UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Generación de imágenes sintéticas de interiores utilizadas por robots móviles"

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

Alberto Embarcadero Ruiz

TUTOR:

Dr. Jesús Savage Carmona
Facultad de Ingeniería, UNAM

Ciudad Universitaria, CD. MX. enero, 2022

Agradecimientos

- A mi hermano Daniel por toda la ayuda en el desarrollo de las ideas para este trabajo.
- A la Universidad Nacional Autónoma de México por proporcionarme una educación invaluable.
- Al Dr. Jesus Savage por el apoyo que me otorgó durante este posgrado.
- Al Maestro Carlos Adrián por toda la ayuda proporcionada con los servidores.
- A los integrantes del laboratorio de Biorobotica.
- Al CONACYT por los recursos otorgados para poder cursar este posgrado.
- Al proyecto DGAPA-PAPIIT IG101721
- Al proyecto PAPIIT IA106520 "Visión activa y robótica basada en comportamientos en el desarrollo de vehículos autónomos".

Dedicatoria

Este tesis esta dedicada a:

- A mis padres, Pascual y Raquel, por todo el amor, apoyo, paciencia y enseñanzas que siempre me han dado para poder llegar hasta aquí.
- A mi hermano Daniel, que no solo me apoyó en la maestría y en esta tesis, sino que siempre me apoya en la vida.
- A mi tía Güerita, sin todo el apoyo que me dio nunca hubiera logrado llegado hasta aquí.

Resumen

La generación de imágenes sintéticas haciendo uso de redes neuronales es una área que se encuentra en constante desarrollo. Sus aplicaciones para la robótica son muy diversas, algunas de ellas son el aumentado de datos o la predicción de movimientos.

La tarea que se busca resolver en este trabajo, consiste en generar una imagen sintética a partir de una imagen de referencia, modificando el ángulo de la toma. Resolver esta tarea presenta diferentes retos, uno de ellos es el poder producir información que no existe en la imagen original. También se debe realizar la modificación del ángulo, con lo cual, las estructuras de la imagen se tienen que desplazar acorde a un valor indicado.

Para poder resolver esta tarea se optó por utilizar modelos generativos, más particularmente GANs (Generative Adversarial Networks), ya que estos tipos de redes producen resultados realistas con un gran acabado visual.

Existe una gran cantidad de arquitecturas GANs propuestas para la resolución de diversas tareas. En este trabajo se propone hacer uso de 3 aproximaciones diferentes para resolver la tarea planteada. Se experimentaron con diferentes variaciones para cada una de las aproximaciones, por lo que a lo largo del trabajo se presentan 14 diferentes modelos que buscan efectuar el cambio de ángulo de una imagen.

Para la obtención de los datos empleados se revisaron diferentes conjuntos de datos, de entre ellos se eligió el que mejor se adaptaba a la tarea. De este conjunto se extrajeron las imágenes con las características necesarias para poder realizar el desarrollo de este trabajo, obteniendo así una gran cantidad de imágenes reales con variaciones en los ángulos de la toma.

Además del entrenamiento de los diferentes modelos, se modificaron e implementaron dos de las métricas más populares para la evaluación de modelos generativos, con las cuales podemos otorgar una calificación de forma automatizada y objetiva a los diferentes modelos entrenados, esto nos permite hacer una comparación entre ellos para poder determinar cuáles fueron los mejores modelos entrenados.

Al evaluar los 14 modelos entrenados se obtuvieron algunos que presentan una buena calidad visual, así como un buen rendimiento en las dos métricas empleadas, de entre ellos, 2 son los que presentan los mejores resultados. Estos modelos son capaces de generar imágenes sintéticas con el cambio de ángulo deseado, otorgando imágenes con gran acabado visual.

Índice general

CAPÍTULO 1

Introducción

El desarrollo de los robots móviles involucra distintas áreas de estudio, las cuales se encargan de mejorar tanto el software como el hardware para que los robots puedan llevar a cabo tareas cada vez más complejas. Una de las tareas que un robot debe ser capaz de cumplir es la de poder reconocer el lugar donde se encuentra. Para completar esta tarea se han desarrollado diversas técnicas de navegación y localización las cuales utilizan sensores para conocer el ambiente en el que se encuentran, estos sensores pueden ser de ultrasonido, láser, cámaras entre otros.

Para el reconocimiento por medio de imágenes, actualmente se utilizan técnicas de aprendizaje profundo para lograrlo [1] [2] [3] [4]. Para poder usar estas técnicas es necesario contar con una gran cantidad de datos para llevar a cabo un correcto entrenamiento de los modelos. Tener disponible la cantidad adecuada de imágenes para el entrenamiento es en sí mismo un problema, ya que para obtener imágenes útiles, se tiene que realizar la captura y preprocesamiento de cada una de ellas.

Con una creciente necesidad de tener más datos para poder entrenar modelos confiables, se han desarrollado diversas técnicas que permiten el acrecentamiento de datos, es decir, generación de datos sintéticos a partir de datos reales [5]. Estos datos sintéticos siguen la distribución de los datos reales por lo que conservan características más relevantes variando otras menos importantes. Una de las ventajas del uso de imágenes sintéticas es el ahorro de recursos en la captura y preprocesamiento de imágenes en el mundo real, sin embargo, se requiere hacer uso de otras técnicas en el procesamiento de las imágenes para hacer imágenes sintéticas útiles para la resolución de alguna tarea.

1.1. Justificación

En diversas situaciones la logística o el tiempo no permiten recolectar una cantidad suficiente de imágenes para realizar de forma adecuada el entrenamiento de algún modelo. Es por ello que en estos casos resulta conveniente recurrir a la generación de imágenes sintéticas para aumentar la cantidad de datos disponibles para el entrenamiento. Estas imágenes poseen características similares a las de las imágenes originales, sin embargo se producen algunas variaciones que permiten entrenar modelos que sean capaces de generalizar correctamente.

En este trabajo se propone entrenar un modelo que tome un conjunto de imágenes obtenidas directamente de una habitación (imágenes reales) y por medio de una GAN (Generative Adversarial Network) generar imágenes sintéticas, las cuales a partir de una imagen real sean capaces de modificar el ángulo vertical y horizontal de la toma, para así aumentar la cantidad de imágenes disponibles.

1.2. Hipótesis

Se propone la siguiente hipótesis:

"Es posible entrenar un modelo generativo utilizando GANs, el cual a partir de una imagen real sea capaz de generar imágenes sintéticas realistas, las cuales varían el ángulo de la toma de acuerdo a un parámetro de entrada."

1.3. Objetivos

Con la finalidad de cumplir la hipótesis planteada, el objetivo principal de esta tesis es:

- Entrenar un modelo generativo que produzca imágenes sintéticas realistas variando los ángulos de la toma.

Para poder cumplir con este objetivo es necesario llevar a cabo los siguientes objetivos secundarios:

- Probar diferentes arquitecturas que sean capaces de producir un modelo generativo con las características planteadas en este proyecto.
- Comparar las imágenes generadas por los diferentes modelos y seleccionar el mejor modelo.

1.4. Estructura de la tesis

Este documento se estructura de la siguiente manera:

- En el presente capítulo define el problema y los objetivos de este trabajo.
- En el capítulo 2 se encuentra el marco teórico, ahí se abordan los conceptos de los modelos discriminativos y los modelos generativos, así como el funcionamiento de las redes neuronales.
- El capítulo 3 aborda el estado del arte y la descripción de las arquitecturas empleadas como base en la generación de los modelos obtenidos, así como una revisión de las métricas más populares para la evaluación de los modelos generativos.
- En el capítulo 4 se encuentra contenido el desarrollo, definición e implementación de las arquitecturas empleadas en esta tesis, así como de las métricas que se emplean en la evaluación de los modelos entrenados.
- En el capítulo 5 se muestran los resultados de los diferentes modelos entrenados y una comparación más detallada de los mejores modelos obtenidos.
- Por último en el capítulo 6 se presentan las conclusiones obtenidas y el trabajo que se puede desarrollar posteriormente.

CAPÍTULO 2

Marco teórico

En este capítulo se abordan los conceptos teóricos que dan sustento a esta tesis. Se parte de la definición de una imagen sintética y el aumentado de datos. Posteriormente se describen los conceptos probabilísticos que se utilizan en las tareas de clasificación y generación, además de abordar la el funcionamiento de diferentes capas empleadas en las redes neuronales.

2.1. Imágenes sintéticas

Antes que nada, para hablar de imágenes sintéticas es necesario definir ambos conceptos "imagen" y "sintético":

- «Una imagen se puede definir como una función bidimensional $f(x, y)$ donde x y y son coordenadas espaciales y la amplitud de cada par coordenado (x, y) es llamado la intensidad o nivel de gris de la imagen en ese punto.» (Gonzalez y Woods, 2002, p. 1) [6]. Para imágenes de color se cuenta con una intensidad o nivel para cada una de las componentes, dependiendo del espacio de color en el que esté definida la imagen.
- Sintético se refiere a

  1. adj. Perteneciente o relativo a la síntesis.
  2. adj. Que procede componiendo, o que pasa de las partes al todo.
  3. adj. Dicho de un producto: Que se obtiene por procedimientos industriales y que reproduce la composición y propiedades de uno natural.

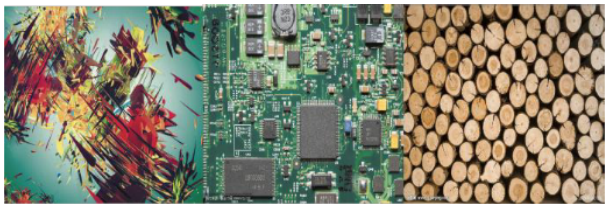
  (Real Academia Española, s.f.) [7].

Por lo tanto las imágenes sintéticas se refieren a imágenes que no son obtenidas directamente de la realidad, es decir, que no fueron realizadas mediante la captura de una escena en el mundo. Estas imágenes fueron procesadas de alguna manera generando imágenes nuevas y en algunos casos imposibles de obtener de otro modo, como es el caso de efectos especiales o renderizados hechos completamente por ordenador.

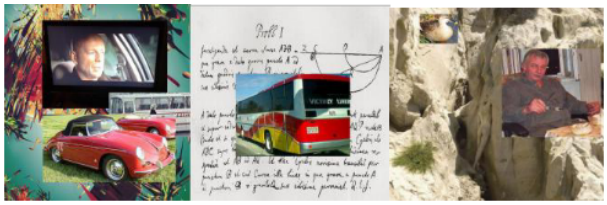
2.1.1. Imágenes sintéticas para el aumentado de datos

La creación de imágenes sintéticas para realizar aumentado de datos hace uso de diversas técnicas para generar datos nuevos que permitan ampliar el conjunto de imágenes disponibles para usar en el entrenamiento de un modelo. Algunas de estas técnicas consisten en aplicar transformaciones a las imágenes, tales como recortar un objeto y posicionarlo en fondos diferentes, con lo cual, se crean variaciones de un mismo objeto en diferentes ambientes [8]. Este tipo de técnicas pueden generar una gran cantidad de datos sintéticos, además de mejorar el

rendimiento de los modelos con diferentes fondos y hacer el reconocimiento de objetos sensible a diferentes escalas.



(a) Ejemplos de imágenes de fondo.



(b) Ejemplos de imágenes generadas con etiquetado artificial.

Figura 2.1: Ejemplo de imágenes en diferentes fondos (Rao *et al.*, 2017 [8])

Otra técnica consiste en aplicar transformaciones a la imagen, tales como escalado, rotación, espejo, perspectiva, variación de brillo y contraste, color, entre otras. Con lo cual al entrenar modelos de clasificación con estas técnicas, estos tienden a ser más robustos contra este tipo de transformaciones [9].

Figura 2.2: Ejemplo de transformaciones realizadas con la biblioteca Albumentations (Buslaev *et al.*, 2020 [9])

Por otro lado existen algunas otras técnicas que lo que pretenden es hacer imágenes completamente generadas por un ordenador, es decir, que no toman directamente fragmentos de las imágenes reales, sino que más bien modelan el espacio en el que fueron tomadas las imágenes para después generar nuevas imágenes. Algunas de estas, realizan una representación del mundo en 3D con lo que posteriormente se puede navegar por el mundo en un simulador y obtener diferentes imágenes sintéticas, las cuales corresponden con el modelado de la realidad, sin embargo,

al realizar el mapeado de las imágenes a un mundo 3D se generan artefactos tanto en texturas como en modelos. Esta técnica es comúnmente empleada para generar simuladores donde se pueda realizar una navegación 3D como en Gibson [10] o en Replica [11].

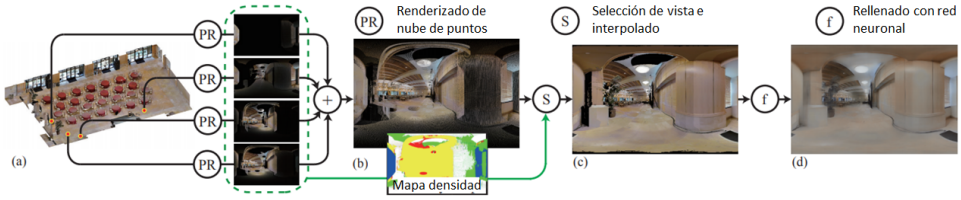


Figura 2.3: Flujo de sintetizado de imágenes en Gibson Dataset (Xia *et al.*, 2018 [10])

Otra técnica que se emplea para la generación de imágenes sintéticas, es obtener un modelo generativo que permita inferir la distribución conjunta del espacio de las imágenes, lo cual, permite generar imágenes que pertenecen a la distribución, pero que no necesariamente se encuentran en el conjunto de entrenamiento. Un ejemplo de esto son las Generative Adversarial Networks propuestas por Goodfellow *et al.* en 2014 [12]. Actualmente el desarrollo de arquitecturas GANs ha sido muy diverso y se han propuesto diferentes modelos para la resolución de diversas tareas. En la sección 3.1 se abordará más a detalle el funcionamiento y algunas arquitecturas de las GANs .

Figura 2.4: Rostros generados por una GAN (Karras *et al.*, 2018 [13])

2.2. Modelos probabilistas

Tratar de modelar un fenómeno del mundo real por medio de aproximaciones probabilistas ha sido un enfoque usado no solo en el aprendizaje de máquina, sino en la ciencia en general, ya que modelar a la perfección los fenómenos del mundo que nos rodea resultaría muy complicado y en la mayoría de los casos es un problema intratable. Es por ello que con el apoyo de los modelos probabilistas podemos modelar un fenómeno a partir de datos obtenidos y encontrar relaciones entre ellos, las cuales nos ayudan a comprender y explicar el comportamiento del fenómeno estudiado.

« *El problema de buscar patrones en datos ha sido fundamental y tiene una larga y exitosa historia... El campo de reconocimiento de patrones está relacionado con encontrar de manera automática regularidades en los datos haciendo uso de algoritmos computacionales y haciendo uso de estas regularidades para poder clasificar los datos dentro de alguna categoría.*» (Bishop, 2006, p. 1) [14]

Los modelos probabilistas nos ayudan a tratar tres fuentes de incertidumbre [15]:

- Propiedades estocásticas inherentes del sistema modelado: Modelos en los cuales se tienen comportamientos aleatorios.
- Observabilidad incompleta: cuando no se pueden observar todas las variables un modelo determinista puede parecer estocástico debido a la falta de información.
- Modelado incompleto: Cuando al realizar el modelo se tiene que descartar información observada, esta información resulta incierta para la predicción del modelo.

El uso de modelos probabilistas en aprendizaje de máquina y aprendizaje profundo nos permiten hacer aproximaciones a las distribuciones de datos y tratar con los problemas de incertidumbre de estos datos, con lo cual podemos generalizar patrones encontrados para posteriormente emplearlos en la resolución de alguna tarea específica. Las dos principales tareas que se realizan son clasificación y generación de datos.

2.2.1. Modelos discriminativos

La principal tarea de un modelo discriminativo es:

- Dada una entrada $\bar{x}$ poder asignar una clase k de C_k clases disponibles, con esto se modela $P(C_k|\bar{x})$, es decir, la probabilidad de pertenecer a una clase dada una entrada.

Para ello los modelos discriminativos pueden obtenerse haciendo uso de diferentes modelos de clasificación de acuerdo a las clases disponibles:

- Dos clases: La tarea a resolver se plantea como una clasificación binaria, dada una entrada $\bar{x}$ pertenece o no pertenece a una clase. Esto en su más simple representación se puede ver como la función:

  $$y(\bar{x}) = W^T\bar{x} + w_0$$

  Donde W representa un vector de pesos y w_0 es el sesgo de la función. Esta función se emplea como una frontera de decisión, donde si $y(\bar{x})$ supera cierto umbral se dice que la clase pertenece a la categoría C_1 y en caso contrario pertenece a la categoría C_0.

- K clases: considerando una extensión del modelo anterior, pero teniendo C_k clases con $k > 2$ se puede realizar una función discriminante para cada par de clases lo cual se conoce como un "clasificador uno contra uno".

  $$y_k(\bar{x}) = W_k^T\bar{x} + w_{k0}$$

  El problema de este tipo de clasificación es que se generan regiones donde no se puede tomar una decisión si se considera cada una de las funciones por separado, ya que se

provocan regiones de no pertenencia a ninguna clase o de pertenencia a más de una clase. Para solucionar este problema se realiza otra manera de tomar la decisión de pertenencia a una clase, se dice que $\bar{x}$ pertenece a la clase C_k si $y_k(\bar{x}) > y_j(\bar{x})$ para toda $j \neq k$, es decir, la clase asignada será la clase con la mayor probabilidad de pertenencia.

- Multiclase: Esta clasificación resulta muy similar al caso anterior, sin embargo no solo se asigna una clase a una entrada, se dice que $\bar{x}$ pertenece a toda C_j si $y_j(\bar{x}) > \alpha$, donde α representa un umbral de decisión.

2.2.2. Modelos generativos

Los modelos generativos realizan el modelado de un fenómeno utilizando su distribución conjunta $P(\bar{x}, \bar{y})$, con ello posteriormente es posible generar datos que pertenecen a la distribución aprendida. Estos nuevos datos obtenidos asemejan el comportamiento real del fenómeno, sin embargo, no se puede garantizar que todos los datos generados por la función sean datos que pudieron ser obtenidos de muestras reales.

Este tipo de modelos tienen un mayor costo computacional para su obtención, ya que resulta imposible, en casi todos los escenarios, contar con todas las muestras de la distribución, razón por la cual se emplea únicamente una aproximación a la distribución conjunta a partir de los datos que sí se pueden obtener y que pertenecen a dicha distribución.

Estos modelos resultan difíciles de evaluar, ya que a diferencia de los modelos discriminativos no existe una etiqueta objetivo, por lo que la evaluación depende de poder decidir si los datos son o no son parte de la distribución dada. Para ello en el caso de la evaluación de los modelos que generan imágenes se han propuesto diferentes métricas que buscan calificar el desempeño de los modelos generativos a partir de una cantidad significativa de muestras, en la sección 3.2 hablaremos más a detalle de las más relevantes.

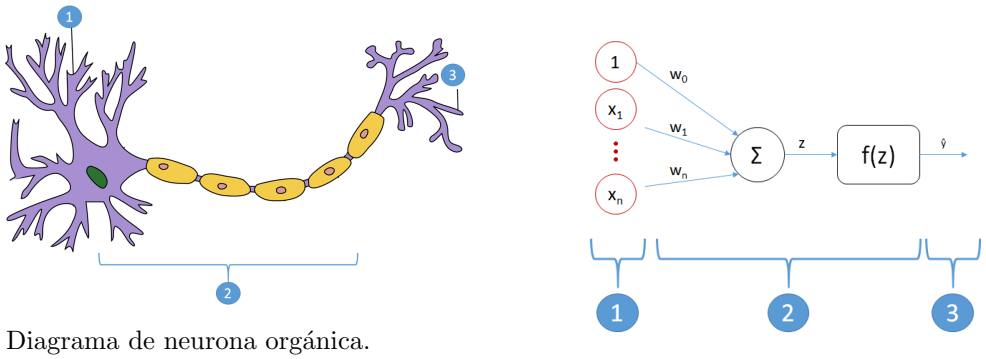
2.3. Redes neuronales

Los modelos de redes neuronales tienen su inspiración en el funcionamiento biológico del cerebro. Una red neuronal natural como la que tenemos en nuestro cerebro se compone de neuronas interconectadas, cada una de estas neuronas reacciona de manera diferente a los estímulos de entrada y produce una señal de salida dependiendo del proceso químico que se desarrolló en ella. La estructura general de una red neuronal artificial es la misma, se compone de diferentes neuronas conectadas entre sí, y cada una de las neuronas recibe una o más señales de entrada $\bar{x}$ y de acuerdo a sus pesos W y sesgo w_0 pasan por una función de activación $f(z)$ para generar una salida.

$$\hat{y} = f\left(W \cdot \bar{x} + w_0\right)$$

En 1958 Frank Rosenblatt propuso el perceptrón como un modelo computacional general en el que introdujo el uso de pesos numéricos y una forma especial de interconexión entre neuronas. Posteriormente Minsky y Papert trabajaron sobre este modelo creando su propio perceptrón del cual investigaron sus capacidades de cómputo y concluyeron que existen problemas que no

pueden ser resueltos con un solo perceptrón, sin embargo al generar una red de perceptrones las funciones que se pueden aproximar son más complejas [16].



(a) Diagrama de neurona orgánica. (Imagen tomada de Wikipedia)

(b) Diagrama de neurona artificial.

Figura 2.5: Comparación entre neuronas orgánicas y artificiales.
1) Es la entrada a la neurona.
2) Representa el proceso químico o procesamiento de la entrada.
3) La salida de la neurona.

Las redes neuronales constan de diversas capas, las cuales se clasifican en tres conjuntos:

- Capa de entrada: se refiere a la capa que se conecta directamente a los datos del fenómeno observado, estos datos pueden ser vectores, matrices o elementos de más dimensiones que caractericen al fenómeno.
- Capas ocultas: son las capas intermedias, de las cuales se desconocen los valores a los que se debe aproximar los pesos, no se tiene información en el ambiente para alimentar estas capas por lo que se utilizan algoritmos que permitan aproximar los valores óptimos a partir de la capa de entrada y la de salida.
- Capa de salida: Es la capa final, esta capa es la que entrega el resultado del modelo al problema dado, por lo que su tamaño y características varía de problema a problema. En un buen modelo la salida de esta capa debe coincidir con el valor esperado dada una entrada.

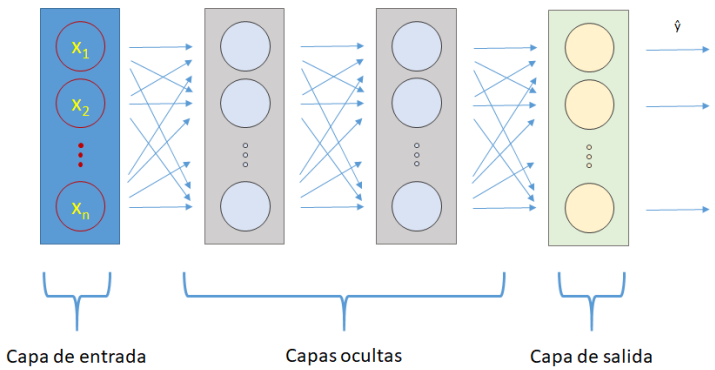


Figura 2.6: Ejemplo de capas en una red neuronal completamente conectada.

2.3.1. Entrenamiento de una red neuronal

Las redes neuronales requieren ajustar los pesos de sus neuronas para generar el modelo de la tarea a resolver. Para ello se utiliza comúnmente un entrenamiento supervisado, el cual consiste en tener datos de entrenamiento previamente etiquetados que permitan a la red evaluar una función de pérdida para así ir mejorando la aproximación al problema a resolver.

El algoritmo de descenso por gradiente se utiliza para minimizar una función, en el caso de las redes neuronales minimiza una función de pérdida. Para esto modifica los pesos W y el sesgo w_0 de acuerdo a dónde la pérdida descienda más rápido en un vecindario [17].

$$\theta[t+1] = \theta[t] - \alpha \nabla L(\theta[t])$$

Donde $L(\theta[t])$ es la función de pérdida, α es el factor de aprendizaje y $\theta[t] = \{W, w_0\}$ en la iteración t.

Para poder calcular el gradiente en la función de pérdida primero se tiene que hacer una propagación hacia adelante, la cual nos dará el valor de predicción de la red con los pesos actuales y una entrada $\bar{x}$. Para una red de n capas, esta propagación hacia adelante está dada por:

$$\begin{aligned}
\mathbf{a}^{\{1\}} &= \bar{\mathbf{x}}^{(i)} \\
\mathbf{z}^{\{2\}} &= \mathbf{W}^{\{1\}} \cdot \mathbf{a}^{\{1\}} + \mathbf{w_0}^{\{1\}} \\
\mathbf{a}^{\{2\}} &= \sigma(\mathbf{z}^{\{2\}}) \\
\mathbf{z}^{\{3\}} &= \mathbf{W}^{\{2\}} \cdot \mathbf{a}^{\{2\}} + \mathbf{w_0}^{\{2\}} \\
\mathbf{a}^{\{3\}} &= \sigma(\mathbf{z}^{\{3\}}) \\
&\vdots \\
\mathbf{z}^{\{n\}} &= \mathbf{W}^{\{n-1\}} \cdot \mathbf{a}^{\{n-1\}} + \mathbf{w_0}^{\{n-1\}} \\
\mathbf{a}^{\{n\}} &= \sigma(\mathbf{z}^{\{n\}}) \\
\hat{y}^{(i)} &= \mathbf{a}^{\{n\}}
\end{aligned} \tag{2.1}$$

El algoritmo de retropropagación busca el mínimo de una función de pérdida utilizando el método de descenso por gradiente. La combinación de pesos en las neuronas que minimizan el error de la función de pérdida es considerada una solución en el problema de aprendizaje [16].

Para aplicar el algoritmo de retropropagación se calcula el error de acuerdo a la función de pérdida $L(y, \hat{y})$ donde y es el valor real y $\hat{y}$ es el valor calculado por la red. Tomando la derivada de la función de pérdida respecto a la capa de salida

$$\delta_n = \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{n\}}}$$

Haciendo la derivada parcial de la función de pérdida respecto a la salida de la capa $n-1$:

$$\begin{aligned}
\delta_{n-1} = \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{n-1\}}} &= \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{n\}}} \frac{\partial \mathbf{z}^{\{n\}}}{\partial \mathbf{z}^{\{n-1\}}} \\
&= \delta_n \mathbf{W}^{\{n-1\}} \frac{\partial \sigma(\mathbf{z}^{\{n-1\}})}{\partial \mathbf{z}^{\{n-1\}}}
\end{aligned} \tag{2.2}$$

Y posteriormente la derivada con respecto a cualquier capa oculta j-ésima $0 < j < n-1$:

$$\begin{aligned}\delta_j &= \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{j\}}} \\ &= \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{n\}}} \frac{\partial \mathbf{z}^{\{n\}}}{\partial \mathbf{z}^{\{n-1\}}} \cdots \frac{\partial \mathbf{z}^{\{j+1\}}}{\partial \mathbf{z}^{\{j\}}} \\ &= \delta_{j+1} \mathbf{W}^{\{j\}} \frac{\partial \sigma(\mathbf{z}^{\{j\}})}{\partial \mathbf{z}^{\{j\}}}\end{aligned} \tag{2.3}$$

Calculamos las siguientes derivadas parciales:

$$\frac{\partial \mathbf{z}^{\{j+1\}}}{\partial \mathbf{W}^{\{j\}}} = \frac{\partial (\mathbf{W}^{\{j\}} \cdot \mathbf{a}^{\{j\}} + \mathbf{w_0}^{\{j\}})}{\partial \mathbf{W}^{\{j\}}} = \mathbf{a}^{\{j\}} \tag{2.4}$$

$$\frac{\partial \mathbf{z}^{\{j+1\}}}{\partial \mathbf{w_0}^{\{j\}}} = \frac{\partial (\mathbf{W}^{\{j\}} \cdot \mathbf{a}^{\{j\}} + \mathbf{w_0}^{\{j\}})}{\partial \mathbf{w_0}^{\{j\}}} = 1 \tag{2.5}$$

Por último calculamos las derivadas respecto a los pesos $W^{\{j\}}$ y sesgos $w_0^{\{j\}}$:

$$\frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{W}^{\{j\}}} = \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{j+1\}}} \frac{\partial \mathbf{z}^{\{j+1\}}}{\partial \mathbf{W}^{\{j\}}} = \delta_{j+1} \mathbf{a}^{\{j\}} \tag{2.6}$$

$$\frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{w_0}^{\{j\}}} = \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{z}^{\{j+1\}}} \frac{\partial \mathbf{z}^{\{j+1\}}}{\partial \mathbf{w_0}^{\{j\}}} = \delta_{j+1} \tag{2.7}$$

Como podemos ver al hacer la retropropagación se calcula el error primero en la capa de salida, posteriormente se puede calcular el gradiente de las demás capas en términos de las ya calculadas hasta llegar a la capa de entrada recordando que $\mathbf{a}^{\{1\}} = \bar{\mathbf{x}}^{(i)}$

Una vez calculado el gradiente para cada uno de las capas y pesos, se aplica la regla de actualización:

$$\mathbf{W}^{\{j\}}[t+1] = \mathbf{W}^{\{j\}}[t] - \alpha \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial \mathbf{W}^{\{j\}}}$$

Esta operación modifica los pesos de cada capa de acuerdo al gradiente calculado, con ello se busca optimizar los pesos para tener la mejor aproximación a la función que define el problema. α es un hiperparámetro conocido como "Learning Rate" o "Factor de Aprendizaje", el cual debe ser cuidadosamente seleccionado para evitar inestabilidad en el entrenamiento.

2.3.2. Tipos de capas

En las redes neuronales existen diferentes tipos de capas, las cuales son empleadas para ayudar a resolver diferentes tareas, a continuación se describirá el funcionamiento de las capas utilizadas en este proyecto.

Capa completamente conectada

La capa "completamente conectada" o "fully connected" es una capa en la cual todas las entradas se conectan con todas las neuronas, es decir, todas las posibles conexiones entre las entradas y las neuronas están presentes.

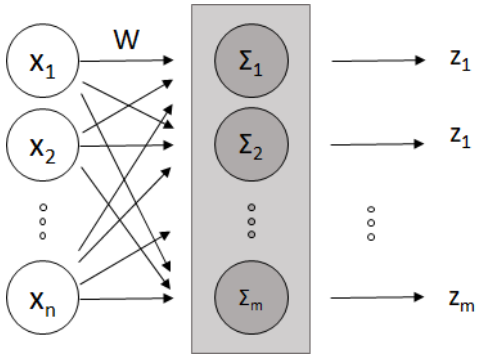


Figura 2.7: Ejemplo de capa completamente conectada con n entradas y m salidas.

Este tipo de capas cuentan con una matriz de pesos W en la cual se almacena el peso que aporta cada entrada a cada neurona, en total esta matriz contiene $n \times m$ elementos, donde n es el número de entradas a la capa y m el número de salidas de esta. Este tipo de capa es empleada principalmente en tareas de clasificación, sin embargo, para tareas de procesamiento de imágenes no resulta ser tan buena opción, ya que la cantidad de parámetros por capa resulta muy grande. Por otro lado la entrada de una capa completamente conectada es un vector unidimensional, por lo cual se tiene que transformar una imagen (matriz 2D) a un vector, lo que ocasiona que se pierda información de la relación espacial que se tiene entre pixeles vecinos.

Capa convolucional

Las capas convolucionales se presentan como una alternativa a las capas completamente conectadas para procesar imágenes [15], estas capas llevan a cabo una operación de convolución entre la imagen y un *kernel* o filtro. La operación de convolución se denota por el operador $*$ y está definida para funciones discretas como:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Esta operación puede ser reescrita como una correlación cruzada, lo cual hace que pierda su propiedad conmutativa [15], sin embargo, para términos de implementación en una capa convolucional esta propiedad no es usada y ahorra tiempo de cómputo al evitar invertir el *kernel*.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

El uso de convolución entre imágenes y filtros es una técnica usada comúnmente en procesamiento digital de imágenes para realizar filtros paso bajas o paso altas y así obtener características como bordes, esquinas, etc. [18]. La salida de una capa se puede ver como una imagen filtrada por el *kernel* de la capa, los valores del *kernel* son los pesos que se aprenden durante el entrenamiento.

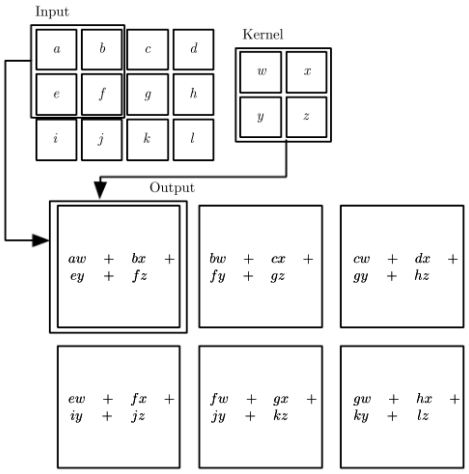


Figura 2.8: Convolución realizada en 2D sin invertir el *kernel* (Goodfellow *et al.*, 2016 [15]).

Como se puede observar en la figura 2.8 la operación se realiza tomando ventanas en la imagen del mismo tamaño que el *kernel* y se realiza la multiplicación elemento a elemento entre la ventana y el *kernel* para después sumarse y generar un valor de salida el cual corresponde al valor de la convolución para ese punto. Posteriormente, la ventana se desplaza por toda la imagen mientras que el *kernel* utiliza los mismos pesos para todas las ventanas, esto tiene como consecuencia que la cantidad de parámetros esté determinada por el tamaño del *kernel*. Las capas convolucionales pueden contar con más de un *kernel*, con lo cual generan no solo una imagen filtrada a la salida, sino un conjunto de imágenes del mismo tamaño, a estas salidas se les conoce como mapas de características y el número de mapas está determinado por un hiperparámetro de la capa.

Capa de reducción

Esta capa realiza una reducción de resolución espacial a la imagen realizando un submuestreo sobre la entrada. Existen diferentes operaciones para hacer esto, las más comunes son:

- MaxPooling: toma el valor máximo dentro de una vecindad y lo pasa a la salida.
- AveragePooling: realiza un promedio de los valores en una vecindad y entrega ese valor a la salida.

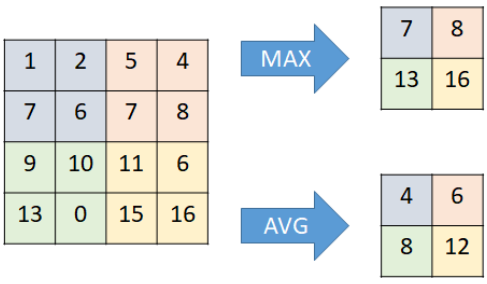


Figura 2.9: Ejemplo de resultado al aplicar MaxPooling (superior) y AveragePooling (inferior).

Normalización por lotes

Esta capa también conocida como «Batch normalization», se emplea para hacer normalización de los valores que se presentan como entradas en las capas ocultas. Hace uso de la media μ_B y la varianza σ_B de los valores de entrada con el fin de evitar el "Internal covariate shif", fenómeno que ocurre cuando las entradas de la red cambian su distribución, lo que provoca la disminución en la velocidad del entrenamiento. Además, permite minimizar la explosión de gradiente o el desvanecimiento del mismo. La operación que realiza se define como:

$$N = \frac{F - \mu_B}{\sqrt{\sigma_B^2 + \eta}}$$

η es un valor agregado a la función para evitar la división entre 0 y así dar estabilidad numérica a la función. Este tipo de normalización ayuda a unificar la distribución de los mapas de características y suaviza el gradiente, lo que ayuda a que el modelo sea mejor generalizando. [19].

2.3.3. Funciones de activación

Algunas de las funciones de activación más empleadas en las redes neuronales son:

- $ReLU$: Esta función se define como $ReLU(x) = max(0, x)$ lo que hace que el rango de esta función sea [0,∞). Esta función resulta muy útil para el procesamiento de imágenes, ya que los valores negativos son enviados a 0. Existen algunas variantes de esta función tales como:
  - $LeakyReLU(x) = \begin{cases} x, & \text{si } x \geq 0 \\ \alpha x, & \text{si } x < 0 \end{cases}$, con $\alpha > 0$ y constante.
  - $PReLU(x) = \begin{cases} x, & \text{si } x \geq 0 \\ \alpha x, & \text{si } x < 0 \end{cases}$, con $\alpha > 0$ y como valor aprendido.
  - $RReLU(x) = \begin{cases} x, & \text{si } x \geq 0 \\ \alpha x, & \text{si } x < 0 \end{cases}$, con $\alpha > 0$ y aleatoria.
  - $ELU(x) = \begin{cases} x, & \text{si } x \geq 0 \\ \alpha(e^x - 1), & \text{si } x < 0 \end{cases}$, con $\alpha > 0$ y constante.

  Todas estas variantes tienen una forma similar, sin embargo a diferencia de la función ReLU estas variantes no envían los valores negativos a 0 lo que permite evitar la desactivación completa de neuronas en las capas ocultas.

- Sigmoide: Esta función se define como $\sigma(x) = \frac{1}{1+e^{-x}}$ y su rango está entre (0,1) además de que su derivada está en función de ella misma $\sigma`(x) = \sigma(x)(1 - \sigma(x))$,por lo que resulta muy útil para realizar cálculos de los gradientes al momento de realizar el entrenamiento de las redes neuronales. Además gracias a que sus valores van de[0,1] se presta para realizar clasificación binaria.

- HTan: Es una tangente hiperbólica y se define como $HTan(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ esta función es muy similar a la sigmoide, sin embargo su rango está entre (-1,1) lo que hace que la función esté centrada en 0, de igual manera funciona bien para hacer la distinción entre dos opciones y en el caso de la generación de imágenes resulta muy útil como función de activación en la capa de salida.

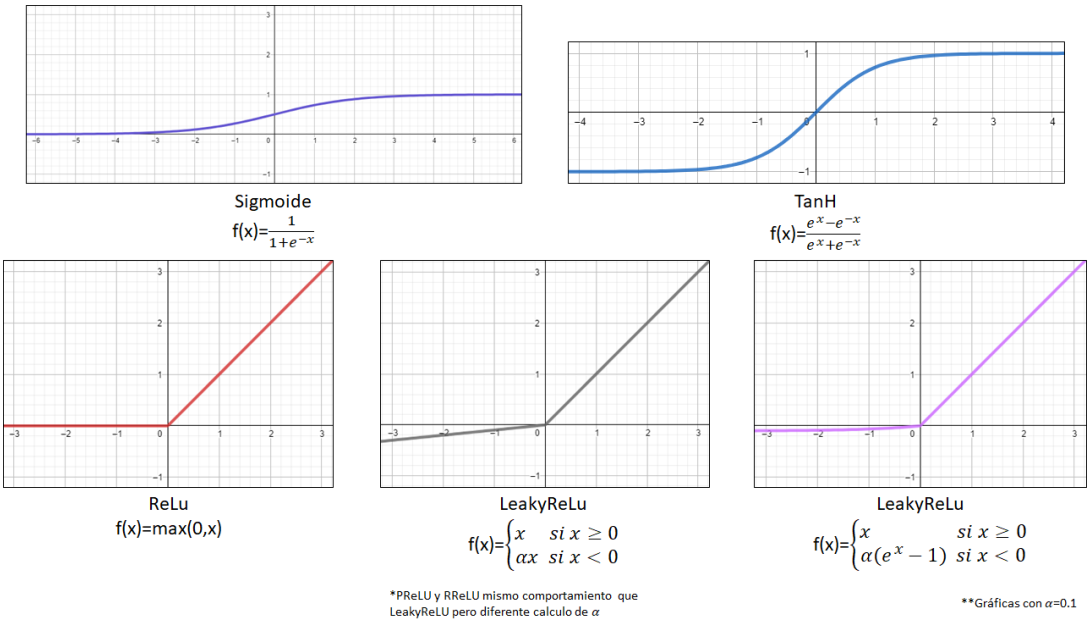


Figura 2.10: Gráficas de las funciones de activación.

CAPÍTULO 3

Estado del Arte

Las investigaciones en las redes neuronales se encuentran en constante desarrollo, es por ello que los avances en el estado del arte mejoran constantemente. Cada vez se generan arquitecturas nuevas que tienen como propósito poder modelar de manera más precisa, fenómenos más complejos y con ello poder resolver diferentes tareas de una mejor manera. En este capítulo se hablará de las arquitecturas GAN que permiten realizar un acercamiento a la resolución del problema planteado en esta tesis, así como las formas más comunes de evaluar los modelos generativos.

3.1. GAN

Las GAN reciben su nombre del inglés Generative Adversarial Network, se les llama redes generativas, ya que producen datos sintéticos siguiendo la distribución de los datos de entrenamiento. Se les llama redes adversarias por el tipo de entrenamiento que utilizan, en este tipo de entrenamiento se emplea un esquema de competencia entre dos redes, un generador $G(z)$ y un discriminador $D(I)$, donde z, en el caso más simple, es un vector de ruido e I es una imagen que puede ser real o sintética. El objetivo del discriminador $D(I)$ es ser capaz de diferenciar una imagen real de una imagen sintética, mientras que el generador $G(z)$ tiene como objetivo generar imágenes que sean capaces de engañar al discriminador, de modo que los datos sintéticos sean clasificados como reales.

Para lograr el entrenamiento se busca resolver una función mín máx que depende tanto del generador como del discriminador:

$$\mathcal{L}_{GAN} = \min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (3.1)$$

Esta función de pérdida utiliza la entropía binaria cruzada, ya que el discriminador solamente es capaz de utilizar dos etiquetas

- 0:sintética
- 1:real

Esta función se descompone en dos funciones de pérdida por cada una de las redes G y D. Para el discriminador se tiene:

$$\mathcal{L}_D = \max_D [\mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]] \quad (3.2)$$

En donde el primer término de la función se maximiza cuando las imágenes de p_{data} son clasificadas como reales. Por otro lado el segundo término de la función es maximizado cuando las imágenes sintéticas ($G(z)$) son clasificadas como tal. Es decir que el discriminador busca clasificar imágenes reales y sintéticas en la categoría a la que realmente pertenecen. Al maximizar

esta función y clasificar de forma correcta, la pérdida del discriminador este se vuelve menor, ya que se usa la entropía binaria cruzada.

Para el generador la función cambia un poco, ya que no todos los términos dependen de este, con lo que se tiene:

$$\mathcal{L}_G = \min_G[\mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))]] \tag{3.3}$$

Esta función busca minimizar que las imágenes sintéticas sean clasificadas como tal, sin embargo, esto genera gradientes planos y problemas de inestabilidad, por lo que se realiza un cambio en la función de pérdida en donde en lugar de buscar minimizar una función se trata de maximizar una función equivalente. Con lo que se obtiene:

$$\mathcal{L}_G = \max_G[\mathbb{E}_{z\sim p_z(z)}[log(D(G(z)))]] \tag{3.4}$$

Con este cambio se busca maximizar la probabilidad de que el discriminador clasifique una imagen sintética como real, es decir, que el generador sea capaz de engañar al discriminador. Esto busca maximizar la pérdida $\mathcal{L}_G$ al hacer imágenes sintéticas lo más parecidas a las reales.

El ciclo de entrenamiento de una GAN se puede realizar optimizando 3.2 primero y posteriormente 3.4 en cada paso del entrenamiento, con lo cual el discriminador primero aprende a diferenciar imágenes reales de sintéticas y posteriormente el generador aprende a engañar al discriminador, dando como resultado imágenes sintéticas semejantes a la distribución de las reales.

3.1.1. GAN

Las GAN fueron propuestas por Ian Goodfellow *et al.* en 2014 [12], en esta primera arquitectura la entrada es un vector de ruido y al entrenar el modelo se busca que la salida sea una imagen semejante a la distribución de las imágenes del conjunto de datos.

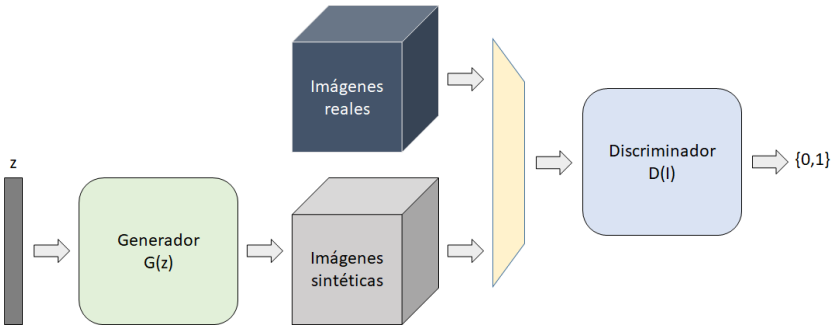


Figura 3.1: Componentes básicos de una GAN.

Esta arquitectura emplea la función de pérdida mostrada en la ecuación 3.1 para su entrenamiento. Como se puede apreciar en la figura 3.1 el generador toma como entrada únicamente el vector de ruido y produce imágenes sintéticas. El discriminador por su parte recibe como entrada una imagen, esta imagen puede ser real o sintética. Al realizar el entrenamiento conocemos la categoría a la que pertenece dicha imagen, ya que se toma directamente del conjunto de datos (real) o se toma del generador (sintética), conociendo esta etiqueta, es posible calcular la pérdida dependiendo del origen de la imagen y la respuesta que se obtenga del discriminador.

Esta arquitectura una vez entrenada puede generar imágenes parecidas a las del conjunto de datos de entrenamiento, sin embargo, no hay forma de condicionar la salida, ya que el generador depende únicamente del ruido a la entrada.

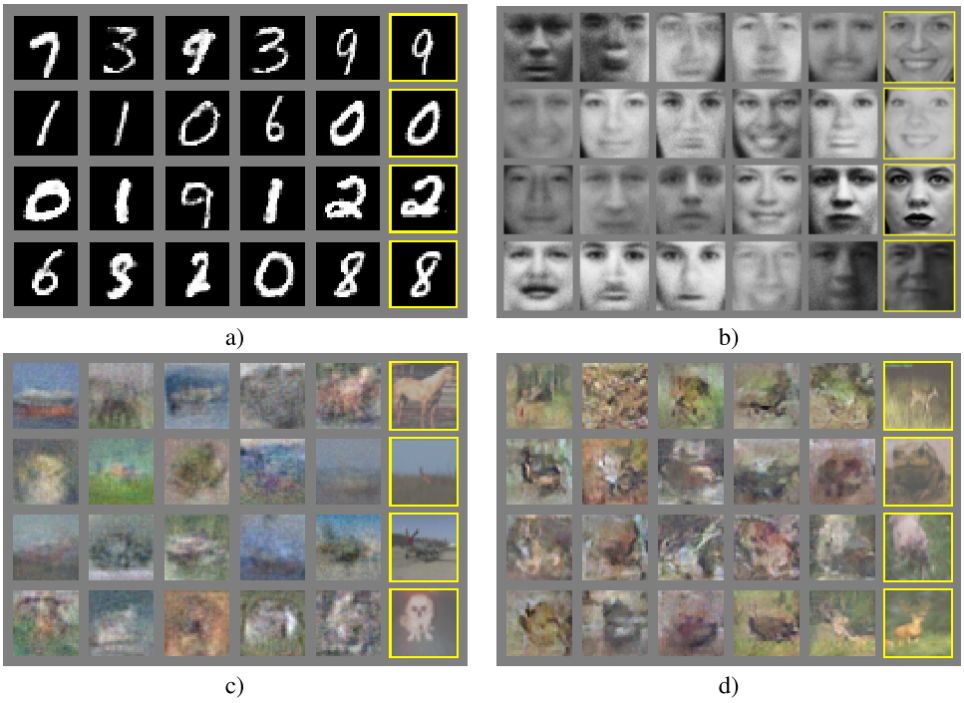
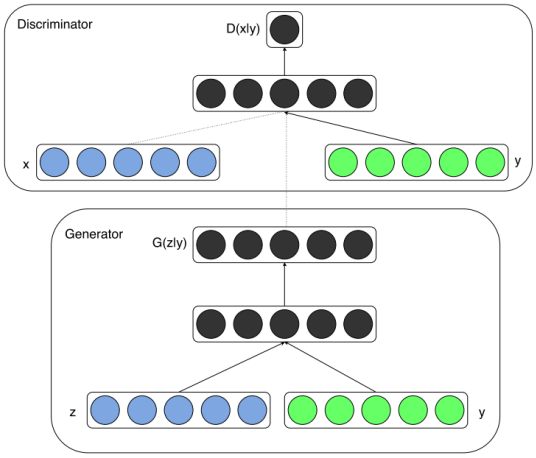


Figura 3.2: Ejemplos de imágenes generadas por una GAN (Goodfellow *et al.*, 2014 [12])

3.1.2. GAN condicional

La arquitectura cGAN (conditional GAN) fue propuesta por Mehdi Mirza y Simon Osindero en 2014 [20]. En esta arquitectura proponen usar no solo un vector de ruido como entrada del generador, además de este, agregan un parámetro adicional que permite indicar la clase de pertenencia de la imagen. Con esto se permite generar imágenes sintéticas pertenecientes a diferentes clases utilizando un mismo modelo. Así como se agrega el parámetro de clasificación al generador, es necesario agregarlo al discriminador, ya que ese parámetro ayudará a determinar si la imagen generada asemeja ser real dada una clase.



(a) Arquitectura de una cGAN.

(b) Imágenes generadas por una cGAN entrenada en MNIST.

Figura 3.3: Arquitectura y resultados de una cGAN (Mirza *et al.*, 2014 [20])

El uso de este parámetro adicional agrega mayor versatilidad a los modelos que pueden ser generados, ya que se puede controlar la clase de la que se quieren obtener datos sintéticos, como en la figura 3.3b en la cual se puede elegir el dígito que se desea generar de acuerdo a un parámetro de entrada.

En esta arquitectura se emplea una función similar obtenida a partir de la ecuación 3.1, pero dependiente de un parámetro de entrada y, el cual representa una clase:

$$\mathcal{L}_{cGAN} = \min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|y)))] \quad (3.5)$$

Con este parámetro condicional x y z dependen de la clase y para determinar la distribución de los datos y generar las imágenes sintéticas. El entrenamiento de esta red se lleva a cabo de la misma manera que en la GAN original, con la diferencia que al hacer el paso hacia adelante tanto en el generador como en el discriminador y se pasa como una entrada.

El generador al depender tanto del ruido como del parámetro de entrada puede generar:

- Diferentes imágenes utilizando un mismo parámetro, lo que se traduce en imágenes con diferente estilo, pero que pertenecen a una misma clase.(Fila de figura 3.3b)
- Diferentes imágenes a partir de un mismo ruido, esta cantidad de imágenes se acota por el número de clases posibles y se traduce en imágenes con un mismo estilo pero pertenecientes a diferentes clases.(Columna de figura 3.3b)

En el caso del discriminador el cambio que se realiza es el de agregar el parámetro condicional como una entrada y este funciona para determinar si la imagen es real o sintética para una clase específica.

3.1.3. Traducción imagen a imagen

En 2017 Isola *et al.* [21] propusieron utilizar como base una cGAN a modo de usar una imagen como condición en lugar de un parámetro numérico y, generando una imagen a partir de otra. Esto se conoce como traducción imagen-imagen.

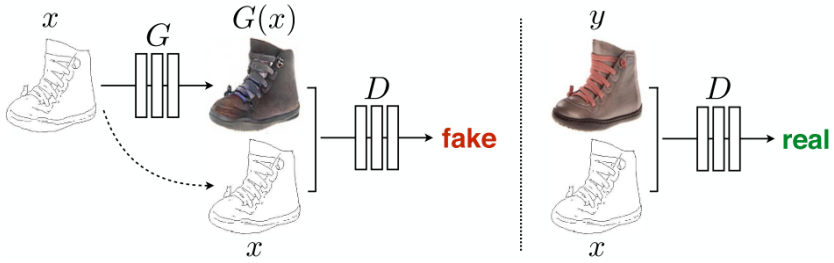


Figura 3.4: Diagrama de la arquitectura pix2pix (Isola *et al.*, 2017 [21])

Esta arquitectura toma una imagen de referencia como entrada para generar una imagen sintética. En esta arquitectura el vector de ruido z no es empleado, ya que la red aprende a ignorarlo, lo que lo vuelve innecesario. El uso de una imagen como referencia hace que esta arquitectura se pueda emplear en diferentes tareas, en las cuales la estructura de la imagen se conserva una vez realizada la traducción.

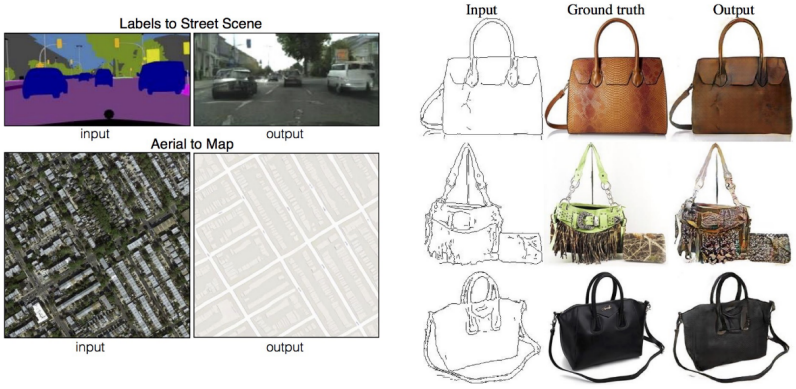


Figura 3.5: Ejemplos de imágenes generadas con pix2pix (Isola *et al.*, 2017 [21])

En el generador de esta arquitectura se emplea una estructura similar a la empleada en U-Net [22], donde al momento de codificar la imagen guarda los mapas de características intermedios antes de hacer un *pooling*, estos mapas posteriormente se concatenan al hacer *upsampling*, lo que permite que la información que se pudo perder durante la codificación sea rescatada y empleada para la generación de la imagen. Esta arquitectura genera muy buenos resultados cuando se requiere que la imagen generada mantenga la misma estructura de la imagen original, es decir, que las traducciones no modifiquen la posición u orientación de la imagen original.

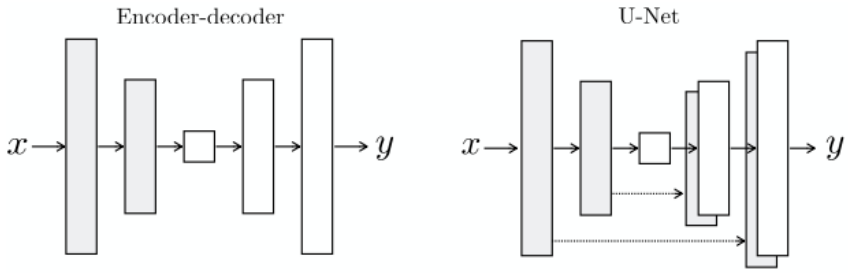


Figura 3.6: Arquitectura U-Net empleada en el generador (Isola *et al.*, 2017 [21]).

La función de pérdida que emplea esta arquitectura es similar a la empleada por la cGAN, sin embargo se incluye un término en la pérdida del generador, el cual hace uso de la norma $L1$ entre imágenes generadas e imágenes objetivo, con lo que se obtiene:

$$\mathcal{L}_{p2p} = \min_{G} \max_{D} \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{3.6}$$

Donde $\mathcal{L}_{L1}(G)$ es:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{3.7}$$

Siendo la norma $L1$ entre la imagen generada por G y la imagen objetivo tomada del conjunto de datos. Normalmente el uso de norma $L1$ o $L2$ en una GAN provoca imágenes borrosas como resultado, sin embargo en esta arquitectura al hacer uso de un generador tipo U-Net las imágenes logran buenos resultados utilizando la norma $L1$ según lo reportado en [21].

Otra de las modificaciones realizadas en la arquitectura se presenta en el discriminador. Este toma como entrada la imagen referencia que se introdujo en el generador y la imagen a evaluar, la cual puede ser real o sintética. Concatena ambas imágenes y las procesa por medio de capas convolucionales de forma que al final regresa un arreglo bidimensional con valores entre 0 y 1, cada uno de estos valores indica si una parte de la imagen es real o sintética, a esto se le conoce como PatchGAN. Este tipo de discriminador no clasifica la imagen completa como real o sintética, en lugar de ello da la opción de poder clasificar la imagen por ventanas y corregir de mejor manera las secciones que no se ven reales, ya que este método actúa como un tipo de pérdida en textura/estilo.

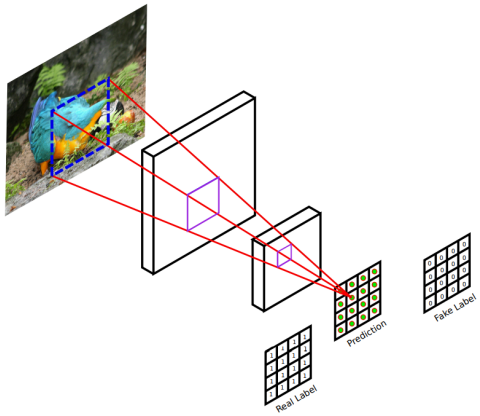


Figura 3.7: Ejemplo de PatchGAN (Demir *et al.*, 2018 [23])

3.1.4. StackGAN

Esta arquitectura fue propuesta por Han Zhang *et al.* en 2017 [24]. Fue diseñada para generar imágenes a partir de descriptores de texto, para lograrlo utiliza 2 GANs apiladas para generar imágenes realistas de 256x256 pixeles. Esto tiene como consecuencia distribuir un problema complicado, como es la generación de imágenes de gran tamaño, en dos subproblemas que resultan menos complejos.

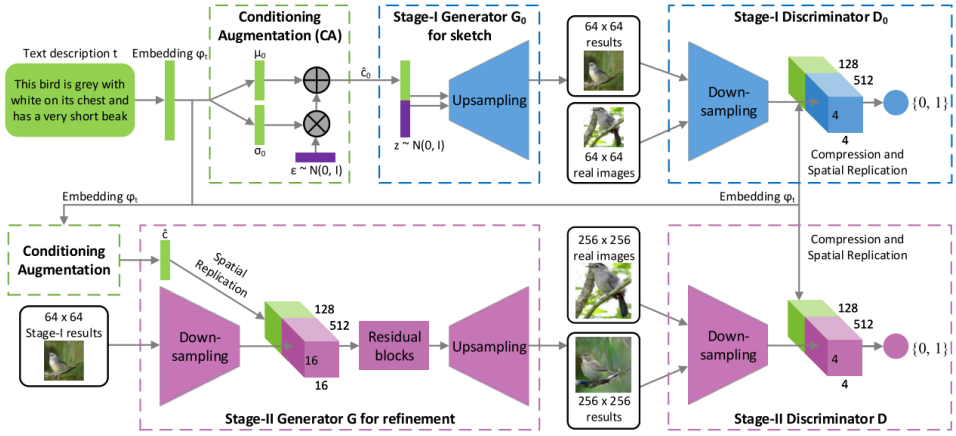


Figura 3.8: Arquitectura de StackGAN (Zhang *et al.*, 2017 [24])

En la primera GAN se obtienen imágenes en baja resolución(64x64) las cuales tienen la información más general de las imágenes, como formas y posiciones, pero no contienen algunos de los detalles más finos. En la segunda GAN se toman las imágenes producidas por la primera GAN y se realiza un refinado en el que se agregan detalles a las imágenes. Cada una de estas GAN se entrena de manera individual, es decir, primero se entrena completamente la primera y después en el entrenamiento de la segunda los pesos de la primera no se modifican.

Figura 3.9: Imágenes generadas por ambas fases empleadas en la arquitectura (Zhang *et al.*, 2017 [24])

Además de hacer uso de dos GANs apiladas en este artículo se propone una forma de procesar la entrada llamada *"Conditioning Augmentation"* con la cual se busca modificar la entrada pasando de un *encoding* del texto a valores de una distribución normal con media μ_0 y varianza σ_0 utilizando la divergencia de Kullback-Leibler.

Stage1 En la primera fase de esta arquitectura el generador se asemeja al de una cGAN. A partir de un vector de ruido y un parámetro condicional (descriptor de texto) genera una imagen realista. El discriminador funciona de la misma manera que en una cGAN, toma una imagen y el parámetro condicional para determinar si una imagen es real o sintética dada una clase definida.

Stage2 En la segunda parte de la arquitectura el funcionamiento del generador varía bastante con respecto a la Stage1. El generador toma como entrada una imagen y un parámetro condicional, sin embargo, en un principio procesa de manera independiente la imagen aplicando convoluciones y *pooling* hasta llegar al espacio latente, en donde se concatena el parámetro previamente procesado por una segunda instancia de *conditioning augmentation*.

Este generador asemeja su arquitectura a la usada en pix2pix, sin embargo no hace uso de la estructura U-Net, incluye un parámetro extra y la imagen de entrada no es del mismo tamaño que la imagen de salida.

Por otro lado el discriminador de esta fase funciona de manera muy similar a el de la fase anterior, con la diferencia que la imagen de entrada pasa de ser de 64x64 a 256x256 pixeles.

3.2. Evaluación de una GAN

Evaluar el desempeño de una GAN resulta ser una tarea complicada, ya que las GANs carecen de una función objetiva que permita comparar su rendimiento con otros modelos [25]. Esto deja algunas incógnitas tanto para el entrenamiento como para la evaluación de una GAN, por ejemplo:

- ¿En qué época se obtiene el mejor modelo?
- ¿Cómo se pueden comparar 2 modelos obtenidos? Ya sea con diferente arquitectura o diferentes hiperparámetros.
- ¿Cómo seleccionar las imágenes que muestren el desempeño correcto de una GAN?

Para tratar de responder a estas incógnitas y con la popularización de los modelos basados en GANs en los últimos años, han surgido diferentes métodos tanto cuantitativos como cualitativos para tratar de evaluar de una forma objetiva los modelos. En [26] se presenta un análisis de ventajas y desventajas de 24 métricas cuantitativas y 5 cualitativas, de igual modo en la actualización del artículo [27] se presenta el análisis de otras 19 métricas cuantitativas y 6 cualitativas. Esta gran cantidad de opciones se han planteado para lograr una evaluación objetiva de los modelos generativos, sin embargo, cada una de estas métricas cuenta con ventajas y desventajas, y ninguna resulta ser totalmente confiable ni infalible.

De entre todas las métricas que se han planteado a lo largo de los años, existen 2 métricas cuantitativas que se emplean con más frecuencia, ya que resultan ser las más confiables para la evaluación de los modelos, estas son, Inception Score (IS) [25] y Fréchet Inception Distance (FID) [28]. Ambas métricas emplean un modelo InceptionV3 preentrenado para realizar la evaluación de los datos generados, para posteriormente aplicar operaciones sobre las distribuciones de probabilidad obtenidas.

3.2.1. Evaluación manual

El método cualitativo más empleado e intuitivo para evaluar un modelo generativo es una evaluación hecha por humanos, ya que la mayoría de las GANs tienen como objetivo generar imágenes foto realistas, es decir, las imágenes generadas deben de ser lo suficientemente buenas

para poder engañar a un humano haciéndole creer que las imágenes son reales, lo que se podría ver como una "prueba de Turing"para imágenes. Por lo que podríamos considerar que un modelo es bueno si es capaz de engañar a los humanos, sin embargo, esta tarea no es objetiva, ya que existe un factor de apreciación individual de cada evaluador.

En este método comúnmente la evaluación de las imágenes sintéticas se realiza de la siguiente manera.

- Se eligen evaluadores, los cuales, son capaces de identificar los objetos o clases de interés en la imagen.
- Se presentan imágenes sintéticas y reales a los evaluadores sin hacer de su conocimiento la naturaleza de estas.
- Los evaluadores etiquetan las imágenes, reales y sintéticas de acuerdo a un criterio dado.
- Se analizan las etiquetas asignadas por los evaluadores contra las etiquetas originales de las imágenes.

Este método a pesar de ser ampliamente utilizado, tiene algunas fallas importantes que deben ser tomadas en cuenta [26]:

- La evaluación humana puede resultar cara y puede estar sesgada según el conjunto de evaluadores empleados, además de que es complicado reproducir la evaluación, ya que no es totalmente objetiva.
- Se requiere una gran cantidad de evaluaciones para poder garantizar que es una buena muestra de calificaciones para el modelo.
- Resulta complicado poder evaluar la capacidad del modelo para generar imágenes diversas de una misma categoría.

Para efectos de este trabajo se optó por descartar este método de evaluación, ya que encontrar evaluadores calificados que sean capaces de indicar de manera exacta la cantidad de grados que existen entre dos tomas de una misma escena resulta fuera del alcance del proyecto. Por lo que se optó por realizar evaluaciones automatizadas con las métricas más empleadas en el campo.

3.2.2. Métrica Inception Score (IS)

El Inception Score fue propuesto por Salimans *et al.* [25] y surge como una alternativa a la evaluación manual de los modelos buscando automatizar el proceso, con lo que se reducen los costos y tiempos necesarios para la evaluación, además de estandarizar la evaluación para diferentes modelos, sin embargo, como se mencionó en el mismo artículo, se debe emplear como una guía aproximada para la evaluación de los modelos y no como una función que se tenga que optimizar.

Para calcular el IS se utiliza una red InceptionV3 [29] preentrenada comúnmente en ImageNet, la cual, asigna probabilidades de pertenencia a una clase dada una imagen. Esta métrica evalúa dos rubros:

- La calidad de la imagen, es decir, que se encuentren objetos bien definidos y que no sean borrosos, de modo que la red InceptionV3 tenga gran seguridad al asignar probabilidades, teniendo un valor alto en el objeto deseado y un valor bajo o nulo en los demás.
- La diversidad de imágenes, donde se busca tener variedad de imágenes para todas las posibles clases. Lo cual se logra si la GAN puede generar diversas imágenes para cada uno de los diferentes objetos permitidos.

El IS está definido como:

$$IS(G) = exp(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x})\|p(y)))$$

Donde $\mathbf{x} \sim p_g$ indica que $\mathbf{x}$ son imágenes producidas por el generador, $D_{KL}(p\|q)$ es la divergencia de Kullback-Leibler entre las distribuciones p y q , $p(y|\mathbf{x})$ es la distribución condicional de la clase, y por último $p(y) = \int_{\mathbf{x}} p(y|\mathbf{x})p_g(\mathbf{x})$ es la distribución de probabilidad marginal.

Mientras mayor sea la diferencia entre la distribución de probabilidad de la clase $p(y|\mathbf{x})$ y la distribución de probabilidad marginal $p(y)$ el valor del IS aumenta. El rango de puntuación está dado por la cantidad de clases que se tienen, por ejemplo, para ImageNet que cuenta con 1000 clases, la puntuación va de [0-1000], donde mientras más alta es la puntuación mejor es el desempeño del modelo. En el artículo orinal se plantea que la cantidad de imágenes a evaluar para poder emplear esta métrica debe ser superior a 50k.

Problemas con IS

Si bien esta métrica se emplea con regularidad en la evaluación de los modelos generativos existen algunos casos en los que esta métrica no funciona de manera adecuada como se menciona en [30].

Sensibilidad a los pesos Si se entrenan dos redes Inception bajo el mismo conjunto de datos y se obtiene el mismo rendimiento en ambos casos, no se garantiza que los pesos de la red sean los mismos. Estas variaciones se deben a la aleatoriedad del entrenamiento y pueden repercutir en el cálculo del IS como se muestra en [30]. Por lo que la calificación obtenida en el modelo generador depende del entrenamiento realizado en la red Inception.

Implementación del IS Para que la muestra de imágenes sea significativa se propone hacer uso de 50k imágenes, sin embargo, al momento de realizar la implementación del cálculo para obtener el IS las imágenes se parten en subconjuntos, comúnmente 10, sobre los cuales se calcula la media y la desviación estándar. Para el caso de conjuntos de datos como ImageNet con 1000 clases la cantidad de imágenes en cada subconjunto no es significativa para realizar correctamente el cálculo sobre las probabilidades marginales, además de que variar el número de subconjuntos tiene un impacto directo en la puntuación como se muestra en [30].

Uso en otro conjunto de datos Comúnmente los modelos preentrenados que se encuentran disponibles para InceptionV3 fueron entrenados sobre el conjunto de datos de ImageNet, sin embargo, en el caso de que el modelo generativo se encuentre entrenado en otro conjunto de

datos, surgen varios problemas con la evaluación. El primer problema se presenta para las clases dadas de ImageNet, ya que estas podrían no estar alineadas a las clases del segundo conjunto de datos o no tienen una correspondencia uno a uno, ya que no todos los conjuntos de datos poseen las mismas 1000 clases. Este cambio en las clases puede llevar a una mayor entropía en la probabilidad condicional. Es por ello que para el uso del IS es altamente recomendable entrenar la red InceptionV3 en el mismo conjunto de datos que el modelo generador.

Optimizar el IS Al tener la red InceptionV3 preentrenada, es posible obtener un modelo generador que optimice el IS, sin embargo, realizar este modelo lleva a generar parches adversariales de la red, que no necesariamente son imágenes realistas como se esperaría obtener en un buen modelo generativo.

Figura 3.10: Imágenes con IS= 900.15 de un máximo de 1000. Este modelo se entrenó buscando obtener un IS alto, sin embargo, no se pueden observar elementos realistas en ellas. (Barratt *et al.*, 2018 [30])

Resolución El IS resulta ser susceptible a cambios de resolución en las imágenes, esto se debe a que las imágenes de entrada en la red InceptionV3 son imágenes de 299x299 pixeles, lo cual, requiere hacer un re-escalado de los datos a esta resolución. Al realizar la clasificación de imágenes con menor resolución los objetos son más borrosos, con lo que $p(y|x)$ se dispersa entre todas las clases y el IS tiende a bajar.

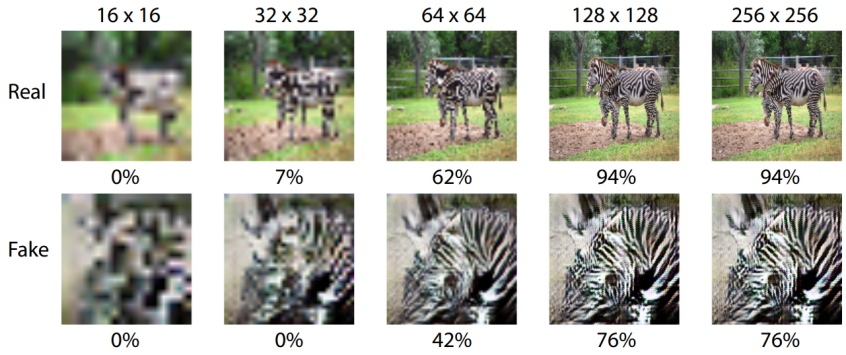


Figura 3.11: Exactitud del IS sobre imágenes reales y generadas por una AC-GAN cambiando únicamente la resolución de las imágenes(Odena *et al.*, 2017 [31])

3.2.3. Métrica Fréchet Inception Distance (FID)

Esta métrica fue propuesta por Heusel *et al.* en 2018 [28]. Surge como una mejora al IS, la FID busca realizar una comparación entre la distribución de las imágenes reales y la distribución de las imágenes sintéticas. Para esta métrica se utiliza nuevamente una red InceptionV3 preentrenada, de la cual se utilizan las salidas antes de la última capa de *pooling*, con lo cual a la salida de esta red se obtiene un vector de 2048 elementos por cada imagen. Esto con la finalidad de obtener las características visuales representativas de las imágenes.

Con esta red se evalúan las imágenes reales y sintéticas, con lo que se obtiene $p(.)$ la distribución de las imágenes generadas y $p_w(.)$ la distribución de las imágenes reales. Posteriormente se obtienen la media y la covarianza (m, C) para $p(.)$ e igualmente (m_w, C_w) para $p_w(.)$. Con estas medias y covarianzas se procede a calcular la "Fréchet distance"ó "Wasserstein-2"definida como:

$$d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + Tr(C + C_w - 2(CC_w)^{1/2}) \tag{3.8}$$

La FID tiene un buen desempeño en términos de discriminación, robustez y eficiencia computacional, ha demostrado ser más robusta ante el ruido que el IS, además, un modelo que genera solo una imagen por clase es capaz de obtener un IS alto, pero obtendría un mal FID. La FID resulta ser más sensible a artefactos añadidos a las imágenes que el IS, como se muestra en la figura 3.12

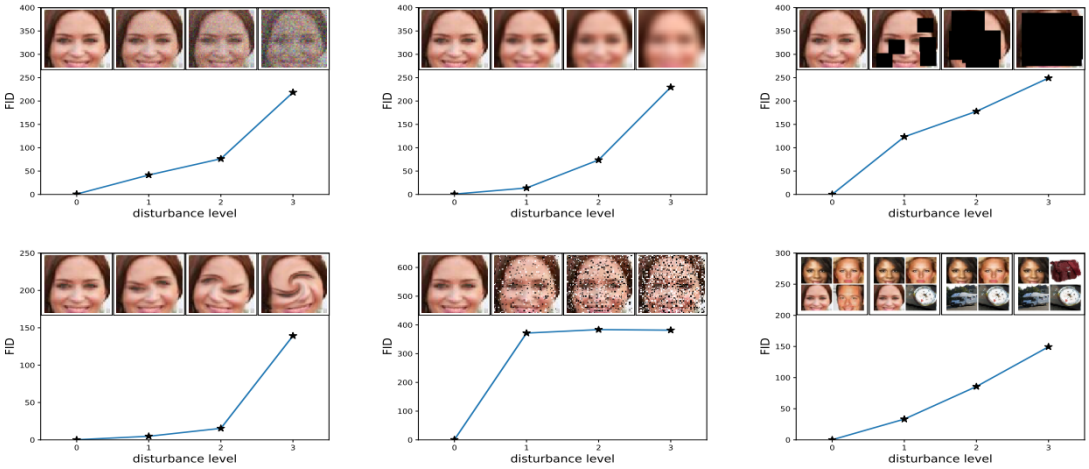


Figura 3.12: Comportamiento de FID con diferentes artefactos en diferentes cantidades. De izquierda a derecha y de arriba a abajo: Ruido gaussiano, desenfoque gaussiano, rectángulos negros, arremolinado, ruido sal y pimienta, inclusión de imágenes de otro conjunto de datos. (Heusel *et al.*, 2018 [28])

CAPÍTULO 4

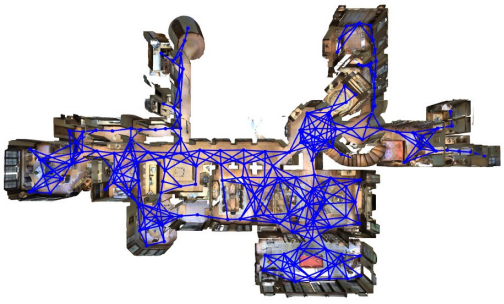
Desarrollo

En este capítulo se hablará de la obtención de las imágenes empleadas en el entrenamiento y en la evaluación de los modelos, además se definirán las diferentes arquitecturas y enfoques empleados para tratar de generar imágenes realistas que resuelvan el problema planteado, así como los detalles de la implementación de las métricas empleadas.

4.1. Conjunto de datos

Para la realización de este proyecto se utilizaron imágenes obtenidas del *dataset* Matterport3D [32], el cual se compone de panoramas 360° tomados en varias locaciones desde diferentes posiciones dentro de estas. Se eligió esta base de datos, ya que las imágenes con las que cuenta son tomas reales y representan un recorrido virtual por las diferentes locaciones, además incorpora una API para poder realizar la exploración y captura de imágenes con los ángulos requeridos en la toma, tanto para el cabeceo vertical como para el horizontal, lo que garantiza que las imágenes obtenidas correspondan al ángulo indicado de manera precisa y así las etiquetas empleadas sean correctas.

(a) Ejemplo de una locación en MatterPort3D, las esferas verdes representan los panoramas de esta.



(b) Diagrama de conexiones en un mapa de MatterPort3D. Las transiciones solo se presentan entre panoramas contiguos.

Figura 4.1: Estructura de mapas en MatterPort3D (Chang *et al.*, 2017 [32])

Cada locación contiene una cantidad diferente de panoramas acorde al tamaño de esta. Cada locación corresponde a una casa o departamento con diferentes habitaciones y estilos de decoración, esto aporta variedad al tipo de habitaciones (sala, baño, cocina, etc.) y ayuda a que el modelo entrenado sea capaz de generalizar de mejor manera.

Como podemos observar en la figura 4.1b las transiciones entre panoramas solo se permiten entre panoramas contiguos, por lo que antes de elegir los panoramas empleados por cada locación, es necesario realizar un recorrido por todos los posibles panoramas. Para lograrlo se implementó un algoritmo BFS [33], ya que al ser una gráfica conexa por locación, se garantiza

visitar todos los panoramas. Una vez visitados todos los panoramas de cada locación se eligieron 10 por cada una y se realizó la captura de imágenes.

Para captura de las imágenes se utilizaron campos de visión de $[-15°, 15°]$ con muestra cada 5°, lo que nos deja 6 tomas por cada referencia, sin embargo como se aplican los mismos ángulos de manera horizontal y vertical se obtienen 48 tomas por cada referencia, como se muestra en la figura 4.2.

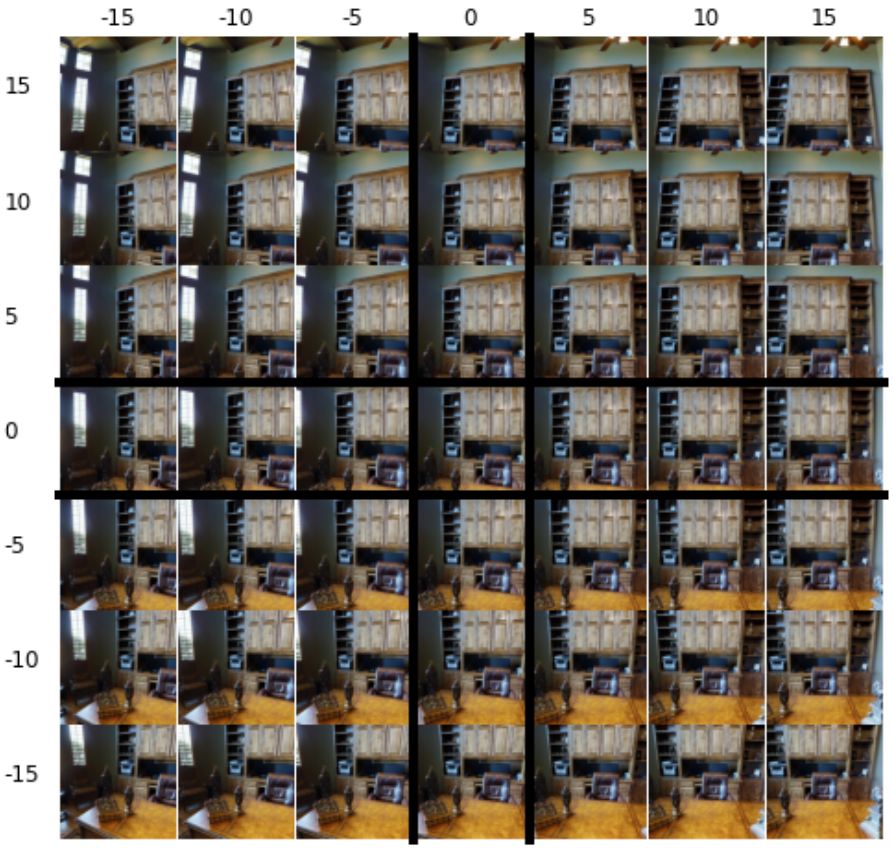


Figura 4.2: Ejemplo de las diferentes tomas obtenidas a partir de una referencia.

Con este procedimiento se obtuvieron dos conjuntos de datos, ambos se componen de 10 panoramas por cada locación. El primer conjunto se utilizó como conjunto de entrenamiento tanto para las GANs como para el modelo empleado en el Inception Score, modelos que se explicarán en las siguientes secciones. Este conjunto está formado de 15 locaciones, lo queda un total de 88200 imágenes, las cuales contienen 1800 ejemplos para cada uno de los ángulos.

El segundo conjunto que se generó es el conjunto de validación, este fue empleado para evaluar el desempeño de la red InceptionV3 utilizada en el IS y posteriormente fue empleado para realizar la evaluación final de las GANs. Cabe destacar que los modelos presentados nunca tuvieron acceso a los datos del conjunto de validación hasta el momento de la evaluación final, evitando así hacer un sesgo en las evaluaciones y poder ver el desempeño de los modelos ante nuevos datos. Este conjunto cuenta con únicamente 4 panoramas, lo que equivale a tener 23520 imágenes en total y 480 ejemplos por cada ángulo.

	Locaciones	Panoramas por locación	Ejemplos por categoría	Espacio en disco	Total de imágenes
Entrenamiento	15	10	1800	15.6GB	88200
Validación	4	10	480	7.96GB	23520

Tabla 4.1: Características de los conjuntos de datos.

4.1.1. Organización

Las imágenes obtenidas están organizadas en la siguiente manera para ambos conjuntos:

- Dataset
  - idLocación
    - idPanorama
      - Imagen

El nombre de la imagen está compuesto por:
anguloReferencia_anguloHorizontal_anguloVertical.png
Donde:

- *anguloReferencia*: se refiere al ángulo respecto al mundo que se toma como referencia.
- *anguloHorizontal*: ángulo que se movió la cámara horizontalmente desde el anguloReferencia.
- *anguloVertical*: desplazamiento vertical a partir del anguloReferencia.

4.2. GAN

En esta sección se muestran las diferentes arquitecturas de GANs empleadas en el desarrollo de este trabajo, las cuales utilizan diferentes aproximaciones al problema para resolverlo de la mejor manera posible.

4.2.1. CGAN U-net

Tomando como base la arquitectura pix2pix [21], se realizaron algunos cambios para poder emplearla como solución a la generación de imágenes en este trabajo. La arquitectura pix2pix como se describe en la sección 3.1.3, es considerada una cGAN con una imagen como condición en lugar de un parámetro simple. La modificación que se consideró para este trabajo, es aumentar el número de parámetros en la entrada de la arquitectura, esto debido a que la arquitectura emplea una imagen como referencia y dos ángulos (horizontal y vertical) como parámetros, a diferencia de pix2pix que solo emplea la imagen referencia.

Generador Utilizando como base la estructura U-Net usada en pix2pix, se toma la imagen de referencia de 256x256 pixeles y se le aplica un bloque con capas convolucionales, ese resultado es almacenado para concatenarse posteriormente. Después se aplica un MaxPooling para conseguir mapas de características con la mitad de tamaño y se repite el proceso hasta llegar a mapas de 8x8 con 128 canales. Independiente a este proceso se pasan los parámetros de los ángulos por capas completamente conectadas para generar 8 matrices de 8x8.

Las matrices generadas a partir de la imagen de referencia y las obtenidas de los ángulos son concatenadas y este es considerado como el espacio latente. A partir de este espacio se aplican bloques capas convolucionales y un UpSampling al doble de tamaño, después de aplicar cada bloque se concatena el resultado guardado anteriormente que corresponde con el tamaño actual de los mapas de características, estas concatenaciones hacen que se utilice la arquitectura U-net. Finalmente se pasan los mapas resultantes por capas convolucionales de las cuales se obtiene una imagen RGB de 256x256 pixeles.

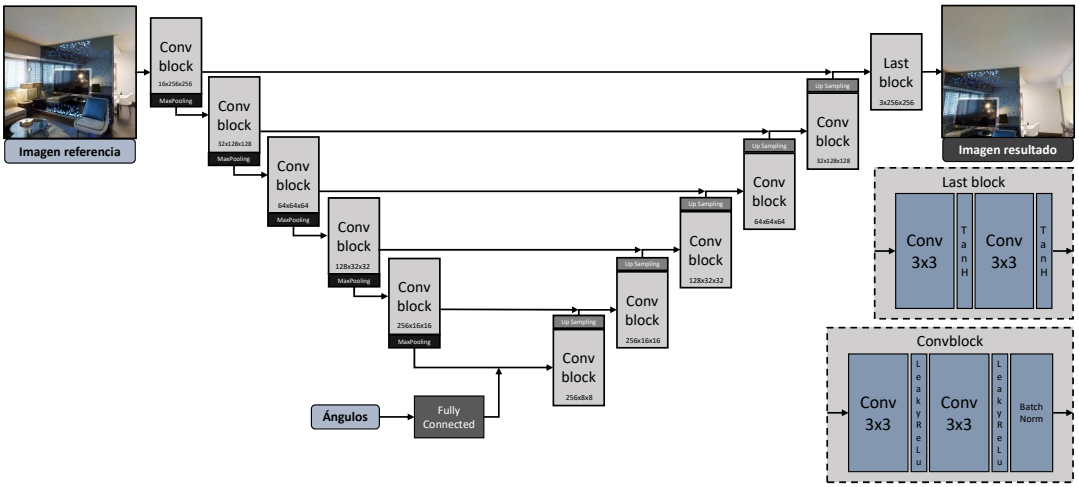


Figura 4.3: Diagrama del generador CGAN modificado haciendo uso de estructura U-Net.

Discriminador Para el discriminador se toman las mismas entradas que en el generador más la imagen a evaluar, la cual puede ser una imagen real del conjunto de datos o una imagen obtenida del generador. Con estos parámetros el discriminador debe de ser capaz de determinar si a partir de la imagen de referencia y los ángulos objetivo, la imagen a evaluar es real o sintética. Para lograrlo se utiliza una arquitectura que aprovecha el principio de una PatchGAN [21].

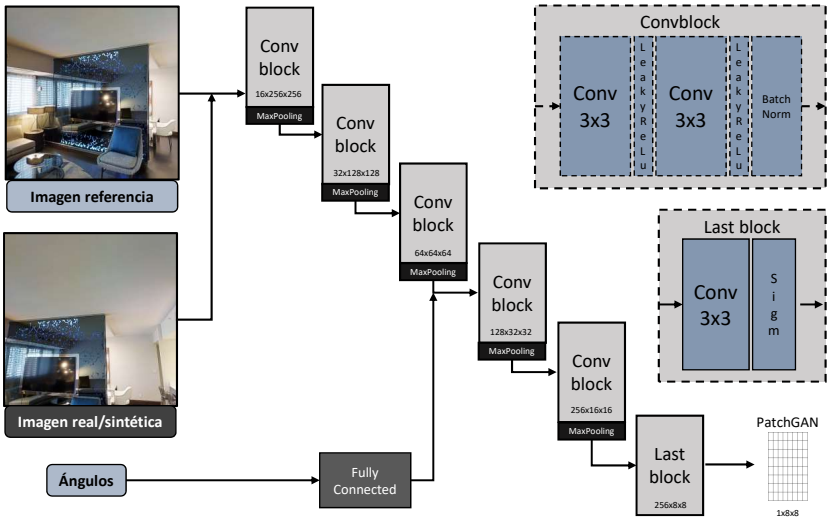


Figura 4.4: Diagrama del discriminador empleado utilizando PatchGAN.

En este tipo de arquitectura la imagen de referencia y la imagen a evaluar se concatenan para pasar por capas convolucionales y MaxPooling. Por otro lado se toman los ángulos objetivo y se procesan en capas completamente conectadas del mismo modo que se realizó en el generador. Después se toman ambos resultados parciales y se concatenan para posteriormente pasar por otras capas convolucionales y MaxPooling donde finalmente se obtiene una matriz de 8x8, la cual tiene valores entre 0 (falsa) y 1 (real). Cada uno de estos valores representa la probabilidad de que una ventana en la imagen evaluada sea real o sintética. Esta evaluación de la imagen por ventanas da la oportunidad de realizar un ajuste más preciso, ya que una parte de la misma imagen puede parecer muy real y otra muy falsa, por lo que el ajuste de los pesos de la red no tiene que realizarse de la misma forma para ambas partes de la imagen.

4.2.2. CGAN sin conexiones U-Net

El uso de conexiones tipo U-Net comúnmente funcionan muy bien para conservar la estructura de la imagen de entrada en la de salida, sin embargo, en el problema planteado en este trabajo esto no es lo que se busca, ya que lo que se plantea hacer es modificar el ángulo de las imágenes y esto implica desplazar las estructuras de la imagen. Es por ello que se realizó otra propuesta modificando la arquitectura anterior para buscar generar imágenes más realistas.

Generador El factor que impide que la estructura general de la imagen de entrada y salida difieran entre sí son las conexiones de tipo U-Net, ya que la información que empleada por estos

saltos pasa por menos capas y por lo tanto, es procesada en menor medida, lo que provoca que las estructuras originales se mantengan en su lugar. Por lo tanto para esta arquitectura se empleó como base la misma arquitectura descrita en la sección anterior, pero eliminando las conexiones tipo U-Net. Con este cambio se espera que las imágenes de salida puedan hacer el desplazamiento de las estructuras en las imágenes sin restricciones, sin embargo, como se menciona en [21] el uso de una pérdida $\mathcal{L}_{L1}$ sin el uso de la arquitectura U-Net puede llevar a resultados difusos o borrosos.

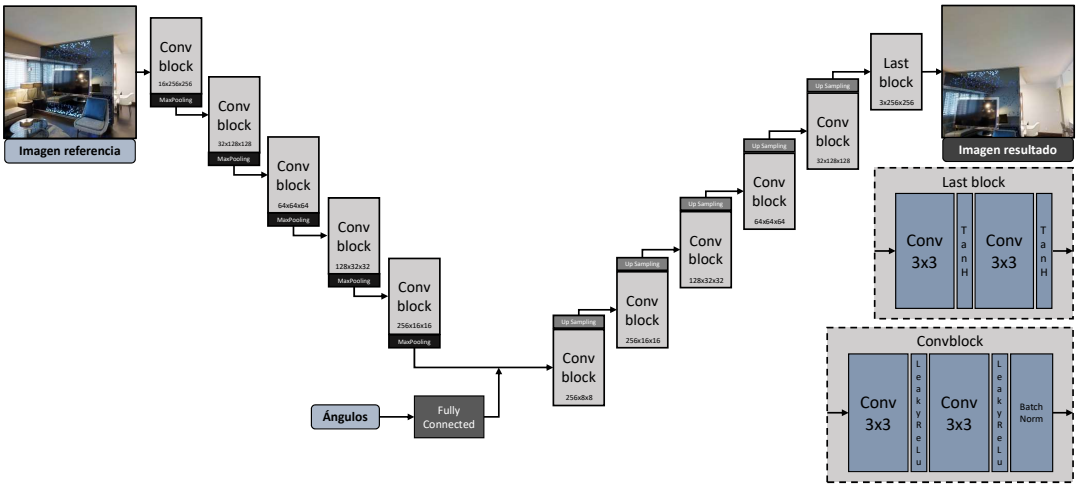


Figura 4.5: Diagrama del generador CGAN modificado sin saltos U-Net.

Discriminador Para esta arquitectura se empleó el mismo diseño del discriminador propuesto en la sección anterior, esto debido a que la modificación propuesta solo afecta al generador. A la salida del generador modificado se obtiene el mismo tipo de información, por lo que discriminarla entre real y falsa se hace de la misma manera.

4.2.3. CGAN híbrida

Generador Buscando mejorar los resultados obtenidos en las dos arquitecturas propuestas anteriormente se realizó una tercera modificación en la arquitectura del generador. Esta modificación propone utilizar saltos únicamente en la mitad inferior de la arquitectura, esto quiere decir que las conexiones solo están presentes en los mapas de características de 64x64 elementos y tamaños inferiores. La idea de esta modificación es la de buscar recuperar información por medio de los saltos U-Net, ayudando a que la imagen no sea borrosa como se espera al no usar ninguna conexión de este tipo. Pero también se espera que al dejar las últimas capas sin las conexiones, no se fuerce a pasar la información de la entrada en la salida, con lo cual las imágenes generadas tengan mayor libertad para realizar los desplazamientos necesarios en las estructuras de las imágenes.

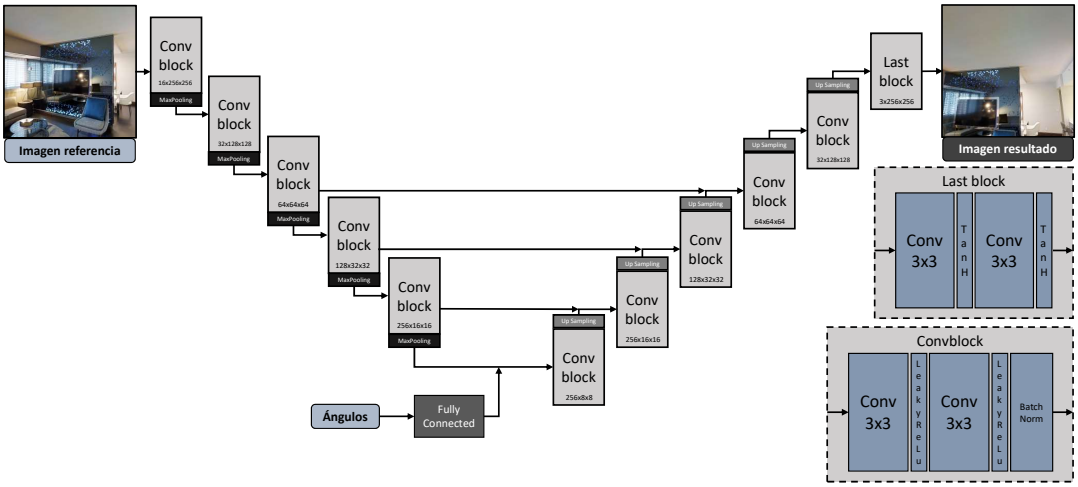


Figura 4.6: Diagrama del generador CGAN híbrido.

Discriminador En esta modificación nuevamente solo se realizaron cambios en el generador, por lo que el discriminador no se modifica y se emplea la arquitectura descrita en la sección 4.2.1.

4.2.4. StackGAN

Los experimentos realizados con las arquitecturas descritas anteriormente generan imágenes de 256x256 pixeles haciendo uso de una sola GAN. Este método no siempre resulta ser el más efectivo para generar imágenes de un tamaño considerable como es el caso en este problema, es por ello que se utilizó otra propuesta de arquitectura, la cual emplea un enfoque diferente para resolver el problema.

Para esta propuesta de arquitectura se tomó como base la propuesta de la StackGAN descrita en la sección 3.1.4, la cual propone emplear 2 GANs apiladas (*Stage1* y *Stage2*) en lugar de una GAN única para resolver una tarea. Esta tarea originalmente consiste en realizar traducción texto a imagen, sin embargo, con unas pequeñas modificaciones es posible pasar a resolver la traducción imagen a imagen, con las particularidades de la tarea que se busca resolver en este trabajo.

Como se plantea en la arquitectura original en este caso se emplearán dos GANs para generar imágenes de 256x256 pixeles, esto con la finalidad de distribuir la complejidad de generar imágenes de ese tamaño en dos tareas más simples. Primero generar imágenes de 64x64 pixeles las cuales no buscan tener detalle, pero sí las características más generales (*Stage1*) y posteriormente añadir los detalles finos para pasar de imágenes de 64x64 a imágenes de 256x256 pixeles (*Stage2*).

Stage1

Es esta fase se generan imágenes de 64x64 pixeles, en las cuales se busca plasmar las estructuras más importantes de la imagen original desplazadas por el ángulo deseado, en esta fase no se espera obtener una gran calidad en el detalle, ya que las imágenes de 64x64 no son capaces de capturar los detalles más finos de una imagen.

Generador El Generador obtiene las mismas dos entradas que se usaron para las arquitecturas basadas en CGAN:

- *Ángulos objetivo*
- *Imagen de referencia*

Debido al cambio de tarea a resolver, la arquitectura empleada por este generador es bastante diferente de la empleada en la Stage1 de la StackGAN original. En esta se emplea un *embedding* de entrada para la generación de imágenes, pero para este trabajo se utiliza una imagen de referencia y dos ángulos. Por la naturaleza de las entradas el generador puede verse como una modificación de la arquitectura pix2pix para traducción imagen a imagen.

Esta arquitectura funciona de la misma manera que el generador propuesto en la sección 4.2.2, pero disminuyendo la cantidad de bloques que se emplean, ya que tanto las imágenes de entrada como de salida son de 64x64 pixeles.

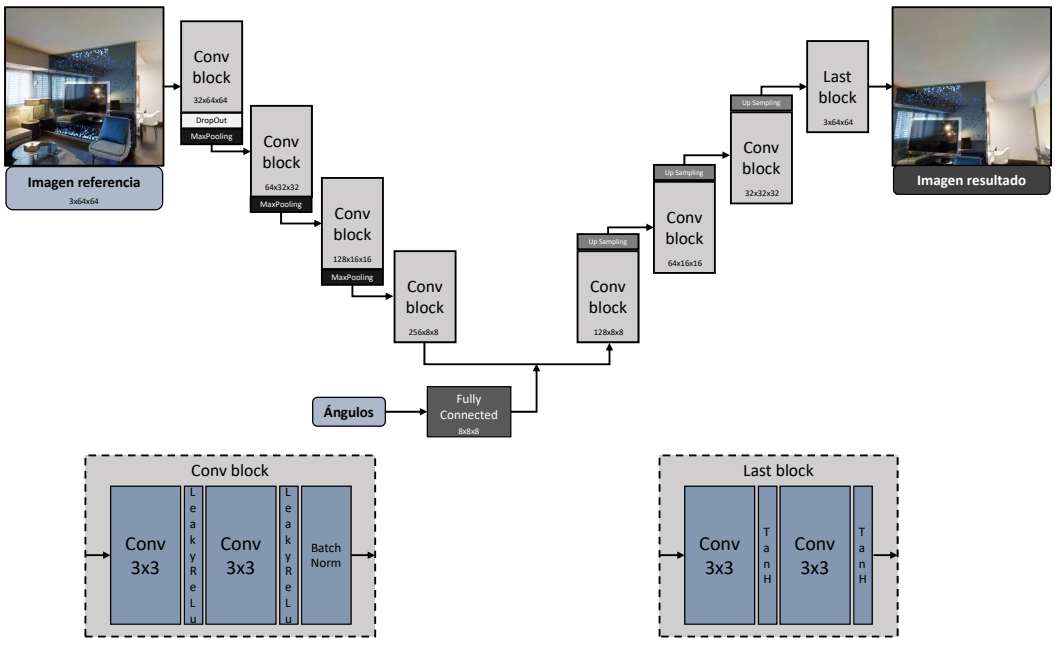


Figura 4.7: Diagrama del generador de la Stage1.

Discriminador De la misma manera que el discriminador descrito en la sección 4.2.1, este discriminador toma como entrada 3 parámetros, utilizando el mismo principio, pero con la salvedad que en este caso las imágenes son de 64x64 pixeles. El usar imágenes más pequeñas

provoca que la cantidad de bloques con capas convolucionales y MaxPooling se reduzca como se aprecia en la figura 4.8.

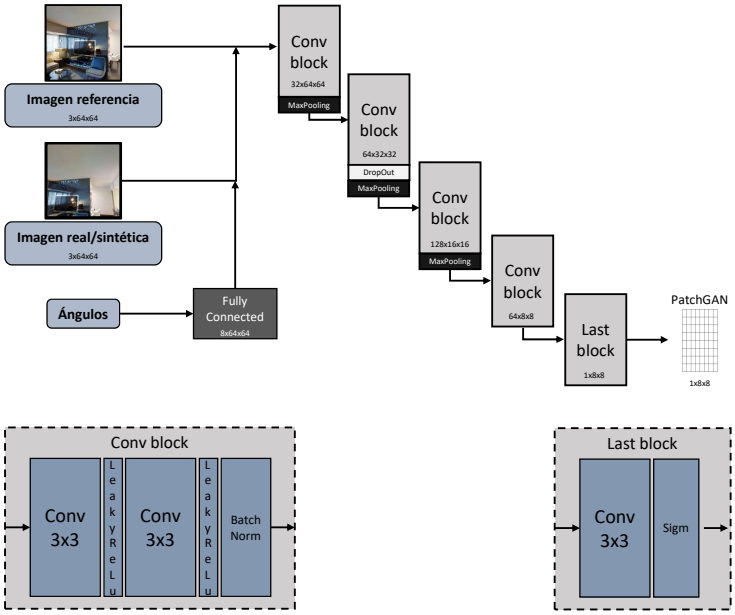


Figura 4.8: Diagrama del discriminador de la Stage1.

Stage2

En esta fase se busca agregar detalles a las imágenes que se obtienen de la Stage1. En este caso la modificación de la arquitectura de la StackGAN original es mínima, ya que originalmente también se busca llevar a cabo una traducción imagen a imagen de una imagen con poco detalle con resolución de 64x64 pixeles a otra de 256x256 pixeles que incluyera detalles más finos.

Generador El generador busca tomar como entrada la imagen producida por la Stage1 y producir imágenes de 256x256 con detalles más finos. Para ello se asume que la imagen de la Stage1 ya tiene aplicada la modificación de ángulos y solo carece de detalles finos. Si esta característica está presente en las imágenes de entrada, es posible utilizar una arquitectura en el generador de la Stage2 que haga uso de las conexiones tipo U-Net, como el propuesto en la sección 4.2.1, esto con la finalidad de mejorar la calidad de las imágenes a la salida. En este caso el generador utiliza tres entradas:

- Imagen S1: imagen generada en la Stage1 (64x64 pixeles), la cual se espera que ya cuente con la transformación de ángulo.
- Ángulos: Se pasan los mismos ángulos que se introdujeron a la Stage1.
- Imagen referencia: Se pasa la imagen de referencia con resolución de 256x256 pixeles, con esto se busca obtener detalles finos de la imagen que puedan ser plasmados en la imagen de salida y que fueron omitidos en la imagen S1.

Como podemos apreciar en la figura 4.9 se toma la imagen de referencia y se pasa por capas convolucionales y MaxPooling hasta llegar a un tamaño de 64x64. Posteriormente se toma ese resultado y se concatena con la imagen S1, donde posteriormente se pasan las imágenes por más capas convolucionales y MaxPooling hasta llegar a elementos de 16x16. Por otro lado de manera independiente se pasan los ángulos por capas completamente conectadas y el resultado se concatena a los elementos de 16x16 obtenidos previamente. Por último se aplican capas convolucionales y UpSampling para reconstruir las imágenes hasta llegar a imágenes de 256x256 pixeles. Las conexiones tipo U-Net están presentes solo para los mapas de características con elementos de tamaño menor o igual a 64x64, ya que es a partir de este punto donde la información de la imagen S1 está disponible.

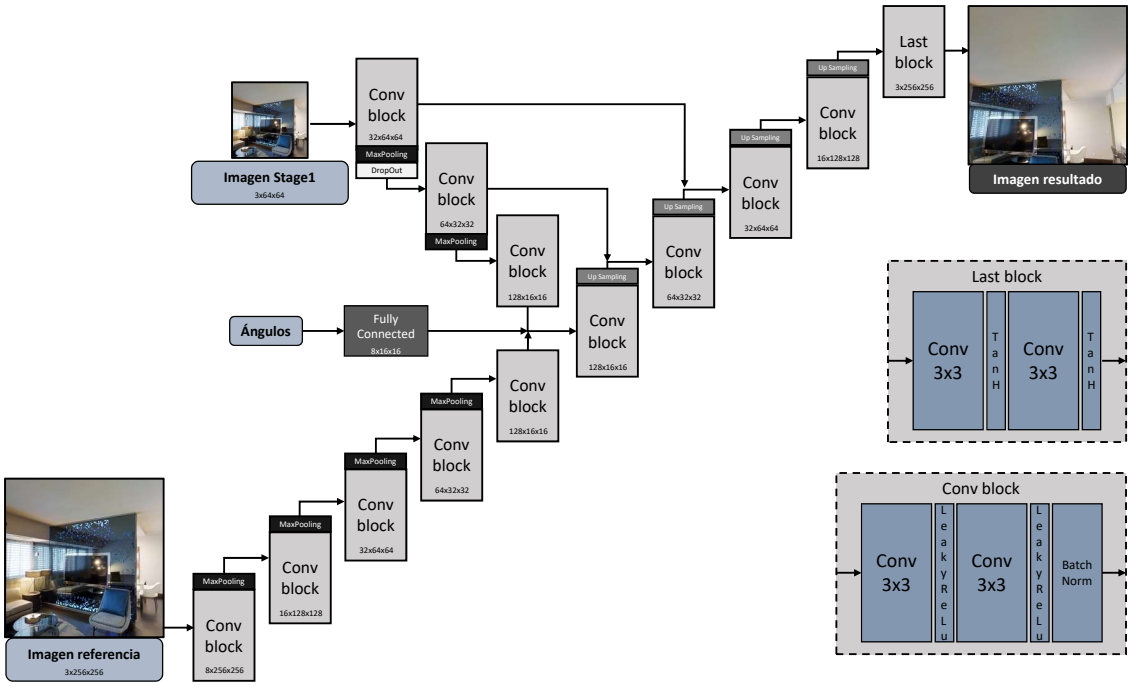


Figura 4.9: Diagrama del generador de la Stage2.

Discriminador Nuevamente el discriminador mantiene la misma estructura trabajada hasta este momento, en este caso a diferencia de la Stage1 las imágenes vuelven a ser de 256x256 pixeles, por lo que la estructura empleada es la misma que se plantea en la sección 4.2.1. Esto se realiza, ya que la tarea del discriminador que es determinar si una imagen es real o sintética dada una referencia y un ángulo se mantiene sin cambios.

4.2.5. FillGAN

Esta arquitectura realiza un cambio significativo en la forma de abordar el problema de la modificación del ángulo en las imágenes. Hasta el momento las arquitecturas propuestas anteriormente, toman como entrada una imagen de referencia y los ángulos como parámetro, y con el uso exclusivo de GANs se obtiene la imagen modificada. Para este nuevo enfoque se

propone apoyar a la GAN haciendo uso de un preprocesamiento de la imagen de referencia, con la finalidad de disminuir la complejidad de la tarea que la GAN tiene que resolver.

Existe un desplazamiento en ángulos entre la imagen de referencia y la imagen que se busca generar. Este desplazamiento se puede simplificar si se ignora el cambio de perspectiva que existe en los objetos. Si se realiza esta simplificación se puede expresar el desplazamiento de las imágenes en cantidad píxeles.

$$\Delta = \frac{size \times ang}{FOV} \tag{4.1}$$

Donde $size$ es el tamaño de la imagen en pixeles, ang es el ángulo de modificación deseado, FOV es el campo de visión de la toma y Δ es el desplazamiento en pixeles. El cálculo de este desplazamiento debe ser calculado tanto horizontal como verticalmente.

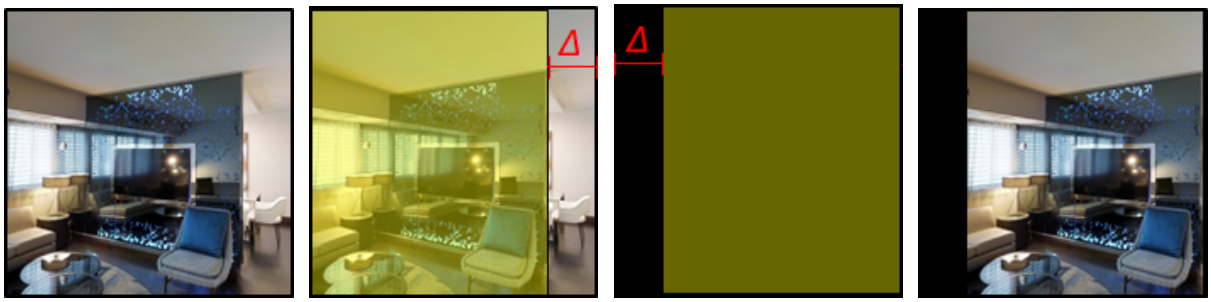
Una vez calculado el desplazamiento, este se puede aplicar en la imagen. Como no se posee más información que la contenida en la imagen de referencia, al desplazar los pixeles se genera un espacio sin información del mismo tamaño del desplazamiento como se observa en la figura 4.10d. De igual manera existe una cantidad de información de la imagen real que se tiene que descartar. El proceso para realizar este desplazamiento es el siguiente:

- Este proceso se aplica tanto de manera vertical como horizontal.
- Se calcula Δ de acuerdo a la ecuación 4.1.
- Se calculan los límites del área útil en la imagen de referencia y se copia la información.

$$A_{ref} = \text{máx}(0, -\Delta), \text{mín}(size - \Delta, size) \tag{4.2}$$

- Se crea una imagen nueva llena de 0s.
- Se calcula el área donde se situará la información copiada anteriormente y se pega.

$$A_{dest} = \text{máx}(0, \Delta), \text{mín}(size + \Delta, size) \tag{4.3}$$

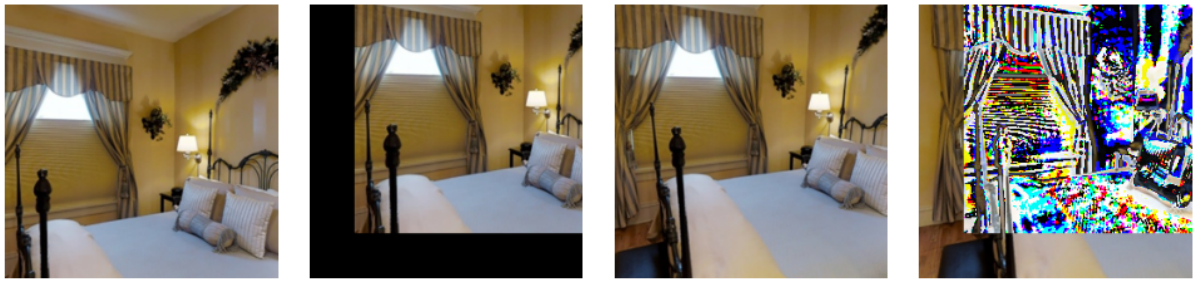


(a) Imagen de referencia. (b) Cálculo del área útil de referencia. (c) Cálculo de área destino de información. (d) Imagen desplazada.

Figura 4.10: Preprocesamiento de imágenes de referencia considerando únicamente un desplazamiento horizontal.

Para el caso del conjunto de datos empleado, se conoce tanto el tamaño $size = (256, 256)$ y el campo de visión ($FOV = 60°$) de las imágenes, por lo que empleando las fórmulas anteriores

se puede calcular la cantidad de pixeles que una imagen se debe desplazar de acuerdo a los ángulos deseados. Sin embargo, como se mencionó anteriormente, la transformación propuesta no corresponde adecuadamente con un cambio real de perspectiva, esto se puede apreciar en la figura 4.11d. En esta imagen se realiza una resta de los valores de los pixeles entre la imagen objetivo (figura 4.11c) y la imagen con la transformación propuesta (figura 4.11b). Como podemos observar las estructuras principales de la imagen se encuentran muy cercanas a las de la imagen objetivo, sin embargo, las diferencias existentes son ocasionadas por la diferencia de perspectiva entre las imágenes. Con esto podemos corroborar que por sí sola esta transformación no es suficiente, pero resulta una aproximación bastante buena para realizar el desplazamiento.



(a) Imagen referencia.

(b) Imagen desplazada $-10°$ vertical y $-10°$ horizontal.

(c) Imagen objetivo.

(d) Diferencia entre imagen desplazada e imagen objetivo.

Figura 4.11: Comparación entre las imágenes deseadas y el resultado del desplazamiento planteado.

Si se emplea la imagen de referencia desplazada como imagen de entrada en una GAN, la tarea a resolver por la GAN cambia. Ya que la GAN no tiene que aprender a desplazar completamente la imagen, ahora solo tiene que aprender a realizar una pequeña corrección de perspectiva. Por otro lado tiene una tarea nueva a resolver, la cual es conocida como rellenado o *inpainting*, en esta tarea la GAN trata de rellenar la información faltante (área del desplazamiento), y tiene que rellenarla de manera que no se pierda coherencia con la información que sí se tiene de la imagen. Este cambio de tarea busca que la GAN se concentre únicamente en corregir y rellenar la imagen dejando a un lado el desplazamiento y mejorando la calidad de las imágenes.

Para este nuevo acercamiento se experimentó con dos variaciones en la arquitectura, las cuales se describen a continuación.

Variante con ángulos

En esta primera variación de la arquitectura se utiliza como parámetros los ángulos deseados, estos no solo se emplean para realizar el desplazamiento inicial de la imagen de referencia, sino que además de esto se introducen tanto en el generador como en el discriminador.

Generador Se introducen los ángulos a unas capas completamente conectadas para formar 8 mapas de características de 8x8, los cuales se concatenan en el espacio latente del generador, justo antes de comenzar con la reconstrucción de la imagen. La idea de introducir los ángulos

como parámetros es la de poder caracterizar los cambios que requiere cada ángulo, por ejemplo, la perspectiva de una escena cambia de manera diferente si es que se desplaza a la izquierda o a la derecha. También se espera la misma zona sin información (área negra) para el mismo ángulo de modificación por lo que al emplear los ángulos como parámetros podría ayudar al modelo a caracterizar la información faltante según el ángulo.

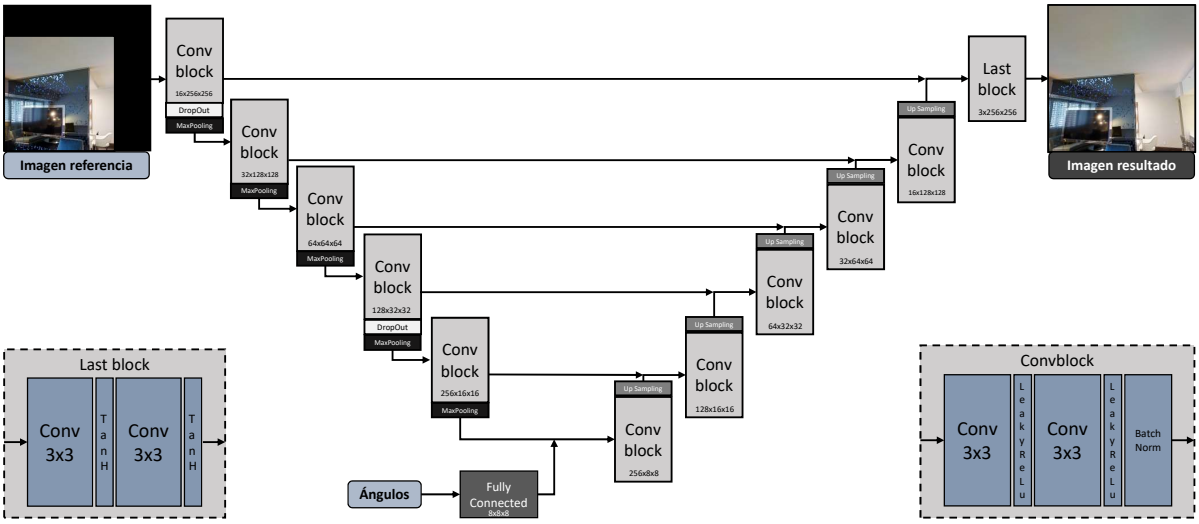


Figura 4.12: Diagrama del generador fillGAN con ángulos.

Discriminador

El discriminador empleado por esta arquitectura es igual al usado en la arquitectura basada en cGAN (sección 4.2.1). Como se comentó anteriormente la tarea de esta GAN cambia respecto a la tarea que resuelve la arquitectura basada en cGAN, sin embargo, el discriminador no cambia la tarea, ya que para ambas arquitecturas el discriminador está encargado de diferenciar una imagen real de una sintética, dado un ángulo y una referencia, sin importar cómo es que se generó la imagen sintética.

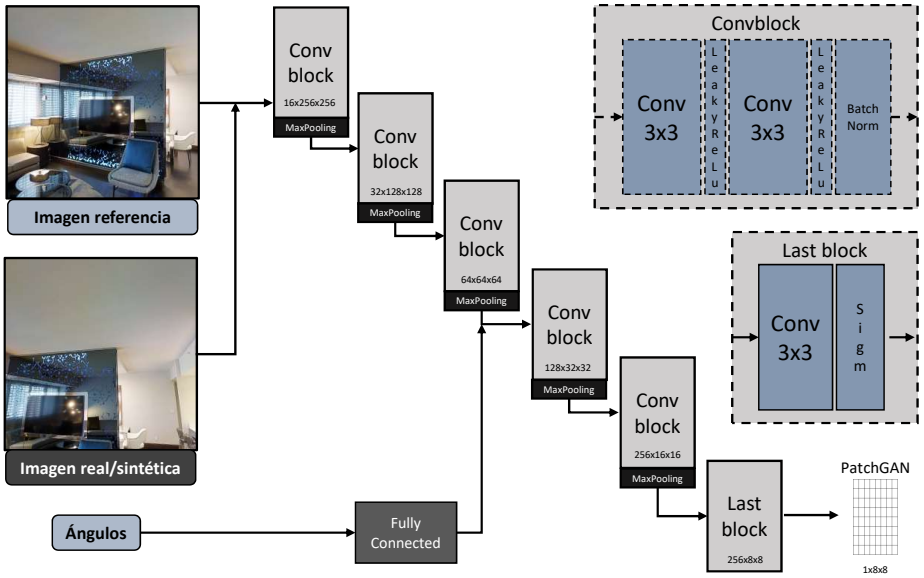


Figura 4.13: Diagrama del discriminador fillGAN con ángulos.

Variante sin ángulos

La segunda variante de la arquitectura se centra únicamente en rellenar la imagen que se introduce al modelo, es decir, en esta arquitectura se emplean los ángulos para realizar el desplazamiento de la imagen de referencia y posteriormente la GAN se encarga de rellenar los espacios faltantes sin tomar en cuenta cuál fue el ángulo de desplazamiento. La idea de no emplear los ángulos como parámetros en la GAN es la de no dividir los ejemplos de entrenamiento en las 49 clases posibles, por el contrario se busca que la red aprenda a rellenar la imagen independientemente del ángulo.

Si comparamos la figura 4.12 y 4.14 con la figura 4.13 y 4.15 podemos observar que la única diferencia entre ambas arquitecturas está en la falta de ángulos y las capas completamente conectadas por las que pasan estos.

Generador

Como ya se mencionó anteriormente en este caso el generador toma únicamente una imagen de referencia desplazada previamente y se espera que produzca una imagen que corrija la perspectiva y rellene el espacio faltante.

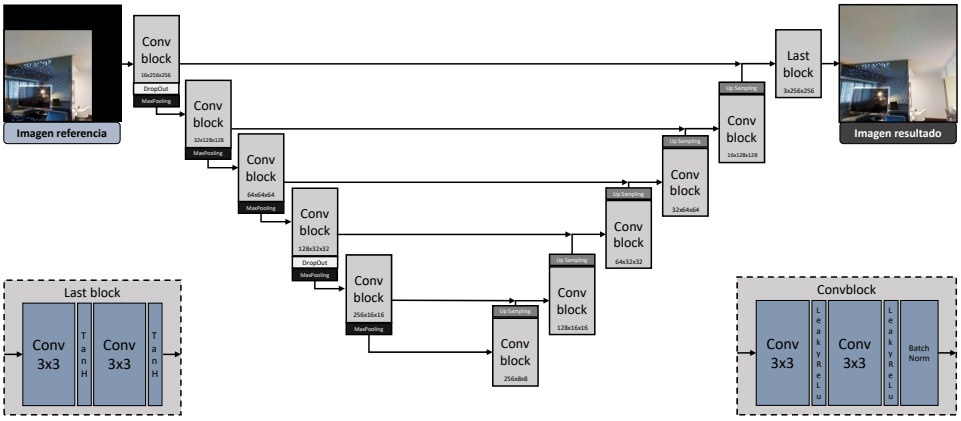


Figura 4.14: Diagrama del generador fillGAN sin ángulos.

Discriminador

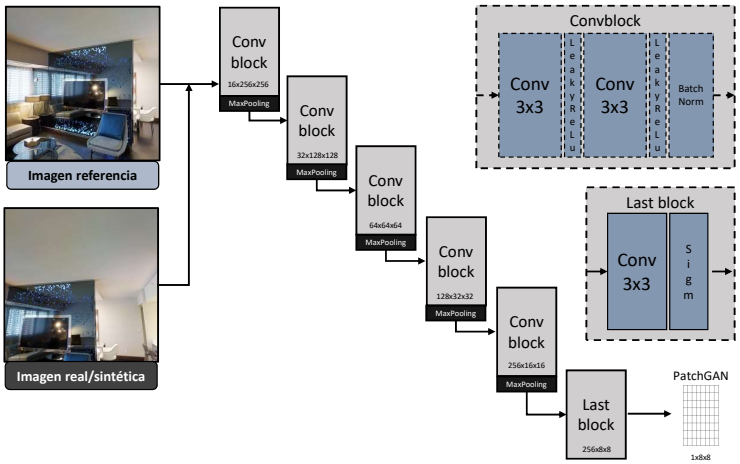


Figura 4.15: Diagrama del discriminador fillGAN sin ángulos.

El discriminador empleado en esta variante de la arquitectura utiliza la misma estructura que se describe en la sección anterior, pero quitando las capas completamente conectadas donde se procesan los ángulos de entrada.

4.3. Evaluación

Para la evaluación de los modelos generados se emplearon dos de las métricas más usadas para evaluar las GANs, Inception Score y Fréchet Inception Distance.

4.3.1. Métrica Inception Score

La métrica Inception Score es una de las más empleadas para la evaluación de GANs, sus principales tareas se basan en identificar si existen objetos bien definidos y variedad de imágenes por clase. Como se menciona en la sección 3.2.2 es altamente recomendable entrenar la red InceptionV3 en el mismo *dataset* en que se trabaja el modelo generador, es por ello que para la implementación realizada de IS en este trabajo se reentrenó la red InceptionV3 sobre el conjunto de datos de entrenamiento mencionado en la sección 4.1.

Las clases que se utilizaron en la capa de clasificación de la red corresponden a los diferentes 49 ángulos que se tienen en el conjunto de datos. Esta modificación es realizada, ya que las clases de interés para este problema no recaen en el tipo de objeto que aparece en la imagen sino en el ángulo entre una imagen y otra, lo que nos lleva a la segunda modificación realizada en la arquitectura de la red.

Una imagen por si sola, ya sea real o sintética, no aporta suficiente información para poder determinar su ángulo de inclinación, ya que no se especifica una referencia a partir de la cual se calcula el ángulo, es por ello que es necesario introducir dos imágenes, una como referencia y otra con un ángulo a partir de la primera, con esto se busca que la red sea capaz de asignar la clase de acuerdo a la variación de ángulo entre ambas imágenes.

Para hacer el re entrenamiento junto con los cambios mencionados anteriormente se realizo lo siguiente:

- Se descargó el modelo preentrenado disponible en PyTorch [34].
- Se cambió la última capa (capa clasificadora) por una capa completamente conectada con 49 salidas.
- Se substituyó la capa de entrada por una capa convolucional con 6 canales de entrada en lugar de los 3 definidos en el modelo original.
- Se realizó el entrenamiento de la red utilizando el conjunto de datos descrito en la sección 4.1.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

(a) Tabla con la arquitectura InceptionV3. (Szegedy *et al.*, 2015 [29])

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×6
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 49

(b) Tabla con la arquitectura modificada para el IS. Los recuadros rojos indican las modificaciones a la arquitectura. (Szegedy *et al.*, 2015 [29])

Figura 4.16: Modificación realizada para IS.

4.3.2. Métrica Fréchet Inception Distance

Esta métrica busca evaluar la distancia entre dos distribuciones de probabilidad, para ello se emplea el modelo preentrenado disponible en PyTorch [34] , del cual se descartó la última capa de *pooling* y la capa de clasificación, con lo que al introducir una imagen en la red se obtiene un vector de 2048 elementos que representa características visuales importantes para la clasificación.

Para obtener la FID de cada modelo contra el conjunto de datos de validación, primero se utilizó cada modelo para generar las mismas imágenes que se encuentran en el conjunto de validación. Cabe mencionar que ninguno de los modelos GAN entrenados vieron estos datos hasta el momento de la evaluación, esto con la finalidad de poder evaluar el desempeño de los modelos ante datos nuevos, además de que ayuda a descartar modelos que únicamente memorizan las imágenes de los datos de entrenamiento y son incapaces de generar variedad de estos.

Para el cálculo de la FID en cada modelo se toman las 23520 imágenes reales y las correspondientes imágenes sintéticas generadas por el modelo a evaluar. Posteriormente se pasan por la red InceptionV3 y se almacenan todos los vectores de salida. Una vez que se tienen los vectores de salida de la distribución real y de la sintética se calcula la media y la desviación estándar, con las cuales posteriormente se realiza el cálculo de la FID de acuerdo a la ecuación 3.2.3.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figura 4.17: Modificación realizada a la red InceptionV3 para obtener la FID. Las capas en rojo fueron eliminadas para obtener los vectores de 2048 elementos. (Szegedy *et al.*, 2015 [29])

CAPÍTULO 5

Experimentos y resultados

Este capítulo contiene el reporte de los resultados de los diferentes modelos entrenados, incluyendo el modelo InceptionV3 empleado en el IS y los modelos GAN basados en las diferentes arquitecturas descritas en el capítulo anterior. Todos los modelos GAN presentados a continuación fueron entrenados con el conjunto de entrenamiento y evaluados con Inception Score y Fréchet Inception Distance sobre el conjunto de validación (conjuntos descritos en la sección 4.1).

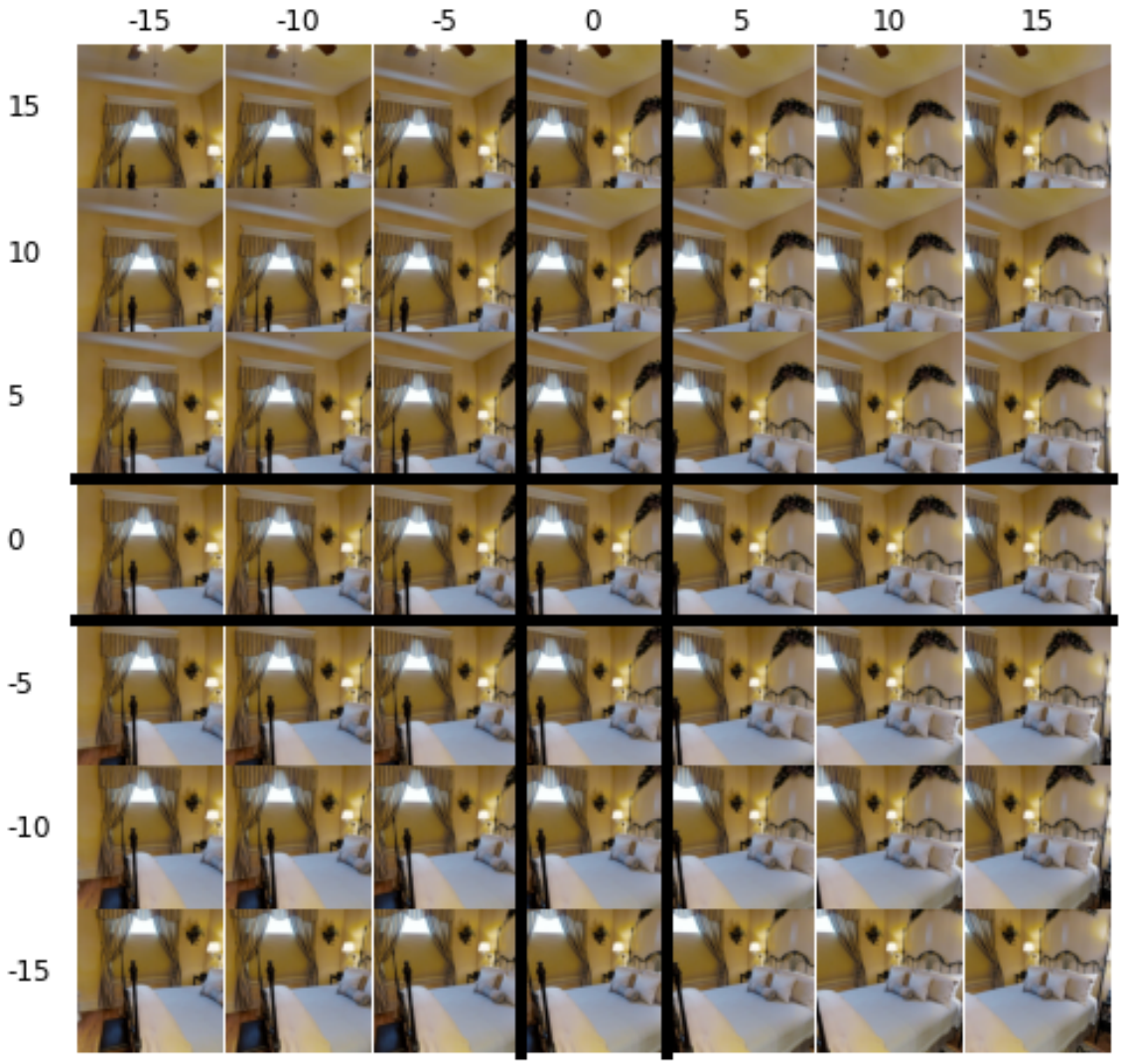
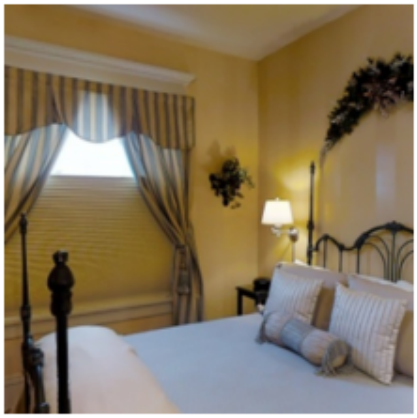


Figura 5.1: Imágenes objetivo a diferentes ángulos.

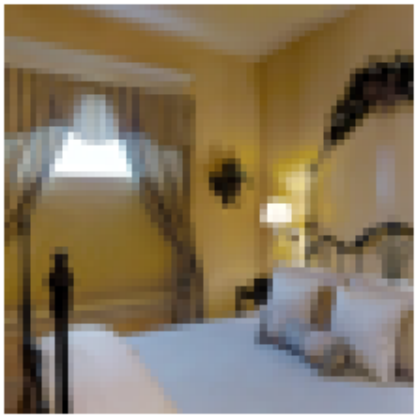
En la figura 5.1 se presenta una composición de las imágenes objetivo a diferentes ángulos, estas son obtenidas del conjunto de datos en validación. Como se puede observar, en las etiquetas de cada fila y columna se indica el ángulo de modificación, horizontal y vertical respectivamente.

En las siguientes secciones se presentarán diferentes composiciones de imágenes con las mismas variaciones en ángulos pero compuestas de las imágenes generadas por cada uno de los diferentes modelos entrenados.

Las imágenes presentadas para cada modelo se generaron a partir de la misma imagen de referencia con una resolución de 256x256 pixeles, a excepción de las generadas por la Stage1 de la StackGAN la cual emplea la misma imagen pero a resolución de 64x64, debido a la naturaleza de la arquitectura. Se utilizó la misma imagen de referencia para poder hacer un comparativo visual entre los resultados de los diferentes modelos.



(a) Imagen de referencia resolución 256x256.



(b) Imagen de referencia resolución 64x64.

Figura 5.2: Comparación de una imagen de referencia a diferente resolución.

5.1. Métricas

Como se ha descrito anteriormente para evaluar las imágenes se utilizará el IS y la FID. En el caso del IS primero es necesario realizar el reentrenamiento de la red InceptionV3, el cual se explica en la siguiente subseccción. En el caso de la FID, no es necesario realizar un reentrenamiento, ya que únicamente requiere hacer uso del modelo preentrenado eliminando las últimas capas.

5.1.1. Red neuronal InceptionV3

Esta red se inicializa con los pesos preentrenados disponibles en PyTorch y se realizan las modificaciones descritas en la sección 4.3.1, para posteriormente reentrenar la red durante 80 épocas sobre el conjunto entrenamiento descrito en la sección 4.1, el cual para este proceso se dividió en dos subconjuntos equilibrados con proporciones 80 % entrenamiento y 20 % prueba, con este nuevo subconjunto de entrenamiento se entrenó la red y con el conjunto de prueba se evaluó el rendimiento del modelo cada época para elegir el mejor modelo entrenado.

Posteriormente se evaluó el mejor modelo obtenido (época 53), con el conjunto de validación, descrito en la sección 4.1, en el cual se reportó el siguiente desempeño:

	precision	recall	f1-score	support
0.0	1.000	1.000	1.000	480
1.0	1.000	1.000	1.000	480
2.0	1.000	1.000	1.000	480
3.0	1.000	1.000	1.000	480
4.0	1.000	1.000	1.000	480
5.0	1.000	1.000	1.000	480
6.0	1.000	1.000	1.000	480
7.0	1.000	1.000	1.000	480
8.0	1.000	1.000	1.000	480
9.0	1.000	1.000	1.000	480
10.0	1.000	1.000	1.000	480
11.0	1.000	1.000	1.000	480
12.0	1.000	1.000	1.000	480
13.0	0.996	1.000	0.998	480
14.0	1.000	1.000	1.000	480
15.0	1.000	1.000	1.000	480
16.0	1.000	1.000	1.000	480
17.0	1.000	1.000	1.000	480
18.0	1.000	1.000	1.000	480
19.0	1.000	1.000	1.000	480
20.0	0.996	1.000	0.998	480
21.0	1.000	0.998	0.999	480
22.0	1.000	1.000	1.000	480
23.0	1.000	1.000	1.000	480
24.0	1.000	1.000	1.000	480
25.0	1.000	1.000	1.000	480
26.0	1.000	1.000	1.000	480
27.0	0.998	0.998	0.998	480
28.0	1.000	1.000	1.000	480
29.0	1.000	1.000	1.000	480
30.0	1.000	1.000	1.000	480
31.0	1.000	1.000	1.000	480
32.0	1.000	1.000	1.000	480
33.0	0.998	1.000	0.999	480
34.0	1.000	0.998	0.999	480
35.0	1.000	1.000	1.000	480
36.0	1.000	1.000	1.000	480
37.0	1.000	1.000	1.000	480
38.0	1.000	1.000	1.000	480
39.0	1.000	1.000	1.000	480
40.0	1.000	1.000	1.000	480
41.0	1.000	0.996	0.998	480
42.0	0.998	1.000	0.999	480
43.0	1.000	1.000	1.000	480
44.0	1.000	1.000	1.000	480
45.0	1.000	1.000	1.000	480
46.0	1.000	1.000	1.000	480
47.0	1.000	0.998	0.999	480
48.0	1.000	0.998	0.999	480
accuracy			1.000	23520
macro avg	1.000	1.000	1.000	23520
weighted avg	1.000	1.000	1.000	23520

Figura 5.3: Calificación del mejor modelo IS reentrenado, evaluado sobre el dataset de validación.

Como se puede apreciar en la tabla 5.3, al evaluar sobre el conjunto de validación, el modelo reentrenado tiene un desempeño muy bueno, ya que para las 49 clases tiene precisión, recall y f1-score con valores de 1 o muy cercano a 1. Al observar estos resultados podemos apreciar que la clasificación que se realiza es muy buena, por lo que el valor que se obtenga al calcular los IS de los modelos generativos no estarán sesgados por un mal modelo, por el contrario se esperan que sean muy acertados.

5.1.2. Valores de referencia

Para las dos métricas conocemos el rango de valores posibles a obtener, para el IS es $[0-49]$ y para la FID es $[0-\text{ínf})$, sin embargo, es necesario tomar una referencia sobre estas dos métricas para conocer la calificación que se puede obtener con los datos reales que tenemos y a partir de ese punto poder hacer la comparación contra los datos sintéticos. Para obtener ese punto de partida, se realizó el cálculo de ambas métricas sobre los conjuntos de imágenes reales, en donde se obtuvieron los resultados mostrados en la tabla 5.1.

	IS	FID vs Entrenamiento
Entrenamiento	48.9427	0.0
Validación	48.8950	45.7902

Tabla 5.1: Tabla de métricas para datos reales.

De la tabla 5.1 podemos observar que el IS en ambos casos se encuentra muy cerca de 49, el cual, es el valor máximo, ya que esta es la cantidad de clases para los ángulos posibles. Recordando que mientras la calificación sea mayor, mejor es el conjunto de imágenes. En un escenario ideal, al ser ambos conjuntos imágenes reales la calificación sería de 49 cerrado, sin embargo como se mencionó anteriormente el modelo InceptionV3 que se emplea no es perfecto, pero es bastante acertado.

Para la FID el caso ideal es cuando la distancia es 0, esta distancia normalmente solo es posible obtenerla al comparar una distribución con ella misma, como es el caso del primer resultado de la tabla 5.1 donde se compara la distribución del conjunto de entrenamiento con ella misma. Este valor fue obtenido con la finalidad de verificar la distancia esperada entre una misma distribución.

Los datos de entrenamiento y validación tienen una distancia de 45.7902, ya que a pesar de ser datos reales con características similares en ambas distribuciones, no son exactamente las mismas imágenes. Este valor sirve como referencia para poder discernir a qué valores se aspira llegar en las calificaciones de los modelos generativos, ya que llegar a un valor de 0 resultaría en un modelo que copia la distribución a la perfección, lo cual, se alejaría de lo esperado en un modelo generativo.

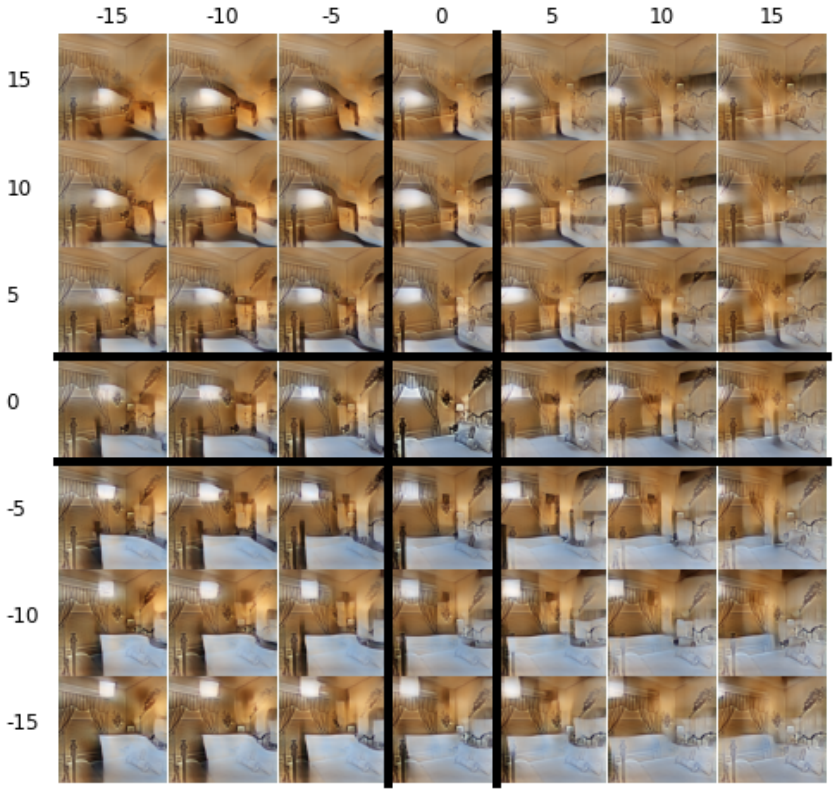
5.2. Modelos basados en CGAN

En esta sección se presentan los resultados obtenidos en las diferentes arquitecturas basadas en una cGAN, con las modificaciones planteadas en el uso de conexiones tipo U-Net en el generador.

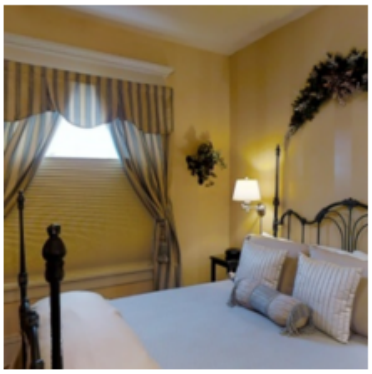
5.2.1. CGAN con U-Net

En este primer acercamiento se utiliza una GAN con una arquitectura descrita en la sección 4.2.1 diseñada para generar una imagen de 256x256 pixeles a partir de una imagen de referencia

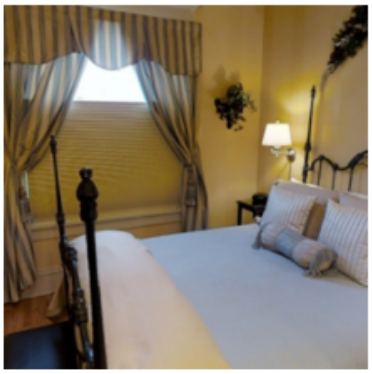
del mismo tamaño y dos parámetros (ángulo horizontal y vertical). Este modelo fue entrenado durante 300 épocas en una tarjeta Nvidia Quadro en un tiempo aproximado de 7 días.



(a) Imágenes generadas a partir de una misma referencia.



(b) Imagen de referencia.



(c) Imagen objetivo $-10°$ vertical y $-10°$ horizontal.

(d) Imagen generada.

Figura 5.4: Imágenes generadas con arquitectura basada en CGAN utilizando conexiones U-Net.

Como se puede apreciar en la figura 5.4a los resultados de este entrenamiento producen imágenes un poco difusas, en las cuales se puede apreciar la variación del ángulo que se busca aplicar, por ejemplo, si se comparan las imágenes de una misma columna podemos apreciar la aparición de un objeto de color blanco mientras el ángulo es menor y su desaparición si se aumenta este. Este objeto como se puede apreciar en la imagen de referencia (Figura 5.4b) o en la imagen objetivo (Figura 5.4c) es una cama de color blanco que aparece en la toma a medida que la cámara enfoca hacia el suelo. De la misma manera al observar las imágenes de una fila

se puede apreciar un rectángulo blanco en la parte superior de cada imagen, el cual al igual que la cama se desplaza a la izquierda o la derecha según el giro de la cámara. Si se pone atención a estas estructuras podemos ver que la modificación de los ángulos es congruente con lo esperado.

Cuando las imágenes se aprecian con un tamaño pequeño como el que se tiene en la figura 5.4a, se notan los patrones descritos anteriormente y se observa un buen comportamiento para el cambio de ángulo, sin embargo, al agrandar una imagen para observarla con mayor calidad (Figura 5.4d), se aprecia una tenue silueta que asemeja una marca de agua, la cual, si se compara con la figura 5.4b se puede apreciar que de esta imagen es de donde proviene esta marca. Esta está principalmente compuesta por características como bordes y esquinas, y estas características aparecen en la imagen debido a que en la arquitectura del generador se emplean las conexiones tipo U-Net, es decir, existen conexiones de salto residuales que se concatenan en los mapas de características para cada nivel de resolución, lo que provoca que esta información pase de la entrada a la salida después de pasar por tan solo 6 capas convolucionales, lo cual no es suficiente para desplazar las estructuras. Como se plantea en [22], la arquitectura es muy robusta para conservar características de la imagen original, lo cual en el caso de este problema específico en donde se busca resolver el cambio de ángulo y requiere necesariamente un desplazamiento de estas características podría no llegar a ser el mejor acercamiento.

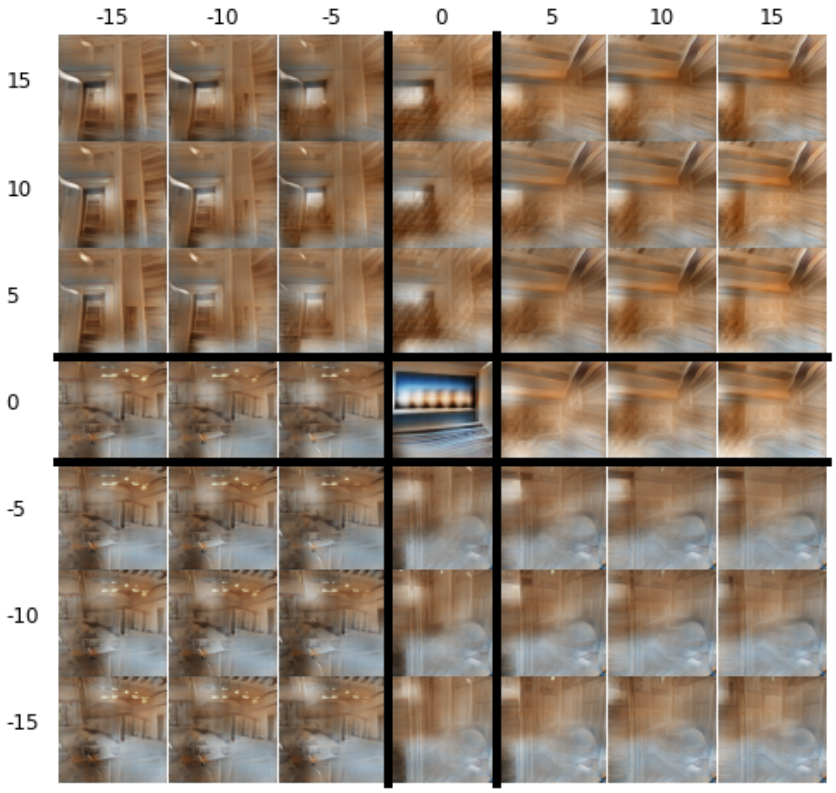
	IS	FID
GAN U-Net	43.7658	61.9843

Tabla 5.2: Métricas para CGAN con U-Net.

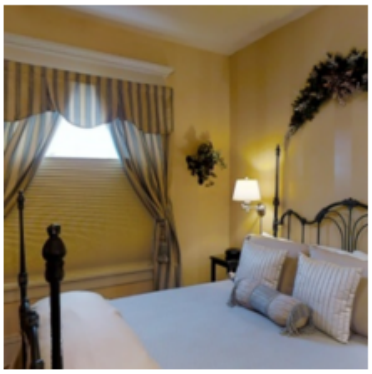
Aunque visualmente las siluetas que aparecen dan la impresión de un resultado no muy bueno en términos de percepción visual, en el caso de las métricas presentadas en la tabla 5.2, se puede apreciar que el valor del IS es bastante alto obteniendo 43.76 de 49 puntos y la distancia de 61.9843 no es una distancia muy alejada de la distribución real si se toma en cuenta que se obtuvo una distancia de 45.7902 entre los conjuntos de entrenamiento y validación. Esto nos muestra que los resultados de esta red son bastante buenos para las dos métricas empleadas en la evaluación de los modelos.

5.2.2. CGAN sin U-Net

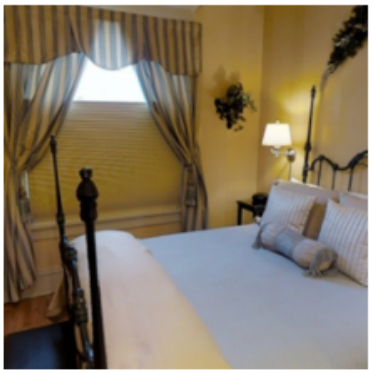
Continuando con las arquitecturas basadas en una CGAN, se implementó la arquitectura descrita en la sección 4.2.2. La cual es muy parecida a la arquitectura utilizada en el entrenamiento anterior pero sin las conexiones U-Net. Con esta modificación se espera que las imágenes generadas se produzcan sin la marca de agua que aparece en el modelo anterior. Este modelo fue entrenado durante 300 épocas en una GPU Nvidia 2080 Super, lo cual tomó poco más de 5 días.



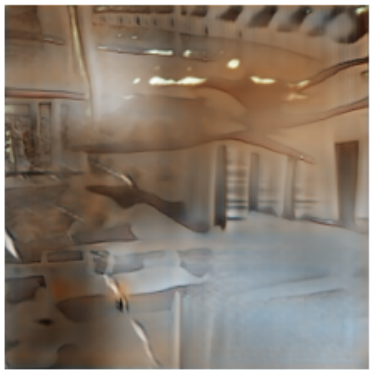
(a) Imágenes generadas a partir de una misma referencia.



(b) Imagen de referencia.



(c) Imagen objetivo $-10°$ vertical y $-10°$ horizontal.



(d) Imagen generada.

Figura 5.5: Imágenes generadas con arquitectura basada en CGAN sin conexiones U-Net.

Los resultados obtenidos no resultan ser muy buenos, ya que se puede apreciar que las imágenes generadas de la figura 5.5a son borrosas y con elementos irreconocibles. Al observar estas imágenes se aprecia una macha blanca en la parte inferior, la cual se desplaza de acuerdo al ángulo indicado, sin embargo, esta mancha no es suficiente para poder identificar el objeto del que se trata. Al realizar un acercamiento a la imagen para obtener una visualización más detallada como en la figura 5.5d, se aprecia claramente que las imágenes generadas no conservan de forma adecuada los detalles de la imagen de referencia (Figura 5.5b), además de que se generan artefactos que no pertenecen a esta.

En general estas imágenes se ven borrosas, comportamiento que era de esperarse según lo propuesto en [21], donde se indica que al emplear la pérdida $\mathcal{L}_{L1}$ junto con la pérdida adversarial, las imágenes se emborronan si no se usa una arquitectura tipo U-Net, sin embargo, se decidió aplicar de igual manera la modificación en las conexiones para buscar eliminar el efecto de la marca de agua que aparece en la arquitectura que sí incluye las conexiones.

Los puntajes obtenidos por este modelo en las métricas (Tabla 5.3) tienen un IS muy bajo de 5.865 y una distancia bastante grande de 150.1266, lo que nos habla de la mala calidad de las imágenes generadas y el bajo rendimiento del modelo. Este valor tan bajo en el IS es generado por la poca definición de los objetos de la imagen, ya que al momento de clasificar, la red InceptionV3 no tiene una gran seguridad de a qué clase pertenece y otorga resultados dispersos. Para la FID se puede apreciar claramente que las imágenes no son parecidas a las del conjunto de datos, sin embargo, no se presenta un valor extremadamente alto, ya que aún se conserva una relación general de color.

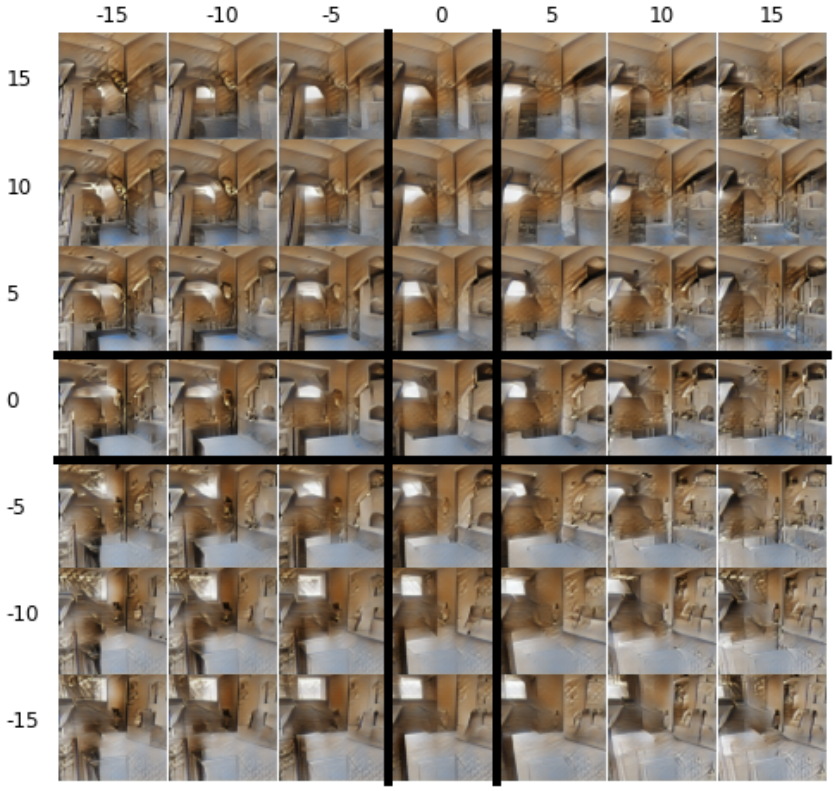
	IS	FID
GAN sin U-Net	5.865	150.1266

Tabla 5.3: Métricas para CGAN sin U-Net.

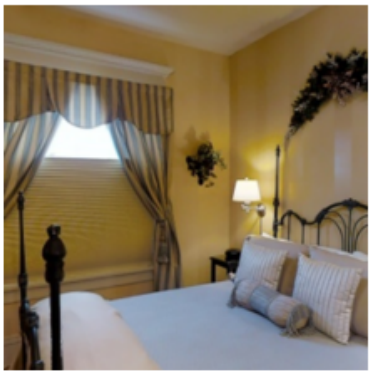
5.2.3. CGAN híbrida

Esta arquitectura descrita en la sección 4.2.3, fue entrenada de la misma manera que los dos modelos anteriores, teniendo un tiempo de entrenamiento similar (5 días aproximados) en una GPU Nvidia 2080 Super. En este modelo se espera llegar a un punto medio entre los dos modelos anteriores, ya que se busca que las imágenes no sean tan borrosas como cuando no tiene conexiones, pero también se espera que las imágenes generadas no tengan las marcas de agua que se producen en el modelo que sí tiene las conexiones.

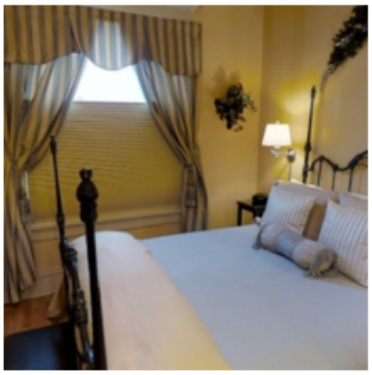
Como se buscaba con los cambios propuestos, estas imágenes resultan menos borrosas que las obtenidas sin las conexiones U-Net, sin embargo, la calidad de las imágenes no resulta ser mejor que las del modelo que tiene todas las conexiones. Si bien se pueden apreciar de forma correcta los desplazamientos de las estructuras más generales como en la figura 5.6a, los detalles más finos terminan por perderse, como se puede apreciar en el acercamiento de la imagen 5.6d.



(a) Imágenes generadas con diferentes ángulos a partir de una misma referencia.



(b) Imagen de referencia.



(c) Imagen objetivo $-10°$ vertical y $-10°$ horizontal.

(d) Imagen generada.

Figura 5.6: Imágenes generadas con arquitectura basada en cGAN utilizando la mitad de conexiones U-Net.

	IS	FID
GAN híbrida	36.595	89.8635

Tabla 5.4: Métricas para CGAN híbrida.

Para las calificaciones obtenidas se puede apreciar que el IS y la FID se encuentran entre las obtenidas por los dos modelos anteriores. Este resultado es congruente con la calidad visual de

las imágenes generadas. Se puede observar que el uso de las conexiones U-Net le dan la robustez a la arquitectura y que su uso repercute en la calidad de las imágenes generadas.

5.3. StackGAN

En estos experimentos se utilizó la arquitectura descrita en la sección 4.2.4. Empleando dos GANs apiladas se busca obtener imágenes de mejor calidad que las obtenidas en el modelo anterior, para ello la primera GAN (Stage1), genera imágenes de 64x64 para posteriormente pasar por la segunda GAN (Stage2) la cual se encarga de generar imágenes de tamaño completo de 256x256 pixeles.

5.3.1. Stage1

Cada uno de los modelos presentados para la Stage 1 fueron entrenados en una tarjeta Nvidia 2080 Super en un tiempo aproximado de 1.5 días. Las imágenes generadas en esta Stage no se espera que tengan una calidad muy alta, ya que son imágenes de 64x64 pixeles, sin embargo, se espera que se pueda observar de forma correcta el desplazamiento de las estructuras más generales. Para estar consientes del nivel de detalle máximo que se podría obtener basta con mirar la figura 5.2b, la cual es una imagen del conjunto de datos, pero a resolución de 64x64 pixeles.

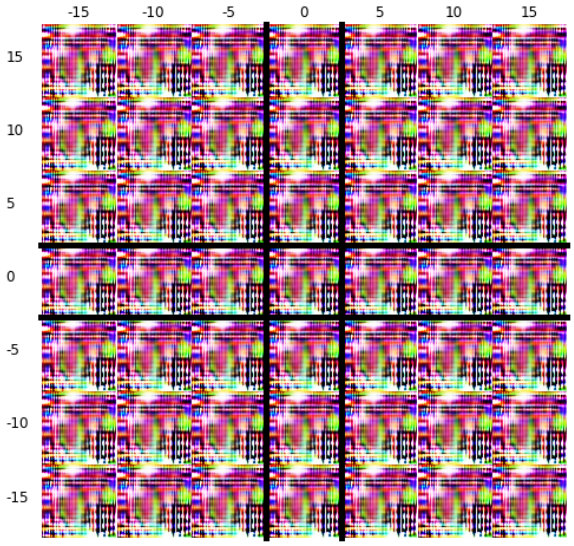
La función de pérdida en esta GAN es la siguiente:

$$\mathcal{L}_{Stage1} = \min_{G}\max_{D} \mathbb{E}_{x\sim p_{data}(x)}[logD(x|p,r)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|p,r)))] + \lambda_1 \mathcal{L}_{L1}(G(z|p,r)) \tag{5.1}$$

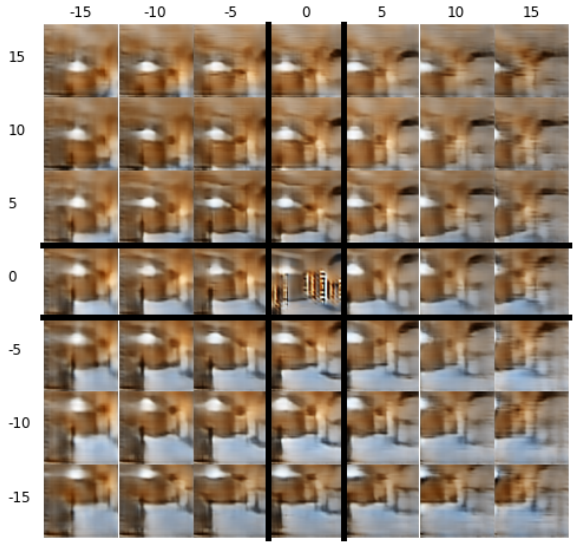
Lo que representa la pérdida adversarial mas la pérdida L1, es decir:

$$\mathcal{L}_{Stage1} = \mathcal{L}_{adversarial}(G,D) + \lambda_1 \mathcal{L}_{L1}(G(z|p,r)) \tag{5.2}$$

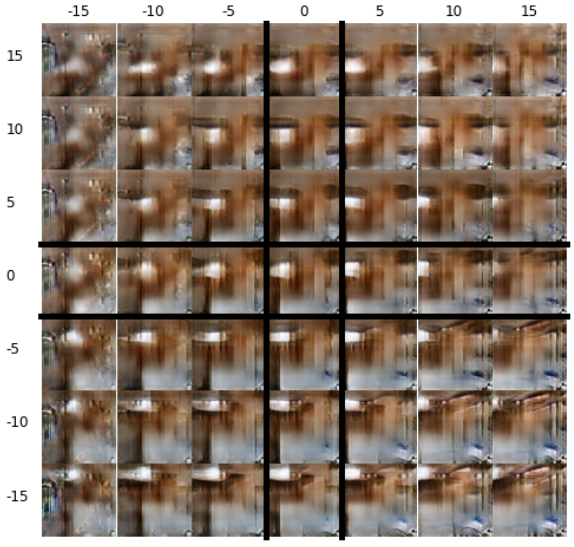
Como se puede observar la pérdida $\mathcal{L}_{L1}$ se encuentra multiplicada por un factor λ_1, de acuerdo a [21], se utilizó $\lambda = 100$ en su modelo, por lo cual se tomó este valor como referencia y se varió este factor de 100 a 0 con pasos de 25, además de usar valores más altos de 200 y 500 para con la finalidad de buscar el mejor resultado. En el caso de 0 la función de pérdida solo toma en consideración la pérdida adversarial.



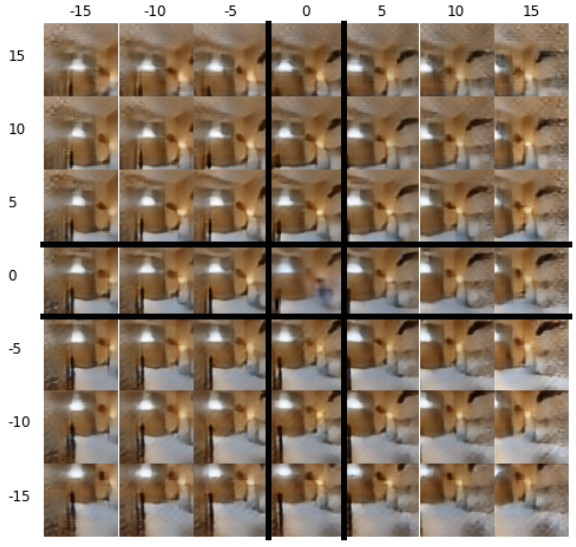
(a) $\lambda_1 = 0$.



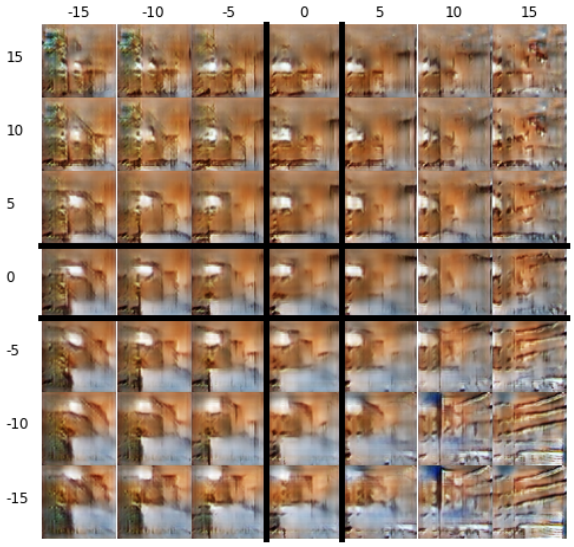
(b) $\lambda_1 = 25$.



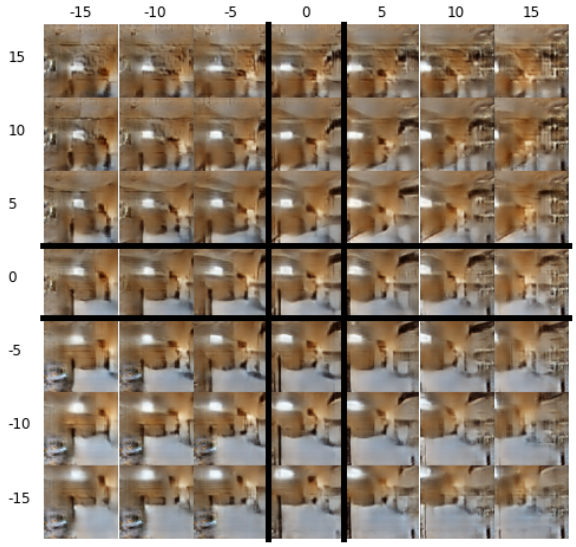
(c) $\lambda_1 = 50$.



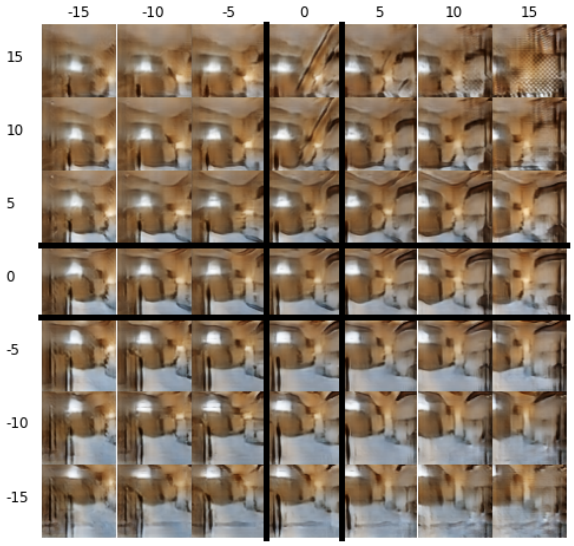
(d) $\lambda_1 = 75$.



(e) $\lambda_1 = 100$.

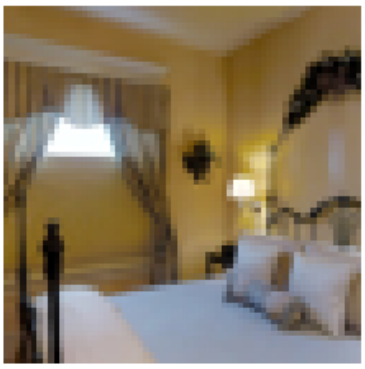


(f) $\lambda_1 = 200$.

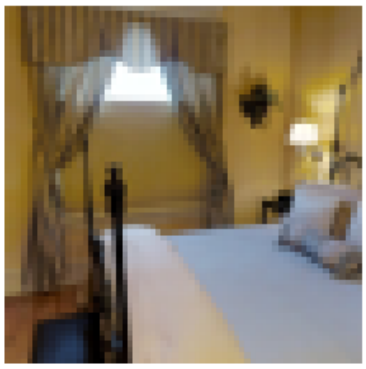


(g) $\lambda_1 = 500$.

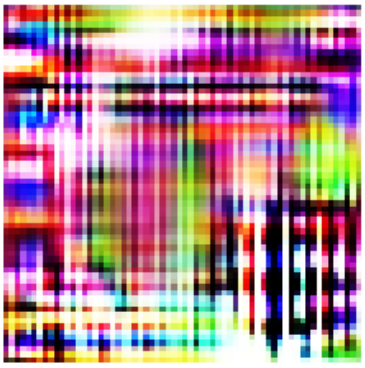
Figura 5.7: Composición de imágenes generadas por la Stage1 con diferentes valores de λ_1.



(a) Imagen usada como referencia.



(b) Imagen objetivo con $-10°$ vertical y $-10°$ horizontal a partir de la imagen referencia.



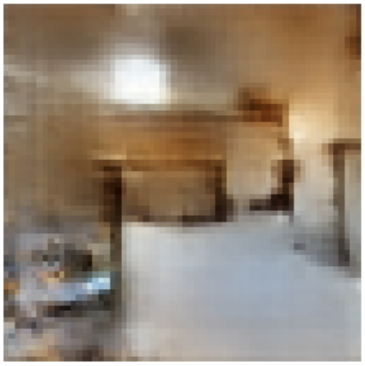
(c) $\lambda_1 = 0$.

(d) $\lambda_1 = 25$.

(e) $\lambda_1 = 50$.

(f) $\lambda_1 = 75$.

(g) $\lambda_1 = 100$.

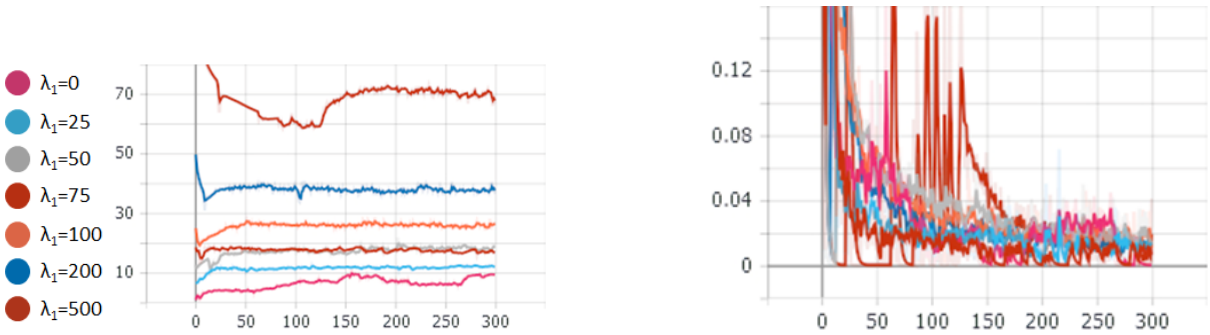


(h) $\lambda_1 = 200$.

(i) $\lambda_1 = 500$.

Figura 5.8: Acercamiento de imágenes generadas por la Stage1 con diferentes valores de λ_1.

Como podemos observar en la figura 5.8 el peor modelo que se entrenó resulta ser el caso con $\lambda_1 = 0$, donde se observa que las imágenes no mantienen la correspondencia de colores con la referencia y tampoco respeta las estructuras de esta. En este caso como se comentó anteriormente solo se tiene en cuenta la pérdida adversarial, con lo cual podemos observar que esta función de pérdida por sí sola no es suficiente para obtener imágenes de buena calidad. También podemos observar que el agregar la pérdida $\mathcal{L}_{L1}$ permite al modelo mantener una correspondencia de color y formas, sin embargo, hay que escoger correctamente el valor de lambda, ya que el ajustar ese hiperparámetro de forma correcta puede ayudar a mejorar o empeorar la calidad de las imágenes.



(a) Pérdida promedio en Generador.

(b) Pérdida promedio en Discriminador.

Figura 5.9: Gráficas con pérdidas promedio para diferentes valores de λ durante el entrenamiento de la Stage1.

En la gráfica de la figura 5.9a se puede apreciar claramente que mientras incrementa el valor de λ_1 el valor de la pérdida promedio aumenta proporcionalmente, esto es provocado por la pérdida $\mathcal{L}_{L1}$ que se utiliza en el generador. De igual manera se puede apreciar un comportamiento similar entre las diferentes pérdidas, ya que en todos los experimentos se utilizó la misma forma de entrenamiento y la misma función de pérdida con la única diferencia del valor de λ_1.

El discriminador minimiza su pérdida buscando poder diferenciar de manera acertada entre imágenes reales y sintéticas. Como podemos observar en la figura 5.9b la pérdida del discriminador baja rápidamente en las primeras épocas del entrenamiento y posteriormente sigue disminuyendo la pérdida de manera gradual. Como se puede observar en la ecuación 5.1 la pérdida $\mathcal{L}_{L1}$ solo se aplica al generador, por lo que el discriminador en los diferentes experimentos realizados se comporta de una manera muy similar, si bien existen algunos picos en las gráficas estos son provocados principalmente por la aleatoriedad de los datos al momento de realizar el entrenamiento.

	IS	FID 256x256	FID 64x64
StackGAN S1 $\lambda = 0$	1.3578	407.6557	379.0878
StackGAN S1 $\lambda = 25$	18.9574	236.6214	165.8834
StackGAN S1 $\lambda = 50$	27.1375	202.6419	148.0629
StackGAN S1 $\lambda = 75$	38.4807	197.8823	133.1316
StackGAN S1 $\lambda = 100$	35.5364	189.8661	128.8827
StackGAN S1 $\lambda = 200$	46.0390	185.9297	111.6952
StackGAN S1 $\lambda = 500$	47.0023	175.4565	108.9398

Tabla 5.5: Métricas para Stage1.

En la tabla 5.5 se tienen 3 columnas, la primera, contiene el puntaje obtenido por el modelo con el IS. Las otras dos columnas contienen las FID obtenidas para cada uno de los modelos presentados, sin embargo, existe una diferencia promedio de aproximadamente 60 unidades entre ambas. Esta diferencia radica en la forma en que la FID fue calculada, en ambos casos la distancia se calculó entre los mismos conjuntos de imágenes, pero con una modificación en la resolución de las imágenes reales.

Como se describió anteriormente la Stage1 produce imágenes de 64x64 pixeles, por lo que es imposible obtener el nivel de detalle que se puede tener en una imagen de 256x256, como se puede apreciar en la figura 5.2. Por lo tanto realizar el cálculo de la FID entre imágenes generadas de 64x64 contra imágenes reales de 256x256, aunque sí representa qué tan alejadas se encuentran las distribuciones, genera un sesgo a las distancias y una desventaja al comparar con otros modelos. Con la finalidad de solventar este problema, se calculó nuevamente la FID, pero esta vez primero se redujo el tamaño de las imágenes reales a 64x64 pixeles y después se realizó el cálculo de las distancias. Como se puede observar en la tabla 5.5 el comportamiento para ambas distancias es similar, sin embargo en términos de puntuación se nota claramente la mejora cuando se compara contra imágenes del mismo tamaño.

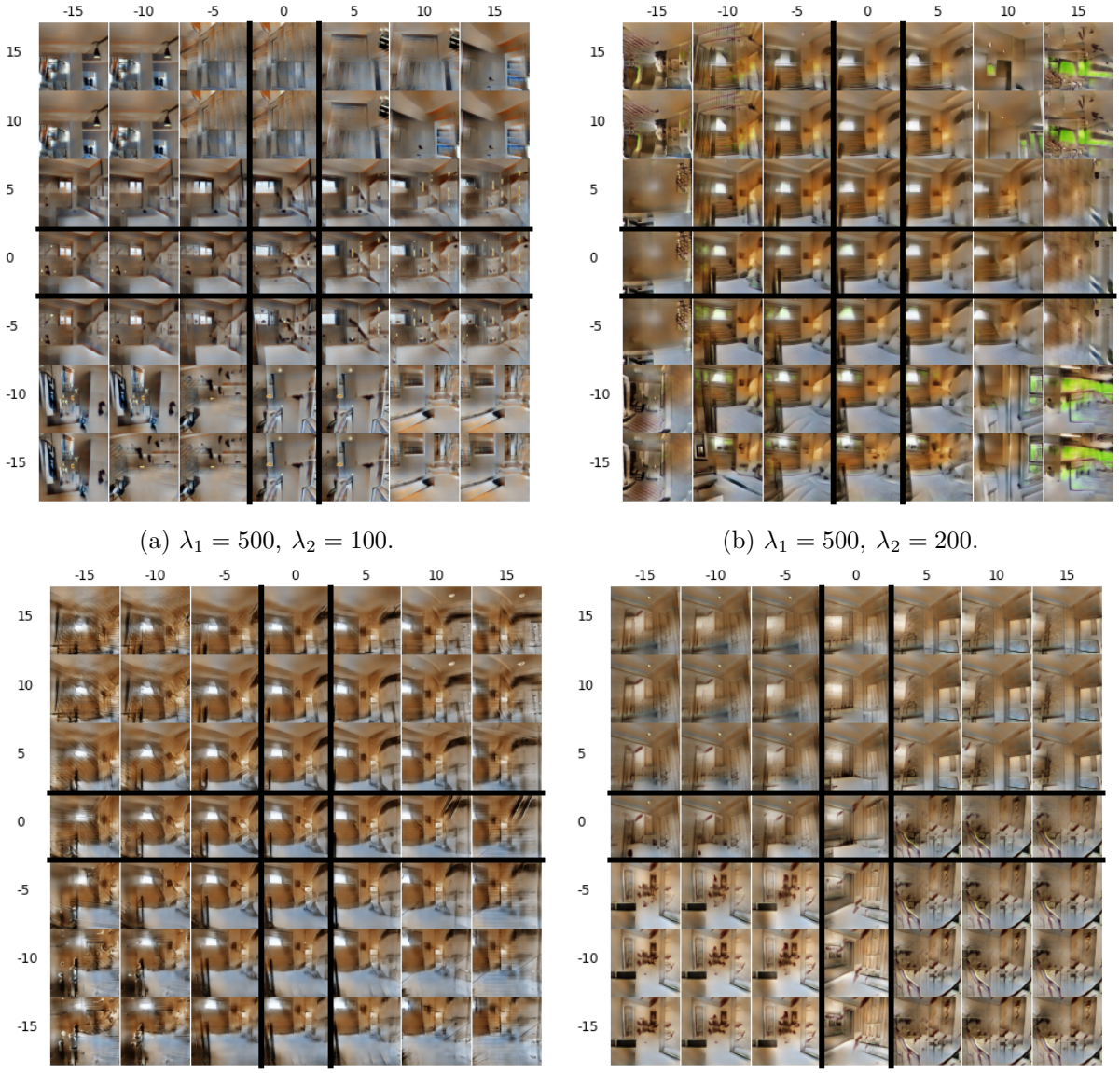
Tanto el IS como la FID son mejores mientras más grande se hace el valor de λ_1. Para el caso de $\lambda_1 = 0$ donde las imágenes son irreconocibles (figura 5.7a) el IS es de 1.3578 lo cual es el valor más bajo obtenido y se encuentra muy cercano al valor mínimo de 0. De igual modo la FID tiene el valor más alto reportado, lo cual nos dice que las imágenes se encuentran más alejadas de la distribución del conjunto de validación que los demás modelos, ya que no conservan relación alguna entre color o forma con las reales.

En el caso del modelo entrenado con $\lambda_1 = 500$ se puede observar que el IS es bastante alto con 47 de los 49 puntos, lo que nos indica que se está generando de manera correcta el desplazamiento de los ángulos. Sin embargo para la FID el valor es de 175 y 108, el cual no es tan bueno. Este comportamiento se debe principalmente a que las imágenes producidas son borrosas, por lo que la relación entre distribuciones no es muy cercana.

5.3.2. Stage2

En esta Stage2 se busca aumentar la calidad de las imágenes obtenidas en la Stage1 y para ello se realizaron los entrenamientos de distintos modelos, en los cuales se varió el valor de λ_2 y el modelo base de la Stage1. Cada uno de estos entrenamientos tomó un tiempo aproximado de 4 días.

Se realizaron entrenamientos con dos diferentes modelos base de la Stage1, primero, se tomó el mejor modelo de la Stage1, el cual fue el entrenado con $\lambda_1 = 500$. Y posteriormente se varió el parámetro λ_2 con valores de 100, 200 y 500, los cuales fueron los mejores valores para la Stage1. El segundo modelo base que se utilizó fue el entrenado con $\lambda_1 = 75$, si bien este modelo no obtuvo un desempeño muy alto en la Stage1, se buscaba comprobar si un modelo podía mejorar en la Stage2 a pesar de un rendimiento medio en la Stage1.



(a) $\lambda_1 = 500,\ \lambda_2 = 100$.

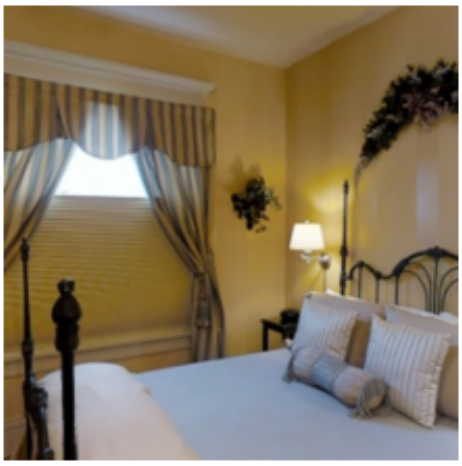
(b) $\lambda_1 = 500,\ \lambda_2 = 200$.

(c) $\lambda_1 = 500,\ \lambda_2 = 500$.

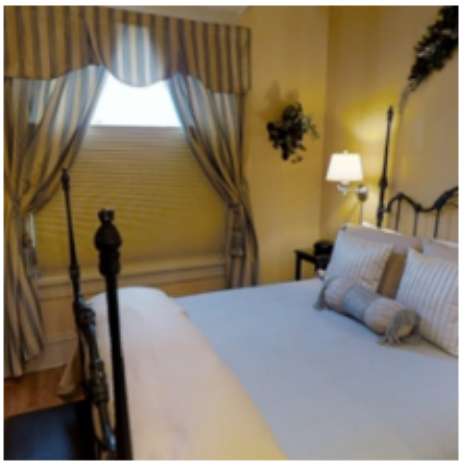
(d) $\lambda_1 = 75,\ \lambda_2 = 75$.

Figura 5.10: Composición de imágenes generadas por la Stage2.

Como podemos apreciar en la composición de imágenes generadas de la figura 5.10, parece que las imágenes generadas están respetando el desplazamiento de las estructuras de acuerdo al ángulo indicado. Sin embargo si ponemos atención a las imágenes más alejadas, es decir, con ángulos más grandes, podemos notar la aparición de algunos artefactos que no existen en la imagen de referencia, además de que su calidad es baja y tienen un acabado difuso. Al hacer un acercamiento a las imágenes como en la figura 5.11, es más notorio este comportamiento, ya que se pierden por completo los detalles de la cama o la ventana.

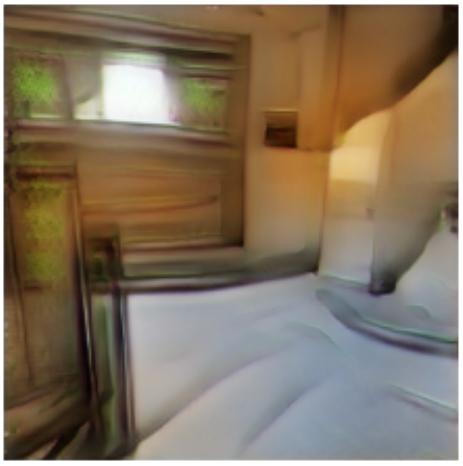


(a) Imagen utilizada como referencia.



(b) Imagen objetivo con $-10°$ vertical y $-10°$ horizontal.

(c) $\lambda_1 = 500,\ \lambda_2 = 100$.



(d) $\lambda_1 = 500,\ \lambda_2 = 200$.

(e) $\lambda_1 = 500,\ \lambda_2 = 500$.

(f) $\lambda_1 = 75,\ \lambda_2 = 75$.

Figura 5.11: Acercamiento de imágenes generadas por la Stage2.

	IS	FID 256x256
S2 $\lambda_1 = 75$ $\lambda_2 = 75$	11.1007	116.2318
S2 $\lambda_1 = 500$ $\lambda_2 = 100$	12.6864	121.2386
S2 $\lambda_1 = 500$ $\lambda_2 = 200$	35.0730	95.7368
S2 $\lambda_1 = 500$ $\lambda_2 = 500$	48.3778	105.4841

Tabla 5.6: Métricas para Stage2.

Si nos enfocamos en las métricas podemos observar un IS muy alto para el caso del modelo con $\lambda_2 = 500$, lo que nos habla de que las estructuras se desplazan de forma correcta entre cada imagen. Sin embargo su FID nos indica que la distribución de las imágenes generadas no se encuentra muy cerca de las imágenes reales. Comparando las métricas entre los diferentes modelos podemos observar que para los modelos entrenados con el modelo de la Stage1 con $\lambda_1 = 500$ disminuyen su IS mientras su λ_2 baja, sin embargo la mejor FID está presente en el modelo con $\lambda_2 = 200$, aunque el peor de los tres para las dos métricas resulta ser el modelo con $\lambda_2 = 100$.

Analizando los resultados obtenidos del modelo entrenado con $\lambda_1 = \lambda_2 = 75$ podemos observar un desempeño muy bajo, ya que el modelo de la Stage1 por sí solo logró obtener un IS de 38.4, a diferencia del modelo entrenado en la Stage2 que solo logró 11.1. Esto nos indica que no cualquier modelo puede mejorar al pasar por la Stage2 y que tratar de ocupar la arquitectura de la StackGAN no resulta ser la mejor aproximación para la resolución del problema planteado.

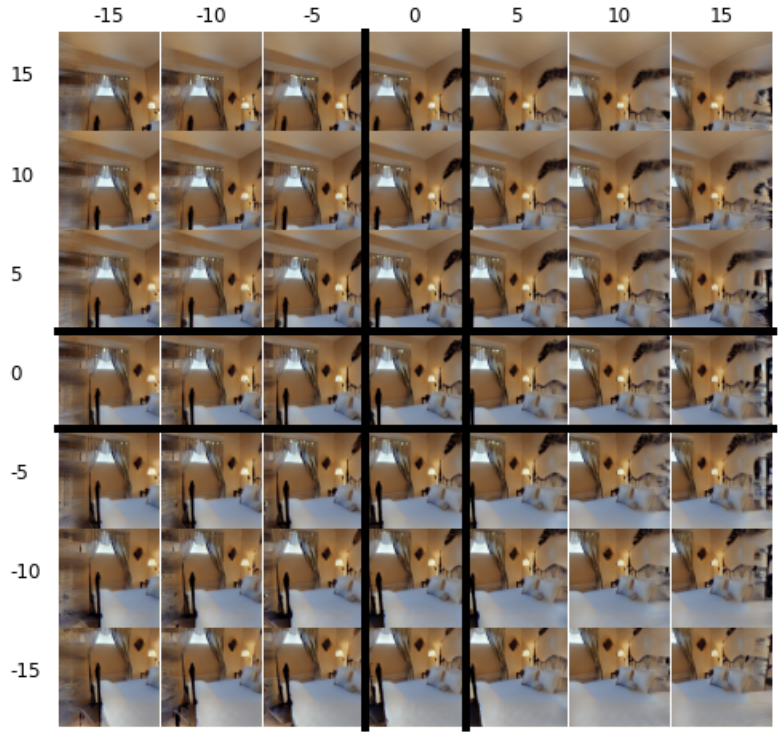
5.4. FillGAN

5.4.1. Con ángulos

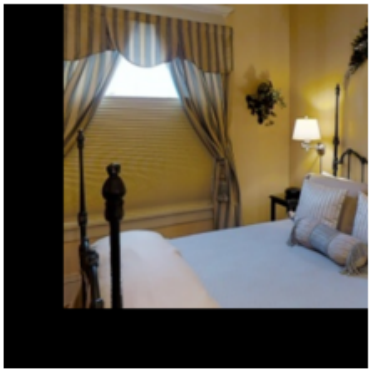
Para la arquitectura descrita en la sección 4.2.5, se entrenó un modelo durante 300 épocas, lo cual tomó un tiempo cercano a 7.5 días utilizando una GPU Nvidia 2080 Super.

Si observamos las imágenes mostradas en la composición de la figura 5.12a podemos apreciar que las imágenes tienen un desplazamiento esperado en las estructuras más importantes, además podemos observar que las franjas que se introducen en las imágenes de referencia (como las de la figura 5.12b) desaparecen de las imágenes generadas y se rellenan respetando el color general de la imagen, además de que se aprecia una coherencia de las formas que se introducen. Al realizar un acercamiento a la imagen generada como en la figura 5.12d, se observa que la imagen conserva bastante bien la estructura de los objetos, si bien se puede apreciar una variación en las texturas de estos la forma en general es consistente con la imagen de entrada.

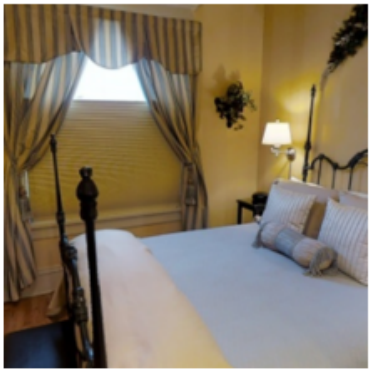
Si observamos con detenimiento, las zonas que no tienen información en la imagen de referencia (recuadros negros) se notan en la imagen de salida, ya que estas zonas son rellenadas y en algunas imágenes se agregan algunos artefactos que no se esperarían observar. De igual manera en algunas de las imágenes generadas se presenta un cambio sutil de color entre la zona rellenada y la zona que en un principio sí poseía información.



(a) Imágenes generadas a partir de una misma referencia.



(b) Imagen de referencia.



(c) Imagen objetivo $-10°$ vertical y $-10°$ horizontal.

(d) Imagen generada.

Figura 5.12: Imágenes generadas con arquitectura fillGAN con ángulos.

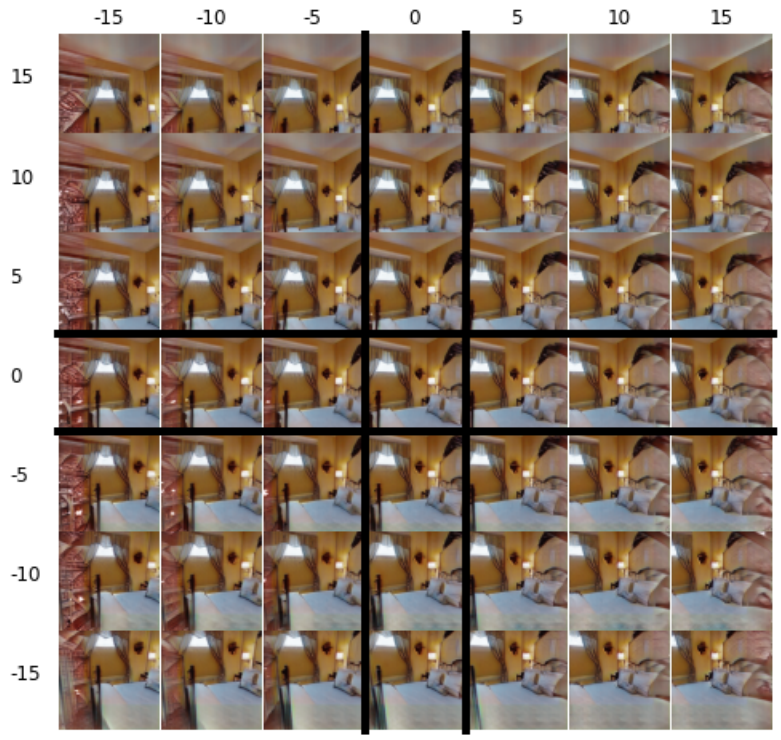
En general las imágenes obtenidas poseen un acabado visual bueno y al evaluar estas imágenes con las métricas planteadas también se obtienen buenos resultados. Como se puede apreciar en la tabla 5.7 el IS obtenido por este modelo es de 48.6991, el cual si se compara con el IS obtenido por el conjunto de imágenes reales de 48.895 es muy cercano y un valor muy alto. Con este valor verificamos que efectivamente está realizando el desplazamiento de las imágenes según lo esperado. Si revisamos el valor de la FID obtenemos 51.8953 unidades, si comparamos esta distancia con el baseline propuesto entre el conjunto de entrenamiento y validación de 45.7902, está apenas 6 unidades por encima de él, por lo que este valor de la FID es muy bueno para un conjunto de imágenes sintéticas.

	IS	FID vs Train
fillGAN con ángulos	48.6991	51.8953

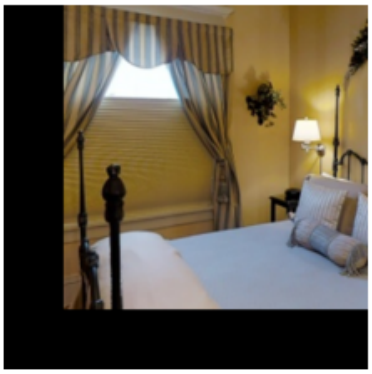
Tabla 5.7: Métricas para fillGAN con ángulos.

5.4.2. Sin ángulos

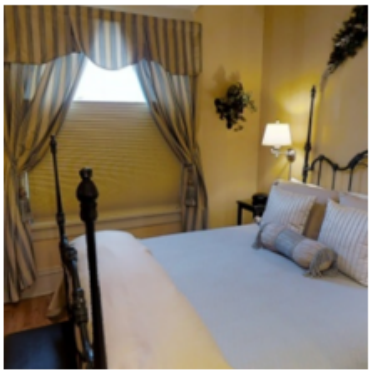
En esta sección se reportan los resultados del entrenamiento de un modelo empleando la arquitectura descrita en la sección 4.2.5, la cual es una variante de la arquitectura presentada anteriormente. Este modelo fue entrenado igualmente durante 300 épocas y tomó un tiempo de aproximadamente 6 días.



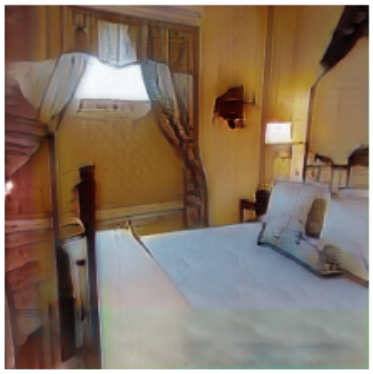
(a) Imágenes generadas a partir de una misma referencia.



(b) Imagen de referencia.



(c) Imagen objetivo $-10°$ vertical y $-10°$ horizontal.



(d) Imagen generada.

Figura 5.13: Imágenes generadas con arquitectura fillGAN sin ángulos.

En la figura 5.13a se aprecia que las imágenes sintéticas que genera el modelo llenan los espacios faltantes con estructuras que parecen ser congruentes con el resto de la imagen, sin embargo, las zonas sin información que se presentan en la imagen de referencia, cuando son rellenadas no se rellenan con colores completamente congruentes con la imagen original. Por ejemplo en la columna -15 de la figura 5.13a se aprecia una franja roja del lado izquierdo, la cual concuerda con la zona sin información de las imágenes de referencia.

Si realizamos un acercamiento a las imágenes podemos observar que las texturas cambian un poco, pero siguen conservando elementos de la imagen original, además de que los espacios son rellenados con formas esperadas en estos.

Haciendo la comparación visual entre la figura 5.12a y la figura 5.13a parece que el acabado de las imágenes es mejor al emplear los ángulos como parte del modelo, ya que las franjas que se rellenan tienen una mejor correspondencia de color con las imágenes esperadas, sin embargo al evaluar ambos modelos con las métricas propuestas el modelo sin ángulos tiene un desempeño mayor. Para el IS la mejora no es muy grande, ya que solo aumenta una décima, pero por otro lado la FID tiene una mejora significativa, ya que esta se reduce cerca de un 30 %. Siendo este valor el mejor obtenido de todos los experimentos realizados en este trabajo.

	IS	FID vs Train
fillGAN sin ángulos	48.79	35.427

Tabla 5.8: Métricas para fillGAN sin ángulos.

5.5. Análisis de resultados

En la tabla 5.9 se incluyen los resultados de las evaluaciones de los diferentes modelos entrenados. Como se puede observar el peor modelo resultó ser la Stage1 con $\lambda_1 = 0$ en donde las imágenes que se obtienen no presentan ninguna relación con las imágenes de los conjuntos de entrenamiento ni validación. En general podemos observar que las calificaciones obtenidas por los modelos entrenados bajo la arquitectura de la StackGAN resultan tener un rendimiento bajo al compararse con los otros dos tipos de arquitecturas, por lo que probablemente no es el mejor acercamiento para buscar resolver el problema planteado.

	IS	FID
Entrenamiento***	48.9427	0.0 *
Validación***	48.8950	45.7902*
GAN con U-NET	43.7658	61.9843
GAN sin U-NET	5.865	150.1266
GAN hibrida	36.595	89.8635
StackGAN S1 $\lambda = 0$	1.3578	379.0878**
StackGAN S1 $\lambda_1 = 25$	18.9574	165.8834**
StackGAN S1 $\lambda_1 = 50$	27.1375	148.0629**
StackGAN S1 $\lambda_1 = 75$	38.4807	133.1316**
StackGAN S1 $\lambda_1 = 100$	35.5364	128.8827**
StackGAN S1 $\lambda_1 = 200$	46.0390	111.6952**
StackGAN S1 $\lambda_1 = 500$	47.0023	108.9398**
S2 $\lambda_1 = 75$ $\lambda_2 = 75$	11.1007	116.2318
S2 $\lambda_1 = 500$ $\lambda_2 = 100$	12.6864	121.2386
S2 $\lambda_1 = 500$ $\lambda_2 = 200$	35.0730	95.7368
S2 $\lambda_1 = 500$ $\lambda_2 = 500$	48.3778	105.4841
fillGAN con ángulos	48.6991	51.8953
fillGAN sin ángulos	48.79	35.427

Tabla 5.9: Tabla recopilatoria de las métricas obtenidas en los modelos entrenados.
*Valores de FID calculados contra el conjunto de entrenamiento.
**Valores FID calculados contra imágenes reales de 64x64 pixeles.
***Valores de referencia.

El modelo cGAN con conexiones U-Net tiene un desempeño aceptable, sin embargo al comparar la calidad de las imágenes y las métricas con los modelos entrenados bajo la aproximación de rellenado se nota una mejoría por estos últimos.

Si comparamos los métodos usados por la fillGAN y la StackGAN podemos observar que a grandes rasgos la tarea que se busca resolver en la Stage1 corresponde con el desplazamiento que se calcula por métodos tradicionales antes de pasar la imagen a la fillGAN, posteriormente el trabajo que se realiza en la Stage2 y en la fillGAN resulta ser bastante similar. Esta comparación nos permite darnos cuenta de que en algunos problemas el uso de redes neuronales no resulta ser la mejor solución, sin embargo el caso de la corrección de perspectiva y rellenado sí resulta ser una tarea que se desempeña de mejor manera por una red neuronal.

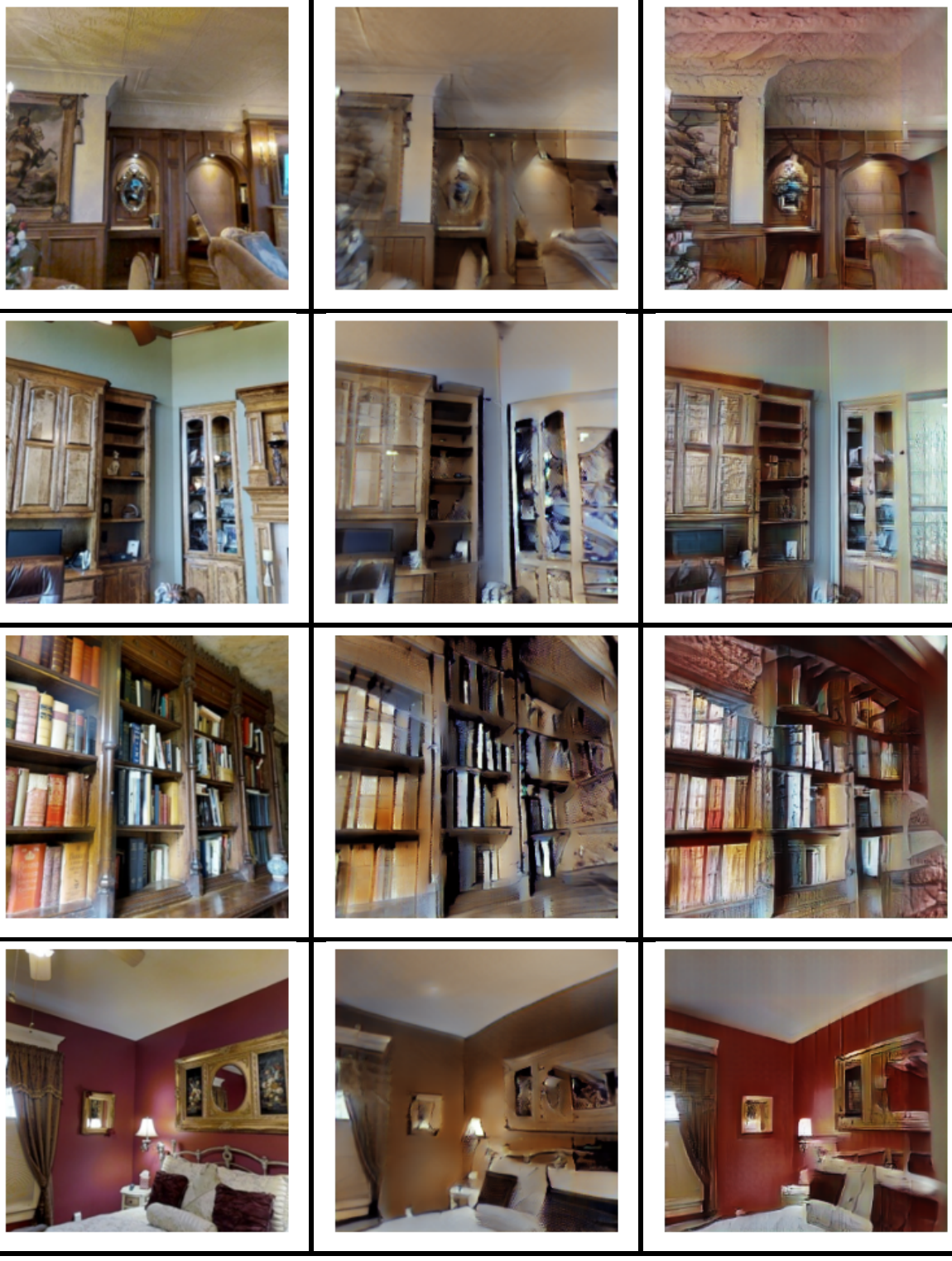
Reales	fillGAN con ángulos	fillGAN sin ángulos
		

Tabla 5.10: Comparativa entre imágenes reales y las generadas por ambas fillGAN en diferentes locaciones.

Como podemos apreciar en la tabla 5.10 las imágenes generadas por la fillGAN con ángulos, rellena con una mejor relación de color las zonas sin información, ya que se conserva el color general de la imagen, a diferencia del modelo que no emplea los ángulos, en el cual es clara la

franja de información faltante y se aprecia un color mayormente rojizo en esta zona. Si comparamos las texturas que se presentan en las imágenes podemos observar una mayor presencia de estas en las imágenes del modelo sin ángulos, lo que ayuda a que las imágenes tengan detalles más definidos y se vean menos borrosas, características que ayudan a mejorar los puntajes obtenidos en las métricas.

Al comparar las imágenes lado a lado podemos observar que en realidad las imágenes sintéticas son de muy buena calidad y se nota el parecido con las imágenes reales, si bien estas no son perfectas, nos permiten observar de manera clara las estructuras más representativas de la escena.

CAPÍTULO 6

Conclusiones

6.1. Conclusiones

En este trabajo se logró entrenar un modelo haciendo uso de una arquitectura GAN que permitiera generar imágenes realistas a partir de una imagen de referencia, de manera que la imagen generada varíe el ángulo de la toma de acuerdo a un parámetro dado, lo que cumple con el objetivo planteado al inicio de este trabajo.

Se implementaron dos de las métricas más populares para la evaluación de modelos generativos (Inception Score y Fréchet Inception Distance). En estas métricas podemos observar una correlación entre la calidad visual de las imágenes y los puntajes reportados. Se probaron diferentes arquitecturas, aproximaciones y variaciones de parámetros para poder lograr los mejores resultados posibles, donde se obtuvieron buenas calificaciones para los mejores modelos entrenados, lo cual nos habla de la buena calidad de las imágenes obtenidas.

Se encontró que para este problema la mejor solución no se consigue haciendo uso únicamente de redes neuronales. Por el contrario, al emplear un procesamiento previo de los datos por métodos tradicionales, se pueden obtener mejores resultados. Al hacer esto la complejidad de la tarea a resolver por la red cambia, con lo cual la red neuronal puede resolver de mejor manera el problema planteado. Si bien la tarea que llevan a cabo las fillGAN no es la misma que las otras arquitecturas utilizadas en este trabajo, el problema que buscan resolver es el mismo. Con esto podemos observar que aunque una red neuronal es capaz de resolver una tarea compleja, utilizar un preprocesamiento previo con algún otro método puede mejorar los resultados considerablemente. Por lo que no hay que dejar toda la solución en manos de una red neuronal.

A partir de experimentos previos a este trabajo, se determinó que los ángulos máximos para obtener resultados aceptables es de $-15°$ a $15°$. Originalmente se planteó utilizar un rango de $-30°$ a $30°$, sin embargo los resultados obtenidos generaban una gran cantidad de artefactos en las imágenes modificadas cuando se aplicaba una variación superior a $15°$ por lo cual se redujo el rango de ángulos.

En el caso de las arquitecturas basadas en pix2pix comprobamos que como se menciona en el artículo original [21], el uso de las conexiones tipo U-Net ayudan en gran medida a mejorar las imágenes de salida, además de que la ausencia de estas conexiones producen imágenes borrosas o difusas. También podemos observar que la presencia de estas conexiones ayudan a mantener las características de la imagen inicial en la imagen generada, haciendo que esta arquitectura sea muy robusta para algunas tareas pero también poco flexible para otras, como en este caso.

Como se observa en los experimentos realizados con la StackGAN, es muy importante la selección correcta en los valores de λ, ya que estos valores nos pueden llevar del peor modelo entrenado a uno medianamente aceptable. Por lo tanto podemos concluir que la selección de estos hiperparámetros es una parte importante al realizar los entrenamientos. Además, podemos

observar que al pasar por la Stage2 los resultados son principalmente borrosos, ya que el valor de λ es muy alto, sin embargo podemos observar que si se utiliza un valor menor también bajan sus puntajes, por lo cual esta no es la mejor aproximación para la resolución de la tarea planteada.

6.2. Trabajo futuro

Este trabajo se centró en generar las imágenes realistas con la modificación de ángulos en la toma, sin embargo el siguiente paso para estas imágenes es el de poder ser empleadas por robots móviles. Este uso se puede realizar sobre diferentes líneas de investigación.

Una de las posibilidades para la continuación de este trabajo es la de emplear las imágenes sintéticas generadas por el modelo para el entrenamiento de un clasificador. En el entrenamiento de este se utilizaría un aumentado de datos con el que se busque reducir los costos de captura de imágenes reales. Para ello es necesario comprobar si estas imágenes son lo suficientemente buenas para poder ser empleadas en un correcto entrenamiento, lo cual basado en los valores reportados en el IS y la FID para el modelo se esperaría que fuera posible.

La segunda línea de trabajo que se puede seguir a partir de las imágenes generadas es la de utilizar un sistema de predicción para un robot, en el cual se busque generar la imagen dado un ángulo y con base en esta se busque predecir la forma en la cual se desplazarían los objetos. Esto se realizaría con la finalidad de saber si el movimiento propuesto es correcto antes de ser realizado por el robot. Con ello se puede evitar realizar movimientos innecesarios y ahorrar tiempo en el desplazamiento de un robot.

Como tercera propuesta está la de utilizar las imágenes sintéticas para reentrenar y afinar los sistemas de reconocimiento de un robot, mientras este se encuentra en estado de inactividad o reposo. Este proceso comúnmente llamado como «Máquinas que sueñan» o en inglés «Dreaming Machines» [35] y parece ser una propuesta alcanzable dada la calidad de las imágenes generadas.

APÉNDICE A

Conexion a servidor

El entrenamiento de los modelos se llevo acabo de manera remota conectando a los servidores del laboratorio de Bio-robótica en la Facultad de Ingeniería, UNAM [36]. Para establecer la comunicación se utilizó una conexión mediante OpenSSH [37].

A.1. Cliente

Para realizar la conexión remota se ingresó mediante el usuario (*user*), la ip y puerto asignados por el soporte del laboratorio (*@unam*). Ademas se utiliza la bandera *-L* la cual permite blindar un puerto para enviarlo desde el servidor al cliente por medio de la conexión. Este puerto (*6060*) servirá para poder visualizar los resultados mediante tensorboard, el cual es ejecutado en el servidor pero gracias a esto, es posible visualizar los resultados desde el cliente.

```
$ssh –L 6060:localhost:6060 user@unam
```

Posteriormente se introdujo la contraseña para autenticarse y establecer la conexión, con lo que resulta posible trabajar desde la terminal del servidor.

A.1.1. Transferencia de archivos

Ademas de la conexión SSH que permite trabajar desde terminal, es necesario realizar transferencias de archivos de manera eficiente e intuitiva, es por ello que se empleó Midnight Commander [38], el cual es un visualizador de archivos GNU que trabaja desde terminal, permite realizar la manipulación de archivos tanto en un servidor como en una maquina local al mismo tiempo.

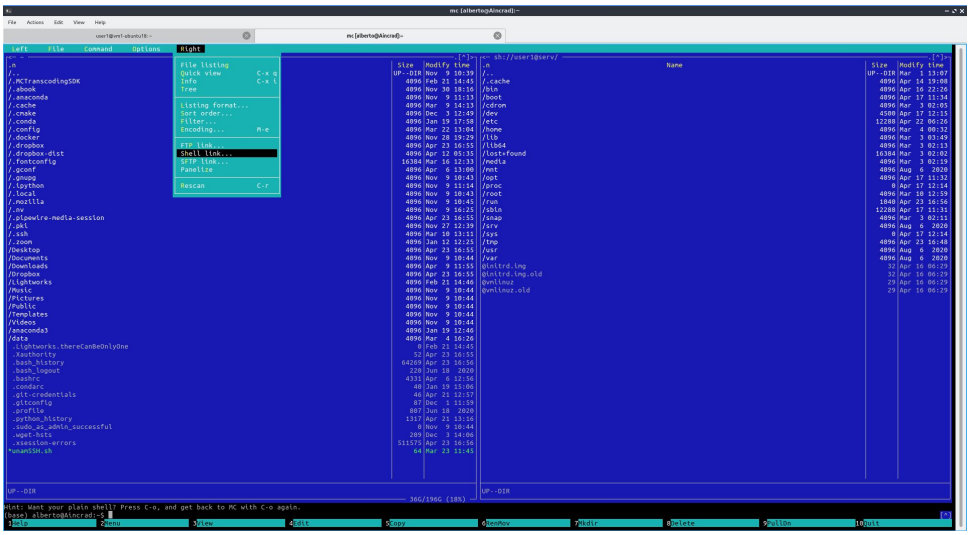


Figura A.1: Interfaz de Midnight Commander.

Dentro de la interfaz se elije uno de los dos paneles para ser el servidor, en este caso el derecho, y se realiza la conexión de la misma manera en que se realiza por SSH. Una vez realizada la autentificación es posible visualizar la estructura de archivos locales y remotos y realizar transferencias entre ambos equipos.

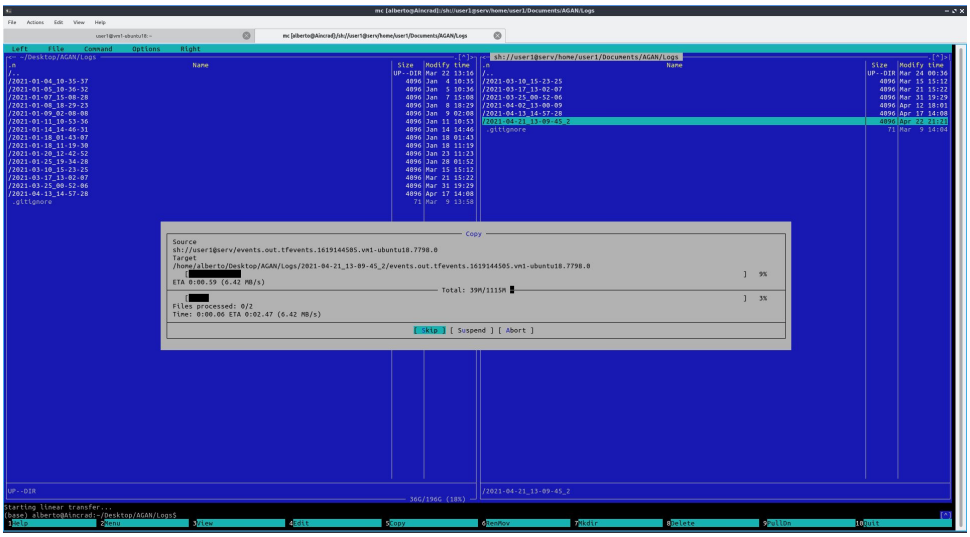


Figura A.2: Transferencia de archivos entre cliente y servidor con Midnight Commander.

A.2. Servidor

Se instalaron los siguientes programas en el servidor:

- Python3.7: para realizar los entrenamientos, el código emplea PyTorch 1.7.1.
- tmux: para trabajar con diferentes terminales en la misma ventana y restablecer sesiones remotas.
- tensorboard: para desplegar los resultados parciales en tiempo real de los entrenamientos.

Además, se instalaron todos los paquetes necesarios en Python 3.7, para correr el código empleado en esta tesis.

Para realizar el entrenamiento se emplea una sesión de tmux con 4 paneles en una ventana.

1. Ejecuta el entrenamiento del modelo.
2. tensorboard: este se corre en la ubicación en la que se guardan los logs y se indica el puerto en el cual trabajará, el cual, coincide con el puerto blindado en la comunicación SSH (*6060*) para poder visualizar los resultados en el equipo remoto.
3. nvidia-smi: una herramienta que permite revisar el estado de las GPU nvidia, este se corre de manera periódica cada segundo haciendo uso del comando watch.
4. htop: permite visualizar el estado del CPU, la memoria y los procesos que se están ejecutando.

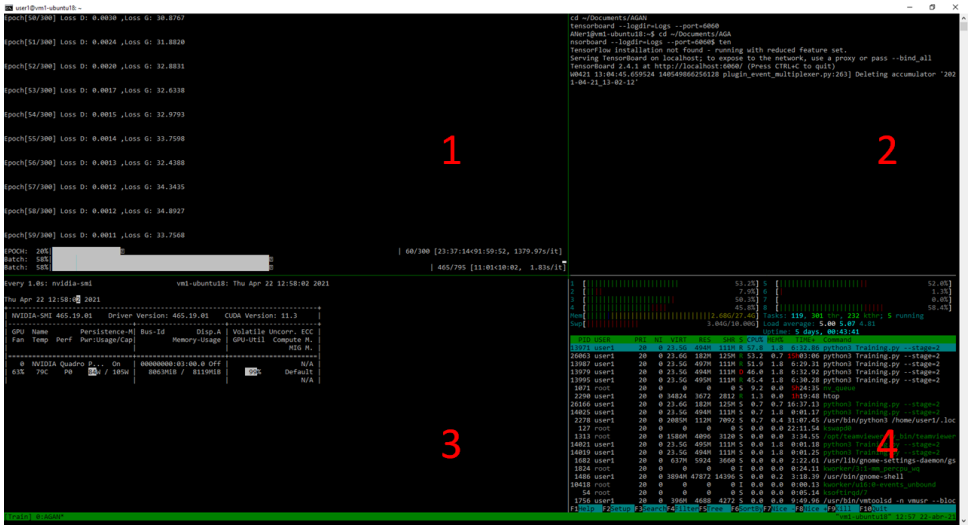


Figura A.3: Captura de los paneles en tmux, la numeración coincide con la descripción anterior.

A.3. SSH optimizado

Si bien estas configuraciones no son necesarias para poder trabajar remotamente, son configuraciones que permiten hacer la conexión con el servidor de manera más eficiente y ágil.

A.3.1. Archivo config

En ∼/.ssh se crea o modifica el archivo *config* y se capturan los datos de la conexión , como es el caso de la dirección IP, puerto y usuario, a esta conexión se le asigna un alias (Host), con lo cual en lugar de introducir todos los datos, solo es necesario capturar el alias.

```
Host unam
HostName 127.0.0.1
Port 8000
User user
```

Figura A.4: Ejemplo de archivo *config*.

```
$ssh user@127.0.0.1 -p 8000    ->    ssh unam
```

A.3.2. Conexión sin contraseña

El protocolo SSH otorga la posibilidad de realizar la autenticación sin la necesidad de introducir una contraseña, para esto utiliza un sistema de ingreso por medio de una llave pública y una llave privada.

Para poder realizar esta autenticación es necesario generar las llaves en la máquina local

```
$ssh-keygen -b 4096 -t rsa
```

Una vez generadas las llaves es necesario pasarlas al servidor para que este las de alta y reconozca al equipo local como una conexión segura siempre y cuando se tengan las llaves como credenciales.

```
$ssh-copy-id user@unam
```

Al introducir este comando el servidor requiere que se capture la contraseña, ya que para dar de alta la llave es necesario verificar que efectivamente el usuario tenga permisos para ingresar.

En el servidor el archivo */.ssh/authorized_keys* se ha modificado y ahora tiene la llave cargada.

```
$sudo service ssh restart
```

Finalmente se puede realizar la conexión sin necesidad de introducir una contraseña, siempre y cuando la conexión se realice desde el equipo local desde donde se realizó el proceso y tenga la llave que se empleó en este.

Bibliografía

[1] L. Zhou, Z. Zhou, and D. Hu, "Scene classification using a multi-resolution bag-of-features model," *Pattern Recognition*, vol. 46, no. 1, pp. 424–433, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2012.07.017

[2] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep learning features at scale for visual place recognition," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, pp. 3223–3230.

[3] H. Xu, L. Yu, J. Hou, and S. Fei, "Automatic reconstruction method for large scene based on multi-site point cloud stitching," *Measurement: Journal of the International Measurement Confederation*, vol. 131, pp. 590–596, 2019. [Online]. Available: https://doi.org/10.1016/j.measurement.2018.09.022

[4] G. Li, L. Yu, and S. Fei, "A deep-learning real-time visual SLAM system based on multi-task feature extraction network and self-supervised feature points," *Measurement: Journal of the International Measurement Confederation*, vol. 168, no. August 2020, p. 108403, 2021. [Online]. Available: https://doi.org/10.1016/j.measurement.2020.108403

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[6] R. Gonzalez, *Digital image processing*. Upper Saddle River, N.J: Prentice Hall, 2002.

[7] R. A. Española, 2019. [Online]. Available: https://www.rae.es

[8] J. Rao and J. Zhang, "Cut and paste: Generate artificial labels for object detection," *ACM International Conference Proceeding Series*, pp. 29–33, 2017.

[9] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information (Switzerland)*, vol. 11, no. 2, pp. 1–20, 2020.

[10] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[11] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 3, no. January, pp. 2672–2680, 2014.

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2018.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[16] R. Rojas, *Neural networks : a systematic introduction*. Berlin New York: Springer-Verlag, 1996.

[17] "Notas de Reconocimiento de Patrones – BioRobotics," Oct 2020, [Online; accessed 8. Oct. 2020]. [Online]. Available: https://biorobotics.fi-p.unam.mx/reconocimiento-de-patrones

[18] R. Szeliski, *Computer Vision (Texts in Computer Science)*, 2010, vol. 42. [Online]. Available: www.springer.com/series/3191{%}5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/20549881

[19] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[20] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[21] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5967–5976, 2017.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[23] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," 2018.

[24] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," 2017.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," 2016.

[26] A. Borji, "Pros and cons of gan evaluation measures," 2018.

[27] ——, "Pros and cons of gan evaluation measures: New developments," 2021.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[30] S. Barratt and R. Sharma, "A note on the inception score," 2018.

[31] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," 2017.

[32] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[33] S. S. Skiena, *The Algorithm Design Manual*. London: Springer, 2008. [Online]. Available: https://www.bibsonomy.org/bibtex/29b2f5050241fea63ff5f49cc29b5bebf/ytyoun

[34] "PyTorch," Apr 2021, [Online; accessed 3. Apr. 2021]. [Online]. Available: https://pytorch.org/hub/pytorch_vision_inception_v3

[35] G. Marzano and A. Novembre, "Machines that dream: A new challenge in behavioral-basic robotics," *Procedia Computer Science*, vol. 104, pp. 146–151, 2017, iCTE 2016, Riga Technical University, Latvia. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091730090X

[36] "BioRobotics, FI, UNAM," Oct 2020, [Online; accessed 31. Oct. 2020]. [Online]. Available: https://biorobotics.fi-p.unam.mx/

[37] "ssh(1) - OpenBSD manual pages," Apr 2021, [Online; accessed 22. Apr. 2021]. [Online]. Available: https://man.openbsd.org/ssh

[38] "Midnight Commander," Apr 2021, [Online; accessed 22. Apr. 2021]. [Online]. Available: https://midnight-commander.org