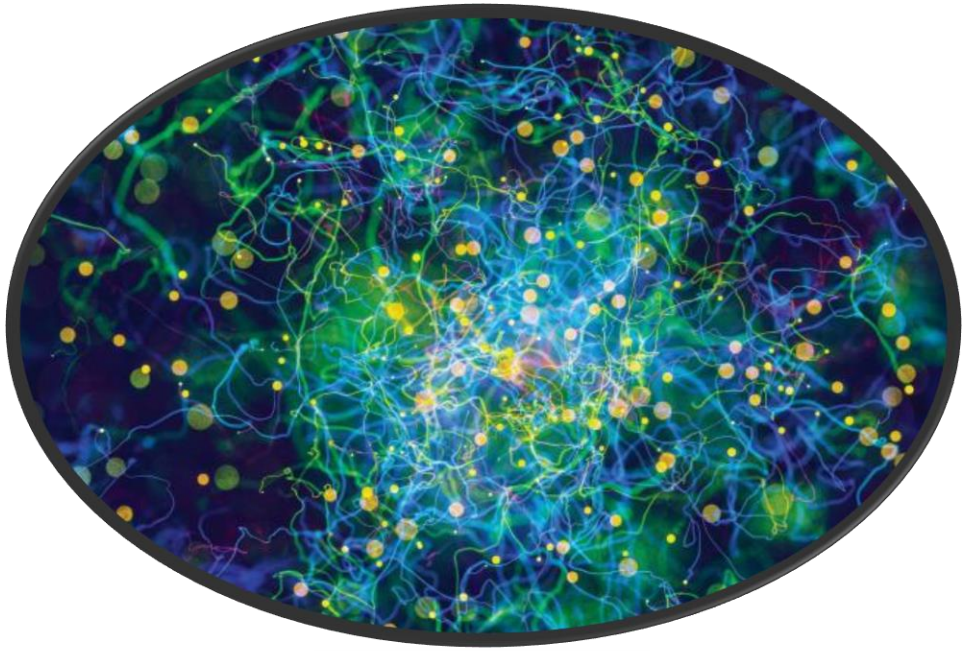
Deep Learning



Día 5

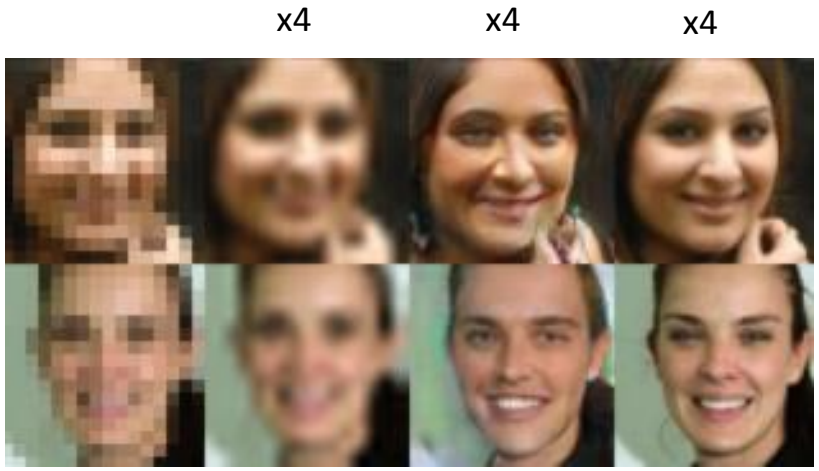
EXPOSITOR: Ing. Giorgio Morales Luna

Junio 2018

5. Generative Adversarial Networks (GANs)

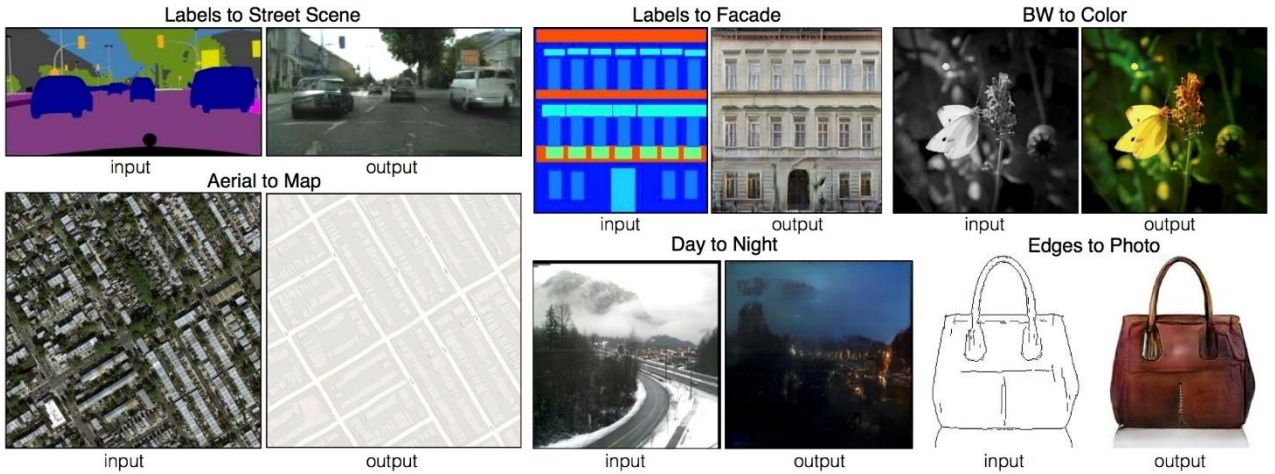
5.1. Aplicaciones

Super-resolution



Domain transfer

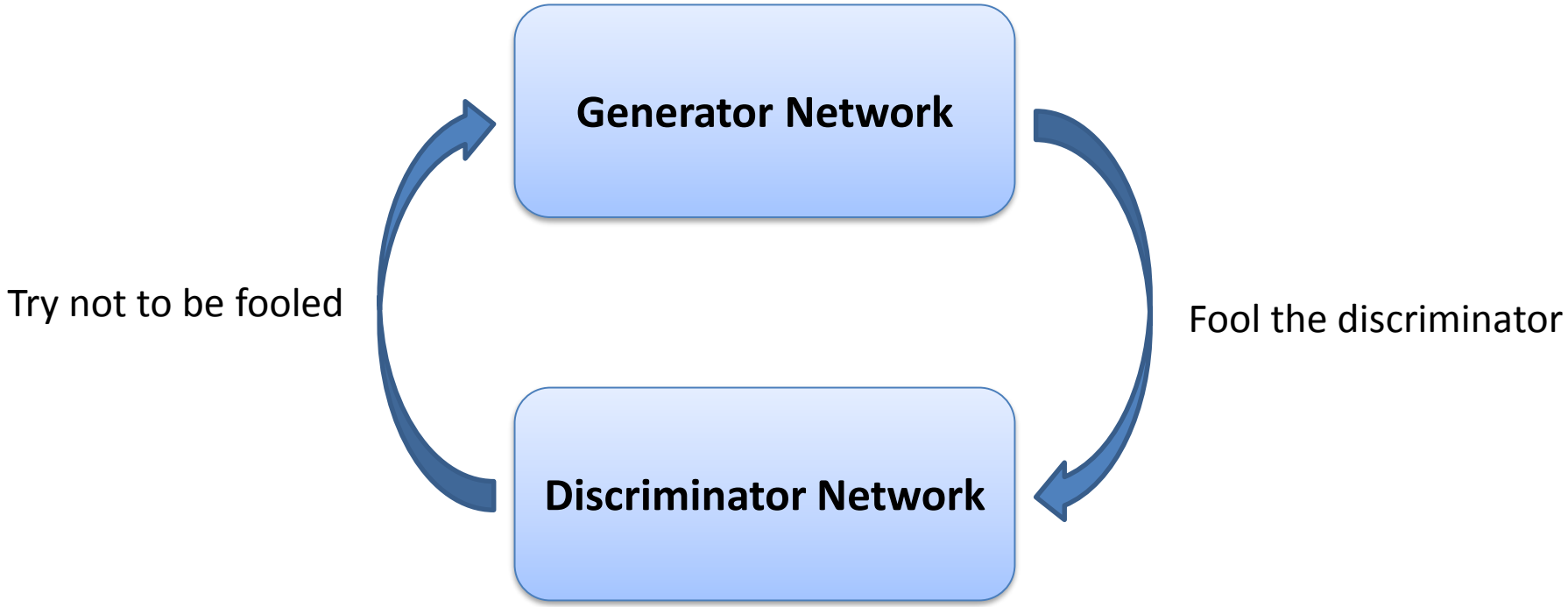
https://affinelayer.com/pixsrv/



"La idea más interesante en los últimos 10 años de Machine Learning" – Yann LeCun

https://github.com/nashory/gans-awesome-applications

5.2. Funcionamiento



5.2. Funcionamiento

Analogía

5.3. Discriminator cost

5.3. Discriminator cost

- Aprendizaje supervisado.

5.3. Discriminator cost

- Aprendizaje supervisado.
- Clasifica imágenes como “reales” o “falsas” (fake).

5.3. Discriminator cost

- Aprendizaje supervisado.
- Clasifica imágenes como “reales” o “falsas” (fake).
- 2 distintas etiquetas -> Clasificación Binaria.

5.3. Discriminator cost

- Aprendizaje supervisado.
- Clasifica imágenes como “reales” o “falsas” (fake).
- 2 distintas etiquetas -> Clasificación Binaria.
- Función de costo: Binary cross-entropy

5.3. Discriminator cost

- Aprendizaje supervisado.
- Clasifica imágenes como “reales” o “falsas” (fake).
- 2 distintas etiquetas -> Clasificación Binaria.
- Función de costo: Binary cross-entropy

$$J = -\left[t \log y + (1 - t) \log(1 - y)\right]$$

5.3. Discriminator cost

Notación del Discriminator

$$J = -\left[t \log y + (1 - t) \log(1 - y)\right]$$

5.3. Discriminator cost

Notación del Discriminator

$$J = -\left[t \log y + (1 - t) \log(1 - y)\right]$$

- t = 1: muestra “real”. t = 0: muestra “fake”.

5.3. Discriminator cost

Notación del Discriminator

$$J = -\left[t\,logy + (1-t)log(1-y)\right]$$

- t = 1: muestra “real”. t = 0: muestra “fake”.
- $y = D(x) = p\,(imagen\ sea\ real \mid imagen)\ \epsilon\ (0,1)$.

5.3. Discriminator cost

Notación del Discriminator

$$J = -\left[t\,logy + (1-t)log(1-y)\right]$$

- t = 1: muestra “real”. t = 0: muestra “fake”.
- $y = D(x) = p\,(imagen\ sea\ real \mid imagen)\ \epsilon\ (0,1)$.
- $y = D(x;\theta_d)$.

5.3. Discriminator cost

Notación del Discriminator

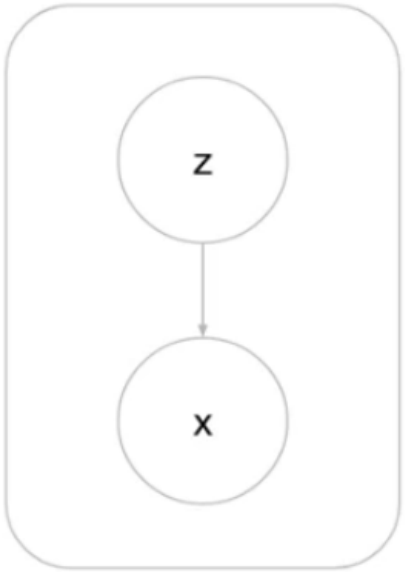
$$J = -\left[t\,logy + (1-t)log(1-y)\right]$$

- t = 1: muestra “real”. t = 0: muestra “fake”.
- $y = D(x) = p\,(imagen\ sea\ real \mid imagen)\ \epsilon\ (0,1)$.
- $y = D(x;\theta_d)$.
- Podemos prescindir de la variable t si consideramos x para las imágenes reales y $\hat{x}$ para las fake.

$$J^{(D)} = -\left[logD(x) + log(1-D(\hat{x}))\right]$$

5.3. Discriminator cost

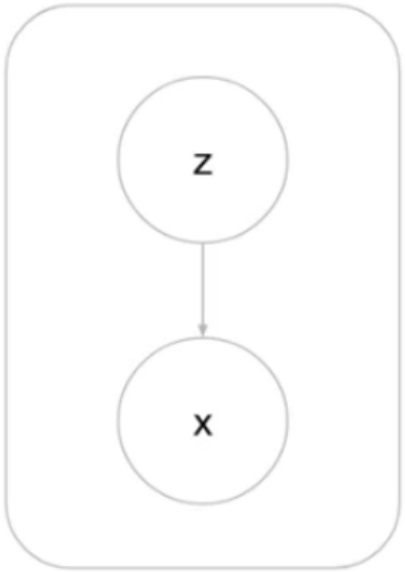
Notación del Generador



- z es una variable de un espacio latente, $z \sim p(z)$
- 2 pasos para samplear:
  1. $z \sim p(z)$
  2. $\hat{x} = G(z)$

5.3. Discriminator cost

Notación del Generador



- z es una variable de un espacio latente, $z \sim p(z)$
- 2 pasos para samplear:
  1. $z \sim p(z)$
  2. $\hat{x} = G(z)$
- Parámetros de G: $G(z; \theta_G)$.

5.3. Discriminator cost

Negative-log-likelihood:

$$J^{(D)} = -\left[logD(x) + log(1 - D(G(z)))\right]$$

Por batches:

$$J^{(D)} = -\left\{\sum_{x} logD(x) + \sum_{z} log(1 - D(G(z)))\right\}$$

5.4. Generator cost

Deben haber dos optimizaciones:

https://arxiv.org/abs/1406.2661

5.4. Generator cost

Deben haber dos optimizaciones:

- El discriminador quiere ser capaz de discriminar entre imágenes reales y fake, así que minimiza el costo $J^{(D)}$.

https://arxiv.org/abs/1406.2661

5.4. Generator cost

Deben haber dos optimizaciones:

- El discriminador quiere ser capaz de discriminar entre imágenes reales y fake, así que minimiza el costo $J^{(D)}$.
- El generador quiere confundir al discriminador: Maximiza el costo del discriminador.

https://arxiv.org/abs/1406.2661

5.4. Generator cost

Deben haber dos optimizaciones:

- El discriminador quiere ser capaz de discriminar entre imágenes reales y fake, así que minimiza el costo $J^{(D)}$.
- El generador quiere confundir al discriminador: Maximiza el costo del discriminador.

$$J^{(G)} = -J^{(D)}$$ Zero-Sum Game

https://arxiv.org/abs/1406.2661

5.4. Generator cost

Deben haber dos optimizaciones:

- El discriminador quiere ser capaz de discriminar entre imágenes reales y fake, así que minimiza el costo $J^{(D)}$.
- El generador quiere confundir al discriminador: Maximiza el costo del discriminador.

$$J^{(G)} = -J^{(D)}$$ Zero-Sum Game

$$\theta_G^* = argmin_{\theta_G} max_{\theta_D} -J^{(D)}$$

https://arxiv.org/abs/1406.2661

5.5. Optimización (Pseudocódigo)

While (no convergencia):

x = cargar un batch de imágenes reales

$\hat{x}$ = samplear un batch de imágenes fake del generador (G)

$\theta_D = \theta_D - learningrate * dJ^{(D)}/d\theta_D$

$\theta_G = \theta_G - learningrate * dJ^{(G)}/d\theta_G$

5.5. Optimización (Pseudocódigo)

While (no convergencia):

x = cargar un batch de imágenes reales

$\hat{x}$ = samplear un batch de imágenes fake del generador (G)

$$\theta_D = \theta_D - learningrate * dJ^{(D)}/d\theta_D$$

$$\theta_G = \theta_G - learningrate * dJ^{(G)}/d\theta_G$$

En la práctica, algunas implementaciones actualizan el generador dos veces por cada actualización del discriminador

5.5. Optimización (Pseudocódigo)

While (no convergencia):

x = cargar un batch de imágenes reales

$\hat{x}$ = samplear un batch de imágenes fake del generador (G)

$$\theta_D = \theta_D - learningrate * dJ^{(D)}/d\theta_D$$

$$\theta_G = \theta_G - learningrate * dJ^{(G)}/d\theta_G$$

$$\theta_G = \theta_G - learningrate * dJ^{(G)}/d\theta_G$$

En la práctica, algunas implementaciones actualizan el generador dos veces por cada actualización del discriminador

5.6. Problemas de Optimización

La función de costo desde la perspectiva del generador, es problemática:

$$J^{(D)} = -\left\{\sum_x \log D(x) + \sum_z \log(1 - D(G(z)))\right\}$$

El gradiente del primer término respecto a θ_D es 0. Sólo el segundo término es relevante

5.6. Problemas de Optimización

5.6. Problemas de Optimización

- Supongamos que el discriminador es muy bueno discriminando imágenes reales de fakes.

5.6. Problemas de Optimización

- Supongamos que el discriminador es muy bueno discriminando imágenes reales de fakes.
- Entonces: $D(G(z))$ es cercano a 0.

5.6. Problemas de Optimización

- Supongamos que el discriminador es muy bueno discriminando imágenes reales de fakes.
- Entonces: $D(G(z))$ es cercano a 0.
- Mientras $D(G(z)) \rightarrow 0$, la pendiente se hace cada vez más pequeña. Bueno para el discriminador, pero no para el generador que cambia en una dirección proporcional al gradiente.

5.6. Problemas de Optimización

- Supongamos que el discriminador es muy bueno discriminando imágenes reales de fakes.
- Entonces: $D(G(z))$ es cercano a 0.
- Mientras $D(G(z)) \rightarrow 0$, la pendiente se hace cada vez más pequeña. Bueno para el discriminador, pero no para el generador que cambia en una dirección proporcional al gradiente.
- En conclusión, cuando el discriminador es muy bueno, el generador mejora su performance cada vez menos.

5.6. Problemas de Optimización

- Supongamos que el discriminador es muy bueno discriminando imágenes reales de fakes.
- Entonces: $D(G(z))$ es cercano a 0.
- Mientras $D(G(z)) \rightarrow 0$, la pendiente se hace cada vez más pequeña. Bueno para el discriminador, pero no para el generador que cambia en una dirección proporcional al gradiente.
- En conclusión, cuando el discriminador es muy bueno, el generador mejora su performance cada vez menos.

Solución

- Usar una función de costo distinta para el generador.
- Invertir el objetivo del generador.

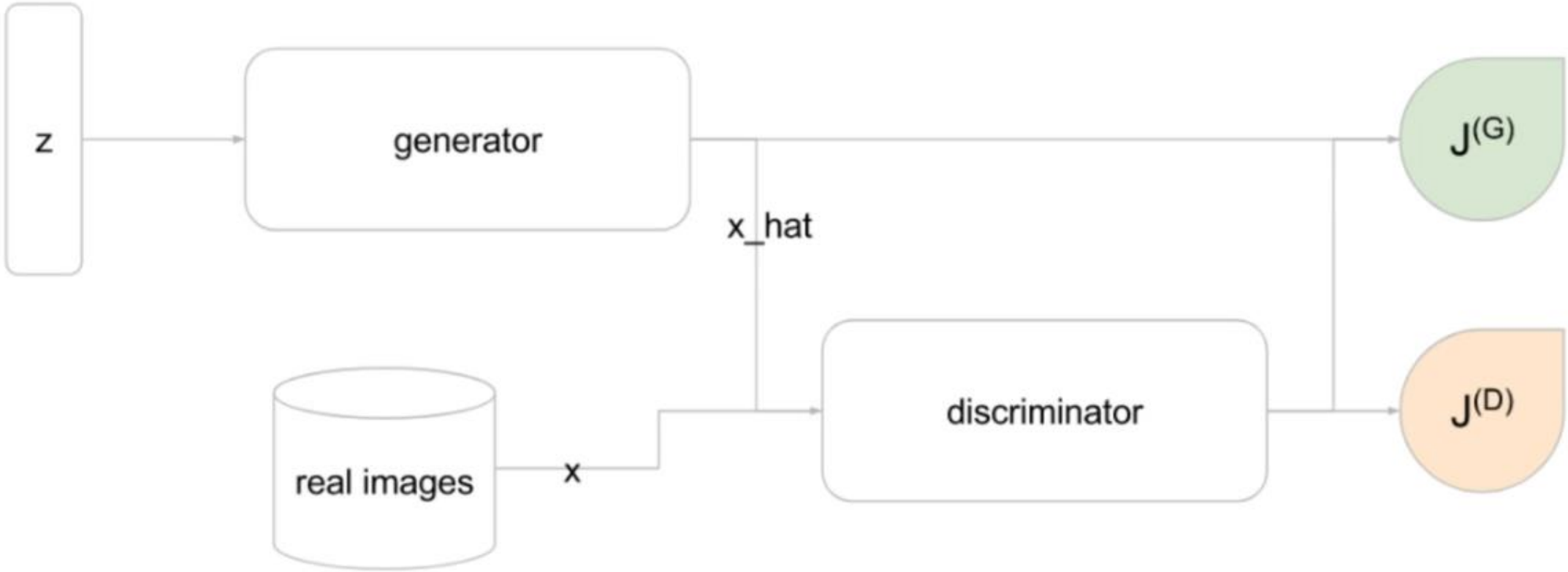
5.6. Problemas de Optimización

- En vez de considerar un target 0 (t = 0) para imágene fakes, el generador considerará que dicho target sea 1 (t = 1).

$$J^{(G)} = -\sum_{z} \{logD(G(z))\}$$

- Tiene sentido porque el término $\log(D(G(z))$ corresponde al costo cuando el target t = 1.

5.7. Diagrama de flujo



5.8. DCGAN

5.8. DCGAN

- Hasta ahora sabemos que tenemos 2 redes neuronales, pero no sabemos qué estructura deben tener.

5.8. DCGAN

- Hasta ahora sabemos que tenemos 2 redes neuronales, pero no sabemos qué estructura deben tener.

- El 2015 se diseñó un tipo eficiente de GAN llamada DCGAN (Deep Convolutional GAN). Muchos modelos actuales están basados en este diseño

5.8. DCGAN

- Hasta ahora sabemos que tenemos 2 redes neuronales, pero no sabemos qué estructura deben tener.

- El 2015 se diseñó un tipo eficiente de GAN llamada DCGAN (Deep Convolutional GAN). Muchos modelos actuales están basados en este diseño

  https://arxiv.org/abs/1511.06434

5.8. DCGAN

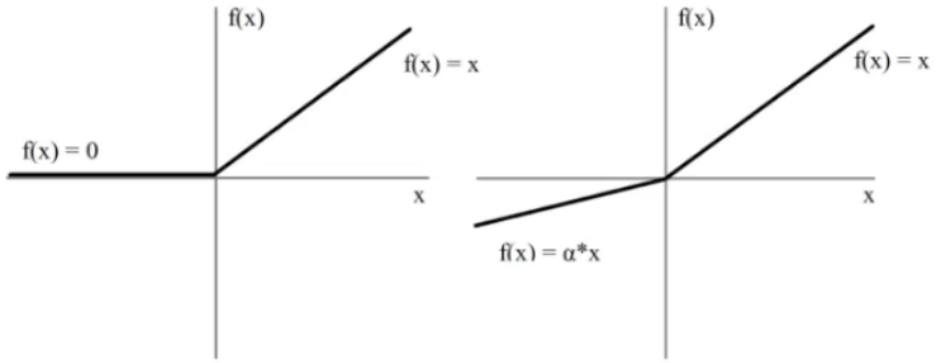
- Hasta ahora sabemos que tenemos 2 redes neuronales, pero no sabemos qué estructura deben tener.

- El 2015 se diseñó un tipo eficiente de GAN llamada DCGAN (Deep Convolutional GAN). Muchos modelos actuales están basados en este diseño

  https://arxiv.org/abs/1511.06434

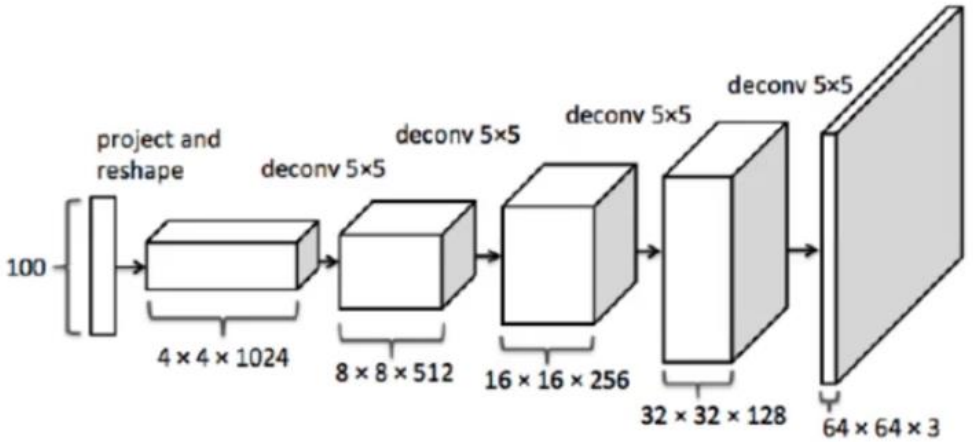
- Características:

  ✓ Batch-normalization.
  ✓ All-convolutional network (no pooling).
  ✓ Optimizador Adam.
  ✓ Leaky-ReLU.

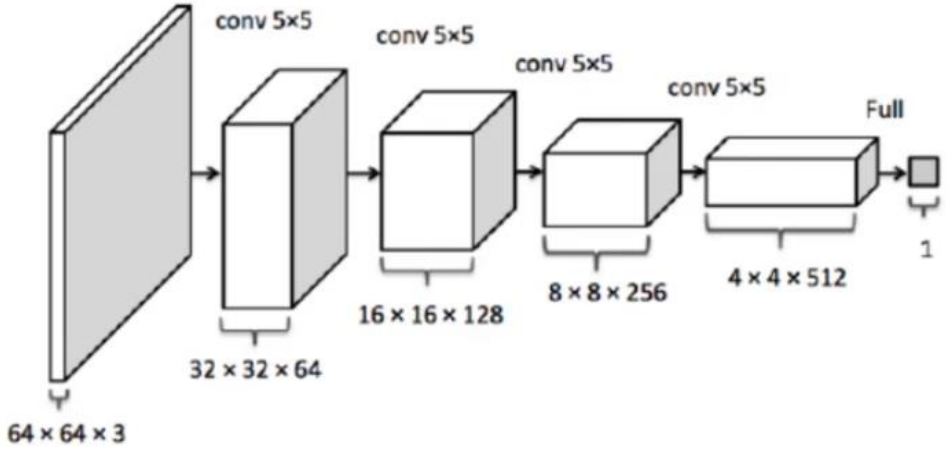


5.8. DCGAN

Generador



Discriminador



5.9. Image-to-image translation with Conditional Adversarial Networks

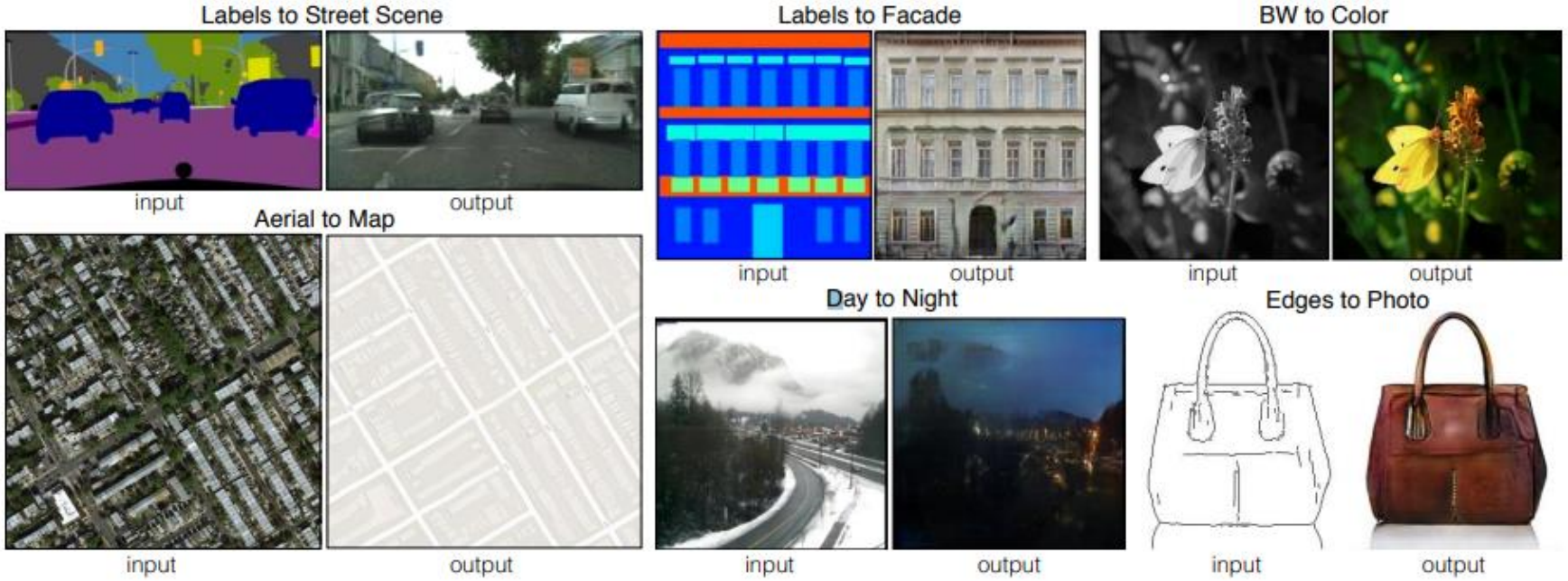


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

https://arxiv.org/abs/1611.07004

5.9. Image-to-image translation with Conditional Adversarial Networks

GAN convencional

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z))].$$

GAN condicional

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],$$

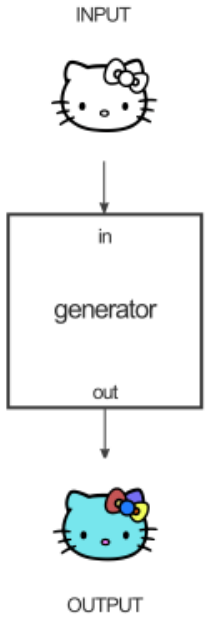
5.9. Image-to-image translation with Conditional Adversarial Networks

Ejemplo de dataset. Input:
Imagen en blanco y negro.
Target: Versión colorizada

5.9. Image-to-image translation with Conditional Adversarial Networks

Ejemplo de dataset. Input: Imagen en blanco y negro. Target: Versión colorizada

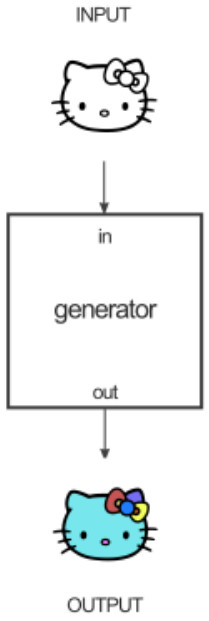
El generador en este caso trata de aprender cómo colorear una imagen en blanco y negro:



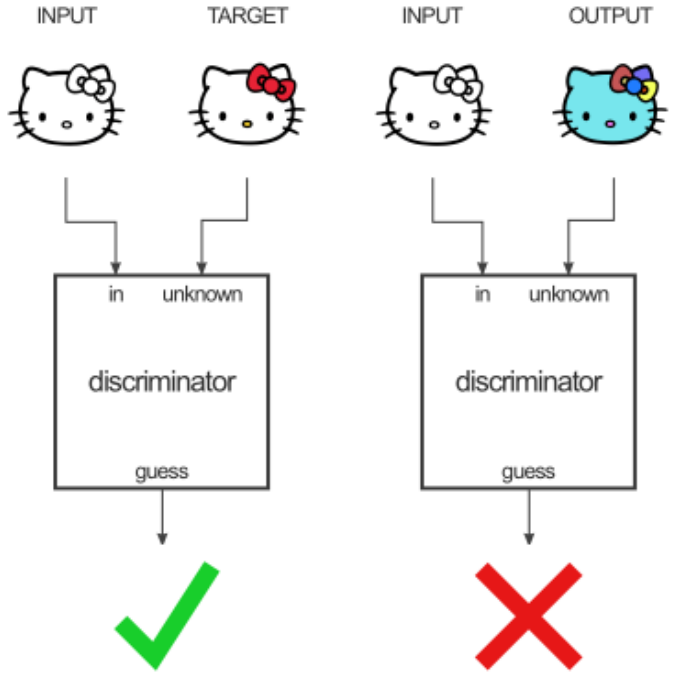
5.9. Image-to-image translation with Conditional Adversarial Networks

Ejemplo de dataset. Input: Imagen en blanco y negro. Target: Versión colorizada

El generador en este caso trata de aprender cómo colorear una imagen en blanco y negro:



El discriminador observa los intentos de colorización del generador y trata de aprender a distinguir la diferencia entre las colorizaciones generadas y la imagen coloreada verdadera proporcionada en el dataset.



5.9. Image-to-image translation with Conditional Adversarial Networks

Generator architecture

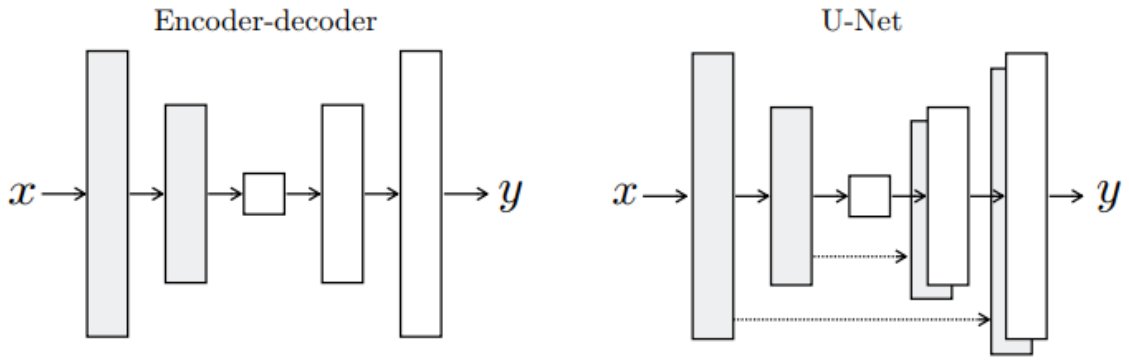
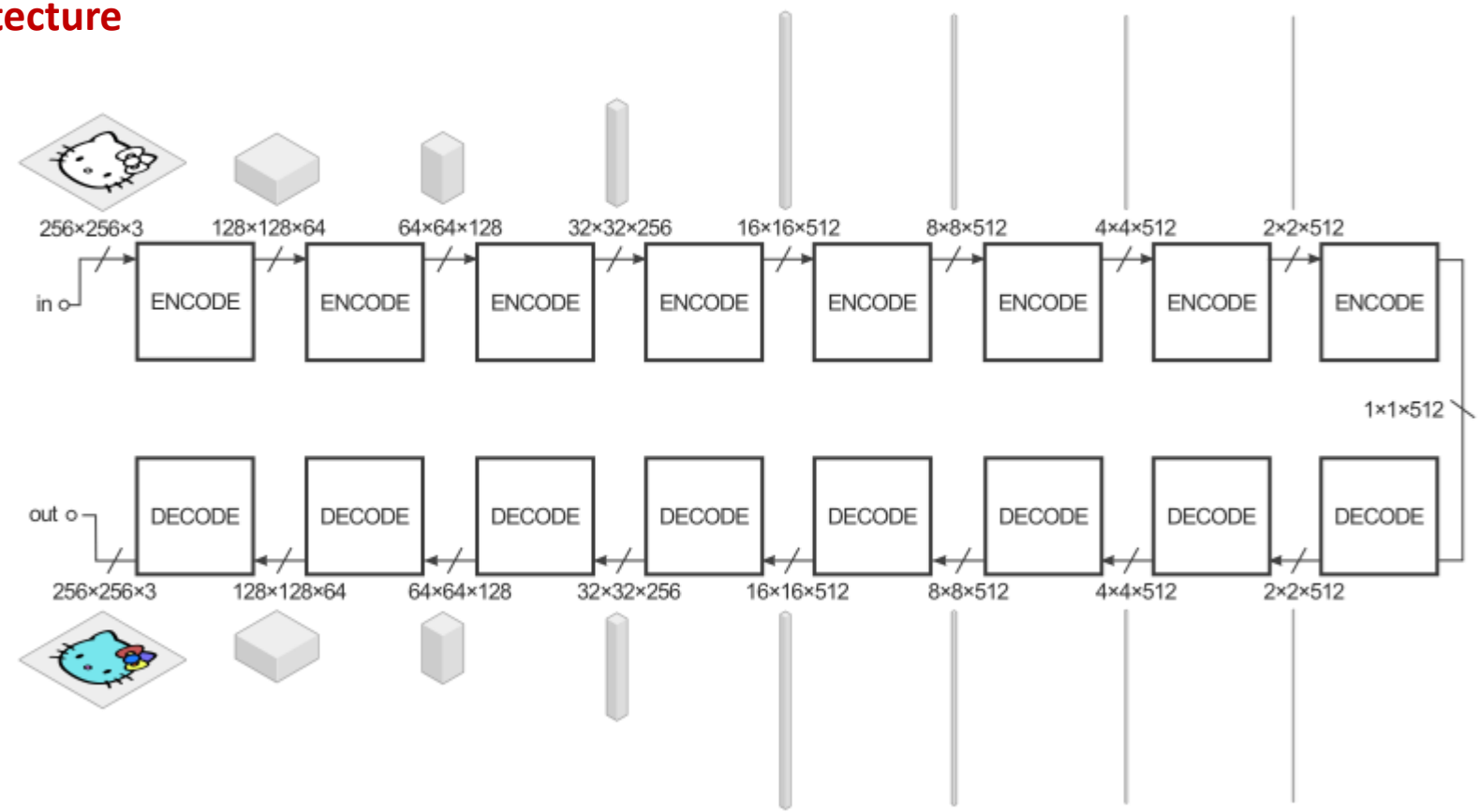


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

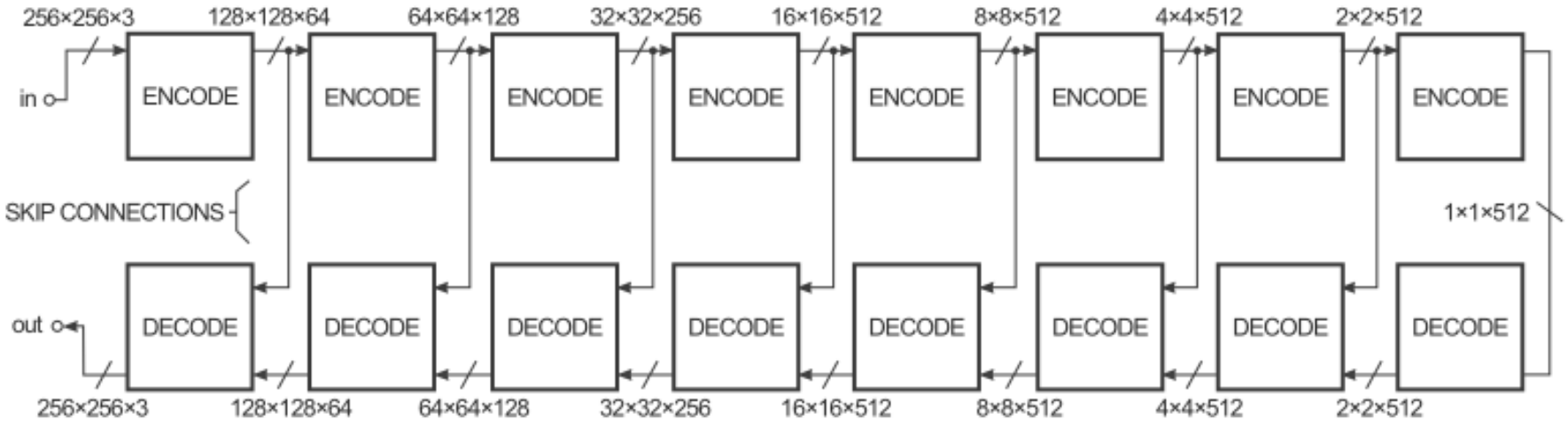
5.9. Image-to-image translation with Conditional Adversarial Networks

Generator architecture



5.9. Image-to-image translation with Conditional Adversarial Networks

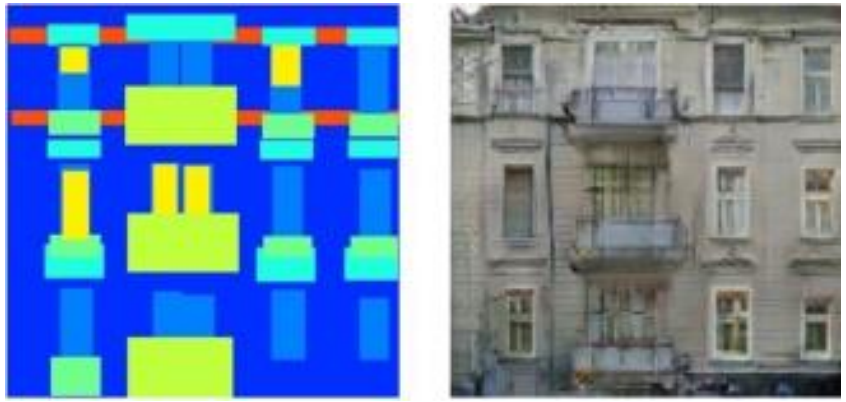
Generator architecture



5.9. Image-to-image translation with Conditional Adversarial Networks

Markovian Discriminator (PatchGAN)

Para comparar con mayor detalle las imágenes originales y las producidas por el generador, es mejor hacerlo localmente; es decir a la escala de “patches”.



5.9. Image-to-image translation with Conditional Adversarial Networks

Markovian Discriminator (PatchGAN)

Para comparar con mayor detalle las imágenes originales y las producidas por el generador, es mejor hacerlo localmente; es decir a la escala de “patches”.

5.9. Image-to-image translation with Conditional Adversarial Networks

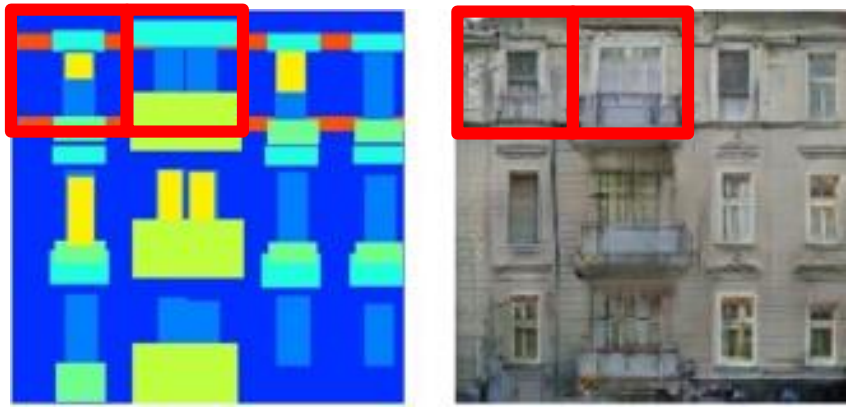
Markovian Discriminator (PatchGAN)

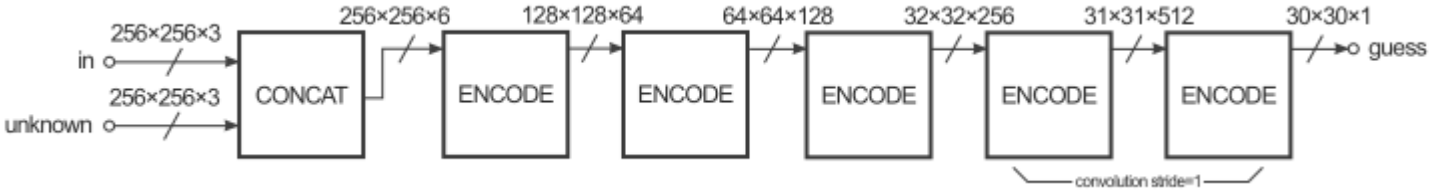
Para comparar con mayor detalle las imágenes originales y las producidas por el generador, es mejor hacerlo localmente; es decir a la escala de “patches”.

5.9. Image-to-image translation with Conditional Adversarial Networks

Markovian Discriminator (PatchGAN)

Para comparar con mayor detalle las imágenes originales y las producidas por el generador, es mejor hacerlo localmente; es decir a la escala de “patches”.

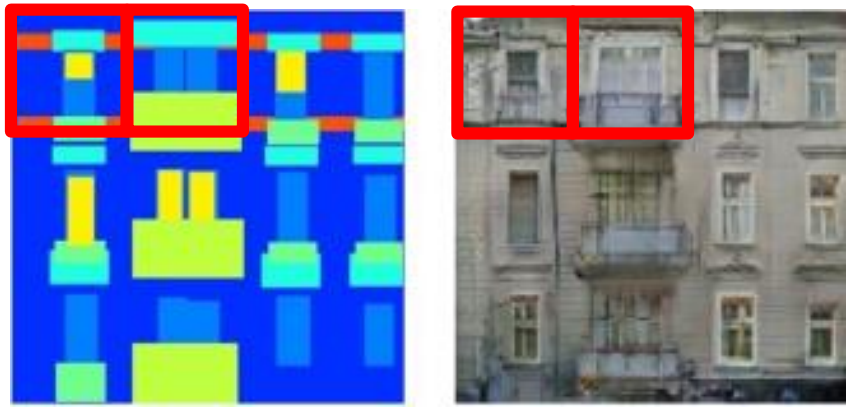


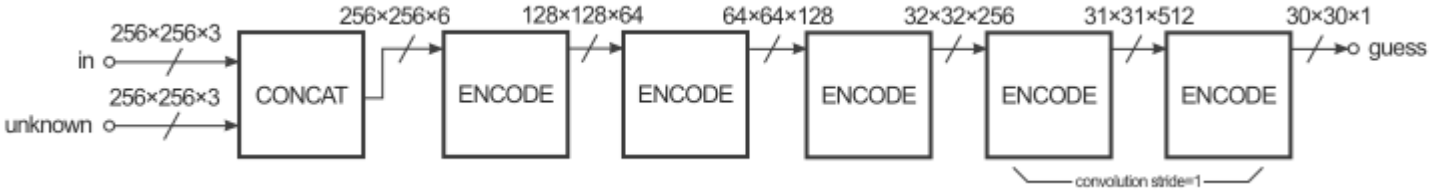


5.9. Image-to-image translation with Conditional Adversarial Networks

Markovian Discriminator (PatchGAN)

Para comparar con mayor detalle las imágenes originales y las producidas por el generador, es mejor hacerlo localmente; es decir a la escala de “patches”.

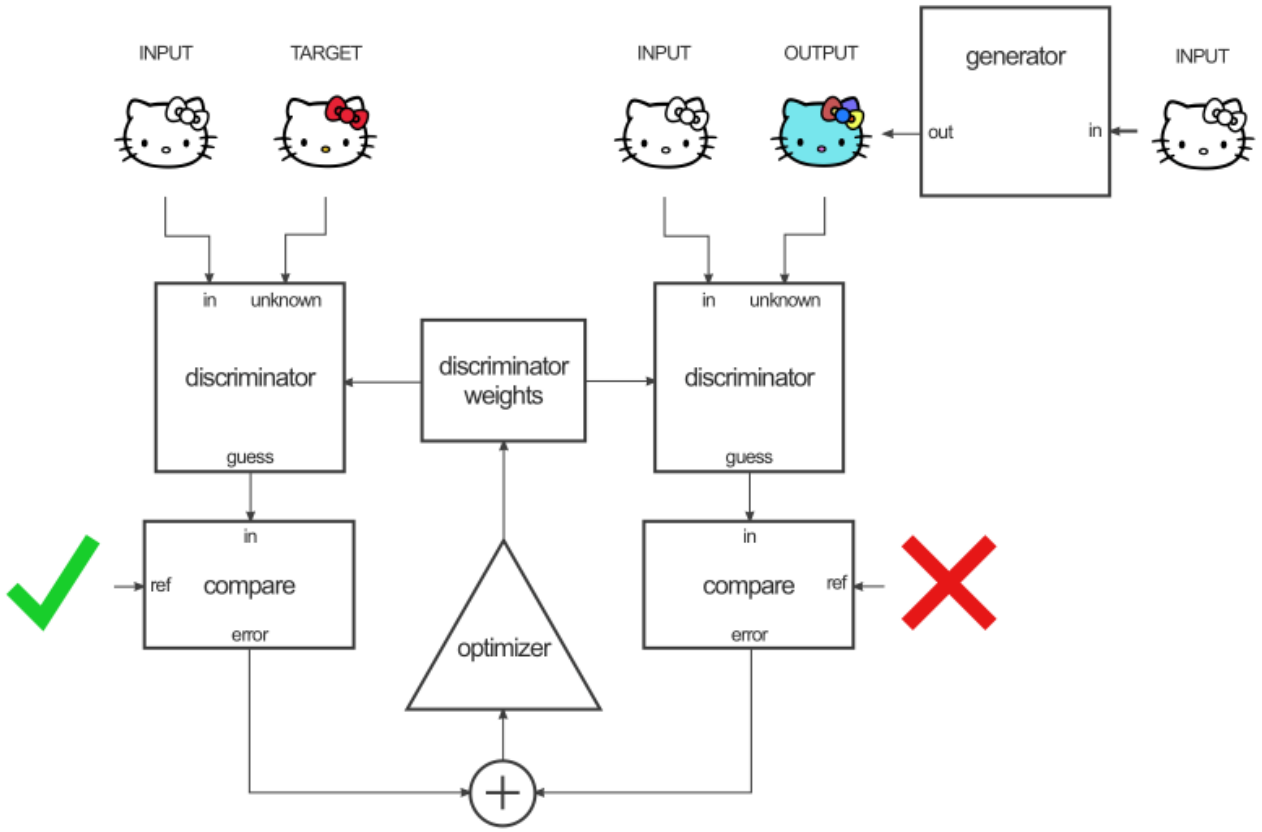




https://fomoro.com/research/article/receptive-field-calculator

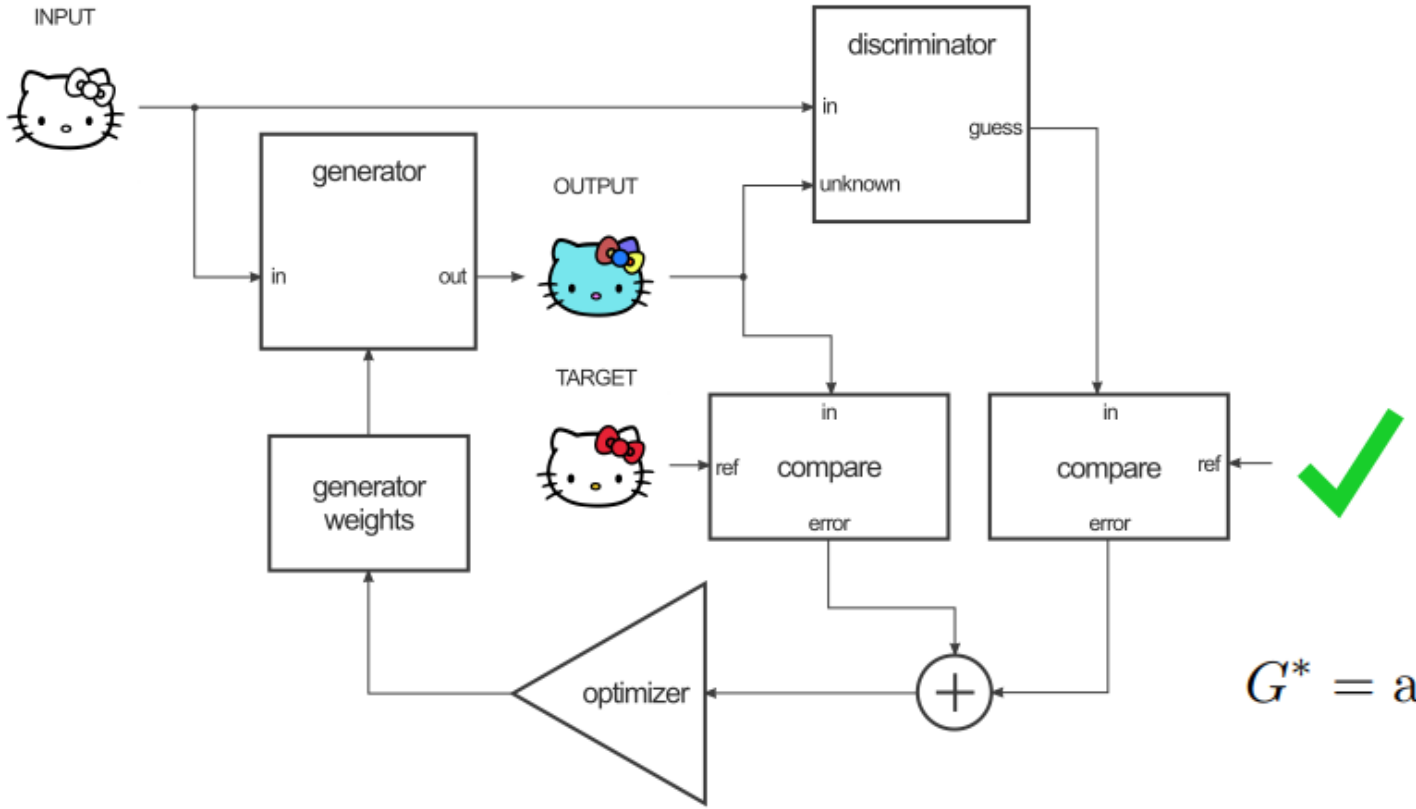
5.9. Image-to-image translation with Conditional Adversarial Networks

Entrenamiento del Discriminador



5.9. Image-to-image translation with Conditional Adversarial Networks

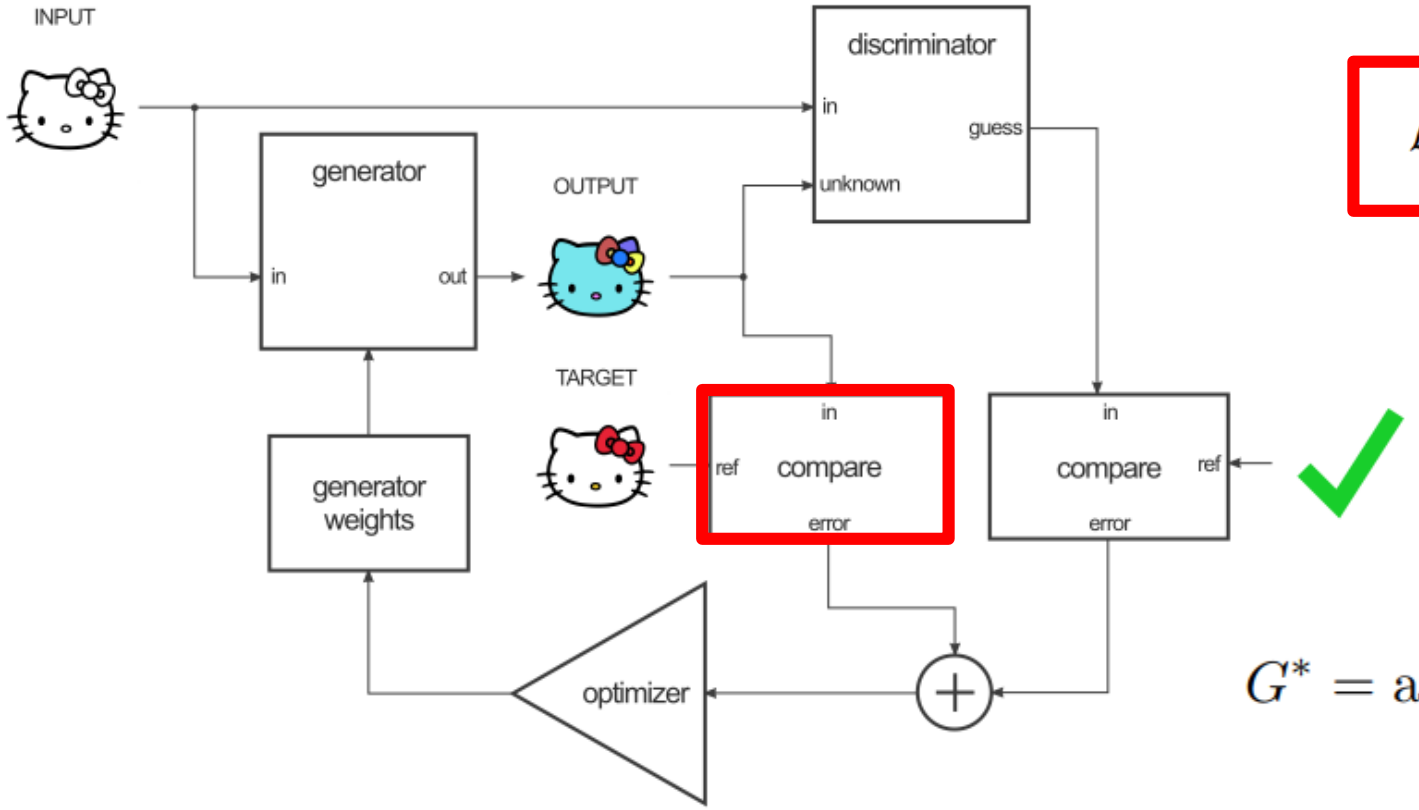
Entrenamiento del Generador



$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

5.9. Image-to-image translation with Conditional Adversarial Networks

Entrenamiento del Generador

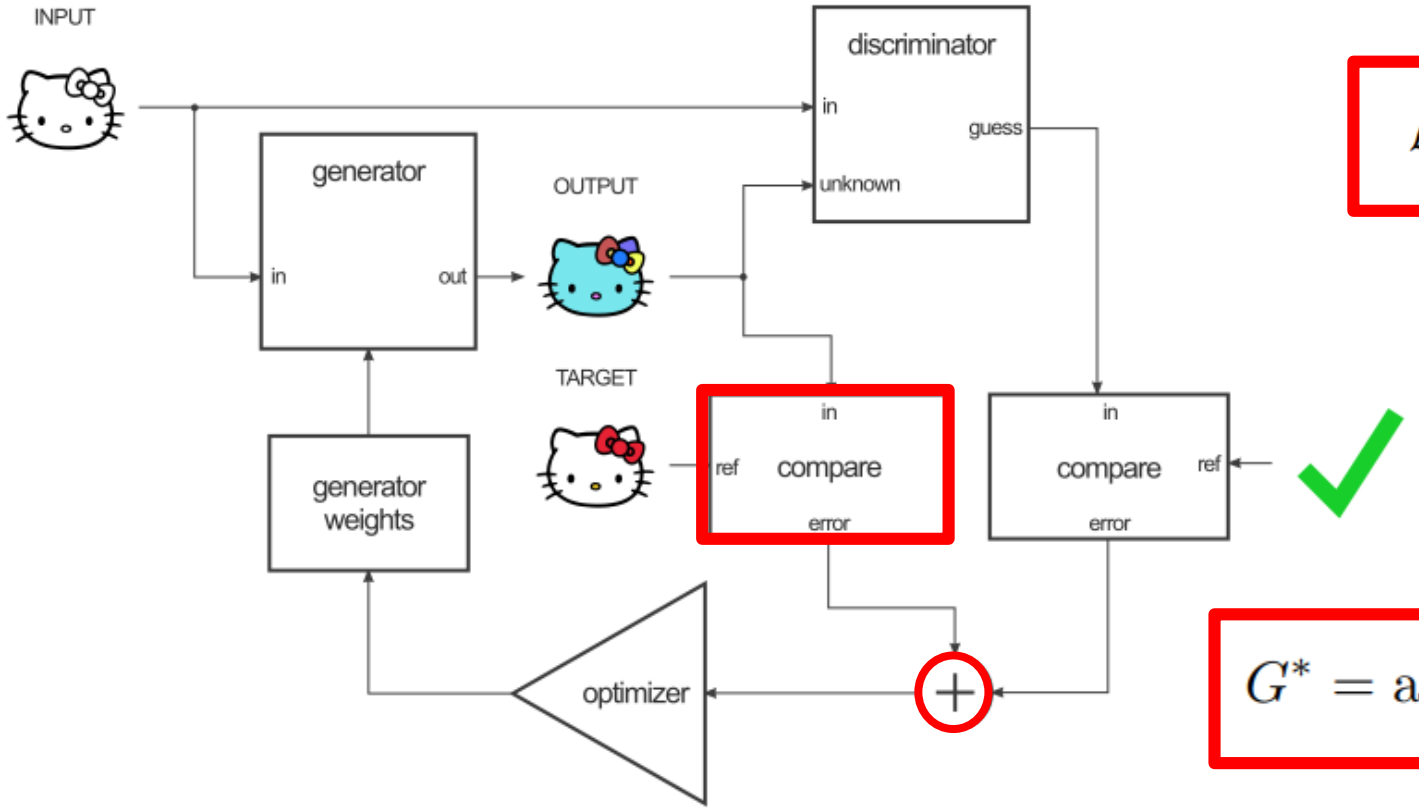


$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

5.9. Image-to-image translation with Conditional Adversarial Networks

Entrenamiento del Generador

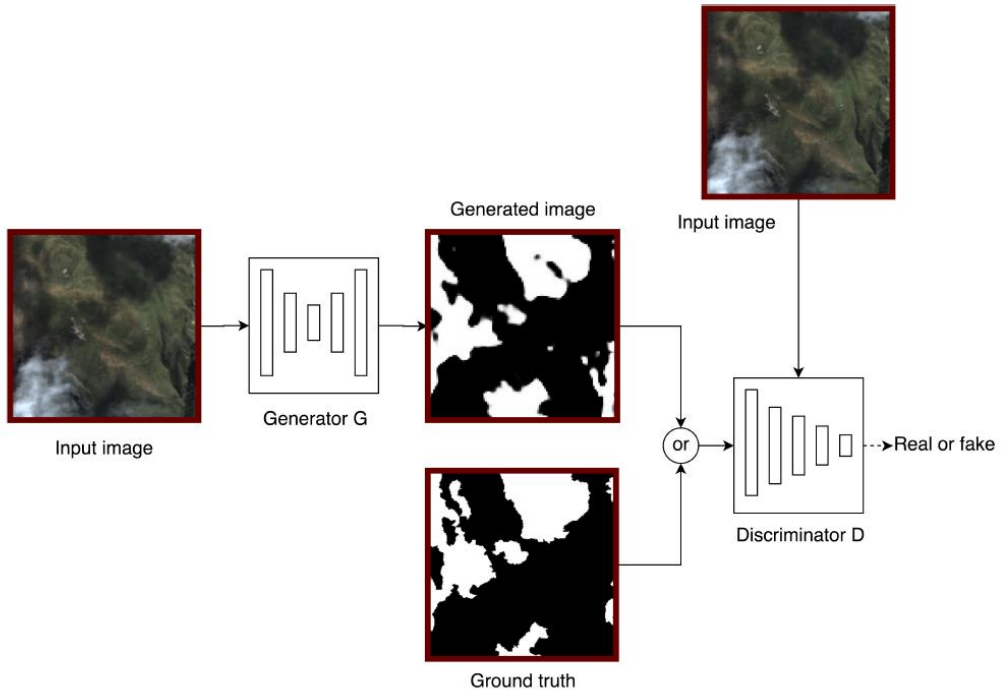


$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

5.10. Shadow segmentation using cGANs

Esquema de Entrenamiento



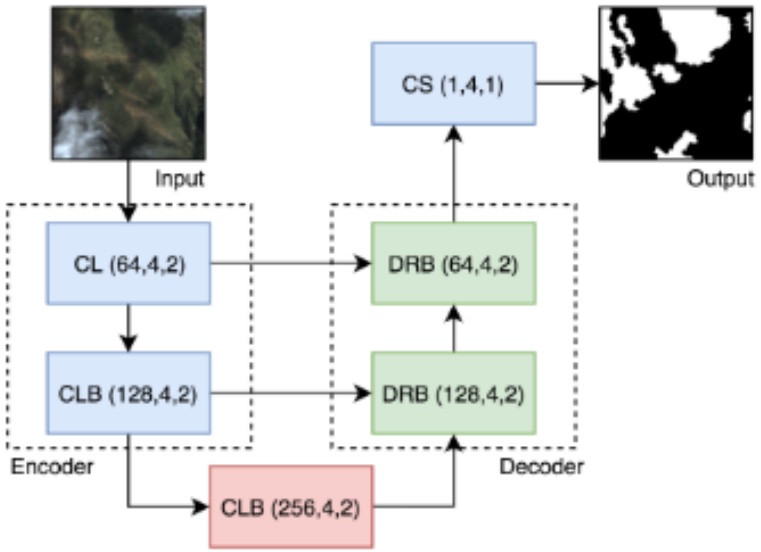
Dataset

http://didt.inictel-uni.edu.pe/dataset/datasetshadow.hdf5

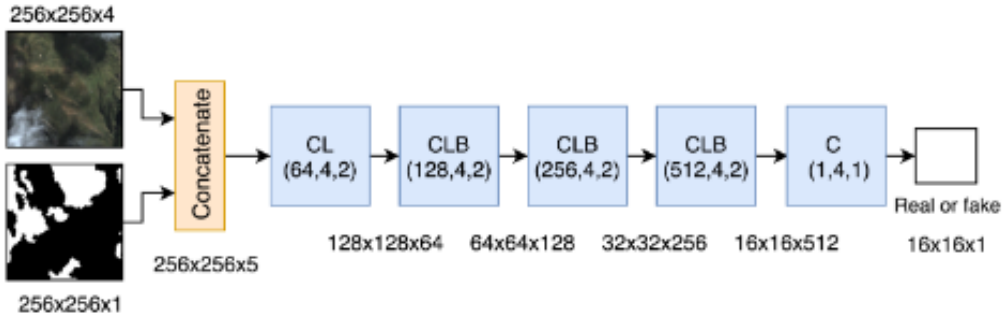
https://www.researchgate.net/publication/328926887_Shadow_Detection_in_High-Resolution_Multispectral_Satellite_Imagery_Using_Generative_Adversarial_Networks

5.10. Shadow segmentation using cGANs

Generador

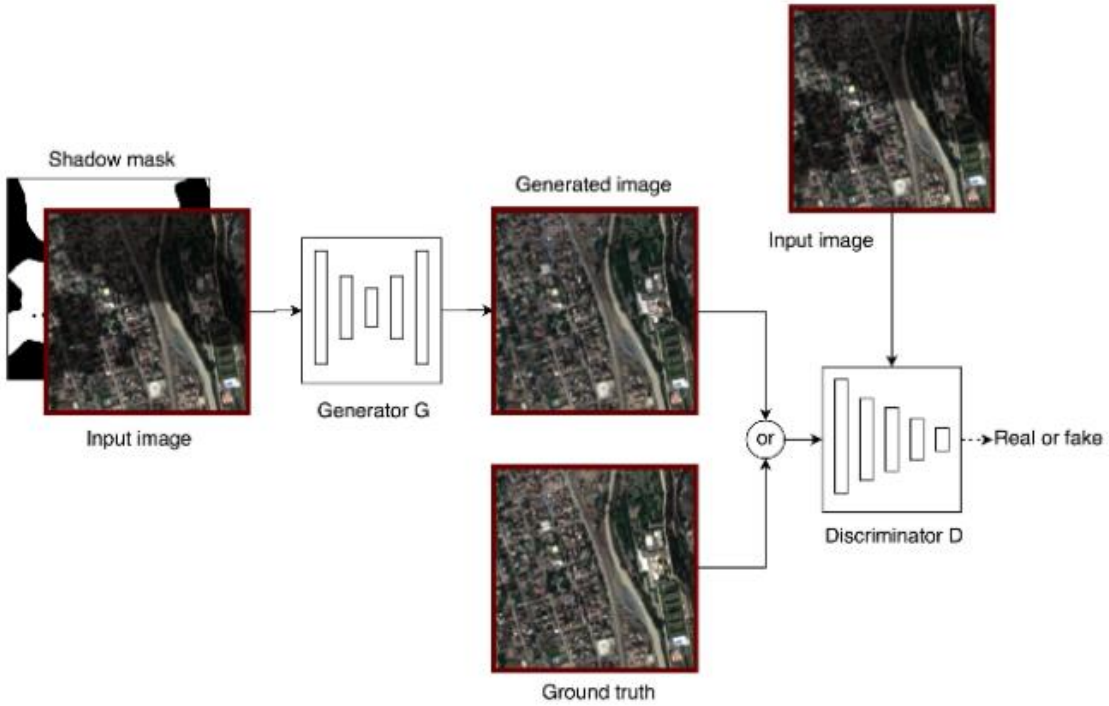


Discriminador



5.10. Shadow segmentation using cGANs

Esquema de Entrenamiento



Dataset

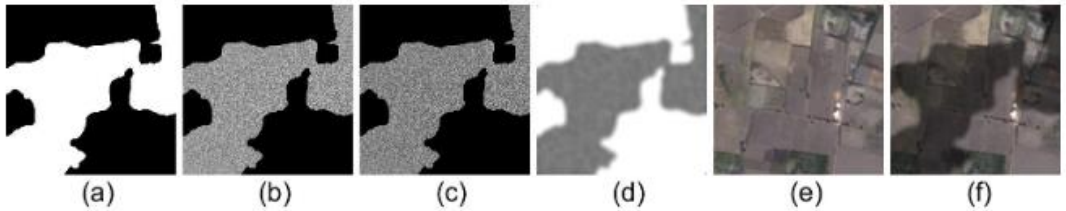


Fig. 2. Shadow generation. (a) Binarized Perlin noise. (b) $P2$ mask. (c) $P3_1$ mask. (d) Filtered $P4_1$ mask. (e) Original image. (f)Altered image with artificial shadows.

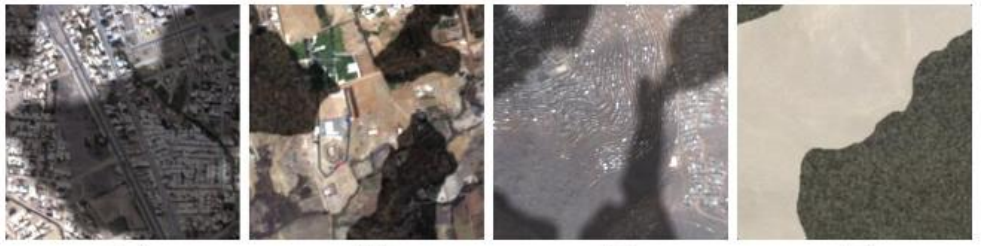


Fig. 3. Sample shadowed images from the ShadowfreePeru dataset. (a)(c) Smooth shadows with blurred edges. (b)(d) Dark shadows without blurred edges.

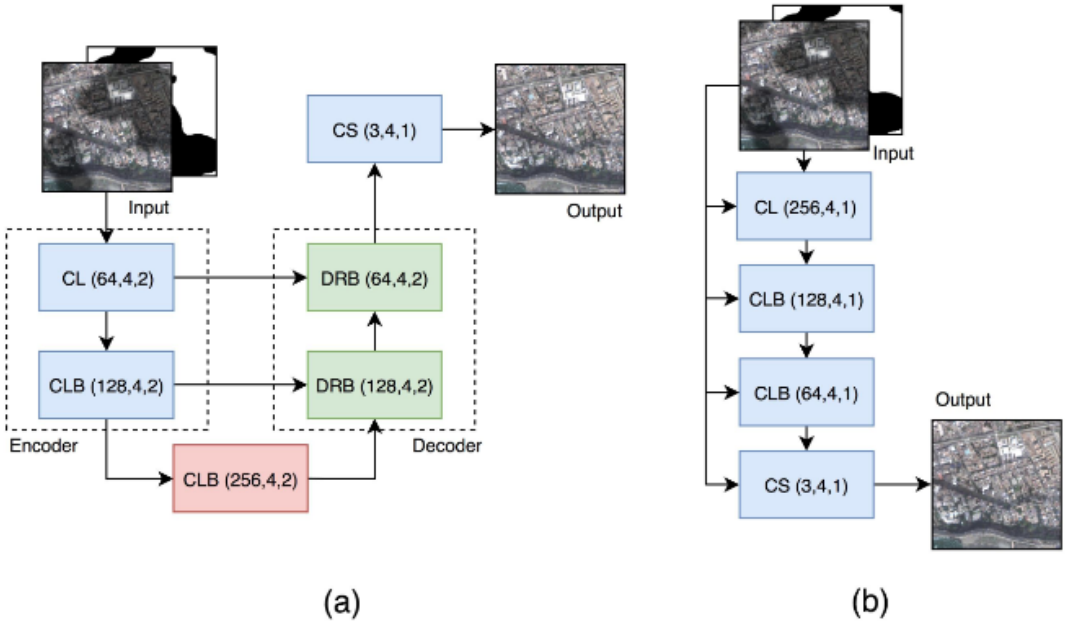
http://didt.inictel-uni.edu.pe/dataset/datasetshadowcorrectioncolor3.hdf5

https://www.researchgate.net/publication/330927171_Shadow_Removal_in_High-Resolution_Satellite_Images_Using_Conditional_Generative_Adversarial_Networks_5th_International_Conference_SIMBig_2018_Lima_Peru_September_3-5_2018_Proceedings

5.10. Shadow removal using cGANs

Generador

Discriminador



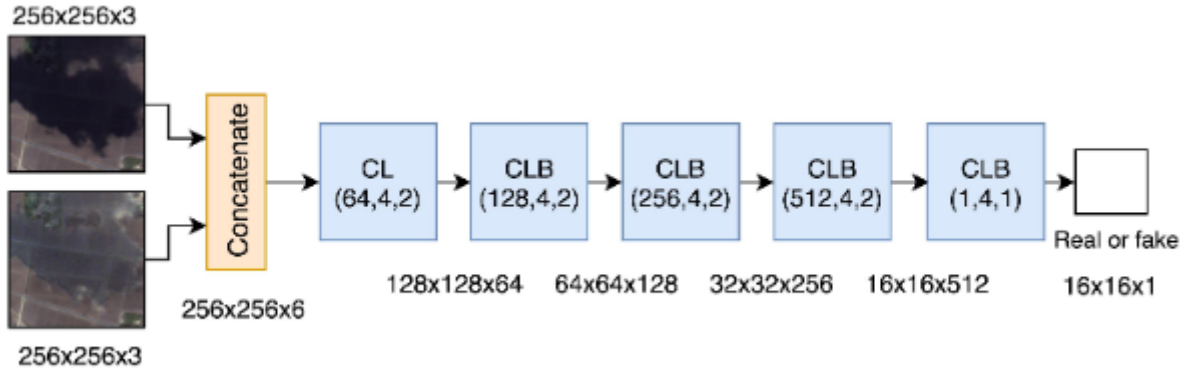


Fig. 4. Generator architectures. (a) cGAN1: U-NET architecture. (b) cGAN3: Continuous condition concatenation architecture.

Resultados

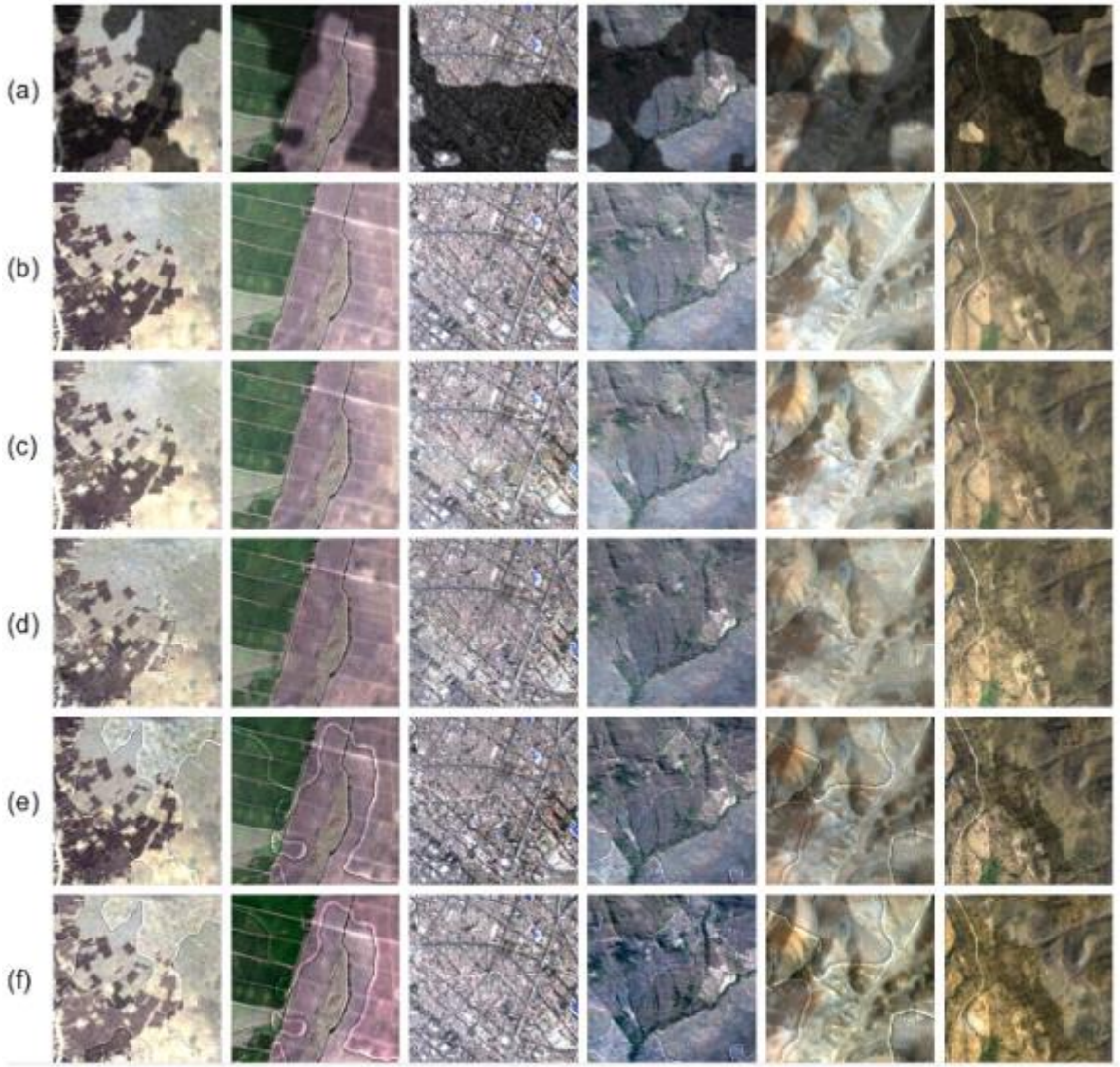


Fig. 7. Fake shadow removal in satellite images. (a) Artificially shadowed images. (b) Original images. (c) Our proposed method. (d) Gong et al. [13]. (e) Zigh et al. [10]. (f) Deb et al. [6].

5.10. Shadow removal using cGANs

Eliminación de sombras reales

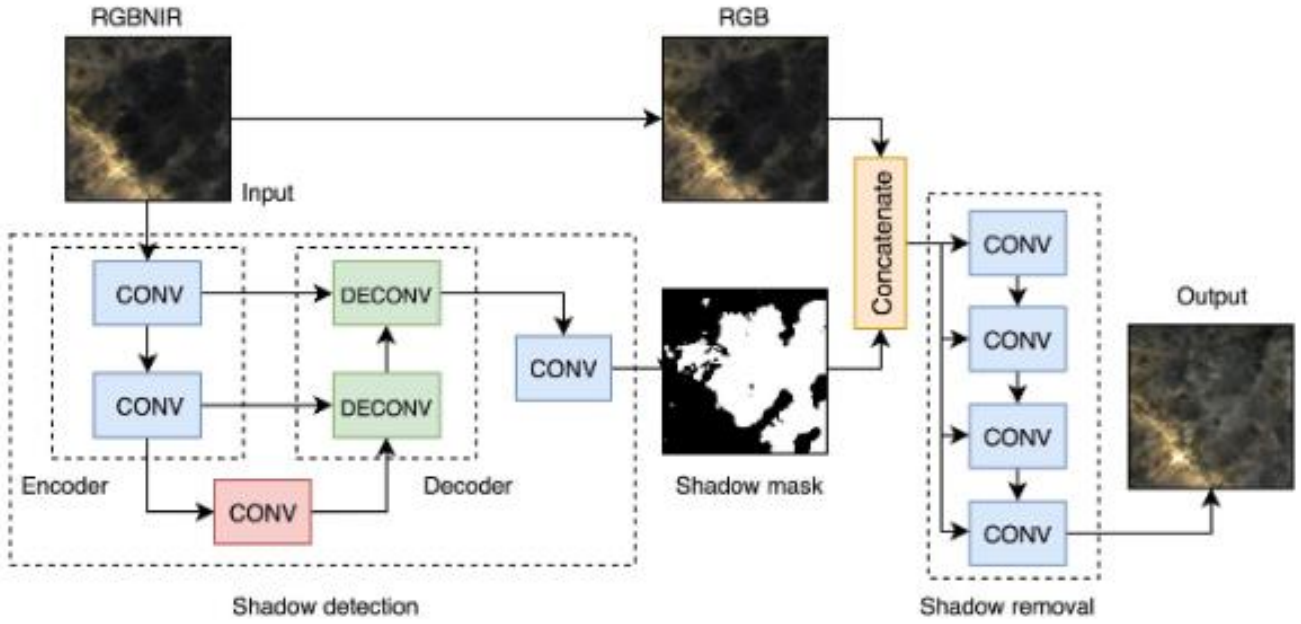


Fig. 8. Block diagram for removing shadows in real images.

Resultados

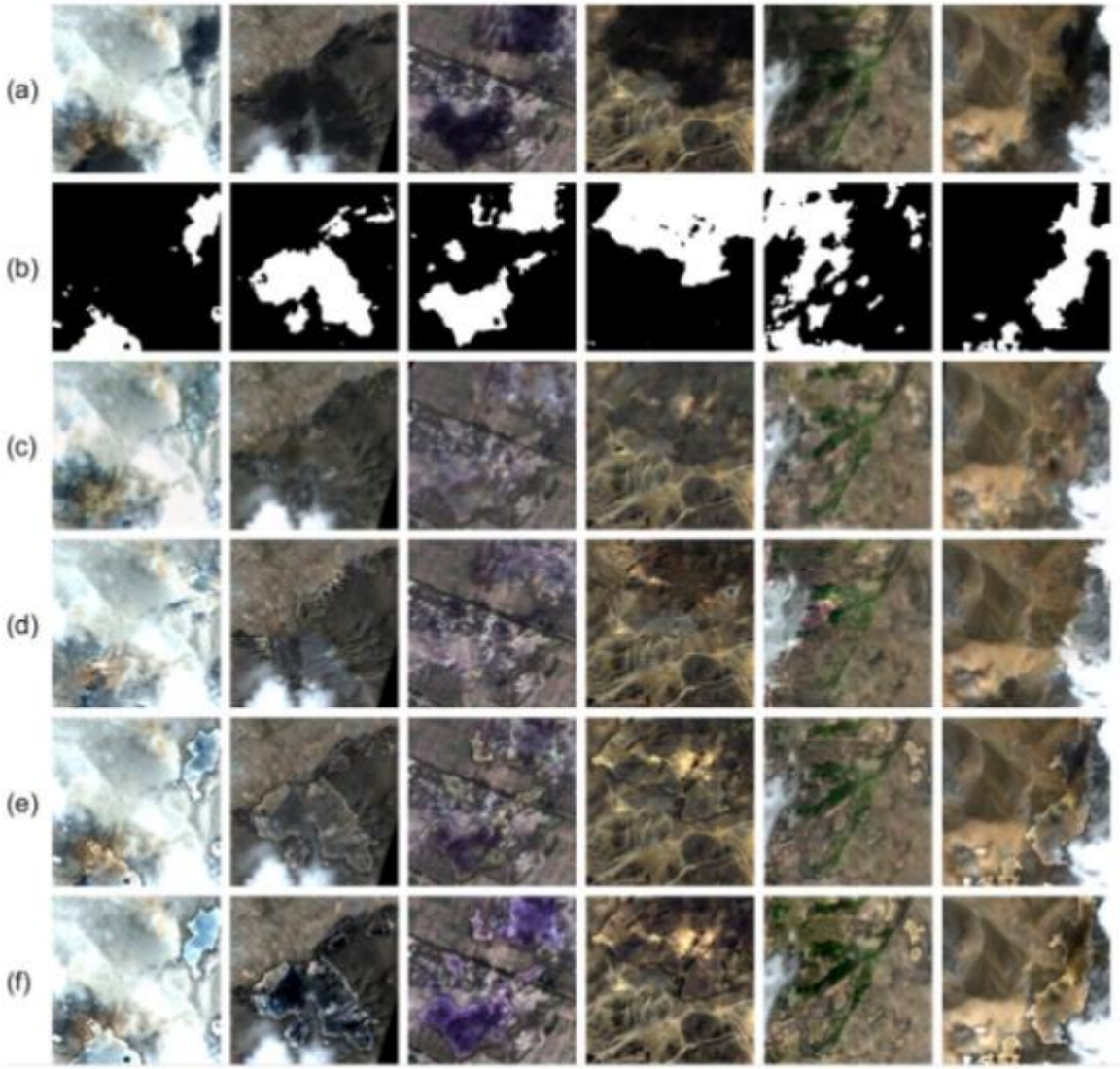


Fig. 9. Real shadow removal in satellite images. (a) Real shadowed images. (b) Detected shadows. (c) Our proposed method. (d) Gong et al. [13]. (e) Zigh et al. [10]. (f) Deb et al. [6].