07MIAR – Redes neuronales y deep learning

Universidad Internacional de Valencia

Generative Adversarial Networks para la síntesis de imagen

08/11/2021

De:
Planeta Formación y Universidades

01
Introducción

Generative Adversarial Networks para la síntesis de imagen

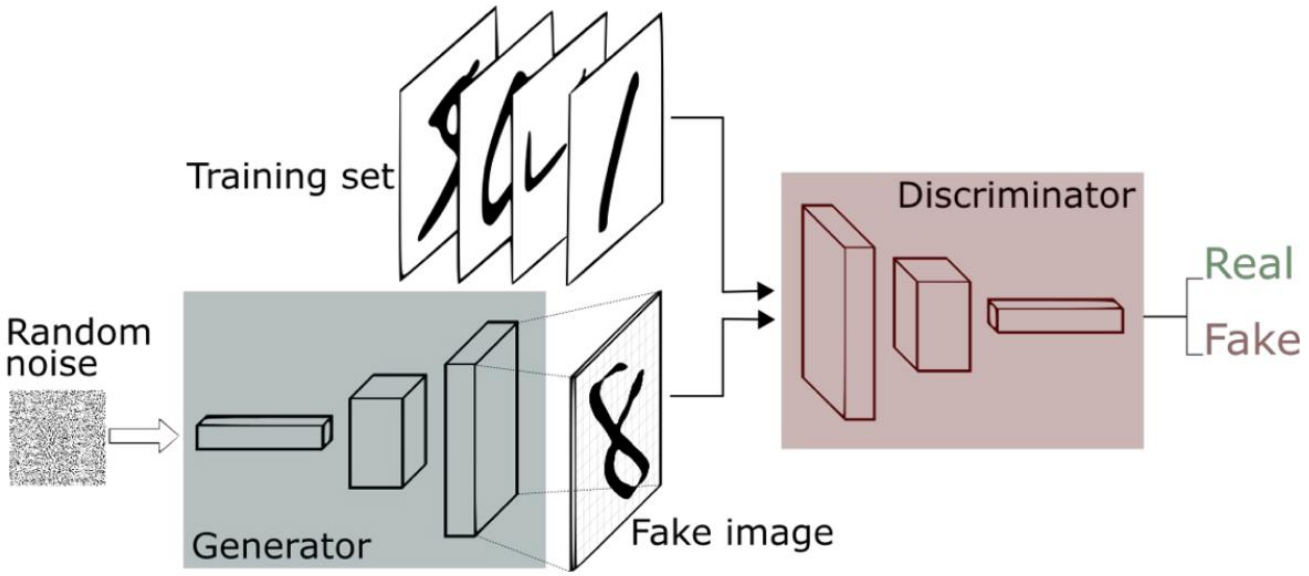
08/11/2021

Generative Adversarial Network: Contexto de la arquitectura

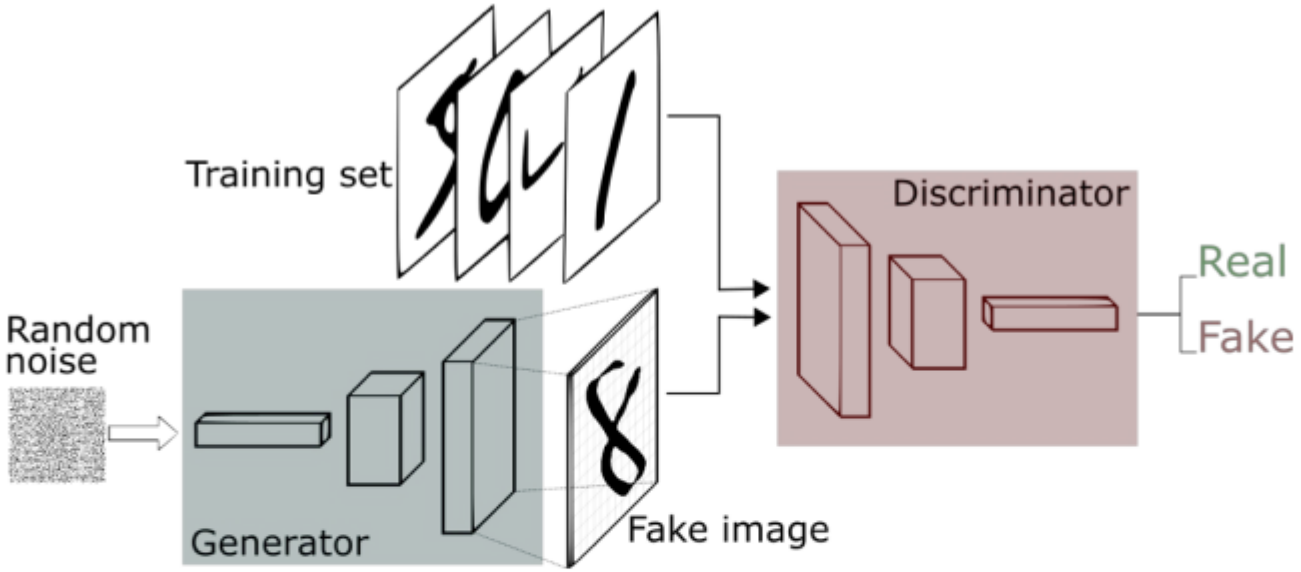
- Una **Generative Neural Network** (GAN) es un tipo de arquitectura de red neuronal para llevar a cabo tareas de **modelado generativo** propuesta por **Goodfellow** et al. en **2014**.
- El modelado generativo implica el uso de un modelo que tiene como objetivo **generar nuevos ejemplos** que probablemente vengan de una distribución existente de muestras pero a su vez serán distintos de una población de instancias existente.
- Una GAN se entrena a partir de **dos modelos de arquitectura de red**. Un **generador** que aprende a generar nuevas muestras y un **discriminador** que aprende a identificar instancias generadas por el generador de instancias reales.
- El **objetivo** de una GAN es que el nuevo **contenido generado** a partir de ruido a la entrada (inicialización) sea tan **realista** (tras ciertas épocas) como para **confundir** al **discriminador**. Mapeo entre vector o matriz de elementos aleatorios a contenido realista.
- Tras el entrenamiento, el modelo generativo será capaz de **crear nuevas muestras sintéticas a demanda.**
- **Muy utilizadas** en el ámbito de la **visión por computador** pero no exclusivamente

Generative Adversarial Network: Funcionamiento

- El **generador** (“falsificador”) trata de generar contenido suficientemente realista como para **engañar** al **discriminador** (“policía”)
- El objetivo del **discriminador** es **discernir** el contenido **real** del **falso** (creado por el generador)
- Se trata de un juego **min-max** en el que compiten entre ellos, i.e. **tarea adversarial**



Generative Adversarial Network: Funcionamiento



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

probabilidad de que D(x) prediga que los datos reales **son** verdaderos

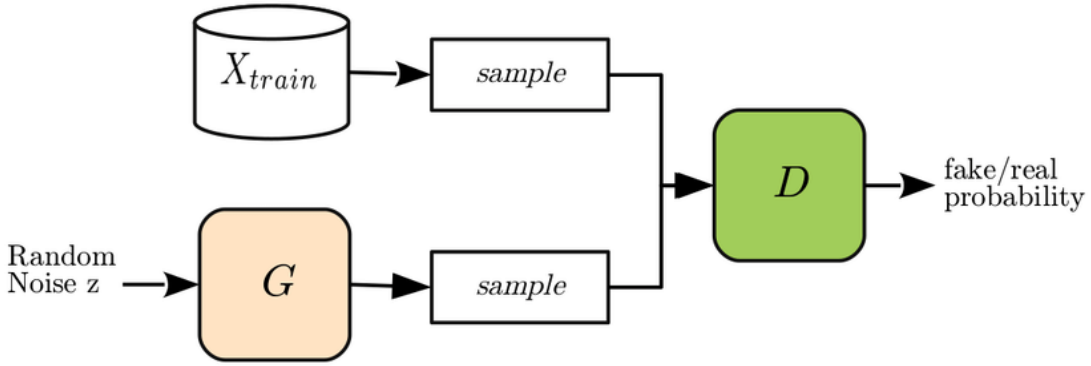
probabilidad de que D(x) prediga que los datos generados, x=G(z), **no** son verdaderos

Generative Adversarial Network: Entrenamiento

- El **entrenamiento** de una GAN se lleva a cabo en dos fases:

1. El **discriminador recibe** imágenes **reales** (etiqueta **1**) y **falsas** (etiqueta **0**) y debe aprender a **discernir** correctamente **ambos tipos de imagen**. Cuando **se equivoca**, se produce un **error** y sus **pesos** se **actualizan** convenientemente.

2. Con los **pesos del discriminador congelados**, el **generador introduce imágenes generadas (*fake*)** con **etiqueta** "real" (**1**) **al discriminador**. Cuando el **discriminador** predice que la **imagen** es **falsa**, al no coincidir la etiqueta con la predicción, se genera un **error** que permite **actualizar los pesos** del generador para que aprenda a sintetizar imágenes cada vez más realistas



Implementaciones

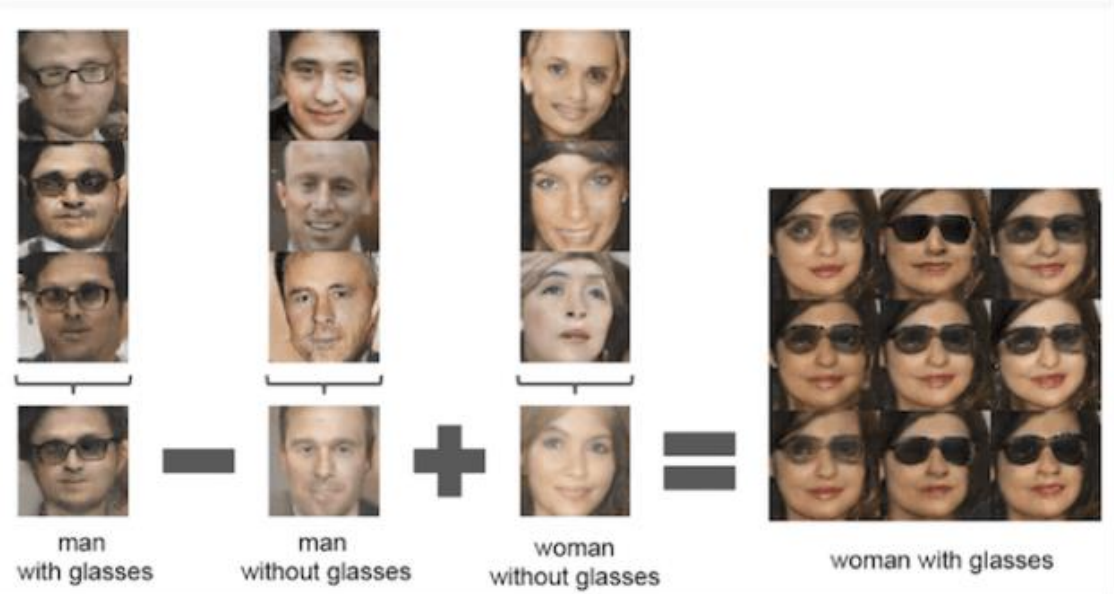
- Keras: **https://github.com/eriklindernoren/Keras-GAN/blob/master/cyclegan/cyclegan.py**
- Pytorch: **https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix**

Aplicaciones GAN: Generación de imagen

Examples of GANs used to Generate New Plausible Examples for Image Datasets.Taken from Generative Adversarial Nets, 2014.

Example of GAN-Generated Photographs of Bedrooms.Taken from Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015.

Aplicaciones GAN: Generación de imagen



Example of Vector Arithmetic for GAN-Generated Faces.Taken from Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015.

Examples of Photorealistic GAN-Generated Faces.Taken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.

Aplicaciones GAN: Generación de imagen

Example of Photorealistic GAN-Generated Objects and ScenesTaken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.

Example of the Progression in the Capabilities of GANs from 2014 to 2017.Taken from The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.

Aplicaciones GAN: Generación de imagen

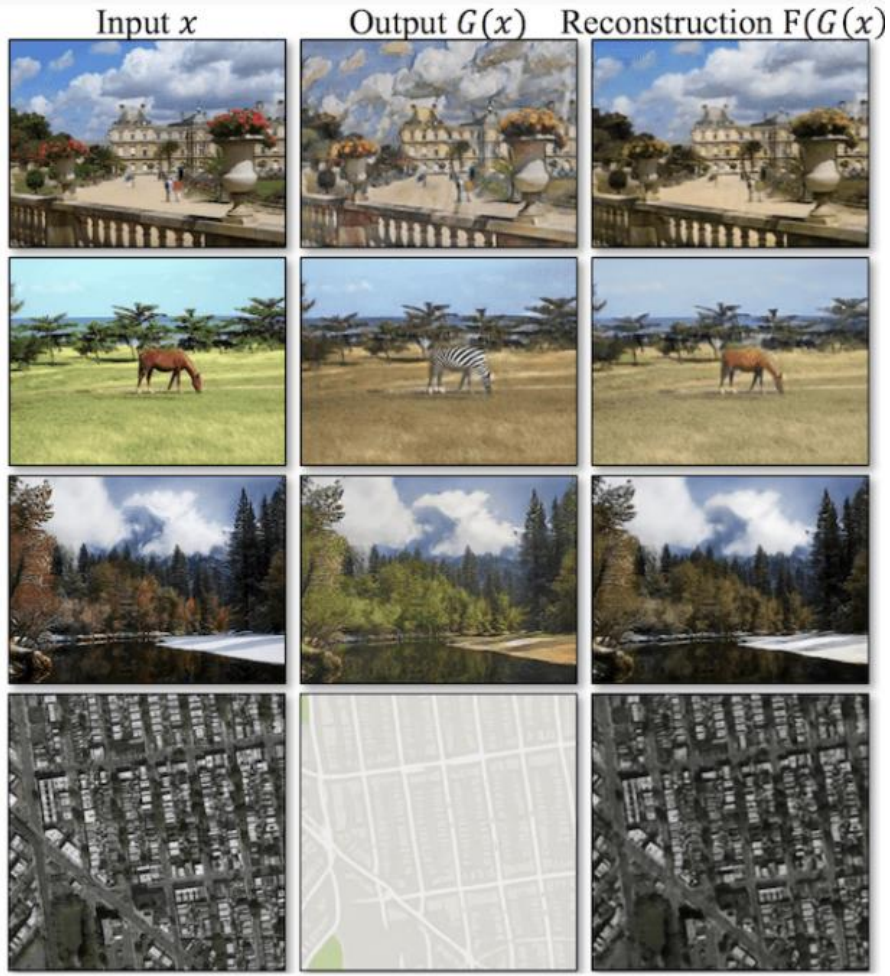
Example of Realistic Synthetic Photographs Generated with BigGANTaken from Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018.

Aplicaciones GAN: Image-to-Image Traslation

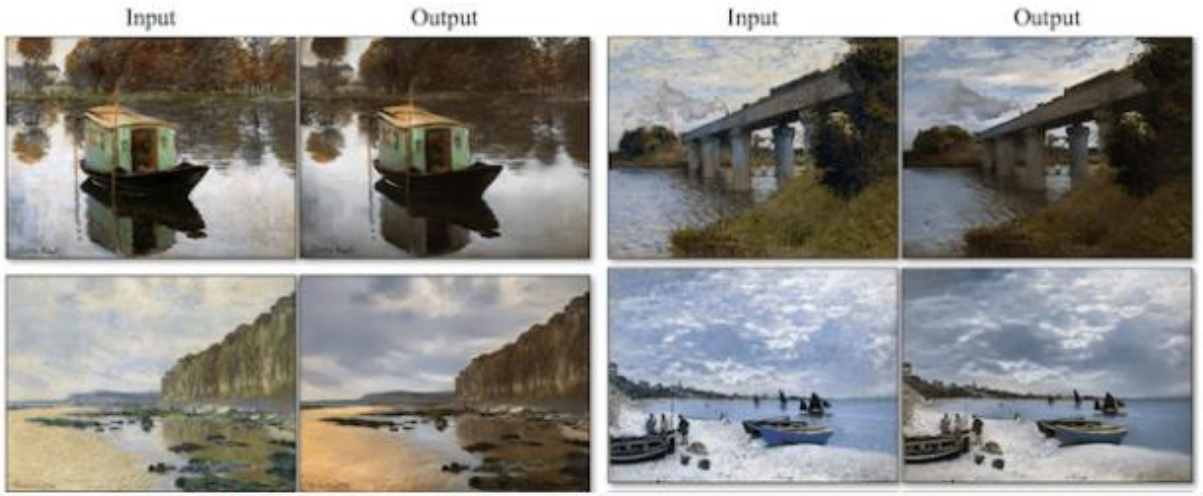
Example of Photographs of Daytime Cityscapes to Nighttime With pix2pix.Taken from Image-to-Image Translation with Conditional Adversarial Networks, 2016.

Example of Sketches to Color Photographs With pix2pix.Taken from Image-to-Image Translation with Conditional Adversarial Networks, 2016.

Aplicaciones GAN: Image-to-Image Traslation

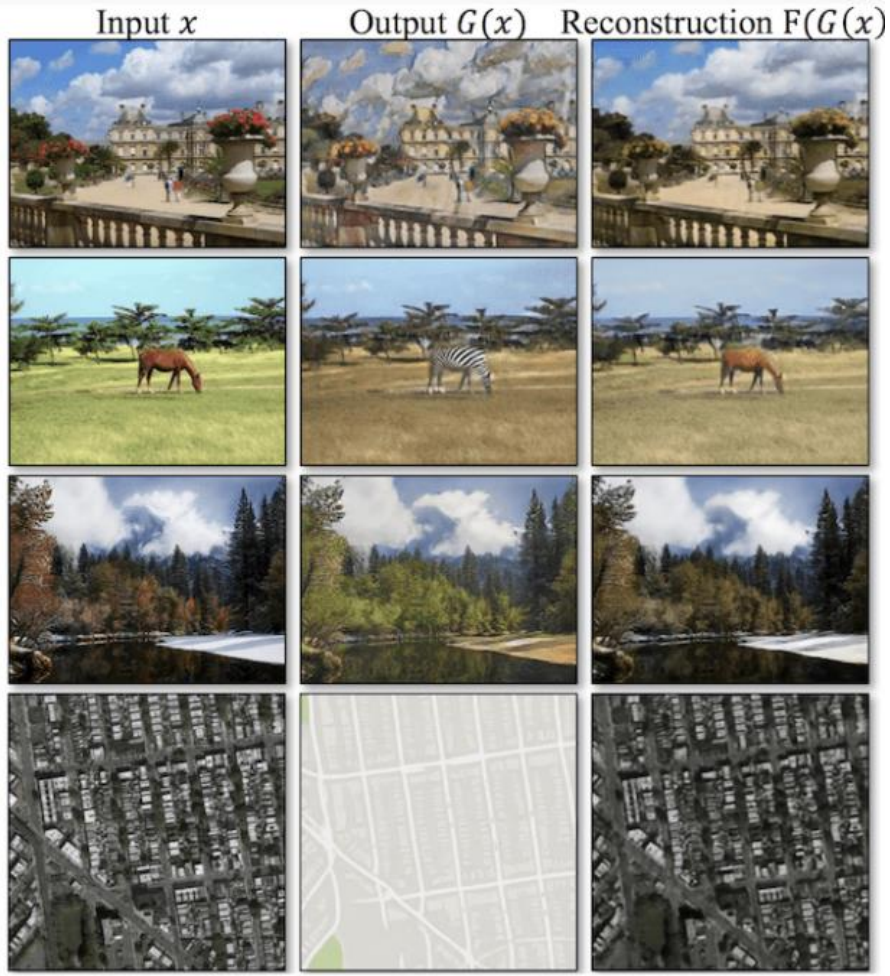


Example of Four Image-to-Image Translations Performed With CycleGANTaken from Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017.

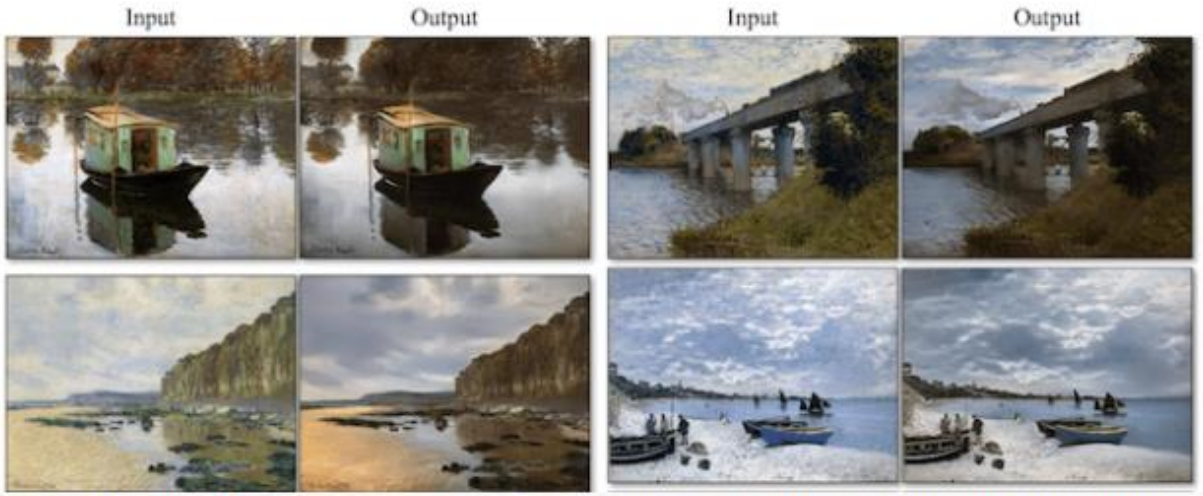


Example of Translation from Paintings to Photographs With CycleGAN.Taken from Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017.

Aplicaciones GAN: Image-to-Image Traslation

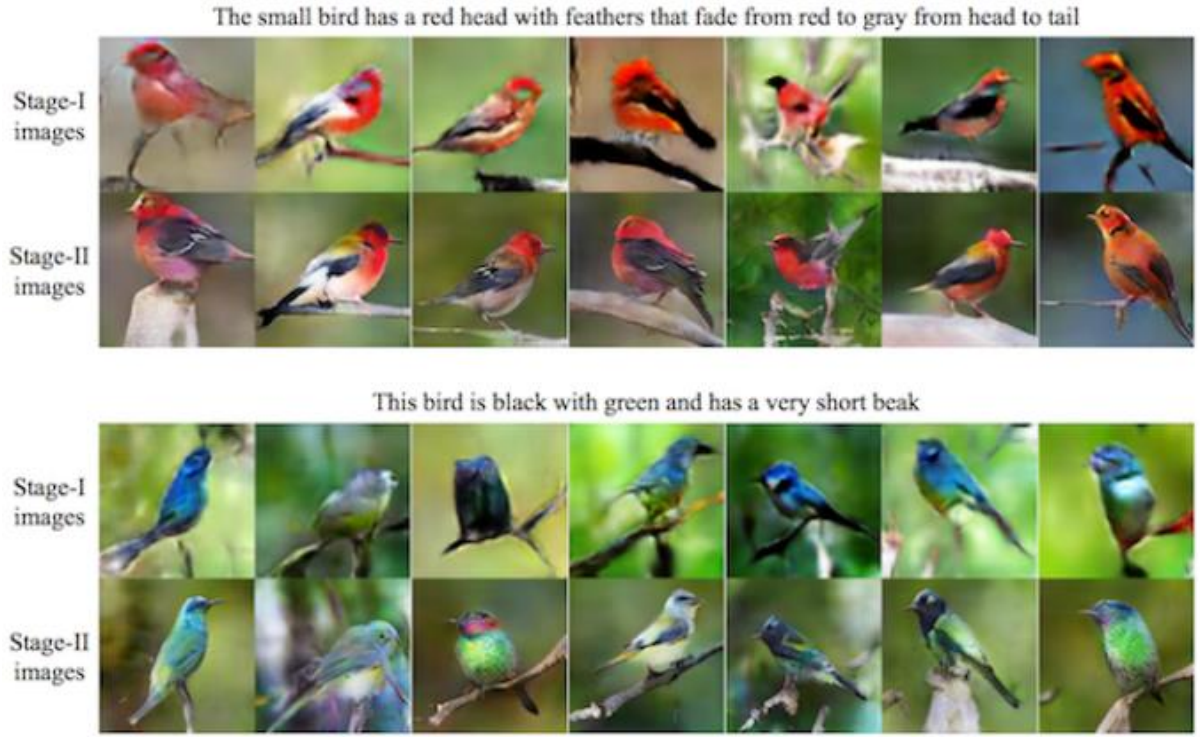


Example of Four Image-to-Image Translations Performed With CycleGANTaken from Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017.

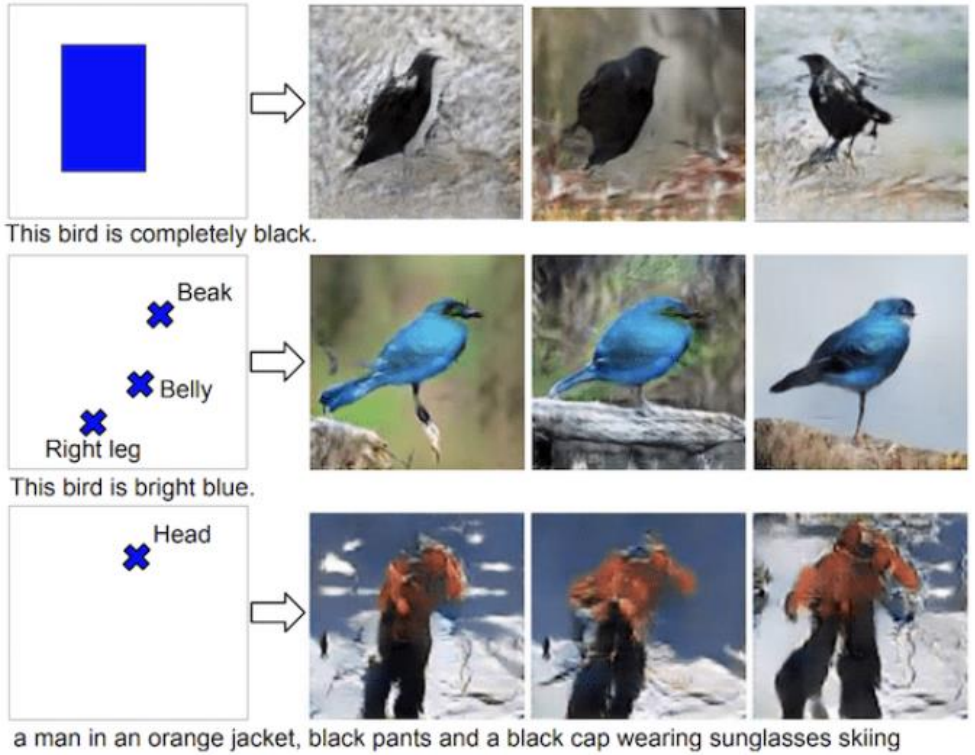


Example of Translation from Paintings to Photographs With CycleGAN.Taken from Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017.

Aplicaciones GAN: Text-to-Image Traslation



Example of Textual Descriptions and GAN-Generated Photographs of BirdsTaken from StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2016.

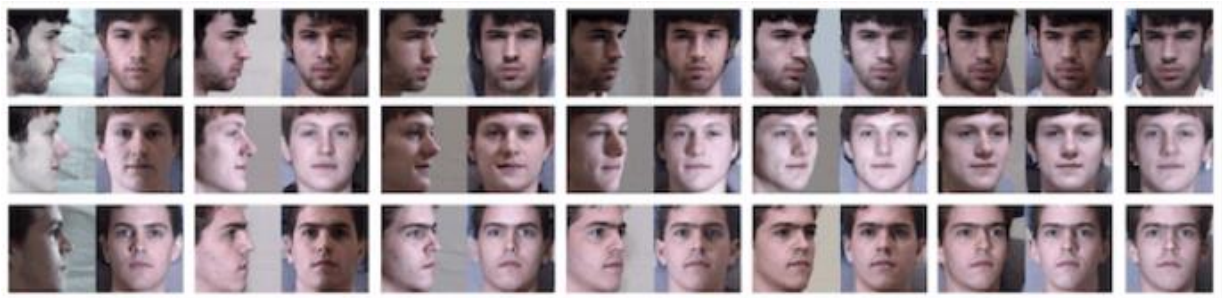


Example of Photos of Object Generated From Text and Position Hints With a GAN.Taken from Learning What and Where to Draw, 2016.

Aplicaciones GAN: Semantic-Image-to-Photo Traslation

Example of Semantic Image and GAN-Generated Cityscape Photograph.Taken from High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, 2017.

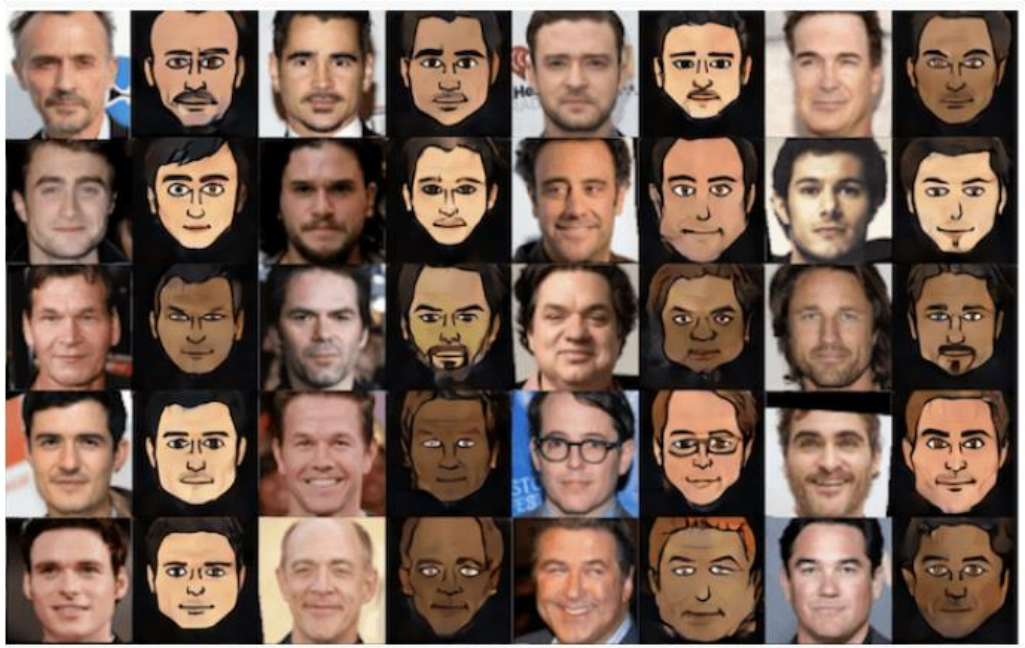
Aplicaciones GAN: Generating new human poses



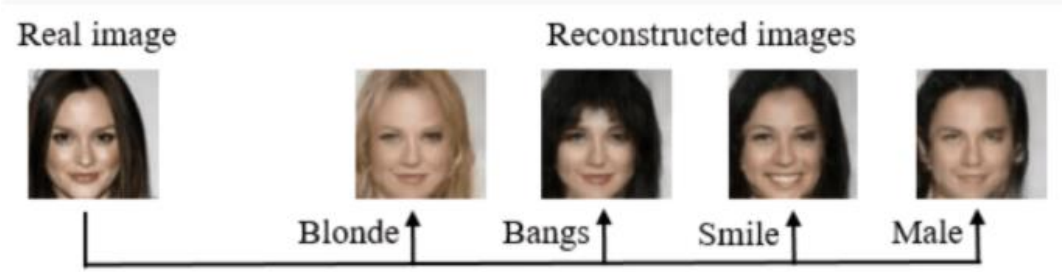
Example of GAN-based Face Frontal View Photo GenerationTaken from Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis, 2017.

Example of GAN-Generated Photographs of Human PosesTaken from Pose Guided Person Image Generation, 2017.

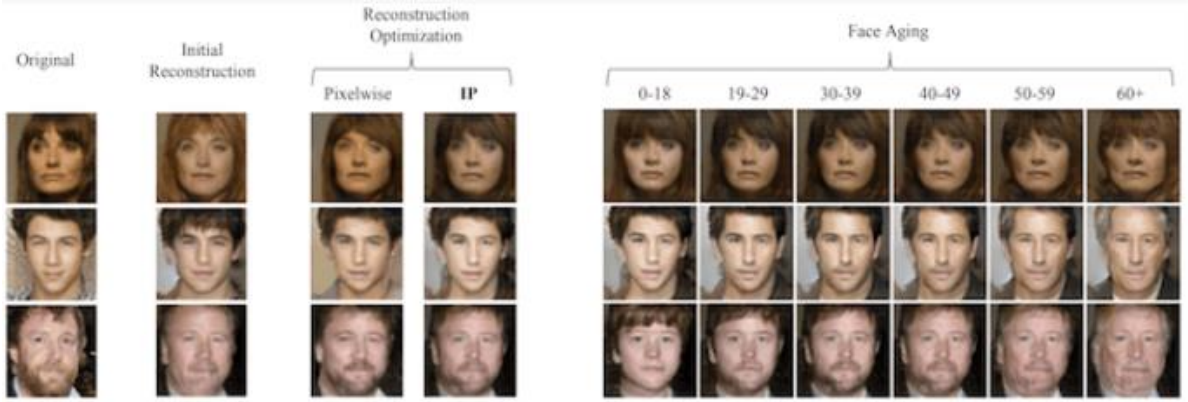
Aplicaciones GAN: Photograph editing



Example of Celebrity Photographs and GAN-Generated Emojis.Taken from Unsupervised Cross-Domain Image Generation, 2016.

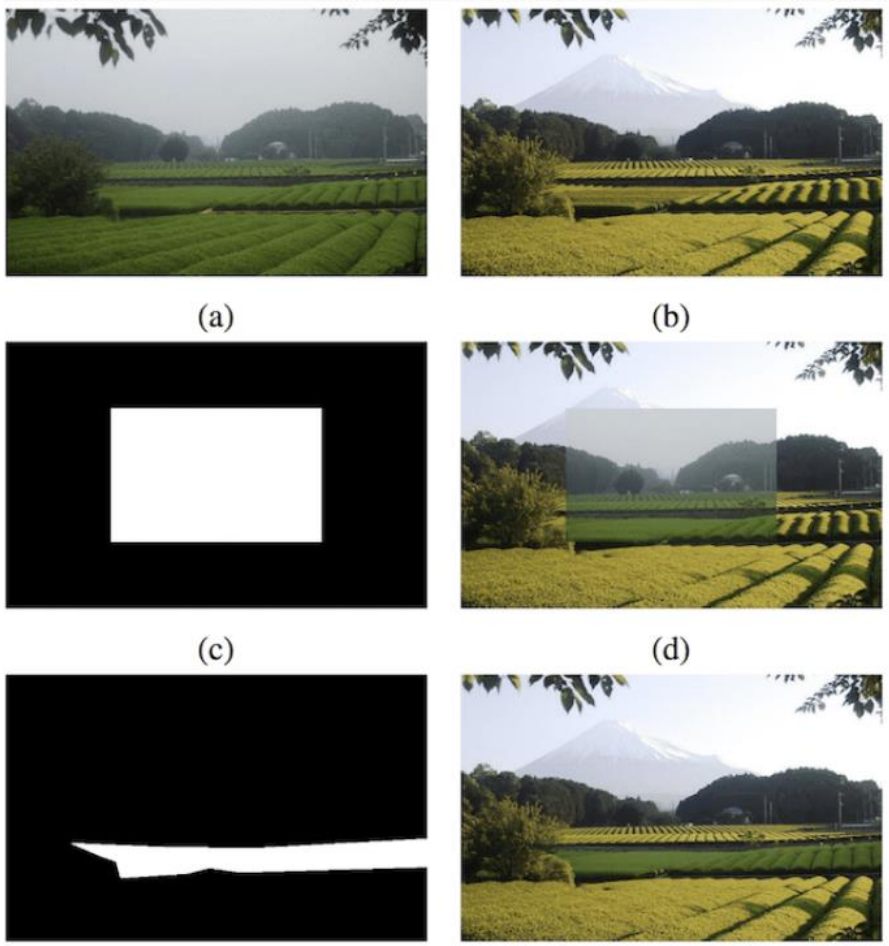


Example of Face Photo Editing with IcGAN.Taken from Invertible Conditional GANs For Image Editing, 2016.

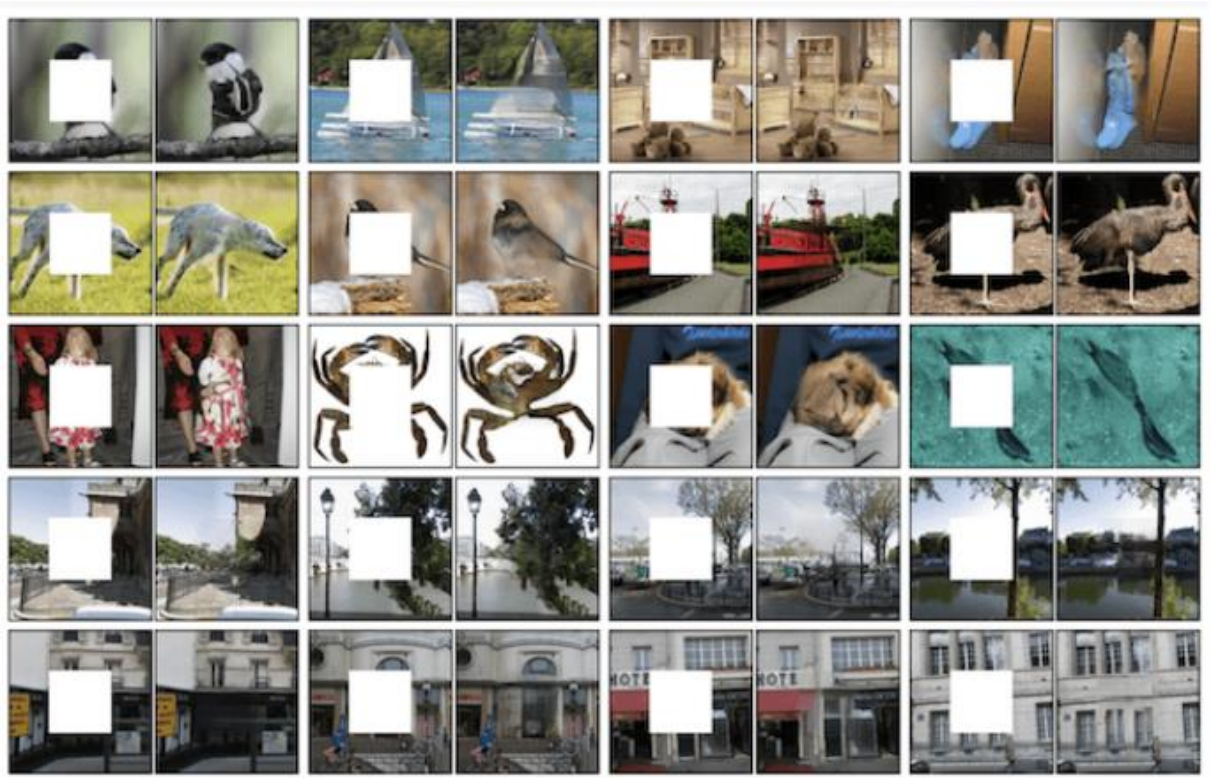


Example of Photographs of Faces Generated With a GAN With Different Apparent Ages.Taken from Face Aging With Conditional Generative Adversarial Networks, 2017.

Aplicaciones GAN: Photo blending & inpainting

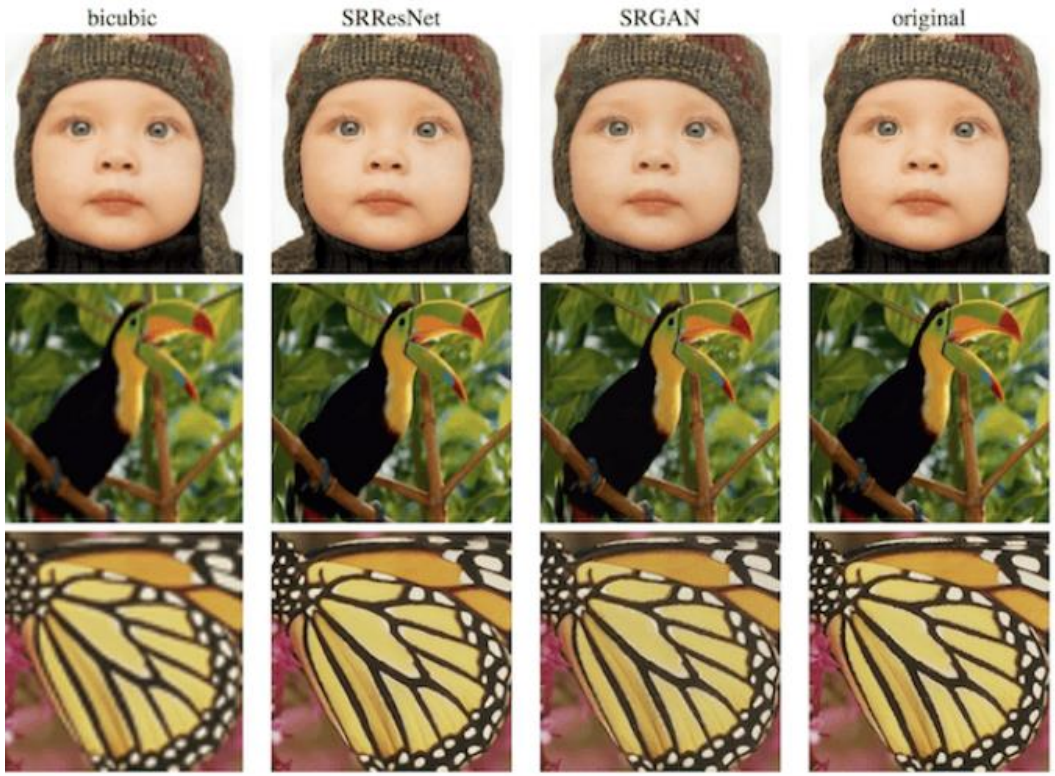


Example of GAN-based Photograph Blending.Taken from GP-GAN: Towards Realistic High-Resolution Image Blending, 2017.

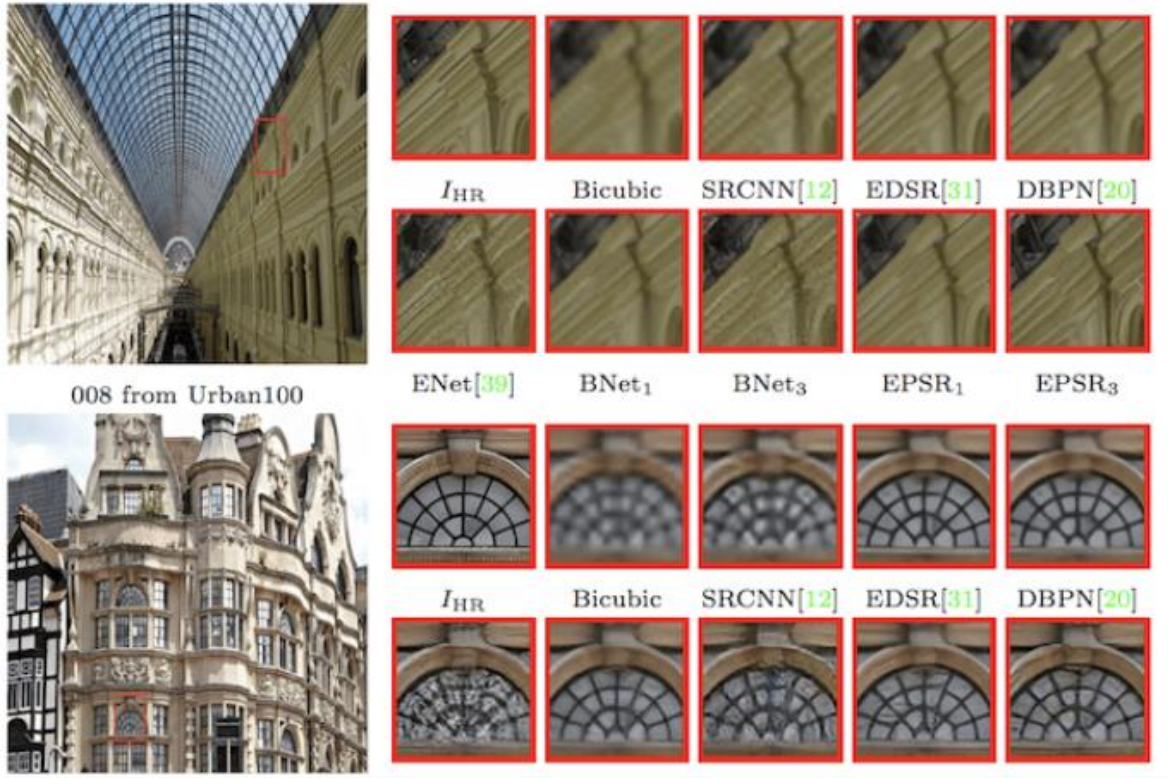


Example of GAN-Generated Photograph Inpainting Using Context Encoders.Taken from Context Encoders: Feature Learning by Inpainting describe the use of GANs, specifically Context Encoders, 2016.

Aplicaciones GAN: SuperResolution

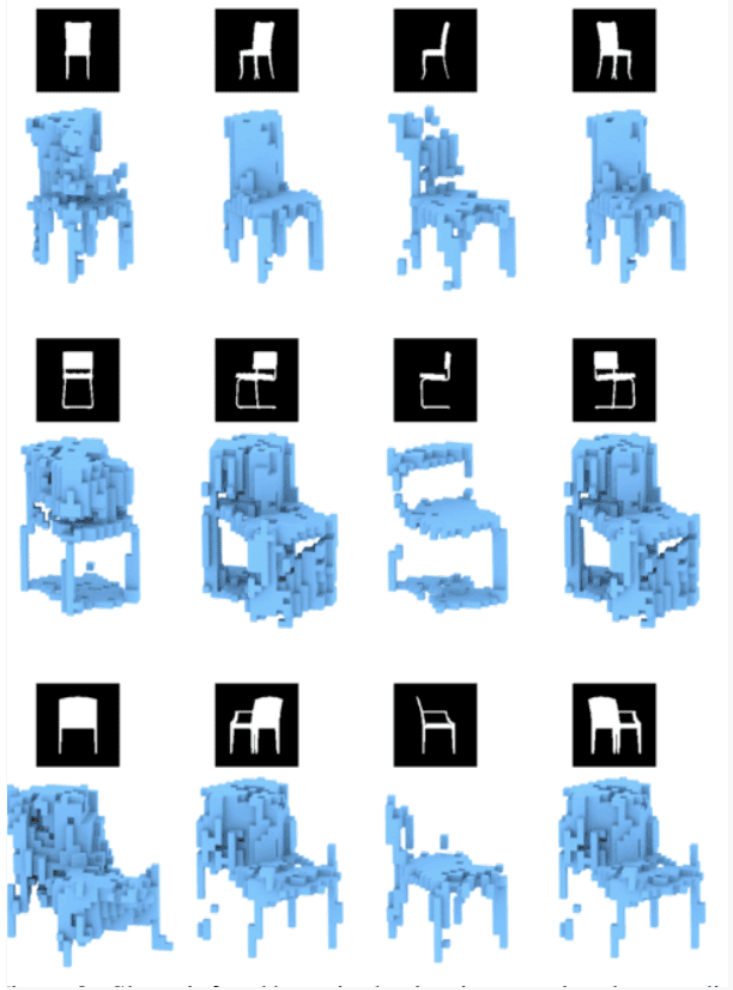


Example of GAN-Generated Images With Super Resolution. Taken from Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, 2016.



Example of High-Resolution GAN-Generated Photographs of Buildings.Taken from Analyzing Perception-Distortion Tradeoff using Enhanced Perceptual Super-resolution Network, 2018.

Aplicaciones GAN: Video Prediction & 3D Object Generation



Example of Three-Dimensional Reconstructions of a Chair From Two-Dimensional Images.Taken from 3D Shape Induction from 2D Views of Multiple Objects, 2016.

Example of Video Frames Generated With a GAN.Taken from Generating Videos with Scene Dynamics, 2016.

02

Conditional GAN

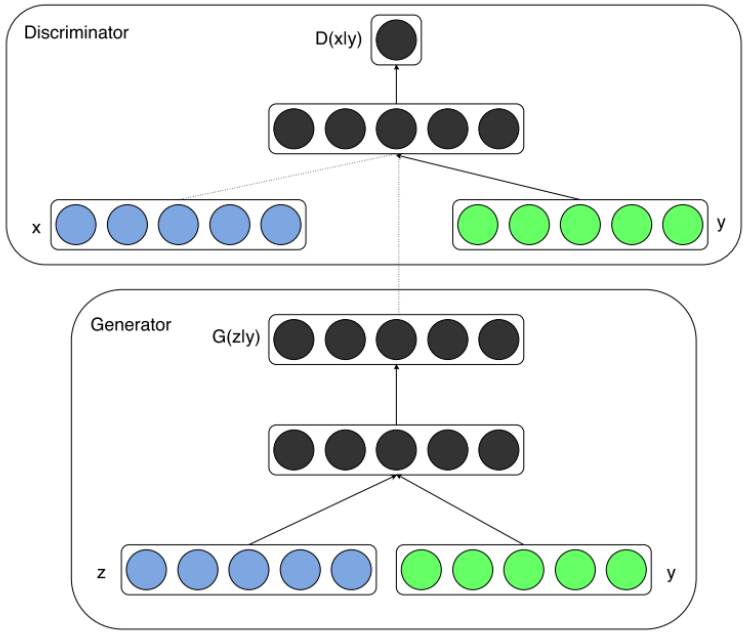
Generative Adversarial Networks para la síntesis de imagen

Introducción

- Propuestas en **2014** por **Mirza et al.** las conditional GAN (**cGAN**) abordan la **problemática** del **escaso control** del **contenido** de la **versión vanilla**.
- Su objetivo es el de **dotar de estructura** al **espacio latente** de entrada al **generador**
- **Introducir información externa** (adicionalmente a las *inputs* que requiere la GAN básica) que "**guía**" el proceso de generación de imágenes:
  - Etiquetas de clase
  - Descripciones de texto
  - Mapas semánticos
  - Imágenes condicionales
  - Máscaras de objetos
  - Mapas de atención

Funcionamiento

- La **información externa** que "**guía**" el proceso de generación de imágenes se le proporciona tanto **al generador** como **al discriminador**.
- Las **tareas** del **generador** y **discriminador** siguen siendo **idénticas** a las de la **GAN convencional**.
- Tanto en el generador como en el discriminador, **el espacio latente y** la **condición** se **combinan** mediante una **representación oculta** (i.e. **MLP**).



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

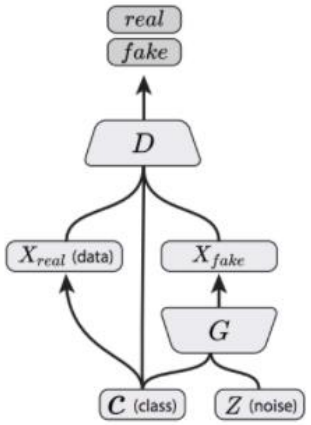
Fig 2 shows some of the generated samples. Each row is conditioned on one label and each column is a different generated sample.

Implementaciones

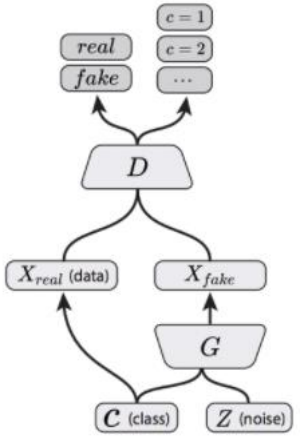
- cGAN: **https://github.com/eriklindernoren/Keras-GAN/tree/master/cgan**
- DCGAN: **https://github.com/eriklindernoren/Keras-GAN/blob/master/dcgan**

Variante: Auxiliary Classifier GAN

- Como hemos visto la **cGAN proporciona** al **generador** tanto un punto en el **espacio latente** como una **etiqueta de clase** con el objetivo de generar una imagen para dicha clase.
- El **discriminador** es **alimentado** por una **imagen** y una **etiqueta de clase** y su objetivo es el de clasificar si dicha imagen es real o falsa.
- La auxiliary classifier GAN (**AC-GAN**) **solo** le **proporciona** al **discriminador** una **imagen de entrada** pero no le dota de información de clase. De hecho es objetivo del discriminador estimarla (adicionalmente a la tarea real vs fake)



Conditional GAN
(Mirza & Osindero, 2014)



AC-GAN
(Present Work)

03

CycleGAN

Generative Adversarial Networks para la síntesis de imagen

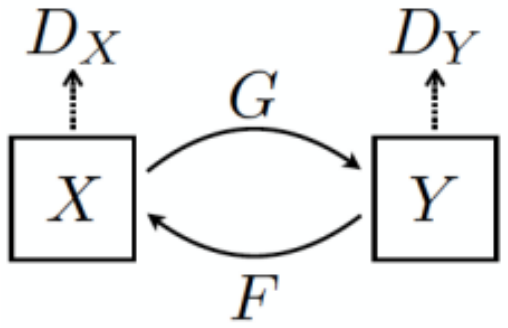
Introducción

- La tarea de ***image-to-image translation*** tiene como objetivo el generar una **imagen sintética** (cierta **modificación**) a partir de una **imagen dada**, e.g. verano a invierno.
- El **entrenamiento** de un modelo para resolver este tipo de tarea requiere de un gran **dataset pareado de ejemplos**. En muchas **ocasiones** resulta **imposible**, e.g. style transfer en pintura.
- Con el objetivo de solventar esta limitación en **2017** nace **CycleGAN** (Jun-Yan Zhu). CycleGAN posibilita la **resolución** de **tareas** de traslación ***image-to-image*** **sin necesidad** de disponer muestras **pareadas** y en sentido bidireccional.
- Los modelos son **entrenados** de manera **no supervisada** a partir de una **gran colección** del **dominito fuente** y el **dominio destino** (sin relación entre ellas).

> *[…] we present a method that can learn to [capture] special characteristics of one image collection and figuring out how these characteristics could be translated into the other image collection, all in the absence of any paired training examples.*

Funcionamiento

- Una **CycleGAN** requiere de un **entrenamiento simultáneo** de dos modelos **generadores** (G y F) y **dos** modelos **discriminadores** (D_X, D_Y).
- $\boldsymbol{G}$ transforma las imágenes **del dominio $\boldsymbol{X}$ al $\boldsymbol{Y}$** mientras que $\boldsymbol{F}$ transforma imágenes **del dominio $\boldsymbol{Y}$ al $\boldsymbol{X}$**.
- $\boldsymbol{D_Y}$ trata de **discernir** si la imagen de entrada es una **imagen real o ficticia** del **dominio $\boldsymbol{Y}$**. De forma análoga trabaja $\boldsymbol{D_X}$ en el **dominio $\boldsymbol{X}$**.



Entrenamiento generadores

- El proceso de entrenamiento de una CycleGAN viene definido por tres términos claramente diferenciados:

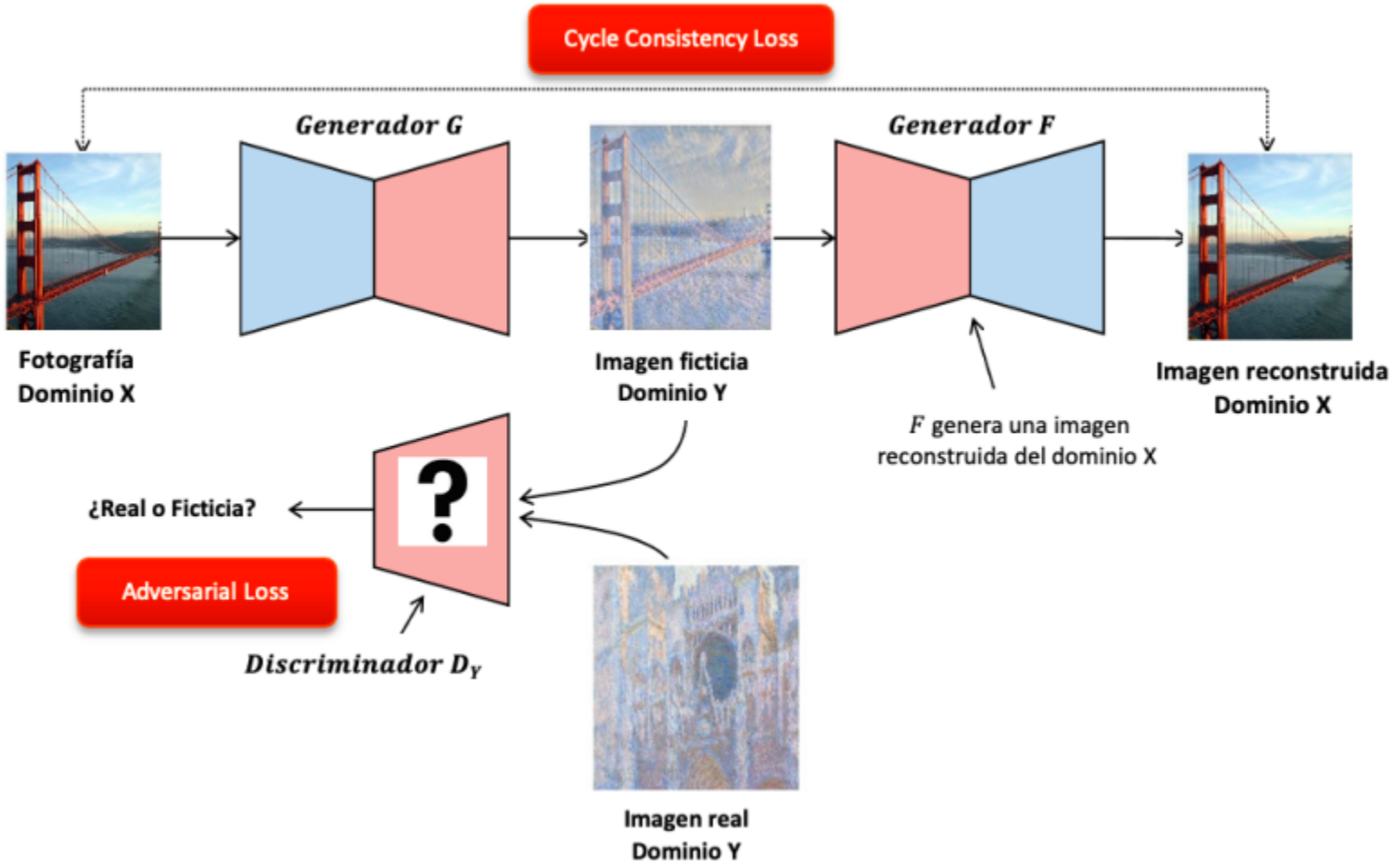
1. Término adversarial ($\mathcal{L}_{adv}$). Mide la **capacidad** de los **generadores** de **crear imágenes** que se asemejan a los **dominios opuestos**. Se corresponde al término de ***binary cross-entropy*** de la versión vanilla.

2. Pérdida de los generadores o ***cycle consistency loss*** ($\mathcal{L}_{cc}$). Surge del concepto de ciclo. Una **imagen** del **dominio** X pasa **por** G creando una imagen del **dominio** Y que le **entra a** F devolviendo dicha imagen al **dominio** X. Esta última imagen reconstruida debe ser muy similar a la imagen de partida. El grado de disimilitud (*cycle consistency loss*) es el criterio a minimizar, se suele emplear MAE o L1.

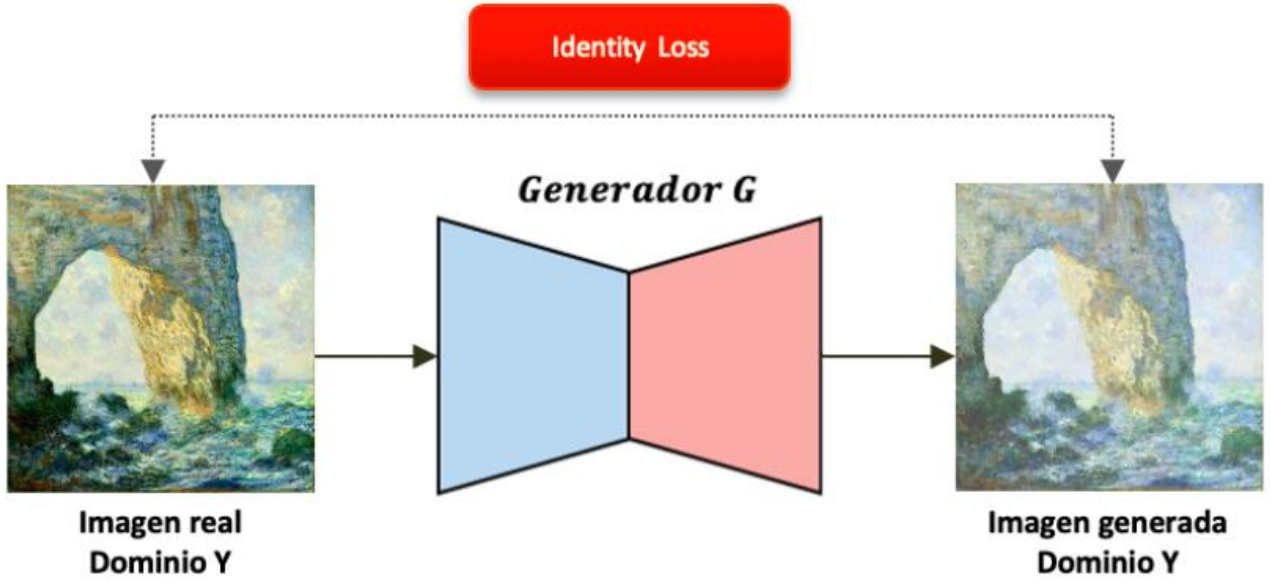
3. Pérdida de identidad o *identity loss* ($\mathcal{L}_{id}$). Si **a** G le **entra** una **imagen** del **dominio** Y, G debería dejar **inalterada la imagen** puesto que ya pertenece al dominio Y que es capaz de generar G. De manera **análoga** ocurre con el generador F y el **dominio** X. Cuanto más modifique un generador una imagen de entrada de su dominio mayor será la pérdida registrada (mediante MAE).

$$\mathcal{L}_{G,F} = \alpha\left(\mathcal{L}_{adv}^{G} + \mathcal{L}_{adv}^{F}\right) + \lambda\left(\mathcal{L}_{cc}^{G} + \mathcal{L}_{cc}^{F}\right) + \delta\left(\mathcal{L}_{id}^{G} + \mathcal{L}_{id}^{F}\right)$$

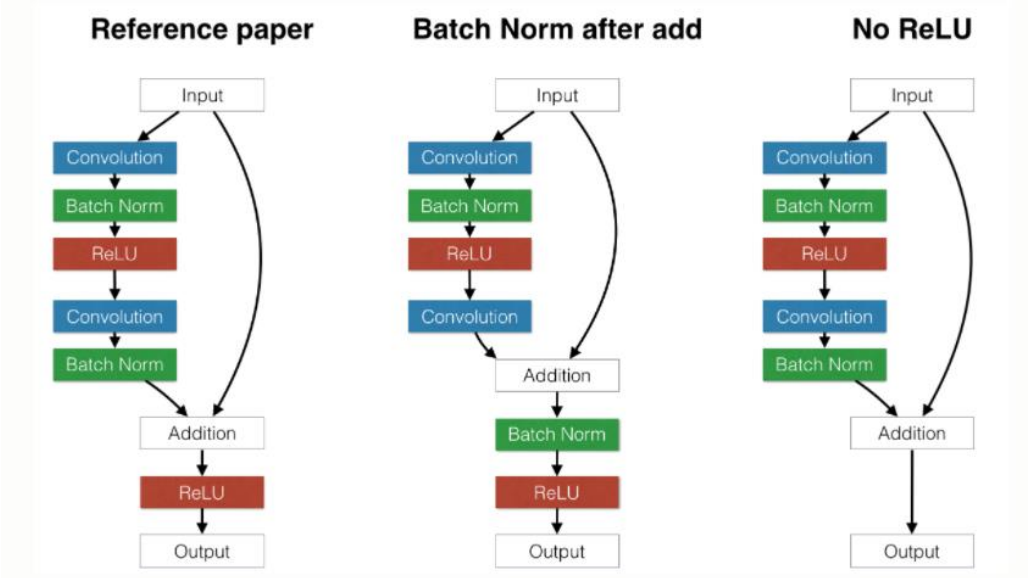
Cycle Consistency Loss



Identity Loss

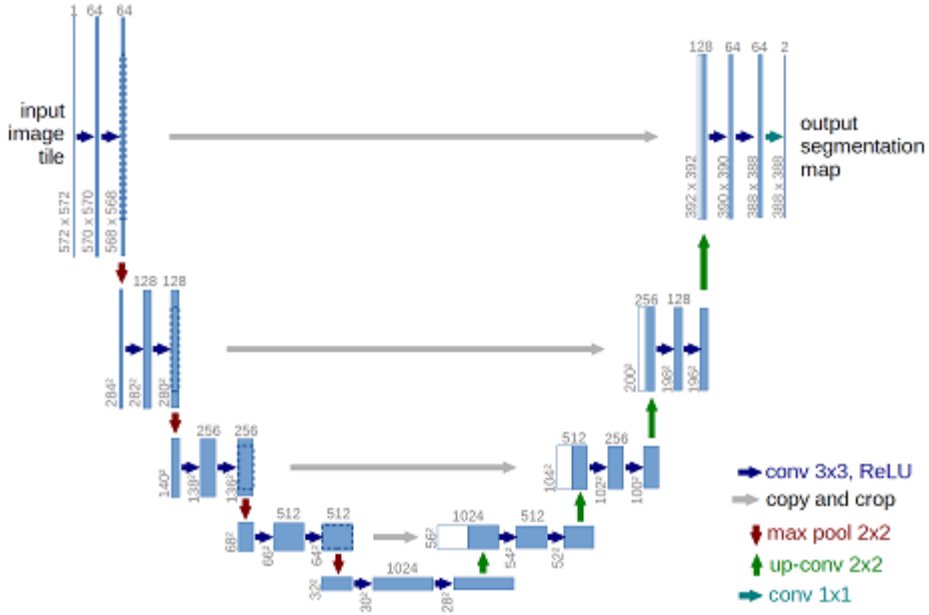


Arquitecturas generadores



https://arxiv.org/pdf/1603.08155.pdf

https://github.com/facebookarchive/fb.resnet.torch

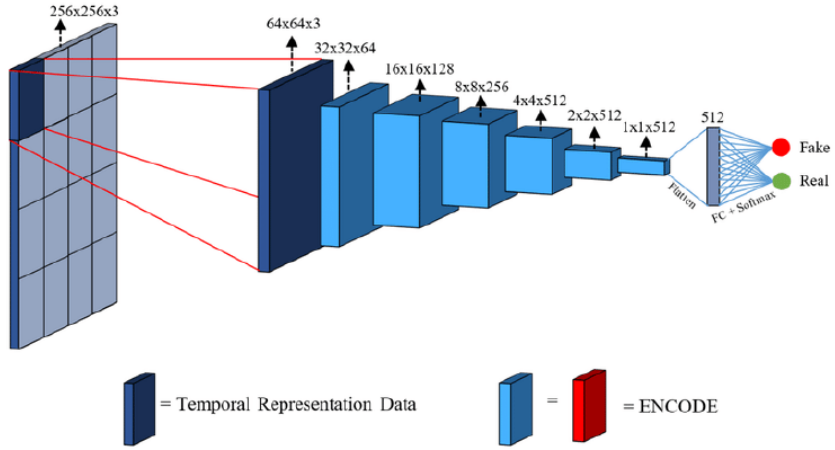


https://github.com/zhixuhao/unet

Entrenamiento discriminadores

- Para **entrenar** D_Y debemos pasarle como **input** a la red discriminadora simultáneamente una **imagen** que realmente pertenezca al **dominio** Y, y una **imagen sintética** creada **por** el **generador** G. De manera **análoga** para entrenar F a partir de imágenes reales y *fake* del **dominio** X.
- La **pérdida** de los **discriminadores** vendrá dada por el cálculo de la **entropía cruzada binaria** obtenida a partir de las **imágenes reales** y **sintéticas**.
- **Discriminador** basado en **PatchGAN**, evalúa (**real/*fake***) el contenido **por parches** y mediante **promedio** emite el **veredicto global** de la **imagen**. Así el discriminador focaliza en el estilo y no en el contenido.

https://arxiv.org/abs/1611.07004v3



Implementaciones

- Keras: **https://github.com/eriklindernoren/Keras-GAN/blob/master/cyclegan/cyclegan.py**
- Pytorch: **https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix**

04

StyleGAN

Generative Adversarial Networks para la síntesis de imagen

Introducción

- **StyleGAN** presentada por **Tero Karras** et al. (NVIDIA) en **2018** focaliza sus esfuerzos en **mejorar** el **generador** de la **vanilla GAN** con el objetivo de sintetizar imágenes de alta calidad.
- Incluyen el uso de una **red de mapeo** del **espacio latente** en uno **intermedio** que **controla** el **estilo** en diferentes puntos del generador. Adicionalmente se introduce una **fuente de ruido** que ejerce de variación sobre cada punto del modelo generador.
- El resultado de estas mejoras se traduce en la generación de **imagen sintética** de **alta calidad** y **fidelidad**. Además, ofrece **control** sobre el **estilo** de las **imágenes** generadas a **distintos nivel** de **detalle** variando los vectores de estilo y ruido.

> *Our generator starts from a learned constant input and adjusts the "style" of the image at each convolution layer based on the latent code, therefore directly controlling the strength of image features at different scales*

Funcionamiento

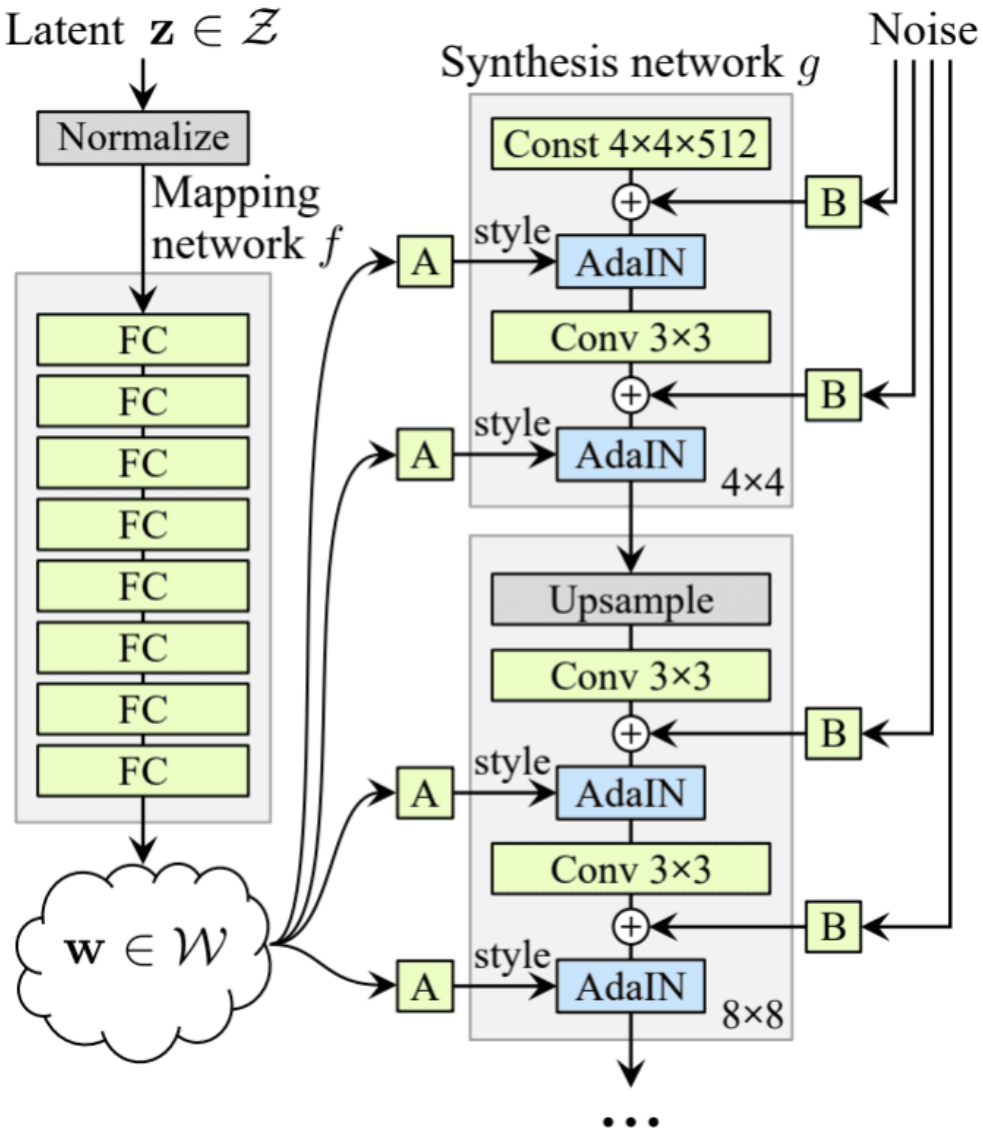
- El generador de **StyleGAN** no toma un punto del espacio latente si no que emplea **dos nuevas fuentes** de **aleatoriedad** para generar una imagen sintética: una **red de mapeo** independiente y **capas de ruido**.
- La salida de la red de mapeo es un **vector** que define el **estilo** a **integrar** en cada punto de la **red generadora** vía una nueva capa denominada Adaptive Instance Normalization (**AdaIN**). Esta nueva capa controla el estilo de las imágenes generadas.
- En cada punto del modelo generador se introduce una **variación estocástica** mediante la adición de **ruido** (sobre los mapas de activación) con el objetivo de dotar al modelo de **alta capacidad** para **interpretar** el **estilo**.
- Esta incorporación de **vectores** de **estilo** y **ruido** permite identificar el estilo y variación estocástica a **nivel local** para un nivel de detalle dado.

Arquitectura generador StyleGAN

- La **red de mapeo** es un **MLP** compuesto por **ocho capas densas**. Tanto el **espacio latente inicial** como el **intermedio** tienen dimensionalidad **512**.
- El **vector** de **estilo** se transforma y se incorpora a **cada bloque** del modelo generador vía **AdaIN**: a) **Estandarización** del mapa de activación de salida a una Gaussiana standard. b) Añadir el ***style vector*** como bias.

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

- Se añade **ruido gaussiano** a cada **mapa de activación** (salida de los bloques convolucionales) **antes de** aplicar **AdaIN**.
- **Distintos muestreos** de **ruido** se generan para cada **bloque**. Este ruido introduce variaciones del estilo para un nivel de detalle dado.



Detalles de entrenamiento e implementación

- A diferencia de la *vanill*a GAN, el **entrenamiento** se lleva a cabo **de manera progresiva** (hasta la convergencia) con **resoluciones** de **imagen** potencia de dos que va **en aumento**. Se van expandiendo generador y discriminador.
- Este entrenamiento progresivo propicia el **cambio** a las ***upsampling layers*** (UpSampling2D + Conv2D) en vez de emplear las capas de convoluciones transpuestas (Conv2DTranspose).
- **Eliminar** el **espacio latente** como **entrada**. En su defecto se establece una **dimensionalidad** de entrada **constante** para cada resolución de imagen y es el espacio latente intermedio el que introduce la información.
- Realmente se generan **dos *style vectors*** de la red de mapeo y **se combinan** en el modelo generador a partir de un ***split point*** que **marca** que **vector** de **estilo** emplea **AdaIN**.

- Keras: **https://github.com/manicman1999/StyleGAN-Keras**
- PyTorch: **https://github.com/tomguluson92/StyleGAN_PyTorch**

Universidad
Internacional
de Valencia

universidadviu.com

De:
Planeta Formación y Universidades