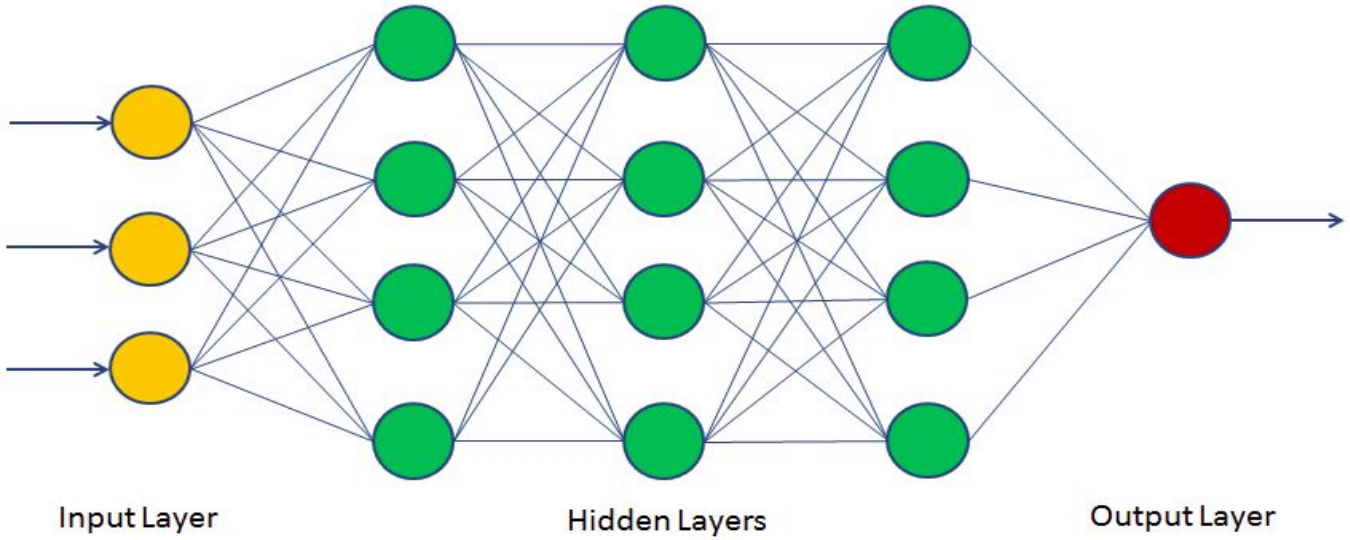
GAN: Generative Adversarial Networks

José María Verde
Seminario (I+A)A

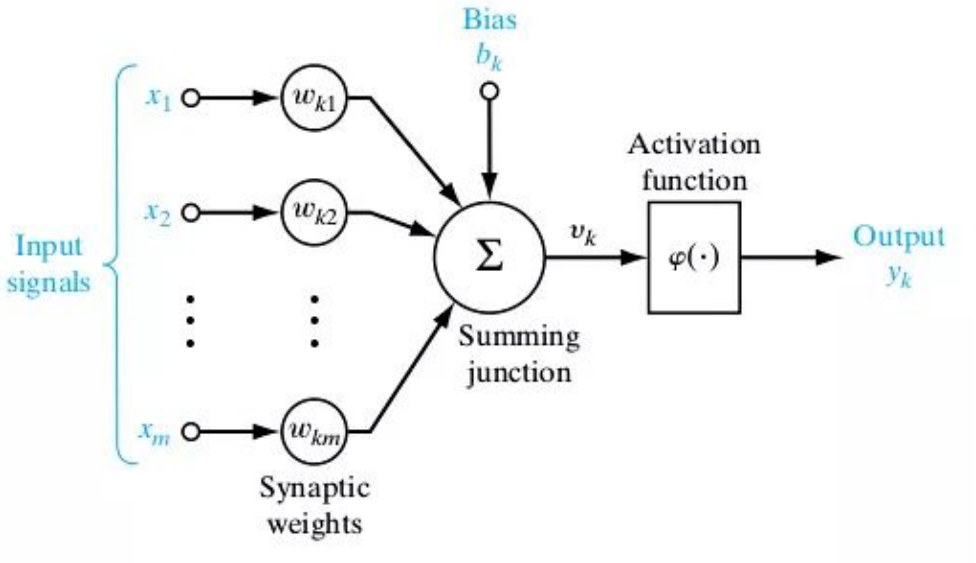
Antes de empezar, ¿qué es una red neuronal?

- Modelo de Aprendizaje Automático.
- Permite detectar patrones complejos.
- Adaptable para realizar todo tipo de tareas.
- Consta de varias capas de neuronas conectadas cada una a la siguiente.
- Cada neurona realiza una operación sobre los datos que recibe y transmite un resultado.

Perceptrón multicapa



La neurona


Bias
b_k
x_1
x_2
x_m
w_{k1}
w_{k2}
w_{km}
Input signals
Synaptic weights
Σ
Summing junction
v_k
Activation function
$\varphi(\cdot)$
Output
y_k

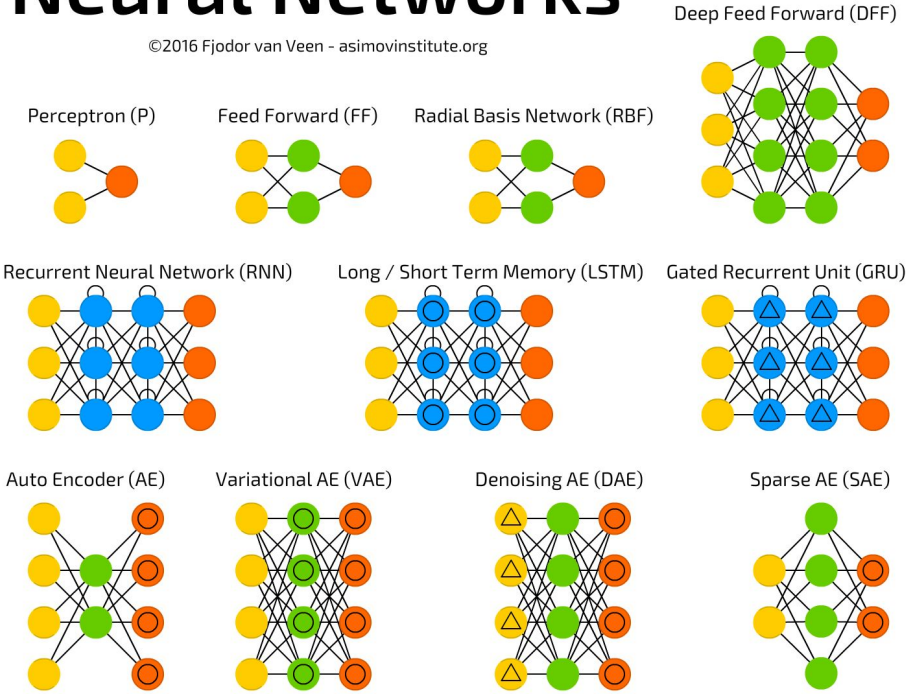
¿Cómo funciona?

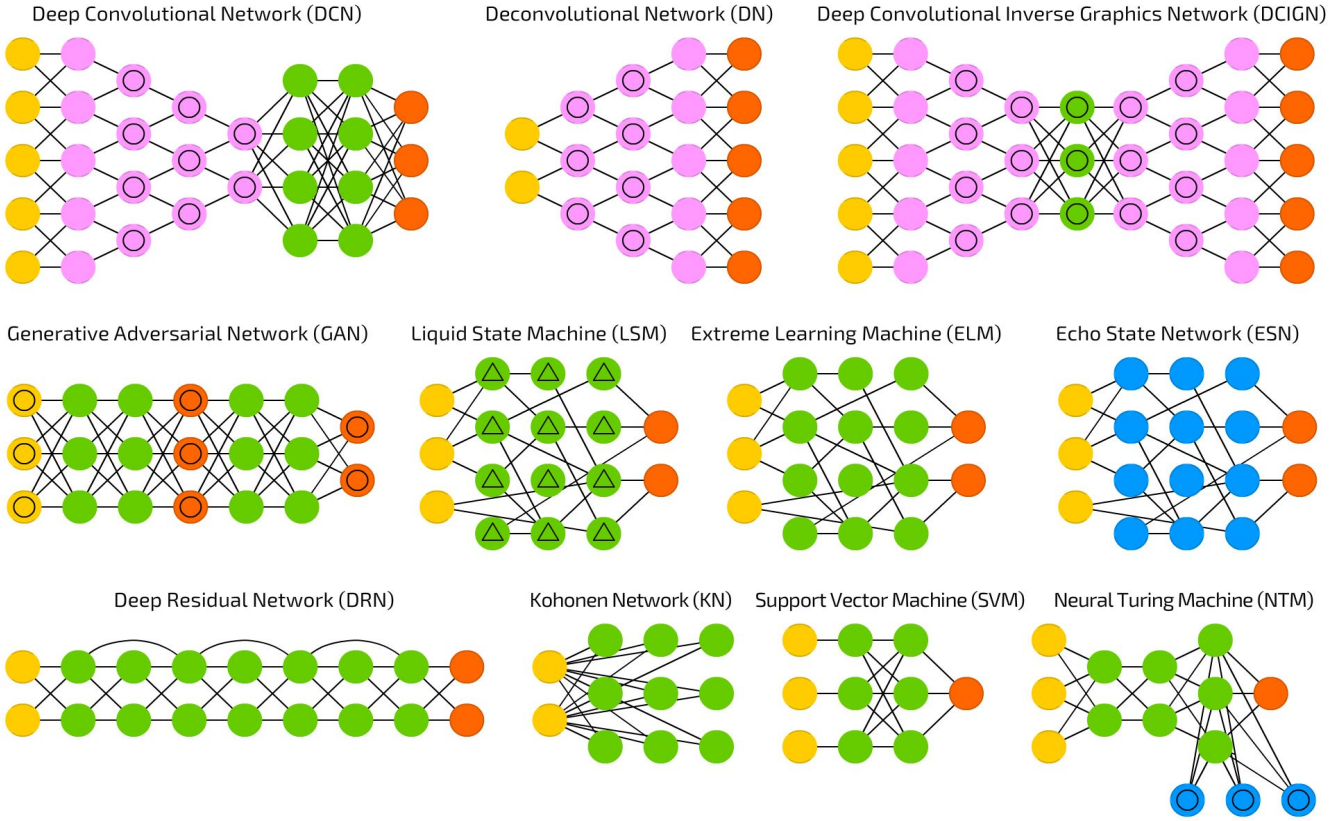
- Conjuntos de datos: (entrada, salida).
- Backpropagation.
- Autorregulación de pesos.

A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



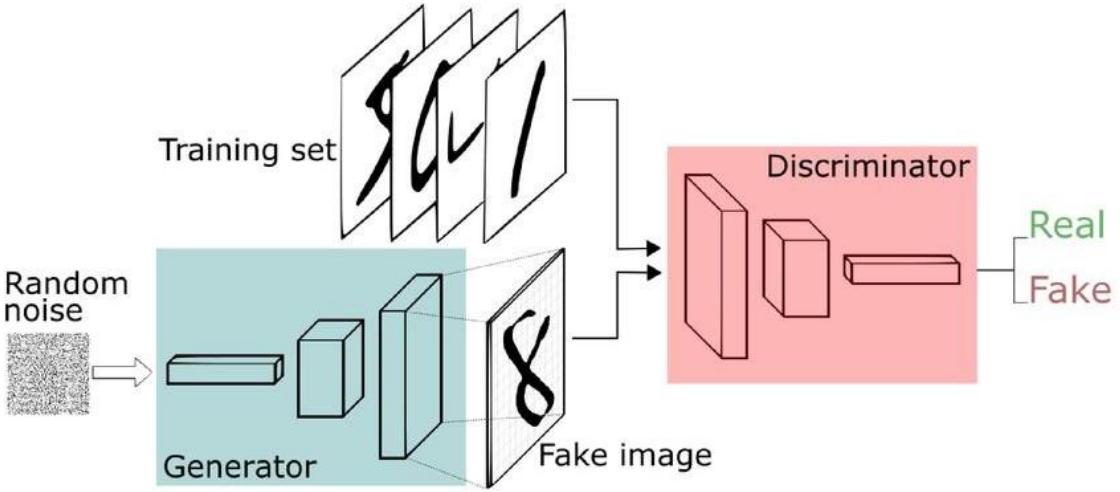

Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

¿Qué es una GAN?

- Red generativa antagónica.
- N = Red neuronal.
- G = Genera cosas.
- A = Aprendizaje adversario.

Juego de suma cero

¿Cómo se hace esto con GAN?



Funciones a optimizar

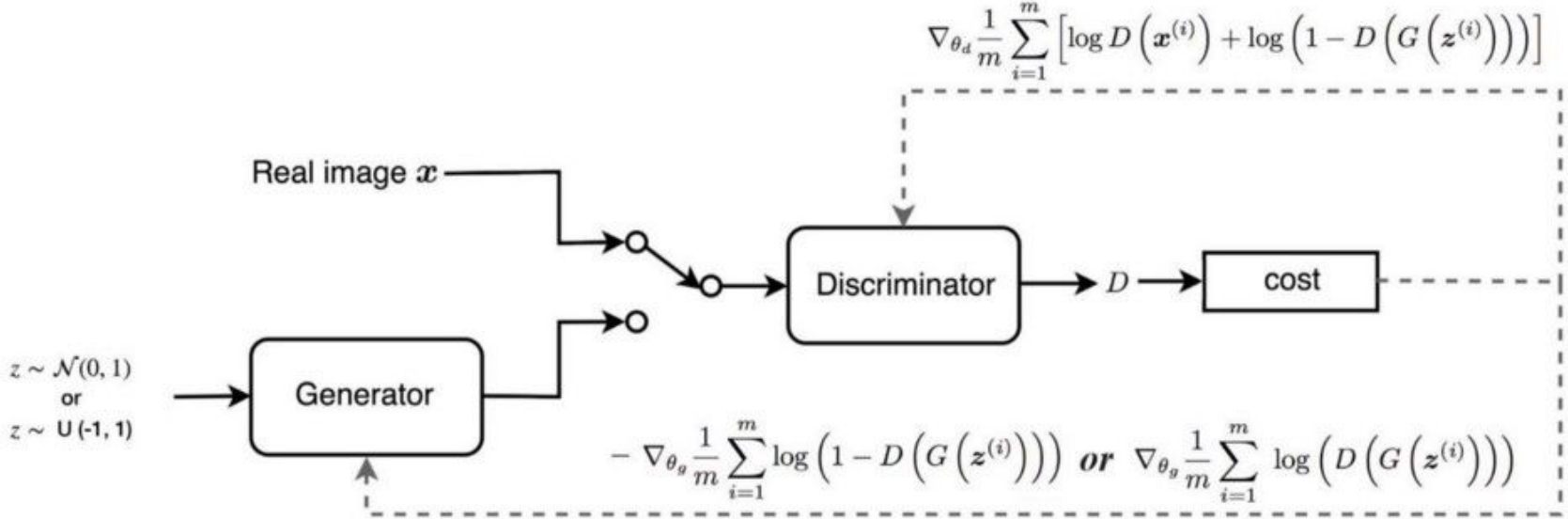
$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

recognize real images better

recognize generated images better

$$\min_G V(G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Optimize G that can fool the discriminator the most.

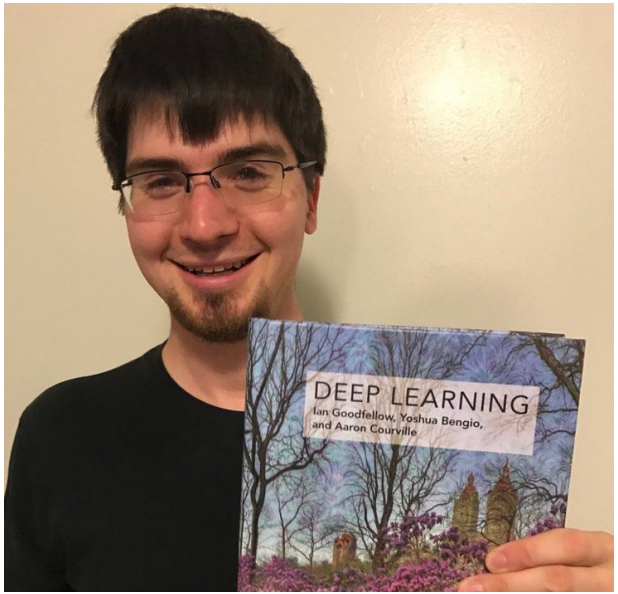

$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$
Real image $\boldsymbol{x}$
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$ **or** $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie[*], Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair[†], Aaron Courville, Yoshua Bengio[‡]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, 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 $\frac{1}{2}$ 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.



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

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

  end for

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

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

end for

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

Dado un generador G* fijado, el valor que tiene la función de coste para el discriminador D óptimo es:

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

La métrica importa

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. □

Divergencias

$$D_{KL}(P \parallel Q) = -\sum_{x} P(x) \log(\frac{Q(x)}{P(x)})$$

$$D_{JS}(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel \frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q \parallel \frac{P+Q}{2})$$

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. □

No todo es tan bonito como parece

ON DISTINGUISHABILITY CRITERIA FOR
ESTIMATING GENERATIVE MODELS

Ian J. Goodfellow
Google Inc., Mountain View, CA
goodfellow@google.com

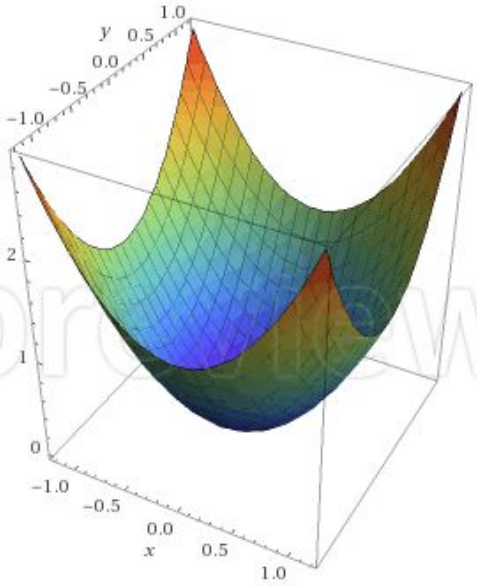
- The existing theoretical work on GANs does not guarantee convergence on practical applications.

¿Por qué?

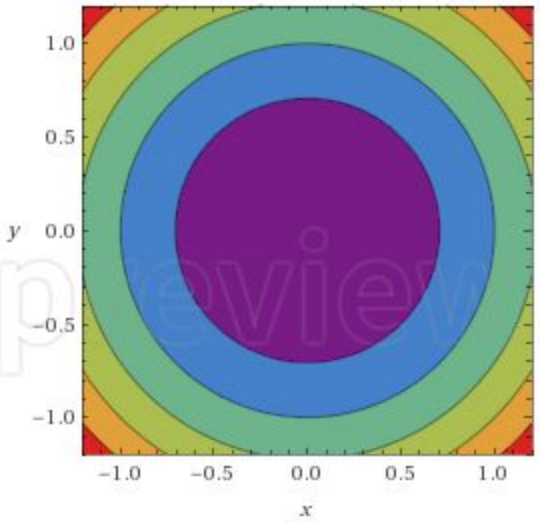
- El resultado teórico emitido por Goodfellow hablaba de convexidad en un espacio de funciones.
- Las GAN, como todas las redes neuronales, realizan optimización en un espacio PARAMÉTRICO dentro de ese espacio de funciones.
- Cuando reducimos el problema a optimización en un espacio paramétrico no se asegura la convexidad, por lo que no hay resultados teóricos a los que agarrarse.

No se deben tomar a la ligera a las matemáticas.

Un ejemplo visual



Computed by Wolfram|Alpha

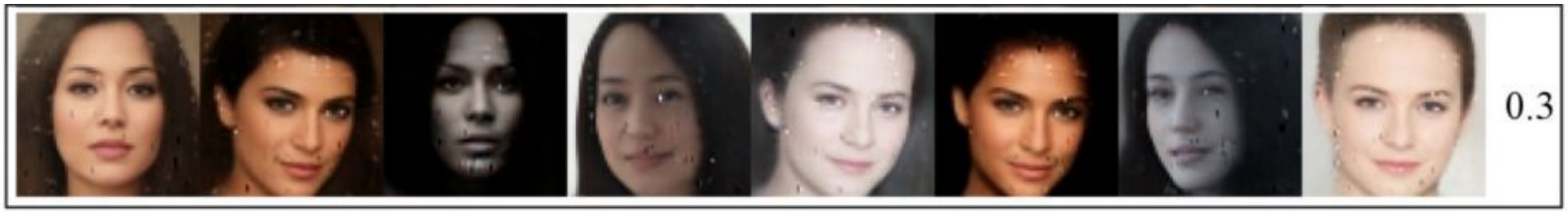


Computed by Wolfram|Alpha

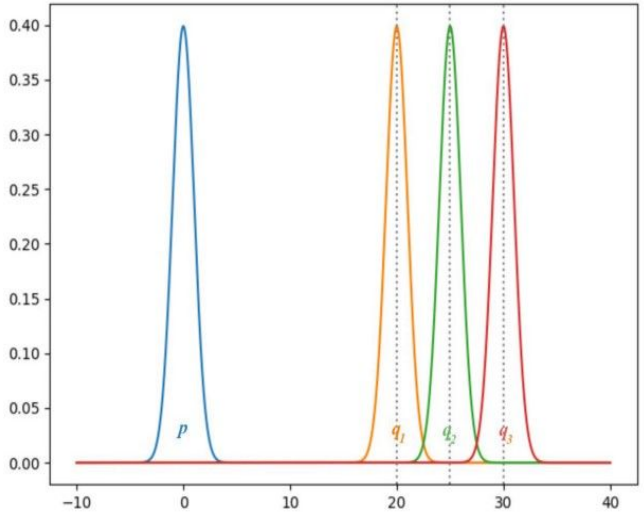
No funciona la teoría, pero tampoco la práctica

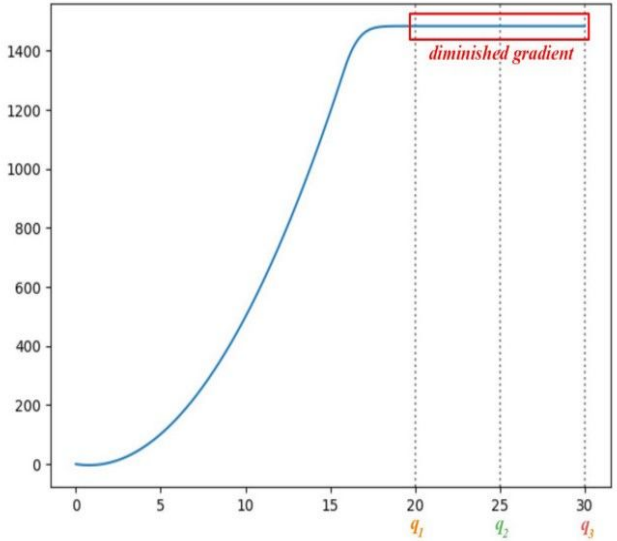
- Oscilaciones de la red alrededor de una solución sin llegar a converger.
- Colapso del generador en unos pocos de ejemplos.
- Más fácil detectar que aprender: el gradiente del generador no dice nada.
- Una de las redes ha evolucionado mucho mejor (overfitting).

Colapso del generador



Gradiente nulo





Cuatro variables gaussianas de distinta media y divergencia JS con respecto a p.

¿Y si probamos con otra métrica?

Wasserstein GAN

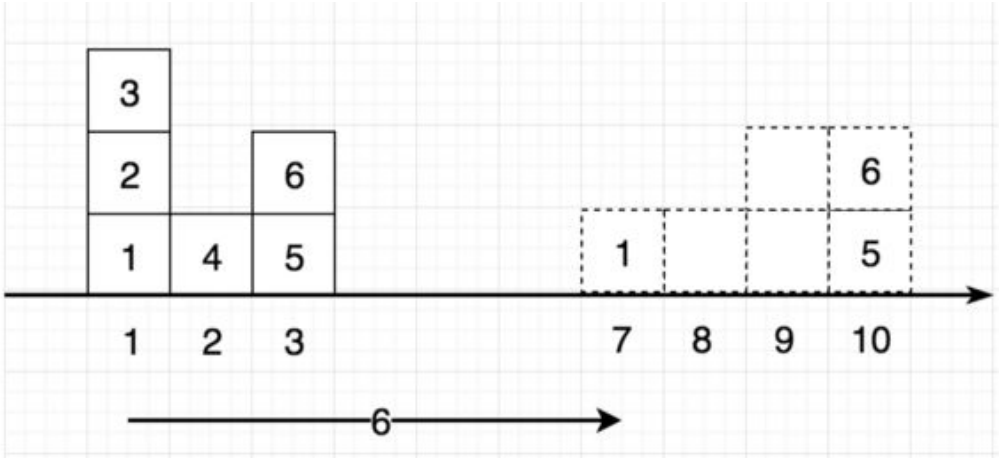
Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

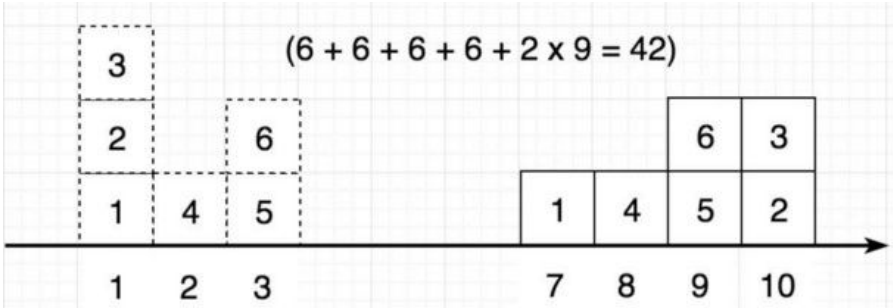
[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

Distancia Earth Mover

¿Cómo podemos transportar la pila de cajas de un lado a otro?

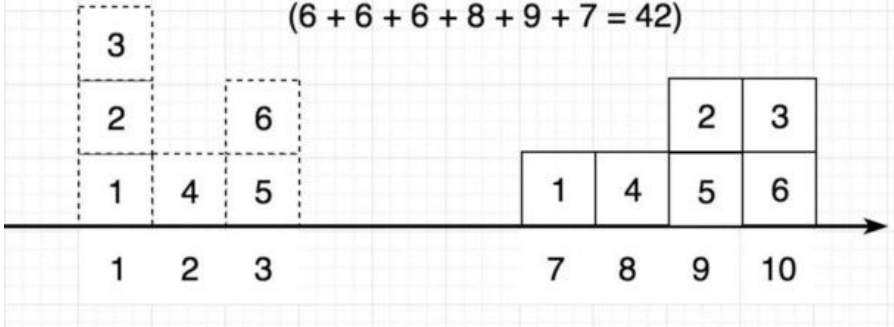
¿Cuál es la forma óptima?





γ_1

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0



γ_2

	7	8	9	10
1	1	0	1	1
2	0	1	0	0
3	0	0	1	1

Más formalmente:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$$

La distancia Wasserstein (o distancia Earth Mover) es el coste del transporte óptimo de una distribución de probabilidad en otra, donde $\mathbb{\Pi}(\mathbb{P}_r, \mathbb{P}_g)$ denota todas las posibles distribuciones de probabilidad conjunta de $\mathbb{P}_r$ y $\mathbb{P}_g$.

¿Qué ventaja tiene esto?

Vamos a comparar distancias

- Sea *X* un espacio de medida compacto (por ejemplo, imágenes de 100x100 píxeles).
- Sea Σ el conjunto de todos los subconjuntos de Borel de *X* (todos los subconjuntos).
- Sean $\mathbb{P}_r$ y $\mathbb{P}_g$ dos distribuciones del espacio de medidas de probabilidad, Prob(*X*).

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x)$$

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

  where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

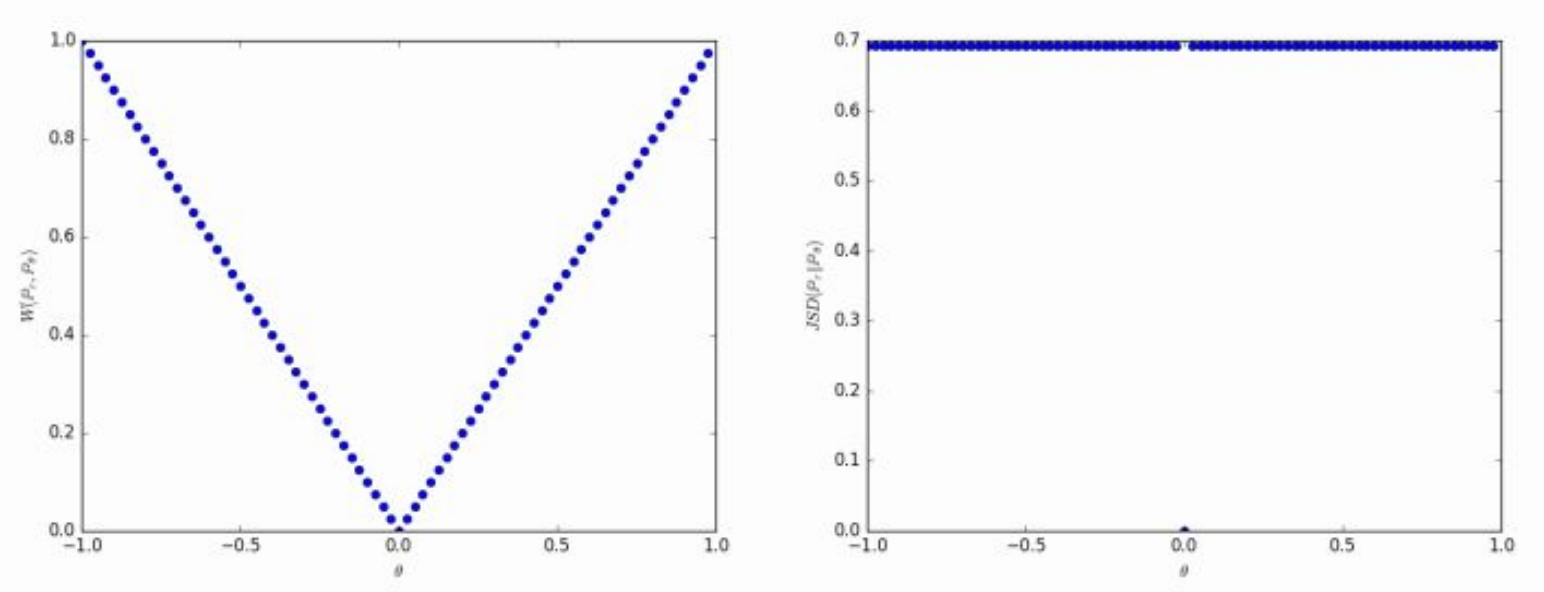
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\ \|x - y\| \ \right]$$

Un ejemplo a pequeña escala

Sea Z ~ U[0,1], sea $\mathbb{P}_0$ el vector aleatorio (0,Z) y sea $\mathbb{P}_\theta$=(θ,Z) otro vector aleatorio parametrizado. Entonces, tenemos que:

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

No existe convergencia en la JS pero sí en la Wasserstein, de hecho el gradiente de la JS es nulo, lo cual explica el problema anterior.



Theorem 1. *Let $\mathbb{P}_r$ be a fixed distribution over $\mathcal{X}$. Let Z be a random variable (e.g Gaussian) over another space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a function, that will be denoted $g_\theta(z)$ with z the first coordinate and θ the second. Let $\mathbb{P}_\theta$ denote the distribution of $g_\theta(Z)$. Then,*

1. *If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.*
2. *If g is locally Lipschitz and satisfies regularity assumption 1, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere.*
3. *Statements 1-2 are false for the Jensen-Shannon divergence $JS(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs.*

Función Lipschitziana

Sea f una función y S un conjunto, f es K-Lipschitziana en S si:

$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|, \forall x_1, x_2 \in S$$

Un pequeño problema: complejidad

Calcular la expresión anterior de la distancia Wasserstein es poco rentable en términos de complejidad, pero por suerte hay un atajo.

Utilizando la dualidad de Kantorovich-Rubinstein (1958) obtenemos una expresión equivalente y adaptable a un discriminador:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

donde el supremo se calcula sobre todas las funciones 1-Lipschitz.

Traduciendo a GAN

La función de coste anterior quedaría como:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$$

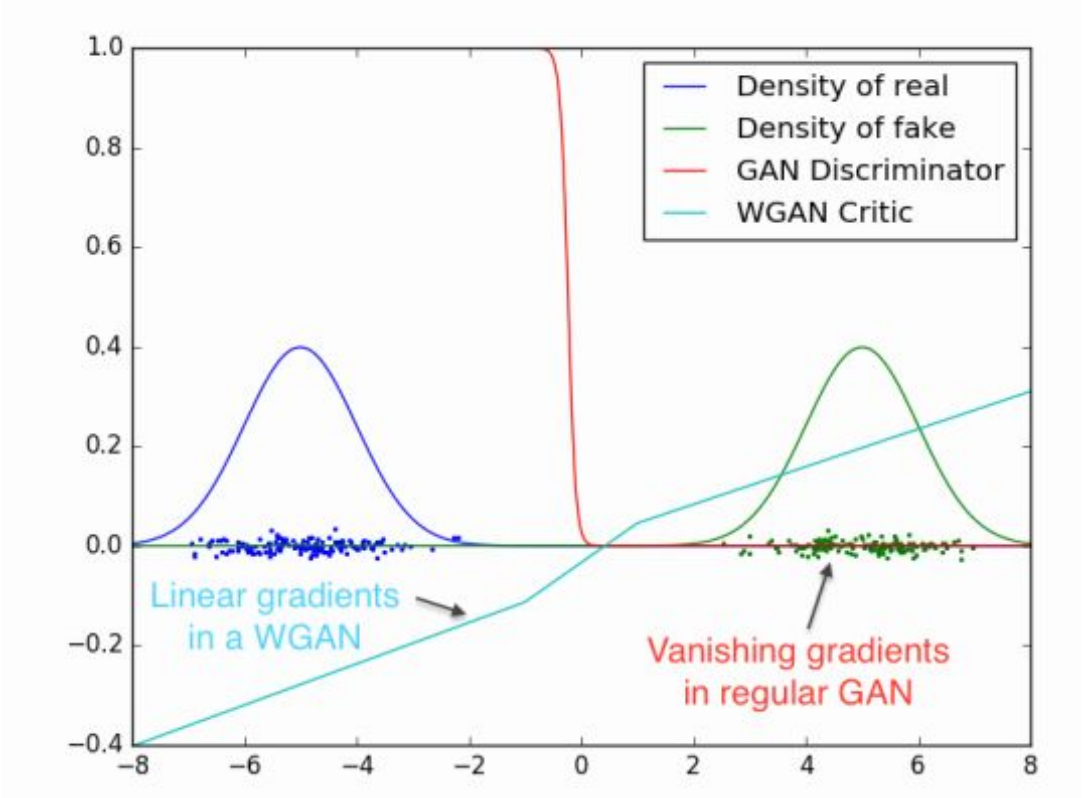
donde W es el espacio en el que se optimizan los parámetros del discriminador y f es 1-Lipschitz (se asegura haciendo clipping de pesos).

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.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while


Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients
in a WGAN
Vanishing gradients
in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

¿Y los resultados?

Improved Training of Wasserstein GANs

Ishaan Gulrajani[1]*, **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow

- Toda la teoría de las WGAN está bien construida, pero el uso del clipping de pesos no convence.
- El clipping provoca que la red no se comporte bien: gradientes nulos y aprendizaje de funciones demasiado simples.

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.

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

Diferencias

- Introducimos a la función de coste una penalización cuando el gradiente se aleje de 1 (una función es 1-Lipschitz si sus gradientes son siempre menores o iguales que uno).
- Se elimina el clipping de pesos.
- En lugar de forzar que sea 1-Lipschitz en todo el espacio, se fuerza entre los datos generados y los datos reales (sampling).

¿Es esta la actualidad?

Nos hemos centrado en la evolución de la función de coste de las GAN, su entrenamiento y su convergencia, pero existen multitud de arquitecturas diferentes, dependiendo del objetivo.

¿Pero y las imágenes del principio?

NVIDIA, 2018

PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Tero Karras NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA
Jaakko Lehtinen NVIDIA and Aalto University

- Debido a lo costoso que es entrenar una GAN de cero, la red empieza generando imágenes de 4x4 y va aumentando la resolución: 8x8, 16x16… hasta 1024x1024.
- Utiliza como función de coste la WGAN-GP en la mayoría de sus experimentos.
- Pero esto no acaba aquí...

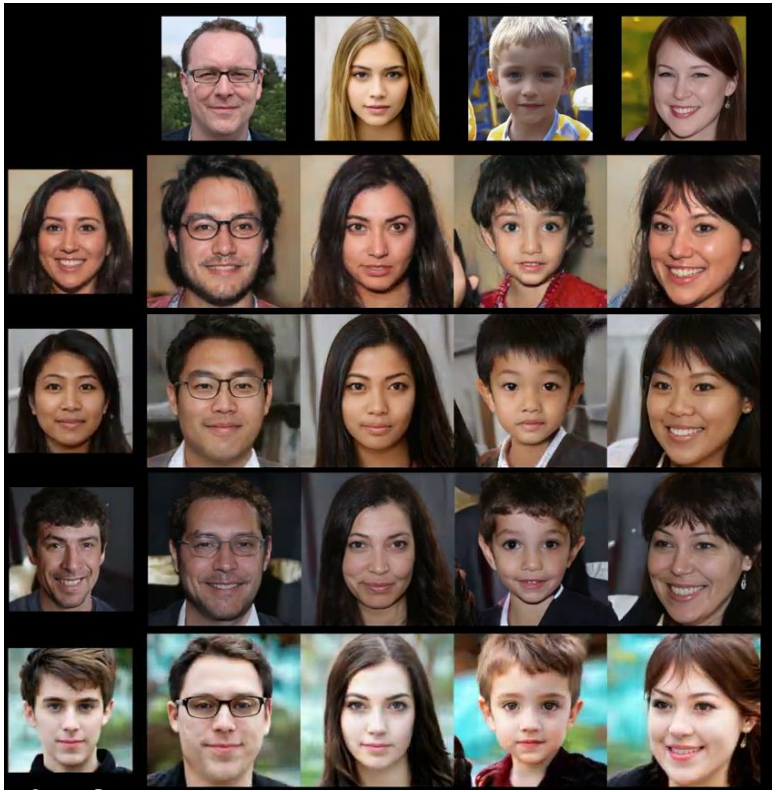
NVIDIA, febrero de 2019 (StyleGAN)

A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras
NVIDIA

Samuli Laine
NVIDIA

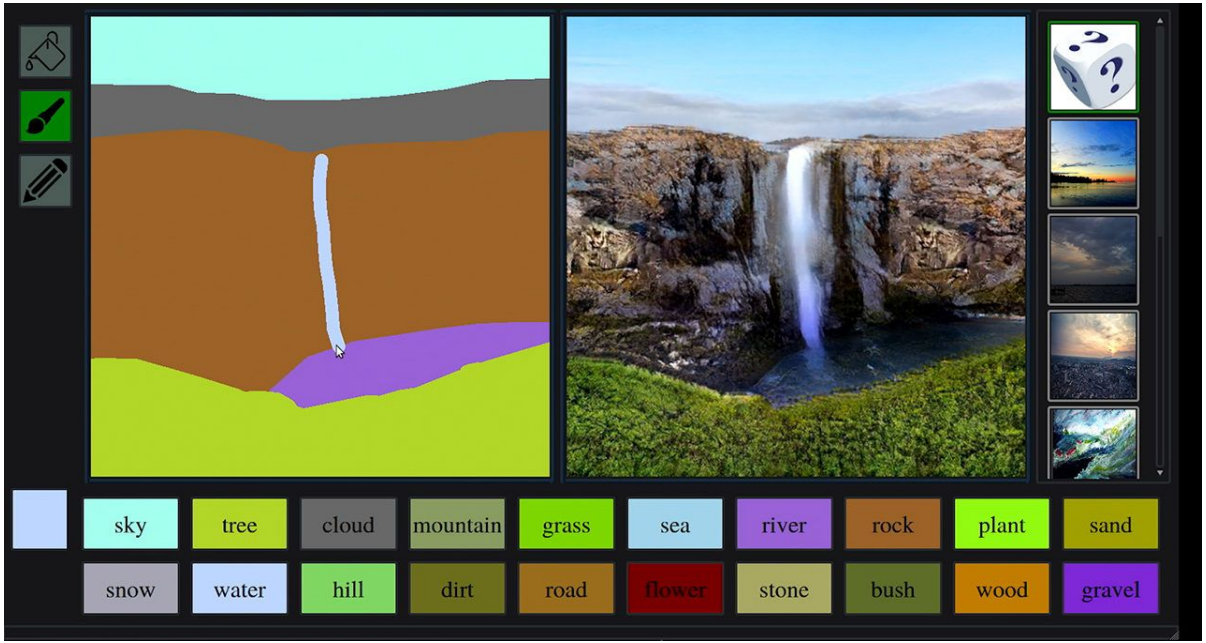
Timo Aila
NVIDIA



No se limita a caras...

GauGAN, Síntesis semántica

https://www.youtube.com/watch?v=NZBN9jRJ-FE



SC-FEGAN, edición de fotos

https://www.youtube.com/watch?v=lEIFHbOOZh8

¿The End?
No, esto es solo el principio

Dudas, sugerencias, comentarios