

Redes Generativas Adversarias (GANs)

por Adrià Cabeza Sant'Anna

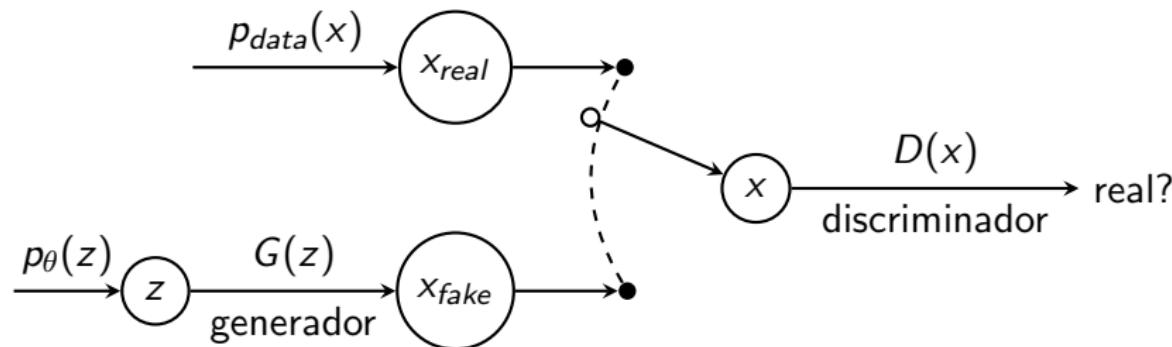
2020

Contenido

- Teoría
 - ¿Qué es una GAN?
 - Loss Function
 - Entrenamiento
- Autocolorizador
 - Generador
 - Discriminador
- Implementación en TF 2.0 [1]

Teoría

¿Qué es una GAN?



Adversarial Loss [2]

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

Loss Function

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

- D quiere maximizar la probabilidad de clasificar la data correctamente:
 - $\log D(\mathbf{x})$ es decir $D(\mathbf{x}) \rightarrow 1$
 - $\log(1 - D(G(\mathbf{z})))$ es decir $D(G(\mathbf{z})) \rightarrow 0$
- G quiere minimizar esa probabilidad a D:
 - $\log(1 - D(G(\mathbf{z})))$ es decir $D(G(\mathbf{z})) \rightarrow 1$

Optimality

Si quereis saber más sobre porqué funciona la Loss function o sobre su *optimality* → Id a la diapositiva 15.

1: **for** 1 to N **do**
 2: **for** 1 to k **do**
 3: Cogemos $z^{(1)}, \dots, z^{(m)} \in p_g(z)$
 4: Cogemos $x^{(1)}, \dots, x^{(m)} \in p_{data}(x)$
 5: Actualizamos el discriminador usando stochastic gradient ascent/descent:

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

6: **end for**
 7: Cogemos $z^{(1)}, \dots, z^{(m)} \in p_g(z)$
 8: Actualizamos el generador usando stochastic gradient ascent/descent:

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

9: **end for**

Memes

Ahora ya podéis entender los memes:

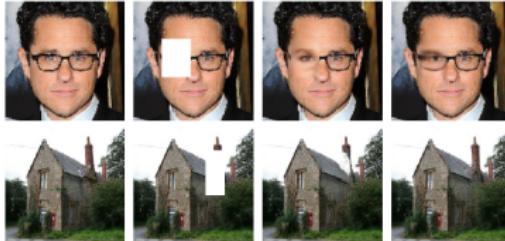


Usos

Ejemplos de aplicaciones

- Superresolución e.g.
<https://github.com/tensorlayer/SRGAN>
- Colorización e.g. <https://github.com/jantic/DeOldify>
- Síntesis de imágenes e.g.
<https://www.thispersondoesnotexist.com/>
- Traducción de imágenes
- Aumentar el tamaño de datasets
- Más de 100 arquitecturas diferentes!!
<https://github.com/hindupuravinash/the-gan-zoo>

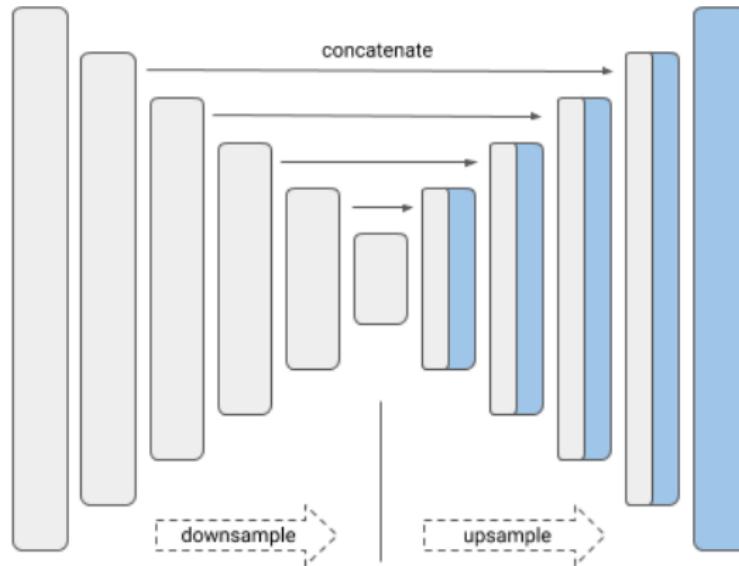
Usos



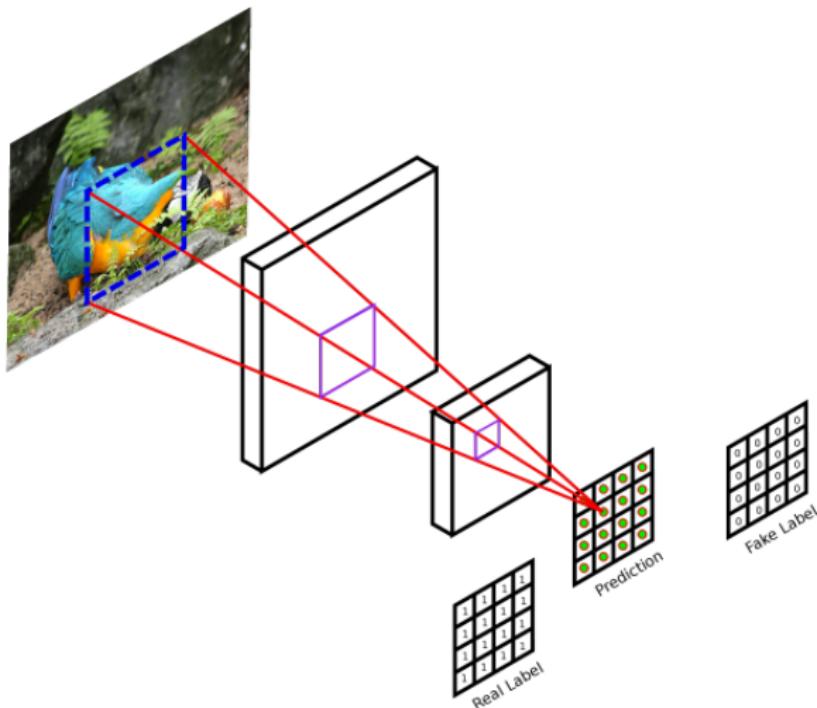
Autocolorizador: Pix2Pix [3]



Auticolorizador: Generador UNET



Auticolorizado: Discriminador PatchGAN



Implementación en TF 2.0

Let's start coding!

<https://colab.research.google.com>



Bibliography

-  "Tensorflow: A system for large-scale machine learning."
<https://www.tensorflow.com/>.
-  I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.
-  P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016.

Optimality de una GAN

Como hemos mencionado anteriormente, la loss function de una GAN consiste en un juego minimax:

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

Interpretación del criterio de training para D

Nuestro objetivo es, dado cualquier generador G , maximizar la cantidad $V(D, G)$ de manera que puede ser interpretado como maximizar el log-likelihood de $P(Y = y|x)$, donde Y indica si x viene de p_{data} ($y = 1$) o de p_g ($y = 0$).

Para demostrar su optimality, primero analizaremos el discriminador óptimo D^* . Si partimos de la loss function teniendo en cuenta solo D , o sea fijando G :

$$p_{data}(x) \cdot \log D(x) + p_g(x) \cdot \log(1 - D(x))$$

podemos buscar su valor máximo (demostración en la próxima diapositiva) para encontrar a D^*

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

Para cualquier $(a, b) \in R^2 \setminus (0, 0)$ la función
 $f(x) = a \cdot \log(x) + b \cdot \log(1 - x)$ llega a su máximo en $[0, 1]$ en $\frac{a}{a+b}$.

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

$$f'(x) = \frac{a}{x} - \frac{b}{1-x} = 0$$

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

$$a \cdot (1-x) - b \cdot x = 0$$

$$x = \frac{a}{a+b}$$

sabiendo que $a + b \neq 0$, comprobamos que sea un máximo:

$$f''(x) = (a \cdot \log(x) + b \cdot \log(1 - x))'' = \frac{-a}{x^2} - \frac{b}{(1-x)^2} < 0$$

Nótese que valor óptimo de D^* se consigue cuando $p_{data} = p_g$:

$$D^*(x) = \frac{p_{data}}{p_{data} + p_g} = \frac{1}{2}$$

Eso quiere decir que el discriminador no tiene ni idea de si la data es de $p_g(0)$ o de $p_{data}(1)$. Lo cual tiene sentido, es al fin y al cabo lo que queremos. Ahora pues ya podemos reemplazar ese valor en la loss function:

$$\begin{aligned} \min_G \max_D V\left(\frac{1}{2}, G\right) &= \mathbb{E}_{x \sim p_{data}(x)} \left[\log \frac{1}{2} \right] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - \frac{1}{2}\right) \right] \\ &= -2 \log 2 = -\log 4 \end{aligned}$$

Ahora solo nos queda demostrar que el mínimo global de la loss function se consigue si y solo si $p_{data} = p_g$ y tiene de valor $-\log 4$.
Y eso lo dejo como deberes...

Si lo hacéis, tenéis mucha curiosidad o no os sale, no dudéis en escribirme a adria@hackupc.com.