UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

RECONOCIMIENTO DE ENTORNOS MEDIANTE EL USO DE REDES GENERATIVAS ADVERSARIAS (GANS)

TRABAJO FIN DE GRADO

Febrero - 2022

AUTOR: Néstor Gómez López

DIRECTORES: Mónica Ballesta Galdeano

Orlando José Céspedes Gómez

Agradecimientos

A todas las personas que me he encontrado en este camino, mis profesores y compañeros, los que se han quedado y los que no, por lo que he aprendido de todas ellas, a mis amigos por hacerme la vida más amena y llevadera y a mi familia por su enorme apoyo y esfuerzo, muchísimas gracias.

Índice

Índice de figuras

1 Introducción

Algo que las personas consideramos tan simple y natural como caminar por un lugar, es una de las tareas más complejas y exigentes que se puede encomendar a una máquina, debido a la infinidad de obstáculos, la complejidad de los entornos reales, así como a condiciones constantemente cambiantes que pueden complicar el reconocimiento de un área, como cambios en la iluminación, cambios en la vegetación en las diferentes estaciones del año, obras que alteren el paisaje urbano, etc.

De esta manera, se requieren una gran cantidad de imágenes con variaciones de un mismo lugar, para que el entrenamiento de una red neuronal que pueda reconocer las características básicas que lo caracterizan sea exitoso.

Centrándonos en la iluminación para simplificar, el objetivo principal de este trabajo es la generación de imágenes con variaciones de iluminación simuladas a partir imágenes reales con el objeto de generar grandes cantidades de datos sin esfuerzo que posteriormente nos permitan entrenar sistemas de reconocimiento visual precisos, para lo que se propone el uso de redes generativas adversarias (GAN).

1.1 Objetivos

El objetivo de este proyecto es encontrar y entrenar un modelo adecuado de red GAN [1] para que sea capaz de transformar imágenes de determinados lugares con un nivel de iluminación a imágenes del mismo lugar con una iluminación distinta siendo reconocible el lugar de la imagen original.

2 Conceptos previos

El marco teórico bajo el que se engloba nuestro trabajo es el campo del machine learning por lo que es necesario comentar este ámbito.

2.1 Machine Learning

El **aprendizaje automático** o *machine learning* es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras *aprendan* [2]. *Aprender* en este contexto quiere decir identificar patrones complejos en millones de datos. **La máquina que realmente aprende es un algoritmo** que revisa los datos y es capaz de predecir comportamientos futuros. *Automáticamente*, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana [3].

El aprendizaje automático tiene una amplia gama de aplicaciones desde el reconocimiento de elementos en imágenes como por ejemplo radiografías que permiten un diagnóstico temprano y preciso de ciertas enfermedades, hasta el reconocimiento de las preferencias de búsqueda de un usuario al navegar por internet.

De todas las aproximaciones al machine learning existentes, en este estudio nos centraremos en las redes neuronales, puesto que constituyen la base de las redes GAN.

2.2 Redes neuronales

Una **red neuronal** es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal: una **capa de entrada**, con unidades que representan los campos de entrada; una o varias **capas ocultas**; y una **capa de salida**, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o **ponderaciones**). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente. Al final, se envía un resultado desde la capa de salida.

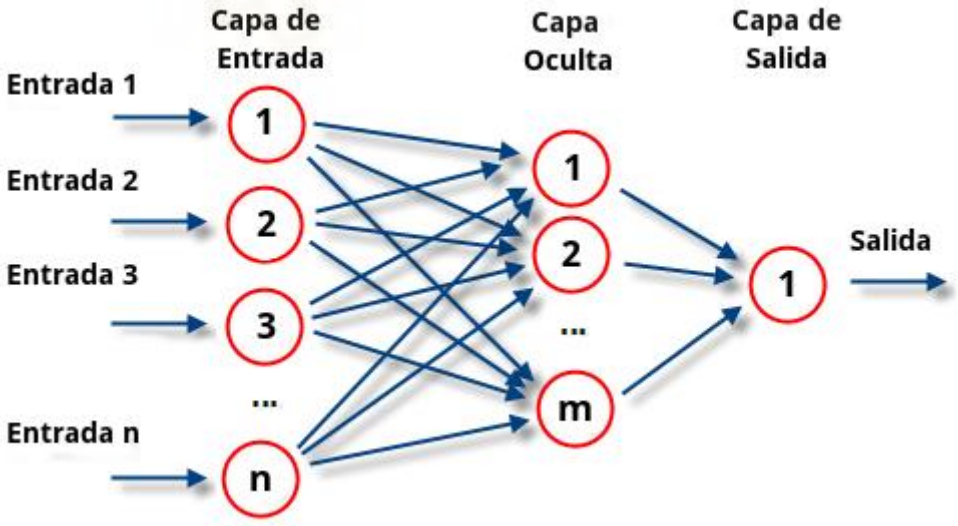


Figura 1: Esquema de una red neuronal.

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción

incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. La red aprende a través del **entrenamiento**. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, y las respuestas que proporciona se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, cambiando las ponderaciones gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado [4].

2.3 Reconocimiento de imágenes y redes convolucionales

Las Redes neuronales convolucionales son un tipo de redes neuronales artificiales donde las "neuronas" corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptrón multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones [5].

Figura 2: Sistemas de conducción autónoma integran el reconocimiento de imágenes para detectar carriles y señales de tráfico.

Por su capacidad de especializarse en aplicaciones que utilizan reconocimiento de imágenes como entrenamiento son el tipo de redes que utilizaremos por lo que debemos conocer su funcionamiento.

REDES CONVOLUCIONALES

Principalmente lo que diferencia a una red convolucional respecto de una red neuronal tradicional son las capas de convolución y las capas de pooling, estas capas sirven para poder extraer características de una imagen automáticamente, cuando normalmente habría que aplicar filtros para indicar a la red que es lo que está viendo (por ejemplo resaltar las líneas de una carretera), por lo que de esta manera se ahorra mucho tiempo y esfuerzo de procesado de las bases de datos, tras estas capas, las redes convolucionales se estructuran como una red convencional de capas totalmente conectadas, que finalmente utilizan las características halladas automáticamente para clasificar la imagen.

Las capas de convolución tienen la función de extraer las características de la imagen tomando grupos de píxeles cercanos y operando contra una pequeña matriz llamada kernel cuya función es generar una matriz de salida que será la nueva capa de neuronas ocultas.

Ese kernel supongamos que tiene un tamaño de 3×3 píxeles y con ese tamaño logra «visualizar» todas las neuronas de entrada que contienen el valor de cada píxel, normalmente va de 0 a 255 pero para la red se normaliza de 0 a 1 (de izquierda-derecha, de arriba-abajo) y así logra generar una nueva matriz de salida, que en definitiva será nuestra nueva capa de neuronas ocultas.

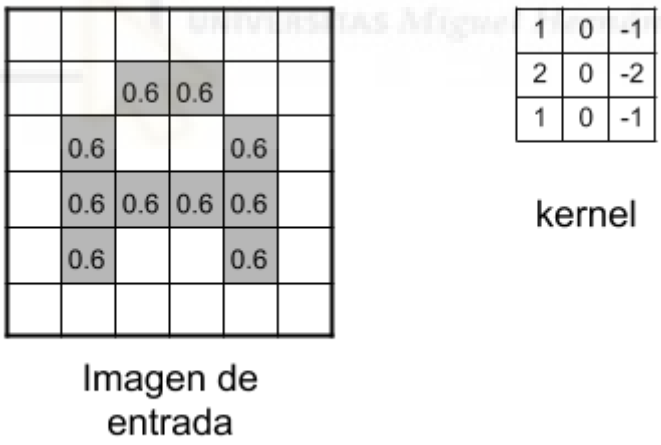


Figura 3: Imagen de entrada y kernel 3X3.

No aplicaremos 1 sólo kernel, si no que tendremos muchos kernel (al conjunto de Kernels se les llama filtros). Imaginen cuántas más serían si tomáramos una imagen de entrada de 224x224x3 (que aún es considerado un tamaño pequeño) …

A medida que vamos desplazando el kernel y vamos obteniendo una «nueva imagen» filtrada por el kernel. En esta primera convolución y siguiendo con el ejemplo anterior, es como si obtuviéramos 32 «imágenes filtradas nuevas». Estas imágenes nuevas lo que

están «dibujando» son ciertas características de la imagen original. Esto ayudará en el futuro a poder distinguir un objeto de otro (por ej. gato o un perro).

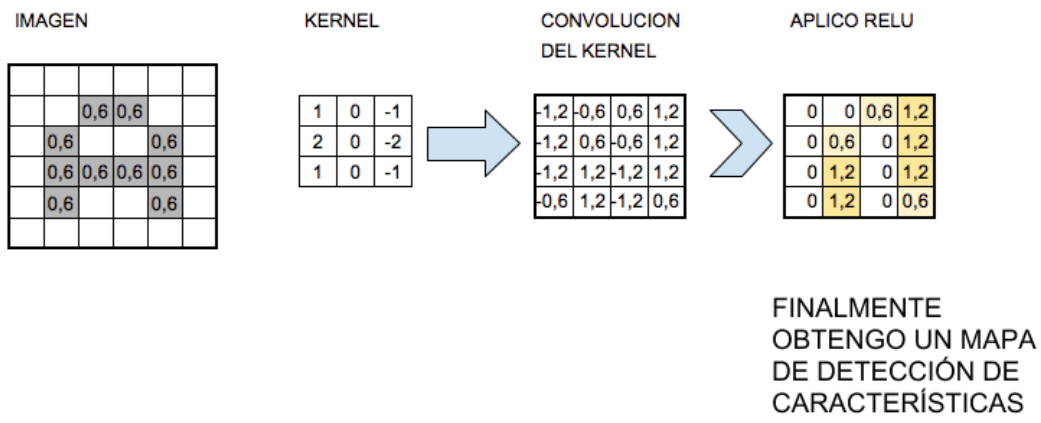


Figura 4: Mapa de características.

Si hiciéramos una nueva convolución a partir de esta capa, el número de neuronas de la próxima capa requeriría un poder computacional importante. Por ello y para reducir el tamaño de la próxima capa de neuronas **hacemos un muestreo preservando las características más importantes que detectó cada filtro**, el pooling es el método utilizado para este muestreo.

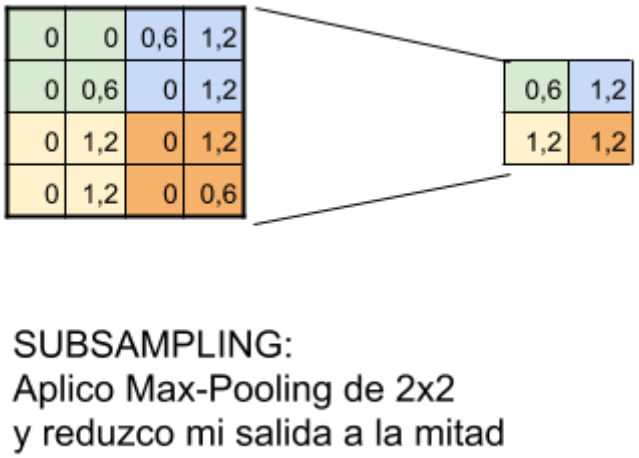


Figura 5: Funcionamiento del pooling.

El paso siguiente sería utilizar la técnica de “Max-pooling” con un tamaño de 2×2. Esto quiere decir que recorreremos cada una de las 32 imágenes de características obtenidas anteriormente de 28x28px de izquierda-derecha, arriba-abajo, PERO en vez de tomar de a 1 píxel, tomaremos de «2×2» (2 de alto por 2 de ancho = 4 píxeles) e iremos preservando el valor «más alto» de entre esos 4 píxeles (por eso lo de «Max»). En este caso, usando 2×2, la imagen resultante es reducida «a la mitad» y quedará de 14×14 píxeles. Luego de este proceso de submuestreo nos quedarán 32 imágenes de 14×14, pasando de 25.088 neuronas a 6272, las cuales son bastantes menos y que -en teoría- deberían seguir almacenando la información más importante para detectar características deseadas.

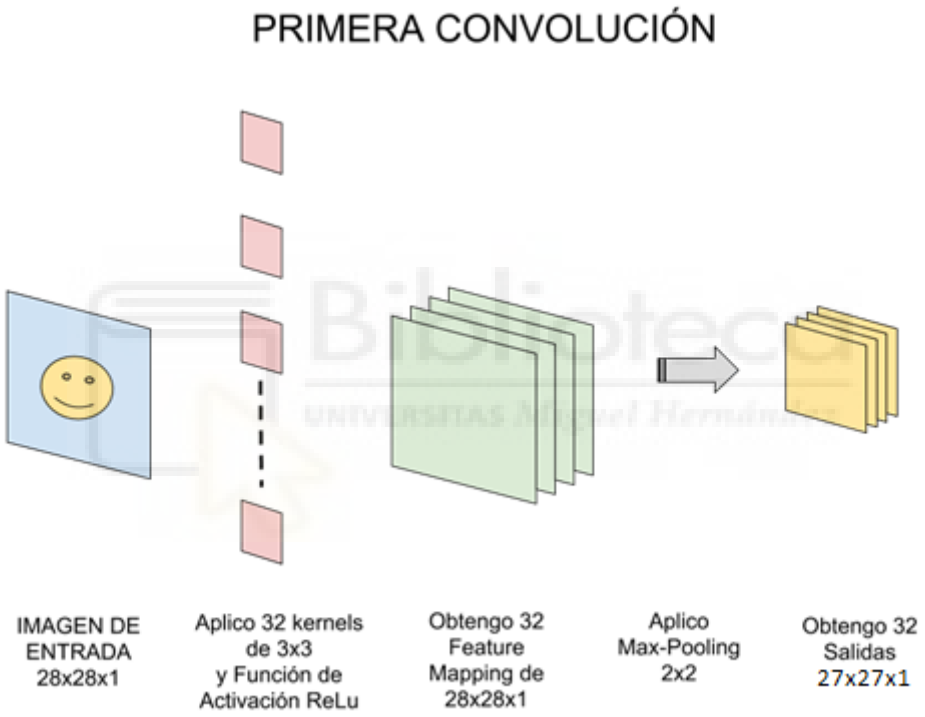


Figura 6: Esquema convolución 1.

La figura 6 representa esa primera convolución: consiste en una entrada, un conjunto de filtros, generamos un mapa de características y hacemos el submuestreo.

La primera convolución es capaz de detectar características primitivas como líneas o curvas. **A medida que hagamos más capas con las convoluciones, los mapas de características serán capaces de reconocer formas más complejas**, y el conjunto total de capas de convoluciones podrá «reconocer»

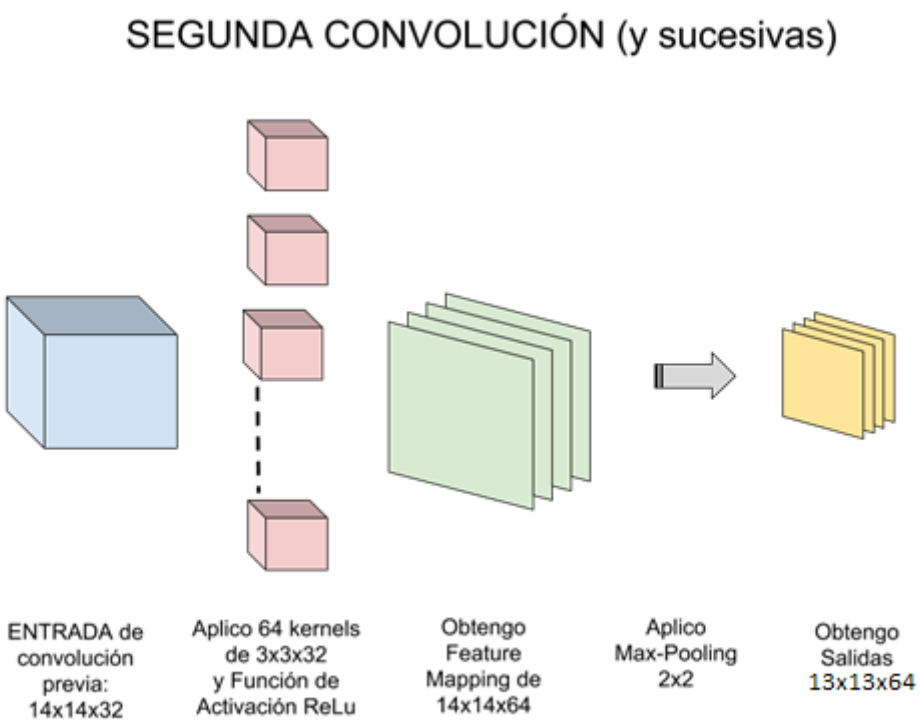


Figura 7: Esquema convolución 2... n.

Tras realizar todas las convoluciones posibles se conectan las salidas con una red neuronal convencional, como se aprecia en la figura 8 [6]:

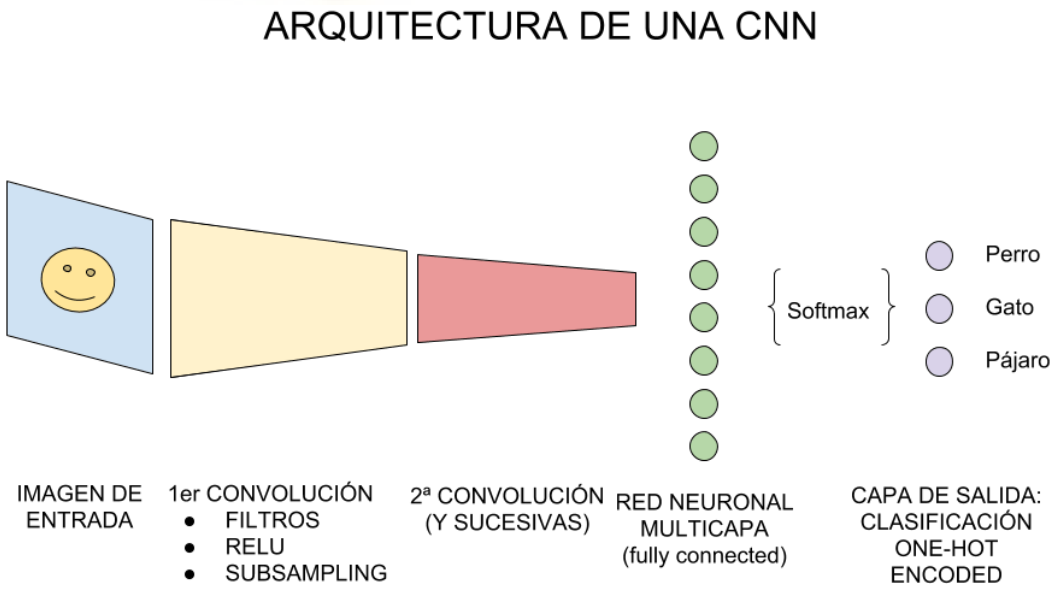


Figura 8: Arquitectura red CNN.

REDES TOTALMENTE CONVOLUCIONALES Y DECONVOLUCIONALES

Antes de introducir las redes GAN debemos conocer estas dos derivaciones de las CNN debido a que son las redes que conforman la estructura de las redes GAN.

Estas redes actúan como encoders y decoders convolucionales formados por las capas propias de las redes CNN que ya hemos visto. En concreto, para el caso que nos ocupa que es el tratamiento de imágenes, el encoder tiene la función de reducir la dimensión de la imagen de entrada a un vector a partir del cual el decoder sea capaz de reconstruir la imagen,

Comúnmente este tipo de redes se entrenan con grupos de imágenes emparejadas en 2 dominios, de manera que uno de ellos es el dominio de entrada mientras que el otro es el dominio objetivo de manera que tras recibir la entrada y convertirla al dominio objetivo esta se compara con la imagen real del dominio objetivo, de esta manera se obtiene la retroalimentación para el entrenamiento, esto se puede representar en una gráfica con el valor de *accuracy* lo que nos permite monitorizar el entrenamiento para encontrar una convergencia y tener la certeza de que hemos alcanzado un punto de entrenamiento óptimo a partir del cual, si continuamos con el entrenamiento no obtendremos ninguna mejora.

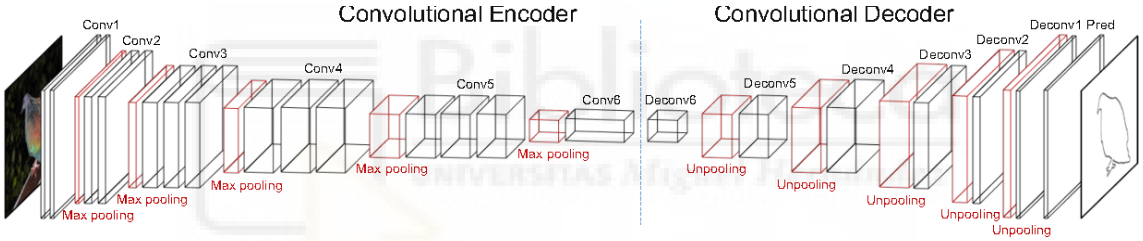


Figura 9: Estructura encoder-decoder.

Una de las aplicaciones para las que se utiliza esta estructura es la segmentación de imágenes [7]. En este caso el entrenamiento se realiza utilizando las imágenes reales y simplificaciones de estas segmentadas dividiendo la imagen en elementos reconocibles mediante otro algoritmo o a mano, como se puede observar en la figura 10.

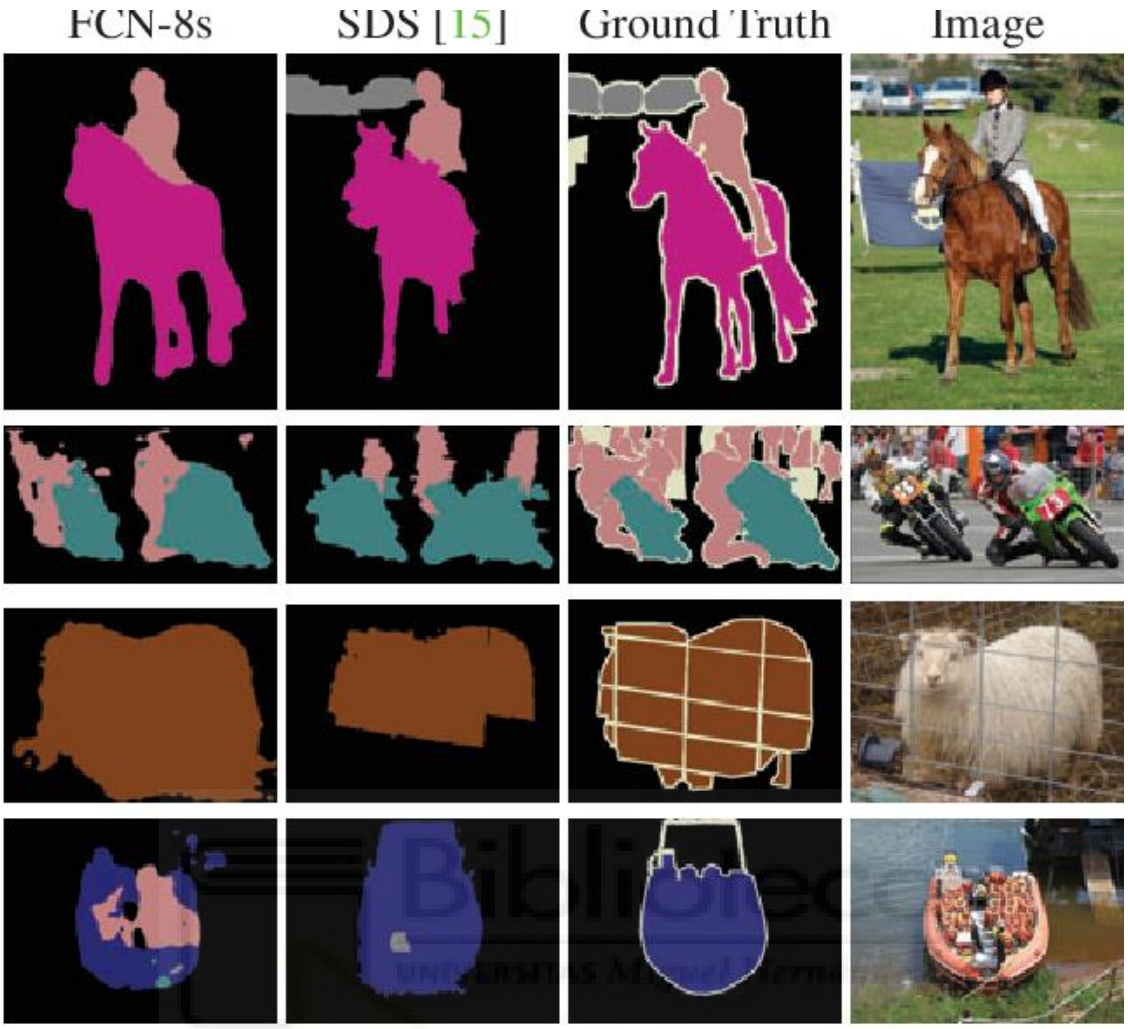


Figura 10: Segmentación mediante FCN.

3 Material y métodos

3.1 Base de datos empleada

Para realizar el entrenamiento de nuestra red se decidió utilizar 3 tipos de imágenes según la iluminación, imágenes de día con cielo despejado, con el cielo nublado e imágenes de noche, para ello se seleccionaron imágenes de estos 3 grupos de la base de datos de COLD [8] la cual consiste en 3 datasets, tomados en entornos interiores de laboratorio de 3 ciudades europeas distintas, el *Autonomus intellingent Systems Laboratory* en la universidad de Freiburg en Alemania, el *Visual Cognitive Systems Laboratory* de la universidad de Ljubljana en Eslovenia y el *Language Technology Laboratory* del centro alemán de investigación de inteligencia artificial de la ciudad de Saarbrücken en Alemania. La base de datos contiene secuencias de imágenes de cámaras convencionales y omnidireccionales junto con escáneres laser y datos de

odometría, no obstante, para este estudio solo utilizaremos las secuencias de imágenes tomadas con las cámaras convencionales ya que el resto de los datos queda en principio fuera de este ámbito [9].

Para realizar el entrenamiento se ha utilizado selección de imágenes de un solo recorrido de cada ubicación en los 3 niveles de iluminación, para Freiburg la parte A del recorrido 1 (Figura 11), para Ljubljana la parte A del recorrido 1 (Figura 12) y para Saarbrücken la parte B del recorrido 3 (Figura 13).

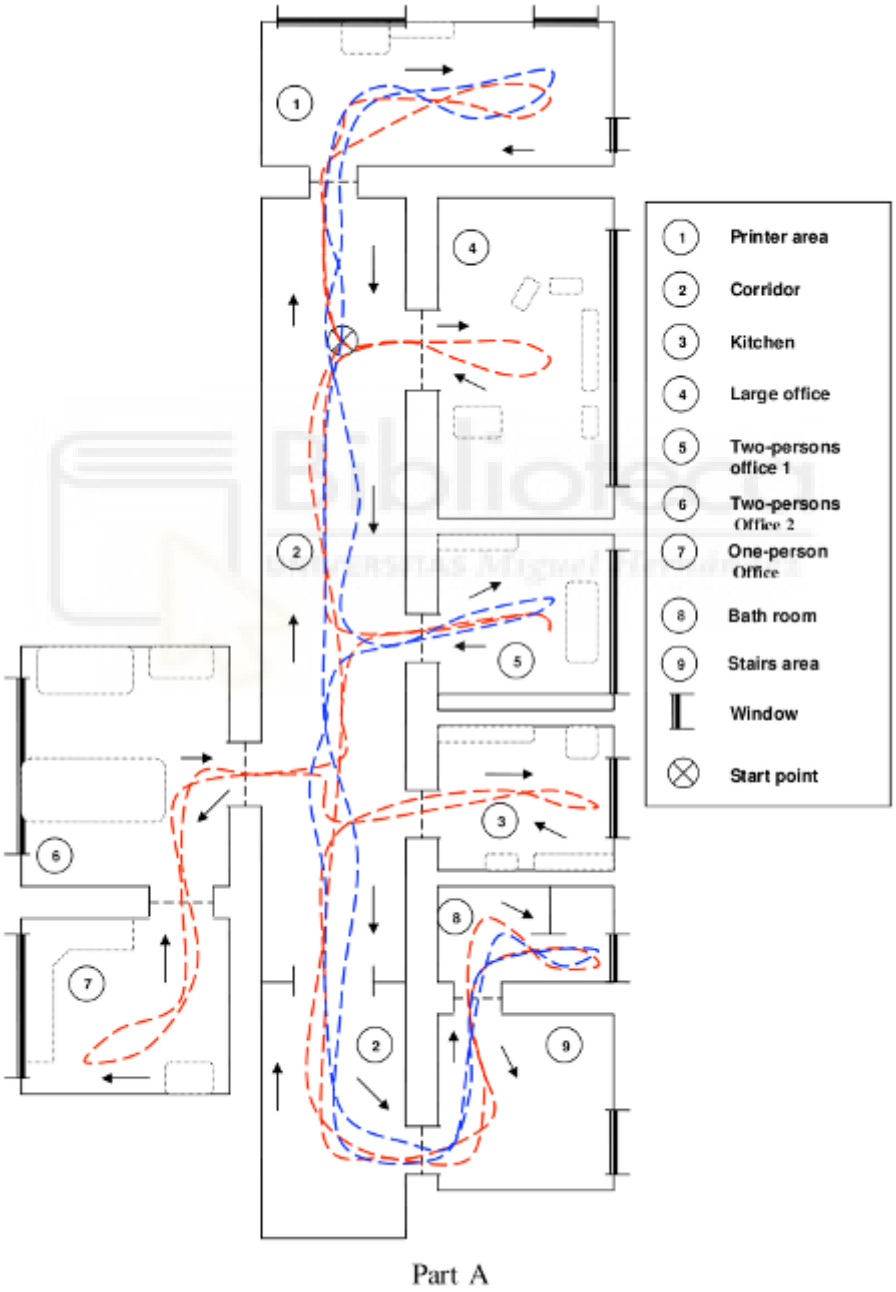


Figura 11: Freiburg parte A.

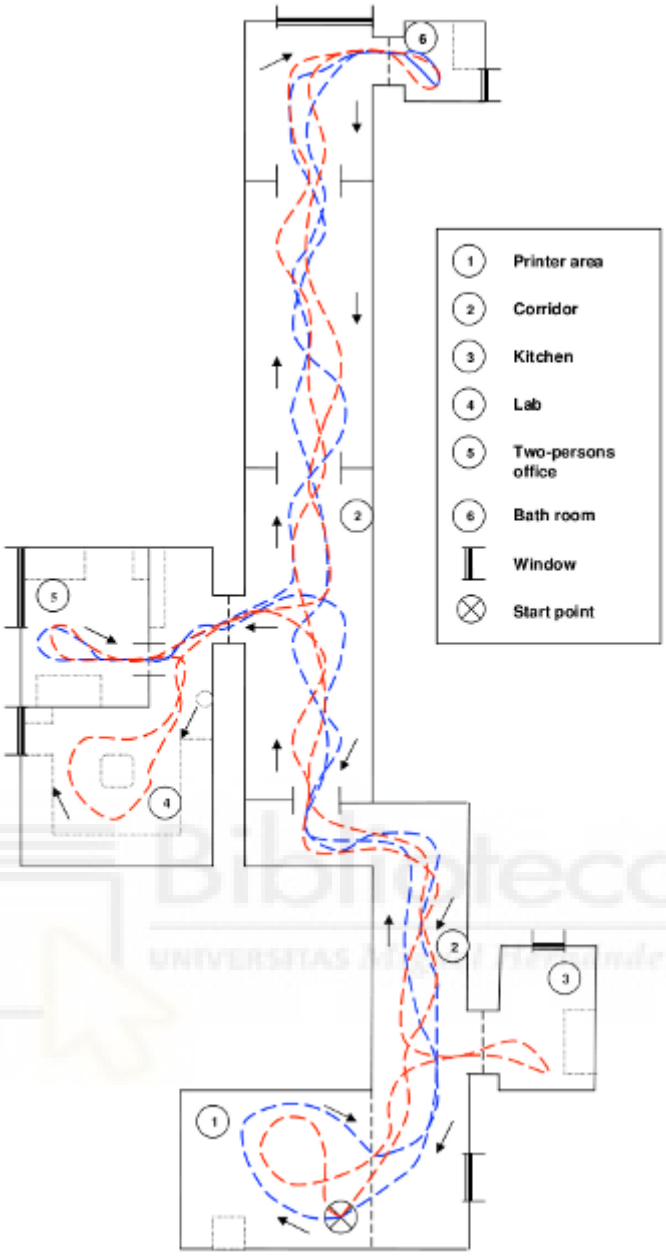


Figura 12: Ljubljana parte A.

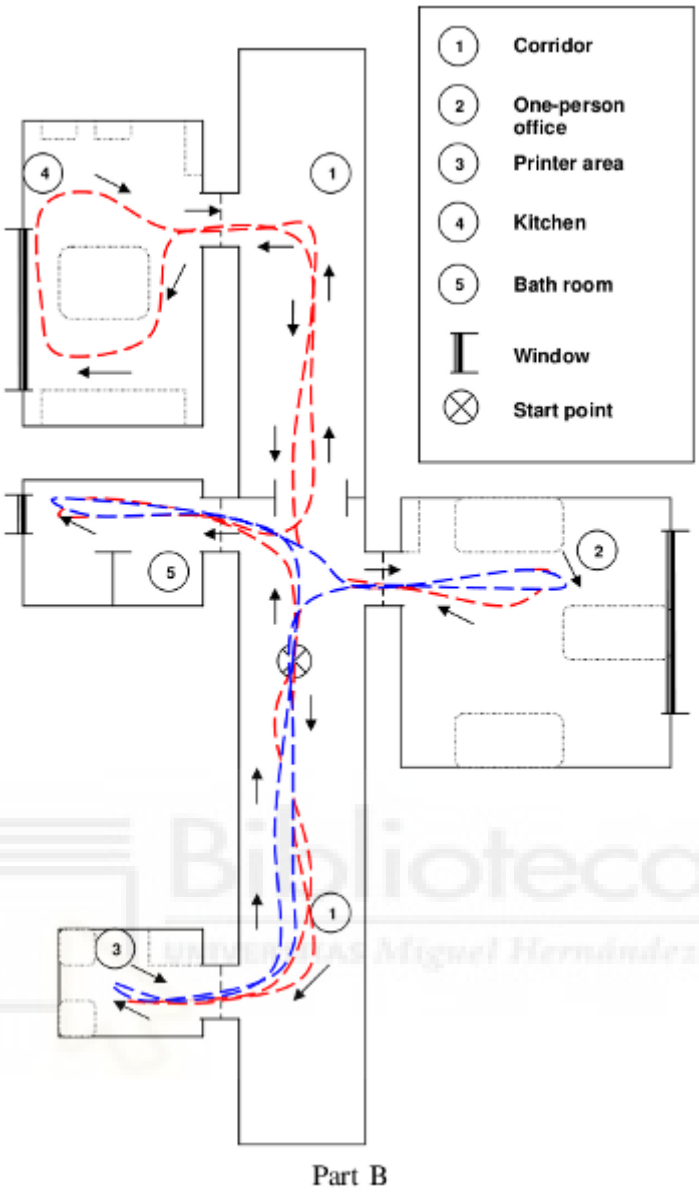


Figura 13: Saarbrücken parte B.

La preparación de la base de datos para el entrenamiento consistió en agrupar las imágenes de los 3 datasets y clasificarlos únicamente por iluminación, quedando unas 2000 imágenes por dominio (4000 por modelo), siendo una cantidad excesiva que habría supuesto un tiempo por época para el entrenamiento demasiado largo dado el hardware disponible, por lo que se realizó una selección quedando finalmente unas 1200 imágenes por dominio.

Posteriormente se buscó una base de datos de imágenes en exteriores con el objeto de probar el desempeño del modelo en entornos más variados, tras evaluar la posibilidad de utilizar un training set de ADEK20 [10] y DAGS [11] se decidió realizar una desde cero, ya que ninguno de estos dos estaba organizado según la iluminación y utilizarlos requería clasificar miles de imágenes manualmente, por lo que se tomaron unas 1257 fotografías de un entorno urbano, 738 de día y 519 de noche.

A parte de la clasificación y la selección no se realizó un reescalado ni ningún procesado extra de las imágenes ya que el programa de entrenamiento realiza este reescalado conforme recibe las imágenes.

3.2 Redes GAN

Después de haber entrado en contexto con un repaso de las tecnologías previas, podemos comentar las redes GAN o redes generativas adversarias [12] que son el objeto de nuestro estudio.

Estas redes permiten generar distintos tipos de datos nuevos del dominio de entrada, en ocasiones indistinguibles de los reales con los que haya sido entrenada, como podrían ser imágenes o música. Por ejemplo, podemos entrenar una red con miles de fotos de personas y una vez finalizado el entrenamiento esta red sería capaz de generar imágenes de personas que no existen pero que nos parecerían totalmente reales.

Figura 14: Imágenes de personas que no existen, generadas por red GAN de NVIDIA.

Las redes GAN son en realidad dos redes neuronales, originalmente una convolucional y otra deconvolucional, que en principio se correspondían con una discriminadora y una generadora respectivamente, pero desde que existen este tipo de redes se han desarrollado multitud de variantes que consisten en variaciones de esta estructura para adaptar las GAN a diferentes aplicaciones.

En esta estructura el propósito de la red generadora es, por ejemplo, el de crear nuevas imágenes a partir de datos aleatorios (ruido), mientras que la red discriminadora se encarga de juzgar si la imagen que recibe es una imagen creada por el generador o si es una imagen real del dominio del entrenamiento. Por esto reciben el nombre de redes antagónicas o adversarias, ya que durante el entrenamiento la red discriminadora intenta aprender a distinguir las imágenes reales de las que genera la red generadora, mientas que esta a su vez intenta conseguir generar imágenes que sean indistinguibles de las reales para “engañar” a la discriminadora. De esta manera al principio la red generadora generará imágenes que serán prácticamente ruido y que la red

discriminadora no tendrá problema en distinguir como falsas, pero conforme se vayan sucediendo las iteraciones con el objetivo de incrementar el error de la discriminadora, la red generadora comenzará a reproducir rasgos y características del dominio de entrada aumentando poco a poco el nivel de fidelidad hasta que finalmente el discriminador no sea capaz de distinguirlas.

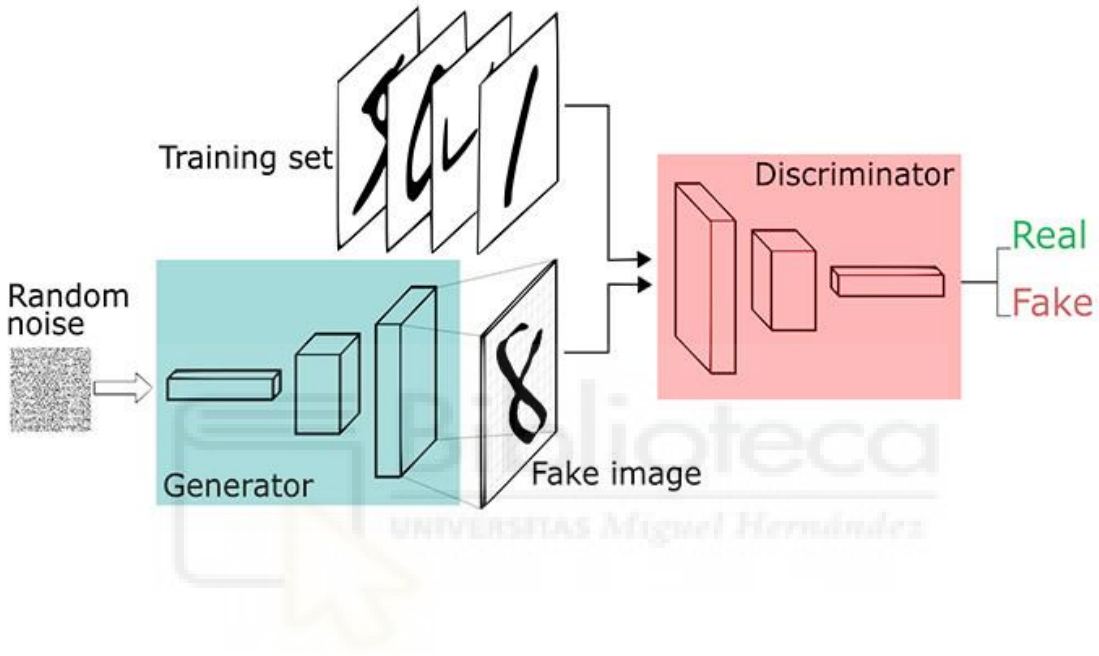


Figura 15: Estructura de una red GAN.

3.2.1 Alternativas valoradas

A continuación, explicaremos brevemente el funcionamiento de varios tipos de redes generativas adversarias que se estudiaron para este proyecto pero que finalmente fueron descartadas en favor de las cycleGAN.

CGAN

La *contitional GAN* o CGAN [13] es una extensión de las redes GAN que añade información extra al discriminador y al discriminador para convertir la red en un modelo condicional. Esta información auxiliar suele ser etiquetas de clase, pero podría introducirse otro tipo de datos.

Para entender su funcionamiento pondremos un ejemplo [14]. Imaginemos que queremos entrenar un modelo para la generación de imágenes de dígitos utilizando el

dataset MNIST. La información *y* sería el propio número que queremos generar que podríamos codificar en un vector. Por ejemplo, si quisiéramos generar un 2, el vector *y* sería (0,0,1,0,0,0,0,0,0,0). Proporcionando esta información al generador y al discriminador durante el entrenamiento y utilizando también esa información extra de las imágenes que utilizamos para entrenar, conseguimos poder generar una vez entrenado y con un solo modelo todos los dígitos. Para poder llevar a cabo la tarea anterior en una GAN tradicional, deberíamos entrenar un modelo por cada dígito.

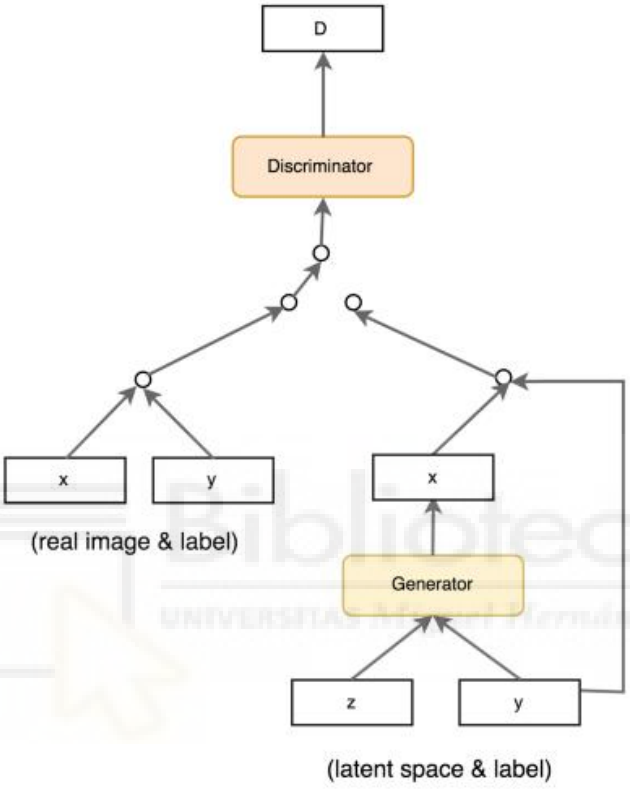


Figura 16: Arquitectura de CGan.

DiscoGAN

La arquitectura de la red DiscoGAN tiene de nuevo el objetivo de, como la CycleGAN, descubrir relaciones entre 2 dominios a través de dos conjuntos de datos no emparejados para ser capaz de transformar una imagen de un dominio de entrada a otra del de salida manteniendo la identidad de la imagen original [15].

De cara a conseguir esta traducción entre dominios se basa también en la reconstrucción de la imagen original para minimizar la pérdida de identidad, pero aun siguiendo el mismo principio que la CycleGAN utiliza otra arquitectura de red diferente que combina dos redes para conseguir este objetivo.

IcGAN

La *Invertible Conditional* GAN [12] es una derivación de la CGAN que tiene como objetivo permitir la modificación de características deliberada presente en dicha arquitectura,

pero sobre imágenes existentes. Para conseguirlo, introduce dos encoders (figura 17) que serán entrenados previamente y tendrán como objetivo “comprimir" la imagen de entrada a una representación latente y codificar las características presentes en la imagen respectivamente [16].

La imagen comprimida se utilizará en la CGAN como el vector de ruido z, y el vector de características extradido actuaría como la información adicional y, pudiendo ser modificado para por ejemplo añadir gafas de sol o cambiar el color del pelo a una persona si utilizásemos el dataseet CelebA como podemos ver en la figura 18.

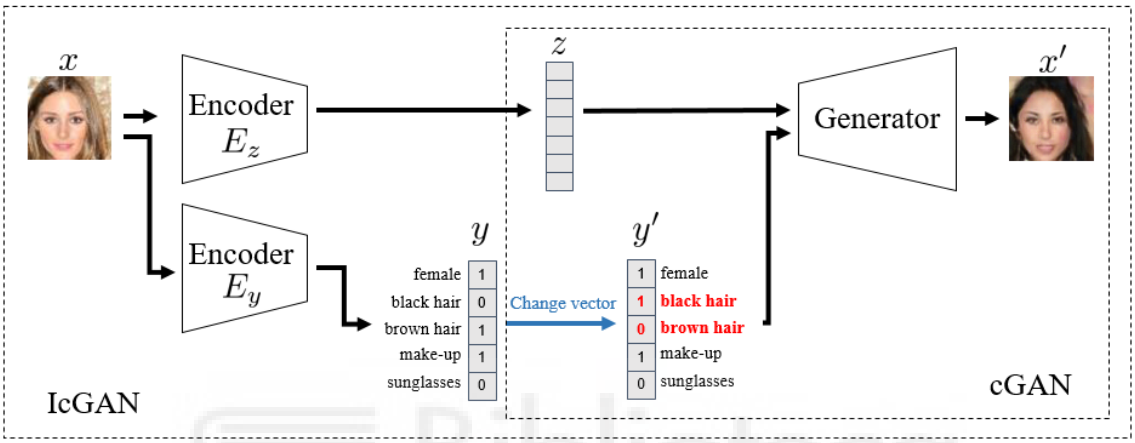


Figura 17: Arquitectura de IcGAN [17].

Figura 18: IcGAN aplicada a CelebA [17].

3.2.2 CycleGAN

Una vez conceptualizado el funcionamiento de las redes GAN y como se componen, pasaremos a tratar la CycleGAN que por los motivos que concluiremos en el siguiente punto es la que hemos escogido para el desarrollo de este estudio.

La CycleGAN [18] es un tipo de red GAN cuyo objetivo es transformar una imagen de un dominio X en otra de dominio Y que, entre otras, tiene la ventaja de no requerir un entrenamiento basado en conjuntos de imágenes emparejadas (a cada imagen X le tiene que corresponder otra de dominio Y similar cambiando el menor número de características de ese dominio excepto las que definen dicho dominio) como sucede con las redes completamente convolucionales. Esto es posible debido a que esta red introduce una transformación extra en el entrenamiento, que después de realizar la transformación del dominio X al dominio Y se vuelve a transformar el resultado al dominio original (X), de esta manera se puede volver a comparar la imagen reconstruida con la original, así se genera el *cycle consistency loss*, proceso que al reconstruir la imagen y compararla con la original, permite evitar la pérdida de identidad de esta.

Estructura y funcionamiento

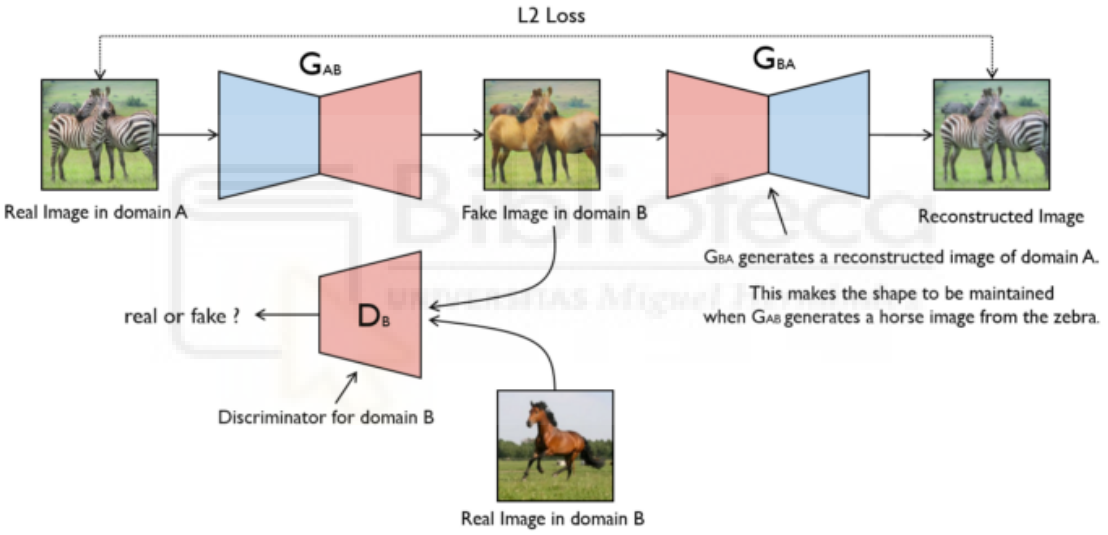


Figura 19: Esquema CycleGAN [19].

Como se puede observar en la figura 19 la red está formada por 2 generadores y 2 discriminadores, el primer generador convierte imágenes del dominio A al dominio B, mientras que el segundo las convierte de B a A.

A cada generador le corresponde por tanto un discriminador que valore si las imágenes son reales o generadas que proporciona la información necesaria para el entrenamiento, tal como sucedería en una red convolucional, pero en este caso, eso no es suficiente para pasar de un dominio a otro, puesto que

no garantiza que la imagen convertida conserve la identidad de la original. Por ejemplo, si entrenamos una red convencional para que convierta imágenes de un paisaje soleado, en el mismo paisaje nublado, la imagen obtenida no correspondería con el paisaje original, puesto que el generador simplemente tendería a transformar la imagen a una

que el discriminador clasifique como real, pudiendo generar un paisaje cualquiera que no se corresponda con el inicial en la mayoría de los casos, ya que el generador solo tiene que conseguir una imagen de un lugar nublado para que el discriminador la de por real, de manera que cumpliría su objetivo sin llevar a cabo la conversión, simplemente inventando un nuevo lugar.

En las redes cycleGAN esto se soluciona mediante un proceso especifico de entrenamiento para este tipo de redes como es el *cycle consistency loss.* El *cycle consistency loss* intenta reforzar la consistencia de la identidad de las imágenes transformadas del dominio X al Y por el generador y, volviendo a transformar el resultado en una imagen del dominio X usando el generador X [20], de manera que se parezca lo máximo posible a la imagen original.

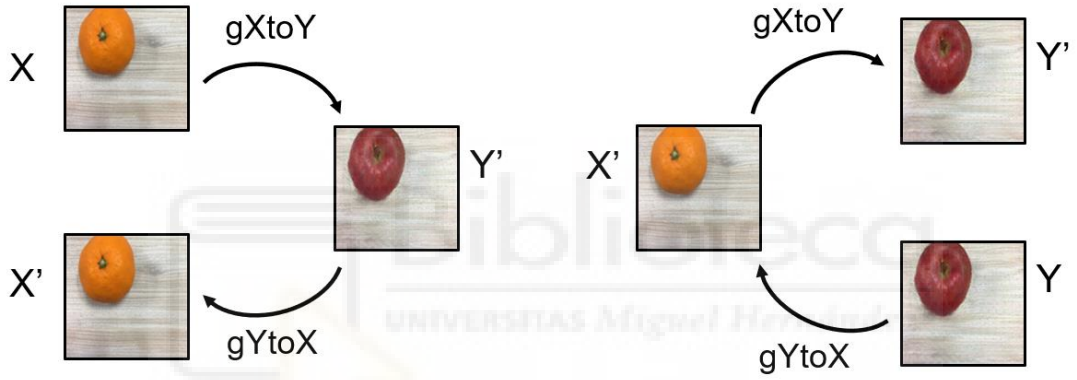


Figura 20: Cycle consistency loss.

Cada red generadora es una red convolucional y está compuesta por tres partes:

- Un encoder que codifica las características representativas de la imagen.
- Una parte transformadora que consta de una serie de bloques residuales convolucionales.
- Un decoder que muestra las características transformadas y genera la imagen final.

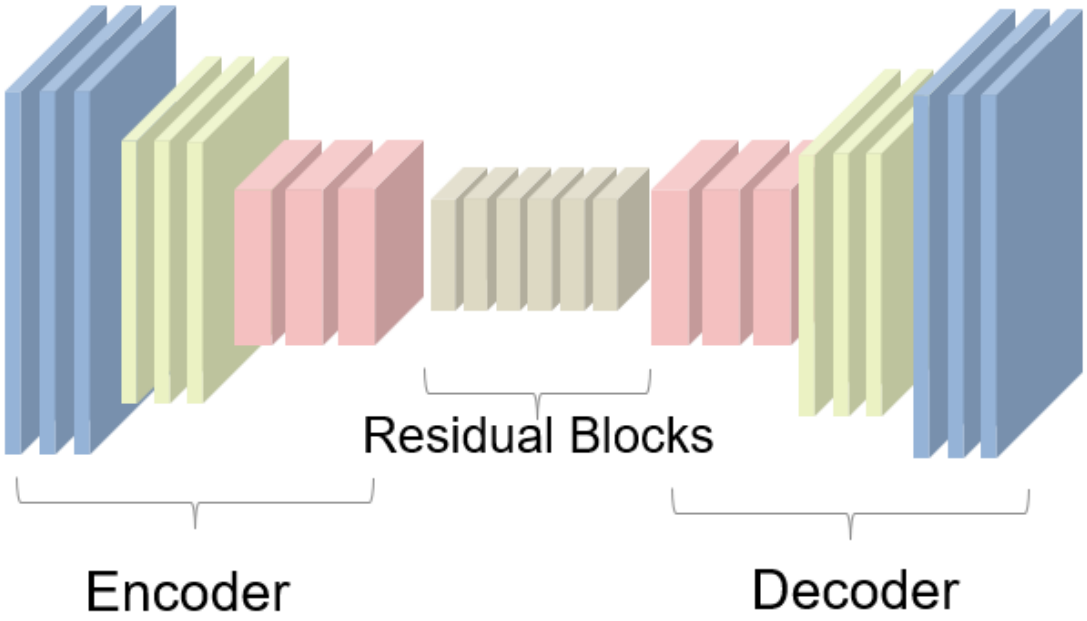


Figura 21: Estructura del generador.

El discriminador tiene la estructura de una red PatchGAN [21] que examina partes de la imagen de entrada y determina la probabilidad de que sea real o falsa.

Figura 22: Estructura del discriminador.

Los creadores de este tipo de redes presentan los resultados de variedad de modelos entrenados con diferentes bases de datos [22].

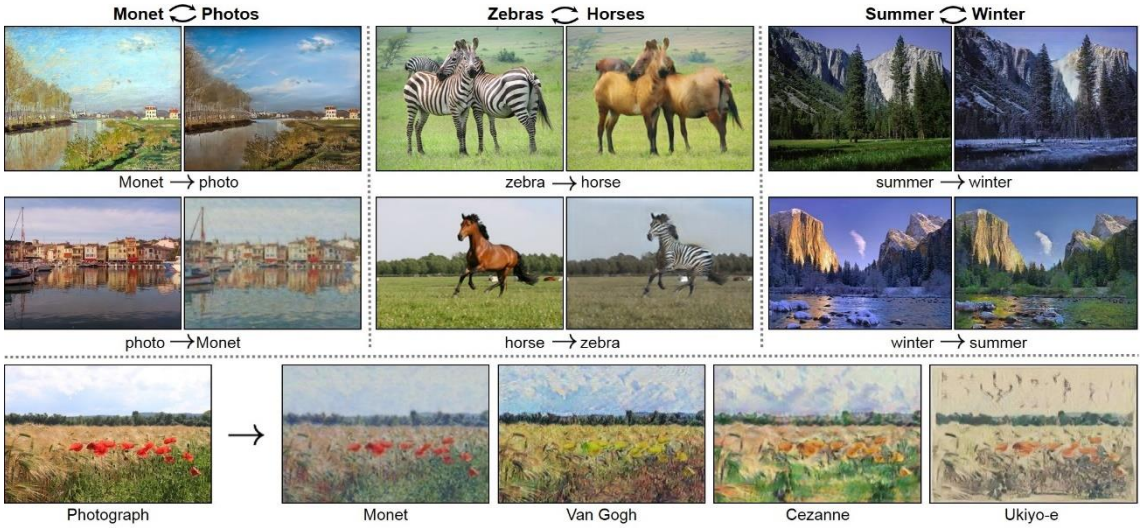


Figura 23: Ejemplos de traslaciones imagen a imagen mediante cycleGAN.

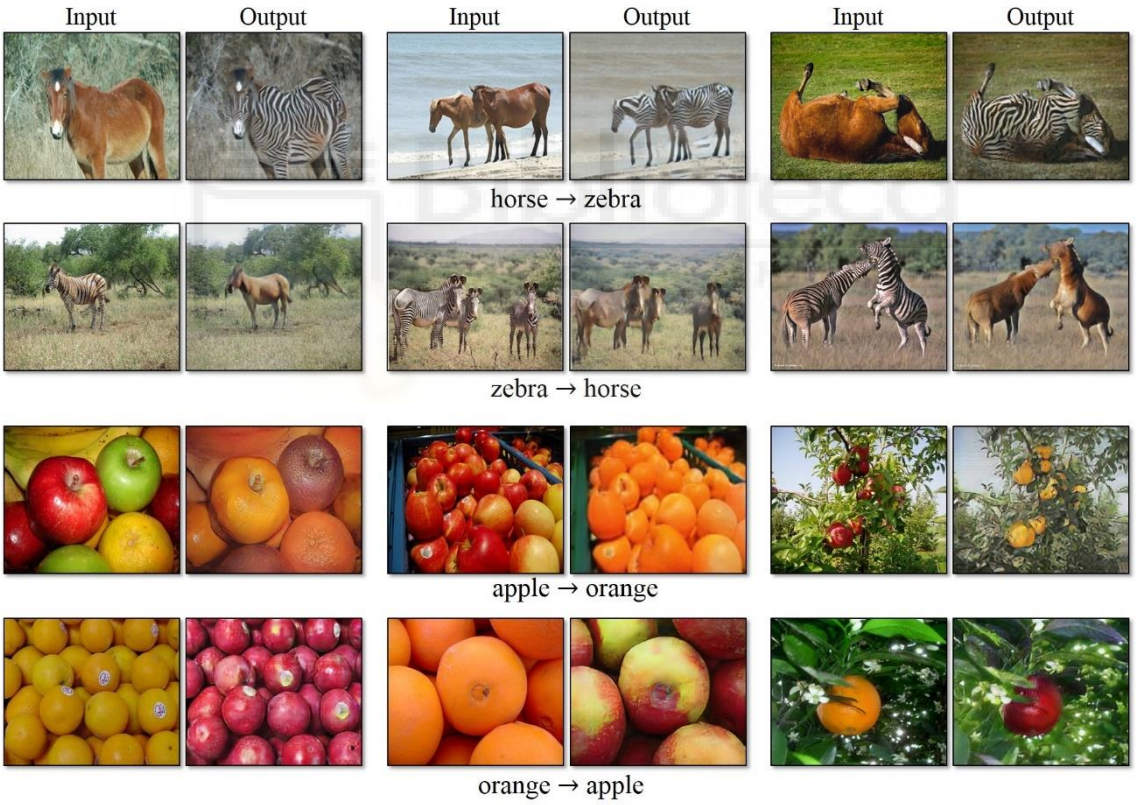


Figura 24: Ejemplos cycleGAN.

En su sitio de guithub aparecen múltiples ejemplos de aplicaciones en las que han sido o pueden ser utilizadas, como crear mapas realistas de viejas ciudades mediante documentos antiguos, cambiar el estilo de una pintura, modificar el renderizado de un videojuego para hacerlo mucho más realista... etc.

4 Experimentación

Para llevar a cabo este proyecto se han realizado los siguientes pasos:

- Realizar un estudio a nivel básico sobre las redes neuronales y su funcionamiento.
- Estudio de las redes GAN y diferentes implementaciones.
- Seleccionar una implementación de GAN para Matlab que se ajuste a este proyecto.
- Encontrar una base de datos adecuada para entrenar nuestro modelo.
- Llevar a cabo el entrenamiento de varios modelos y estudiar su viabilidad para el objetivo deseado.

4.1 Selección de implementación GAN

Tras investigar sobre las diferentes alternativas de redes GAN que podrían adecuarse a este trabajo se decidió utilizar la cycleGAN por diferentes motivos. Inicialmente quedaron descartadas todas las redes que generan imágenes desde cero o a partir de ruido ya que de base nuestra intención es generar imágenes a partir de otras imágenes, tras esto quedaron también descartadas las redes que no contaban con algo como el *cycle consistecy loss*, ya que tampoco nos sirve ninguna red que no sea capaz de conservar la identidad de la imagen original, o que para ello requiera bases de datos emparejadas.

La IcGAN nos habría permitido entrenar un solo modelo para todos los niveles de iluminación, sin embargo, carece del *cycle consistecy loss*, por lo que las imágenes tenderían a perder su identidad original, no siendo claramente reconocible el lugar inicialmente fotografiado tras la reconstrucción modificada, CGAN tiene carencias similares ya que IcGAN es una derivación de esta. Finalmente pese a ser DiscoGAN una candidata válida se decidió trabajar con la red CycleGAN debido a que resultó más sencillo encontrar una implementación válida en Matlab que además permitía ser adaptada a cualquier base de datos de una forma relativamente sencilla.

4.2 Herramientas

Matlab

Todo el trabajo se ha llevado a cabo en Matlab, incluyendo herramientas incluidas en este, como Deep learning toolbox, image procesing toolbox y parallel computing toolbox.

Equipo utilizado en el desarrollo

Todo el trabajo se ha ejecutado en un equipo de sobremesa con las siguientes Especificaciones:

- Sistema operativo: Windows 10.
- CPU: AMD Ryzen 5 3600X @ 3.8 GHz.
- GPU: NVIDIA GTX 1060 6GB
- Ram: 16GB DDR4

4.3 Entrenamiento

Para transformar la iluminación de imágenes hemos trabajado con 3 niveles de iluminación (soleado, nublado y noche), dado que el modelo de esta red es reversible, para transformar cualquiera de estos 3 niveles a cualquiera de los 2 restantes solo necesitamos tener 3 modelos de la red (soleado-noche, nublado-soleado, noche-nublado).

Inicialmente se realizó una prueba para comprobar el funcionamiento de la red con la base de datos para la que estaba configurada, con el objetivo de evaluar la capacidad de la red de obtener y conservar las características de una imagen más allá de ser capaz de transformar un aspecto determinado de forma exitosa, en este caso la red estaba configurada para convertir imágenes de naranjas en manzanas y viceversa. Como se puede observar en la figura 25, la red modifica de forma correcta la pieza de fruta, reproduciendo la superficie de la mesa sin apenas variaciones respecto a la original, por lo que intuimos que la red será capaz de adaptarse fácilmente a otros entornos, así decidimos continuar el estudio con esta implementación.

Figura 25: Test tras el primer entrenamiento.

En las redes neuronales tradicionales, para el proceso de entrenamiento se dispone de los valores de loss y accuracy tanto para el entrenamiento como para test y a partir de cómo evolucionan estos valores conforme se suceden las épocas podemos valorar el estado del entrenamiento y marcar un valor que consideremos óptimo donde detener el entrenamiento. Para el entrenamiento de una red GAN ocurre que el valor de

precisión se pierde debido a que no tenemos una salida objetivo-fija, con la que podamos comparar y valorar de una forma cien por cien objetiva si las imágenes generadas se corresponden con la salida deseada o no, por lo que solo disponemos del loss.

Sin embargo, como podremos comprobar más adelante, los valores de loss de la CycleGAN no muestran la evolución de los resultados que se están consiguiendo ni se ve reflejado el momento en el que el entrenamiento pasa su punto óptimo y comienza a degenerarse y a crear artefactos en las imágenes finales. Por este motivo, los entrenamientos en este tipo de redes deben monitorizarse de manera visual y finalizar el entrenamiento en función de si los resultados cumplen o no nuestro criterio. En este caso para, evaluar el progreso del entrenamiento es debido realizar un entrenamiento extenso y guardar un modelo cada cierto número de épocas para posteriormente compararlos y comprobar cual da mejor resultado, de otra manera habría que estar pendiente de la salida durante toda la duración del entrenamiento y debido a su duración esto no es factible [16].

Las opciones para el algoritmo de optimización Adam utilizadas en todos los entrenamientos son:

- Una ratio de entrenamiento de 0,0002.
- Un decaimiento del gradiente de 0,5.
- Un decaimiento cuadrático del gradiente de 0,999.

Pese a que lo ideal para el entrenamiento hubieran sido unas 1000 épocas, dado que el proceso requiere mucho tiempo se ha recortado a 500 épocas para la base de datos de COLD y a 200 con la base de datos de exteriores, habiendo alcanzado una duración total entre los 4 entrenamientos de unas 2 semanas y 4 días, es por esto que únicamente se ha repetido el entrenamiento para 3 dominios con la base de datos de COLD, mientras que con la base de datos de exteriores se ha limitado a 2 y por lo tanto un solo entrenamiento en lugar de 3.

Los conjuntos de imágenes no están emparejados ya que no es necesario con este tipo de redes, tampoco están etiquetadas, el entrenamiento las reconoce como de un dominio u otro por la carpeta en la que están almacenadas.

Entrenamiento con base de datos COLD

El entrenamiento del modelo para transformar imágenes de soleado a noche se realizó mediante 1244 imágenes del dominio sunny, frente a 1344 imágenes del dominio night en 500 épocas, resultando en 622000 iteraciones lo que ha supuesto una duración total de 98 horas 47 minutos y 45 segundos, para los otros dos las cifras son bastante similares.

Soleado-noche

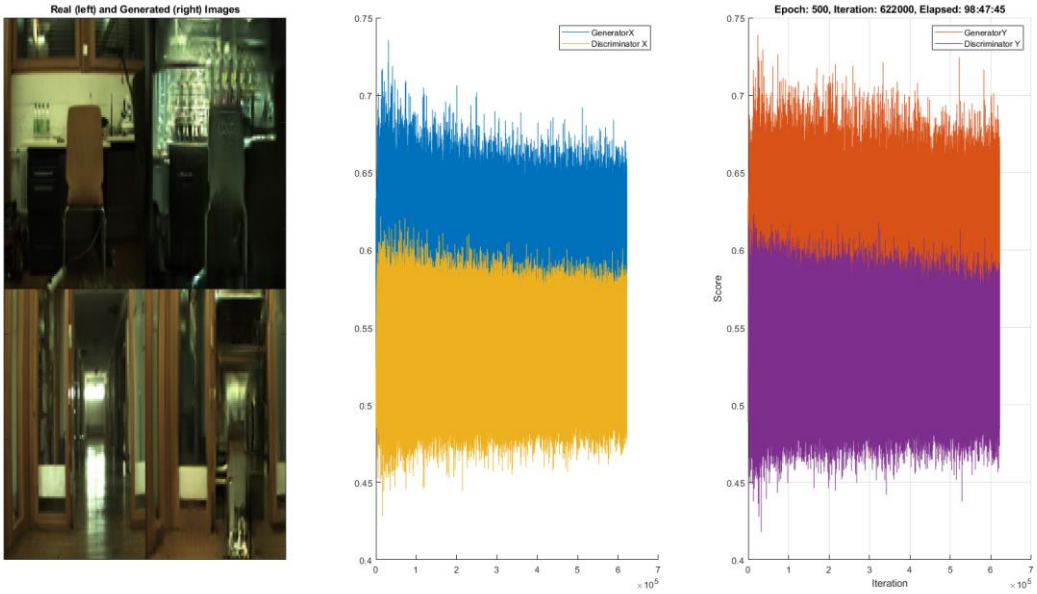


Figura 26: Evolución entrenamiento 1 de soleado a noche con la base de datos de COLD.

En este punto aprovecharemos lo que se muestra en la figura 26 para explicar por qué la debemos controlar la evolución del entrenamiento de forma visual en este tipo de redes. Como introducimos anteriormente para una red GAN no disponemos de la información que nos da la gráfica del accuracy debido a que para obtener este valor es necesario disponer de imágenes reales del dominio de salida para poder realizar una comparación, por lo que solo podemos obtener la gráfica loss, pues bien, tal y como se observa en la figura anterior con este valor tampoco obtenemos información acerca de la evolución del entrenamiento que nos permita evaluar el estado del entrenamiento ya que como se puede apreciar, pese a que al principio si se aprecia una ligera reducción de las oscilaciones en ningún momento se consigue ningún tipo de convergencia entre el generador y el discriminador y lo que se muestra es más bien ruido, el resto de entrenamientos muestran el mismo resultado.

Nublado-soleado

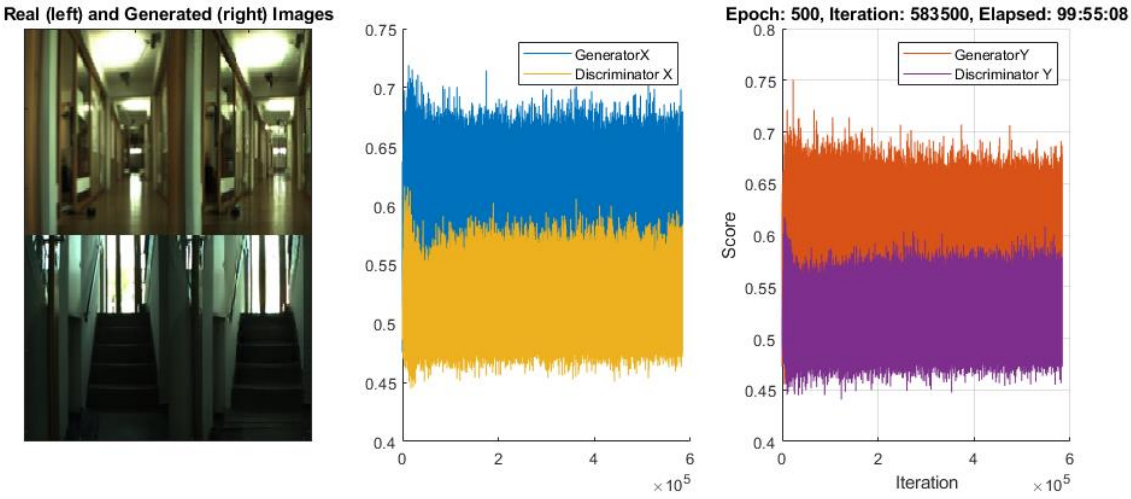


Figura 27: Evolución entrenamiento 1 de nublado a soleado con la base de datos de COLD.

Noche-nublado

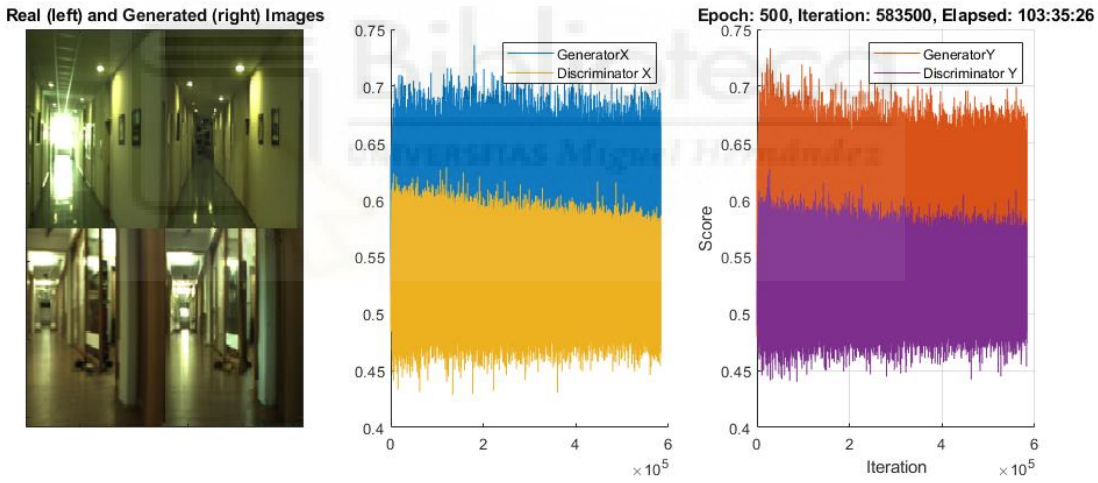


Figura 28: Evolución entrenamiento 1 de noche a nublado con la base de datos de COLD.

Entrenamiento con imágenes de exteriores

El entrenamiento del modelo con imágenes de exteriores se realizó mediante 738 imágenes del dominio sunny, frente a 519 imágenes del dominio night en 200 épocas, durando 23 horas y 45 minutos.

Soleado-noche

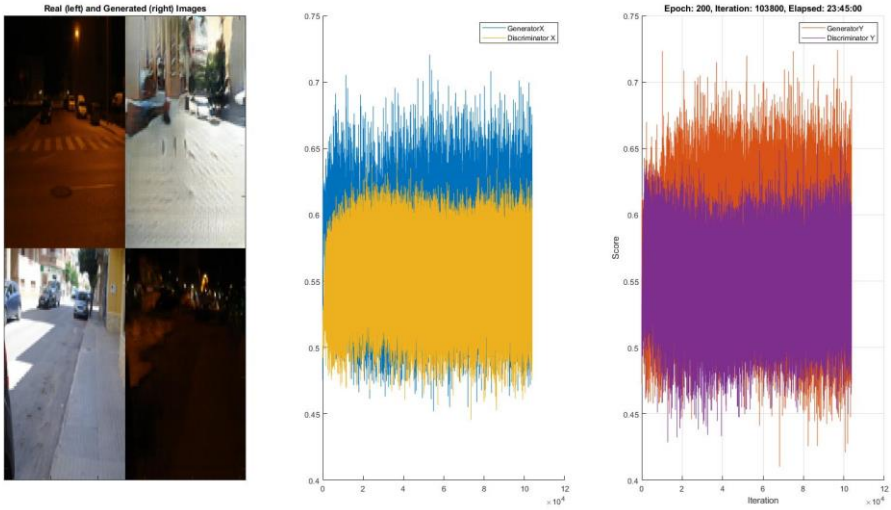


Figura 29: Evolución entrenamiento 2.

4.4 Experimentos

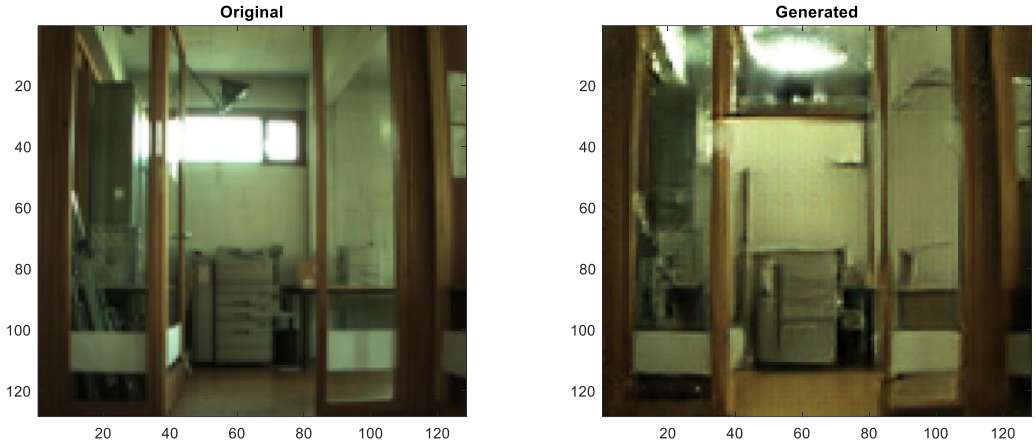
Como hemos visto, para las redes Cycle GAN es necesario un método visual para evaluar el entrenamiento de la red, para este caso seleccionaremos una imagen de cada entorno (Freiburg, Ljubljana y Saarbrücken) para cada modelo de transformación (Soleado-noche, Nublado-soleado y Noche-nublado), con esto repetiremos cada transformación con cada modelo guardado según el número de épocas (100, 200, 300, 400 y 500 épocas), de esta manera, comparando las imágenes de salida obtenidas por los modelos de menos entrenado a más entrenado, podremos saber si hemos logrado alcanzar el punto óptimo de entrenamiento y a partir de qué número de épocas no conviene seguir entrenando la red.

Las imágenes elegidas para realizar las pruebas son de los mismos datasets utilizados para el entrenamiento, pero se han seleccionado de entre las que se descartaron para el mismo por lo que son imágenes nuevas para la red.

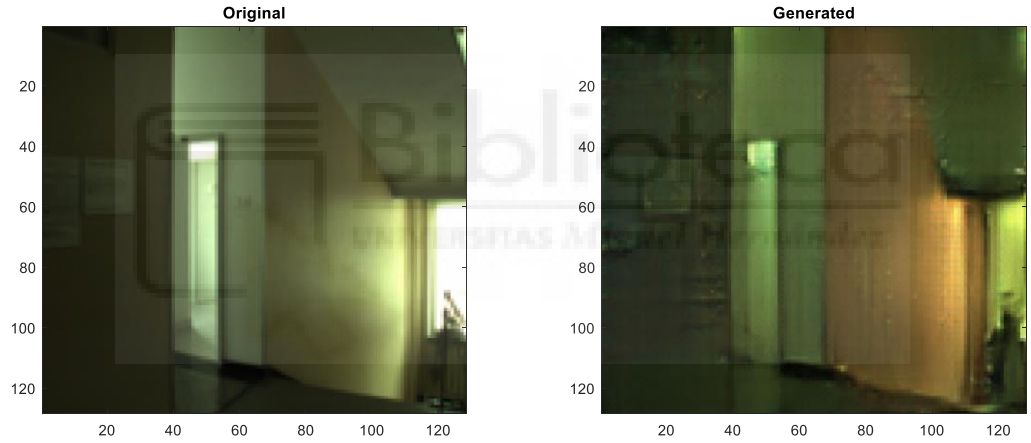
Soleado-noche

Resultados para 100 épocas:

Freiburg



Ljubljana



Saarbrücken

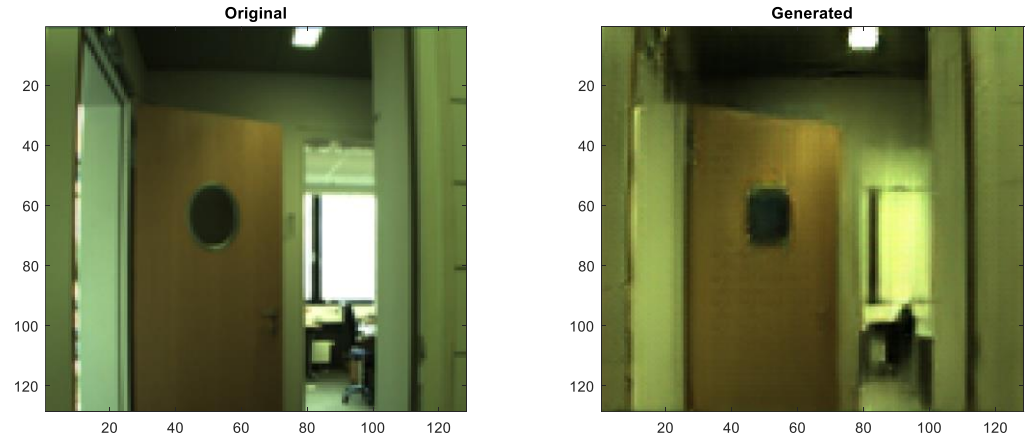
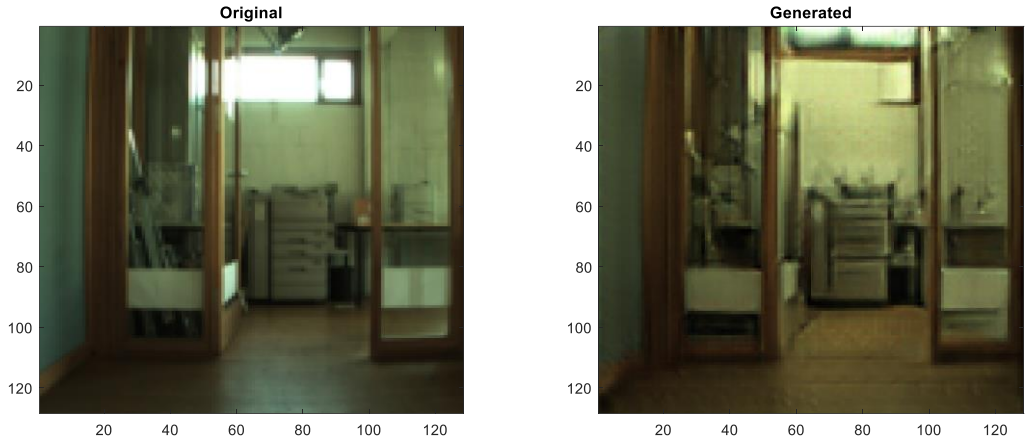


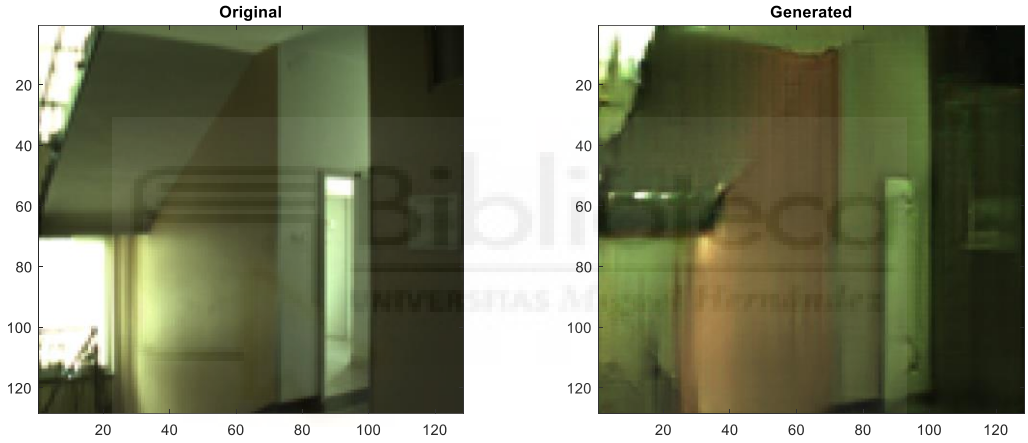
Figura 30: Conversión de soleado a noche realizada por el modelo entrenado con 100 épocas.

Resultados para 200 épocas:

Freiburg



Ljubljana



Saarbrücken

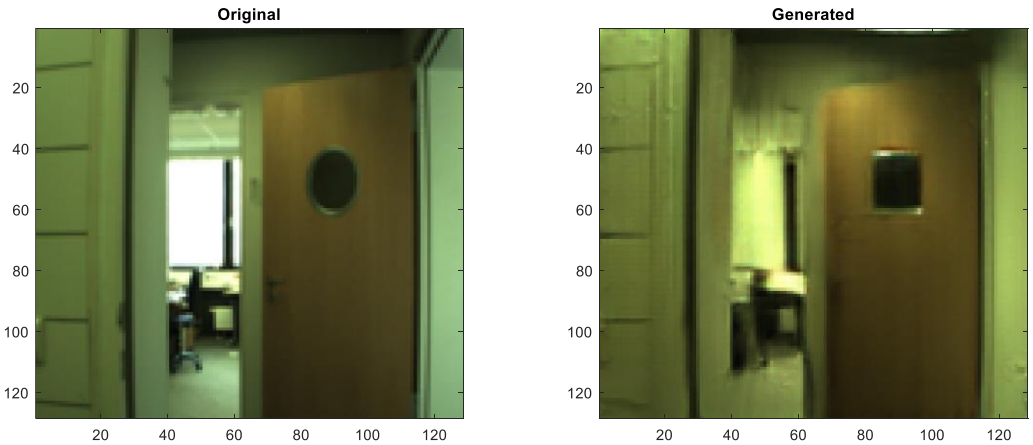
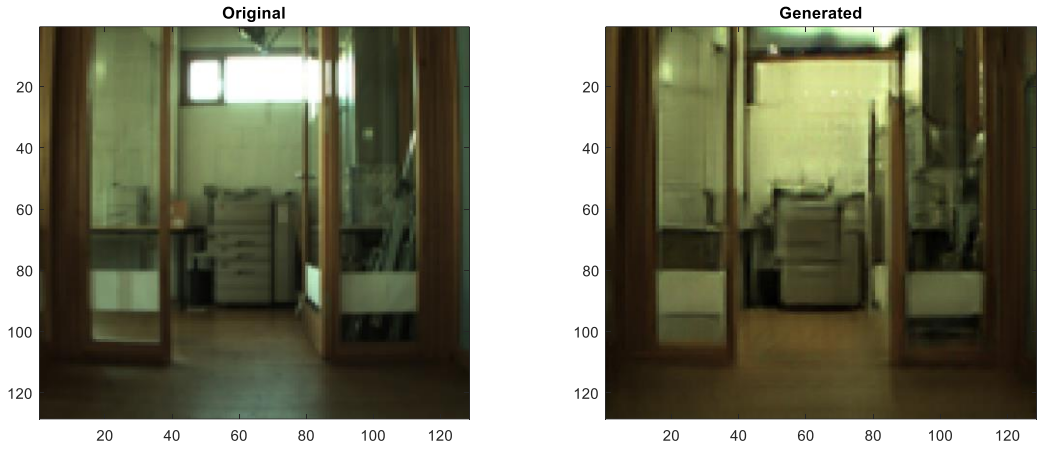


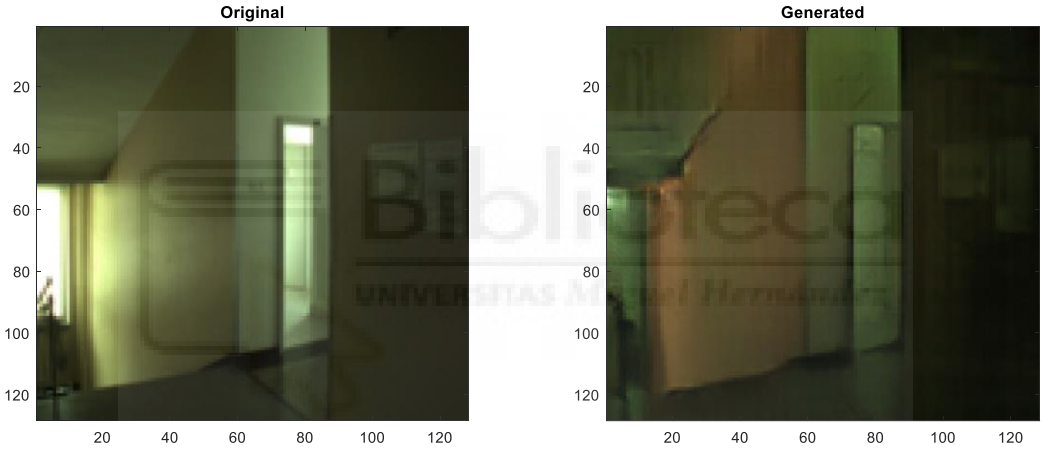
Figura 31: Conversión de soleado a noche realizada por el modelo entrenado con 200 épocas.

Resultados para 300 épocas:

Freiburg



Ljubljana



Saarbrücken

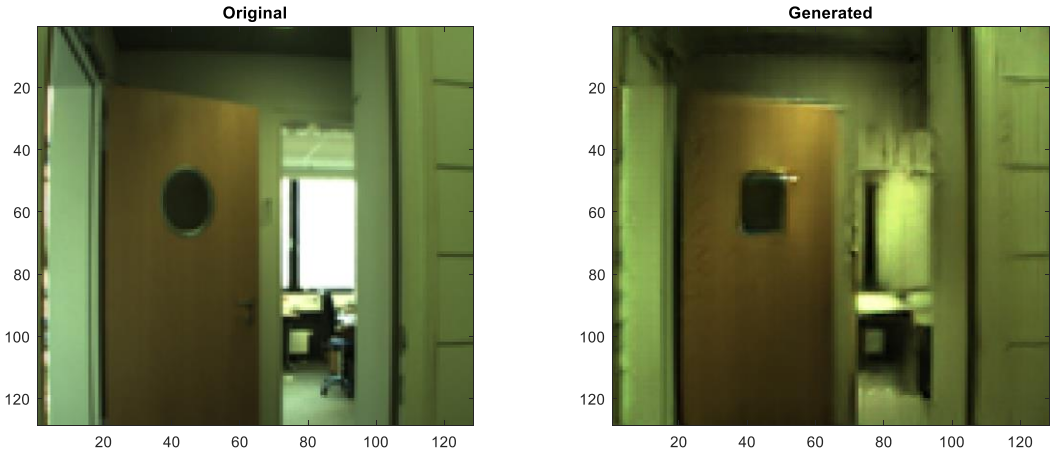
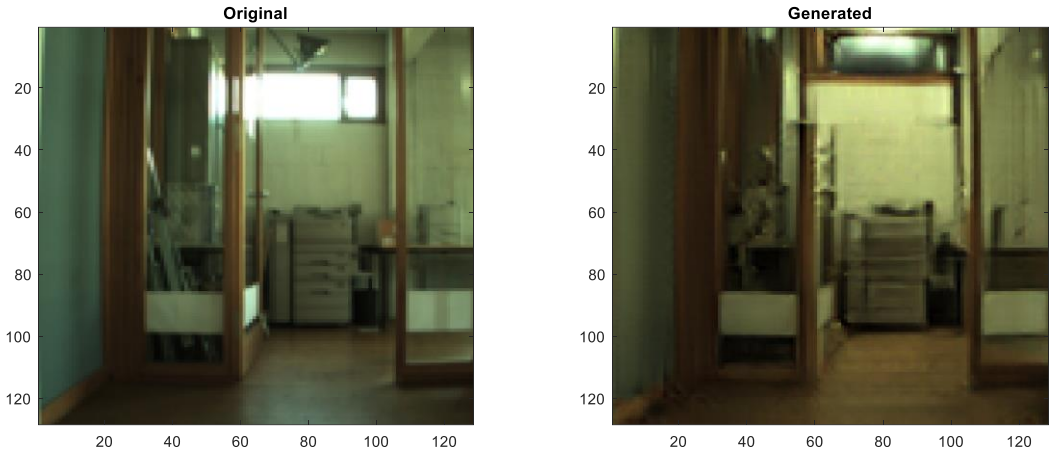


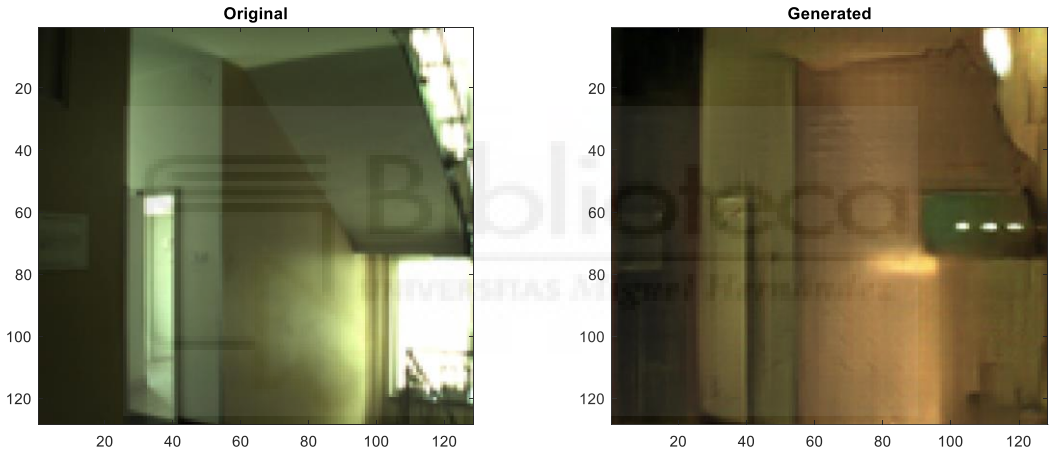
Figura 32: Conversión de soleado a noche realizada por el modelo entrenado con 300 épocas.

Resultados para 400 épocas:

Freiburg



Ljubljana



Saarbrücken

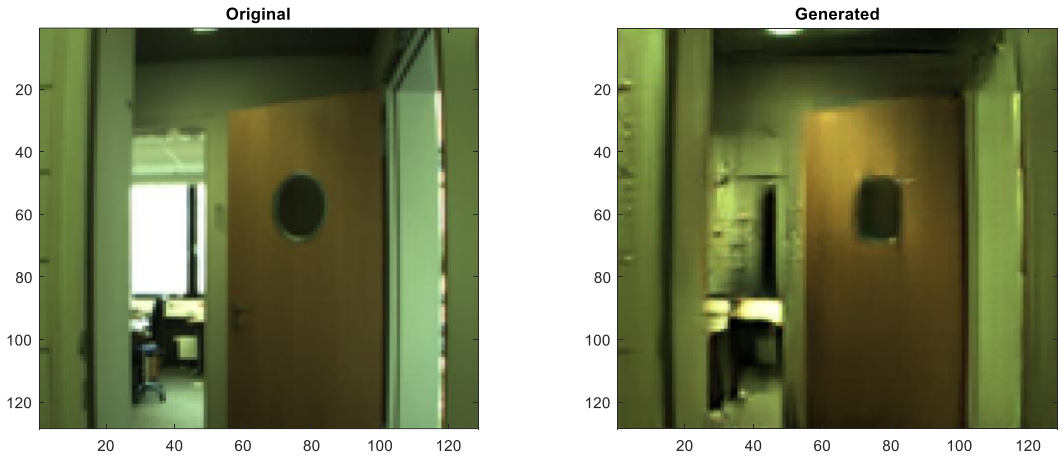
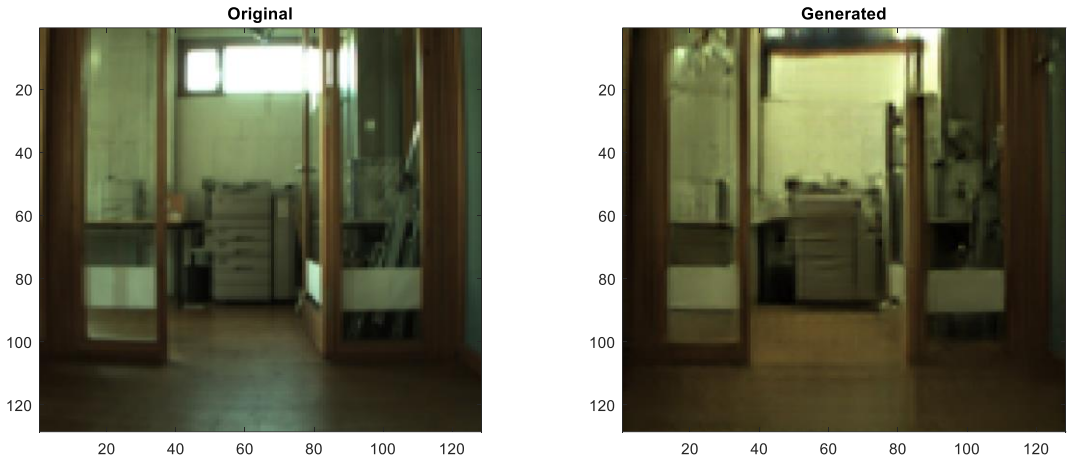


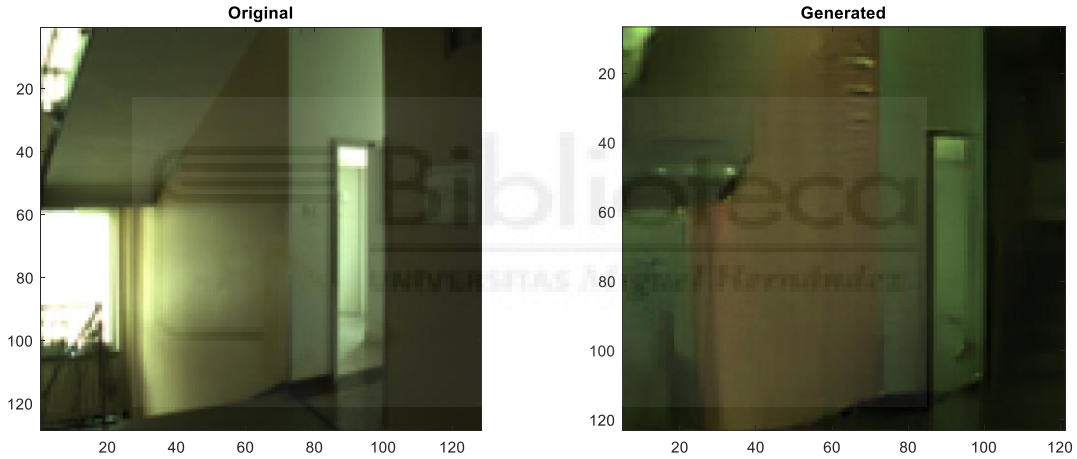
Figura 33: Conversión de soleado a noche realizada por el modelo entrenado con 400 épocas.

Resultados para 500 épocas:

Freiburg



Ljubljana



Saarbrücken

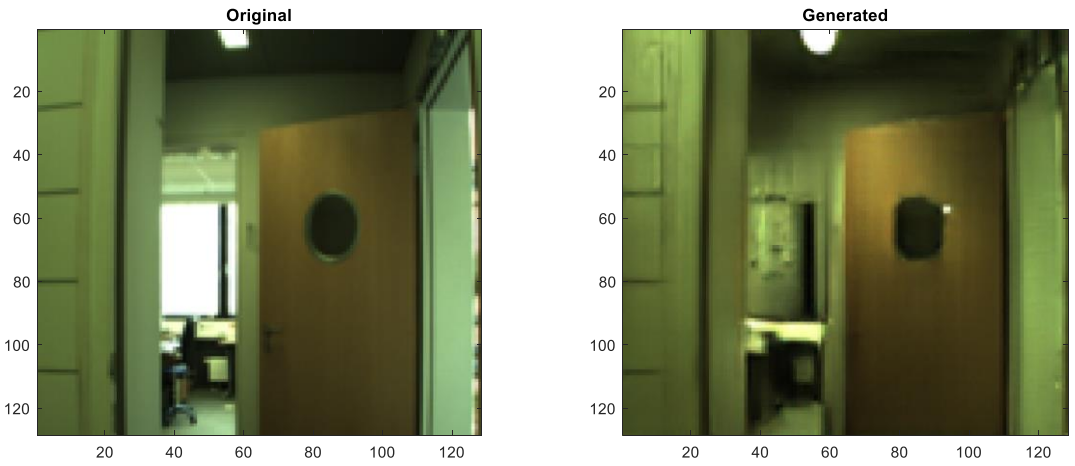
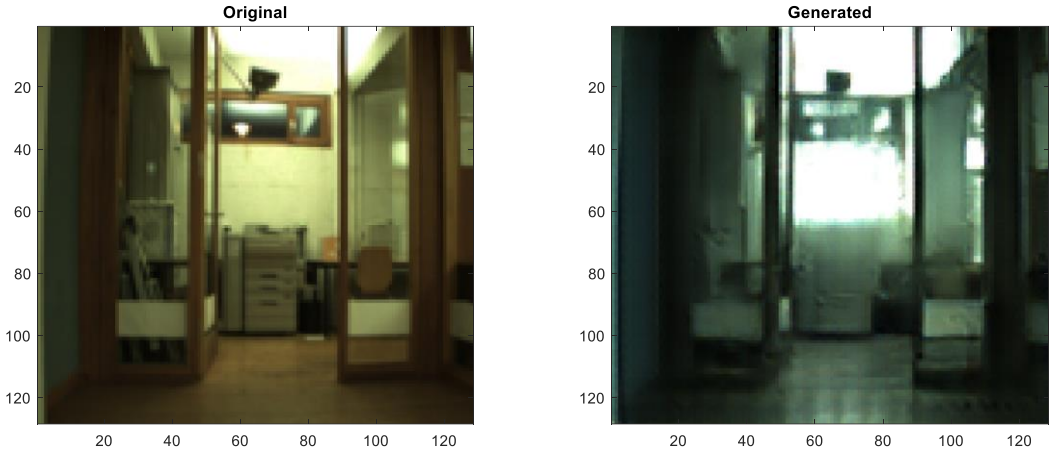


Figura 34: Conversión de soleado a noche realizada por el modelo entrenado con 500 épocas.

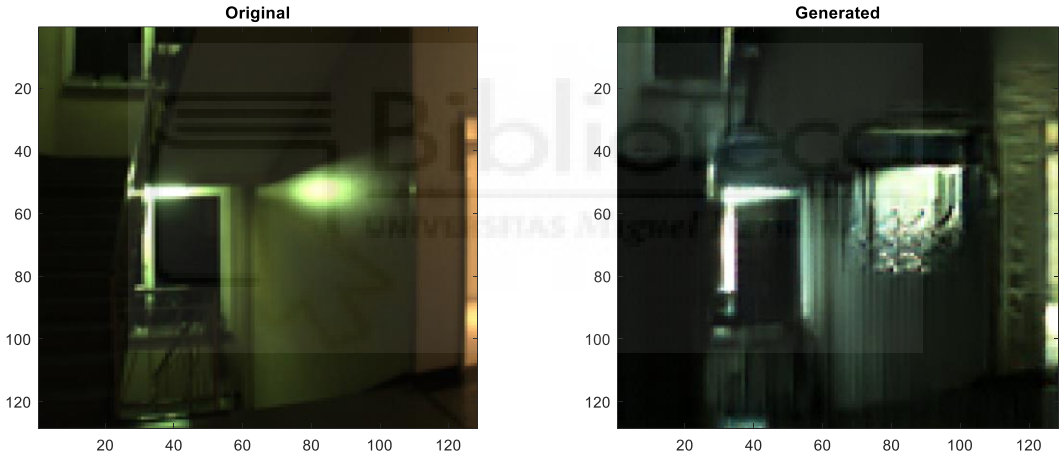
Noche-nublado

Resultados para 100 épocas:

Freiburg



Ljubljana



Saarbrücken

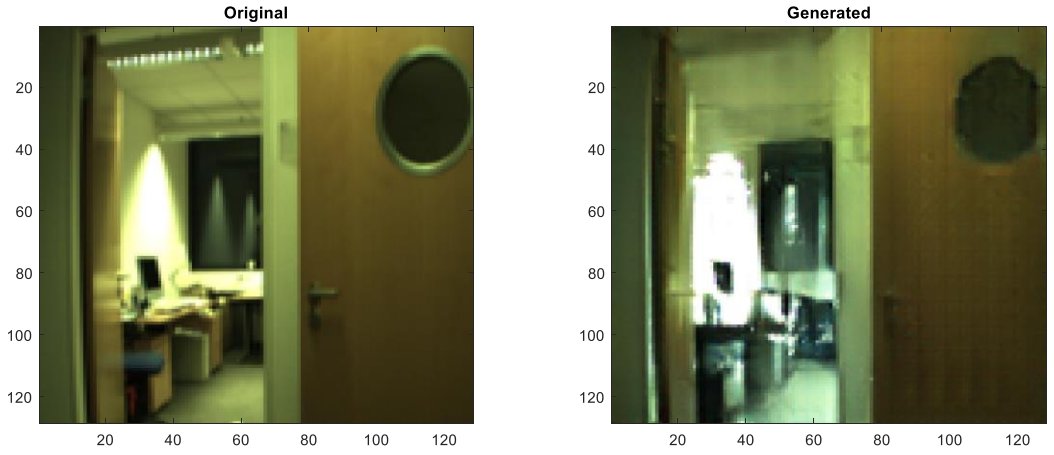
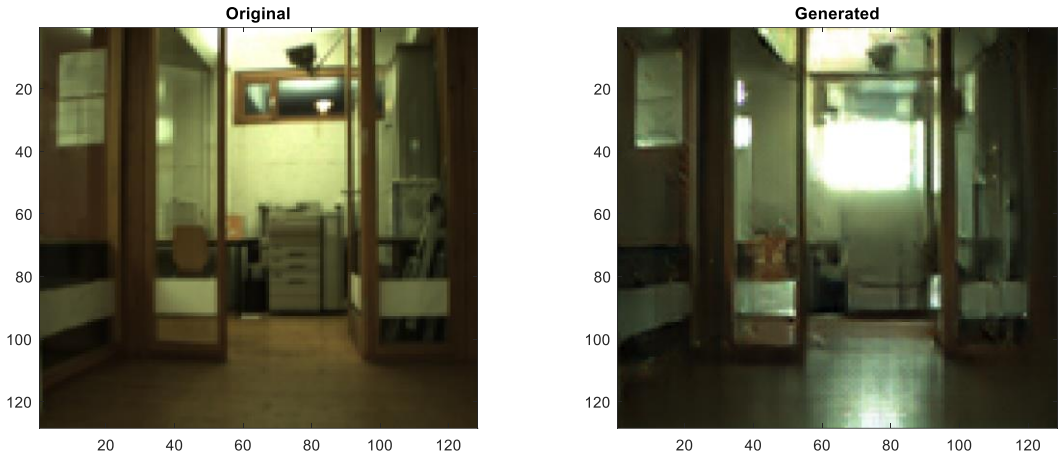


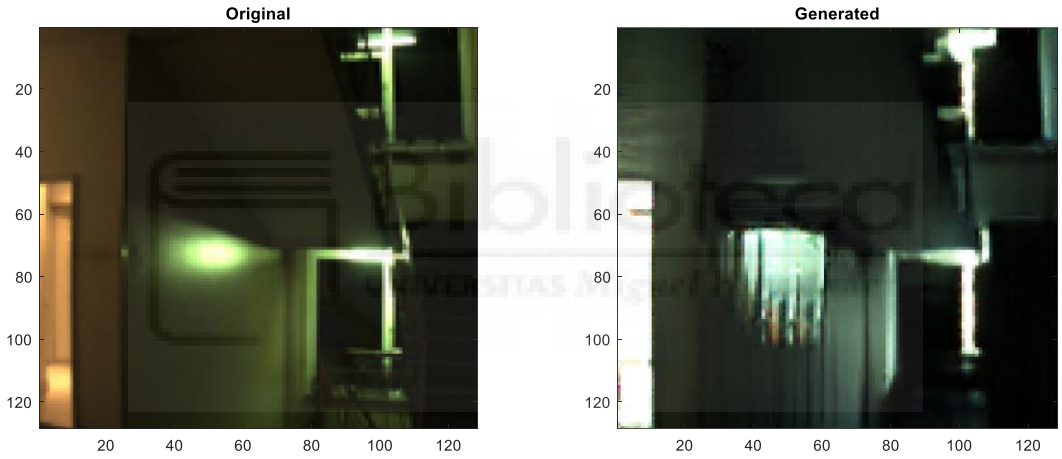
Figura 35: Conversión de noche a nublado realizada por el modelo entrenado con 100 épocas.

Resultados para 200 épocas:

Freiburg



Ljubljana



Saarbrücken

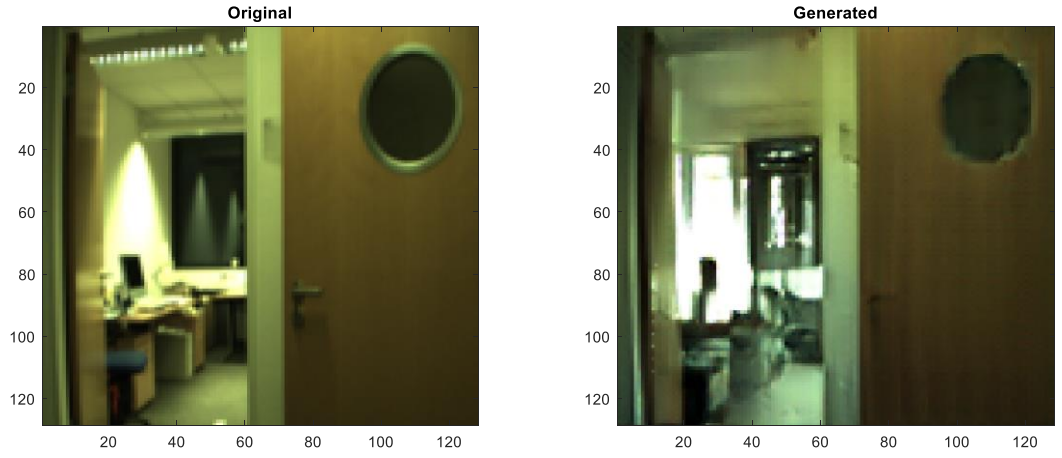
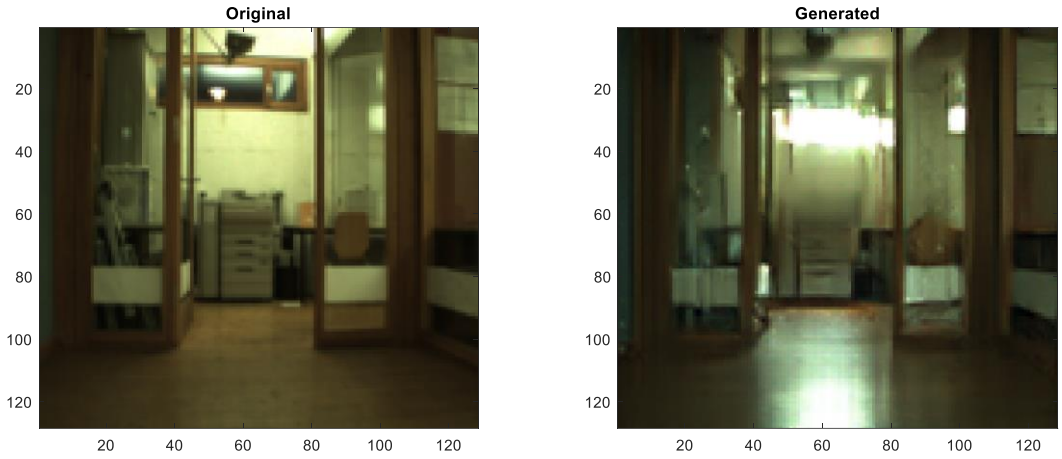


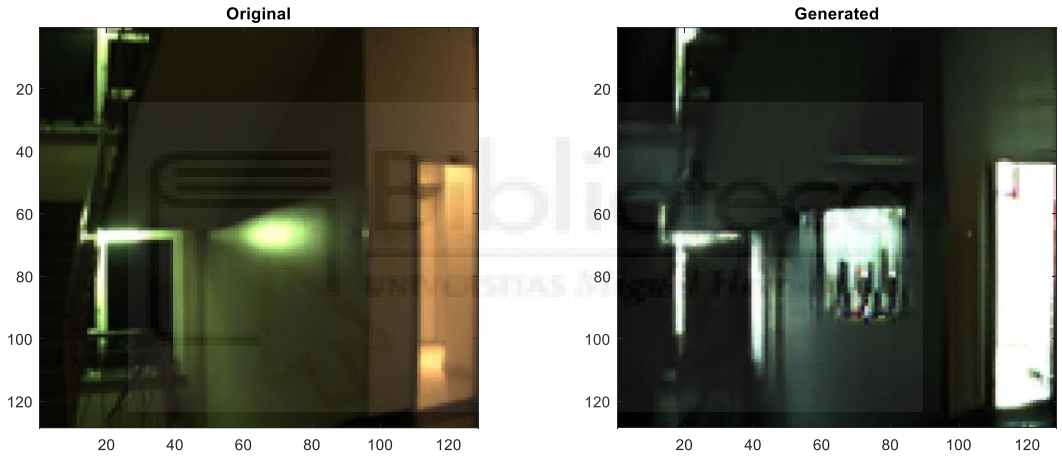
Figura 36: Conversión de noche a nublado realizada por el modelo entrenado con 200 épocas.

Resultados para 300 épocas:

Freiburg



Ljubljana



Saarbrücken

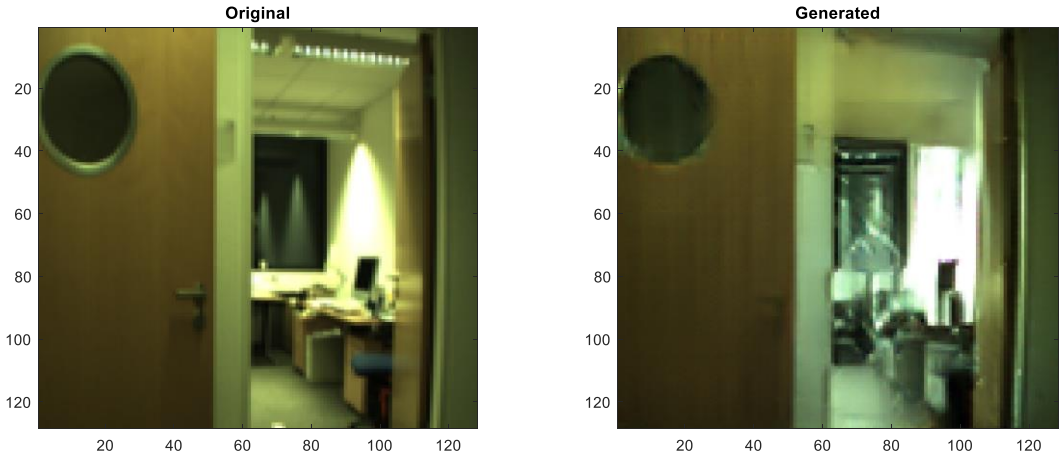
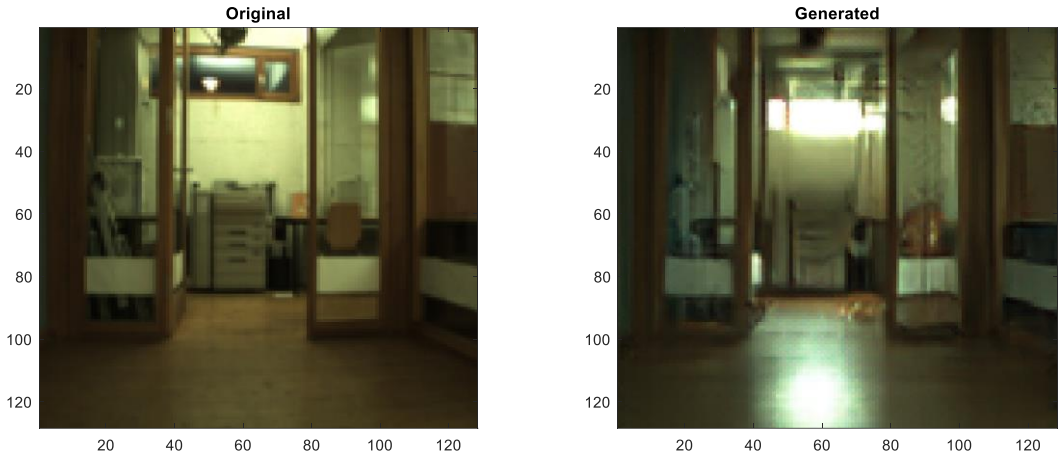


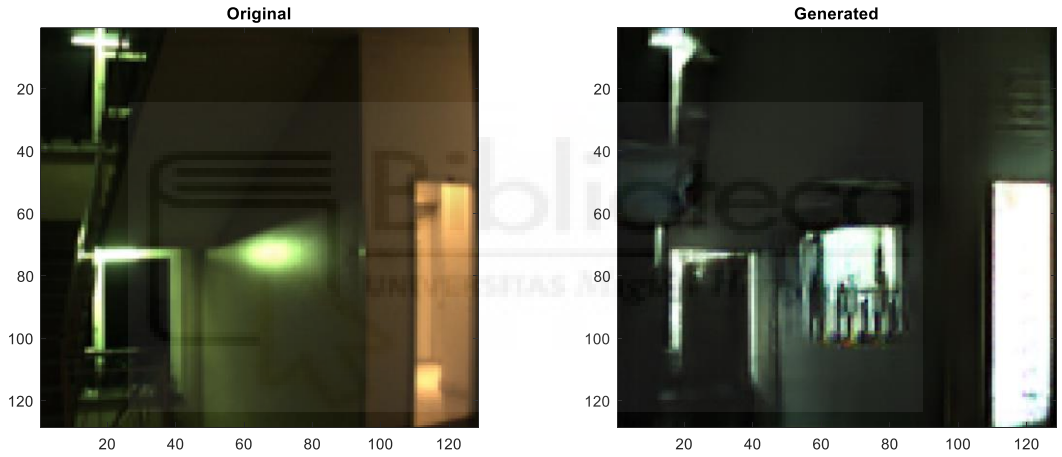
Figura 37: Conversión de noche a nublado realizada por el modelo entrenado con 300 épocas.

Resultados para 400 épocas:

Freiburg



Ljubljana



Saarbrücken

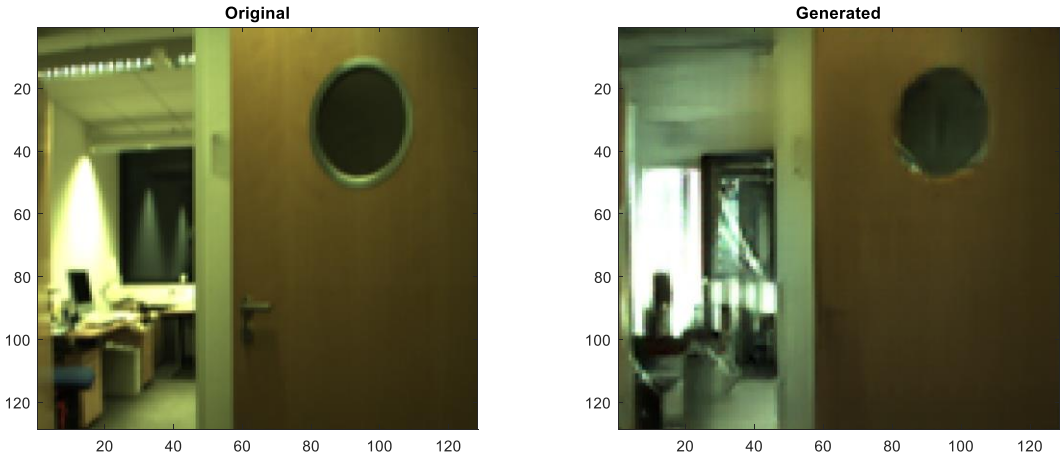
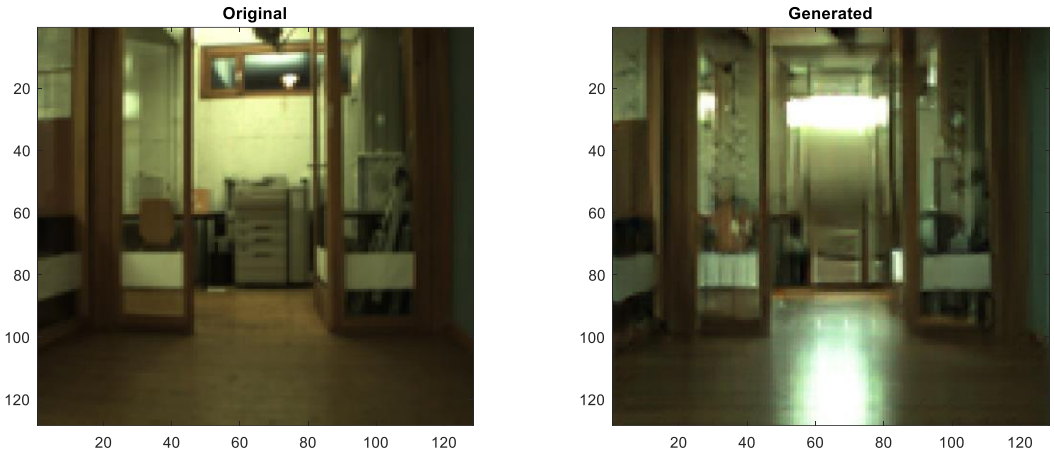


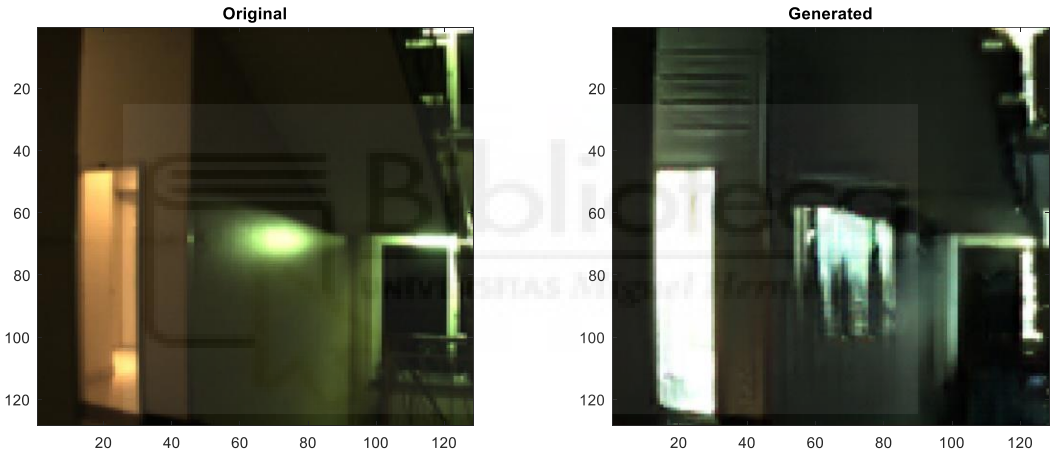
Figura 38: Conversión de noche a nublado realizada por el modelo entrenado con 400 épocas.

Resultados para 500 épocas:

Freiburg



Ljubljana



Saarbrücken

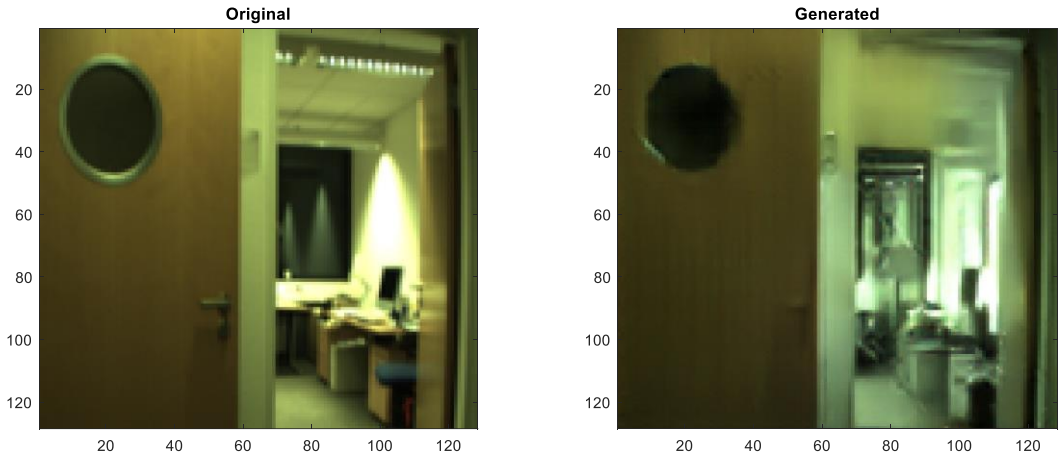
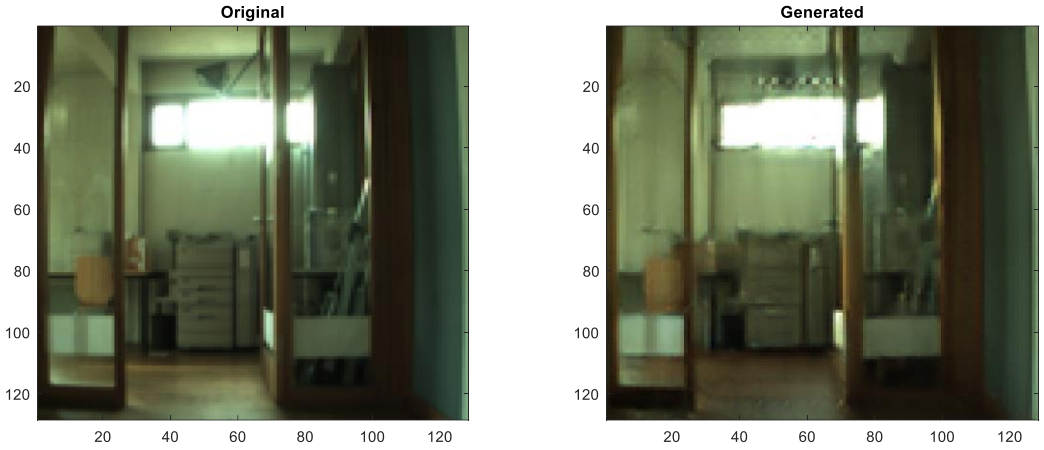


Figura 39: Conversión de noche a nublado realizada por el modelo entrenado con 500 épocas.

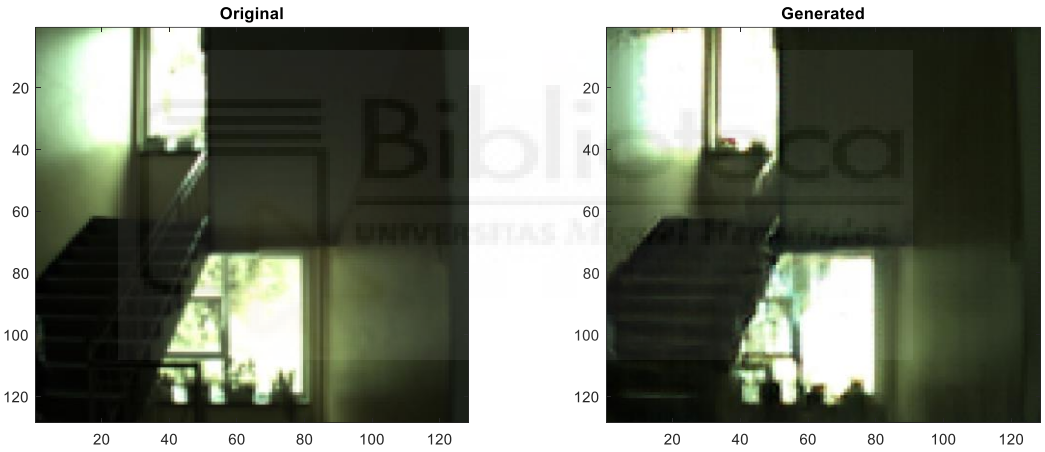
Nublado-soleado

Resultados para 100 épocas:

Freiburg



Ljubljana



Saarbrücken

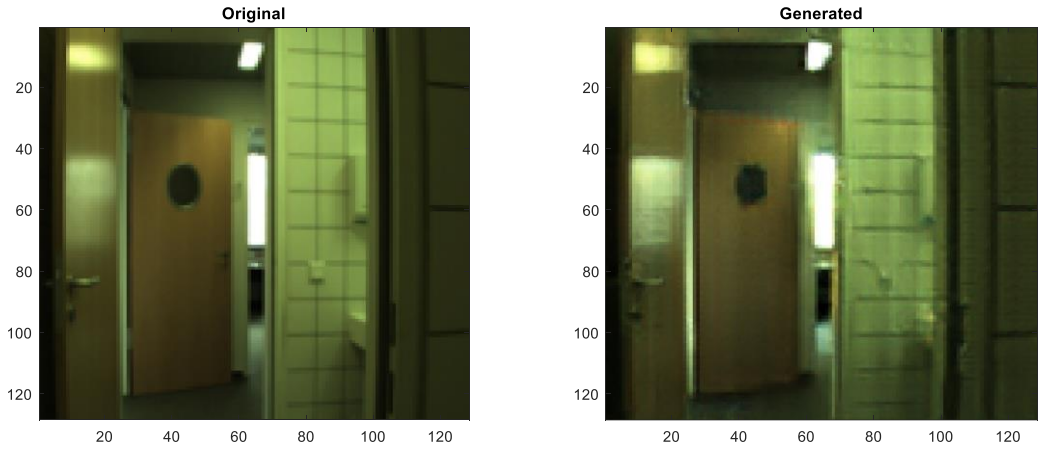
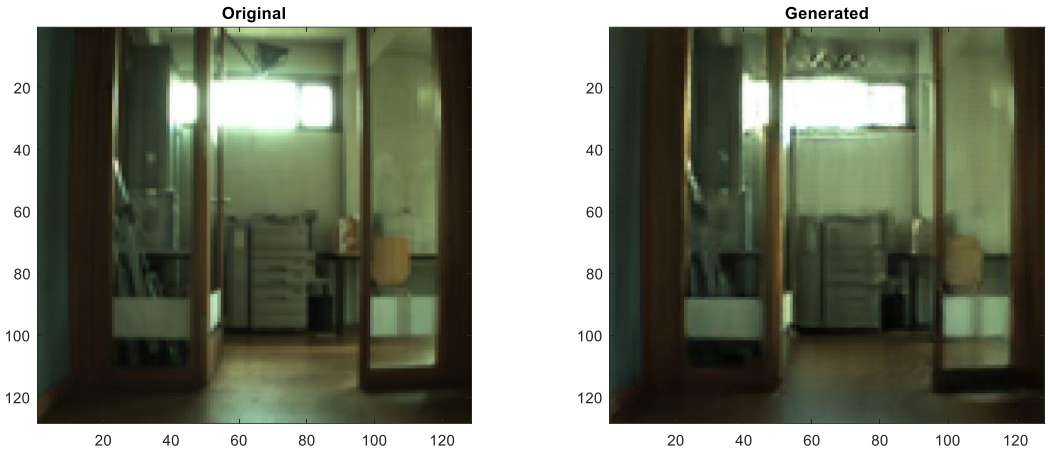


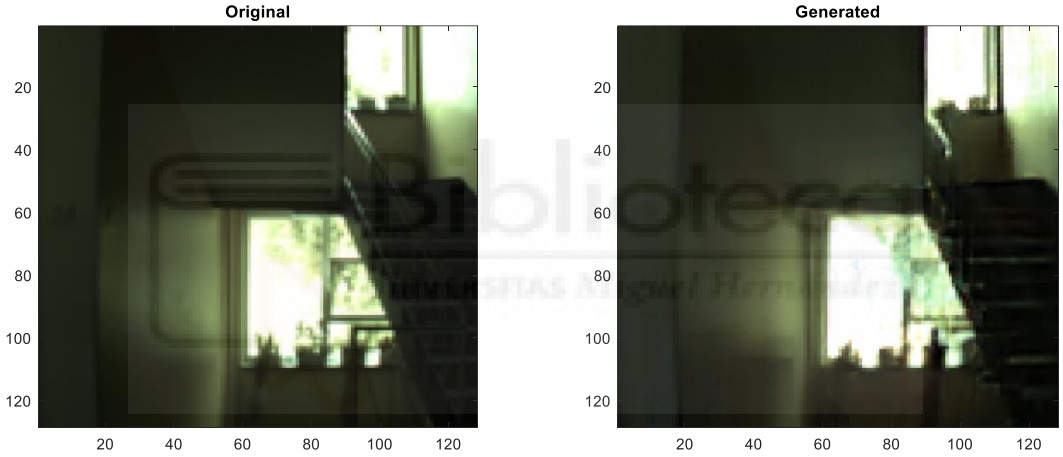
Figura 40: Conversión de nublado-soleado realizada por el modelo entrenado con 100 épocas.

Resultados para 200 épocas:

Freiburg



Ljubljana



Saarbrücken

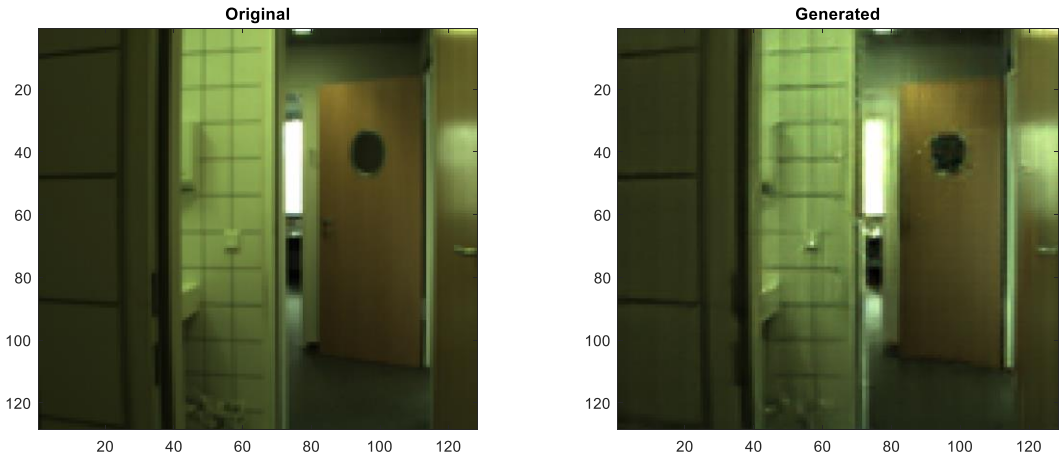
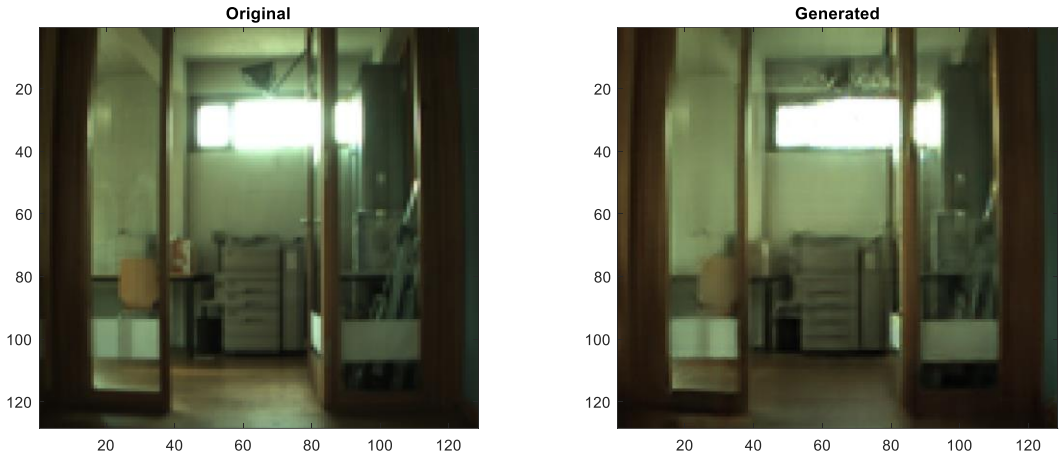


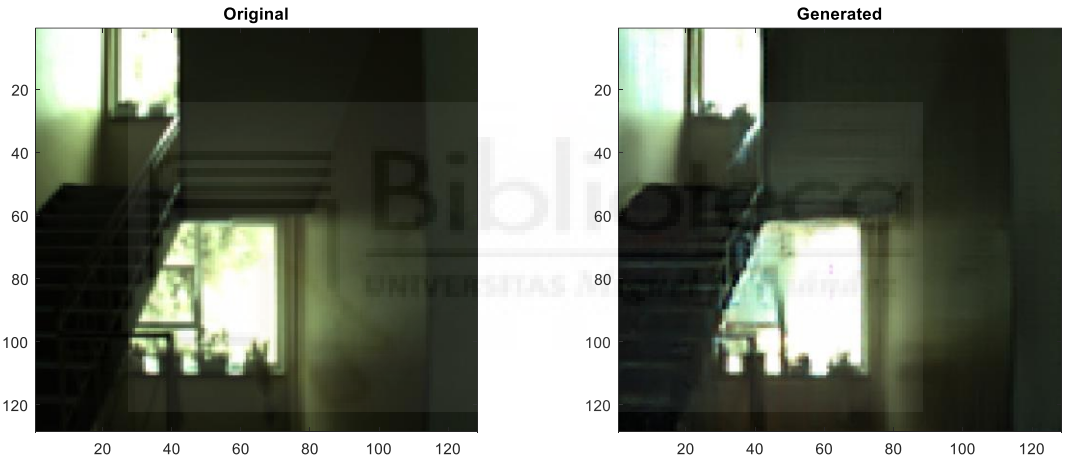
Figura 41: Conversión de nublado-soleado realizada por el modelo entrenado con 200 épocas.

Resultados para 300 épocas:

Freiburg



Ljubljana



Saarbrücken

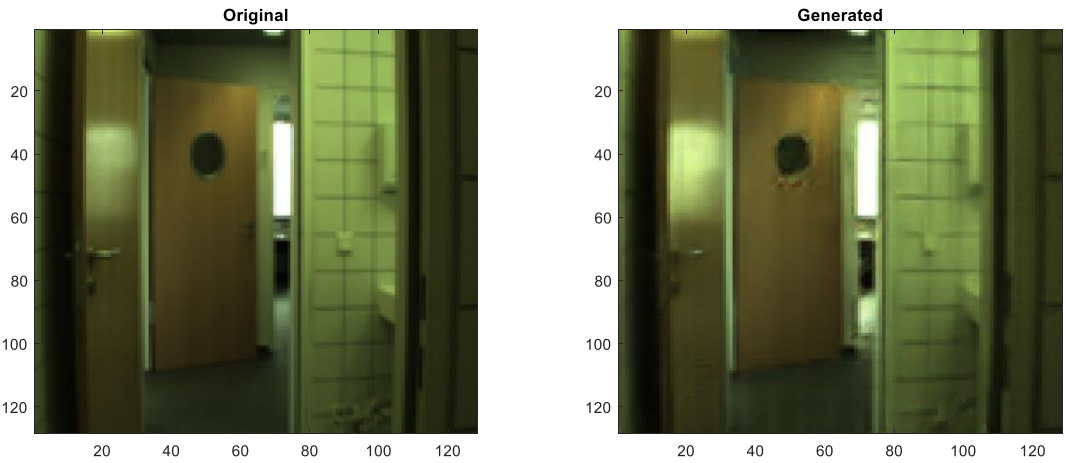
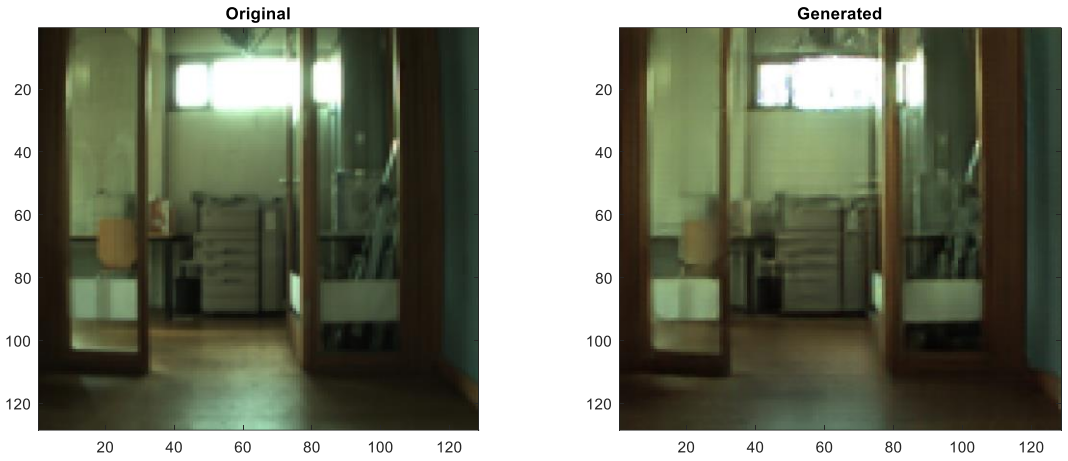


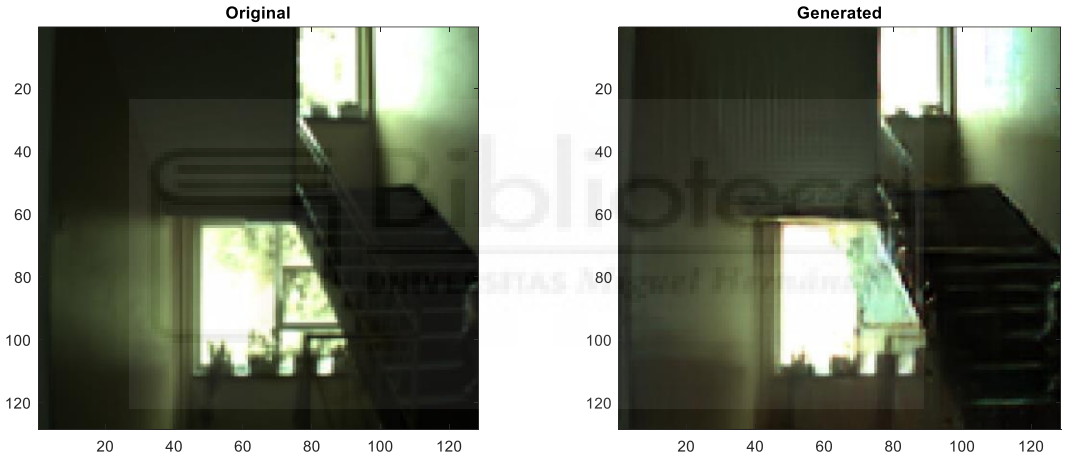
Figura 42: Conversión de nublado-soleado realizada por el modelo entrenado con 300 épocas.

Resultados para 400 épocas:

Freiburg



Ljubljana



Saarbrücken

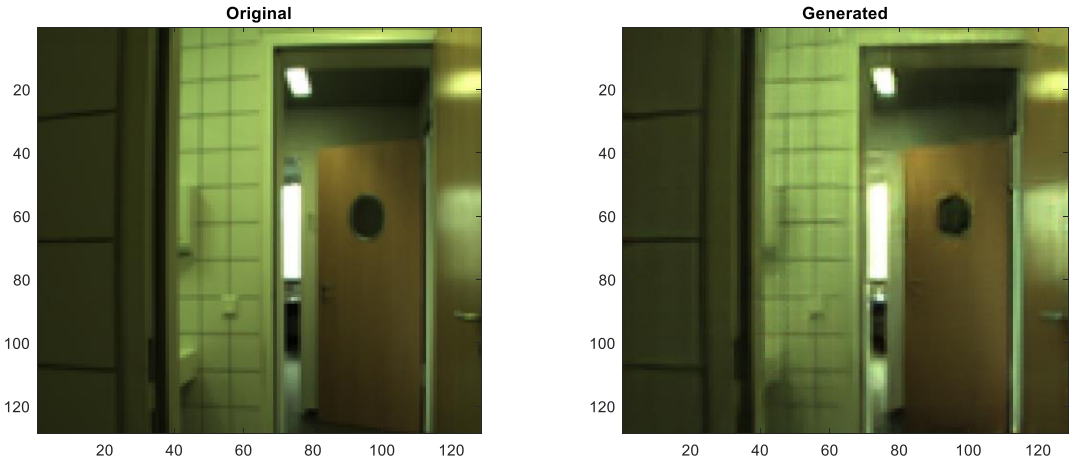
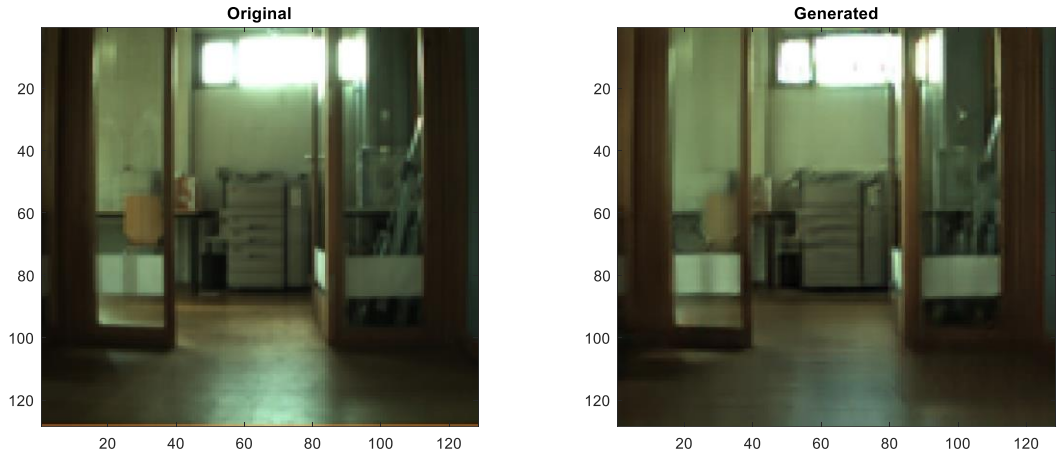


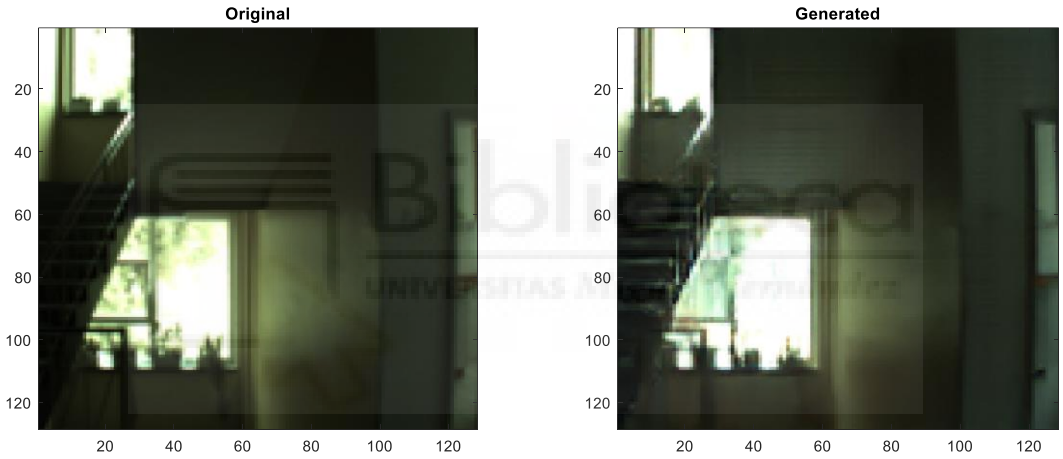
Figura 43: Conversión de nublado-soleado realizada por el modelo entrenado con 400 épocas.

Resultados para 500 épocas:

Freiburg



Ljubljana



Saarbrücken

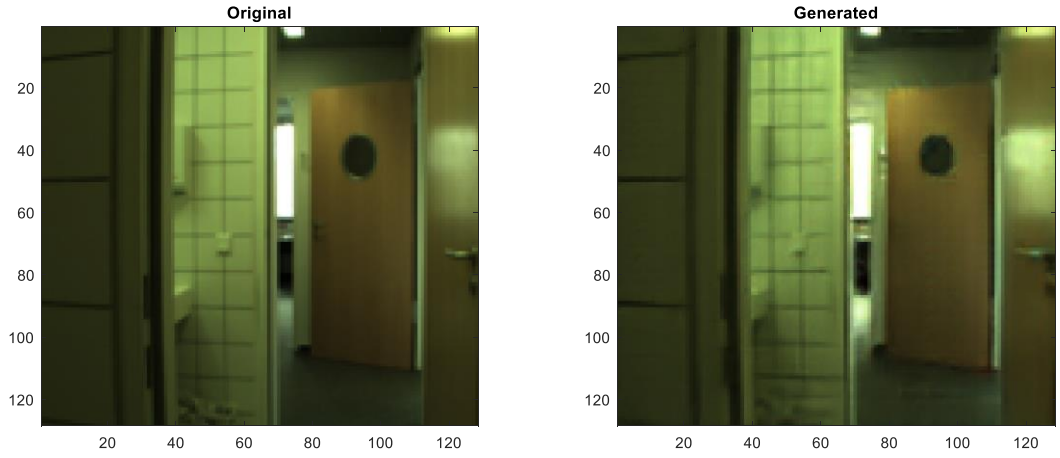


Figura 44: Conversión de nublado-soleado realizada por el modelo entrenado con 500 épocas.

Finalmente podemos decir que los modelos entrenados son capaces de distinguir los matices que caracterizan la iluminación de las imágenes y de reconstruirlas aplicando los cambios deseados, se puede apreciar por regla general que los modelos van mejorando en la tarea de reconstruir las imágenes a medida que se suceden las épocas, pudiendo observar una reducción de los artefactos cuanto más entrenado está el modelo.

Para las pruebas con imágenes exteriores se utilizaron fotografías tomadas manualmente a parte de las que se usaron para el entrenamiento, en este caso hemos realizado las conversiones en los dos sentidos aprovechando que el modelo es reversible, es decir hemos realizado las transformaciones tanto del dominio x a y como de y a x.

Resultados para 100 épocas:

Dia-noche

Figura 45: Conversión día noche 1 (100 épocas).

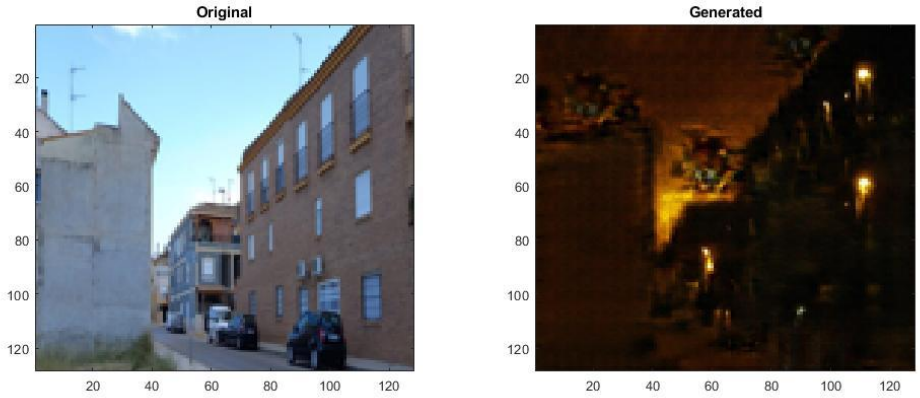


Figura 46: Conversión día noche 2 (100 épocas).

Noche-día

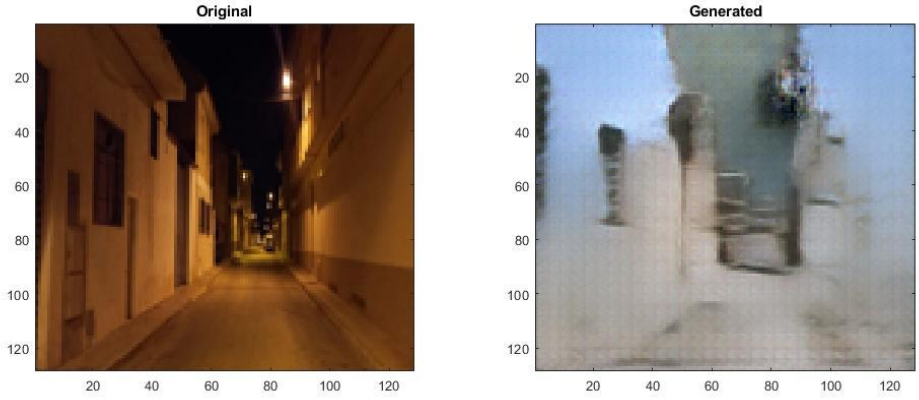


Figura 47: Conversión noche-día 1 (100 épocas).

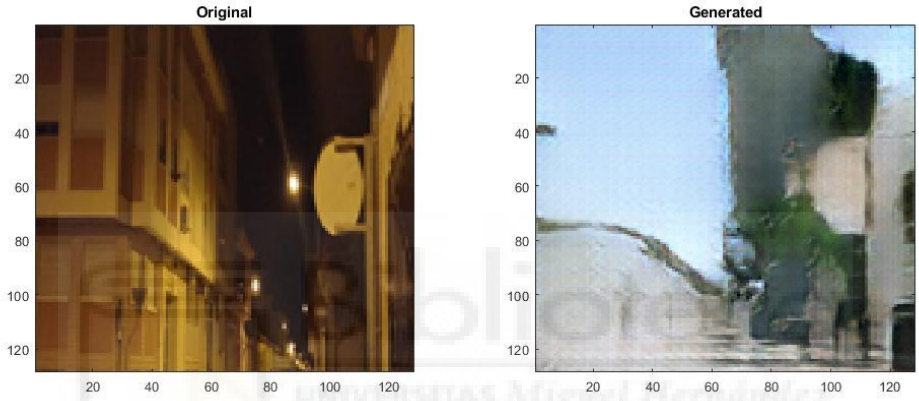


Figura 48: Conversión noche-día 2 (100 épocas).

Resultados para 200 épocas:

Día-noche

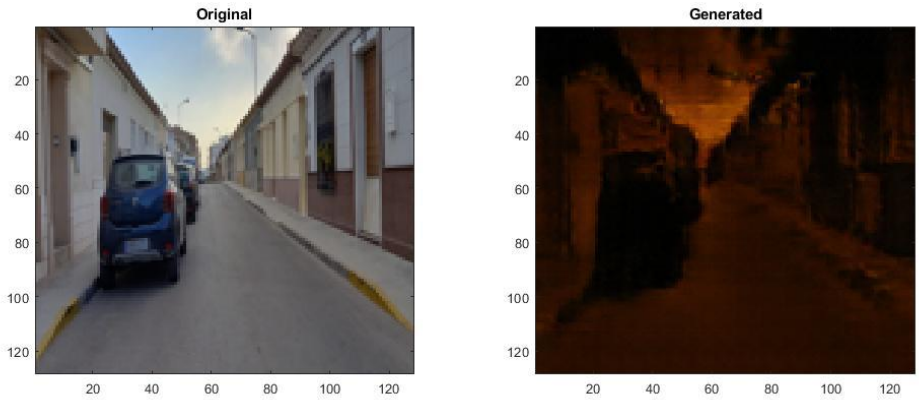


Figura 49: Conversión día-noche 1 (200 épocas).

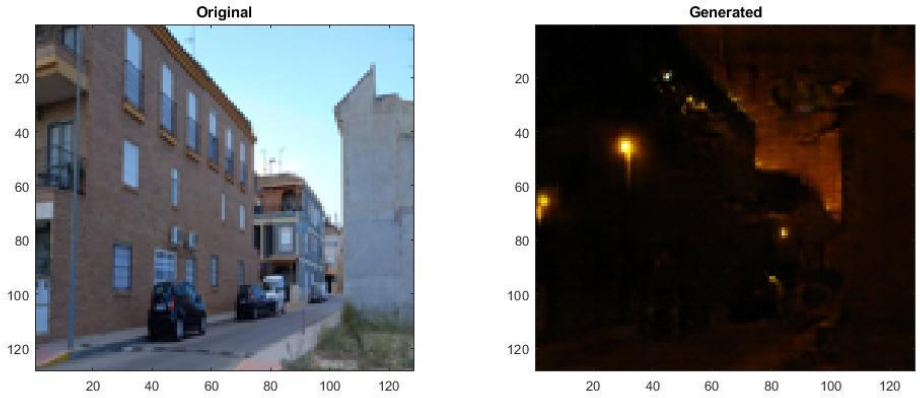


Figura 50: Conversión día-noche 2 (200 épocas).

Noche-día

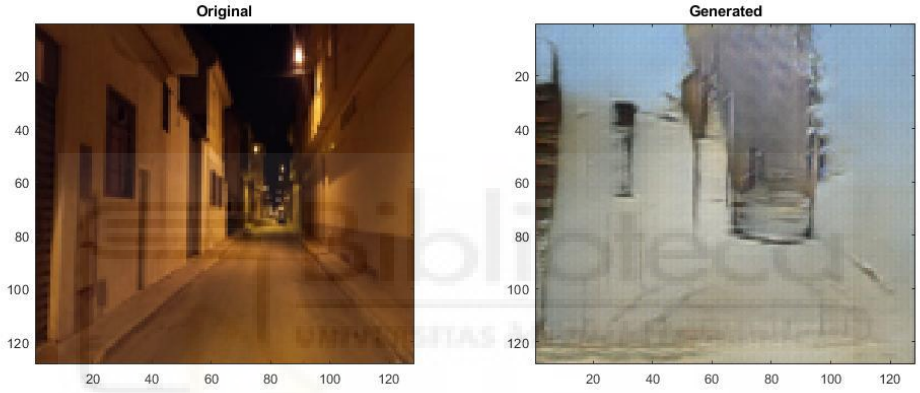


Figura 51: Conversión noche-día 1 (200 épocas).

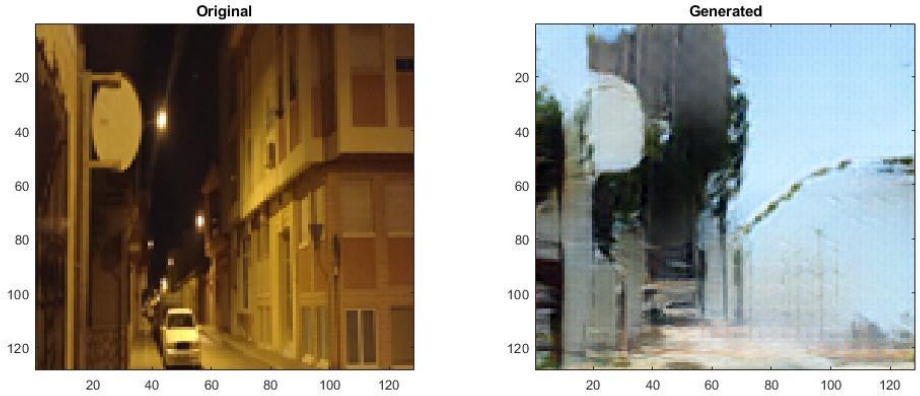


Figura 52: Conversión noche-día 2 (200 épocas).

A la vista de estos resultados, podemos apreciar que la conversión de condiciones de iluminación en exteriores es más compleja que en interiores, sobre todo cuando tratamos de generar imágenes diurnas a partir de imágenes nocturnas, esto se debe a que la red se encuentra con una mayor falta de información en las zonas oscuras siendo

incapaz en muchos casos de discernir que elementos se encuentran en esas zonas tendiendo por ello a generar una mayor cantidad de artefactos e inconsistencias en las imágenes. En las pruebas podemos apreciar que existe cierta disminución de estos errores en los modelos entrenados con 200 épocas respecto a los entrenados con 100 épocas, no obstante, al igual que en el entrenamiento con imágenes en interiores no hemos alcanzado un punto óptimo de entrenamiento, por lo que no sabemos si podemos llegar a una calidad aceptable con los modelos para imágenes en exteriores simplemente aumentando las iteraciones del entrenamiento, o si sería necesario algún tipo de procesado previo de las imágenes que resalten características ocultas en las zonas más oscuras de las imágenes. Por todo ello es necesario continuar investigando estas conversiones.

5 Conclusiones y trabajos futuros

El hecho de que durante el entrenamiento no hayamos alcanzado un punto en el que los artefactos y defectos en las imágenes producidas lleguen a desaparecer del todo o comiencen a incrementarse de nuevo, nos indica que no hemos encontrado un punto de entrenamiento óptimo y que podríamos continuar entrenando los modelos durante más épocas consiguiendo mejoras sustanciales en la cantidad de la salida, por lo que las 500 épocas en las que se ha realizado el entrenamiento no son suficientes para alcanzar dicho punto, no obstante por limitaciones de hardware y de tiempo no se continuará su búsqueda.

Para finalizar este proyecto, evaluaremos el alcance del trabajo realizado y aportaremos algunas ideas de cómo mejorar los resultados obtenidos y como continuar con el proyecto.

Respecto a los resultados de este trabajo, se ha probado que las redes generativas antagónicas son útiles para realizar cambios en ciertos elementos de imágenes que deseemos, desde convertir una naranja en una manzana a convertir el día en la noche. Se ha conseguido encontrar un modelo de red GAN adecuado para el objetivo inicial, que es el de obtener una red neuronal capaz de multiplicar automáticamente la variedad de imágenes de bases de datos más sencillas, que aplicadas posteriormente al campo de la navegación de vehículos autónomos podrían aumentar la robustez de esta.

A parte de las metas de carácter general del proyecto, también ha supuesto un aprendizaje personal, ya que es el primer acercamiento al desarrollo de estas tecnologías sobre las que había leído vagamente pero prácticamente sin profundizar.

Por último, para seguir trabando y mejorar los resultados obtenidos daré algunas ideas que considero pueden ser de utilidad:

- Realizar un entrenamiento más intensivo mediante hardware más potente, alcanzando no menos de 1000 épocas que es para lo que está diseñado el modelo, siendo así más probable hallar un punto óptimo.
- Utilizar bases de datos con mayor cantidad de imágenes y de entornos más variados, sobre todo de exteriores.
- Evaluar el impacto de hardware dedicado a inteligencia artificial en el entrenamiento del modelo, como el que incluyen ciertas tarjetas gráficas de nueva generación.
- De cara a algo más cercano a una aplicación práctica realizar un programa que sea capaz de reconocer el estado de iluminación de las imágenes y automáticamente llame al modelo pre entrenado de la red cycleGAN que corresponda para realizar las conversiones pertinentes.

Bibliografía

[1] Wikipedia, «Wikipedia,» 12 Abril 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Red_generativa_antag%C3%B3nica. [Último acceso: 10 Agosto 2021].

[2] Wikipedia, «Wikipedia,» 15 mayo 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico. [Último acceso: 10 agosto 2021].

[3] A. Gonzalez, «Cleverdata,» 20 Febrero 2021. [En línea]. Available: https://cleverdata.io/que-es-machine-learning-big-data/. [Último acceso: 15 Agosto 2021].

[4] IBM, «IBM,» 20 Enero 2021. [En línea]. Available: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model. [Último acceso: 17 Agosto 2021].

[5] Wikipedia, «Wikipedia,» 2021 Enero 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Red_neuronal_convolucional. [Último acceso: 17 Agosto 2021].

[6] J. Barrios, «Health Big Data,» 1 Febrero 2019. [En línea]. Available: https://www.juanbarrios.com/redes-neurales-convolucionales/. [Último acceso: 18 Agosto 2021].

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell, Fully convolutional networks for semantic segmentation, In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[8] A. Pronobis, «COLD,» 9 Septiembre 2020. [En línea]. Available: https://www.cas.kth.se/COLD/. [Último acceso: 18 Agosto 2021].

[9] O. Céspedes, "Localización de un robot móvil utilizando información visual y redes neuronales convolucionales", Trabajo de fin de grado, Grado en Ingeniería electrónica y automática industrial, Universidad Miguel Hernandez de Elche, Elche, Alicante, España, 2020.

[10] MIT, «MIT,» 2017. [En línea]. Available: https://groups.csail.mit.edu/vision/datasets/ADE20K/. [Último acceso: 3 Octubre 2021].

[11] Stanford, «DAGS,» 2009. [En línea]. Available: http://dags.stanford.edu/projects/scenedataset.html. [Último acceso: 3 Octubre 2021].

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aron Courville, And Yoshua Bengio. Generative adversarial Networks, arXiv e-prints, page arXiv:1406.2661, Jun 2014.

[13] Mehedi Mirza and Simon Osindero, Conditional generative adversarial nets, CoRR,abs/1411.1748, 2014.

[14] J. Hui, «Medium.com,» 3 Junio 2018. [En línea]. Available: https://jonathan-hui.medium.com/gan-cgan-infogan-using-labels-to-improve-gan-8ba4de5f9c3d. [Último acceso: 20 Agosto 2021].

[15] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim, Learning to Discover Cross-Domain Relations with Generative adversarial Networks. arXiv e-prints, page arXiv:1703.05192, Mar 2017.

[16] D. Azuar, "Transformación de emociones mediante redes generativas antagónicas", Trabajo de fin de grado, Grado en Ingeniería Informática, Universidad de Alicante, Alicante, España, 2019.

[17] G. Perenau, Icgan official repository, https://github.com/Guim3/IcGAN, 2015.

[18] Jun-Yan Zhu, Taesung Park, Philip Isola, Alexei A. Efros, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, arXiv:1703.10593v7 [cs.CV], 2017.

[19] R. Vijay, «Towards data science,» 15 Noviembre 2019. [En línea]. Available: https://towardsdatascience.com/image-to-image-translation-using-cyclegan-model-d58cfff04755. [Último acceso: 20 Agosto 2021].

[20] Li, Chuan, and Michael Wand, Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In Computer Vision – ECCV 2016, edited by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, 9907:702–16. Cham: Springer International Publishing, 2016.

[21] Paperswithcode, «Paperswithcode,» 21 Noviembre 2016. [En línea]. Available: https://paperswithcode.com/method/patchgan. [Último acceso: 1 Septiembre 2021].

[22] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros y U. Berkeley, «Guithub,» 2017. [En línea]. Available: https://junyanz.github.io/CycleGAN/. [Último acceso: 2 septiembre 2021].

[23] Guim Perarnau, Joost van de Weijer, Bodgan Raducanu and Jose M. Álvarez, CoRR, abs/1611.06355, 2016.