

Redes Generativas Adversariales

Generative Adversarial Nets - Goodfellow et al. (2014)



Prof. Ricardo Nanculef & Ignacio Loayza - Departamento de Informática UTFSM

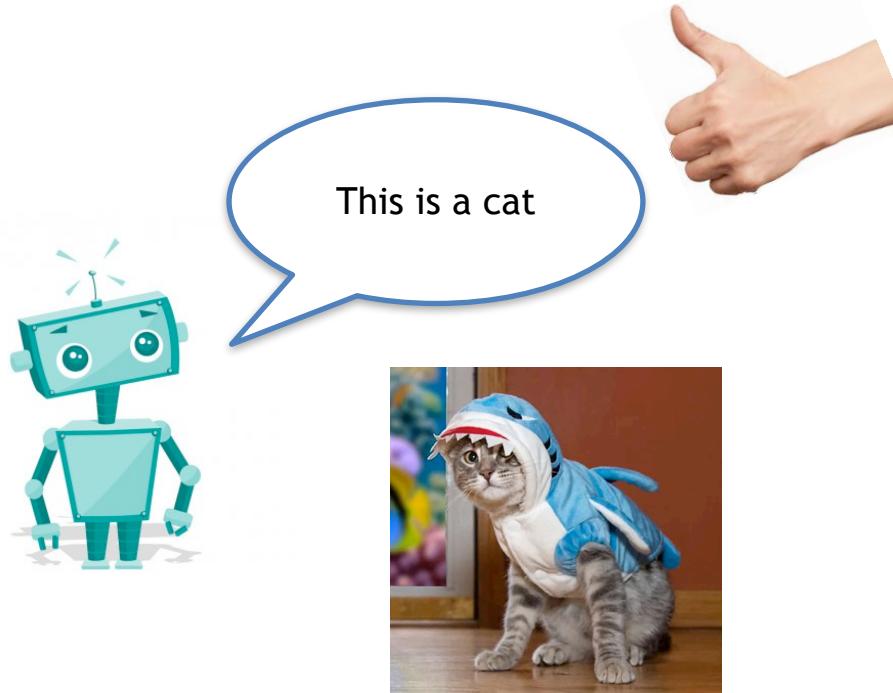
Contexto y motivación

Esa paper es del 2014, en aquellos tiempos el fuerte eran los modelos discriminativos:

2009: Lanzamiento de ImageNet (acá hay un gato, acá no hay perro, etc.)

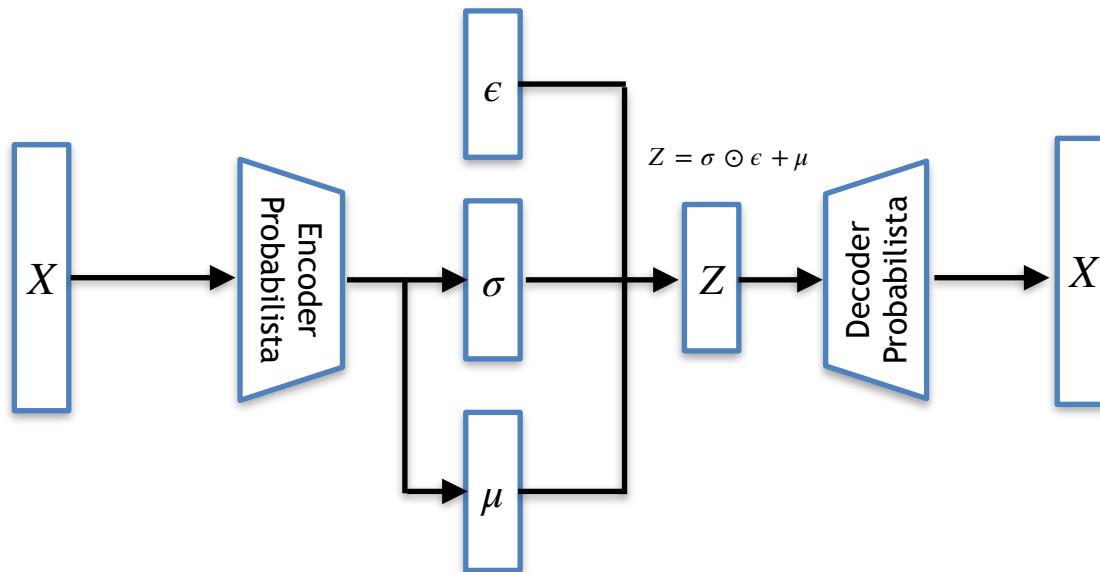
2012: Recognizing Cats on Youtube - Andrew Ng & Jeff Dean (ambos de Google Brain)

2014: Leap in Face Recognition - Investigadores de Facebook publican DeepFace (Identificador de rostros con 97.35% de accuracy)



Contexto y motivación

Hasta ese momento los modelos generativos tendían fuertemente a modelos más "clásicos" (Modelos de Markov, etc) y recientemente se había estado experimentando con modificaciones al modelo autoencoder para formular lo que ahora conocemos como VAE (el paper es del año anterior a GAN).



Contexto y motivación

Las GAN nacen formalmente con el trabajo de Goodfellow y compañía titulado "**Generative Adversarial Nets**" publicado el 2014 en NIPS (conferencia famosa, revísenla!).

Este tipo de arquitectura, como el mismo nombre lo dice, es generativa eso quiere decir que apunta a aprender $P(X, Y)$ en su espacio latente.

La idea principal es tener dos modelos: Un generador G que se encarga de generar datos de una distribución que debe ser aprendida para parecerse lo más posible a la distribución de los datos de entrenamiento, por otro lado, un modelo discriminador D se encarga de discernir si un determinado elemento fue generado por el generador G (o en realidad, de la distribución aprendida por G) o proviene de los datos originales.

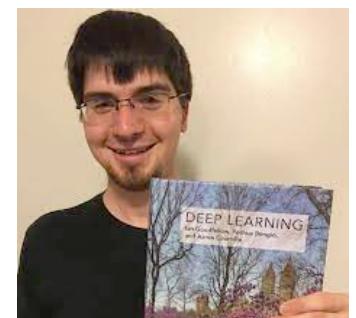
Generative Adversarial Nets

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

Ian Goodfellow

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.



Contexto y motivación

Si tenemos buenos modelos discriminativos, por qué no usar uno para que le enseñe de forma implícita a otro a "engaño"?

En palabras de los mismos autores:

"This framework corresponds to a minimax two-player game"

Vamos a entrenar iterativamente al discriminador para aprender a discernir entre data real y la del generador y cuando este haya aprendido relativamente bien la tarea, vamos a dejar que el generador ahora aprenda a generar mejores objetos (más parecidos a la data real)



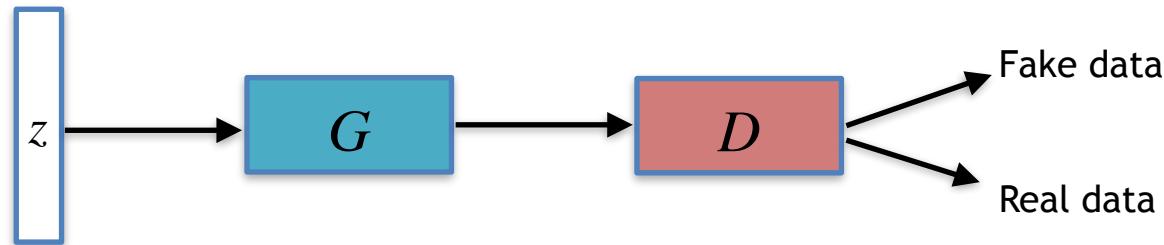
Redes Adversariales - Formulación

Sea un modelo Generador $G : Z \rightarrow X \in \mathbb{R}^m$ que genera elementos en el mismo espacio que el de la data X a partir de un vector $z \in Z$ mediante una distribución p_g . Notar que para poder aprender la distribución p_g se requiere definir un prior sobre las variables de input: $p_z(z)$ (En la práctica esto podría ser $N(0,1)$ por ejemplo).

$$z \sim p_z(z) \rightarrow G(z; \theta_g) \rightarrow x \sim p_g(x)$$

↓
En el paper lo implementan con un MLP

Sea también un modelo discriminador $D : X \rightarrow \{0,1\}$ (implementado también como un MLP en el paper original) cuya tarea es determinar si un elemento x pertenece a la distribución de los datos p_{data} o a p_g (En el modelo original se considera de manera que $D(x; \theta_d)$ representa la probabilidad de que x pertenezca a p_{data}).



Redes Adversariales - Formulación

El entrenamiento de este modelo implica alternar entre entrenar el generador y entrenar el discriminador mediante el siguiente objetivo $V(G, D)$:

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

Real data Fake data

Notar que el término de la derecha $D(G(z))$ es la probabilidad de que el discriminador prediga como "falso" el elemento generado por G .

El discriminador es entrenado para aprender a maximizar los dos términos presentes, por un lado (izquierda) trata de predecir los elementos de los datos como "reales" (pertenecen a p_{data}) con alta probabilidad (confianza) y por otro lado trata de predecir los elementos generados por G (los vectores z) como pertenecientes a P_g con alta confianza.

Redes Adversariales - Formulación

El entrenamiento de este modelo implica alternar entre entrenar el generador y entrenar el discriminador mediante el siguiente objetivo $V(G, D)$:

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

↓ ↓
Real data Fake data

El generador es entrenado para minimizar la expresión de la única forma en la que puede interferir en ella:
Haciendo que D tenga menos certeza de si $G(z)$ pertenece a p_g o a p_{data} .



Redes Adversariales - Loss

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

Notar también que el orden de la formulación de la optimización es **muy** relevante: El hecho de formular el problema como un minimax implica que cuando se debe entrenar el generador G esto se hace contra un discriminador que ya tuvo la oportunidad de aprender a diferenciar entre lo que generó G y la data real.

El punto anterior es importante porque si el discriminador es malo para discriminar entonces los gradientes que se van a propagar hacia G van a ser malos también, dicho con un ejemplo, alguien que va a corregir la evaluación de un/a alumno/a debe saber lo que está haciendo para evitar confundir con la revisión al/la alumno/a. Por esta razón veremos luego que el algoritmo en realidad mantiene G fijo mientras entrena por algunas iteraciones solo a D , de manera que este aprenda a distinguir (“alcance”) lo que se le muestra, solo entonces, cuando se ha obtenido un discriminador relativamente bueno es que se entrena G ahora para volver a aprender una nueva forma debe engañar a D



Redes Adversariales - Loss

Como mencionan los autores, en la práctica al inicio del entrenamiento es una mala idea usar esta función de pérdida:

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

El problema está específicamente en los momentos en los que se debe entrenar el generador G . Al inicio del entrenamiento es casi seguro que lo que genere G sea fácilmente distinguible de la data real producto que G aún no ha sido entrenado, en estos casos el valor que D entrega el término $D(G(z))$ es cercano a 1 (es fácilmente distinguible que es falso, que pertenece a p_g).

Cuando ocurre lo anterior tenemos:

$$\lim_{D(G(z)) \rightarrow 1} \log(1 - D(G(z))) = +\infty$$

Para evitar esto los autores entrena a G durante las primeras iteraciones para minimizar $\log[D(G(z))]$ en lugar de $\log[1 - D(G(z))]$.



Redes Adversariales - Entrenamiento

El entrenamiento de G y D se hace alternadamente, primero se entrena el discriminador durante k pasos maximizando la función de valor $V(D, G)$, luego, se actualiza el generador una sola vez, la idea es no cambiar demasiado el generador (y por lo tanto p_g) o el discriminador podría "perderle el rastro".

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:

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

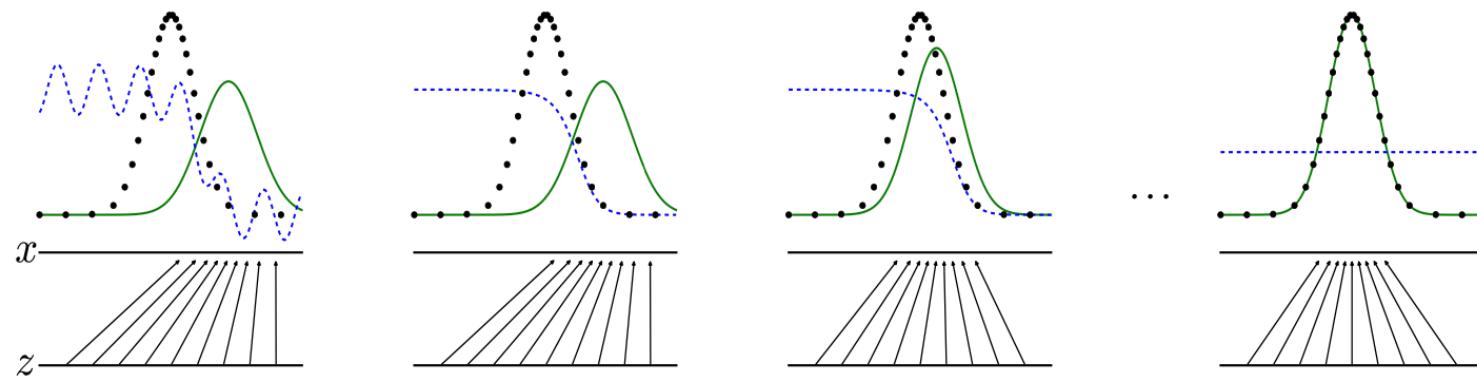
end for

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



Redes Adversariales - Entrenamiento

El entrenamiento de G y D se hace alternadamente, primero se entrena el discriminador durante k pasos maximizando la función de valor $V(D, G)$, luego, se actualiza el generador una sola vez, la idea es no cambiar demasiado el generador (y por lo tanto p_g) o el discriminador podría "perderle el rastro".



Redes Adversariales - Entrenamiento

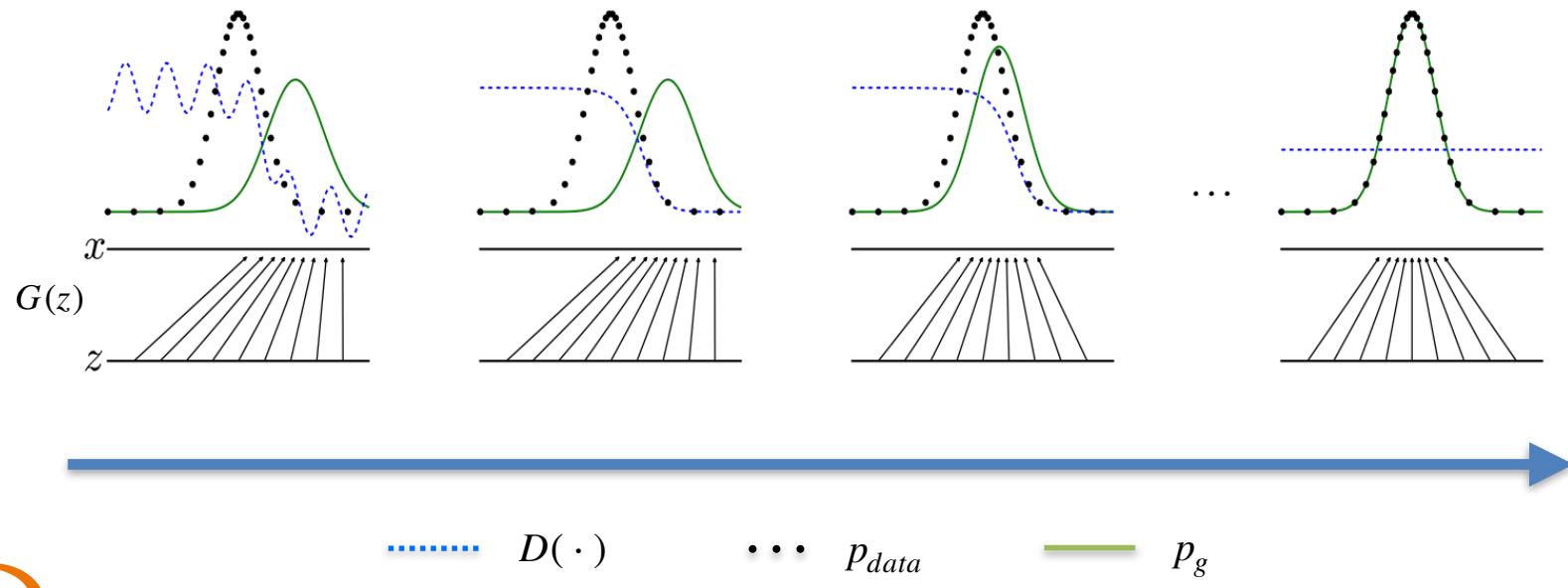
El entrenamiento de G y D se hace alternadamente, primero se entrena el discriminador durante k pasos maximizando la función de valor $V(D, G)$, luego, se actualiza el generador una sola vez, la idea es no cambiar demasiado el generador (y por lo tanto p_g) o el discriminador podría "perderle el rastro".

Discriminador

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

Generador

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



Redes Adversariales - Aspectos Matemáticos

La propuesta de los autores es muy interesante, sin embargo, considerando el fuerte background matemático de los métodos generativos usados en esa época, Goodfellow et al. Se enfrentaban al (para nada simple) problema de mostrar que el usar un modelo no entrenado para entrenar a otro era algo que tenía sentido. Veamos algunas de las justificaciones que dan en el paper a la respuesta de:

¿Por qué esto debería funcionar?



Discriminador Óptimo

El criterio de entrenamiento que definimos para el discriminador, dado un G fijo, estaba dado por el valor de $V(G, D)$:

$$\begin{aligned} V(D, G) &= \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \int_x p_{data}(x) \log(D(x)) dx + \int_x p_z(z) \log(1 - D(G(z))) dz \\ &= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned}$$

Esta función tiene la forma $a \log(y) + b \log(1 - y)$ si tratamos de maximizarla para y (que en las ecuaciones anteriores sería D):

$$f = a \log(y) + b \log(1 - y)$$



Discriminador Óptimo

Esta función tiene la forma $a \log(y) + b \log(1 - y)$ si tratamos de maximizarla para y (que en las ecuaciones anteriores sería D):

$$f = a \log(y) + b \log(1 - y)$$

Primera derivada

$$\frac{\partial f}{\partial y} = \frac{a}{y} - \frac{b}{1 - y}$$



$$\frac{a}{y} - \frac{b}{1 - y} = 0$$

$$a(1 - y) = by$$

$$\boxed{\frac{a}{a + b} = y}$$

Segunda derivada

$$\frac{\partial^2 f}{\partial y^2} = -\frac{a}{y^2} - \frac{b}{(1 - y)^2}$$



$$\boxed{-\frac{a}{y^2} - \frac{b}{(1 - y)^2} < 0 \quad \forall y \text{ Tq. } a, b > 0}$$

La segunda derivada siempre es negativa si $a, b > 0$, lo cual siempre ocurre porque en la ecuación original $a = p_{data}(x)$ y $b = p_g(x)$.

Discriminador Óptimo

Lo anterior quiere decir que el discriminador óptimo en este juego, es aquél que toma valor según la función de probabilidad $\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$

Proposición 1: Sea un G fijo, el discriminador óptimo es entonces:

$$D_G^* = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$



Generador Óptimo - Optimalidad para p_g

Preguntemos ahora que ocurre con el generador y la distribución que este debe aprender: ¿Qué valor debe tomar p_g para poder minimizar globalmente la función de valor $V(D, G)$?

Consideremos el objetivo de entrenamiento del discriminador D :

$$V(G, D) = \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z))) dz$$

Esta expresión puede ser interpretada como la log-verosimilitud para estimar la probabilidad condicional $P(Y = y|x)$ donde Y indica si x viene de p_{data} o p_g . Luego, el criterio de entrenamiento anterior para el discriminador con respecto al generador, considerando un discriminador óptimo, puede ser re-escrito como:

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



Generador Óptimo - Optimalidad para p_g

Es posible demostrar que el valor mínimo para $C(G)$ ocurre solo cuando $p_g = p_{data}$. En efecto, consideremos el discriminador óptimo bajo $p_{data} = p_g$:

$$\begin{aligned} D_G^*(x) \Big|_{p_g=p_{data}} &= \frac{p_{data}(x)}{2 \cdot p_{data}(x)} \\ &= \frac{1}{2} \end{aligned}$$

Luego, analizando la función de valor:

$$\begin{aligned} V(G, D_G^*) &= -\log 2 \int_x p_G(x) - \log 2 \int p_{data}(x) \, dx \\ &= -2\log 2 = -\log 4 \end{aligned}$$

Este es un candidato a mínimo global, paralelamente sigamos con $C(G)$, lo que queremos es mostrar que el valor $-\log 4$ en efecto es el mínimo global que puede alcanzar esta función.



Generador Óptimo - Optimalidad para p_g

Desarrollando $C(G)$:

$$\begin{aligned} C(G) &= \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) + p_G \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) dx \\ &= \int_x (\log 2 - \log 2) p_{data}(x) + p_{data}(x) \log \left(\frac{p_{data}(x)}{p_G + p_{data}(x)} \right) \\ &\quad + (\log 2 - \log 2) p_G(x) + p_G(x) \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) dx \\ &= -\log 2 \int_x p_G(x) + p_{data}(x) dx + \int_x p_{data}(x) \left(\log 2 + \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right) \\ &\quad + p_G \left(\log 2 + \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) \right) dx \end{aligned}$$



Generador Óptimo - Optimalidad para p_g

Notar ahora que $\int_x p_G = \int_x p_{data} = 1$ por lo que el primer término de la expresión anterior queda igual a $-\log 4$, además, podemos reducir un poco los términos de la segunda integral:

$$\begin{aligned} C(G) &= -\log 2 \int_x p_G(x) + p_{data}(x) \, dx + \int_x p_{data}(x) \left(\log 2 + \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right) + p_G(x) \left(\log 2 + \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) \right) \, dx \\ &= -\log 4 + \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) + p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx \\ &= -\log 4 + \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx + \int_x p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx \end{aligned}$$



Generador Óptimo - Optimalidad para p_g

Notar ahora que $\int_x p_G = \int_x p_{data} = 1$ por lo que el primer término de la expresión anterior queda igual a $-\log 4$, además, podemos reducir un poco los términos de la segunda integral:

$$\begin{aligned} C(G) &= -\log 2 \int_x p_G(x) + p_{data}(x) \, dx + \int_x p_{data}(x) \left(\log 2 + \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right) + p_G(x) \left(\log 2 + \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) \right) \, dx \\ &= -\log 4 + \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) + p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx \\ &= -\log 4 + \boxed{\int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx} + \boxed{\int_x p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx} \end{aligned}$$

Estos términos son la divergencia KL entre las distribuciones p_{data}/p_G y $\frac{p_{data} + p_G}{2}$!



Generador Óptimo - Optimalidad para p_g

Notar ahora que $\int_x p_G = \int_x p_{data} = 1$ por lo que el primer término de la expresión anterior queda igual a $-\log 4$, además, podemos reducir un poco los términos de la segunda integral:

$$\begin{aligned} C(G) &= -\log 2 \int_x p_G(x) + p_{data}(x) \, dx + \int_x p_{data}(x) \left(\log 2 + \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right) + p_G(x) \left(\log 2 + \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) \right) \, dx \\ &= -\log 4 + \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) + p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx \\ &= -\log 4 + \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx + \int_x p_G(x) \log \left(\frac{p_G(x)}{(p_G(x) + p_{data}(x))/2} \right) \, dx \\ \Rightarrow C(G) &= -\log 4 + KL\left(p_{data} \middle| \frac{p_{data} + p_G}{2}\right) + KL\left(p_G \middle| \frac{p_{data} + p_G}{2}\right) \end{aligned}$$



Generador Óptimo - Optimalidad para p_g

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

En la expresión anterior, notar que la divergencia KL es siempre no-negativa, por lo que el mínimo de este criterio de entrenamiento para el generador ocurre cuando $KL\left(p_{data} \middle| \frac{p_{data} + p_G}{2}\right) = 0$ (análogamente para p_G) y dicho mínimo es en efecto $-\log 4$, por lo que este mínimo es global. Más aún, considerar la siguiente divergencia:

Divergencia de Jensen-Shannon

La divergencia de Jensen-Shannon se dice que es una versión simétrica y suavizada de la divergencia de Kullback-Leibler. La divergencia de Jensen-Shannon (JSD) se define como:

$$JSD(P\|Q) = \frac{KL(P\|M)}{2} + \frac{KL(Q\|M)}{2}$$

Donde $M = \frac{1}{2}(P + Q)$



Generador Óptimo - Optimalidad para p_g

Esto quiere decir que podemos re-expresar la forma encontrada para el criterio de entrenamiento del generador con respecto a esta divergencia:

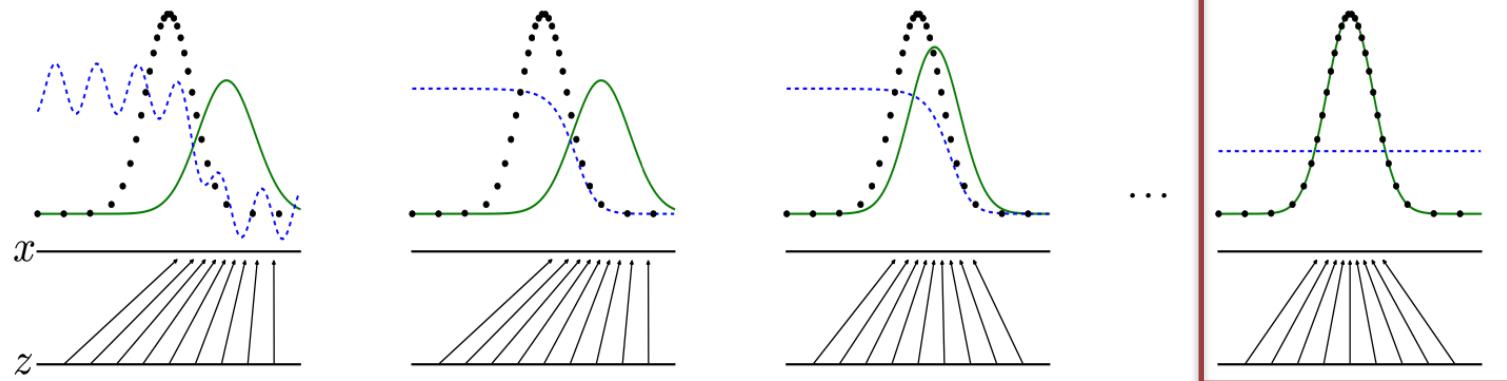
$$C(G) = -\log 4 + 2 \cdot JSD(p_{data} \| p_G)$$

Lo interesante de esta expresión es que JSD solo es igual a 0 cuando $p_{data} = p_G$, por lo que no solo la distribución p_G minimiza el criterio de entrenamiento cuando es igual a p_{data} sino que también (p_{data}) es **el** mínimo global y único del criterio de entrenamiento.



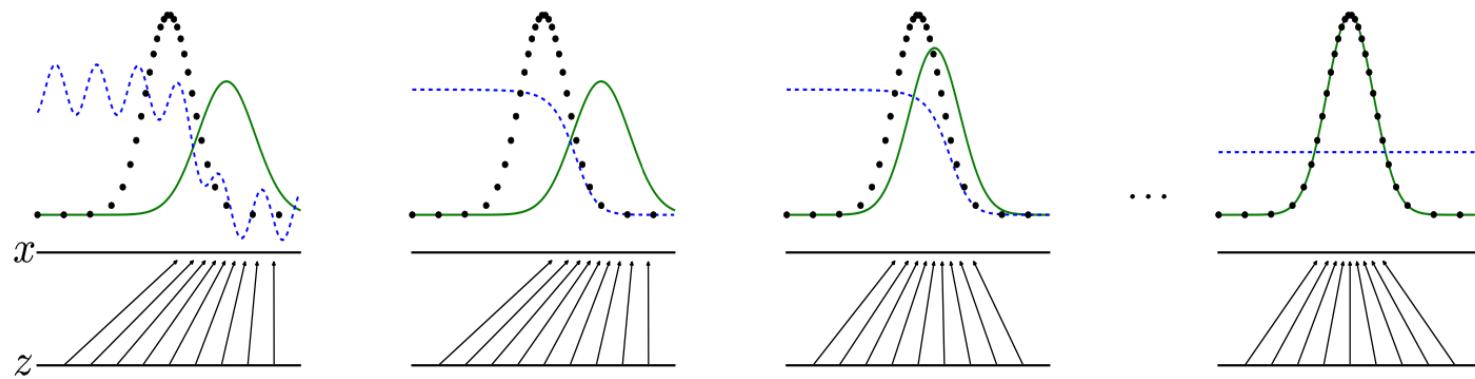
Generador Óptimo - Optimalidad para p_g

Los dos resultados anteriores son críticos para entender el por qué funcionan estos modelos, si el criterio de entrenamiento del generador no apuntase a que su distribución p_g replique la distribución de los datos entonces no habría ninguna garantía de que el modelo va a aprender algo relativamente coherente. Esto también tiene sentido cuando lo vemos desde el punto de vista del discriminador:



Convergencia?

La demostración de convergencia queda de tarea para ustedes
(Está en el paper)



Resultados experimentales

Terminemos discutiendo los resultados que obtuvieron los autores cuando crearon este modelo.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	2110 ± 50
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	225 ± 2	2057 ± 26



Resultados experimentales

Terminemos discutiendo los resultados que obtuvieron los autores cuando crearon este modelo.



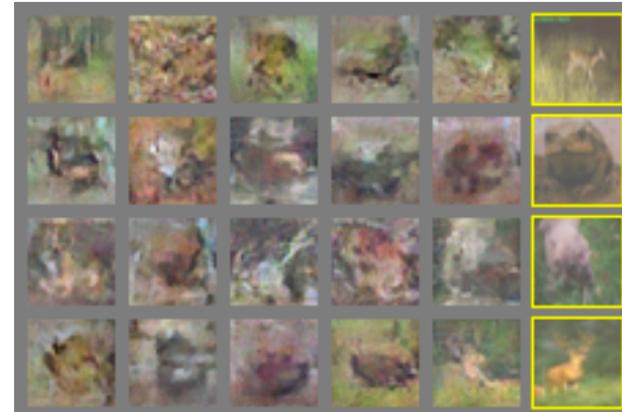
a)



b)



c)



d)

Usos contemporáneos

Lo que vimos acá es el modelo básico, a lo largo de los años numerosas mejoras y modificaciones se han hecho sobre este modelo, acá hay un ejemplo dentro del área de la visión computacional:



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham,
Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi
Twitter

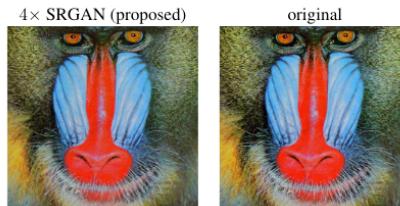
{cledig, ltheis, fhuszar, jcaballero, aacostadiaz, aaitken, atejani, jtrotz, zehanw, wshi}@twitter.com

Abstract

Despite the breakthroughs in accuracy and speed of single image super-resolution using faster and deeper convolutional neural networks, one central problem remains largely unsolved: how do we recover the finer texture details when we super-resolve at large upscaling factors? The behavior of optimization-based super-resolution methods is principally driven by the choice of the objective function. Recent work has largely focused on minimizing the mean squared reconstruction error. The resulting estimates have high peak signal-to-noise ratios, but they are often lacking high-frequency details and are perceptually unsatisfying in the sense that they fail to match the fidelity expected at the higher resolution. In this paper, we present SRGAN, a generative adversarial network (GAN) for image super-

1. Introduction

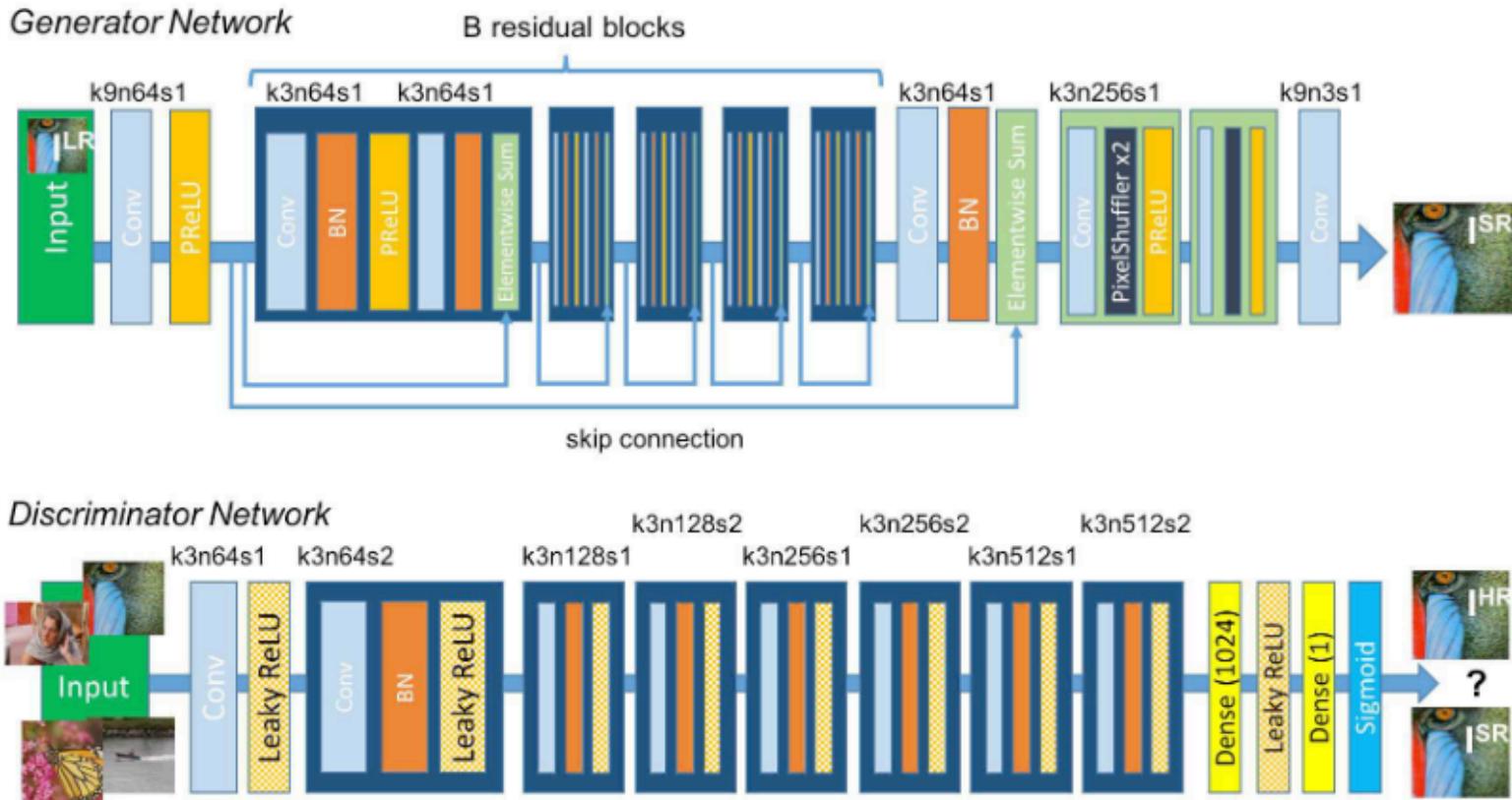
The highly challenging task of estimating a high-resolution (HR) image from its low-resolution (LR) counterpart is referred to as super-resolution (SR). SR received substantial attention from within the computer vision research community and has a wide range of applications [62, 70, 42].



Christian Ledig



Usos contemporáneos



Usos contemporáneos



Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Fin

