet



Van Gogh



Cezanne



Ukiyo-e

Gracias por su atención.

Referencias I

-  Goodfellow, I. Pouget-Abadie, J. Mirza, M.
Generative adversarial nets.
NIPS, 2014.
-  Arjovsky, M. Bottou, L.
Towards principled methods for training generative adversarial networks.
arXiv preprint arXiv:1701.07875, 2017.
-  Arjovsky, M. Chintala, S. Bottou, L.
Wasserstein gan.
International Conference on Learning Representations, 2017.
-  Gulrajani, i. Ahmed, F. Arjovsky, M. Dumoulin, V. Courville, A.
Improved Training of Wasserstein GANs.
arXiv:1704.00028v3, 2017.

Referencias II

-  Zhu, J. Park, T. Isola, P. Efros, A.
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.
arXiv:1703.10593v5, 2018.