APLICACIÓN DE ARQUITECTURAS DE MODELOS GENERATIVOS A LA GENERACIÓN DE IMÁGENES

Escenas diurnas a nocturnas de Estadios de Futbol para EA Sports

Brahyan Camilo Marulanda Muñoz
Universidad Autónoma de Occidente
Yumbo, Valle del Cauca, Colombia
brahyan.marulanda@uao.edu.co

Henry Carmona Collazos
Universidad Autónoma de Occidente
Cali, Valle del Cauca, Colombia
henry.carmona@uao.edu.co

Daniel Alejandro Tobar Álvarez
Universidad Autónoma de Occidente
Cali, Valle del Cauca, Colombia
daniel.ale_tobar@uao.edu.co

Diego Iván Perea Montealegre
Universidad Autónoma de Occidente
Cali, Valle del Cauca, Colombia
diego.perea@uao.edu.co

***Abstract*—**This paper will be focused on the training of a Generative Network (GANs) for the generation of images, using a customized data set (soccer stadium scenes), by means of which it will be possible to convert a daytime scene into a nighttime scene. Within the processing, there will be different steps that will structure its development. It starts with the definition of the images (an application context), then we will have the training of the neural network model to be used which, for the specific case, will make use of the CycleGAN architecture that allows the mapping between domains without using pairwise images, then the explanation of this architecture and its applicability will be delimited, exposing the most relevant fragments of its coding made in Google Collab, after that, information relevant to the validation of the neural network model used will be shown, illustrating the results obtained and an analysis of the same.

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

***Resumen*—**El desarrollo del presente documento estará enfocado principalmente al entrenamiento de una Red Generativa (GANs) para la generación de imágenes, usando un data set personalizado (escenas de estadios de futbol), mediante la cual será posible la conversión de una escena diurna en nocturna. Dentro del procesamiento, se tendrán diferentes incisos que estructurarán el desarrollo de este. Se inicia con la definición de las imágenes (un contexto de aplicación), luego se ello se tendrá el entrenamiento del modelo de red neuronal a utilizar que, para el caso específico, se hará uso de la arquitectura CycleGAN que permite el mapeo entre dominios sin hacer uso de imágenes por pares, luego se delimitará la explicación de esta arquitectura y su aplicabilidad, exponiendo los fragmentos más relevantes de su codificación realizada en Google Collab pues luego de ello, se mostrará información relevante a la validación del modelo de red neuronal utilizado, ilustrando los resultados obtenidos y un análisis de los mismos.

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

INTRODUCCIÓN

Desde la llegada de la Inteligencia Artificial, la ingeniería ha presentado ciertos cambios en cuanto a la forma de abordar las problemáticas y, por ende, ha recurrido al uso de herramientas que optimicen y generen soluciones más ágiles, seguras y potentes para que sean aplicadas a procesos complejos cotidianos, los cuales eran solucionados anteriormente por los humanos ya sea de forma empírica o con trabajos arduamente preparados [1]. Según NetApp, empresa estadounidense de gestión de datos y servicios de datos en la nube, "la Inteligencia Artificial (IA)[1] es la base a partir de la cual se imitan los procesos de inteligencia humana mediante la creación y la aplicación de algoritmos creados en un entorno dinámico de computación. O bien, dicho de forma sencilla, la IA consiste en intentar que los ordenadores piensen y actúen como los humanos". Por tal motivo, tiene diferentes contextos de aplicación o formas de realizarse, las cuales corresponden al Machine Learning y la Inteligencia Computacional. De este modo, el Machine Learning, es la forma de IA en la que, mediante algoritmos que descubren *patterns* (es decir, patrones recurrentes) de diferentes tipos de Data (números, palabras, imágenes, estadísticas, etc.), se tiene un aprendizaje y a partir de ello, se mejora el rendimiento en la ejecución de una tarea específica [2]. Ahora, el Deep Learning es un subconjunto del machine learning donde las redes neuronales son algoritmos inspirados en la biología del cerebro humano que aprenden grandes cantidades de datos. Este es el caso de las Redes Neuronales Generativas Adversarias (GANs) cuya noción es el uso del Deep Learning para generar cualquier tipo de datos, entre ellos, imágenes que parecen reales. Desde su aparición, han estado en constante evolución y se han convertido en una de las arquitecturas más usadas para generar patrones, pues su uso no se remite a un fin, sino que pueden llegar hasta la creación de datasets. Por eso, son una de las arquitecturas que más relevancia han tomado últimamente y ha permitido desarrollar aplicaciones con resultados bastante llamativos. Entre las aplicaciones más destacadas, se tiene la generación de ejemplos para datasets de imágenes, fotografías de rostros humanos, fotografías realistas, personajes de dibujos animados o hasta la conversión de escenas diurnas en nocturnas o viceversa, que es el enfoque del presente documento, donde se hará uso de la técnica de CycleGAN (Image-to-Image) para la conversión de escenas diurnas a nocturnas relacionadas con la generación de imágenes para estadios de fútbol, que podrían tener un potencial en videojuegos como FIFA de EA Sports o NBA2k23, donde se busca brindar a la comunidad *gamer* un mayor realismo deportivo. Lo esencial de esta aplicación es poder transformar y/o generar imágenes en un menor tiempo que un editor, como podría ser Adobe Photoshop.

[1] Información tomada de NetApp [En Línea]. Disponible en: https://www.netapp.com/es/artificial-intelligence/what-is-artificial-intelligence/

MARCO TEÓRICO

En primer lugar, se le debe atribuir una fundamentación a lo que corresponde los modelos generativos para ir desglosando cierta terminología mediante el desarrollo del paper. En primero lugar, para definir de forma esencial las GANs, se tomará como referente TechTarget [3],donde se precisa la siguiente información:

Una red generativa antagónica (GAN) es un modelo de aprendizaje automático (ML) en el que dos redes neuronales compiten entre sí para ser más precisas en sus predicciones. Las GAN generalmente se ejecutan sin supervisión y utilizan un marco de juego cooperativo de suma cero para aprender. Dichas redes estructurales, se conocen como el *Generador* y el *Discriminador*. El primero, es una red neuronal convolucional y el discriminador es una red neuronal deconvolucional. El objetivo del generador es fabricar artificialmente salidas que podrían confundirse fácilmente con datos reales. El objetivo del discriminador es identificar qué productos recibe han sido creados artificialmente. Esencialmente, las GAN crean sus propios datos de entrenamiento. A medida que continúe el ciclo de retroalimentación entre las redes adversas, el generador comenzará a producir resultados de mayor calidad y el discriminador mejorará en el marcado de datos que se han creado artificialmente.

En cuanto a funcionamiento, el primer paso para establecer una GAN es identificar el resultado final deseado y recopilar un conjunto de datos de entrenamiento inicial basado en esos parámetros. Estos datos se aleatorizan y se introducen en el generador hasta que adquiere una precisión básica en la producción de salidas. Después de esto, las imágenes generadas se introducen en el discriminador junto con los puntos de datos reales del concepto original. El discriminador filtra la información y devuelve una probabilidad entre 0 y 1 para representar la autenticidad de cada imagen (1 se correlaciona con real y 0 se correlaciona con falso). Estos valores se comprueban manualmente para verificar si tienen éxito y se repiten hasta que se alcanza 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, según [4], estas ventajas son:

a. Son un método de aprendizaje no supervisado: la adquisición de datos etiquetados es un proceso manual que lleva mucho tiempo. Las GAN no requieren datos etiquetados; Pueden ser entrenados usando datos no etiquetados a medida que aprenden las representaciones internas de los datos.

b. Las GAN generan datos similares a los datos reales. Debido a esto, tienen muchos usos diferentes en el mundo real. Pueden generar imágenes, texto, audio y video que son indistinguibles de los datos reales. Las imágenes generadas por las GAN tienen aplicaciones en marketing, comercio electrónico, juegos, anuncios y muchas otras industrias.

Asimismo, se tienen casos de uso populares para GAN pues se están convirtiendo en un modelo de ML popular para las ventas minoristas en línea debido a su capacidad para comprender y recrear contenido visual con una precisión cada vez más notable. Los casos de uso 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.

A. Image2Image

Como el concepto hace parte de las redes generativas, es necesario el uso de traductores entre formatos de archivos. En este caso específico, se delimita Image to Image, cuya principal tarea es el cambio de imágenes de un dominio a que tengan el estilo (o características) de las imágenes de otro dominio. La traducción de imagen a imagen es una clase de problemas de visión y gráficos donde el objetivo es aprender el mapeo entre una imagen de entrada y una imagen de salida. Se puede aplicar a una amplia gama de aplicaciones, como la transferencia de estilo de colección, la transfiguración de objetos, la transferencia de estaciones y la mejora de fotos [5]. Para la conversión de escenas diurnas a nocturnas se utiliza entonces este lineamiento donde se destaca que, en la actualidad, ya existe un dataset llamado *day2night*, dispuesto en GitHub, https://github.com/NVlabs/MUNIT/issues/52.

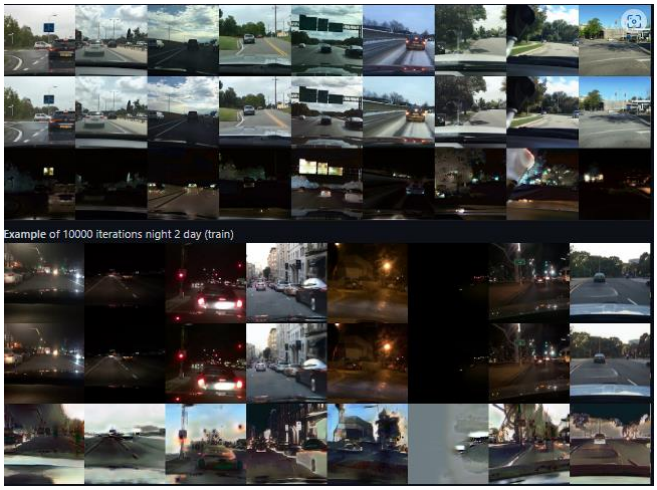


Ilustración 1. Ejemplos de las transformaciones/generaciones de imágenes.

B. Arquitecturas

En la guía se mencionan tres arquitecturas para este tipo de aplicación, las cuales corresponden a pix2pix, DiscoGAN y CycleGAN. Sin embargo, existen muchas más de estas [6] en aras de abordar diferentes problemáticas desde el Machine Learning. Las mencionadas corresponden a:

a. **Pix2Pix**, es un GAN condicional (cGAN) que fue desarrollado por Phillip Isola, et al. A diferencia de Vainilla GAN, que utiliza solo datos reales y ruido para aprender y generar imágenes, cGAN utiliza datos reales, ruido y etiquetas para generar imágenes.

   En esencia, el generador aprende el mapeo de los datos reales, así como del ruido. Del mismo modo, el

discriminador también aprende la representación de las etiquetas, así como de los datos reales. Esta configuración permite que 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 utiliza una distribución de condición (o datos) como una guía o un plano para generar una imagen objetivo [6].

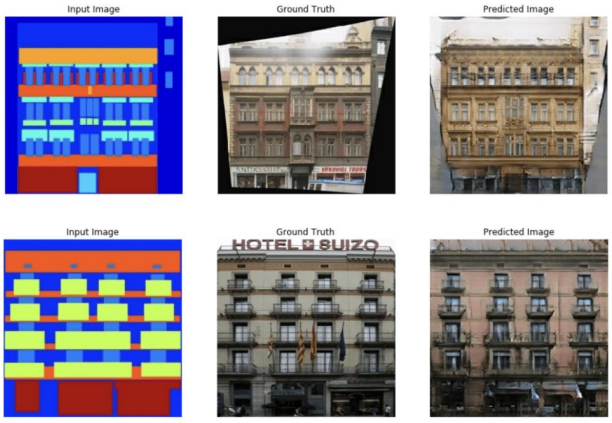
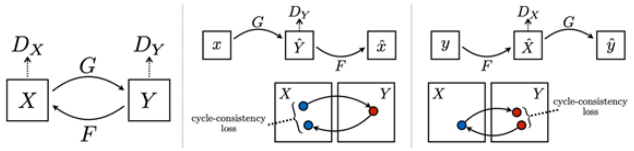


Ilustración 2. Traducción de imágenes con Pix2Pix.

b. **CycleGAN** es una arquitectura GAN muy popular que se utiliza principalmente para aprender la transformación entre imágenes de diferentes estilos.

Como ejemplo, este tipo de formulación puede aprender: un mapa entre imágenes artísticas y realistas, una transformación entre imágenes de caballo y cebra, una transformación entre la imagen de invierno y la imagen de verano, y así sucesivamente.

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



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 [7].

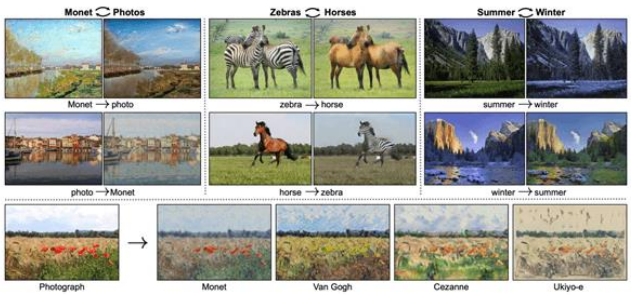


Ilustración 3. Aplicaciones de CycleGAN.

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.

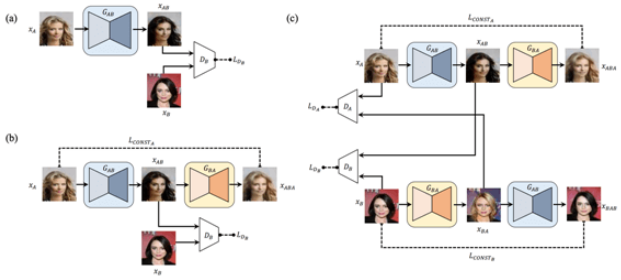


Ilustración 4. Aplicaciones de DiscoGAN.

C. Pre-elección de CycleGAN

La razón por la cual en un inicio el equipo de trabajo decide hacer uso de esta arquitectura es porque CycleGAN no se necesita de imágenes pares para poder hacer la traducción, a diferencia de pix2pix, sin embargo, es más preciso [8]. Además, para el caso específico de la transformación día-noche de las imágenes, el enfoque CycleGAN se destaca por encima de Unit al correlacionar las imágenes con las de Dia y Noche, así:

RetinaNet	Day	Night	All
Real Data Images (NEXET)	0.8664	0.8406	0.8535
CycleGAN	0.8701 (+0.42%)	**0.8571 (+1.96%)**	**0.8636 (+1.18%)**
UNIT	**0.8749 (+0.98%)**	0.8512 (+1.26%)	0.8631 (+1.12%)

Faster R-CNN	Day	Night	All
Real Data Images (NEXET)	0.9015	0.8822	0.8919
CycleGAN	**0.9087 (+0.79%)**	0.8881 (+0.66%)	0.8984 (+0.72%)
UNIT	0.9066 (+0.56%)	**0.8929 (+1.21%)**	**0.8998 (+0.88%)**

Ilustración 5. Unit vs CycleGAN.

Se hace la salvedad que Unit, UNsupervised Image-to-Image Translation Networks, tiene como objetivo aprender una distribución conjunta de imágenes en diferentes dominios mediante el uso de imágenes de las distribuciones marginales en dominios individuales. Dado que existe un conjunto infinito de articulaciones distribuciones que pueden llegar a las distribuciones marginales dadas, se podría inferir nada sobre la distribución conjunta de las distribuciones marginales sin supuestos adicionales. Para abordar el problema, se hace un espacio compartido-latente y se propone un marco de traducción de imagen a imagen no supervisado basado en GAN acopladas [9].

Uno de los posibles *potenciales* o aplicaciones que podría traer consigo el uso de esta arquitectura o enfoque es el hecho del manejo de información en grandes. El futbol, sus dinámicas y la estructura de sus estadios son las cosas más representativas del deporte. Por tal motivo, con el uso de CycleGAN se espera hacer la transformación de día a noche de los estadios de futbol y resaltar de ellos, sus características más importantes al observar cuales han sido las modificaciones que se ha realizado durante el procesamiento de estas, pues son estas cantidades de imágenes con las que trabajan grandes empresas de videojuegos, tal es el caso de EA Sports para su juego FIFA 23.

D. Decisión Final

Teniendo presente que existieron unos problemas en el momento de utilizar la arquitectura de CycleGAN, el equipo de trabajo opta por precisar la arquitectura Pix2Pix, la cual es una red neuronal profunda que se basa en el entrenamiento tipo GAN de una red tipo UNet. El modelo pix2pix calcula la transformación de una imagen *x* en una imagen *y*, las cuales difieren, generalmente, en estilo. Según [10], una idea general es que este tipo de arquitecturas consta de 2 redes neuronales que resuelven tareas diferentes. La primera es la red generadora, la cual se encargará de aprender a generar el contenido que nosotros queremos, y la segunda es la red discriminadora, que se encargará de aprender a identificar si la imagen ha sido generada por la red generadora o si es una imagen del dataset original.

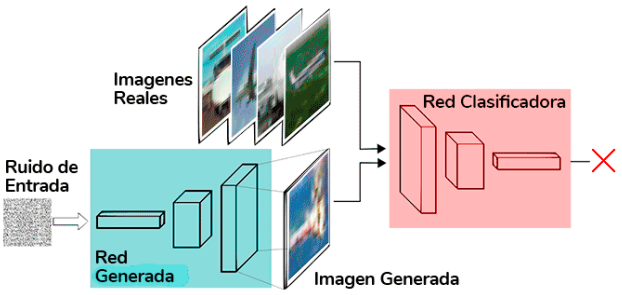


Ilustración 6. Red generada y discriminadora (Red clasificadora).

Ahora se tiene un encoder que, en términos sencillos se encarga, mediante la imagen de entrada, de comprimir la información para así, con el decoder obtener esta información comprimida y generar la imagen deseada. Un problema que puede presentar esto es que si la información se comprime demasiado luego habrá problemas para poder decodificarla, por este motivo se usa skip conection, consiste en crear conexiones entre las capas de la red para que la información pueda atajarse en ellas o incluso saltarse alguna capa.

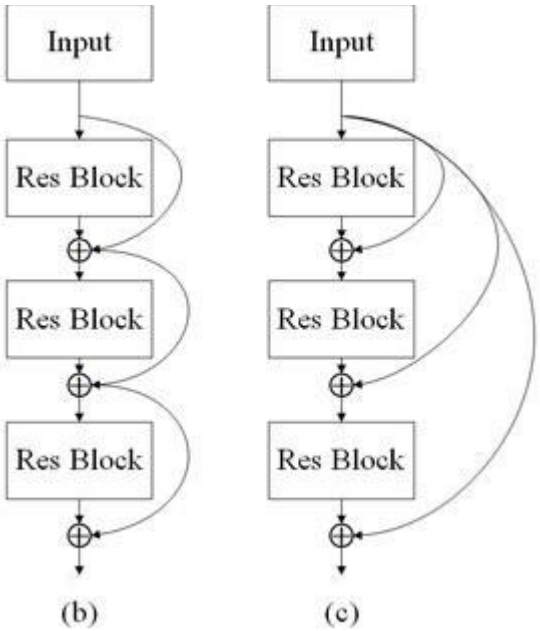


Ilustración 7. Ejemplo de skip conection.

Como se observa, no es una red perfecta sobre todo si el entrenamiento se hace durante mucho tiempo, pues entre mayor sea la documentación o el conocimiento sobre la perdida, más preciso será el modelo. Por tal motivo, se necesitan horas de *train* para obtener resultados aceptables y más de 400 imágenes en el dataset. Así, con muchas imágenes, probablemente se podría mejorar algunas imágenes generadas pero el problema sería el consumo computacional y que de carácter obligatorio es necesario el uso de una GPU.

Ahora, de acuerdo con Tensorflow, Pix2Pix no tiene una aplicación especifica, se puede aplicar a una amplia gama de tareas, incluida la síntesis de fotos a partir de mapas de etiquetas, la generación de fotos coloreadas a partir de imágenes en blanco y negro, la conversión de fotos de Google Maps en imágenes aéreas e incluso la transformación de bocetos en fotos [11].

E. Conexión con EA Sport

El equipo de FIFA 23 también se basó en la IA de aprendizaje automático que se introdujo en el juego del año pasado. Se utilizó una captura avanzada de partidos 11vs11 para ayudar a impulsar la IA para FIFA 23, ya que esto proporcionará un aumento en el rendimiento y la precisión de las redes neuronales [12].

Ilustración 8. Estadio del Chelsea: Stamford Bridge.

Según [13], desde 2022, un algoritmo de machine learning analiza alrededor de 9 millones de fotogramas de captura avanzada en partidos, para luego desarrollar animaciones que brinden más realismo orgánico a la forma en que interactúan los jugadores en el campo. Esto lo realiza mediante el algoritmo Advanced 11v11 Match Capture. En pocas palabras, el objetivo de la incorporación de inteligencia artificial en los juegos es proporcionar historias, escenarios, niveles, personajes realistas y personalización cada vez más complejos e ilimitados.

Ilustración 9. Stamford Bridge at night.

Ilustración 10. Parche de Inteligencia Artificial para NBA2K23[2].

F. Aprendizaje automático (Machine Learning o ML)

La captura avanzada de partidos 11 contra 11 de HyperMotion2 permitió a nuestro algoritmo de aprendizaje automático (ML) aprender de muchos más fotogramas de datos de los que disponíamos anteriormente para crear nuevas características. Hemos sido capaces de usar ML para aumentar el rendimiento y la precisión de nuestras redes neuronales, que pueden generar una variedad de animaciones de movimiento en tiempo real proporcionando una estructura más robusta para FIFA 23 y los futuros títulos de EA SPORTS FC. En FIFA 23, esto ha llevado a la evolución y desarrollo de varias características que están basadas en el aprendizaje automático, como bregar y los regates técnicos, que detallaremos a continuación [14].

Ilustración 11. Estadio del Boca: La Bombonera de noche.

ENUNCIADO

La guía del proyecto final del curso orientado por el docente Andrés Felipe Escobar[3], propone entrenar una red generativa para la generación de imágenes o de audio, usando un data set personalizado, donde se recomienda usar para el desarrollo, las herramientas para Deep Learning Keras y/o TensorFlow.

En aras de poder aplicar la conversión de escenas diurnas a nocturnas, que es un contexto que tiene un alto potencial en videojuegos como el FIFA de EA Sports y NBA de 2K GAMES, se realiza en primer lugar, la creación del dataset personalizado, donde se precisó una carpeta que contenía tanto imágenes de estadios de día como de noche de los clubes mas importantes de las ligas más llamativas del futbol (Ligue 1, Liga Española, Premier Leage, etc). Aquí se dividió en partes iguales la cantidad de imágenes y se le correspondió a cada club un total de 25 imágenes y así, con los 24 clubes se completaron las 600. Luego de ello, se tomaron imágenes aleatorias de Google para precisar las de validación mediante lo cual se logra adquirir 50 imágenes más, teniendo un dataset de 650 imágenes en total.

Club	Dia	Noche
AC Milán / Inter	13	12
América de Cali	12	13
Atlético de Madrid	12	13
Barcelona	12	13
Boca	12	13
Deportivo Cali	12	13
Chelsea	13	12
Flamengo	12	13
Independiente	12	13
Juventus	13	12
Liverpool	12	13
Lyon	12	13
Manchester United	12	13
Mónaco	13	12
Monterrey	12	13
Nacional	12	13
PSG	12	13

[2] Imagen tomada de: https://www.givemesport.com/88043901-nba-2k23-courtside-report-reveals-impressive-ai-gameplay-improvements

[3] Docente de la asignatura de Redes Neuronales y Deep Learning, aescobaro@uao.edu.co.

Real Madrid	12	13
River Plate	12	13
Roma	12	13
Santos	12	13
Santos Laguna	12	13
Sao Paulo	12	13
Tigres	12	13
TOTAL (+50 Validación)	292	308

Cada imagen tenía un tamaño correspondiente y eran heterogéneas entre sí, debido a ello, se utiliza el aplicativo de IloveIMG y con él, fue posible hacer un redimensionamiento de estas para que tuvieran un tamaño de 256x256. En este inciso se presentaron algunos inconvenientes, no obstante, se logró redimensionar la mayoría de ellas.

Ilustración 12. Imágenes de día y de noche de los estadios.

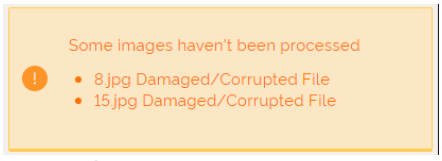


Ilustración 13. errores en el proceso.

Después de hacer una respectiva documentación, el equipo de trabajo logra darse cuenta de que la conversión de estas escenas era posible realizarlas desde la aplicación de las redes GAN, donde existen diversas arquitecturas o topologías y cada una de ellas cuenta con una ventaja o desventaja sobre las demás. Por eso, estos conceptos de delimitan en el marco teórico y con esta información, se prosigue a realizar cada uno de los siguientes incisos:

Entrenamiento del modelo de red neuronal a utilizar, donde es posible usar alguno de los múltiple tutoriales que hay en internet para tal fin.

Validación del modelo de red neuronal utilizado.

Explicación de la arquitectura de red neuronal usada.

DESARROLLO DE LA SOLUCIÓN

Para llevar a cabo la solución se hace uso de la herramienta Colaboratory de Google, mediante la cual en un Notebook utilizando Python y Tensorflow, se implementa la arquitectura Pix2Pix obtenida del tutorial Generando FLORES realistas con IA - Pix2Pix de DotCSV. La codificación realizada se encuentra en el siguiente Colab, sin embargo, a continuación, se realiza la descripción de los aspectos más relevantes del código.

Inicialmente se realiza todo lo relacionado al procesamiento de los datos de entrenamiento y validación, correspondiente a la ruta en la que se encontraran los componentes del dataset, el preprocesamiento de los datos y la definición de cuál será el target y cuál será la entrada, que para este caso específico se tiene como entrada las imágenes diurnas para que sean mapeada a imágenes nocturnas.

Posteriormente se implementa la arquitectura Pix2Pix, para la cual se consideran cada uno de los aspectos presentes en el artículo relacionado a la arquitectura [15]. En este sentido, Pix2Pix es una red Generativa Adversaria Condicionada, es decir, una red a la que se le puede condicionar la salida a partir de un input. Adicionalmente, como modelo generativo cuenta con un discriminador y un generador, en el que el generador tiene una arquitectura de tipo U-NETs, correspondiente a una red convolucional con un encoder, decoder y skip conections conectadas al mismo nivel. Por otra parte, el discriminador cuenta una arquitectura denominada Patch GAN, la cual realiza la evaluación de las imágenes por lotes de pixeles.

Ahora, a nivel de codificación haciendo uso de Tensorflow, se genera una función denominada downsample, la cual representa el tipo de capa que se utiliza en el encoder, esta función tiene como parámetros el número de filtros y si se realiza el Batch Normalization o no, debido a que según esto se agrega o no un sesgo; esta función genera 3 capas con una de convolución, otra de Batch Normalization y una de activación con una ReLU. Con estas capas se generará el embudo de cuello de botella de la red neuronal. Adicionalmente, en estas capas se debe especificar que los parámetros deben ser inicializados con un ruido gausiano de 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=initia
lizer,
                    use_bias=not apply_batchn
orm))

  if apply_batchnorm:
    result.add(BatchNormalization())

  result.add(LeakyReLU())
  return result
```

Por otro lado, para las capas del decodificador se hace uso de otra función correspondiente a upsample, en la cual se ve reflejada otro tipo de capa, en la cual se cuenta con capa Convolucional Transpuesta, un Batch Normalization y una capa de activación ReLU, adicionalmente tiene 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_initializ
er=initializer,
                             use_bias=False))

  result.add(BatchNormalization())

  if apply_dropout:

    result.add(Dropout(0.5))

  result.add(ReLU())

  return result
```

Posteriormente, se construye el generador, en el cual se realiza la unión de las diferentes capas mediante la función presente a continuación, en la que simplemente se unen las capas y se inserta el número de parámetros y las condiciones de si tendrá Batch Normalization o si tendrá el Dropout del 50%, con una capa convolución transpuesta a la salida, reflejando la arquitectura 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=i
nitializer,
                         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)
```

Con el modelo del generador ya realizado, se procede a crear el discriminador mediante una función en la que se define una red convolucional la cual recibe dos entradas, correspondiente a la imagen real y a la imagen falsa. Adicionalmente, el discriminador cuenta con una capa convolucional transpuesta de 1 solo canal, mediante el cual se logra analizar por cada píxel de la imagen, si la región a la que pertenece el pixel parece real o no.

```
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=Fals
e)(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_initia
lizer=initializer,
                                padding="same
")(down4)

  return Model(inputs=[ini, gen], outputs=las
t)
```

A pesar de contar con otros aspectos relevantes dentro del código como lo son los entrenamientos, definidos dentro de una función y la generación de las funciones de perdida tanto para el discriminador como para el generador, no se describen debido a que se haría demasiado extenso el presente inciso. De esta forma, con la arquitectura implementada en el código, se procede a realizar el entrenamiento mediante 500 épocas, las cuales, a pesar de contar con una GPU, tardan un tiempo aproximado de 2 horas. Vale la pena mencionar que en el transcurso del entrenamiento se logra observar el progreso de la red mediante un conjunto de imágenes almacenada en el directorio output. Finalmente, con el modelo entrenado se obtienen nuevas imágenes a partir de nuevos datos de entrada, las cuales se presentan y analizan como resultados en el siguiente inciso.

RESULTADOS

A partir del modelo entrenado, es posible analizar los resultados de la arquitectura Pix2Pix para la problemática planteada correspondiente al mapeo de imágenes diurnas a imágenes nocturnas, específicamente las relacionadas a los estadios, debido a su potencial aplicación en juegos como EA Sport FIFA. Como primer resultado se tiene el tiempo de entrenamiento requerido para una cantidad de 500 épocas, correspondiente a 2 horas aproximadamente, lo cual es un valor elevado considerando las limitaciones de tiempo con las que se cuenta en los espacios de desarrollo como Colab. Por otra parte, los resultados obtenidos con respecto al cambio de dominio entre imágenes, presente en la figura 14, permiten observar como a pesar de que la respuesta no alcanza a ser la deseada, algunos aspectos determinantes entre el día y la noche, como el cielo, se tornan más opacos.

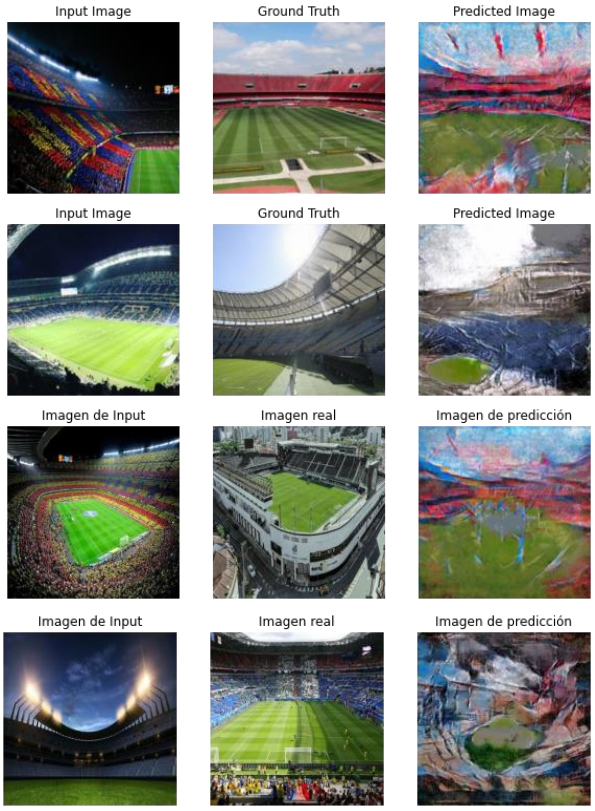


Ilustración 14. Predicción del modelo con datos de entrenamiento (2 ultimas filas) y datos de validación (2 primeras filas).

El no alcanzar la respuesta deseada se puede relacionar a dos aspectos principales correspondientes al entrenamiento y al dataset. Lo anterior, debido a que al realizar un numero limitado de entrenamientos el modelo no alcanza a entrenarse lo suficiente para el mapeo entre dominios de estas imágenes, lo cual a su vez se dificulta, ya que los estadios cuentan con un conjunto de patrones variados como el color de la grama y la silletería, los cuales varían según los estadios. Por otra parte, el contar con un dataset limitado y no emparejado para las características propias de los estadios (color, estructura y demás) se dificulta para arquitecturas como Pix2Pix, lo cual probablemente se podría haber mejorado mediante el uso de arquitecturas como CycleGAN, la cual no requiere de imágenes emparejadas. La dificultad de obtener imágenes emparejadas para esta aplicación radica en que no es posible encontrar imágenes desde el mismo ángulo y posición tanto para la noche como el día.

CONCLUSIÓN

Con la realización del presente documento, el equipo logra adquirir más experticia experimental relacionada con el manejo, manipulación y diseño de redes neuronales artificiales para la solución de problemas asociados a la vida real o aplicaciones cotidianas, siendo en este caso, la forma en la que, mediante una red GAN, se logra en lo que cabe la palabra, estimar como seria las escenas diurnas a escenas nocturnas de los estadios de fútbol.

Inicialmente, existieron ciertos factores que se debieron tener en cuenta durante el proceso para que no fuese solo "hacer por hacer" sino que se desarrollara de la forma más correcta. Uno de ellos, de vital importancia, fue el entendimiento del dataset seleccionado; aunque se lea como un factor no determinante, realmente el hecho de definir la cantidad de imágenes por cada club, revisar su tamaño, dejarlas todas en formato JPG, permitió que se tuviera un panorama más amplio de lo que es el contexto de la aplicación de los modelos generativos y la extracción de características estáticas de los estadios pues en el día y en la noche influyen diferentes factores que los hacen lucir de manera diferente.

Luego de ello y de redimensionar todas las fotos para tener unanimidad, se destaca el proceso de búsqueda de las diferentes arquitecturas pues con esa indagación, es la forma en la que se tienen en cuenta cuales son las ventajas, desventajas y repercusiones que se deben tener en cuenta para aprender no solamente a montar una red neuronal, sino interpretar y validar lo que realmente se está haciendo. Así, al usar Pix2Pix, se tiene presente que el haber dispuesto más de XXX Épocas y haber entrenado aún más el modelo, tener paridad en las imágenes y una mayor capacidad de cómputo, hubiera permitido que se obtuvieran a la salida, imágenes casi que reales las cuales tienen un alto potencial de aplicación en los videojuegos o en si, en aplicativos que involucren IA.

Sobre los resultados obtenidos, se podría decir que no son los esperados, sin embargo, se puede observar cómo aspectos relevantes como las tonalidades del cielo toman un color opaco, lo cual evidencia que es una problemática que se puede resolver mediante un mayor entrenamiento, un dataset más elaborado junto con la posible inmersión en otras arquitecturas como Cycle GAN. No obstante, es una aplicación con una gran cantidad de retos debido a las geometrías variadas con las que cuentan los estadios, los colores de la silletería, así como el uso de iluminarias de alta potencia que imitan artificialmente la iluminación del día en la cancha.

Por tal motivo, fue inspirador poder realizar todo el entregable y adquirir conocimientos relacionados sobre cómo abordar problemas de Generación de Imágenes con Machine Learning, pues son problemas que se caracterizan en que la variable de entrada y de salida corresponden a imágenes con paridad o sin ellas y los modelos deben de ser capaz, teniendo en cuenta sus arquitecturas convolucionales o superficiales (por no decir Densas), de recrear a partir de un *Ground Thruth* que, la imagen generada, con tal precisión, que al ojo humano, sea casi que creíble que corresponde a una fotografía real o que no se pueda distinguir si en realidad lo es, siendo esto uno de los pilares fundamentales en la creación de dinámicas para diferentes videojuegos o aplicativos móviles donde se busca brindarle al usuario, una mayor experiencia y afinidad.

REFERENCIAS

[1] "7 Aplicaciones de la Inteligencia Artificial en la Ingeniería Industrial". https://www.edsrobotics.com/blog/aplicaciones-inteligencia-artificial-en-ingenieria-industrial/ (consultado sep. 02, 2022).

[2] "Machine Learning: definición, funcionamiento, usos". https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos (consultado sep. 02, 2022).

[3] "What is generative adversarial network (GAN)? - Definition from WhatIs.com". https://www.techtarget.com/searchenterpriseai/definition/generative-adversarial-network-GAN (consultado nov. 13, 2022).

[4] "Ventajas de las GAN - Proyectos de Redes Generativas Adversariales [Libro]". https://www.oreilly.com/library/view/generative-adversarial-networks/9781789136678/4e0a40ae-358c-44a5-b360-7e87c61813b3.xhtml (consultado nov. 15, 2022).

[5] "Image-to-Image Translation. Image-to-image translation is a class… | by Yongfu Hao | Towards Data Science". https://towardsdatascience.com/image-to-image-translation-69c10c18f6ff (consultado nov. 15, 2022).

[6] "Pix2pix: Decisiones clave sobre la arquitectura del modelo - neptune.ai". https://neptune.ai/blog/pix2pix-key-model-architecture-decisions (consultado nov. 15, 2022).

[7] "6 GAN Architectures You Really Should Know - neptune.ai". https://neptune.ai/blog/6-gan-architectures (consultado nov. 15, 2022).

[8] J. Y. Zhu, T. Park, P. Isola, y A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2242–2251, mar. 2017, doi: 10.48550/arxiv.1703.10593.

[9] M. Y. Liu, T. Breuel, y J. Kautz, "Unsupervised Image-to-Image Translation Networks", *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 701–709, mar. 2017, doi: 10.48550/arxiv.1703.00848.

[10] "Modelo Pix2Pix (Redes neuronales) | by Anayara Moreno Merino | Medium". https://anayaraulpgc.medium.com/modelo-pix2pix-redes-neuronales-dcc47090030f (consultado nov. 17, 2022).

[11] "pix2pix: traducción de imagen a imagen con una GAN condicional | TensorFlow Core". https://www.tensorflow.org/tutorials/generative/pix2pix (consultado nov. 17, 2022).

[12] "EA redobla la apuesta con la tecnología HyperMotion2 en FIFA 23 - Hardcore Gamer". https://hardcoregamer.com/news/ea-doubles-down-with-hypermotion2-technology-in-fifa-23/425600/ (consultado nov. 15, 2022).

[13] "FIFA 23 incorpora inteligencia artificial para más realismo y sorprendentes habilidades | 442". https://442.perfil.com/noticias/futbol/fifa-23-incorpora-inteligencia-artificial-para-mas-realismo-y-sorprendentes-habilidades.phtml (consultado nov. 15, 2022).

[14] "FIFA 23 - Análisis detallado de las mecánicas de juego - EA SPORTS". https://www.ea.com/es-es/games/fifa/fifa-21/news/pitch-notes-fifa-23-gameplay-deep-dive (consultado nov. 15, 2022).

[15] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, y B. A. Research, "Image-to-Image Translation with Conditional Adversarial Networks", Consultado: nov. 17, 2022. [En línea]. Available: https://github.com/phillipi/pix2pix.