

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

MÁSTER EN TRATAMIENTO ESTADÍSTICO
COMPUTACIONAL DE LA INFORMACIÓN



TRABAJO DE FIN DE MÁSTER

Redes Generativas Antagónicas

Antón Makarov Samusev

Director

Francisco Javier Yáñez Gestoso

Madrid, 2019

Resumen

Resumen

Abstract

Abstract

Índice general

1. Introducción	7
2. Preliminares	8
2.1. Conceptos básicos	8
2.2. Redes neuronales	8
2.2.1. Aprendizaje profundo	8
2.2.2. Redes neuronales convolucionales	8
3. Redes generativas antagónicas	9
3.1. Idea general	9
3.2. Aspectos teóricos	9
3.3. Otros modelos generativos	10
4. Generación de arte	12
4.1. DCGAN	12
4.2. Preprocesado	13
4.3. Arquitectura	13
4.4. Resultados	13
4.4.1. Recursos	13
5. Conclusión	15
Bibliografía	16

Capítulo 1

Introducción

Introducción¹ [3], [1], [5], [2]

¹Todo el código de este trabajo se puede encontrar en el repositorio de GitHub <https://github.com/ant-mak/tfm>, donde se incluyen las instrucciones para la reproducción de los resultados del proyecto.

Capítulo 2

Preliminares

2.1. Conceptos básicos

Definición 1 (Divergencia de Kullback-Leibler). *KL div*

Definición 2 (Divergencia de Jensen-Shannon). *JS div*

Definición 3 (Equilibrio de Nash). *Eq. Nash*

2.2. Redes neuronales

2.2.1. Aprendizaje profundo

2.2.2. Redes neuronales convolucionales

Capítulo 3

Redes generativas antagónicas

En este capítulo describiremos las ideas principales, tanto teóricas como prácticas de las redes generativas antagónicas, generative adversarial networks o GANs en inglés. Fueron propuestas por primera vez por Ian Goodfellow en 2014 [3], mezclando conceptos de aprendizaje automático no supervisado, supervisado y teoría de juegos. Desde entonces han suscitado una actividad investigadora muy importante.

3.1. Idea general

El objetivo de las GANs es generar muestras de una distribución, cosa que a priori no parece complicado, existen métodos para generar muestras de variables aleatorias conocidas como por ejemplo la transformada inversa. Sin embargo, supongamos que queremos generar muestras de fotos de perros. Tenemos la seguridad de que la distribución es extremadamente compleja de describir, dado que hay perros de distintos colores, tamaños, razas, etc.

La idea fundamental de las GANs es poner dos redes neuronales a competir entre sí. Una red, llamada generadora (G), está dedicada a obtener imágenes a partir de ruido, mientras que otra, llamada discriminadora (D), trata de averiguar si la imagen es real o ficticia. Es frecuente ilustrar esta idea mediante la analogía de falsificadores de billetes que tratan de engañar a la policía. Los falsificadores empiezan dibujando billetes que no tienen nada que ver con los reales, intentando utilizarlos para pagar, momento en el que son atrapados por la policía. Los falsificadores por tanto se dan cuenta de que están dibujando mal y modifican su técnica, de manera que a lo largo del tiempo aprenden a burlar a la policía, que no es capaz de distinguir billetes reales de falsos. Una representación esquemática se puede ver en la Figura (3.1)

El algoritmo genérico para el entrenamiento de este tipo de redes se describe en el Algoritmo 1.

3.2. Aspectos teóricos

Veamos algunos de los resultados teóricos más relevantes:

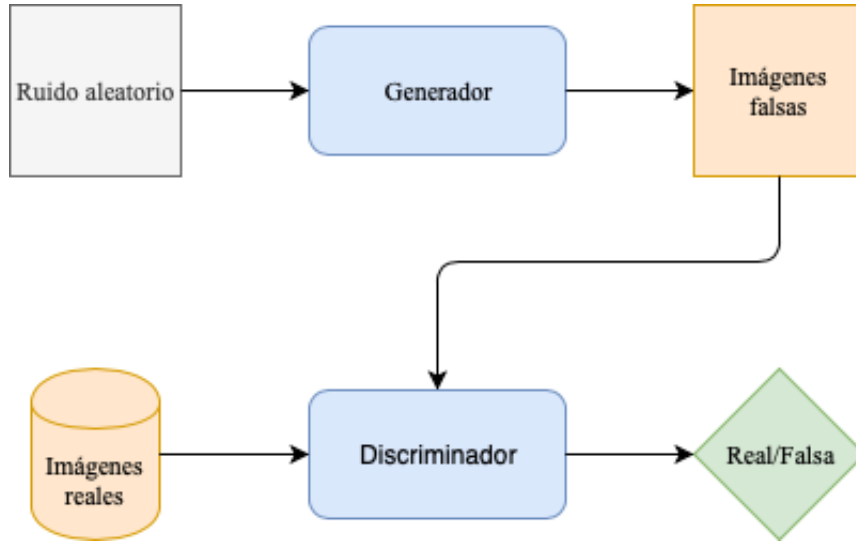


Figura 3.1: Diagrama conceptual de las redes generativas antagónicas.

Teorema 1. *Para un generador G fijo, el discriminador D óptimo es:*

$$D_G^*(x) = \frac{p_{\text{datos}}(x)}{p_{\text{datos}}(x) + p_g(x)} \quad (3.1)$$

Demostración. El discriminador busca maximizar su función de utilidad, dada por $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_x p_{\text{datos}}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(g(z))) dz \\ &= \int_x p_{\text{datos}}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned} \quad (3.2)$$

□

3.3. Otros modelos generativos

Variational autoencoders, noise contrastive estimation etc.

Algoritmo 1: Entrenamiento de una red generativa antagónica genérica.

1 for *Iteraciones de entrenamiento* **do**

2 Tomar muestra $(z^{(1)}, z^{(2)}, \dots, z^{(m)})$ de tamaño m de $p_g(z)$;

3 Tomar muestra $(x^{(1)}, x^{(2)}, \dots, x^{(m)})$ de tamaño m de $p_{data}(x)$;

4 Actualizar el discriminador ascendiendo su gradiente:

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

5 Actualizar el generador ascendiendo su gradiente:

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

6 end

Capítulo 4

Generación de arte

En esta sección nos vamos a apoyar sobre el artículo [5] para implementar en `PyTorch` una red convolucional profunda generativa antagónica, con la cual vamos a tratar de generar cuadros realistas.

4.1. DCGAN

Las redes convolucionales profundas generativas antagónicas (deep convolutional generative networks, DCGAN) son una extensión de las GANs tradicionales, que dan una serie de recomendaciones a la hora de implementarlas para incrementar la estabilidad y obtener imágenes de mejor calidad. Los puntos más relevantes mencionados en el artículo son:

- En el discriminador, utilizar convoluciones con paso en lugar de capas de pooling.
- En el generador, utilizar convoluciones fraccionales con paso en vez de capas de pooling.
- Utilizar BatchNorm tanto para generador como discriminador.
- No utilizar capas fully connected
- Utilizar funciones de activación ReLU en el generador, salvo para la última capa, en la que se propone usar tanh
- Utilizar funciones de activación LeakyReLU en el discriminador.

RECORDAR QUE STRIDE Y POOL SON PARECIDOS Y DESCRIBIR LEAKYRELU.

Como hemos dicho, nuestro objetivo es generar arte. Para ello vamos a necesitar un conjunto de datos con imágenes de una gran cantidad de cuadros. Dichas imágenes las hemos obtenido de una competición de Kaggle. Este conjunto de datos consta de aproximadamente 100000 imágenes, ocupando un total de 49 GB en el disco duro.

4.2. Preprocesado

El conjunto de datos tenía algunos defectos, como por ejemplo que había imágenes que no se descargaron correctamente o que el propio conjunto de datos tenía algunas muestras corruptas. Dichas imágenes las eliminamos.

Una vez hemos limpiado el conjunto de datos, es necesario realizar algunas transformaciones para que todo funcione como esperamos. En primer lugar hemos homogeneizado los tamaños y las proporciones. Pasamos así de cuadros de todos los tamaños y de distintas formas a cuadrados de 64×64 píxeles. Posteriormente, les hacemos un recorte centrado y finalmente los cargamos como tensores.

4.3. Arquitectura

Siguiendo las recomendaciones del artículo [5], hemos decidido utilizar la arquitectura que se muestra en la Figura 4.3. Se ha entrenado durante 30 épocas, utilizando el algoritmo de optimización Adam [4] con una tasa de aprendizaje de 0,0002.

4.4. Resultados

4.4.1. Recursos

Cabe destacar que ha sido imprescindible el uso de un ordenador con GPU compatible con CUDA, en concreto se ha utilizado una Nvidia Quadro P5000. El entrenamiento de principio a fin ha tardado aproximadamente 24 horas. También se han realizado pruebas con CPU en un portátil de prestaciones modestas y se ha observado que el rendimiento era unas 20 veces menor.

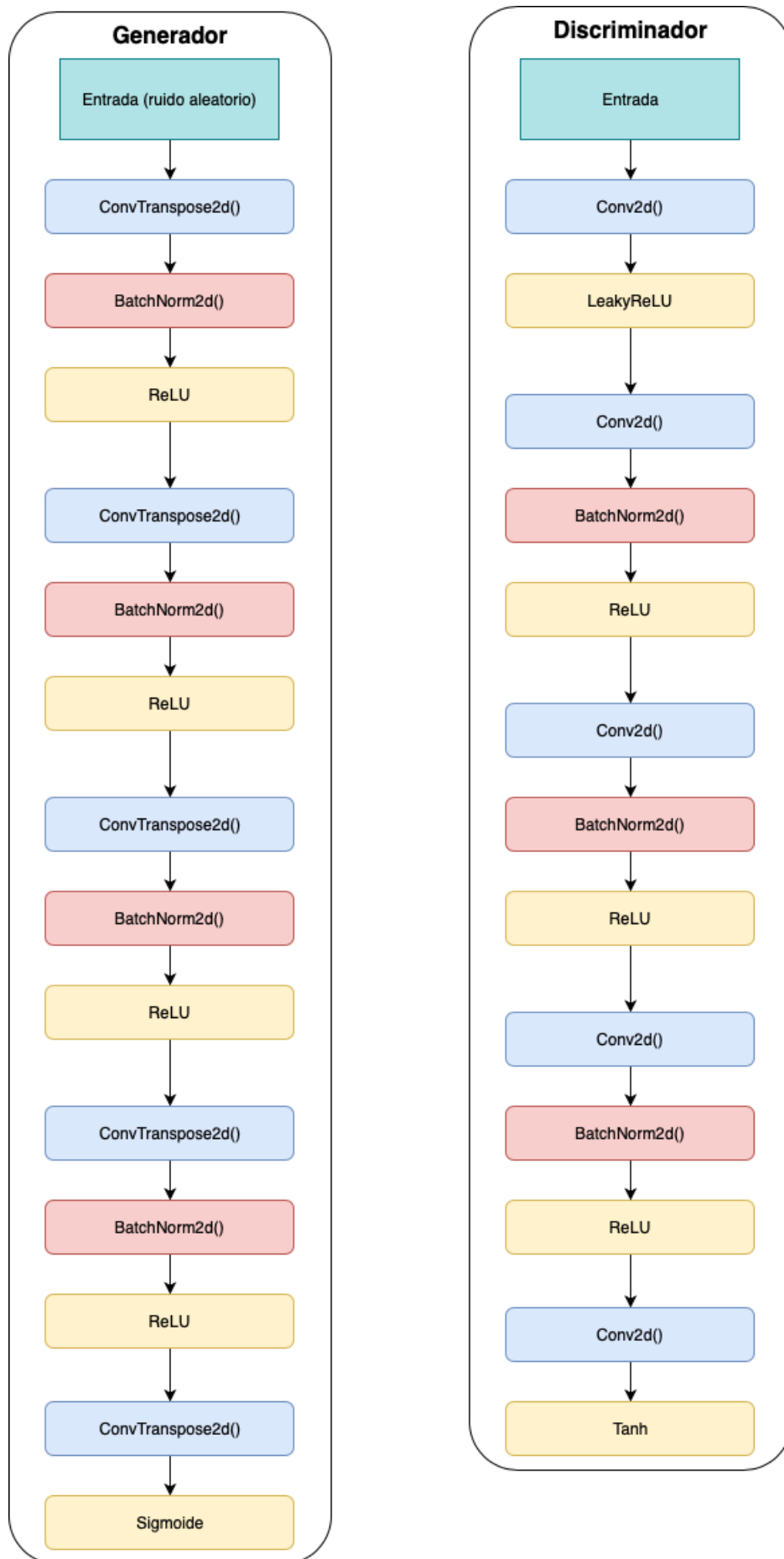


Figura 4.1: Arquitectura de las redes (Cambiar, poner tamaños de entrada y salida).

Capítulo 5

Conclusión

Bibliografía

- [1] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.
- [2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [5] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.