

Generative Adversarial Networks

Métodos Generativos, curso 2025-2026

Guillermo Iglesias, guillermo.iglesias@upm.es

Jorge Dueñas Lerín, jorge.duenas.lerin@upm.es

Edgar Talavera Muñoz, e.talavera@upm.es

5 de noviembre de 2025

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos | UPM



Modelos generativos

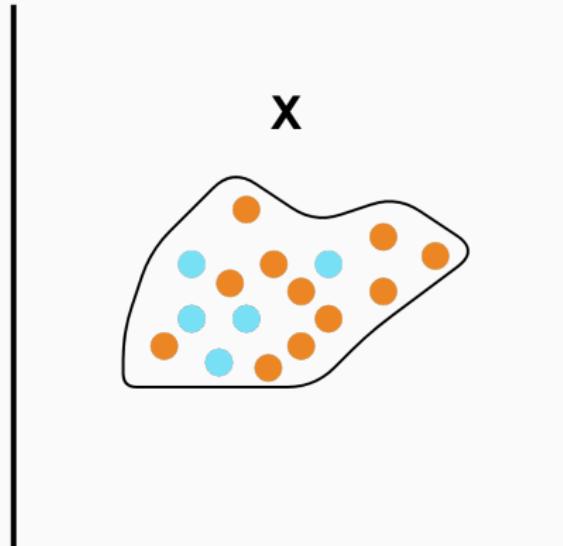
Modelos generativos

- Distribución de datos X de la cual se quieren generar nuevas instancias.



Modelos generativos

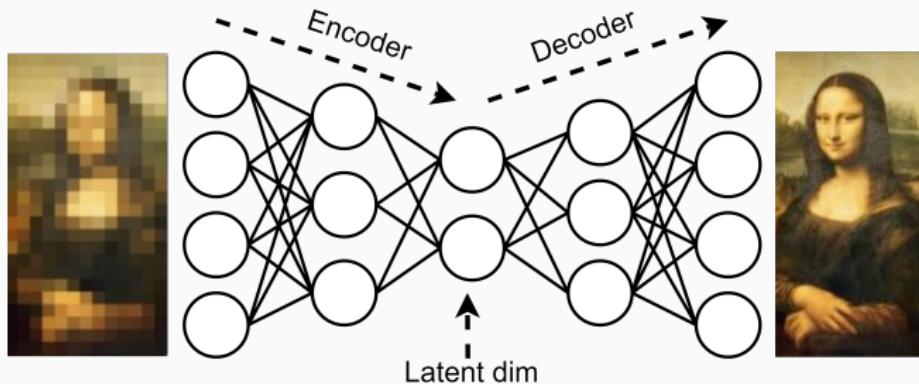
- Se quieren generar nuevos puntos **distintos**.
- **Aprender la distribución** de origen.
- Generar **valores indistinguibles** de los de entrada.



Nuevas instancias (en azul) de la distribución de datos X (en naranja).

Autoencoder

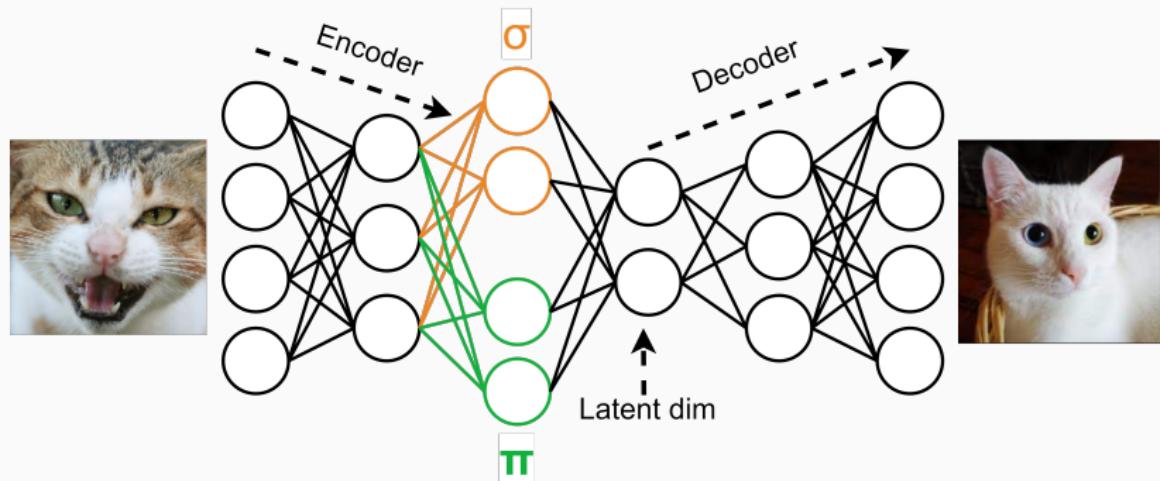
- Capaz de aprender la distribución de entrada.
- Sólo replica los datos de entrada.
- No consigue generalizar y generar nuevos datos.



Modelo de Autoencoder capaz de reconstruir una imagen de baja resolución.

Variational Autoencoder

- Aprende a generalizar gracias al uso de una **función de densidad**.
- Genera datos **indistinguibles** de los de entrada.
- Genera nuevos datos **sin copiar** directamente los ejemplos.



Modelo de Variational Autoencoder capaz de generar imágenes de gatos.

Modelos generativos: principales problemas

- Estos modelos no tienen la **calidad suficiente**.
- Se buscan alternativas capaces de generar resultados **más nítidos**.
- La aproximación de **aprender** directamente la distribución genera resultados **poco diversos**.

Generative Adversarial Network (GAN): Fundamentos

Definición

- Las redes GAN fueron propuestas en el año 2014 por Ian Goodfellow[1].
- Se definen como un modelo **neuronal generativo** basado en la **teoría de juegos** que tiene como objetivo **replicar** una distribución de datos.
- Se basa en un juego de **suma cero** que involucra **2 redes neuronales distintas**.

Ejemplo: Falsificador vs. policía

- Se quieren hacer **falsificaciones** de billetes.
- Se busca que la policía sea **incapaz de distinguir** entre los billetes falsos y los reales.



Falsificador



- Excelente artista.
- Es capaz de **generar nuevos** billetes.
- Inicialmente **no sabe** lo que es un billete.

Quiere **engaÑar** al policía con billetes **falsos**.

Policía



- **Discrimina** la veracidad de cada billete.
- Inicialmente **no sabe** lo que es un billete.

Quiere **separar los billetes falsos de los verdaderos.**

Redes GAN: Intuición previa

- El ladrón intenta **engañoso** al policía generando billetes que se **parezcan** a los reales.
- Si engaña al policía sabe que ese billete es de **buenas calidades**.



Redes GAN: Intuición previa

- El policía intenta no ser engañado **discriminando** las diferencias **más sutiles** entre los billetes.
- Si es **engaño** aprende cuáles son esos **detalles** que hacen que el billete sea falso.

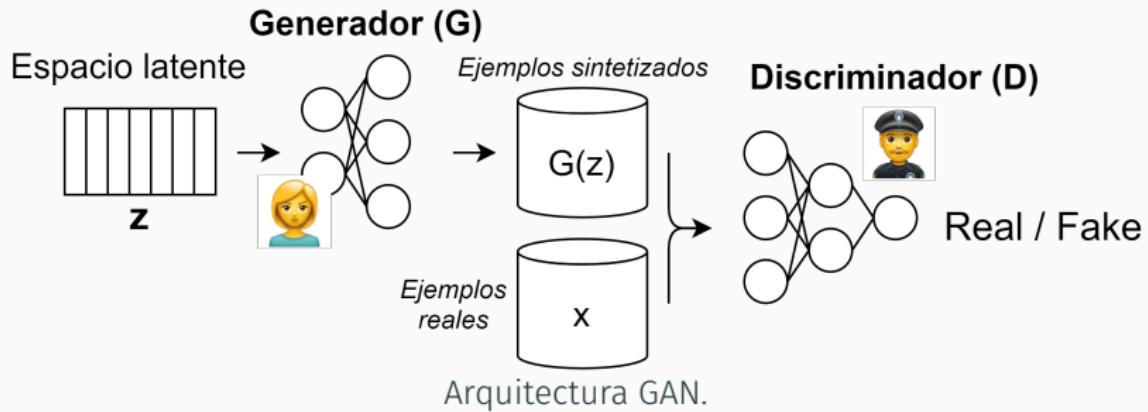


Redes GAN: Intuición previa

- El policía y el falsificador **compiten** por quien consigue vencer al otro.
- A medida que el tiempo pasa los billetes del falsificador son **más realistas** pero el policía aprende a **diferenciar mejor** los pequeños detalles.
- La competición entre ambos hacen que mejoren de manera **simultánea y paralela**.

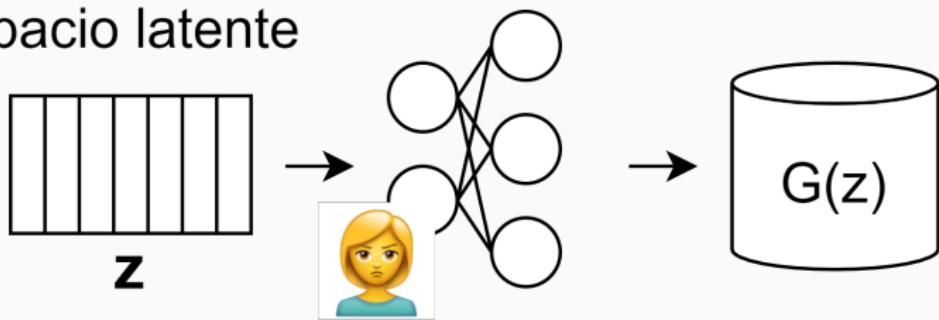


Esquema general de una red GAN



Generador (G)

Espacio latente

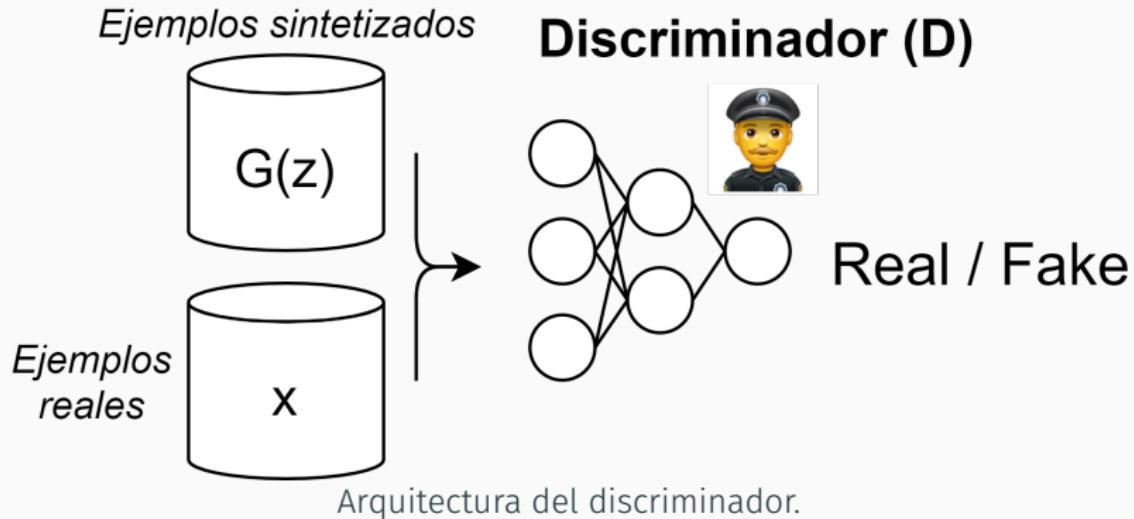


Ejemplos sintetizados

Arquitectura del generador.

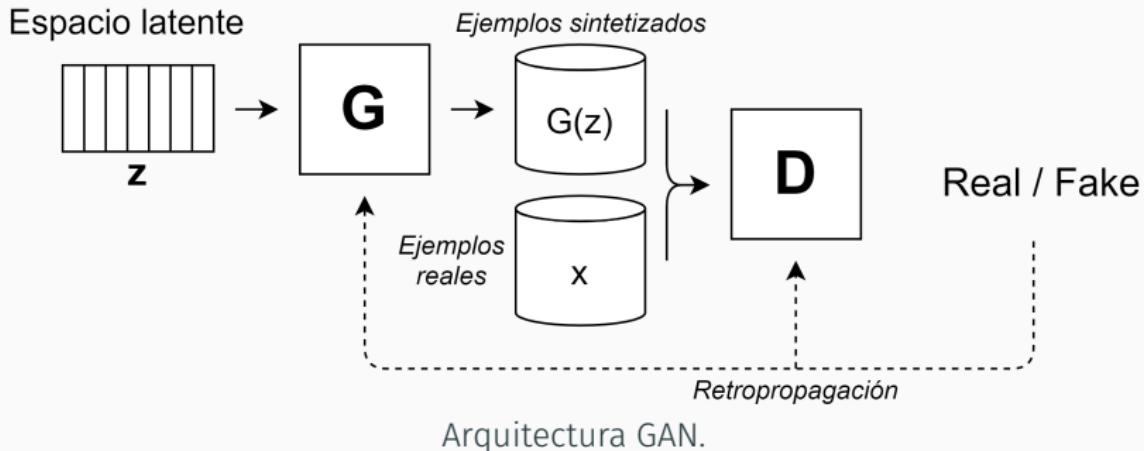
Genera datos a partir de un **espacio latente** que le proporciona aleatoriedad.

Discriminador



Discrimina los datos a partir de la distribución real y la distribución generada por el generador.

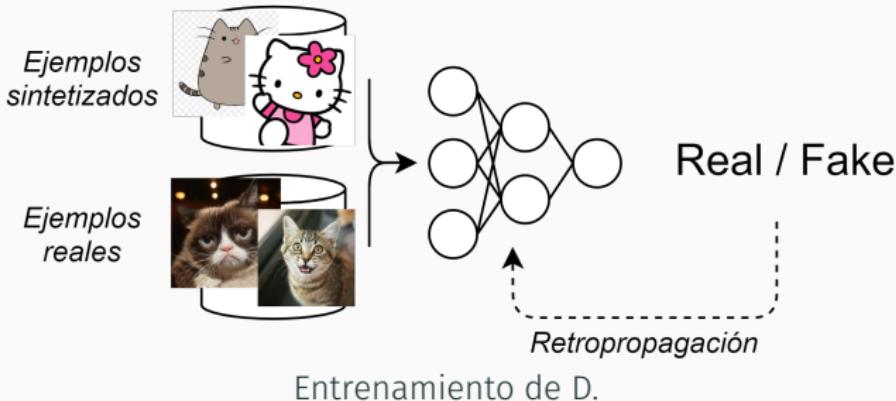
Esquema general de una red GAN



- Durante el entrenamiento de una GAN se produce una **competición** entre **G** y **D**.
- En ella ambos modelos mejoran **progresivamente** de manera **simultánea**.

GAN: Entrenamiento

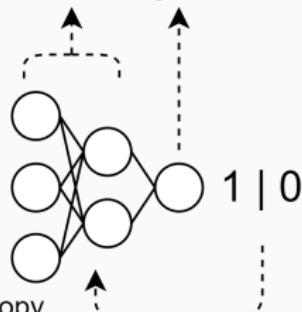
Entrenamiento del Discriminador



- El D no es más que un **clasificador** binario.
- Su objetivo es **diferenciar** qué datos son **reales** y cuales **falsos**.
- Durante su entrenamiento realiza un aprendizaje para diferenciar ambas **distribuciones**.
- Se tiene que adaptar **constantemente** a las mejoras en las imágenes generadas por el G.

Entrenamiento del Discriminador

Activaciones: ReLu Sigmoid



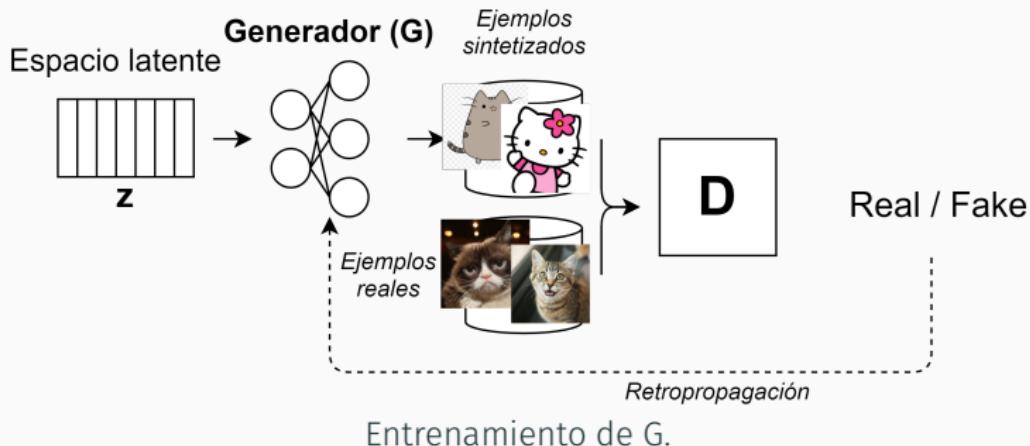
Loss: binary_crossentropy

Pérdida y activaciones de D.



- Las activations de las capas ocultas y de entrada son **cualquiera** que permita actualizar los pesos **correctamente**.
- Al ser una clasificación **binaria** la activación de la capa de salida será la sigmoide (en el rango $[0, 1]$).
- La función de **pérdida** es la Entropía cruzada binaria.

Entrenamiento del Generador



- El G genera **nuevos** datos a partir del espacio latente.
- La **actualización** de pesos se hace a través del **D**.
- Su entrenamiento **depende** de si el **D** es capaz de diferenciar los datos.

Espacio latente

Se genera a través de ruido Gausiano o uniforme.

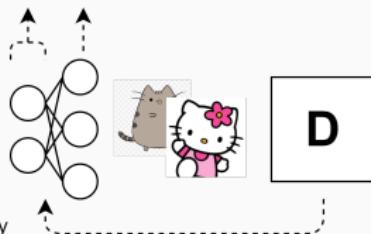
0.52	0.23	0.12	0.15	0.97
z				

Vector de espacio latente.

- Es un **vector n-dimensional** de números reales.
- Actúa de **semilla** para la generación de nuevos datos.
- Introduce la aleatoriedad necesaria a la red.

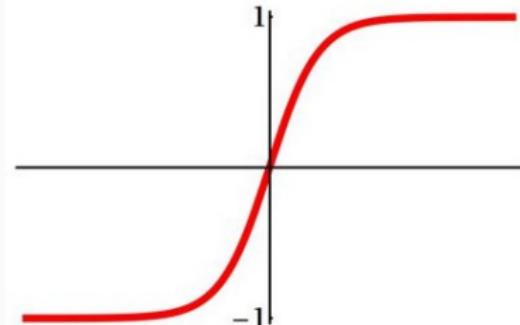
Entrenamiento del Generador

Activaciones: ReLu Tanh



Loss: binary_crossentropy

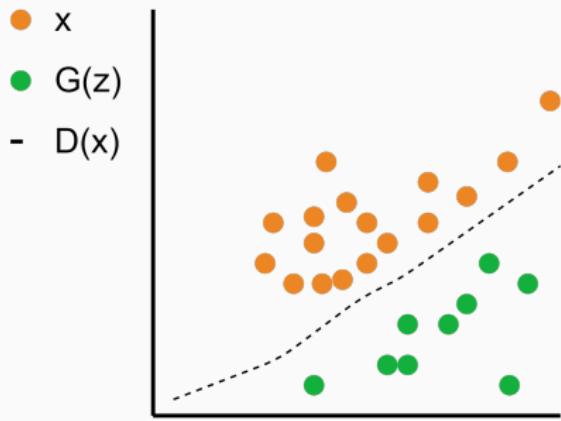
Pérdida y activaciones de G.



Función de activación tangencial.

- Las activations de las capas ocultas y de entrada son **cualquiera** que permita actualizar los pesos **correctamente**.
- La **activación** de la capa de salida será aquella que **permite generar** los datos de manera **correcta**.
- La función de **pérdida** es la Entropía cruzada binaria, con origen en la salida del D.

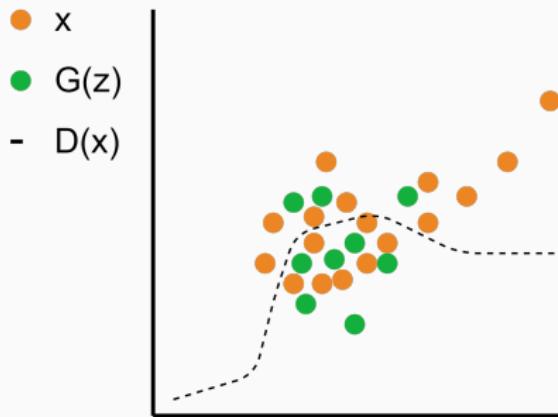
Entrenamiento: Primeras iteraciones



Distribución de datos reales
(naranja) y generados (verde)
durante las primeras iteraciones de
un entrenamiento.

- Es muy fácil la **diferenciación** entre ambas distribuciones.
- Los datos generados son de **muy mala calidad** (ruido).

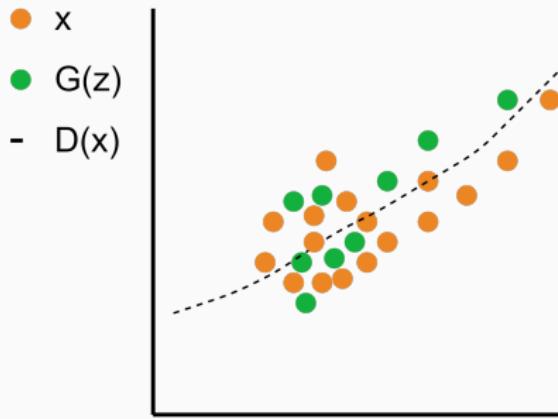
Entrenamiento: Iteraciones intermedias



Distribución de datos reales
(naranja) y generados (verde)
durante las iteraciones intermedias
de un entrenamiento.

- El generador comienza a generar datos que engañan al discriminador.
- Todavía existen diferencias notables entre ambas distribuciones.

Entrenamiento: Final del entrenamiento



Distribución de datos reales
(naranja) y generados (verde) tras
haber finalizado un entrenamiento
exitoso.

- Ambas distribuciones son **indistinguibles**.
- El discriminador es **incapaz** de **diferenciar** de qué distribución proceden los datos.
- Se ha llegado al **equilibrio de Nash**[2].

Minimax game

- El generador busca **minimizar** la siguiente expresión:

$$\log[1 - D(G(z))] \quad (1)$$

- El discriminador busca **maximizar** la siguiente expresión:

$$\log[D(x)] + \log[1 - D(G(z))] \quad (2)$$

Minimax game

- El juego de **minimax** entre ambos modelos se denota por la siguiente expresión:

$$\min_G \max_D L(D, G) = \log[D(x)] + \log[1 - D(G(x))] \quad (3)$$

- **G** intenta engañar a **D**.
- **D** intenta discriminar correctamente ambas **distribuciones**.

GAN: Problemas principales

Problemas de las GAN

Las GAN cuentan con una serie de **problemas comunes**. Estos suelen presentarse en los entrenamientos de una GAN debido a las particularidades de su **arquitectura**.

Se revisarán los siguientes:

- Inestabilidad
- Problemas derivados del gradiente:
 - Gradient explosion
 - Gradient vanishing
- Mode Collapse

El problema de la **inestabilidad** se debe a que las GAN cuentan con **2** redes neuronales al mismo tiempo.

- Las redes neuronales de por sí son modelos **inestables**.
- Al hacer que dos modelos **interactúen** entre sí se **aumenta** esta **inestabilidad**.
- Mantener una **coordinación** entre ambas redes es un **proceso delicado**.

Inestabilidad

Para intentar evitar la **inestabilidad** se suelen tomar una serie de **precauciones** que intenten equilibrar el **entrenamiento**:

- Que las **arquitecturas** de la D y la G tengan la **misma forma** pero inversa.
- En caso de que el D sea muy **certero** (caso más habitual), limitar su potencia haciendo uso de **regularización**.

Que las dos redes no estén en **sincronía** puede hacer que ninguna entrene correctamente, ya que el G **necesita** que el D falle de vez en cuando y viceversa.

Problemas del gradiente

Los problemas **derivados del gradiente** son comunes a todas las redes neuronales. Estos están **directamente influenciados** por el número de capas de la red.

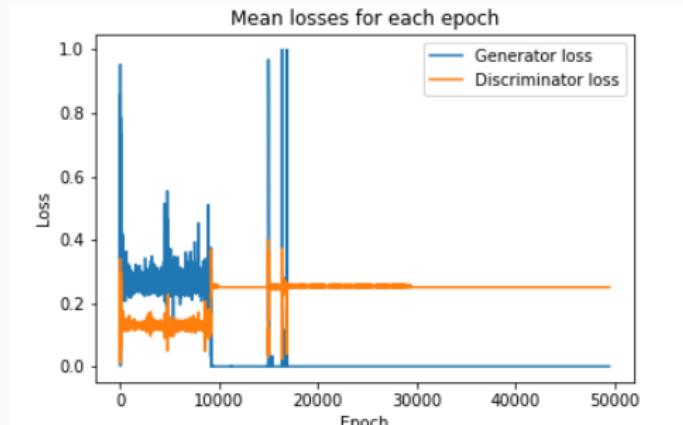
Se diferencian dos tipos:

- Gradient explosion.
- Gradient vanishing.

Al realizarse la **retropropagación** los **valores de pérdida** pasan de unas capas a otras. En este algoritmo las derivadas de cada neurona pueden llegar a **descontrolarse**.

$$W'_x = W_x - \alpha \left(\frac{\partial \text{Loss}}{\partial W_x} \right) \quad (4)$$

Problemas del gradiente: Gradient explosion

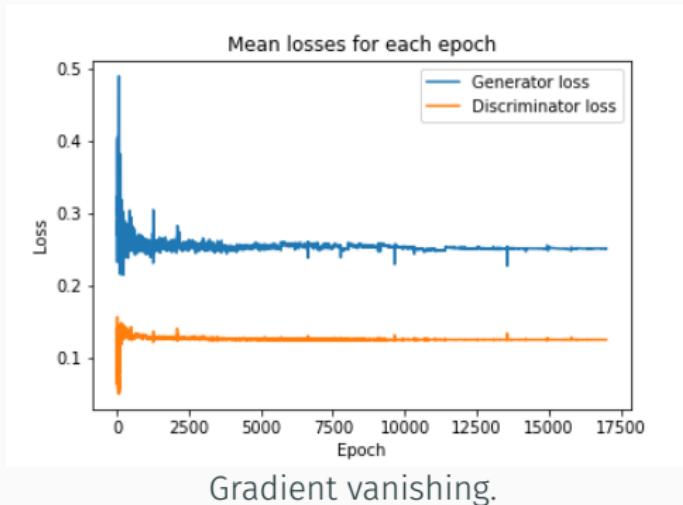


Gradient explosion.

El **gradient explosion**, también conocido como **exploding gradients** sucede cuando la actualización de pesos toma valores **muy elevados**.

Se identifica con valores de pérdidas de **NaN** o **muy exageradas**.

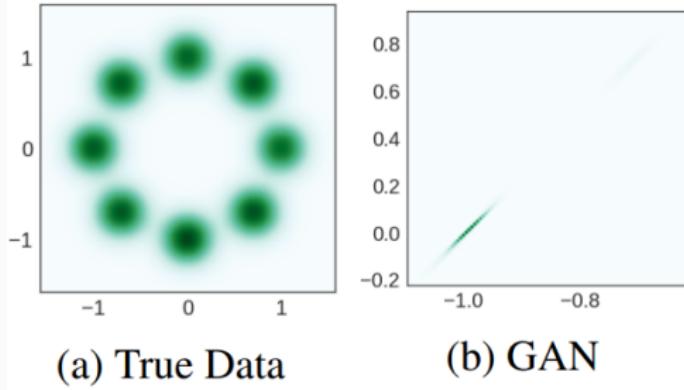
Problemas del gradiente: Gradient Vanishing



Cuando sucede **gradient vanishing**, también llamado **vanishing gradients**, la actualización de pesos se hace **nula** por tener valores muy pequeños.

Se identifica cuando la pérdida es **constante en el tiempo**.

Mode Collapse



Mode collapse, figura extraída de [3].

El **colapso modal** sucede cuando la información generada sólo pertenece a un **subgrupo** de la total.

Métricas de evaluación de métodos generativos

La "calidad" de los datos generados

¿Cómo sabemos si una GAN genera “buenas” imágenes?

A diferencia de modelos discriminativos (con métricas objetivas como accuracy o loss), los modelos generativos no tienen una métrica de evaluación directa.

La función de pérdida no se correlaciona bien con la calidad percibida.



Dos imágenes pueden ser igualmente *probables* pero muy diferentes visualmente [3].

- Mean Squared Error (MSE) o derivados loss no reflejan la percepción humana ni la diversidad del conjunto.

Necesitamos métricas que capturen al mismo tiempo:

- **Realismo**: Que los datos generados sean similares a los del *dataset* original.
- **Diversidad**: Que los datos generados no sean siempre iguales.
- **Originalidad**: Que los datos generados no sean copias del *dataset*.

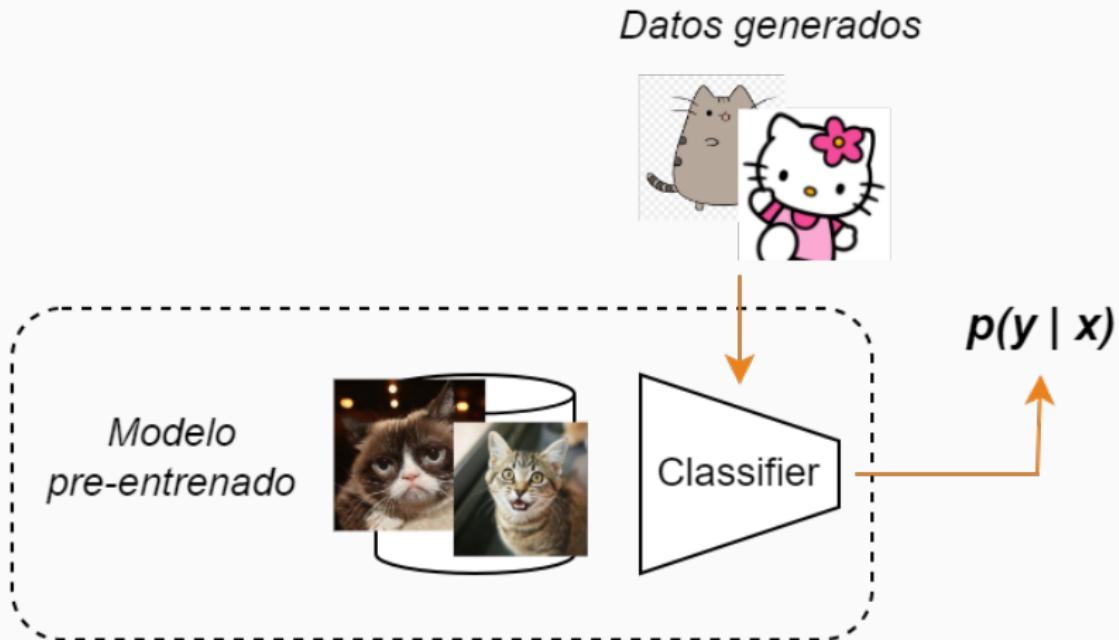
Usando un modelo auxiliar

La principal solución es utilizar un **modelo externo pre-entrenado** que evalue la calidad de los datos generados.

De esta manera se analizará el **comportamiento** del modelo ante datos generados.

- Se busca que el comportamiento del **dataset generado** sea realista
- El modelo pre-entrenado **no entrena ni sabe nada** de los datos generados

Usando un modelo auxiliar



*normalmente se usa como modelo pre-entrenado la red **Inceptionv3** [4] entrenada con el dataset Imagenet

Inception Score (IS) [5]

- Definición

$$IS = \exp(E_x[KL(p(y|x)p(y))]) \quad (5)$$

- $p(y | x)$: Predicciones de la red para los datos generados
- $p(y)$: Distribución de etiquetas predichas por el modelo

Se busca que las **predicciones** sean de una **clase concreta**. Lo que significaría que la imagen es **realista**.

Características de la IS [5]

- Fácil de calcular
- No mide **similitud** con los datos reales, sólo tiene en cuenta los generados
- **Depende del clasificador**
 - Funciona mejor cuanto más similar sea el dataset a imagenet
 - Que es un dataset de fotos reales

- Definición

$$FID = \mu_r \mu_g^2 + Tr(\Sigma_r + \Sigma_g 2(\Sigma_r \Sigma_g)^{1/2}) \quad (6)$$

- Compara las **distribuciones** (media μ y covarianza Σ) de features de Inception entre datos reales y generados
- Las features son las salidas de una de **las últimas capas** de InceptionV3, normalmente la penúltima

Características de la FID [6]

- Compara la **distribución de datos real** y la **generada**
- Mide **realismo y diversidad**
- Sensible al tamaño de la muestra
- Supone que las predicciones se organizan con una **distribución gausiana**

Kernel Inception Distance (KID) [7]

- Definición

$$KID = MMD^2(\phi(x_r), \phi(x_g)) \quad (7)$$

- Calcula la Maximum Mean Discrepancy (MMD) entre las features de Inceptionv3
- De esta manera no se asume nada sobre las distribuciones de las features
- Es especialmente útil para muestras pequeñas

GAN: Aplicaciones

Las **GAN** son un tipo de arquitectura muy ligada a sus **aplicaciones en el mundo real**. Al tratarse de **modelos generativos** han despertado un gran interés en la sociedad.

- Generación de datos.
- Domain-to-domain translation.
- Data augmentation

Generación de datos aleatorios

Es la aplicación **directa** de las GAN.

- Consiste en generar datos de cierto tipo **sin control** sobre la salida.
- Durante los últimos años se ha conseguido **mejorar mucho la calidad** de los datos generados.

Generación de datos aleatorios: Ejemplos



Resultados de la arquitectura
BigGAN[8].



Resultados de la arquitectura
Alias-Free GAN (StyleGAN-3)[9].

Domain-to-domain translation

El **Domain-to-domain translation** es una aplicación **no exclusiva** de las GAN. Sin embargo estas redes funcionan muy bien para esta aplicación.

Es una de las aplicaciones **más vistosas** y que más ha ayudado a la popularidad de las GAN en los últimos años.

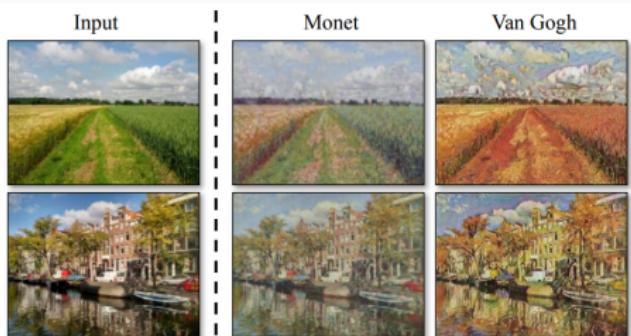
- La idea es traducir un conjunto de datos de entrada de **un dominio a otro distinto**.

Domain-to-domain translation: Ejemplos

This flower has yellow petals along with green and yellow stamen



Resultados de la arquitectura TAC-GAN[10].



Resultados de la arquitectura CycleGAN[11].

Data augmentation

Data Augmentation consiste en generar **nuevos datos** de un dataset para aumentar el número de ejemplos. Esto es especialmente útil cuando se cuenta con conjuntos de datos **insuficientes** para entrenar, por ejemplo, una red neuronal.

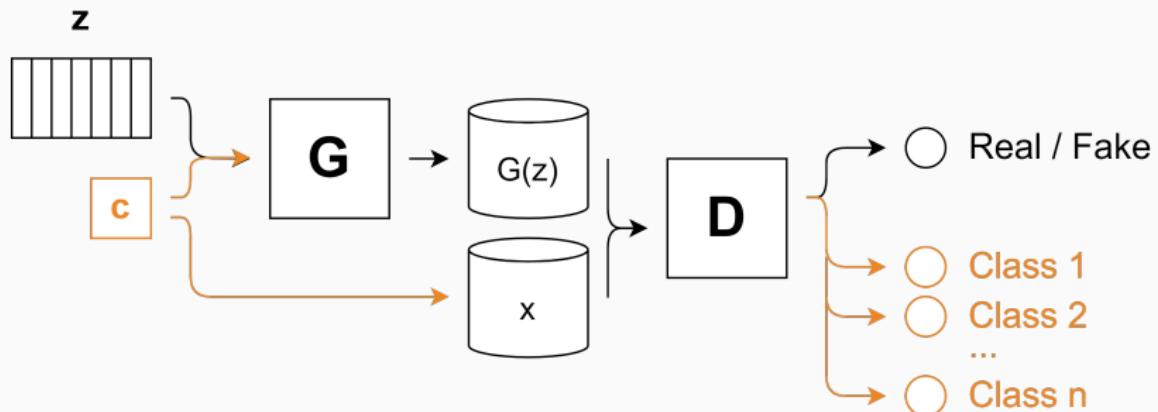
También toma especial importancia cuando se cuenta con **desbalanceo** entre clases de un dataset.

Las GAN tienen como principal ventaja que generan nuevos datos muy similares a la distribución de datos original, pero sin copiar ni modificarlos directamente.

GAN: Arquitecturas

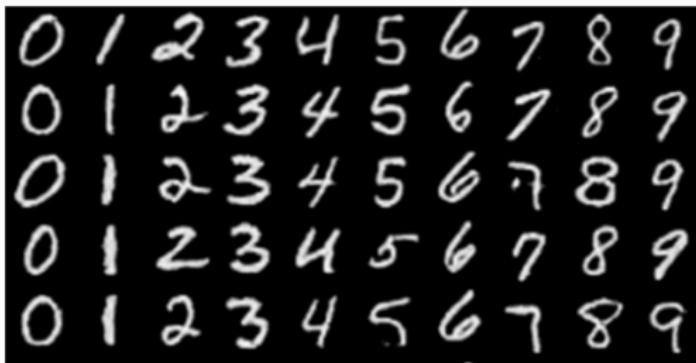
Auxiliary Classifier GAN (ACGAN)[12]

- Capaz de separar en distintas **clases** los datos con los que trata.
- Realiza un entrenamiento **supervisado** usando las etiquetas del dataset.
- El discriminador distingue además de la **veracidad** de cada dato, la clase a la que pertenece. De esta manera es capaz de aprender los distintos tipos de datos.



Arquitectura de una ACGAN

- Capaz de separar en distintas **clases** los datos con los que trata.
- Realiza un entrenamiento **supervisado** usando las etiquetas del dataset.
- El discriminador distingue además de la **veracidad** de cada dato, la clase a la que pertenece. De esta manera es capaz de aprender los distintos tipos de datos.



Resultados de una ACGAN para la generación de dígitos del dataset MNIST[13]

Image-to-Image Translation with Conditional Adversarial Nets (pix2pix)[14]

- La arquitectura pix2pix fue presentada para realizar generación de imágenes condicionada por una entrada, que también es una imagen.
- La idea es traducir la imagen de entrada a otro dominio, en lo que se conoce como image-to-image translation.

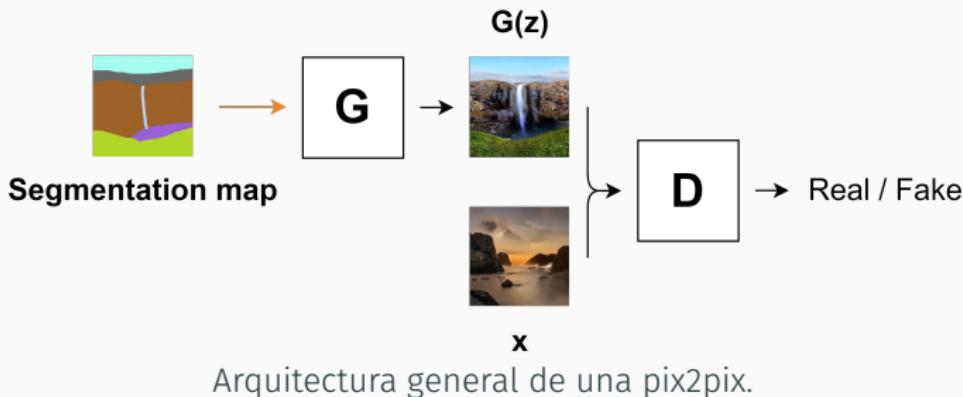


Image-to-Image Translation with Conditional Adversarial Nets (pix2pix)[14]

- El discriminador se encarga de juzgar la veracidad de cada uno de los **parches** en los que se divide una imagen. Esta arquitectura se conoce como **PatchGAN**[14].
- De esta manera se consigue que cada sección de la imagen sea realista.



Discriminador de la arquitectura PatchGAN[14].

Image-to-Image Translation with Conditional Adversarial Nets (pix2pix)[14]

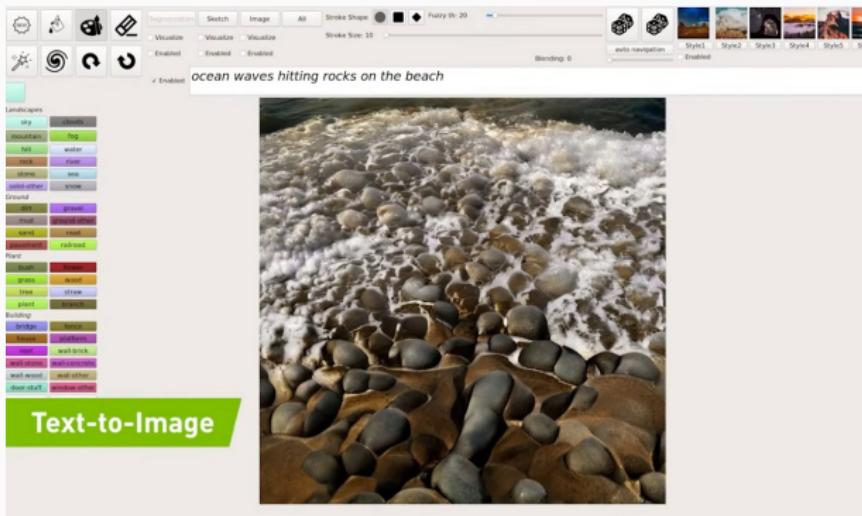
- El nuevo discriminador necesita una **función de pérdida** que permita unir la clasificación de todos los **parches**.

$$L = \sum [\log D(x_i) + \log (1 - D(G(z_i))) + \lambda L_{L1}(G)] \quad (8)$$

$$L_{L1}(G) = y - G(z, x) \quad (9)$$

GauGAN[15]

- Arquitectura presentada por NVIDIA que consigue mejorar aún más los resultados de la pix2pix y pix2pixHD.



<http://gaugan.org/gaugan2/>.

Image-to-image translation

- CGAN, 2014[16]
- ACGAN, 2016[12]
- pixpix, 2018[14]
- CycleGAN, 2017[14]
- DiscoGAN, 2017[17]
- GauGAN, 2019[15]
- CSGAN, 2019[18]

Mejoras en la calidad

- DCGAN, 2016[19]
- ProGAN, 2017[20]
- StyleGAN, 2018[21]
- Alias-FreeGAN, 2021[9]
- CSGAN, 2021[18]

Super resolution

- SRGAN, 2017[22]
- srcaGAN, 2021[23]
- WSRGAN, 2021[24]

Crecimiento de las redes

- ProGAN, 2017[20]
- DGGAN, 2021[25]

Referencias i

- [1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.
Generative adversarial nets.
Advances in neural information processing systems, 27, 2014.
- [2] AA Cournot and Augustin Cournot.
Researches into the mathematical principles of the theory of wealth.
1897.
- [3] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton.
Veegan: Reducing mode collapse in gans using implicit variational learning.
Advances in neural information processing systems, 30, 2017.

Referencias ii

- [4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna.
Rethinking the inception architecture for computer vision (2015).
arXiv preprint arXiv:1512.00567, 2015.
- [5] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.
Improved techniques for training gans.
Advances in neural information processing systems, 29, 2016.
- [6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.
Gans trained by a two time-scale update rule converge to a local nash equilibrium.
Advances in neural information processing systems, 30, 2017.

Referencias iii

- [7] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton.
Demystifying mmd gans.
arXiv preprint arXiv:1801.01401, 2018.
- [8] Andrew Brock, Jeff Donahue, and Karen Simonyan.
Large scale gan training for high fidelity natural image synthesis.
arXiv preprint arXiv:1809.11096, 2018.
- [9] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila.
Alias-free generative adversarial networks.
Advances in Neural Information Processing Systems, 34:852–863, 2021.

- [10] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, and Muhammad Zeshan Afzal.
Tac-gan-text conditioned auxiliary classifier generative adversarial network.
arXiv preprint arXiv:1703.06412, 2017.
- [11] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros.
Unpaired image-to-image translation using cycle-consistent adversarial networks.
In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
- [12] A Odena, C Olah, and J Shlens.
Conditional image synthesis with auxiliary classifier gans arxiv e-prints.(oct.
arXiv preprint stat.ML/1610.09585, 2016.

Referencias v

- [13] Yann LeCun.
The mnist database of handwritten digits.
<http://yann.lecun.com/exdb/mnist/>, 1998.
- [14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros.
Image-to-image translation with conditional adversarial networks.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [15] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu.
Semantic image synthesis with spatially-adaptive normalization.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

- [16] Mehdi Mirza and Simon Osindero.
Conditional generative adversarial nets.
arXiv preprint arXiv:1411.1784, 2014.
- [17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim.
Learning to discover cross-domain relations with generative adversarial networks.
In *International conference on machine learning*, pages 1857–1865. PMLR, 2017.
- [18] Kishan Babu Kancharagunta and Shiv Ram Dubey.
Csgan: Cyclic-synthesized generative adversarial networks for image-to-image transformation.
arXiv preprint arXiv:1901.03554, 2019.

- [19] Alec Radford, Luke Metz, and Soumith Chintala.
Unsupervised representation learning with deep convolutional generative adversarial networks.
arXiv preprint arXiv:1511.06434, 2015.
- [20] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of gans for improved quality, stability, and variation.
arXiv preprint arXiv:1710.10196, 2017.
- [21] Tero Karras, Samuli Laine, and Timo Aila.
A style-based generator architecture for generative adversarial networks. arxiv e-prints.
arXiv preprint arXiv:1812.04948, 2018.

- [22] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al.
Photo-realistic single image super-resolution using a generative adversarial network.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.
- [23] Baozhong Liu and Ji Chen.
A super resolution algorithm based on attention mechanism and srgan network.
IEEE Access, 9:139138–139145, 2021.

- [24] Hangpu Cao and Sicheng Mi.
Weighted srgan and reconstruction loss analysis for accurate image super resolution.
In *Journal of Physics: Conference Series*, volume 1903, page 012050. IOP Publishing, 2021.
- [25] Lanlan Liu, Yuting Zhang, Jia Deng, and Stefano Soatto.
Dynamically grown generative adversarial networks.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8680–8687, 2021.

Contribuciones de las diapositivas

- Autor original de las diapositivas: Guillermo Iglesias Hernández