Aplicación de Arquitecturas de Modelos Generativos a la Generación de Imágenes de Escenas Nocturnas a Diurnas de paisajes

Isabella Herrarte López, Gabriel Jeannot Viaña, Valentina Loaiza Mejia, Andrea Saavedra Viveros

Facultad de Ingeniería, Universidad Autónoma de Occidente  
Cali, Valle del Cauca, Colombia

[isabella.herrarte@uao.edu.co](mailto:Isabella.herrarte@uao.edu.co)[gabriel.jeannot@uao.edu.co](mailto:gabriel.jeannot@uao.edu.co)[valentina.loaiza@uao.edu.co](mailto:valentina.loaiza@uao.edu.co)[andrea.saavedra@uao.edu.co](mailto:Andrea.saavedra@uao.edu.co)

*.*

**Abstract**— The main purpose of this paper is to focus on training a Generative Adversarial Network (GAN) to generate images using a customized dataset consisting of landscape scenes. The goal is to achieve the conversion of night scenes to day scenes. During the process, different aspects that will guide the development of this project will be addressed. We start by defining the application context, i.e., the nature of the images used. Then, we proceed with the training of the neural network model that will be used for this particular case. The CycleGAN architecture will be used, which allows mapping between domains without the need for paired images. Then, this architecture and its applicability will be explained in detail, highlighting the most relevant fragments of its coding performed in Google Colab. Subsequently, relevant information on the validation of the neural network model used will be presented, showing the results obtained and analyzing them.

**Keywords**—EA Sports; Google Collab; Images; Patterns; Processing; GANs; Neural network.

**Resumen**— El propósito principal de este documento es centrarse en el entrenamiento de una Red Generativa Adversarial (GAN) para generar imágenes utilizando un conjunto de datos personalizado que consiste en escenas de paisajes. El objetivo es lograr la conversión de escenas nocturnas a diurnas. Durante el proceso, se abordarán diferentes aspectos que guiarán el desarrollo de este proyecto. Se comienza definiendo el contexto de aplicación, es decir, la naturaleza de las imágenes utilizadas. A continuación, se procede con el entrenamiento del modelo de red neuronal que se empleará para este caso en particular. Se utilizará la arquitectura CycleGAN, que permite el mapeo entre dominios sin necesidad de contar con imágenes emparejadas. Luego, se explicará en detalle esta arquitectura y su aplicabilidad, destacando los fragmentos más relevantes de su codificación realizada en Google Colab. Posteriormente, se presentará información relevante sobre la validación del modelo de red neuronal utilizado, mostrando los resultados obtenidos y realizando un análisis de los mismos.

**Palabras Clave**—EA Sports; Google Collab; Imágenes; Patrones; Procesamiento; red GAN; red Neuronal

# **Introducción**

Desde la aparición de la Inteligencia Artificial, la disciplina de la ingeniería ha experimentado transformaciones en la manera de enfrentar los desafíos, lo que ha llevado al uso de herramientas que buscan mejorar y crear soluciones más eficientes, seguras y poderosas. Estas soluciones se aplican a procesos complejos que son comunes en la vida cotidiana y que antes eran abordados de forma empírica por los seres humanos o requerían una preparación extensa y laboriosa [1]. Según la empresa estadounidense NetApp, especializada en gestión de datos y servicios en la nube, la Inteligencia Artificial (IA) se basa en la imitación de los procesos de inteligencia humana mediante la creación y aplicación de algoritmos en un entorno computacional dinámico. En términos simples, la IA busca que las computadoras piensen y actúen como los seres humanos. Debido a esto, la IA se presenta en diferentes contextos de aplicación, como el Aprendizaje Automático (Machine Learning) y la Inteligencia Computacional [2]. En particular, el Machine Learning es una forma de IA en la que se utilizan algoritmos para descubrir patrones recurrentes en diferentes tipos de datos (números, palabras, imágenes, estadísticas, etc.). A través de este proceso de aprendizaje, se logra mejorar el rendimiento en la ejecución de tareas específicas [3]. El Deep Learning, una rama del aprendizaje automático (machine learning), utiliza redes neuronales inspiradas en la biología del cerebro humano para procesar grandes cantidades de datos. Un ejemplo de esto son las Redes Neuronales Generativas Adversarias (GANs), que se basan en el Deep Learning para generar diversos tipos de datos, incluyendo imágenes que parecen reales. Desde su introducción, las GANs han evolucionado constantemente y se han convertido en una de las arquitecturas más utilizadas para generar patrones, y su utilidad no se limita a un propósito específico, ya que incluso pueden ser empleadas para crear conjuntos de datos. Por esta razón, las GANs han ganado relevancia recientemente y han permitido desarrollar aplicaciones con resultados impactantes. Algunas de las aplicaciones más destacadas incluyen la generación de ejemplos para conjuntos de datos de imágenes, fotografías de rostros humanos, imágenes realistas, personajes de dibujos animados e incluso la conversión entre escenas diurnas y nocturnas, que es el enfoque del presente documento. En este caso, se utilizará la técnica de CycleGAN (Image-to-Image) para convertir escenas diurnas a nocturnas en el contexto de generación de imágenes de estadios de fútbol. Estas imágenes podrían tener un potencial en videojuegos como FIFA de EA Sports o NBA2k23, con el objetivo de proporcionar a la comunidad de jugadores un mayor realismo deportivo. La aplicación principal de esta técnica es la capacidad de transformar y/o generar imágenes de manera más rápida que un editor tradicional, como Adobe Photoshop.

# **Marco Teórico**

Para comenzar, es necesario establecer una base teórica que abarque los modelos generativos y así poder explorar en detalle ciertos términos a lo largo del documento. Para lograr esto, se utilizará TechTarget como referencia principal al definir de manera fundamental las GANs [4], donde se precisa la siguiente información:

Una Red Generativa Adversarial (GAN) es un modelo de aprendizaje automático en el cual dos redes neuronales compiten entre sí para mejorar la precisión de sus predicciones. Las GAN funcionan generalmente sin supervisión y utilizan un marco de juego cooperativo de suma cero para aprender. Estas redes consisten en dos componentes principales: el Generador y el Discriminador. El Generador es una red neuronal convolucional que se encarga de generar salidas que se asemejen a datos reales. El Discriminador, por otro lado, es una red neuronal deconvolucional que tiene la tarea de distinguir entre datos generados artificialmente y datos reales.

El objetivo del Generador es producir salidas que sean indistinguibles de los datos reales, mientras que el Discriminador busca identificar las salidas generadas artificialmente. En esencia, las GAN crean sus propios datos de entrenamiento. A medida que se establece un ciclo de retroalimentación entre las redes adversas, el Generador comienza a generar resultados de mayor calidad, y el Discriminador mejora en su capacidad para discernir entre datos reales y generados artificialmente.

En términos de funcionamiento, el primer paso para configurar una GAN es establecer el resultado deseado y recopilar un conjunto inicial de datos de entrenamiento basado en esos parámetros. Estos datos se aleatorizan y se introducen en el Generador hasta que logre una precisión básica en la generación de salidas. Luego, las imágenes generadas se ingresan en el Discriminador junto con datos reales correspondientes al concepto original. El Discriminador analiza la información y devuelve una probabilidad entre 0 y 1 para indicar la autenticidad de cada imagen (1 se correlaciona con real y 0 se correlaciona con falso). Estos valores se verifican manualmente para determinar el éxito y se repite el proceso hasta alcanzar el resultado deseado.

Entre las principales ventajas de las GAN, se tiene que estas pueden tener ciertas ventajas sobre otros métodos de aprendizaje supervisado o no supervisado, estas ventajas son [5]:

1. Las GAN son una técnica de aprendizaje no supervisado, lo que significa que no necesitan datos etiquetados para su entrenamiento. A diferencia de otros métodos que requieren la laboriosa tarea de etiquetar manualmente los datos, las GAN pueden aprender las representaciones internas de los datos utilizando datos no etiquetados. Esto permite un proceso de adquisición de datos más eficiente y menos dependiente de la labor humana.

2. Las GAN tienen la capacidad de generar datos que son similares a los datos reales, lo que les otorga una amplia gama de aplicaciones en el mundo real. Pueden generar imágenes, texto, audio y video que son prácticamente indistinguibles de los datos reales. Estas imágenes generadas por las GAN encuentran utilidad en diversos campos como el marketing, el comercio electrónico, los juegos, la publicidad y muchas otras industrias.

Además, las GAN están siendo ampliamente utilizadas en casos de uso populares, especialmente en el ámbito de las ventas minoristas en línea, debido a su capacidad para comprender y recrear contenido visual con una notable precisión. Algunos de los casos de uso comunes incluyen:

* Rellenar imágenes de un esquema.
* Generar una imagen realista a partir de texto.
* Producir representaciones fotorrealistas de prototipos de productos.
* Convertir imágenes en blanco y negro en color.

En la producción de vídeo, las GAN se pueden utilizar para:

* Modele patrones de comportamiento humano y movimiento dentro de un marco.
* Predecir fotogramas de vídeo posteriores.
* Crear deepfake, que es la creación, manipulación o alteración de contenido con características falsas.

## Image2Image

Dentro del contexto de las redes generativas, es necesario utilizar traductores que permitan la conversión entre diferentes formatos de archivos. En este caso en particular, se utiliza la técnica conocida como "Image to Image", cuyo objetivo principal es transformar imágenes de un dominio a otra que posea el estilo o características de imágenes pertenecientes a otro dominio. La traducción de imagen a imagen aborda problemas relacionados con la visión y los gráficos, centrándose en aprender la correspondencia entre una imagen de entrada y una imagen de salida. Esta técnica tiene una amplia gama de aplicaciones, como la transferencia de estilos artísticos, la transformación de objetos, la transferencia de estaciones y la mejora de fotografías [6]. Para la conversión de escenas nocturnas a diurnas se utiliza entonces este lineamiento donde se destaca que, en la actualidad, ya existe un dataset llamado *day2nigh*t, dispuesto en GitHub, <https://github.com/NVlabs/MUNIT/issues/52>.



*Ilustración 1. Ejemplos de las transformaciones/generaciones de imágenes.*

## Arquitecturas

En la guía se enumeran tres arquitecturas específicas utilizadas para este tipo de aplicación: pix2pix, DiscoGAN y CycleGAN. No obstante, cabe destacar que hay muchas otras arquitecturas disponibles que abordan diversas problemáticas dentro del campo del Machine Learning. Las mencionadas anteriormente se caracterizan por:

**a. Pix2Pix**, es una variante de las GAN conocida como GAN condicional (cGAN), desarrollada por Phillip Isola y colaboradores. A diferencia de la GAN convencional, que utiliza solo datos reales y ruido para generar imágenes, el cGAN utiliza tanto datos reales como etiquetas adicionales para generar imágenes.

En términos simples, el generador en el cGAN aprende a mapear tanto los datos reales como el ruido, mientras que el discriminador aprende a representar tanto las etiquetas como los datos reales. Esta configuración permite que el cGAN sea adecuado para tareas de traducción de imagen a imagen, donde el generador está condicionado a una imagen de entrada para generar una imagen de salida correspondiente. En otras palabras, el generador usa una distribución de condición (o datos) como una guía o plano para generar una imagen objetivo [7].



*Ilustración 2. Traducción de imágenes con Pix2Pix.*

**b. CycleGAN,** es una arquitectura GAN ampliamente conocida y popularmente utilizada para aprender la transformación entre imágenes de diferentes estilos.

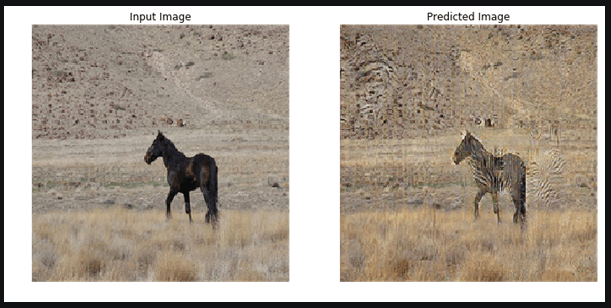
Por ejemplo, esta arquitectura es capaz de aprender el mapeo entre imágenes artísticas y realistas, transformaciones entre imágenes de caballos y cebras, conversiones de imágenes de invierno a verano, y muchas otras aplicaciones similares.

FaceApp es uno de los ejemplos más populares de CycleGAN, donde los rostros humanos se transforman en diferentes grupos de edad.

Diagrama

Descripción generada automáticamente

Como se ha mencionado anteriormente, hay 2 tipos de funciones que se están aprendiendo, una de ellas es G que transforma **X** en **Y** y la otra es F que transforma **Y** a **X** y comprende dos modelos GAN individuales. Por lo tanto, se dispone de 2 funciones de discriminador Dx, Dy [8].

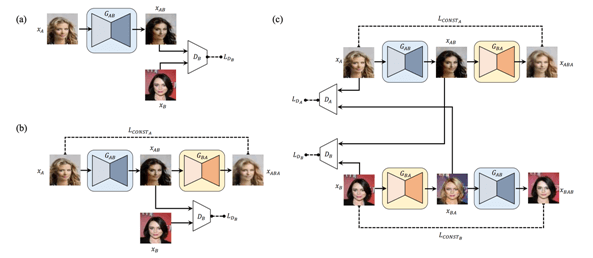


*Ilustración 3 Caballos y cebras.*

c. Teniendo en cuenta la última referencia, **DiscoGAN** se hizo muy popular debido a su capacidad para aprender relaciones entre dominios dados datos no supervisados. Para los humanos, las relaciones entre dominios son muy naturales. Dadas las imágenes de dos dominios diferentes, un humano puede descubrir cómo se relacionan entre sí. El concepto central de DiscoGAN es muy similar a CycleGAN:

Ambas partes de la arquitectura aprenden 2 funciones de transformación individuales, uno aprende una transformación del dominio X al dominio Y, mientras que el otro aprende un mapeo inverso y ambos usan la pérdida de reconstrucción como una medida de qué tan bien se reconstruye la imagen original después de una transformación dos veces en todos los dominios.

Ambos siguen el principio de que, si se transforma una imagen de un dominio1 a dominio2 y luego se pretende regresar al dominio1 nuevamente, entonces debe coincidir con la imagen original.

*Ilustración 4. Aplicaciones de DiscoGAN.*

## Decisión Final

El equipo de trabajo decide enfocarse en la arquitectura específica de Pix2Pix, la cual es una red neuronal profunda basada en el entrenamiento GAN de una red UNet. El modelo de Pix2Pix se utiliza para calcular la transformación de una imagen x en una imagen y, que generalmente difieren en estilo. En términos generales, esta arquitectura consta de dos redes neuronales que desempeñan diferentes tareas. La primera es la red generadora, encargada de aprender a generar el contenido deseado, mientras que la segunda es la red discriminadora, cuya función es aprender a distinguir si una imagen fue generada por la red generadora o si es una imagen original del conjunto de datos [9].

En la configuración actual, se cuenta con un encoder que tiene la función de comprimir la información de la imagen de entrada, y luego el decoder utiliza esta información comprimida para generar la imagen deseada. Sin embargo, surge un problema potencial cuando la información se comprime demasiado, ya que podría dificultar la decodificación posterior. Para abordar este problema, se utiliza una técnica llamada skip connection, que implica crear conexiones entre las capas de la red. Estas conexiones permiten que la información se ataje en las capas intermedias o incluso se salte algunas capas, evitando así una compresión excesiva y facilitando la decodificación.

Diagrama

Descripción generada automáticamente

*Ilustración 6. Ejemplo de skip connection.*

Como se puede observar, la red no es perfecta, especialmente si se entrena durante largos períodos de tiempo. Cuanto más documentación o conocimiento se tenga sobre la pérdida, más preciso será el modelo. Por lo tanto, se requiere invertir varias horas en el entrenamiento para obtener resultados aceptables, además de contar con un dataset de más de 400 imágenes. Si se tienen muchas imágenes, es posible que se pueda mejorar la calidad de algunas imágenes generadas, pero esto conlleva un mayor consumo computacional y el uso obligatorio de una GPU.

En cuanto a su aplicación, según Tensorflow, Pix2Pix no tiene una aplicación específica, ya que se puede utilizar para una amplia variedad de tareas. Estas incluyen la síntesis de fotos a partir de mapas de etiquetas, la generación de fotos a color a partir de imágenes en blanco y negro, la conversión de fotos de Google Maps en imágenes aéreas y la transformación de bocetos en fotografías, entre otras posibilidades [9].

# **Enunciado**

La propuesta del proyecto final del curso, consiste en utilizar un conjunto de datos personalizado para entrenar una red generativa. Esta red podría generar imágenes o audio. Se sugiere utilizar las herramientas de Deep Learning, como Keras y/o TensorFlow, para llevar a cabo este desarrollo.

Con el objetivo de lograr la conversión de escenas de día a escenas nocturnas en el contexto propio de los paisajes, se determina obligatorio la organización de un dataset donde se realiza la división entre las imágenes tomadas en horas del día y horas de la noche, con una marcada diferencia entre ambas. El dataset posee un total de 100 imágenes, 50 para cada carpeta, con la opción de lograr ampliar esta cantidad pero evitando realizarlo por resultados negativos.

Una muestra de las imágenes se ofrece a continuación, comenzando por la vista de día y su similar de noche.



*Ilustración 7. Imágenes de día y noche de los paisajes.*

# **Desarrollo de la Solución**

La implementación de la solución se lleva a cabo utilizando la herramienta Colaboratory de Google. Se usa un Notebook en Python y Tensorflow, y se implementa la arquitectura Pix2Pix obtenida del tutorial “Generando FLORES realistas con IA - Pix2Pix” de DotCSV. El código completo se encuentra disponible en el siguiente Colab. A continuación, se describen los aspectos más relevantes del código.

Inicialmente, se realiza el procesamiento de los datos de entrenamiento y validación. Esto incluye definir la ruta donde se encuentran los componentes del dataset, realizar el preprocesamiento de los datos y establecer cuál será el target y cuál será la entrada. En este caso específico, las imágenes diurnas se emplean como entrada y se mapean a imágenes nocturnas.

A continuación, se implementa la arquitectura Pix2Pix, teniendo en cuenta los aspectos mencionados en el artículo relacionado a la arquitectura [15]. Pix2Pix es una red generativa adversaria condicionada, lo que significa que se puede condicionar la salida en función de una entrada. El modelo generativo consta de un discriminador y un generador. El generador sigue una arquitectura de tipo U-NET, que incluye un encoder, decoder y skip connections conectadas al mismo nivel. Por otro lado, el discriminador utiliza una arquitectura llamada Patch GAN, que evalúa las imágenes por lotes de píxeles.

A nivel de codificación, usando Tensorflow, se crea una función llamada "downsample" que representa el tipo de capa empleada en el encoder. Esta función toma como parámetros el número de filtros y si se aplica Batch Normalization o no. Se generan tres capas, una de convolución, una de Batch Normalization y una de activación con una ReLU. Estas capas forman el embudo de cuello de botella de la red neuronal. Además, se especifica que los parámetros deben inicializarse con un ruido gaussiano con media 0 y desviación estándar de 0.02.

def downsample(filters, apply\_batchnorm=True):

  result = Sequential()

  initializer = tf.random\_normal\_initializer(0, 0.02)

  result.add(Conv2D(filters,

                    kernel\_size=4,

                    strides=2,

                    padding="same",

                    kernel\_initializer=initializer,

                    use\_bias=not apply\_batchnorm))

  if apply\_batchnorm:

    result.add(BatchNormalization())

  result.add(LeakyReLU())

  return result

Por otro lado, en las capas del decodificador se utiliza otra función llamada “upsample”. Esta función representa otro tipo de capa que consta de una capa convolucional transpuesta, un Batch Normalization y una capa de activación ReLU. Además, se incluyen como parámetros el número de filtros y si se aplica o no un Dropout del 50 %.

def upsample(filters, apply\_dropout=True):

  result = Sequential()

  initializer = tf.random\_normal\_initializer(0, 0.02)

  result.add(Conv2DTranspose(filters,

                             kernel\_size=4,

                             strides=2,

                             padding="same",

                             kernel\_initializer=initializer,

                             use\_bias=False))

  result.add(BatchNormalization())

  if apply\_dropout:

    result.add(Dropout(0.5))

  result.add(ReLU())

  return result

Luego, se procede a construir el generador mediante la siguiente función, la cual se encarga de combinar las distintas capas. En esta función se especifican el número de parámetros, así como las condiciones para incluir la normalización en lotes o la reducción del 50 % de las conexiones (Dropout). Además, se usa una capa de convolución transpuesta en la salida, lo que refleja la estructura de la red U-Net.

def Generator():

  inputs = tf.keras.layers.Input(shape=[None,None,3])

  down\_stack = [

    downsample(64, apply\_batchnorm=False),

    downsample(128),

    downsample(256),

    downsample(512),

    downsample(512),

    downsample(512),

    downsample(512),

    downsample(512),

  ]

  up\_stack = [

     upsample(512, apply\_dropout=True),

     upsample(512, apply\_dropout=True),

     upsample(512, apply\_dropout=True),

     upsample(512),

     upsample(256),

     upsample(128),

     upsample(64),

  ]

  initializer = tf.random\_normal\_initializer(0, 0.02)

  last = Conv2DTranspose(filters = 3,

                         kernel\_size = 4,

                         strides=2,

                         padding="same",

                         kernel\_initializer=initializer,

                         activation="tanh")

  x = inputs

  s = []

  concat = Concatenate()

  for down in down\_stack:

    x = down(x)

    s.append(x)

  s = reversed(s[:-1])

  for up, sk in zip(up\_stack, s):

    x = up(x)

    x = concat([x, sk])

  last = last(x)

  return Model(inputs=inputs, outputs=last)

Una vez completado el modelo del generador, se proceed a crear el discriminador mediante una función que define una red convolucional. Esta red toma como entrada tanto una imagen real, como una imagen falsa. Además, el discriminador incluye una capa convolucional transpuesta como un solo canal. Esta capa permite analizar, píxel por píxel, si la región a la que pertenece el píxel parece o no auténtica.

def Discriminator():

  ini = Input(shape=[None, None, 3], name = "input\_img")

  gen = Input(shape=[None, None, 3], name = "gener\_img")

  con = concatenate([ini ,gen])

  initializer = tf.random\_normal\_initializer(0, 0.02)

  down1 = downsample(64, apply\_batchnorm=False)(con)

  down2 = downsample(128)(down1)

  down3 = downsample(256)(down2)

  down4 = downsample(512)(down3)

  last = tf.keras.layers.Conv2D(filters=1,

                                kernel\_size=4,

                                strides=1,

                                kernel\_initializer=initializer,

                                padding="same")(down4)

  return Model(inputs=[ini, gen], outputs=last)

Dentro de la implementación del código, se encuentra una función denominada "discriminator\_loss" encargada de calcular la pérdida del discriminador en una red generativa adversaria (GAN). Esta función es esencial para el entrenamiento de la GAN, ya que evalúa la discrepancia entre las salidas del discriminador para las imágenes reales y generadas. Mediante el uso de una función de pérdida predefinida, se calcula la pérdida individual para ambas tipos de imágenes, y luego se suma para obtener la pérdida total del discriminador. Este cálculo de pérdida permite mejorar la capacidad del discriminador para distinguir entre imágenes auténticas y generadas. Al minimizar la pérdida total del discriminador, se logra un mejor equilibrio en la generación de imágenes realistas por parte de la red generativa.

def discriminator\_loss(disc\_real\_output, disc\_generated\_output):

real\_loss = loss\_object(tf.ones\_like(disc\_real\_output), disc\_real\_output)

generated\_loss = loss\_object(tf.zeros\_like(disc\_generated\_output), disc\_generated\_output)

total\_disc\_loss = real\_loss + generated\_loss

return total\_disc\_loss

Dentro del contexto de la red generativa adversaria (GAN), se implementa la función 'generator\_loss' que calcula la pérdida del generador. Esta función combina dos aspectos fundamentales: la pérdida de ganancia (gan\_loss) y el error medio absoluto (l1\_loss). La pérdida de ganancia evalúa la discrepancia entre las salidas generadas y las esperadas, mientras que el error medio absoluto mide la diferencia promedio entre las imágenes generadas y las imágenes objetivo. La pérdida total del generador se obtiene ponderando el error l1 con un factor LAMBDA. De esta manera, la función de pérdida guía el entrenamiento del generador para producir imágenes realistas y fieles a las imágenes objetivo.

LAMBDA = 100

def generator\_loss(disc\_generated\_output, gen\_output, target):

gan\_loss = loss\_object(tf.ones\_like(disc\_generated\_output), disc\_generated\_output)

#Mean absolute error.

l1\_loss = tf.reduce\_mean(tf.abs(target - gen\_output))

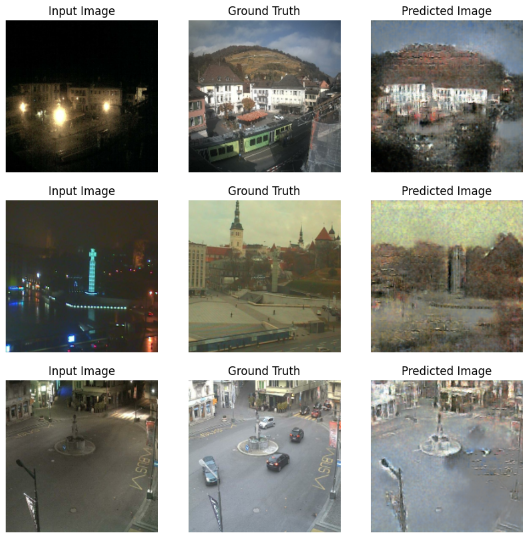
total\_gen\_loss = gan\_loss + (LAMBDA \* l1\_loss)

return total\_gen\_loss

El código para los entrenamientos, aunque importante, no se describe en detalle en este inciso para evitar mayor extensión. Sin embargo, utilizando la arquitectura implementada en el código, se lleva a cabo el entrenamiento durante 500 épocas. Aunque se cuenta con el uso de una GPU, el proceso puede tomar aproximadamente 2 horas. Durante el entrenamiento, se puede observar el progreso de la red a través de un conjunto de imágenes almacenadas en el directorio de salida. Una vez finalizado el entrenamiento, se generan nuevas imágenes usando datos de entrada adicionales, las cuales se presentan y analizan como resultados en el siguiente apartado.

# **Resultados**

A partir del modelo entrenado, se puede analizar los resultados de la arquitectura Pix2Pix para resolver el desafío de transformar imágenes nocturnas a diurnas, en particular en relación con paisajes. Como primer resultado, se encontró que el tiempo de entrenamiento necesario para completar 500 iteraciones es de aproximadamente 2 horas, lo cual es considerable considerando las limitaciones de tiempo en entornos de desarrollo como Colab. Por otro lado, los resultados obtenidos en el cambio de dominio entre imágenes, como se muestra en la figura 14, revelan que, aunque no se logra la respuesta deseada en su totalidad, algunos aspectos clave entre el día y la noche, como el cielo, se vuelven más oscuros.



*Ilustración 14. Resultados*

La falta de lograr la respuesta deseada puede atribuirse a dos aspectos principales: el entrenamiento y el conjunto de datos utilizado. En primer lugar, debido a que se realizaron un número limitado de entrenamientos, el modelo no pudo adquirir suficiente capacidad para realizar el mapeo entre los diferentes paisajes, lo cual se complica aún más debido a la variedad de patrones presentes en los paisajes, como el color de la vegetación y los asientos, los cuales varían según cada paisaje. Por otro lado, la disponibilidad de un conjunto de datos limitado y no emparejado en términos de las características específicas de los paisajes, como el color y la estructura, representa un desafío para arquitecturas como Pix2Pix.

# **Conclusiones**

A partir del desarrollo experimentado por los miembros del equipo y los resultados de este documento, se adquirió una mayor

# **Referencias**

| [1] | EDS Robotics, «Aplicaciones de la inteligencia artificial en la ingeniería industrial,» EDS Robotics, 2 Diciembre 2021. [En línea]. Available: https://www.edsrobotics.com/blog/aplicaciones-inteligencia-artificial-en-ingenieria-industrial/. [Último acceso: 26 Mayo 2023]. |
| --- | --- |
| [2] | NetApp, «What is Artificial Intelligence?,» NetApp, [En línea]. Available: https://www.netapp.com/what-is-artificial-intelligence/. [Último acceso: 26 Mayo 2023]. |
| [3] | DataScientest, «Machine Learning: definición, funcionamiento, usos,» DataScientest, [En línea]. Available: https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos. [Último acceso: 26 Mayo 2023]. |
| [4] | K. Yasar, «generative adversarial network (GAN),» TechTarget, [En línea]. Available: https://www.techtarget.com/searchenterpriseai/definition/generative-adversarial-network-GAN. [Último acceso: 26 Mayo 2023]. |
| [5] | K. Ahirwar, Generative Adversarial Networks Projects. |
| [6] | P. Isola, J.-Y. Zhu, T. Zhou y A. Efros, «Image-to-Image Translation with Conditional Adversarial Networks,» 2018. |
| [7] | N. Barla, «Pix2pix: Key Model Architecture Decisions,» Neptune.ai, 25 Abril 2023. [En línea]. Available: https://neptune.ai/blog/pix2pix-key-model-architecture-decisions. [Último acceso: 26 Mayo 2023]. |
| [8] | S. Das, «6 GAN Architectures You Really Should Know,» Neptune.ai, 21 Abril 2023. [En línea]. Available: https://neptune.ai/blog/6-gan-architectures. [Último acceso: 26 Mayo 2023]. |
| [9] | Tensorflow, «pix2pix: traducción de imagen a imagen con una GAN condicional,» Tensorflow, [En línea]. Available: https://www.tensorflow.org/tutorials/generative/pix2pix?hl=es-419. [Último acceso: 26 Mayo 2023]. |
| [10] | J.-Y. Zhu, T. Park, P. Isola y A. Efros, «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,» 2020. |
| [11] | M.-Y. Liu, T. Breuel y J. Kautz, «Unsupervised Image-to-Image Translation Networks,» 2018. |