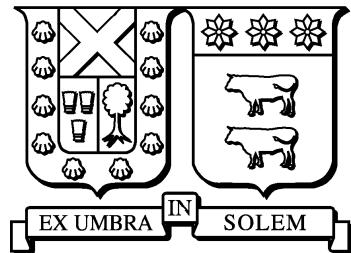


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



LENSGEN: GENERADOR DE IMÁGENES
ARTIFICIALES DE LENTES
GRAVITACIONALES VIA GANS

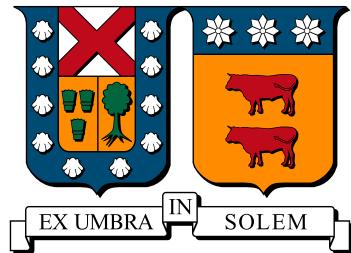
FELIPE NICOLÁS VEGA VALENCIA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: RICARDO ÑANCULEF

MARZO 2021

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



**LENSGEN: GENERADOR DE IMÁGENES
ARTIFICIALES DE LENTES
GRAVITACIONALES VIA GANS**

FELIPE NICOLÁS VEGA VALENCIA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: RICARDO ÑANCULEF

PROFESOR CORREFERENTE: JOSÉ LUIS MARTÍ LARA

PROFESOR CORREFERENTE EXTERNO: UMBERTO RESCIGNO

MARZO 2021

Agradecimientos

En primer lugar, quiero agradecer a mi familia por todo su apoyo durante estos años de estudio en la universidad. Especialmente a la Gorda (mamá), ya que permitió que estudiara sin tener que endeudarme, para terminar pagando un sobreprecio abusivo tan propio del sistema educativo chileno, el cual espero que cambie con el “Nuevo Chile”. También destaco al Julio y al Lauta, quienes me ayudaron con el glosario de esta memoria.

También agradezco a mis amigos de la universidad. Recuerdo con mucho cariño las juntas, salidas de trekking, ventanas en el LDS, e incluso las jornadas de estudio. Mención especial al Pablo por mandarme memes del proceso de memoria por Instagram.

Si bien están incluidos en el grupo anterior, me gustaría destacar a los miembros de *Frogware*, el equipo con quienes tuve la suerte de cursar la Feria de Software, ya que hicieron que el ramo más estresante de toda mi vida universitaria fuera un viaje más agradable y entretenido. Tengo grandes recuerdos de las noches de trabajo, comida, y videos sin sentido en la casa del Nacho.

Por último, quiero agradecer al profesor Ricardo Ñanculef por ayudarme con mi idea de realizar mi memoria en el mundo de la generación de imágenes, como también por las agradables conversaciones en las reuniones por Zoom. Agradezco también a Umberto Rescigno, quien fue de vital ayuda para la realización de los experimentos de esta memoria.

*Para mi hermana Carla.
Se la ganó en el “Cachipún”.*

Resumen

Los lentes gravitacionales representan una de las herramientas más potentes de la astrofísica para comprender la materia oscura. Actualmente se han encontrado en un número limitado, pero se espera que próximos catálogos aumente este número en varios órdenes de magnitud.

En este trabajo se propone la construcción de un generador de imágenes de lentes gravitacionales fuertes mediante la utilización de redes GAN (Red Generativa Antagónica), diseñando experimentos cuantitativos y cualitativos para su validación.

Los resultados obtenidos muestran que la GAN entrenada logra generar imágenes de lentes con la suficiente calidad para “engaños” a un astrónomo experto y una red detectora especializada, contando con un entrenamiento estable y sin la presencia de overfitting. Sin embargo, también se muestra que la red no aprende a generar imágenes de lentes muy inusuales que están presentes, en baja cantidad, en el conjunto de entrenamiento.

Keywords: Lentes gravitacionales, Generative Adversarial Networks.

Abstract

Gravitational lenses represent one of the most powerful tools for understanding the dark matter. Currently they have been found in a limited number, but future surveys are expected to increase this number by several orders of magnitude.

In this work its proposed the construction of a gravitational lens image generator, by using GAN networks (Generative Adversarial Networks), designing quantitative and qualitative experiments for its validation.

The results show that the trained GAN manages to generate lens images with sufficient quality to “mislead” an expert astronomer and a specialized detector network, counting on a stable training without overfitting evidence. However, results also shows that the GAN doesnt learn to generate unusual lens images that are present, in low quantity, in the training set.

Keywords: Gravitational lenses, Generative Adversarial Networks.

Tabla de Contenidos

Agradecimientos	III
Resumen	V
Abstract	VI
Tabla de Contenidos	VII
Lista de Tablas	XI
Lista de Figuras	XII
Glosario	XVI
Introducción	1
1. Definición del Problema	3
1.1. Objetivos	6
1.1.1. Objetivo General	6
1.1.2. Objetivos Específicos	6

2. Antecedentes y Trabajos Relacionados	7
2.1. Antecedentes	7
2.1.1. Aprendizaje Automático	7
2.1.2. Redes Neuronales Artificiales	8
2.1.3. Redes Neuronales Convolucionales	14
2.1.4. Generative Adversarial Networks (GAN)	20
2.2. Trabajos Relacionados	31
2.2.1. Generación de Imágenes de Lentes Gravitacionales	31
2.2.2. Ganadores Challenge Detección de Lentes	34
2.2.3. GAN en Astronomía	37
3. Propuesta de Solución y Validación	40
3.1. Propuesta de Solución	40
3.1.1. Definición del entrenamiento	40
3.1.2. Arquitectura de la red	41
3.1.3. Propuesta	42
3.2. Propuesta de Validación	43
3.2.1. Primer experimento	44
3.2.2. Segundo experimento	46
4. Resultados y Análisis	48
4.1. Definición del entrenamiento	49
4.1.1. Arquitectura base	49

4.1.2.	<i>One-side label smoothing</i>	51
4.1.3.	<i>Noisy labels</i>	53
4.1.4.	<i>Cambio en la inicialización de pesos</i>	55
4.1.5.	Resumen	57
4.2.	Definición de la arquitectura	58
4.2.1.	Cambios en el número de filtros	59
4.2.2.	Disminución de un bloque en el generador	61
4.2.3.	Cambios en el número de capas convolucionales por bloques	63
4.2.4.	Resumen	65
4.3.	Validación	66
4.3.1.	Overfitting	66
4.3.2.	Etiquetado manual	69
4.3.3.	Etiquetado automático	72
4.3.4.	FID modificado	73
4.3.5.	Subclasificación de lentes	75
4.3.6.	Descripción cualitativa de L-NL y NL	78
4.3.7.	Resumen	80
4.4.	Entrenamiento estable	81
4.5.	Diseño para la evaluación de resultados	82
4.6.	Evaluación de resultados	83
4.7.	Trabajo futuro	85
Referencias		86

Índice de Términos	91
5. Primer Apéndice	91
5.1. Arquitecturas	91
5.1.1. Arquitectura base	91
5.1.2. Arquitectura experimentos	92
6. Segundo Apéndice	94
6.1. Resultados	94
6.1.1. Overfitting	94

Lista de Tablas

4.1.	Tabla resumen de los resultados para los experimentos de definición de las estrategias de entrenamiento.	58
4.2.	Tabla resumen de los resultados para los experimentos de definición de la arquitectura de la red.	66
4.3.	Anotaciones entregadas por el experto en el experimento de validación.	69
4.4.	Matriz de confusión del proceso de etiquetado realizado por el experto.	72
4.5.	Anotaciones entregadas por la red <i>Lensfinder</i>	72
4.6.	Proporción de lentes por cada subclase, para los datos previamente etiquetados como L.	75

Lista de Figuras

2.1.	Diagrama de una neurona de una ANN.	10
2.2.	Diagrama de una red neuronal Feed Forward.	11
2.3.	Operación de convolución en capas convolucionales	16
2.4.	Ejemplo de <i>Max Pooling</i>	18
2.5.	Primeros dos pasos de una operación de convolución transpuesta.	19
2.6.	Diagrama de una CNN con 3 bloques convolucionales.	19
2.7.	Esquema del funcionamiento de una GAN.	21
2.8.	Dormitorios generados en una iteración [Radford <i>et al.</i> , 2015].	28
2.9.	Interpolación del espacio latente de la DCGAN [Radford <i>et al.</i> , 2015]. .	28
2.10.	Diagrama de la red Lensfinder.	35
2.11.	Bloques ResNet utilizado por CMU DeepLens.	36
3.1.	Arquitectura propuesta tanto para el generador como para el discriminador de la DCGAN implementada.	43
4.1.	Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje utilizando la arquitectura base.	49
4.2.	Imágenes generadas con la arquitectura base para cada una de las tasas de aprendizaje utilizadas.	50

4.3. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje utilizando <i>One-side label smoothing</i> sobre la arquitectura base.	51
4.4. Comparación del comportamiento del entrenamiento entre la arquitectura base y el uso de <i>one-side label smoothing</i>	52
4.5. Imágenes generadas al utilizar <i>one-side label smoothing</i> sobre la arquitectura base para cada una de las tasas de aprendizaje utilizadas.	53
4.6. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al incluir <i>noisy labels</i>	54
4.7. Comparación del comportamiento del entrenamiento al agregar el uso de <i>noisy labels</i>	54
4.8. Imágenes generadas al utilizar <i>noisy labels</i> sobre la arquitectura base para cada una de las tasas de aprendizaje utilizadas.	55
4.9. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al obtener los pesos iniciales de la red a partir de una distribución normal con media 0 y varianza 0.2.	56
4.10. Comparación del comportamiento del entrenamiento cambiar el inicializador de pesos.	56
4.11. Imágenes generadas al utilizar una distribución normal con media 0 y varianza 0.2 para inicializar los pesos de la red GAN.	57
4.12. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al cambiar el número de filtros de la arquitectura base.	59
4.13. Comparación del comportamiento del entrenamiento al cambiar la cantidad de filtros de la arquitectura base.	60
4.14. Imágenes generadas al cambiar el número de filtros de la arquitectura base.	61
4.15. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al eliminar un bloque convolucional en el generador.	62

4.16. Comparación del comportamiento del entrenamiento al eliminar un bloque convolucional en el generador.	62
4.17. Imágenes generadas al cambiar eliminar un bloque convolucional del generador.	63
4.18. Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al agregar una capa convolucional en los bloques del generador.	64
4.19. Comparación del comportamiento del entrenamiento al agregar una capa en los bloques del generador.	64
4.20. Imágenes generadas al cambiar eliminar un bloque convolucional del generador.	65
4.21. Interpolación entre 10 puntos aleatorios obtenidos aleatoriamente desde el espacio latente.	67
4.22. Imágenes generadas aleatorias junto a las imágenes más cercanas, según activaciones de la red Lensfinder, del conjunto de entrenamiento.	68
4.23. Imágenes de lentes gravitacionales donde la etiqueta del <i>challenge</i> difiere con la del astrónomo.	70
4.24. Imágenes sin lentes gravitacionales donde la etiqueta del <i>challenge</i> difiere con la del astrónomo.	71
4.25. Histograma de las predicciones de la red Lensfinder para los 3 conjuntos de datos con lentes del experimento de validación.	73
4.26. Imágenes generadas vía GAN para distintos epochs.	74
4.27. Proporción de lentes por cada subclase, para los datos previamente etiquetados como L.	76
4.28. 3 interpolaciones entre imágenes con un anillo.	76
4.29. Imágenes de lentes etiquetadas como QL, en conjunto a las primeras 6 imágenes NL etiquetadas de la misma manera.	77

4.30. Imágenes clasificadas como L-NL con características muy inusuales, provenientes del conjunto de lentes originales.	78
4.31. Imágenes generadas por GAN clasificadas como NL, donde la resolución no permite su detección como lente.	79
4.32. Imágenes del conjunto de lentes original clasificadas como NL, donde la resolución no permite su detección como lente.	79
5.1. Arquitectura utilizada como base para los experimentos. Diseñada para ser utilizada sobre Cifar-10 [Brownlee, 2019].	91
5.2. Arquitectura utilizada en el primer experimento de definición de arquitectura.	92
5.3. Arquitectura utilizada en el segundo experimento de definición de arquitectura.	92
5.4. Arquitectura utilizada en el tercer experimento de definición de arquitectura. Corresponde a la arquitectura seleccionada.	93
6.1. Imágenes generadas por la GAN 1, en conjunto con las imágenes más cercanas del conjunto de entrenamiento.	94
6.2. Imágenes generadas por la GAN 2, en conjunto con las imágenes más cercanas del conjunto de entrenamiento.	95

Glosario

ANN

Artificial Neural Network.

AUROC

Área bajo la curva ROC. Métrica utilizada para evaluar el desempeño de un clasificador.

Bit-flip

Operación donde se invierte el valor de un bit.

Categoría *Ground Based*

Categoría del *Gravitational Lens Finding Challenge*. Consiste en la clasificación de imágenes de 4 canales, basadas en el catálogo KiDs.

Categoría *Space Based*

Categoría del *Gravitational Lens Finding Challenge*. Consiste en la clasificación de imágenes de un canal, basadas en el telescopio Euclid.

CNN

Convolutional Neural Network.

DCGAN

Deep Convolutional Generative Adversarial Network. Adaptación del framework GAN para arquitecturas convolucionales.

Dropout

Técnica utilizada en redes neuronales donde se “apagan” neuronas durante el entrenamiento, con el objetivo de evitar el *overfitting*.

Epoch

Paso de todo el conjunto de entrenamiento por un algoritmo de redes neuronales.

Espacio latente

Espacio multidimensional que contiene características que codifican una representación interna significativa de un objeto.

Flatten

Operación que colapsa un arreglo multidimensional en una sola dimensión.

Forward pass

Paso de datos por las capas de una red neuronal, con el objetivo de calcular el *output*.

GAN

Generative Adversarial Network.

Gradiente descendente

Algoritmo de optimización iterativo, donde en cada paso se avanza en la dirección opuesta al gradiente de la función que se desea optimizar.

Gravitational Lens Finding Challenge

Competencia de detección de lentes gravitacionales fuertes en imágenes.

Ground truth

Valor medido para la variable objetivo en los datos utilizados para el entrenamiento y validación de un método de aprendizaje automático.

Hubble Ultra Deep Field

Imagen más profunda de una región del espacio utilizando luz visible.

Keras

Framework de Deep Learning sobre Python.

Learning Rate

Tasa de aprendizaje. Controla los pasos de actualización en los algoritmos de optimización vía gradiente.

Lensfinder

Red detectora de lentes gravitacionales fuertes, ganadora de la categoría *Space Based* del *Gravitational Lens Finding Challenge*.

Lente gravitacional

Fenómeno en la cual la luz de un fuente distante se curva en la proximidad de un objeto masivo situado entre fuente y observador.

Lente gravitacional fuerte

Tipo de lente gravitacional donde se producen distorsiones fácilmente visibles en el camino de la luz de la fuente.

Materia oscura

Tipo de materia que no emite ningún tipo de radiación electromagnética.

Modelo de concordancia cosmológica

Modelo Λ CDM, teoría astrofísica que describe la estructura y evolución del universo.

Perfil NFW

Perfil Navarro–Frenk–White. Distribución espacial de masa oscura de un halo identificado a partir de simulaciones de N-cuerpos.

Red Inception

Red convolucional que utiliza filtros de distintos tamaños a la misma profundidad, permitiendo que la red utilice el mejor tamaño en lugar de definirlo por defecto.

Relatividad general

Teoría de campos gravitatorios y sistemas de referencias desarrollada por Albert Einstein.

Simulación de N-cuerpos

Simulación de un sistema dinámico de partículas bajo la influencia de fuerzas físicas.

Tensorflow

Framework de aprendizaje automático.

Introducción

El fenómeno de lentes gravitacionales tiene grandes aplicaciones en la astrofísica, siendo una de éstas su uso como telescopios naturales. Por ejemplo, mediante la utilización de un telescopio del observatorio ALMA¹ se detectó un lente que permitió a un grupo de investigadores descubrir una galaxia ubicada a billones de años luz de la Vía Láctea, la que presenta características similares a nuestra galaxia [Rizzo *et al.*, 2020].

La cantidad de lentes detectados actualmente es limitada, sin embargo, con la construcción de nuevos telescopios se espera que aumente su número de gran manera, por lo que serán inviables de detectar mediante inspección humana, generando así la necesidad de métodos automáticos para su detección. Al tener un número limitado de lentes gravitacionales, se han diseñado *pipelines* de simulación[Metcalf *et al.*, 2018][Li *et al.*, 2016] para la generación de imágenes que permitan calibrar estos métodos. Sin embargo, la generación de imágenes mediante estas simulaciones es muy costosa, ya que requiere el uso de súper computadoras.

En este trabajo se propone la utilización de *Generative Adversarial Networks* para la generación de imágenes sintéticas, mediante su entrenamiento con imágenes ya generadas en simulaciones, para así disminuir el procesamiento necesario para la generación de nuevos lentes.

El documento se estructura de la siguiente manera: en el capítulo 1 se entrega una definición más profunda del problema y se detallan los objetivos de esta memoria. En el capítulo dos se entregan los antecedentes teóricos para la compresión del trabajo realizado, una revisión por los trabajos relacionados a la generación de lentes gravitacionales, y el

¹<https://www.almaobservatory.org/es/comunicados-de-prensa/alma-observa-la-galaxia-mas-distante-parecida-a-la-via-lactea/>

uso de GANS en el ámbito astronómico. A continuación, en el capítulo 3 se presenta la propuesta de solución y validación, detallando los experimentos realizados y los mecanismos de validación de éstos, para presentar los resultados y el análisis en el capítulo 4. Finalmente, en el capítulo 5 se documentan las conclusiones y el trabajo futuro.

Capítulo 1

Definición del Problema

Se conoce como *lente gravitacional* al fenómeno en el cual la radiación electromagnética de una fuente luminosa es desviada en la proximidad de una distribución de masa ubicada entre la fuente y un observador. En una interpretación relativista, este efecto se debe a una distorsión espacio-temporal alrededor de un objeto masivo. Curiosamente, un lente gravitacional actúa sobre los caminos de luz al igual que un lente común lo hace en óptica geométrica: puede dividir la fuente en múltiples imágenes, distorsionadas y magnificadas. Por ejemplo, cuando el objeto fuente es una galaxia y el lente también lo es, se forma un anillo parcial o completo alrededor o a través de la galaxia que actúa como lente.

Este fenómeno representa una de las aplicaciones más prometedoras de la *Relatividad General* en astrofísica y es una línea de investigación bastante emocionante. De hecho, con su amplia fenomenología, esta disciplina es una herramienta multipropósito única: entre otros, los lentes pueden rastrear y cuantificar materia oscura, pueden proveer pistas sobre su naturaleza mediante diferentes pruebas, y pueden ser utilizadas como telescopios naturales para estudiar las fuentes de fondo magnificadas. En particular, los lentes gravitacionales permiten probar el modelo de concordancia cosmológica actual mediante el modelamiento de galaxias que actúan como lentes [Grillo *et al.*, 2016]. Esto ayudaría en la caracterización de la materia oscura y permitiría (al menos) aliviar cierto número de discrepancias entre las predicciones del paradigma estándar de la cosmología y las observaciones de escala galáctica [Bullock y Boylan-Kolchin, 2017]. Sin embargo, en la era actual de la *Precisión Cosmológica*, para que estas técnicas sean efectivas,

deben ser aplicadas a la mayor cantidad de muestras que se puedan conseguir, y sus resultados deben ser lo más precisos y exactos posibles.

Considerando el contexto anterior, las galaxias son objetos bastante adecuados para el estudio de lentes, ya que estos sistemas astrofísicos pueden mostrar los típicos efectos del lente gravitacional fuerte, como la presencia de múltiples imágenes producidas desde una misma fuente. La identificación de estas imágenes tiene un rol clave en el análisis de una única galaxia: mientras más sean estas imágenes, más son las restricciones en el perfil de masa utilizadas para modelar el sistema galáctico, como la luminosidad de la galaxia más su halo de materia oscura circundante.

Por otro lado, la alta precisión del modelo alcanzable al explotar todo el conjunto de imágenes de un solo sistema de galaxias no es el único prerrequisito para realizar pruebas cosmológicas más rigurosas. Es necesario detectar una muestra estadísticamente relevante de múltiples sistemas de imágenes en sistemas galácticos. Si bien hasta el momento se ha encontrado un número muy limitado, se espera que telescopios como el Square Kilometer Array (SKA), Large Synoptic Survey Telescope (LSST) y el Euclid Space Telescope (EST) aumenten el número de lentes gravitacionales en varios órdenes de magnitud [Metcalf *et al.*, 2018], haciendo inviables los métodos de detección que tomen mucho tiempo y/o requieran inspección humana, por lo que se requerirá de métodos automáticos de detección para procesar toda la información contenida en los catálogos que están por venir.

Dada esta necesidad de diseñar métodos automáticos para la detección de lentes gravitacionales, es necesario contar con imágenes para su calibración y comprobar su precisión, por lo que se debe simular este fenómeno de alguna manera. Con este objetivo en mente, en [Metcalf *et al.*, 2018] se diseña un pipeline de simulación de imágenes de lentes gravitacionales fuertes que puedan ser utilizadas para diseñar y calibrar métodos automáticos. Sin embargo, este proceso de simulación está compuesto por una serie de procedimientos complejos que demandan una gran cantidad tiempo e intervención humana, por lo que realizar nuevas simulaciones es muy costoso. Lo anterior cobra vital importancia si se considera que estas simulaciones fueron liberadas en el *Gravitational Lens Finding Challenge* [Metcalf *et al.*, 2018], con el objetivo de motivar a la comunidad a diseñar métodos automáticos de detección de lentes. Los ganadores de las dos categorías existentes fueron métodos de aprendizaje profundo (*deep learning*) [Schaefer *et al.*, 2018][Lanusse *et al.*, 2017], los cuales son conocidos

por requerir de una gran cantidad de casos para su entrenamiento, que como se mencionó anteriormente, no existen en los catálogos actuales.

La complejidad del proceso de simulación motiva el diseño de métodos para obtener nuevas imágenes sintéticas de manera menos costosa, y así apoyar el entrenamiento de detectores basados en *deep learning* y otras técnicas que requieran de calibración. En el área de métodos generativos, una de las técnicas que ha mostrado gran efectividad en la generación de imágenes sintéticas son las redes neuronales GAN (*Generative Adversarial Networks*) [Goodfellow *et al.*, 2014], las cuales en su versión original consideran un generador y un discriminador que compiten entre sí: la red generadora es entrenada de tal manera que logre engañar al discriminador, aprendiendo un mapeo entre el espacio latente y la distribución de probabilidad de los datos, mientras que la red discriminadora es entrenada para determinar si el input que recibe es real o sintético. De esta manera, una vez que el discriminador no es capaz de distinguir entre una imagen real y una imagen sintética, se puede utilizar el generador ya entrenado para obtener nuevas imágenes, las cuales siguen la distribución de probabilidad de los datos reales.

El desempeño obtenido por este tipo de redes en tareas de generación de imágenes sintéticas, tales como logos, caras, paisajes, etc, motiva el uso de su variante convolucional DCGAN (*Deep Convolutional Generative Adversarial Network*)[Radford *et al.*, 2015] para generar nuevas imágenes sintéticas a partir de las simulaciones presentes en el *Gravitational Lens Finding Challenge*, buscando así disminuir el costo asociado a su generación.

Para la evaluación de los resultados obtenidos, lo más común es la realización de experimentos donde expertos, en este caso astrónomos especializados en lentes gravitacionales, determinan la validez de las imágenes generadas por el modelo. Sin embargo, en este proyecto se propone también la utilización de mecanismos cuantitativos para evaluar la calidad de las imágenes sintéticas: ya sea utilizando redes neuronales convolucionales entrenadas para la detección de lentes [Lanusse *et al.*, 2017] para determinar si es posible engañar al clasificador con las imágenes generadas; o mediante la utilización de métricas clásicas para GANS, siendo modificadas para adaptarse al dominio de los lentes gravitacionales.

1.1. Objetivos

1.1.1. Objetivo General

- Construir un generador de imágenes artificiales de fenómenos de lensing gravitacional fuerte utilizando redes neuronales GAN.

1.1.2. Objetivos Específicos

- Determinar la arquitectura para el generador y el discriminador de una red neuronal GAN, de tal manera que el entrenamiento de la red sea estable.
- Diseñar un mecanismo cuantitativo para la evaluación de las imágenes generadas por la red.
- Evaluar y documentar los resultados obtenidos considerando el mecanismo cuantitativo desarrollado y validación por parte de astrónomos expertos en lensing gravitacional.

Capítulo 2

Antecedentes y Trabajos Relacionados

En este capítulo se revisan conceptos teóricos de las redes neuronales GAN, considerando aspectos de las redes neuronales clásicas, redes neuronales convolucionales y *Generative Adversarial Network* (Red Generativa Antagónica). También se realiza una revisión de los métodos existentes para la generación de imágenes sintéticas de lentes gravitacionales fuertes y clasificadores implementados sobre estas imágenes en el marco de un *challenge* de detección. Finalmente, se termina con una revisión del uso de GANS en otros contextos del ámbito astronómico.

2.1. Antecedentes

2.1.1. Aprendizaje Automático

Se conoce como Aprendizaje Automático al conjunto de teorías, modelos y algoritmos que permiten que una máquina logre resolver una determinada tarea sin tener que programar el algoritmo específico que la resuelve. Para esto, lo que se busca es entrenar una máquina, es decir, que a partir de un conjunto de ejemplos de la tarea a resolver, la máquina “aprenda” a reproducir el comportamiento asociado a dicho conjunto. La idea detrás de esto es que la máquina logre completar la tarea deseada en presencia de nuevos datos.

De manera más formal, en [Mitchell, 1997] se define el Aprendizaje Automático de la siguiente manera: “Se dice que un programa aprende de una experiencia **E** con respecto a una tarea **T** y medida de desempeño **P** si el desempeño en la tarea **T** medida por **P** mejora con la experiencia **E**”. A modo de ejemplo, si considera el detector de spam de algún gestor de correos, **T** sería “Determinar si un correo es spam o no”, **E** corresponde a un conjunto de correos etiquetados de manera previa, y **P** podría ser la cantidad de correos etiquetados de manera correcta.

Cuando se habla de entrenar una máquina, lo que se busca es encontrar los valores de los parámetros de un determinado modelo. Para esto se separa el conjunto de datos en dos conjuntos; entrenamiento y pruebas. El conjunto de entrenamiento es el que se utiliza para ajustar los parámetros del modelo y el conjunto de pruebas se utiliza para medir el desempeño de la máquina frente a datos nuevos. Cuando la máquina logra mantener la performance alcanzada durante el entrenamiento en el conjunto de prueba se dice que la máquina logra generalizar la tarea, en caso contrario, la máquina cae en *overfitting*, es decir, se sobre-ajusta a los datos de entrenamiento, memorizando el ruido asociado a éstos en lugar de aprender, lo que perjudica la performance de la máquina frente a datos nuevos.

Para lograr entrenar una máquina, es necesario contar con un conjunto datos suficientemente grande, por lo que la gran cantidad de datos generados actualmente, sumado a los avances en hardware, han permitido que el aprendizaje automático cada vez sea más utilizado, no solo en el ámbito académico, sino que también en aplicaciones en la industria.

2.1.2. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales, ANN por su sigla en inglés, son un método de aprendizaje automático inspirado en el comportamiento del cerebro humano, en particular, la manera en la que una neurona transmite los impulsos eléctricos a las neuronas a la que está conectada. Considerando dicha idea, las ANN son una estructura de grafo donde los nodos, denominados neuronas, ejecutan una regla de computación utilizando la salida de las neuronas que se conectan a ella.

Neurona

La neurona es la unidad básica de una ANN, la cual se encarga de realizar un determinado cómputo. Para realizar dicho cómputo, la neurona cuenta con un vector de pesos W , el que se combina linealmente con un vector de entrada X , el cual proviene de las neuronas conectadas a ella, y finalmente aplicando una función de activación. También se considera un escalar de sesgo b , sumado de manera previa a la función de activación. En la ecuación 2.1 se muestra el computo realizado por una neurona.

$$f(x) = \sigma \left(\sum_i^n w_i \cdot x_i + b \right) \quad (2.1)$$

- w_i : i-ésimo elemento del vector de pesos.
- x_i : i-ésimo elemento del vector de entrada.
- b : escalar de sesgo.
- σ : función de activación.

La función de activación es, generalmente, una función no lineal que define la salida de cada neurona a partir de la entrada que esta recibe. Dentro de las funciones de activación más utilizadas se encuentran las siguientes:

- **Sigmoidal**: también conocida como función Logística, acota el valor de activación entre 0 y 1. Si el resultado de la combinación lineal es menor a 0, esta función mantiene la activación en 0, mientras que si es positiva ajusta el valor rápidamente a 1. Considerando su rango de valores, suele ser utilizada para representar probabilidades en clasificación binaria.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

- **Tangente Hiperbólica**: función de forma similar a la función sigmoidal, pero considerando un rango de valores entre -1 y 1. Suele tener mejor desempeño que la función anterior.

$$\sigma(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.3)$$

- **ReLU:** o Rectificadora, es una función altamente utilizada en redes utilizadas sobre imágenes. Mantiene como 0 todos los valores de entrada negativos, comportándose de manera lineal para valores positivos. Al no estar acotada por la derecha permite acelerar el entrenamiento de la red.

$$\sigma(x) = \max(0, x) \quad (2.4)$$

- **Leaky ReLU:** modificación de la función ReLU, la cual permite valores negativos para evitar que una neurona no presente activación. Considera un parámetro α que controla el comportamiento para valores de entrada menores a 0.

$$\sigma(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases} \quad (2.5)$$

En la figura 2.1 se observa el diagrama de una neurona de una ANN.

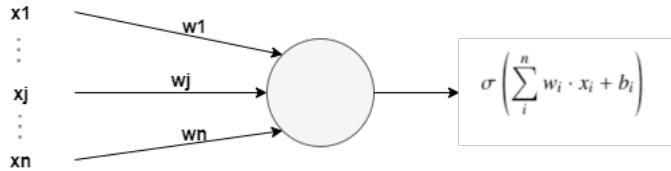


Figura 2.1: Diagrama de una neurona de una ANN.

Red Neuronal Feed Forward

Considerando la neurona descrita anteriormente es posible definir una Red Neuronal *Feed Forward*, también conocida como Perceptrón Multicapa (MLP), la cual agrupa dichas neuronas en capas, concatenando estas para lograr la salida deseada. Las neuronas entre capas consecutivas están completamente conectadas, es decir, una neurona de una capa recibe como entrada la salida de todas las neuronas de la capa anterior.

Este tipo de red considera tres tipos de capas:

- **Capa de entrada:** primera capa de una red *Feed Forward*, se especializa en la lectura de los datos de entrada.
- **Capa de salida:** última capa de una red *Feed Forward*, se especializa en la tarea a realizar y considera neuronas con una función de activación acorde a dicha tarea. Por ejemplo, en clasificación binaria la capa de salida consideraría una neurona con activación sigmoidal.
- **Capas ocultas:** capas intermedias entre la capa de entrada y la capa de salida. Se encargan de procesar los datos recibidos por la red.

La cantidad de capas ocultas de una red *Feed Forward* define la profundidad de la red. En un comienzo se utilizaban redes con una sola capa oculta, pero conforme avanzó la investigación en el área, sumado a las mejoras de hardware, se comenzó a diseñar redes mucho más profundas. En la figura 2.2 se observa el diagrama de una red *Feed Forward* con dos capas ocultas.

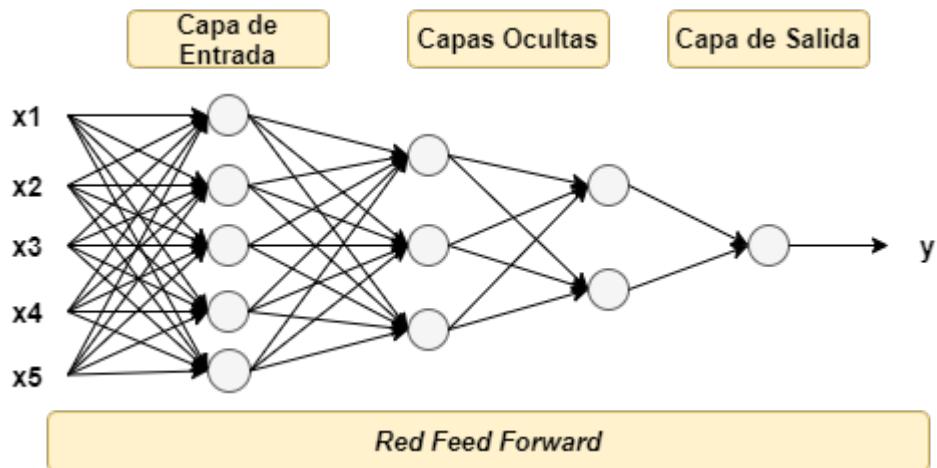


Figura 2.2: Diagrama de una red neuronal Feed Forward.

Matemáticamente, la notación utilizada para el cómputo realizado por una red neuronal de L capas se define de la siguiente manera:

- $W^{(l)}$: matriz de parámetros de la l-ésima capa, compuesta por los vectores de parámetros de cada una de sus neuronas.

- $b^{(l)}$: vector de parámetros de sesgo de la l-ésima capa.
- σ_l : función de activación de la l-ésima capa.
- $a^{(l)}$: activación (output) de la l-ésima capa, se calcula a partir de la matriz de parámetros de la capa y la activación de la capa anterior.

$$a^{(l)} = \sigma_l \left(W^{(l)} a^{(l-1)} + b^{(l)} \right) \quad (2.6)$$

- \hat{y} : salida de la red, la forma depende de la tarea a resolver.

$$\hat{y} = f(x, W) = a^{(L)} = \sigma_L \left(W^{(L)} a^{(L-1)} + b^{(L)} \right) \quad (2.7)$$

Entrenamiento

Como se mencionó en la sección de Aprendizaje Automático 2.1.1, en el entrenamiento de una ANN se busca ajustar los valores de los parámetros de sus capas, de tal manera que se mejore del desempeño con respecto a la tarea deseada. Este proceso se realiza a partir de un conjunto de N ejemplos de entrenamiento $\{(x^n, y^n)\}_{n=1}^N$. Considerando lo anterior, es necesario tener un mecanismo que indique que tan buena es la máquina para reproducir el comportamiento que se busca. Esto se logra utilizando una función de pérdida (*Loss function*), la que indica el error de la red al resolver la tarea deseada. Por ejemplo, en clasificación binaria se suele utilizar la función *binary crossentropy* (equación 2.8), la cual mide que tan buena es una clasificación entregada como una probabilidad.

$$L(y, \hat{y}) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})) \quad (2.8)$$

Una vez definida la función de costo, el criterio de optimización para los parámetros de la red es el siguiente:

$$\min_W \frac{1}{N} J(W) = \frac{1}{N} \cdot \sum_{n=1}^N L(\hat{y}^{(n)}, y^{(n)}) = \frac{1}{N} \sum_{n=1}^N L(f(x^{(n)}, W), y^{(n)}) \quad (2.9)$$

El algoritmo más utilizado para optimizar el criterio anterior es ***Backpropagation*** [Rumelhart *et al.*, 1986], el cual resolvió uno de los principales problemas en el entrenamiento de redes neuronales: el valor de la activación de una capa depende de los parámetros de todas las capas anteriores, generando una gran dependencia. *Backpropagation* resuelve el dilema anterior aplicando la regla de la cadena, de tal manera es posible calcular el gradiente de la función de costo en todas las capas, partiendo desde la capa de salida hasta la capa de entrada, utilizando para ello programación dinámica.

Calculando los gradientes con *Backpropagation*, es posible optimizar el criterio 2.9 utilizando el algoritmo iterativo Gradiente Descendente:

$$W_t = W_{t-1} - \eta_t \vec{\nabla} J|_{W_{t-1}} \quad (2.10)$$

- W_t : parámetros de la red en la iteración t.
- η_t : tasa de aprendizaje, controla que tanto se avanza en la dirección contraria al gradiente.
- $\vec{\nabla} J|_{W_{t-1}}$: gradiente del criterio de optimización con respecto a los parámetros de la iteración anterior. Se calcula utilizando *Backpropagation*.

Este algoritmo se aplica hasta cumplir algún criterio de término, el cual puede ser un número máximo de iteraciones, alcanzar convergencia, etc.

La tasa de aprendizaje es un hiper parámetro que controla la convergencia del método de gradiente descendente por lo que debe ser sintonizado, ya que si esta es muy grande el entrenamiento podría divergir, mientras que si es muy pequeña el entrenamiento demoraría mucho en alcanzar la convergencia. Considerando lo anterior, existen métodos que se encargan de adaptar esta tasa para garantizar la tasa de convergencia inicial y para escapar de puntos sillas durante el entrenamiento. Uno de los algoritmos con tasa de aprendizaje adaptativa más utilizados es ***Adam*** (por *adaptative moment estimation*) [Kingma y Ba, 2014], el cual realiza una estimación de los dos primeros momentos de los gradientes para el ajuste de pesos. Este funciona bien en la práctica, saliendo favorecido en comparación con otros algoritmos de optimización.

2.1.3. Redes Neuronales Convolucionales

Una Red Neuronal Convolutional, *Convolutional Neural Network* (CNN) en inglés, es una red neuronal especializada en problemas de visión computacional, representando una gran mejora sobre los benchmarks existentes.

Para ilustrar su eficacia es necesario entender los problemas que tiene una red neuronal Feed Forward en este tipo de tareas, por ejemplo, en clasificación sobre imágenes RGB de 250x250 pixeles. Al adaptar una imagen al formato vectorial necesario para la red se obtendría un vector de $250 \cdot 250 \cdot 3 = 187500$ dimensiones, por lo que al considerar una capa oculta de 32 neuronas se obtiene un total de 6 millones de parámetros solo para la primera capa (sin contar el sesgo). En general, en visión computacional se utilizan imágenes mucho más grandes, por lo que las dimensiones de los input suponen un problema para redes Feed Forward. Otro de los problemas que presentan estas redes es que aprenden a detectar patrones solo en un sector de una imagen, por lo que una simple traslación en la imagen afecta de gran manera el desempeño, situación que es muy común en visión por computadora.

Para solucionar los problemas anteriores, una Red Convolutional aplica las siguientes ideas:

- **Conectividad local:** se limitan las conexiones con las que cuenta una neurona con respecto a las neuronas de la siguiente capa. De esta manera, cada neurona tiene un campo específico de acción, lo que disminuye el número de parámetros.
- **Compartimiento de parámetros:** fuerza a que grupos de neuronas tengan los mismos parámetros. Con esto, si una neurona aprende a detectar un patrón, la red será capaz de detectar dicho patrón en toda la imagen. Las neuronas que comparten parámetros se agrupan en una estructura denominada *feature maps*.

Convolución

El efecto provocado por un grupo de neuronas con compartimiento de parámetros es, matemáticamente, equivalente a una operación de convolución. Esta operación es bastante utilizada en el procesamiento de señales, ya que permite obtener la respuesta de

una señal frente a un impulso para todo instante de tiempo.

De manera formal, una convolución discreta para una señal unidimensional frente a un impulso se define de la siguiente manera:

$$s(i) = (x * k)(i) = \sum_{p=-\infty}^{\infty} x(p)k(i-p) \quad (2.11)$$

- $x(i)$: Señal en un tiempo discreto i .
- $k(i)$: Impulso en el tiempo discreto i .
- $s(i)$: Señal resultante en el tiempo discreto i . Se puede interpretar como el resultado de superponer el impulso frente a la señal.

En el caso de una red convolucional, x corresponde al input mientras que k es conocido como **filtro**. Considerando nuevamente el ejemplo de clasificación de imágenes, la aplicación de la operación de convolución se puede interpretar como el efecto de aplicar como filtro el *feature map* formado por neuronas que comparten parámetros. En dicho caso, como la imagen corresponde a una estructura 2D, el filtro a utilizar también tiene dicha forma, siendo representado matemáticamente de la siguiente manera:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (2.12)$$

- I : Input, en este caso una imagen.
- K : Filtro, en este caso un arreglo de parámetros bi-dimensional.
- S : Resultado de la operación.

Capa Convolucional

Las redes convolucionales consideran la aplicación de la operación de convolución en sus capas, dando origen así a la denominada Capa Convolutinal. Este tipo de capa utiliza como filtro un conjunto de *feature maps* de un tamaño definido, aplicando este

sobre el dato de entrada, generando un efecto de ventana móvil que recorre toda la imagen.

La notación utilizada para la operación aplicada para la capa l -ésima de una red convolucional es la siguiente:

$$a^{(l)} = \sigma_l \left(a^{(l-1)} * W^{(l)} + b^{(l)} \right) \quad (2.13)$$

- $W^{(l)}$: filtro compuesto por un arreglo de *feature maps*.
- $a^{(l)}$: activación (output), obtenido a partir de la activación de la capa anterior.
- σ_l : función de activación.
- $b^{(l)}$: parámetros de sesgo.

Un ejemplo de la aplicación de esta capa se ilustra en la figura 2.3, en la cual se aplican dos pasos de la convolución definida por un filtro de 2×2 compuesto de un *feature map* sobre una imagen de 4×4 .

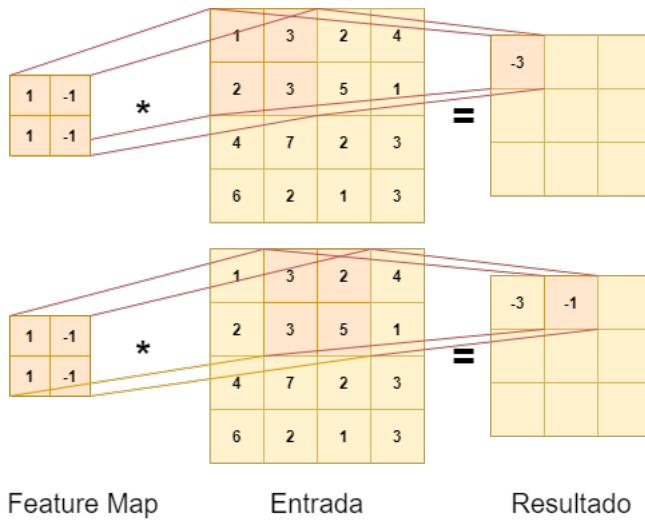


Figura 2.3: Operación de convolución en capas convolucionales

Cada capa considera 4 parámetros para su funcionamiento, los cuales definen el tamaño del output resultante:

- **Número de feature maps (n_c):** cada capa puede aplicar uno o más feature maps distintos.
- **Tamaño de cada feature map:** define el tamaño (n_h, n_w) del campo de acción de cada filtro. Usualmente se utilizan filtros cuadrados, pero es posible considerar filtros rectangulares.
- **Stride (s):** controla que tanto se mueve la ventada definida por el feature map.
- **Padding (p):** agrega elementos en los extremos de la imagen de entrada, lo cual es útil para detectar de mejor manera posibles patrones ubicados en los bordes. El *padding* más utilizado es el *0-padding*, el cual agrega ceros en los extremos.

Considerando una capa con n_c filtros, de tamaño (h, w) , *stride* s y *padding* p , el cual recibe como entrada una imagen de dimensiones (H, W, C) , genera como salida del siguiente tamaño:

$$\left(\left\lfloor \frac{H + 2p - n_h}{s} \right\rfloor, \left\lfloor \frac{W + 2p - n_w}{s} \right\rfloor, n_c \right).$$

Downsampling y Upsampling

Si bien, con la aplicación de conectividad local se reduce el número de parámetros, al aumentar la profundidad de la CNN surge la necesidad de reducir aún más la cantidad de parámetros utilizados, por lo que se debe disminuir la dimensionalidad entre capas.

Una manera de obtener este efecto es aplicar operaciones de ***Pooling***. En este tipo de operación, se define un campo de acción sobre el cual se aplica una operación definida para obtener un valor representante de dicho campo, por ejemplo, en el denominado ***Max Pooling*** se elige como resultado el valor máximo del campo de acción, mientras que con ***Average Pooling*** se toma el promedio de los valores. Al aplicar operaciones definidas, este artefacto no considera parámetros entrenables para su utilización.

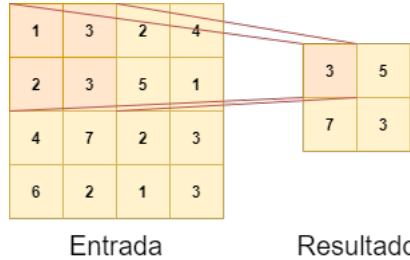


Figura 2.4: Ejemplo de *Max Pooling*.

Otra manera de lograr disminuir la dimensionalidad es utilizar un *stride* mayor a uno, dando origen a lo que se conoce como ***strided convolution***. Como se utiliza una capa convolucional la red entrena los parámetros de ésta, aprendiendo su propio operador de síntesis en lugar de aplicar uno pre-definido.

En ciertas aplicaciones, como segmentación de imágenes y generación de imágenes sintéticas, es necesario contar con algún método para aumentar la dimensionalidad ya que es necesario que la salida de la red cuente con dimensiones predeterminadas. Con este fin, es posible usar algún método de interpolación, siendo una de las más utilizadas es el uso de la interpolación de vecinos más cercanos (Nearest Neighbors). En ésta interpolación se calcula la posición esperada de cada punto de la imagen resultante con respecto a la imagen original, replicando el valor del pixel con la posición a menor distancia de la posición esperada. Otra opción es la interpolación Bi-lineal, donde entre dos pixeles consecutivos se agregan elementos obtenidos desde una escala lineal.

También es posible utilizar Convoluciones Transpuestas (*Transposed Convolution*) para aumentar las dimensiones. Este tipo de operación considera feature maps actuando como ventanas móviles al igual que en una convolución normal, pero cada elemento se considera como un escalar que es multiplicado por el filtro, generando un campo de acción sobre el resultado del tamaño del filtro. Para hacer más drástico el aumento se utiliza el parámetro de *stride*, el cual tiene el mismo efecto sobre la ventana que en una convolución normal. En la figura 2.5 se muestran los primeros dos pasos de una convolución transpuesta sobre una imagen de 2x2, considerando un filtro de tamaño 2x2 y 2 de *stride*, cuyo efecto se ve reflejado en la manera en que se desplaza el campo de acción.

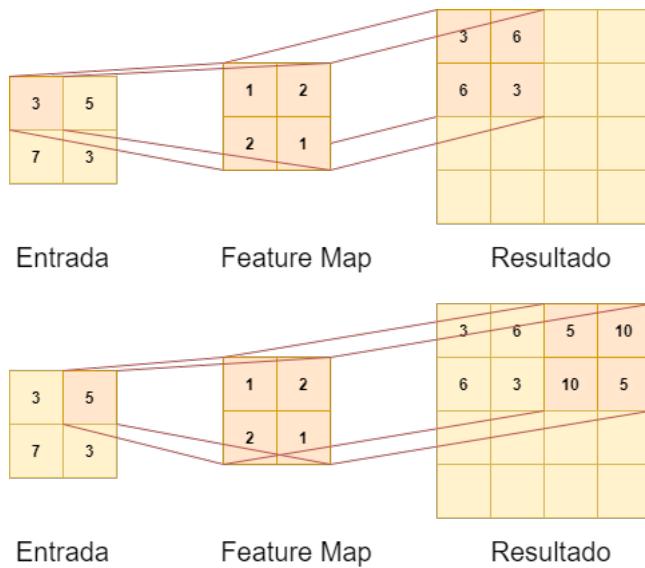


Figura 2.5: Primeros dos pasos de una operación de convolución transpuesta.

Arquitectura Clásica de una CNN

Una red convolucional clásica forma bloques con una o más capas convolucionales seguidas de un método de *downsampling*, generalmente *Max Pooling*. Estos bloques se concatenan, disminuyendo las dimensiones mientras se aumenta el número de canales en cada capa. Finalmente se aplana las características extraídas por los bloques convolucionales en un vector 1D, agregando capas completamente conexas (red Feed Forward) entregando finalmente la salida deseada.

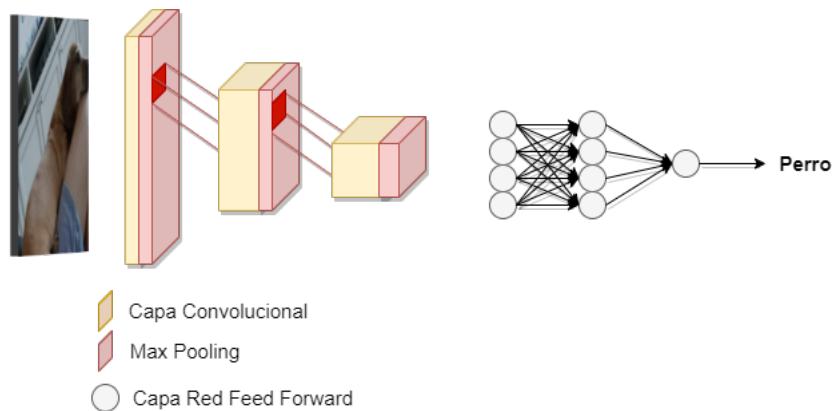


Figura 2.6: Diagrama de una CNN con 3 bloques convolucionales.

Este tipo de redes tiene un gran desempeño en tareas de visión computacional, por ejemplo, la conocida AlexNet [Krizhevsky *et al.*, 2012] logró reducir el error de clasificación sobre *Imagenet* [Deng *et al.*, 2009] de un 0,26 a 0,16 en el challenge del año 2012.

Desde la irrupción de las redes convolucionales cada vez se utilizan redes más profundas y sofisticadas, por ejemplo la red VGG [Simonyan y Zisserman, 2014], que estandariza los hiperparámetros para cada bloque convolucional, la ResNet [He *et al.*, 2015], que incluye conexiones entre capas no consecutivas (*skip-connections*), o la red Inception [Szegedy *et al.*, 2014], que considera caminos paralelos con distintos tamaños de filtros.

2.1.4. Generative Adversarial Networks (GAN)

Una *Generative Adversarial Network* (GAN) es una red resultante al aplicar el marco de trabajo desarrollado en [Goodfellow *et al.*, 2014], el cual fue creado para estimar modelos generativos mediante un proceso antagónico. En dicho *framework*, el modelo generativo se enfrenta a un discriminador encargado de determinar si un ejemplo proviene desde la distribución de probabilidad de los datos, o la distribución de probabilidad del modelo generativo.

El ejemplo clásico para entender como funciona el *framework* GAN es el caso de los falsificadores y los policías: un falsificador desea crear dinero falso para utilizarlo como si fuera verdadero, mientras la misión del policía es detectar el dinero falso. A medida que pasa el tiempo ambos equipos van mejorando sus técnicas para derrotar a su adversario, hasta que llega el punto que el falsificador puede crear dinero idéntico al verdadero, siendo imposible para el policía lograr detectar las falsificaciones. En la figura 2.7 se puede observar como actúa una GAN considerando la situación anterior.

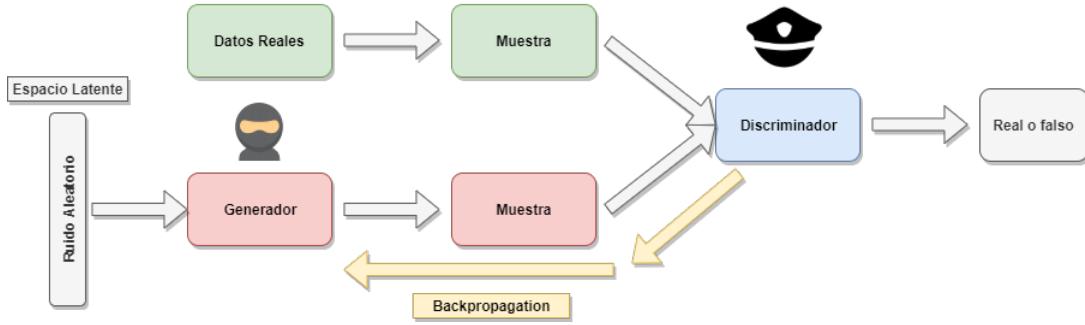


Figura 2.7: Esquema del funcionamiento de una GAN.

Si bien, este framework puede ser aplicado empleando distintas arquitecturas tanto para el generador como para el discriminador, Goodfellow estudia el caso en que tanto el generador como el discriminador son Perceptrones Multicapa (MLP), considerando un generador que toma como input un ruido proveniente de una determinada distribución de probabilidad y entrega como salida una muestra en el espacio de los datos, y un discriminador que parte del espacio de los datos y entrega un escalar indicando la probabilidad de que una muestra provenga desde el conjunto de datos de entrenamiento y no desde el modelo generativo.

Para entender mejor este *framework* es necesario realizar las siguientes definiciones:

- p_g : distribución aprendida por el generador.
- p_{data} : distribución de probabilidad de los datos.
- $p_z(z)$: *prior* sobre el ruido de entrada del generador.
- $G(z; \theta_g)$: mapeo desde el ruido al espacio de los datos. En este caso corresponde a una función diferenciable representada por un MLP con parámetros θ_g .
- $D(x; \theta_d)$: MLP que entrega un escalar. En este caso dicho escalar corresponde a la probabilidad de que una muestra provenga desde p_{data} y no p_g .

La idea detrás del *framework* es entrenar D para maximizar la probabilidad de asignar la etiqueta correcta tanto a los ejemplos de entrenamiento como a las muestras obtenidas desde G . Por otro lado, se entrena G para minimizar $\log(1 - D(G(z)))$, es decir, que el discriminador clasifique como verdadero un ejemplo proveniente de G .

Considerando lo anterior, lo que se busca con el entrenamiento es resolver el siguiente juego *Min-Max*

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}} [\log(D(x))] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.14)$$

Lamentablemente, la ecuación 2.14 no provee el gradiente suficiente para que el generador aprenda de buena manera, debido a que en fases tempranas del entrenamiento el discriminador puede rechazar con mucha confianza las muestras obtenidas desde G al ser claramente diferentes a los ejemplos de entrenamiento. Dado lo anterior, es mejor entrenar al generador para que maximice $D(G(z))$, lo que provee mejores gradientes en fases tempranas de entrenamiento mientras se mantiene la dinámica entre G y D. En la práctica, al momento de entrenar al generador se deben invertir las etiquetas sobre las imágenes sintéticas¹.

Dentro de las ventajas con las que cuentan las GAN se encuentra el hecho de que no necesitan de Cadenas de Markov para su entrenamiento, dado que es posible utilizar *back-propagation*. Además, como el generador obtiene los gradientes desde el discriminador, no tiene contacto directo con los datos de entrenamiento, ganando ventaja estadística frente a otros modelos generativos, esto debido a que se evita copiar el input en los parámetros del generador [Goodfellow *et al.*, 2014].

Por otro lado, una de las grandes desventajas que tienen las GAN es que el discriminador debe estar sincronizado de buena manera con el generador, de lo contrario, lo más probable es que el generador caiga en *mode collapse*, es decir, que grandes porciones del espacio latente caen en el mismo objeto en el espacio de los datos.

Garantías Teóricas

Además de proponer el juego Min-Max de la ecuación 2.14, [Goodfellow *et al.*, 2014] brinda garantías teóricas sobre el óptimo global de dicha ecuación, como también de la convergencia de un algoritmo de entrenamiento basado en *back-propagation*.

¹En la gran mayoría de los *frameworks* modernos de *deep learning*, el efecto de minimizar $D(G(z))$ se logra indicándole al discriminador que las imágenes sintéticas son reales al momento de entrenar al generador. Para este efecto se invierte el *ground truth* considerando 1 (imagen real) en lugar de 0 (imagen sintética)

En el caso del óptimo global, se demuestra que este se alcanza cuando $p_g = p_{data}$, es decir, cuando el generador logra capturar completamente la distribución de los datos de entrenamiento. El razonamiento tras la demostración es el siguiente: Si se considera cualquier generador G , el criterio de entrenamiento para encontrar el discriminador óptimo es maximizar la ecuación 2.14

$$\begin{aligned} V(G, D) &= \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z))) dz \\ V(G, D) &= \int_x p_{data} \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned} \quad (2.15)$$

Como para cualquier (a, b) en $\mathbb{R} - (0, 0)$, la función $f(x) = a \cdot \log(y) + b \cdot \log(1 - y)$ tiene un máximo en $\frac{a}{a+b}$, el discriminador óptimo es el siguiente

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \quad (2.16)$$

Es interesante notar que el objetivo del entrenamiento para obtener D_G^* se puede interpretar como la maximización del criterio de máxima verosimilitud cuando se estima $P(Y = y|x)$ considerando Y como un indicador si una muestra x proviene desde p_{data} o p_g , entregando una mirada estadística de este proceso.

Considerando el clasificador óptimo 2.16, el juego Min-Max de la ecuación 2.14 se puede escribir de la siguiente manera

$$C(G) = \mathbb{E}_{x \sim p_{data}} \left[\log \left(\frac{p_{data}}{p_{data} + p_g} \right) \right] + \mathbb{E}_{x \sim p_g} \left[\log \left(\frac{p_g}{p_{data} + p_g} \right) \right] \quad (2.17)$$

Ahora, considerando $p_g = p_{data}$, el discriminador óptimo tiene un valor de $D_G^* = \frac{1}{2}$, por lo que reemplazando en la ecuación 2.17 se tiene que

$$\mathbb{E}_{x \sim p_{data}} [-\log(2)] + \mathbb{E}_{x \sim p_g} [-\log(2)] = -\log(4) \quad (2.18)$$

Restando la expresión anterior desde la ecuación 2.17

$$\begin{aligned}
C(G) + \log(4) &= \mathbb{E}_{x \sim p_{data}} \left[\log \left(\frac{p_{data}}{p_{data} + p_g} \right) \right] + \mathbb{E}_{x \sim p_g} \left[\log \left(\frac{p_g}{p_{data} + p_g} \right) \right] \\
&\quad - \mathbb{E}_{x \sim p_{data}} [-\log(2)] - \mathbb{E}_{x \sim p_g} [-\log(2)] \\
C(G) + \log(4) &= \int_{x \sim p_{data}} p_{data} \left[\log \left(\frac{p_{data}}{p_{data} + p_g} \right) \right] + p_{data} \log(2) \\
&\quad + \int_{x \sim p_g} p_g \left[\log \left(\frac{p_g}{p_{data} + p_g} \right) \right] + p_g \log(2) \\
C(G) &= -\log(4) + \int_{x \sim p_{data}} \log \left(\frac{2p_{data}}{p_{data} + p_g} \right) + \int_{x \sim p_g} \log \left(\frac{2p_g}{p_{data} + p_g} \right) \\
C(G) &= -\log(4) + KL \left(p_{data} \middle\| \frac{p_{data} + p_g}{2} \right) + KL \left(p_g \middle\| \frac{p_{data} + p_g}{2} \right) \quad (2.19)
\end{aligned}$$

Donde KL corresponde a la divergencia de *Kullback-Leibler*. La expresión anterior se puede escribir en términos de la divergencia de *Jensen-Shannon* entre las distribuciones p_{data} y p_g

$$C(G) = -\log(4) + 2 \cdot JSD(p_{data} || p_g) \quad (2.20)$$

La cual es conocida por ser no negativa y 0 solo cuando las distribuciones que compara son iguales, por lo que el mínimo para $C(G)$ es $-\log(4)$, el cual se alcanza solo cuando $p_g = p_{data}$.

Entrenamiento

Con respecto al entrenamiento de la GAN, debido a su estructura es posible entrenar tanto el generador como el discriminador a la vez utilizando *backpropagation*, tal como se ve en el algoritmo 1.

Algoritmo 1 Entrenamiento de GAN vía gradiente descendente estocástico con *minibatch* [Goodfellow *et al.*, 2014]. k es un hiperparámetro para ajustar el número de pasos de entrenamiento para el discriminador.

- 1: **for** Número de iteraciones del entrenamiento **do**
 - 2: **for** k pasos **do**
 - 3: Muestrear un *minibatch* de tamaño m desde el prior de ruido $p_g(z)$
 - 4: Muestrear un *minibatch* de tamaño m desde la distribución que genera los datos $p_{data}(x)$
 - 5: Actualizar el discriminador vía gradiente ascendente
- $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)}))) \right]$$
- 6: **end for**
 - 7: Muestrear un minibatch de tamaño m desde el prior de ruido $p_g(z)$
 - 8: Actualizar el generador vía gradiente descendente
- $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \left[\log(1 - D(G(z^{(i)}))) \right]$$
- 9: **end for**
-

Si se considera una capacidad y tiempo de entrenamiento suficientes, p_g converge a p_{data} . Lo anterior basado en la convexidad del criterio de entrenamiento. Al ser convexa en p_g , las subderivadas del criterio de entrenamiento incluyen la derivada de la función donde se encuentra su máximo, por lo que al realizar una actualización vía gradiente descendente para p_g considerando D óptimo para el G actual, p_g converge al óptimo global, el cual se demostró que es p_{data} .

Deep Convolutional Generative Adversarial Networks (DCGAN)

Una de las desventajas de la GAN original es que son muy inestables durante su entrenamiento, provocando que la salida del generador muchas veces no tenga sentido. En particular, cuando se intenta escalar utilizando redes convolucionales para la generación de imágenes, estas presentan mucho ruido, siendo bastante incomprensibles.

Con el objetivo de aplicar Redes Neuronales Convolucionales (CNN) de manera no supervisada, se desarrolla una nueva clase de arquitecturas llamadas *Deep Convolutional Generative Adversarial Networks* (DCGAN) [Radford *et al.*, 2015], las cuales siguen una serie de restricciones sobre las arquitecturas clásicas de una CNN, permitiendo que el entrenamiento de una GAN sea estable, logrando imágenes generadas de alta calidad.

Para determinar dichas restricciones se realizó una exploración de modelos, identificando una familia de arquitecturas que permiten un entrenamiento más estable para modelos profundos, que entregan mejor resolución en sus resultados. Para lo anterior se basan en ideas utilizadas en las arquitecturas CNN.

Las restricciones aplicadas son las siguientes

- **Up/Downsampling:** utilizar capas convolucionales con *stride* en lugar de la utilización de capas de *pooling*. Este enfoque es utilizando en la DCGAN tanto en el generador, aprendiendo su propio *upsampling*, como en el discriminador. La idea de utilizar este enfoque proviene de la conocida *All Convolutional Net* [Springenberg *et al.*, 2014], donde logran alcanzar el estado del arte en clasificación de imagen de la época utilizando redes con solo capas convolucionales.
- **No capas Fully Connected:** para lograr esto, en la primera capa de la GAN, la cual toma una muestra del ruido desde el espacio latente, se realiza un *reshape* para obtener un tensor que es utilizado como entrada en la primera capa convolucional. Para la última capa del discriminador, se realiza un proceso de *flatten* desde la última capa convolucional, obteniendo un vector que se entrega a una única neurona con activación sigmoidal.
- **Batch Normalization** [Ioffe y Szegedy, 2015]: para la estabilización del entrenamiento. Esta técnica hace frente a problemas de inicialización, ayudando a que el gradiente fluya en modelos profundos. En la exploración realizada, el uso de *Batch Normalization* fue crítico para que el generador evitara el *mode collapse*, logrando obtener transiciones suaves entre imágenes. Sin embargo, si esta técnica se aplica en todas las capas se obtiene un modelo inestable, lo que puede ser evitado si no se aplica en la capa de salida del generador, ni en la capa de entrada del discriminador.
- **Funciones de Activación:** utilización de *ReLU* como función de activación en el

generador salvo en la capa de salida, la cual utiliza *tangente hiperbólica* como activación. Esto produce que el modelo aprenda más rápido a cubrir el espacio de colores de la distribución de los datos. Para el discriminador, se determinó que el uso de *LeakyReLU* obtiene mejores resultados en casos donde se requiere alta resolución.

En términos del entrenamiento de la DCGAN, se realizó un escalamiento de las imágenes para caer en el rango de la función *Tanh*. Se entrenó mediante gradiente descendente estocástico con un tamaño de *batch* de 128. Los pesos iniciales de la red fueron obtenidos desde una distribución normal centrada en 0, con desviación estándar de 0.02. Para la función de activación *LeakyReLU* del discriminador se determinó una pendiente de 0.2 en el tramo izquierdo, esto en todo el discriminador. Como optimizador se utilizó *Adam* con una tasa de aprendizaje de 0.0002 y *momentum* de 0.5, valor que ayudó a estabilizar el entrenamiento.

Para la experimentación se realizaron pruebas considerando tres *datasets* distintos: *LSUN bedroom* [Yu *et al.*, 2015], *Imagenet* [Deng *et al.*, 2009] y un dataset propio de rostros de personas. Al ver la mejora visual de las imágenes generadas sobre LSUN se pensó que la DCGAN podría estar sufriendo de *overfitting*, por lo que utilizaron 3 millones de ejemplos de entrenamiento, los que fueron pasados por un *de-noising autoencoder* cuyo espacio latente fue binarizado utilizando ReLu, generando un *hash* semántico que permitió eliminar duplicados en tiempo lineal. En la figura 2.8 se pueden observar resultados obtenidos sobre el *dataset* LSUN luego de un *epoch* de entrenamiento, donde los autores mencionan que es muy poco probable que exista *overfitting* dada la utilización de una *learning rate* pequeña y *minibatch SGD*.



Figura 2.8: Dormitorios generados en una iteración [Radford *et al.*, 2015].

Para validar las capacidades de la DCGAN, la utilizaron como extractor de características sobre *datasets* supervisados, para luego evaluar el desempeño de clasificadores lineales entrenados sobre las características extraídas. Utilizando el *dataset* CIFAR-10 [Krizhevsky, 2009], ocupan las *features* convolucionales de las capas del discriminador de la DCGAN para entrar una SVM regularizada (L2). Este enfoque supera resultados obtenidos mediante la utilización de K-means para el aprendizaje de características [Coates y Ng, 2012] utilizando menos *feature-maps* (82.8 % versus 82.0 % de *accuracy*), pero no supera los resultados de la Exemplar CNN [Dosovitskiy *et al.*, 2015] (84.3 % de *accuracy*), aunque algo a destacar es que la DCGAN no fue nunca entrenada sobre CIFAR-10.

También exploraron el espacio latente de la DCGAN entrenada sobre LSUN, mostrando que a medida que se avanza por dicho espacio, lentamente hay ciertos objetos del dormitorio que comienzan a desaparecer y/o transformarse en otros. Esto muestra que la DCGAN aprende representaciones relevantes, tal como se muestra en la figura 2.9, donde se observa como un posible televisor lentamente se transforma en una ventana.



Figura 2.9: Interpolación del espacio latente de la DCGAN [Radford *et al.*, 2015].

Hay que mencionar, que si bien la DCGAN presenta una familia de arquitecturas

convolucionales que permiten estabilizar el entrenamiento al extender el framework GAN, en ocasiones sigue presentando algunas de sus desventajas, como el *mode collapse*, el cual fue detectado en alguno de los experimentos realizados por los autores.

Evaluación de resultados

Uno de los grandes desafíos en la utilización de GANs para la generación de datos sintéticos es la manera de evaluar los resultados obtenidos, esto debido a que el criterio de entrenamiento clásico no asegura nada sobre la calidad de las imágenes.

En [Salimans *et al.*, 2016], utilizan anotadores humanos para evaluar la calidad, solicitando que distingan entre imágenes reales y sintéticas, pero este enfoque tiene la desventaja de que depende mucho de la motivación de los etiquetadores, como también de la tarea que se les entregue. Además, si se les entrega feedback sobre el proceso de etiquetado, los anotadores son capaces de detectar más fácilmente los defectos en la generación de imágenes. Dado lo anterior, proponen un método automático para evaluar los resultados, el cual se conoce como **Inception Score (IS)**, debido a que utiliza la salida de la red Inception [Szegedy *et al.*, 2014] pre-entrenada sobre ImageNet para el cálculo del score. Esta métrica se basa en las siguientes ideas:

- Si una imagen generada contiene claramente un objeto, la salida de una red clasificadora $p(y|x)$ será poco entrópica ya que se clasificará con confianza.
- Si el generador logra generar imágenes de distintas clases, la probabilidad marginal $p(y) = \int p(y|x = G(z))dz$ debe ser altamente entrópica.

Considerando las ideas anteriores, el score propuesto es el siguiente:

$$e^{\mathbb{E}_{x \sim p_{data}} KL(p(y|x)||p(y))} \quad (2.21)$$

donde la exponencial se agrega para que sea más fácil la comparación entre modelos. Para el cálculo de la probabilidad marginal se recomienda utilizar $N = 50000$ para realizar una estimación.

Este score tiene la ventaja que se correlaciona bien con la evaluación humana, por lo que ha sido ampliamente utilizado para la evaluación de métodos generativos sobre imágenes. Sin embargo, en [Barratt y Sharma, 2018] se ilustran algunos defectos del IS:

- **Sensible a los pesos**

Este score es muy sensible a los pesos de la red pre-entrenada. Encuentran grandes diferencias usando la misma red entrenada en distintos frameworks de Deep Learning.

- **Cálculo del score**

Si bien se utiliza $N = 50000$ para la estimación de la probabilidad marginal, esto lo hacen separando dicho N en chunks de datos más pequeños, lo que provoca que el número de datos no sea suficiente para la estimación. También se encuentra que el IS es sensible a la elección de este parámetro.

- **Uso fuera de ImageNet**

Al calcular este score en un dataset que no sea ImageNet los resultados no serán buenos ya que la red pre-entrenada no considera las clases necesarias.

- **Optimización**

Se ilustra que al utilizar este score como criterio de optimización lleva a la generación de *Adversarial Examples*, es decir, imágenes sin sentido aparente pero que logran engañar a la red pre-entrenada.

- **No reporta overfitting**

Si la red generadora memoriza los ejemplos de entrenamiento, inevitablemente se obtendrá una probabilidad condicional poco entrópica, logrando obtener un IS alto.

Otra métrica utilizada para la evaluación de imágenes sintéticas es *Frechét-Inception Distance (FID)* [Heusel *et al.*, 2017], una mejora sobre el Inception Score que sigue utilizando una red pre-entrenada pero utiliza estadísticas del conjunto de datos reales. Basado en la noción de que $p_g = p_{data}$ sí y solo sí los momentos de ambas distribuciones son iguales, calculan la media μ y la covarianza Σ tanto para las imágenes sintéticas como para las imágenes reales, esto utilizando *features* capturadas desde la última

capa de pooling del modelo Inception. Con los momentos calculados a partir de las *features* obtenidas ajustan dos distribuciones gaussianas multivariadas, las que comparan utilizando la distancia de Frechét.

$$d^2((\mu_g, \Sigma_g), (\mu_{data}, \Sigma_{data})) = \|\mu_g - \mu_{data}\|_2^2 + Tr(\Sigma_g + \Sigma_{data} - 2(\Sigma_g \Sigma_{data})^{\frac{1}{2}}) \quad (2.22)$$

Este score también se correlaciona con el juicio humano, además de ser consistente frente a imperfecciones en las imágenes generadas, siendo mejor en este ámbito que el Inception Score [Heusel *et al.*, 2017].

2.2. Trabajos Relacionados

2.2.1. Generación de Imágenes de Lentes Gravitacionales

Se estima que futuros telescopios aumentarán la cantidad de lentes gravitacionales en tres órdenes de magnitud, provocando que los métodos de detección basados en inspección visual sean imposibles de aplicar en las decenas de millones de imágenes que se esperan, lo que hace latente la necesidad por métodos automáticos de detección de lentes.

The strong gravitational lens finding challenge

Motivados por la situación anterior, investigadores del grupo *GLENCO*² crean el ***Gravitational Lens Finding Challenge*** [Metcalf *et al.*, 2018], donde se entregan imágenes sintéticas de lentes gravitacionales con el objetivo de que se desarrolle nuevos métodos de detección de lentes gravitacionales. En este challenge se ofrecen dos categorías con un conjunto de entrenamiento de 20000 imágenes de 101x101 pixeles, entregando a cada equipo uno de cinco conjuntos de pruebas con 100000 imágenes. El objetivo de cada equipo es entregar un score de confianza para cada imagen, considerando un rango de 0 a 1, con 0 indicando que no hay un lente en la imagen y 1 indicando la presencia de este.

²<http://glenco.unibo.it/>

Como la cantidad de lentes encontrados actualmente ronda los miles, los autores del challenge desarrollan un *pipeline* de generación de lentes sintéticos donde utilizan una simulación de N-cuerpos (*N-Body simulation*) [Boylan-Kolchin *et al.*, 2009] para obtener halos de materia oscura, los cuales fueron caracterizados en términos de su masa, su tamaño y su radio medio de masa.

El catálogo de halos obtenidos se ingresa en GLAMER [Metcalf y Petkova, 2014] [Petkova *et al.*, 2014], una librería escrita en C++ para la simulación de lentes, donde GLAMER ajusta el perfil de masa de los halos al modelo NFW [Navarro *et al.*, 1996]. Utilizando la misma librería, extraen los halos con un Radio de Einstein (radio de los anillos formados) que sean mayor a 1.5 veces la resolución de la imagen deseada. Esta colección de halos se utilizan como lentes gravitacionales, los cuales son rotados para contar con aún más lentes.

Para modelar las fuentes, toman objetos del *Hubble Ultra Deep Field* donde se obtienen 9350 fuentes con 4 bandas, las cuales son obtenidas mediante la aplicación de funciones *shapelet* que modelan la emisión de luz de las galaxias [Meneghetti *et al.*, 2008]

Para crear las imágenes toman un lente y lo rotan aleatoriamente, luego toman un objeto fuente aleatorio y se comparan sus redshifts (longitud de onda desplazada al espectro rojo): si el redshift del objeto es menor al del lente se elige otro objeto³, caso contrario se determina la distancia del centro del lente al punto más lejano de su cáutica, para colocar la fuente dentro de 3 veces la distancia calculada. Lo anterior fue diseñado para favorecer la producción de lentes que se vean claramente.

También simulan galaxias visibles utilizando modelos semi-analíticos [Guo *et al.*, 2011] las cuales se agregan a la imagen siguiendo las mismas rotaciones que el lente seleccionado. Esto se realiza con el objetivo de que no sea tan fácil ajustar un modelo que separe el lente de la fuente, ya que estas galaxias no simulan la población real de estas. Finalmente, calculan el mapa de masa completo utilizando la Transformada Rápida de Fourier sobre una estructura de árboles para calcular las desviaciones de la luz causadas por dicho mapa.

Estas simulaciones se ajustan a las categorías del challenge, aplicando una función de

³El redshift es comúnmente utilizado como medida de distancia, por lo que este caso implica que el objeto fuente está más cerca del observador que el mismo lente.

dispersión de punto Gaussiana (PSF) para así tener las características que tendrían los telescopios Euclid o KiDS. Para la categoría basada en Euclid, llamada *Space Based*, se consideran imágenes con un solo canal, mientras que para la categoría *Ground Based* se utilizan imágenes con 4 canales.

PICS: SIMULATIONS OF STRONG GRAVITATIONAL LENSING IN GALAXY CLUSTERS

Otro pipeline de simulación es **PICS** (*Pipeline for Images of Cosmological Strong lensing*) [Li *et al.*, 2016], el cual fue diseñado para cerrar la distancia entre la gran cantidad de datos de las nuevas survey y las simulaciones de N-cuerpos. Este pipeline considera un estimador de densidad, cálculo de los ángulos de desviación utilizando métodos de Fourier, módulos para la construcción de imágenes de los objetos fuentes, galaxias perteneciente al lente y otros objetos de interés, y finalmente un módulo que ajusta las imágenes para que coincidan con un determinado telescopio.

En términos de la simulación también utilizan una simulación de N-cuerpos para la distribución de masas, pero utilizando HACC [Habib *et al.*, 2016], que tiene mejor performance y corre en una gran cantidad de arquitecturas de super-computadoras. Para obtener los mapas de masa utilizan el método *Delaunay tessellation field estimator (DTFE)* [Schaap y van de Weygaert, 2000] en lugar el perfil NFW. Para construir las imágenes de los objetos fuente utilizan objetos del *Hubble Ultra Deep Field* al igual que el método anterior, pero no aplica métodos para reducir el ruido de éstos, sino que los utiliza directamente. Al calcular el ángulo de las desviaciones producidas por los mapas de masas utilizan la Transformada Rápida de Fourier, pero a diferencia del método del challenge, utilizan una estructura de grillas y no sobre árboles, permitiendo más flexibilidad a cambio de costo computacional.

Con las componentes anteriores aseguran la generación de lentes gravitacionales fuertes. Para asegurar que las imágenes sean realistas agregan otros componentes: galaxias pertenecientes al lente como tal, galaxias presentes en la linea de visión que no están afectadas significativamente por el lente, y *faint stars*. Finalmente este *stack* de imágenes se pasa por el módulo que ajusta las imágenes, utilizando PSF y añadiendo el ruido característico del telescopio deseado.

PICS es eficiente computacionalmente dentro de la escala en la que actúa, esto considerando el uso de super-computadoras trabajando paralelamente. Dentro de las evaluaciones en el tiempo de cómputo, utilizando 24 cores en procesadores Xeon E5-2620 es posible simular 20000 imágenes en una semana, mientras que con la super-computadora “Edison” (NERSC Cray XC30) es posible generar 1 millón de imágenes en una semana, esto utilizando solo el 4 % del poder computacional disponible.

2.2.2. Ganadores Challenge Detección de Lentes

En el *Gravitational Lens Finding Challenge* se recibieron 24 participaciones entre ambas categorías, las cuales consideraron una gran variedad de técnicas: inspección visual, detección de arcos, *Machine Learning* y *Deep Learning*, siendo más utilizadas las técnicas de aprendizaje automático. Los ganadores en ambas categorías fueron soluciones basadas en Redes Convolucionales: una arquitectura basada en VGG [Schaefer *et al.*, 2018] para la categoría *Space Based*, y una arquitectura basada en ResNet [Lanusse *et al.*, 2017] para la categoría *Ground Based*.

Deep convolutional neural networks as strong gravitational lens detectors

Lensfinder, la red ganadora de la categoría Space Based, está basada en la red VGG que considera el uso de bloques compuestos por dos capas convolucionales con sus respectivas funciones de activación, *Batch Normalization* y *Max Pooling*. Esta red utiliza 4 de los bloques descritos anteriormente, aumentando el número de filtros utilizados en una potencia de dos entre ellos. El último de estos bloques no considera una capa de *pooling* y agrega *Dropout* entre las capas convolucionales para combatir un posible *overfitting*. Para realizar la clasificación de las imágenes consideran dos capas *fully connected* con 1024 neuronas, utilizando para la salida una capa con una neurona con activación sigmoidal. La arquitectura completa de Lensfinder se presenta en la figura 2.10.

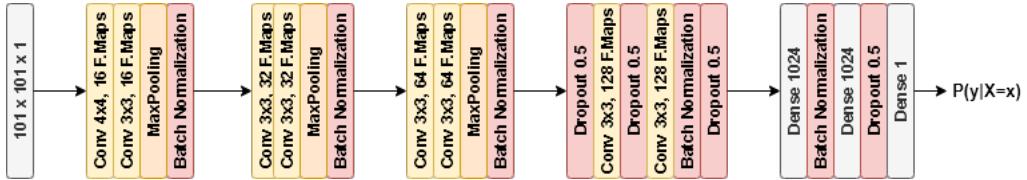


Figura 2.10: Diagrama de la red Lensfinder.

Para la activación utilizada en el resto de capas consideran una modificación de la función ReLU [Arpit *et al.*, 2016], presentada en la ecuación 2.23. Esta modificación busca que los datos de distribuyan de forma normal estándar durante el *forward pass*, logrando el efecto de reducir la varianza de la función de costo durante el entrenamiento.

$$f(x) = \frac{1}{\sqrt{\pi - 1}} \left(\sqrt{2\pi} \max(0, x) - 1 \right) \quad (2.23)$$

Como pre-procesamiento aplican una normalización de las imágenes, restando la media y dividiendo por la desviación estándar de los ejemplos de entrenamiento. Aumentan el *dataset* aplicando rotaciones en 90 grados, flips horizontales y verticales, cambios en la intensidad de los píxeles en un rango de 0.8 a 1.2, utilizando el mismo factor para todos, y desplazamientos leves en la imagen. Durante el entrenamiento utilizan un *mini-batch* balanceado de 30 imágenes, utilizando *Adam* como optimizador durante 250 a 300 epochs, con binary-crossentropy como función de costo. Del total de imágenes de entrenamiento utilizan el 75 % para entrenar y el 15 % para validación.

En términos de resultados obtienen un *AUROC* de 0.93, siendo el mejor desempeño en su categoría superando a otros 13 métodos de detección. Los investigadores también prueban otros tipos de arquitecturas más complejas, pero estas obtienen resultados similares, por lo que se opta por la más simple.

CMU DeepLens: Deep Learning For Automatic Image-based Galaxy-Galaxy Strong Lens Finding

En la categoría *Ground Based* el ganador es una solución que utiliza una ResNet (residual network) [Lanusse *et al.*, 2017] para la detección de lentes. *CMU DeepLens* considera el uso de dos tipos de bloques residuales en su arquitectura, donde uno de ellos mantiene

las dimensiones de la imagen mientras que el segundo disminuye las dimensiones a la mitad utilizando *strided convolutions*. Una de las características de estos bloques es que utilizan bloques pre-activados, es decir, aplican la función de activación antes de aplicar la convolución y no después como es usual, quedando esta después de la aplicación de *Batch Normalization*. Los bloques de construcción se presentan en la figura 2.11, donde a la izquierda se ilustra el bloque que mantiene las dimensiones, y a la derecha se encuentra el bloque que aplica *downsampling* en las dimensiones mientras que aumenta los canales. Cabe destacar que en el bloque encargado del *downsampling* aplica *batch normalization* y la función de activación antes de separar los caminos, ya que al utilizar pre-activaciones el input de la capa convolucional del *shortcut* es igual al input de la primera capa convolucional del bloque.

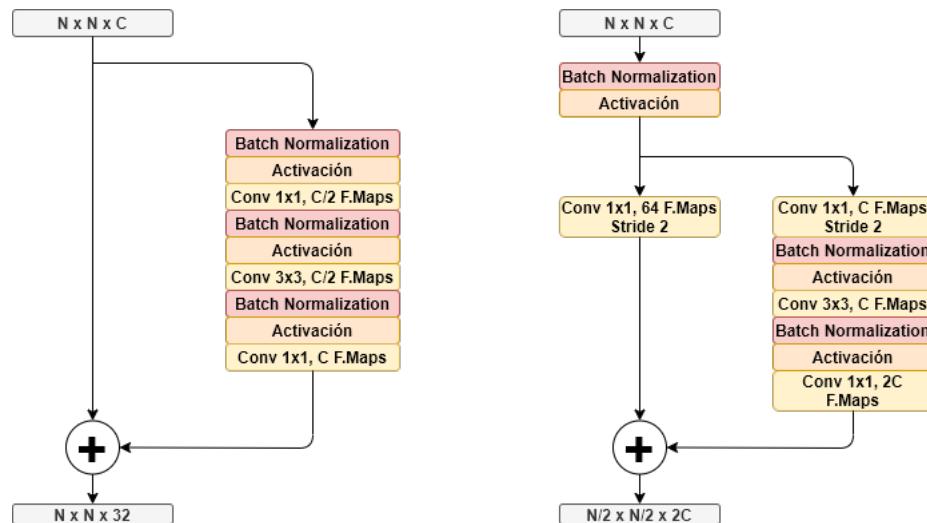


Figura 2.11: Bloques ResNet utilizado por CMU DeepLens.

La arquitectura está formada por la concatenación de 2 o 3 bloques que mantienen las dimensiones seguidos de un bloque de *downsampling*. Este tipo de construcción se repite 5 veces, llegando a un total de 46 capas de profundidad. Cada uno de estos bloques utiliza la activación ELU (*Exponential Linear Unit*). Para realizar la clasificación, utilizan *Global Average Pooling* para terminar con una neurona con activación sigmoidal.

En términos de pre-procesamiento, realizan una normalización para cada banda de las imágenes. Para esto restan la media de las imágenes desde el set de entrenamiento, para luego normalizar por la desviación estándar σ . También se aplica *clipping* considerando

un rango definido por $k\sigma$, con $k = 250$. En caso de tener píxeles enmascarados⁴ se dejan con un valor de 0. Al igual que la solución anterior, aumentan los datos utilizando rotaciones, *flips* y *zooms*. Al igual que Lensfinder utilizan *binary crossentropy* como función de costo, entrenando durante 120 epochs con *Adam* como optimizador sobre *mini-batch* de 128 imágenes. Aplican un *scheduler* para la tasa de aprendizaje, reduciendo ésta en un orden de magnitud cada 40 epochs.

Esta red ganó la categoría *Ground Based* del *challenge* con un AUROC de 0.98, superando a otros 9 participantes. Algo a destacar es que, si bien para su participación en el challenge entrenaron con las imágenes entregadas, toda la experimentación para lograr llegar a esta arquitectura la realizan sobre un *dataset* creado por los autores utilizando *PICS* para la simulación.

2.2.3. GAN en Astronomía

Si bien, el trabajo que se presenta en esta memoria es el primero en generar imágenes de lentes gravitacionales fuertes utilizando GANS, existen trabajos donde se utilizan GANS sobre otro tipo de datos astronómicos.

CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks

CosmoGAN [Mustafa *et al.*, 2019] es una red GAN que busca emular el proceso de simulación de mapas de convergencia de lentes gravitacionales débiles, con el fin de disminuir el cómputo en super-computadoras necesario para su generación. Para esto, utilizan mapas generados a partir de una simulación de N-cuerpos, logrando generar mapas con las mismas características estadísticas que la simulación original.

Como los mapas son invariantes a traslaciones utilizan capas convolucionales, aplicando una arquitectura basada en la DCGAN. Para el generador emplean un *prior* de 64 dimensiones obtenido desde una distribución normal estándar conectado a una capa *fully connected* desde la cual se aplica un reshape para obtener la forma de los feature maps.

⁴En el trabajo asociado al *challenge* [Metcalf *et al.*, 2018] se menciona la existencia de regiones enmascaradas debido a la eliminación de elementos indeseados presentes en las imágenes simuladas

Consideran 4 capas de convoluciones transpuestas con un stride de 2x2, cada una seguida de Batch Normalization salvo la última capa. Utilizan ReLU como activación salvo para la capa de salida, que utiliza la función tangente hiperbólica. Para el discriminador utilizan un espejo del generador pero utilizando capas convolucionales normales con activación LeakyReLU.

Para el entrenamiento utilizan los parámetros sugeridos por los autores de la DCGAN, terminando el entrenamiento cuando los mapas generados pasan un test de Kolmogorov–Smirnov sobre la intensidad de los píxeles y un test visual, en el cual un experto intenta diferenciar los mapas generados de los provenientes de la simulación.

El enfoque para la evaluación de resultados es que los mapas generados mantengan las características estadísticas del *dataset* original. Calculan estadísticas de primer orden sobre la intensidad de los píxeles, estadísticas sobre la densidad espectral y estadísticas no gaussianas, aplicando test estadísticos considerando como hipótesis nula que las estadísticas de los mapas generados y los mapas de validación provienen de la misma distribución, mostrando que las imágenes simuladas mantienen las estadísticas originales.

Enabling Dark Energy Science with Deep Generative Models of Galaxy Images

El objetivo de este trabajo es la generación de imágenes de galaxias condicionadas a las estadísticas propias de éstas [Ravanbakhsh *et al.*, 2016] utilizando imágenes desde el *Hubble Space Telescope*. Para este objetivo consideran la utilización de un *Variational Autoencoder Condicional* (VAE) y una DCGAN condicionada para mejorar el hecho que los VAE usualmente generan imágenes borrosas.

Los autores intentaron utilizar la DCGAN original pero no lograron ajustar los hiperparámetros de este tipo de arquitectura sobre su dataset. Al no poder entrenar correctamente la GAN cambian el objetivo del entrenamiento: en lugar de utilizar un discriminador utilizan un regresor ya entrenado sobre el conjunto de entrenamiento, y entran el generador hasta que se obtengan resultados tan buenos como los obtenidos por el regresor en las imágenes originales.

Para la evaluación de resultados, se calculan estadísticas sobre la intensidad de los píxeles considerando la forma y el tamaño de las galaxias generadas. Calculando el segundo momento, se observa que las galaxias son ligeramente más elípticas que las originales,

pero en general mantienen la distribución original.

Capítulo 3

Propuesta de Solución y Validación

En este capítulo se presenta la propuesta de solución para el problema de generación de lentes gravitacionales fuertes, incluyendo los experimentos realizados para obtener la estrategia de entrenamiento y la arquitectura definitiva de la red, como también la propuesta de validación, la cual incluye el diseño de los experimentos y los métodos de evaluación de estos.

3.1. Propuesta de Solución

Se propone la utilización de una arquitectura y esquema de entrenamiento basado en la DC-GAN, utilizando el *framework* Keras [Chollet, 2015] con Tensorflow [Abadi *et al.*, 2015] como Back-End para su implementación.

Para definir la propuesta de solución se consideran dos tipos de experimentos: primero se evalúan distintas técnicas para el entrenamiento de la red, para luego experimentar con cambios sobre la arquitectura de la GAN.

3.1.1. Definición del entrenamiento

En este tipo de experimentos se mantiene una arquitectura base diseñada a partir CIFAR-10 [Brownlee, 2019], sobre la cual se evalúa distintas técnicas para mejorar su estabilidad

y calidad en el *dataset* del *Gravitational Lens Finding Challenge*. Esta arquitectura se presenta en la figura 5.1 del apéndice 1 (capítulo 5).

Las técnicas y configuraciones de entrenamiento consideradas en los experimentos son las siguientes:

1. **One-side label smoothing** [Salimans *et al.*, 2016]: consiste en cambiar la etiqueta de 1 a 0.9 en los ejemplos reales al entrenar al discriminador. Esto provoca que el discriminador no sea tan fuerte en fases tempranas del entrenamiento.
2. **Noisy labels**, consiste en realizar un “*bit flip*” en algunas etiquetas aleatorias al entrenar el discriminador.
3. **Cambios en la inicialización de pesos**, se prueban dos inicializaciones: *Random Normal*, como en la DCGAN original, y *Glorot Uniform* [Glorot y Bengio, 2010], la inicialización por defecto en el *framework* utilizado.

Cada una de estas técnicas se aplica de manera incremental, determinando la utilidad de éstas evaluando tanto el comportamiento de la función de pérdida, específicamente la varianza del discriminador y generador, como también la calidad de las imágenes generadas.

En cada prueba se consideran distintas tasas de aprendizaje iniciales, ya que este hiperparámetro influye de gran manera durante el entrenamiento, considerando 20 epochs de entrenamiento para cada una.

3.1.2. Arquitectura de la red

Para definir la arquitectura de la red se realizan experimentos variando la arquitectura de la red, comparando el desempeño versus la arquitectura base utilizada en la sección anterior. Estas variaciones, aplicadas de manera incremental, se presentan a continuación

1. Cambios en el número de filtros

La arquitectura base mantiene el número de filtros entre los bloques convolucionales del generador y en los primeros dos bloques del discriminador, por lo que se prueba

el diseño original de la DCGAN [Radford *et al.*, 2015] en el cual se aumentan en potencias de dos en el generador, y disminuye en potencias de dos en el discriminador.

2. Disminución de un bloque en el generador

Usualmente, la arquitectura del discriminador es más profunda que la del generador, por lo que se experimenta con la disminución de un bloque convolucional en el generador.

3. Cambios en el número de capas convolucionales por bloques

Muchas veces las arquitecturas utilizadas sobre imágenes consideran el uso de más de una capa convolucional en cada bloque. Dado lo anterior, se aumenta el número de estas capas en los bloques del generador.

Al igual que en los experimentos anteriores, para cada tipo de arquitectura se considera 20 epochs de entrenamiento bajo distintas tasas de aprendizaje iniciales, considerando los mismos criterios de evaluación que en la subsección anterior. Estas arquitecturas se presentan con detalle en las figuras 5.2, 5.3 y 5.4 del capítulo 5.

3.1.3. Propuesta

En base a los resultados obtenidos en los experimentos anteriores, presentados y discutidos en el siguiente capítulo, la arquitectura final utilizada se presenta en la figura 3.1. Para lograr obtener las dimensiones deseadas para la imagen (101x101) se aplica *Zero-padding* en el comienzo del generador y un *resize* basado en interpolación de vecinos más cercanos al final de este.

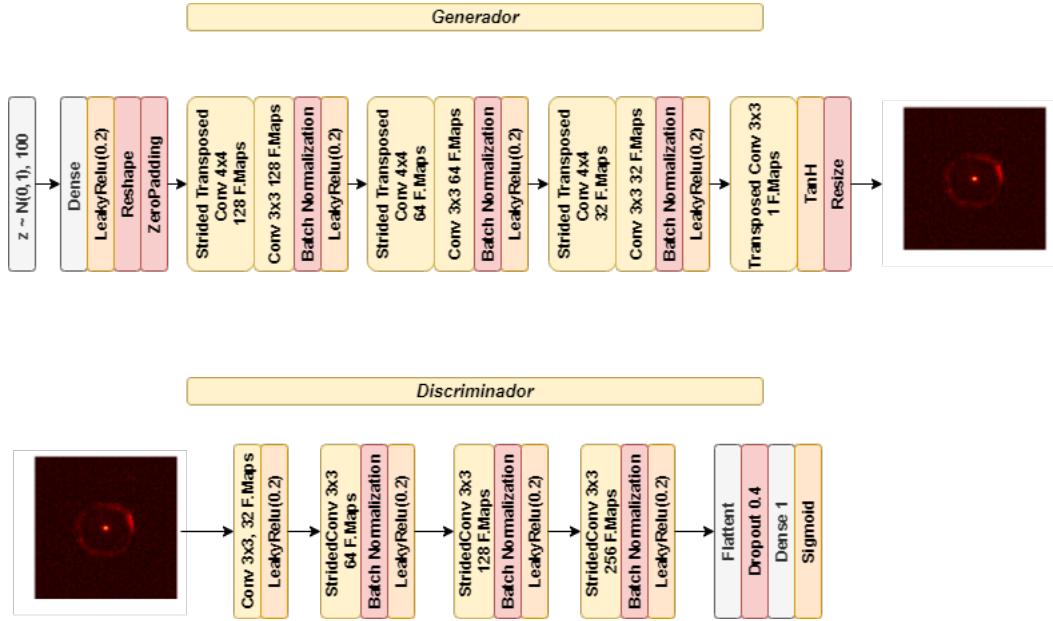


Figura 3.1: Arquitectura propuesta tanto para el generador como para el discriminador de la DCGAN implementada.

Para el discriminador se consideran bloques clásicos de una CNN, utilizando *strided convolutions* para reducir las dimensiones entre los bloques. En el caso de generador se utilizan convoluciones transpuestas con 2 de stride para aumentar las dimensiones. Se utilizan filtros de tamaño 4 en las convoluciones transpuestas para evitar que un mismo pixel dependa de dos campos de acción, dado que dicho fenómeno provoca un efecto de grilla en la imagen generada [Odena *et al.*, 2016].

Durante el entrenamiento se considera el uso de *one-side label smoothing* y *noisy labels*. El tamaño de batch es de 64, el cual fue elegido debido a que es la máxima potencia de 2 que soporta el hardware utilizado. Se considera una actualización de los parámetros entrenables del discriminador por cada ajuste en el generador.

3.2. Propuesta de Validación

Para la validación de la propuesta anterior se consideran dos experimentos: el primero, para determinar si las imágenes generadas por la GAN efectivamente contienen lentes gravitacionales, mientras que el segundo es para analizar la diversidad en los tipos de

lentes generados, así como los posibles errores en su generación. En ambos experimentos se considera la participación de un experto en lentes gravitacionales fuertes.

3.2.1. Primer experimento

Este experimento considera la utilización de 400 imágenes: 200 imágenes sin la presencia de lentes gravitacionales y 200 imágenes de lentes. De las 200 imágenes de lentes se toman 100 imágenes aleatorias desde el dataset original y 100 imágenes sintéticas, mientras que la totalidad de las imágenes sin lentes se obtienen desde el set de imágenes originales.

Para la generación de las imágenes sintéticas se utilizan dos redes considerando la arquitectura de la figura 3.1, cada una entrenada durante 100 epochs, obteniendo 50 imágenes de cada una. Para esto, en cada red se obtienen 50 muestras aleatorias desde el espacio lantente, guardando tanto la imagen en formato *png*, como guardando el arreglo de *Numpy* correspondiente en formato *npz*.

Al momento de generar una imagen sintética, o seleccionar una imagen del dataset original, se le asigna un identificador único. Las 400 imágenes, en formato *png*, ya individualizadas, se envían al experto solicitando que etiquete cada una de estas indicando la presencia o no de un lente gravitacional fuerte, permitiendo adicionalmente una categoría intermedia para indicar incerteza. El objetivo de este tipo de etiquetación es comparar el desempeño del experto entre imágenes reales y sintéticas; si el desempeño es similar se puede decir que las imágenes sintéticas tienen una calidad tal que son capaces de engañar al experto, mientras que si el desempeño al etiquetar los lentes sintéticos es menor, es necesario mejorar la calidad de éstos. Además, el hecho de considerar dos redes distintas permite comparar el desempeño de éstos considerando los métodos de evaluación presentados en la sección 3.2.1, para estudiar la correlación de éstos con el juicio del experto.

Evaluación

Para la evaluación de los experimentos descritos anteriormente se considera la utilización de las siguientes métricas/métodos

■ ***FID* modificado**

Como se explicó en el capítulo 2 sección 2.1.4, los *scores* basados en la red *Inception* tienen problemas al ser aplicados en dominios fuera de *ImageNet*. Dado lo anterior, se propone una modificación de la distancia *Frechét-Inception*, cambiando la red Inception original, entrenada sobre *Imagenet*, por la red *Lensfinder* [Schaefer *et al.*, 2018].

Para obtener las activaciones desde esta red se replica el trabajo realizado por los autores, ya que esto permite tener más control sobre las capas de la red al momento del cálculo del FID. La replica realizada considera la misma estrategia de entrenamiento, es decir, el uso de *mini-batches* balanceados de tamaño 30, optimizador Adam con una tasa de aprendizaje inicial de 0.001, 200 *epochs*, y los mismos métodos para aumentación de datos. Si bien, en el paper original no se menciona el pre-procesamiento utilizado para las imágenes, se utiliza el mismo procedimiento que la implementación en *Tensorflow* realizada por los autores [Geiger, 2017]. Esta réplica logra alcanzar un *AUROC* de 0,93, el mismo logrado por *Lensfinder* en su participación oficial en el *challenge* de detección de lentes.

Se opta por una modificación sobre FID y no *Inception score* debido a que el generador implementado solo considera la generación de imágenes de lentes gravitacionales, por lo que no se espera que la probabilidad marginal $p(y)$ sea altamente entrópica al considerar una sola clase, lo que es contrario a las motivaciones detrás de la formulación del IS.

■ ***Accuracy***

Considerando las anotaciones proporcionadas por el experto en lentes gravitacionales, se considera la utilización de la *accuracy* para medir cuántas de las imágenes generadas por la GAN logran convencer a un experto versus la cantidad de imágenes del challenge que logran el mismo efecto. Es de esperar que si la GAN logra capturar de buena manera la distribución de probabilidad de los datos, la *accuracy* obtenida para ambos conjuntos de imágenes sea similar.

También se considera la *accuracy* obtenida mediante el uso de la red clasificadora implementada para el cálculo de FID. De esta manera, se mide esta métrica tanto para el etiquetador experto en el dominio como para un método de clasificación automático.

■ Interpolación del espacio latente

Tanto el FID como la *accuracy* no reportan *overfitting* en el generador, por lo que es necesario evaluar si esto ocurre. Para este fin se realiza una interpolación en el espacio latente para evaluar cualitativamente las transiciones entre dos puntos del espacio latente. De existir *overfitting* es de esperar que estas transiciones sean bruscas, mientras que transiciones suaves son signo de que la red aprendió un mapeo significativo.

En la DCGAN original [Radford *et al.*, 2015] los autores utilizan una interpolación lineal para este efecto. Sin embargo, en [White, 2016] se indica que dicha interpolación no es la ideal en modelos generativos que tengan un espacio latente de alta dimensionalidad (> 50), proponiendo el uso de *Slerp* (Spherical Linear Interpolation) [Shoemake, 1985], el cual considera un camino por el “Gran Círculo” en una hyper esfera n-dimensional.

$$slerp(p_0, p_1, t) = \frac{\sin(1-t)\theta}{\sin(\theta)}p_0 + \frac{\sin(t\theta)}{\sin(\theta)}p_1 \quad (3.1)$$

- p_0, p_1 : puntos en el espacio latente.
- t : parámetro que define la posición del nuevo punto entre p_0 y p_1 . Toma valores en el rango $[0, 1]$.
- θ : ángulo entre los puntos p_0 y p_1 .

Considerando que el espacio latente de la GAN implementada en este trabajo es de 100 dimensiones, se utiliza *slerp* para generar distintas interpolaciones a partir de dos puntos seleccionados aleatoriamente desde el espacio latente, interpolando 6 puntos entre ellos. Esto se realiza para cada modelo sometido a validación, permitiendo así evaluar la presencia de *overfitting* en cada red.

3.2.2. Segundo experimento

Durante el desarrollo del experimento anterior, cuyos resultados se presentan en el siguiente capítulo, el astrónomo experto detectó dos situaciones de interés:

1. En el conjunto de datos utilizados para el experimento existe presencia de distintas clases de lentes: **Anillos de Einstein**, donde se separan sistemas con un anillo,

de sistemas con múltiples anillos; ***Quassarlike***, que incluyen replicación de puntos con gran luminosidad, las cuales son características cuando hay un Quasar involucrado; **Insuales**, los cuales presentan sistemas con múltiples imágenes, pero con características muy inusuales; y finalmente **Otros**, que son básicamente sistemas con múltiples imágenes y características ordinarias.

2. Existen imágenes con evidencia de lentes gravitacionales, pero que también cuentan otras características que no deberían existir según la teoría.

La primera situación es de interés para el análisis de la diversidad de las imágenes generadas, ya que se espera que la GAN aprenda a generar todos los tipos de lentes presentes en el conjunto de entrenamiento. Por otro lado, la segunda situación podría indicar posibles errores de la GAN en la generación de lentes.

En base a lo anterior, se definen dos nuevas tareas a realizar por el astrónomo:

1. Del conjunto de imágenes ya etiquetadas como lentes en el experimento anterior, se solicita una nueva clasificación considerando los 5 sub-tipos de lentes. El objetivo es estudiar si la GAN logra generar la misma diversidad de lentes presentes en el dataset original, o en caso contrario determinar que clases presentan problemas.
2. Para las imágenes de lentes en la cual existe una discrepancia entre la etiqueta del astrónomo y la etiqueta del *challenge*, se solicita una explicación cualitativa de los elementos que llevaron a dicha discrepancia. El objetivo de esta tarea es determinar si estas características solo están presentes en las imágenes generadas por GAN; de ser así, significa que la red GAN presenta errores en su generación, mientras que si estos elementos también están presentes en las imágenes del *challenge* indicaría que el error proviene de la simulación.

Capítulo 4

Resultados y Análisis

En este capítulo se presentan los resultados y análisis de los experimentos detallados en el capítulo anterior.

Cada resultado fue obtenido utilizando el siguiente hardware

- **CPU:** Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30GHz.
- **RAM:** 12GB SODIMM DDR4 2400 MHz.
- **GPU:** NVIDIA GeForce GTX 1050 4GB.
- **HDD:** 1TB.

Las versiones de los principales lenguajes y librerías utilizadas son las siguientes

- Python 3.5.2.
- Keras 2.3.1.
- Tensorflow-GPU 2.1.0.
- Jupyter Notebook 6.0.3.

4.1. Definición del entrenamiento

Para la definición de la estrategia de entrenamiento se analiza tanto el comportamiento de la función de pérdida en el discriminador y el generador, como la calidad de las imágenes generadas. El análisis del comportamiento del entrenamiento se realiza mediante gráficos de la función de pérdida, en los cuales se limita el eje de las ordenadas entre 0 y 5 ya que en el comienzo del entrenamiento usualmente se producen valores extremos.

Cada una de las pruebas considera la siguiente configuración de entrenamiento

- **Epochs:** 20.
- **Mini-batch:** 64, definido así al ser la potencia de dos más grande que soporta la memoria de la GPU.
- **Tasa de aprendizaje:** {0.0004,0.0003,0.0002,0.0001,0.00009}.
- **Optimizador:** ADAM con $\beta_1 = 0,5$.

4.1.1. Arquitectura base

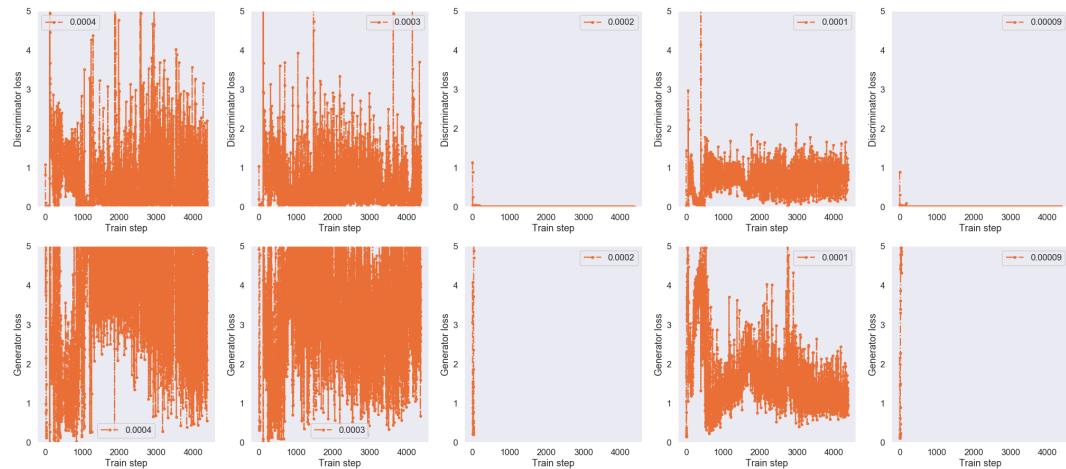


Figura 4.1: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje utilizando la arquitectura base.

En la figura 4.1 se presenta el comportamiento de la función de pérdida utilizando la arquitectura base (figura 5.1) para cada una de las tasas de aprendizaje consideradas. Se logra observar una gran varianza tanto para discriminador y generador al utilizar 0,0004 y 0,0003 como tasa de aprendizaje, esto siendo más notorio en el generador, donde se observan valores mayores a 5 durante prácticamente todo el entrenamiento. Los valores habituales, al menos sobre Cifar-10, rondan entre 1 y 2. Para las tasas de aprendizajes de 0,0002 y 0,00009, se observa que el entrenamiento diverge completamente, mostrando la gran inestabilidad de esta arquitectura base en el dataset de lentes gravitacionales, y dejando clara la sensibilidad del entrenamiento frente al valor de la *learning rate* inicial. Para 0,0001 se obtiene un entrenamiento algo más estable, con una función de pérdida con mucha menos varianza que para las primeras tasas de aprendizaje, mostrando un comportamiento en el discriminador cercano al esperado.

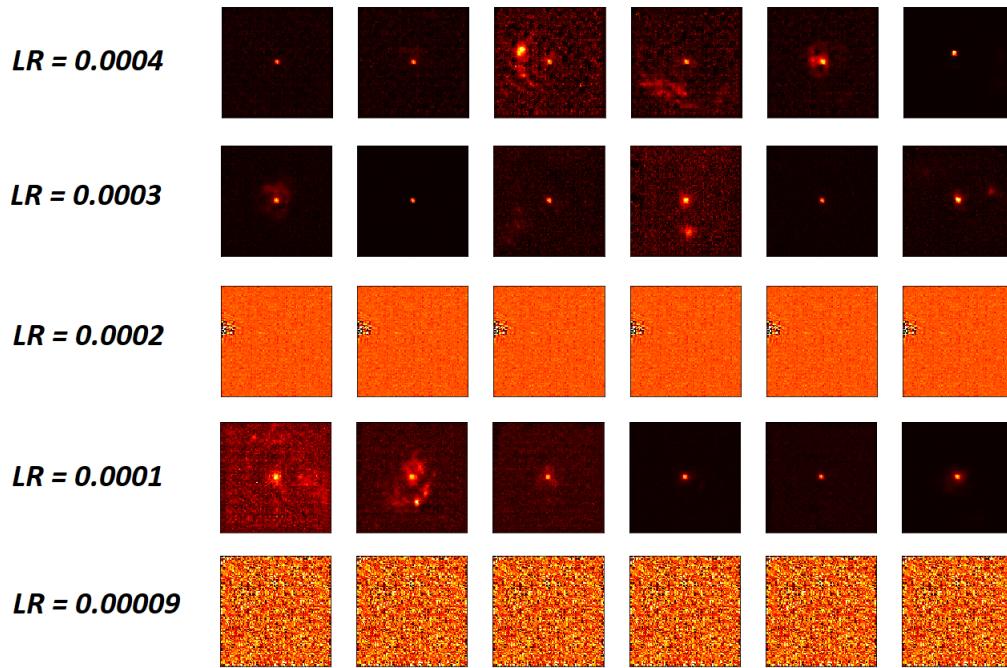


Figura 4.2: Imágenes generadas con la arquitectura base para cada una de las tasas de aprendizaje utilizadas.

Una vez concluido el entrenamiento y para cada tasa de aprendizaje se generan 6 imágenes, las cuales se presentan en la figura 4.2. Para las tasas de aprendizaje donde el entrenamiento diverge, se obtienen imágenes con solo ruido. Además, estas imágenes son

iguales, indicando que el generador sufre de *mode collapse*. Para las otras *learning rates*, se observan pixeles más marcados, los cuales forman un efecto de tablero de ajedrez. Ninguna de las imágenes generadas forma arcos, siendo lo más cercano la segunda imagen para la tasa de aprendizaje de 0,0001. Cabe destacar que para esta última tasa de aprendizaje se producen imágenes un poco más diversas que para las otras tasas, mostrando una generación de mejor calidad, lo que se condice con el comportamiento obtenido durante el entrenamiento.

4.1.2. *One-side label smoothing*

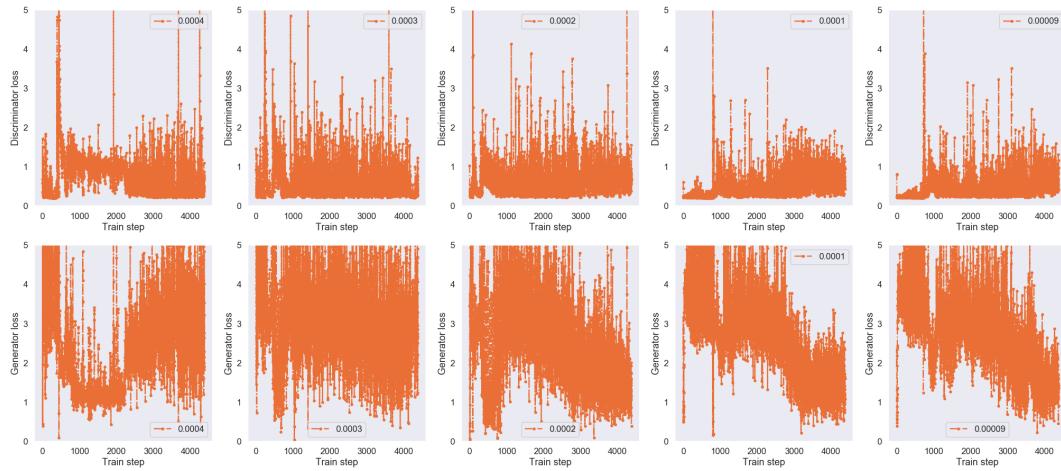


Figura 4.3: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje utilizando *One-side label smoothing* sobre la arquitectura base.

El comportamiento al aplicar *one-side label smoothing* durante el entrenamiento se presenta en la figura 4.3. Uno de los primeros cambios que es posible apreciar en comparación a la arquitectura base es que el entrenamiento ya no diverge para las tasas de aprendizaje 0,0002 y 0,00009, mostrando que esta técnica ayuda a mejorar la estabilidad del entrenamiento. Considerando el efecto que tiene esta técnica en el discriminador, es posible determinar que la divergencia en la arquitectura base se debe a un discriminador muy potente al inicio del entrenamiento, el cual no deja espacio para la mejora del generador. En términos de la varianza de la función de pérdida, se observa que esta disminuye notoriamente en el generador con tasa de aprendizaje inicial de 0,0004, y de manera un poco menos notoria para 0,0003. La varianza en los discriminadores

disminuye levemente, salvo en los casos en los cuales el entrenamiento divergía. La reducción de las varianzas se observa de mejor manera en la figura 4.4, la cual muestra una comparación entre ambos experimentos.

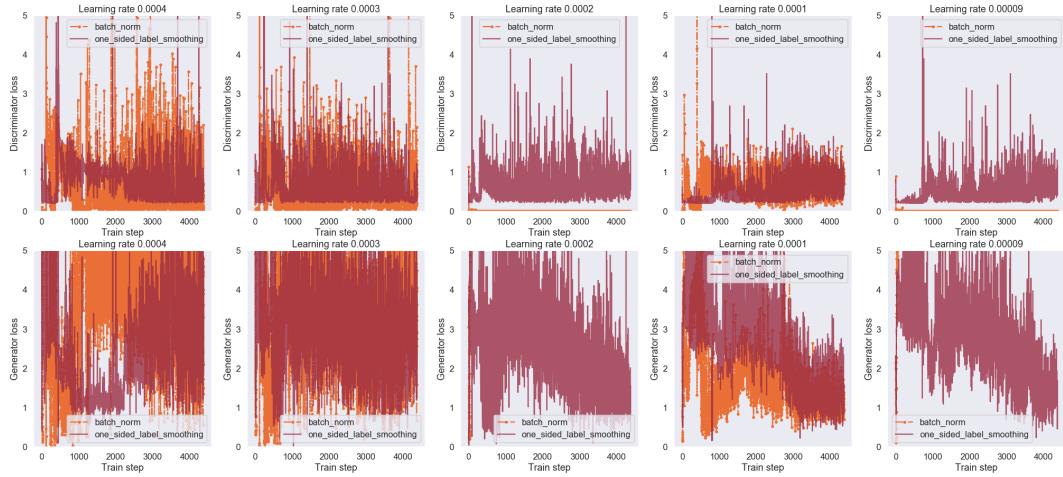


Figura 4.4: Comparación del comportamiento del entrenamiento entre la arquitectura base y el uso de *one-side label smoothing*.

En términos de las imágenes generadas se mantiene el efecto de tablero de ajedrez. Sin embargo, se logra ver más variedad en la generación que al no utilizar la técnica, mostrando así el efecto positivo que se obtiene al utilizarla. También se logra apreciar más estructuras similares a un anillo generado por el fenómeno de lente gravitacional. Las imágenes se presentan en la figura 4.5, siendo la primera imagen con $lr = 0,0003$ y la segunda y tercera con $lr = 0,0002$ las que presentan formas similares al anillo deseado.

Considerando la reducción en las varianzas de las funciones de pérdida, y el aumento leve en la calidad de las imágenes, se toma la decisión de considerar *one-side label smoothing* en el entrenamiento de la solución final.

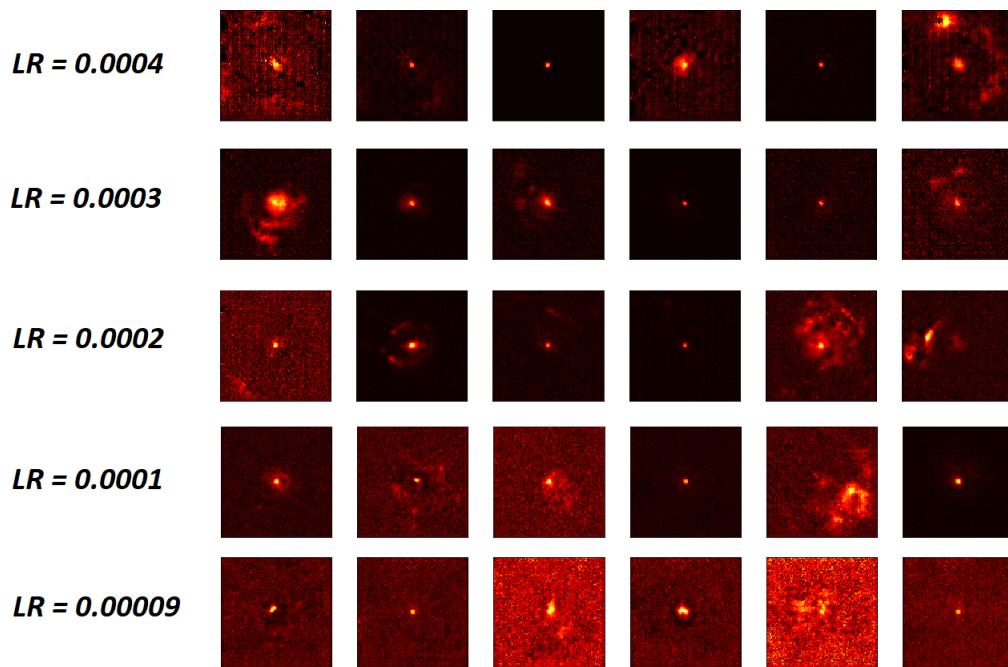


Figura 4.5: Imágenes generadas al utilizar *one-side label smoothing* sobre la arquitectura base para cada una de las tasas de aprendizaje utilizadas.

4.1.3. Noisy labels

La figura 4.6 muestra el comportamiento del entrenamiento al aplicar ruido en las etiquetas al entrenar. A simple vista no se aprecia una reducción notoria en las varianzas, pero si se observa una disminución de los *peaks* en la función de pérdida en el discriminador, estando estos mucho más bajos.

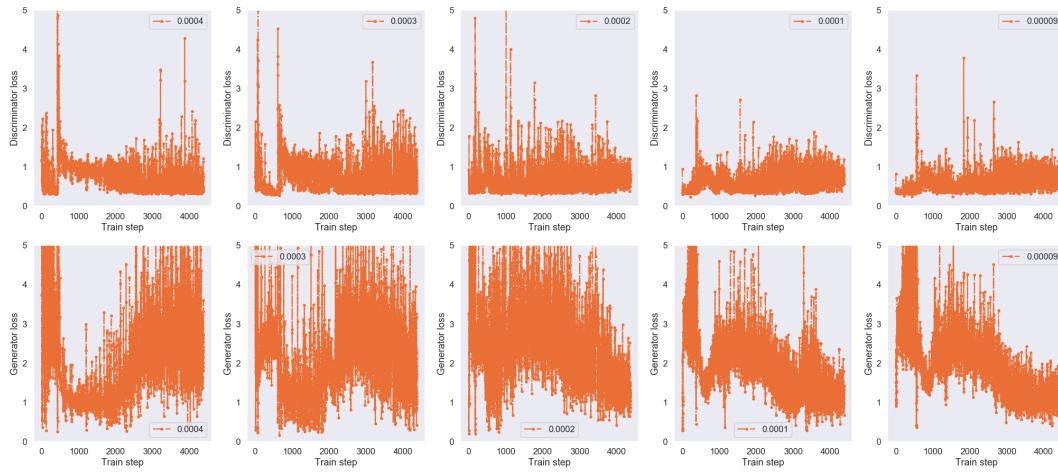


Figura 4.6: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al incluir *noisy labels*.

Al observar la comparación de la figura 4.7, se aprecia que la varianza se reduce levemente, esto tanto en discriminador como generador. También se ve de mejor manera la reducción de los *peaks* comentados anteriormente.

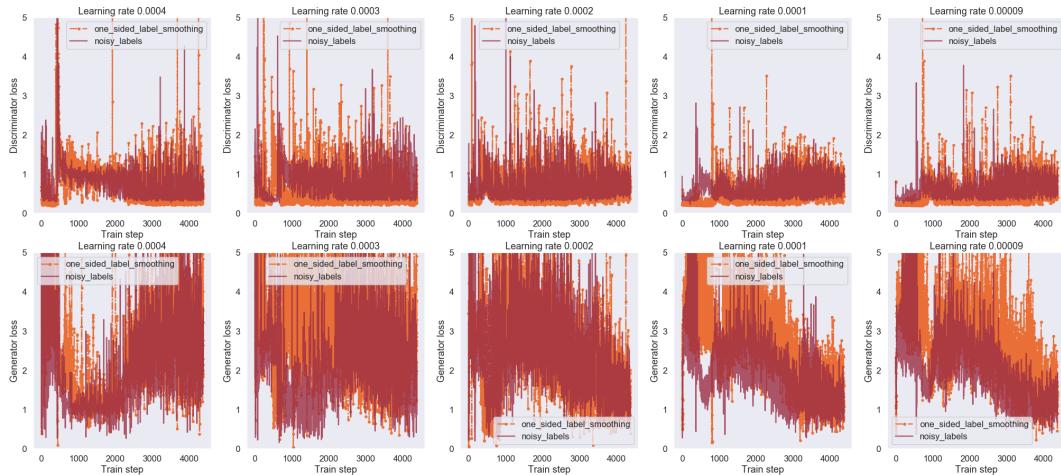


Figura 4.7: Comparación del comportamiento del entrenamiento al agregar el uso de *noisy labels*.

En términos de las imágenes generadas, no se aprecian mayores diferencias salvo en las imágenes obtenidas a partir del generador con tasa de aprendizaje inicial de 0,0004, las cuales presentan manchas negras en la parte superior.

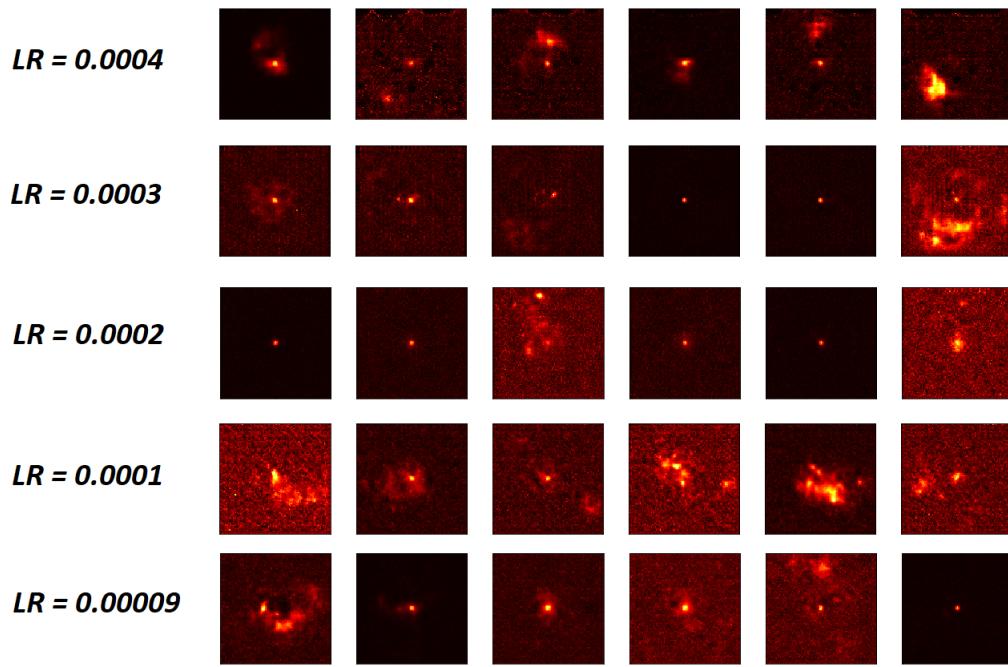


Figura 4.8: Imágenes generadas al utilizar *noisy labels* sobre la arquitectura base para cada una de las tasas de aprendizaje utilizadas.

Se decide adoptar esta técnica para la solución final debido a la disminución de los *peaks* en la varianza del discriminador, la disminución de la varianza en general y la mantención de la calidad de las imágenes. El último criterio toma en cuenta las manchas para la primera *learning rate*, a las cuales se les baja su relevancia debido a que solo aparecen para la versión del experimento más inestable.

4.1.4. *Cambio en la inicialización de pesos*

En la figura 4.9 se observa el comportamiento al cambiar la manera en que se inicializan los pesos, pasado de una inicialización *glorot uniform*, la cual obtiene sus parámetros iniciales desde la distribución uniforme $U\left(-\sqrt{\frac{6}{(n_{in}+n_{out})}}, \sqrt{\frac{6}{(n_{in}+n_{out})}}\right)$, a una inicialización desde una distribución normal con media 0 y varianza 0.2¹.

¹Parámetros utilizados en la DCGAN [Radford *et al.*, 2015]

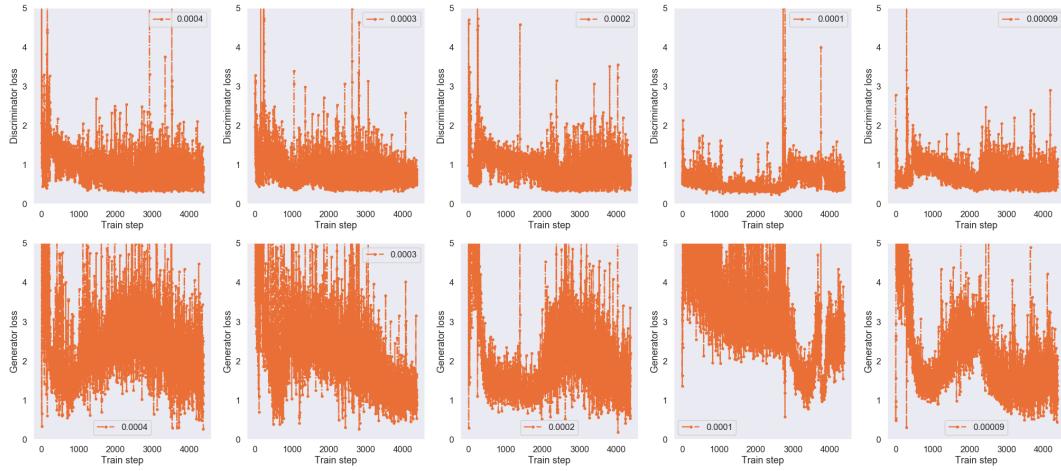


Figura 4.9: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al obtener los pesos iniciales de la red a partir de una distribución normal con media 0 y varianza 0.2.

Se observa que la función de pérdida en el discriminador es más compacta, lo que se aprecia de mejor manera en la comparación de la figura 4.10, donde además se nota la disminución de la varianza en los generadores. Las versiones más estables son las que consideran una tasa de aprendizaje de 0,0001 y 0,00009.

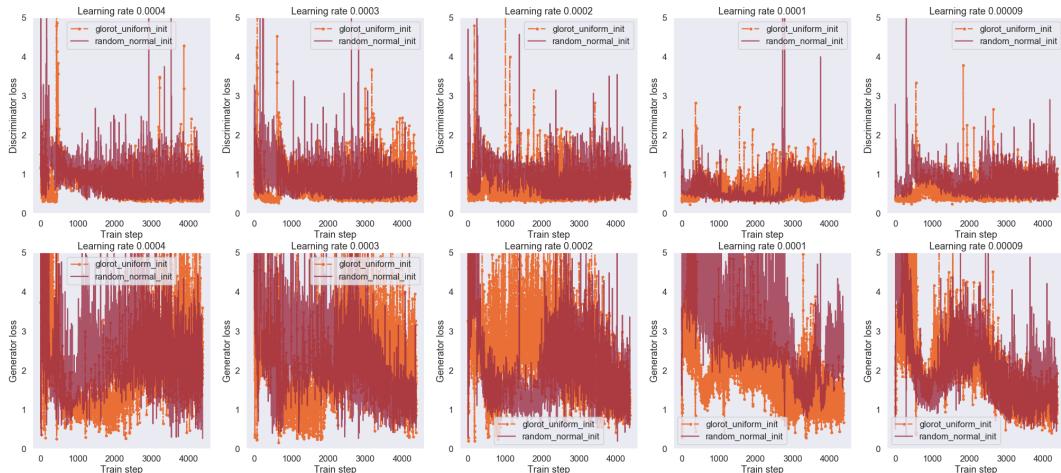


Figura 4.10: Comparación del comportamiento del entrenamiento cambiando el inicializador de pesos.

Observando las imágenes generadas se aprecia que aparece una estructura de grilla en

las imágenes, siendo éste más notorio en las versiones con tasa de aprendizaje más grande (0,0004 y 0,0003) y menos notoria para la versión con tasa de aprendizaje más baja (0,00009). También hay que destacar que las imágenes generadas para una tasa de aprendizaje de 0,0001 tienen una caída muy grande en su calidad, presentando el fenómeno de las grillas de manera más grande y de color negro. Las imágenes analizadas se presentan en la figura 4.11.

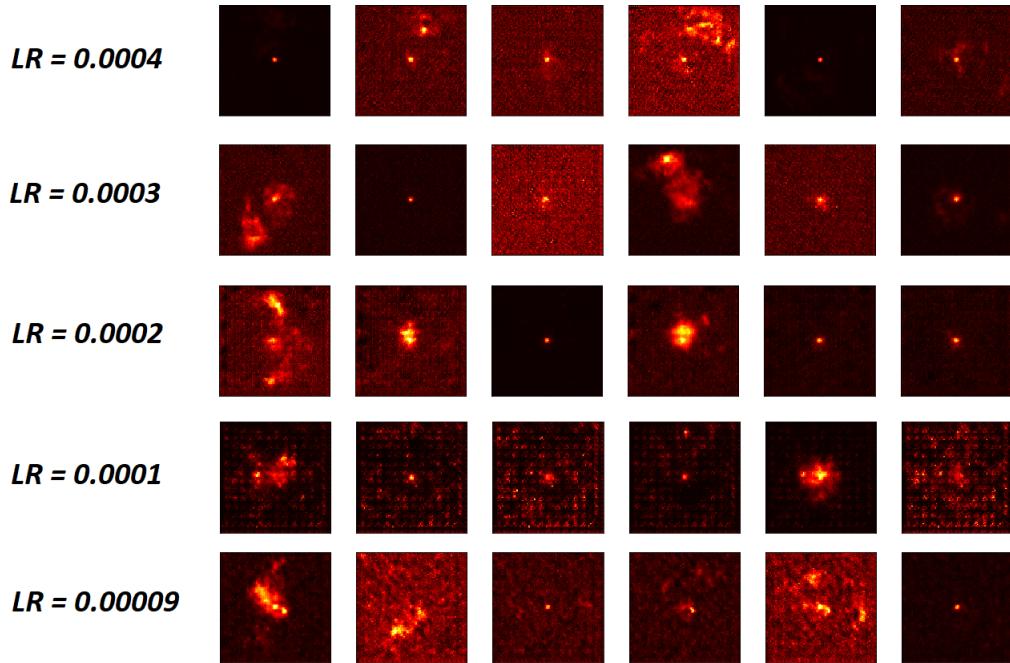


Figura 4.11: Imágenes generadas al utilizar una distribución normal con media 0 y varianza 0.2 para inicializar los pesos de la red GAN.

Si bien, este cambio ayuda a mejorar la estabilidad del entrenamiento, también provoca una caída notoria en la calidad de las imágenes, por lo que se decide no adoptar esta inicialización, manteniendo la opción utilizada anteriormente.

4.1.5. Resumen

En la tabla 4.1 se presenta un resumen de los resultados obtenidos en los experimentos para la definición de las estrategias de entrenamiento.

Técnica	Efecto en entrenamiento	Efecto en imágenes	Conclusión
<i>One-side label smoothing</i>	Mejora la estabilidad Reducción de la varianza de la función de pérdida	Imágenes más variadas	Se adopta la técnica
<i>Noisy labels</i>	Reducción de peaks y la varianza (levemente) de la función de pérdida	Se mantiene la calidad	Se adopta la técnica
<i>Random Normal Initializer</i>	Reducción de la varianza de la función de pérdida	Genera un efecto de grilla, empeorando la calidad	Se descarta la técnica

Tabla 4.1: Tabla resumen de los resultados para los experimentos de definición de las estrategias de entrenamiento.

4.2. Definición de la arquitectura

Al igual que en los experimentos para la definición de las estrategias de entrenamiento, en estos experimentos también se analiza el comportamiento de la función de pérdida, y la calidad de las imágenes generadas. En este caso se realizan modificaciones en la arquitectura base, tales como la modificación en la cantidad de filtros, cambios en la profundidad de la red y cambios en la profundidad de los bloques convolucionales utilizados.

La configuración del entrenamiento para estos experimentos es la siguiente

- **Epochs:** 20.
- **Mini-batch:** 64, definido así al ser la potencia de dos más grande que soporta la memoria de la GPU.
- **Tasa de aprendizaje:** {0.0004,0.0003,0.0002,0.0001,0.00009}.
- **Optimizador:** ADAM con $\beta_1 = 0.5$.
- **Técnicas:** uso de *one-side label smoothing* y *noisy labels*.

4.2.1. Cambios en el número de filtros

En este enfoque se cambia la manera en que se define el número de filtros en cada bloque convolucional. En la arquitectura base (figura 5.1 apéndice 1), el generador mantiene el número de *feature maps* en 128, por lo que en este enfoque se cambia esta situación a un número de filtros descendente basado en potencias de dos, mientras que en el discriminador se aumentan en potencias de dos. Un primer cambio inmediato en el entrenamiento que trae este cambio es la disminución de los parámetros entrenables de la red, pasando de aproximadamente $4,7M$ a $2,3M$ parámetros, lo que baja el tiempo de entrenamiento aproximadamente a la mitad (6 minutos por *epoch* a 3 minutos aproximadamente).

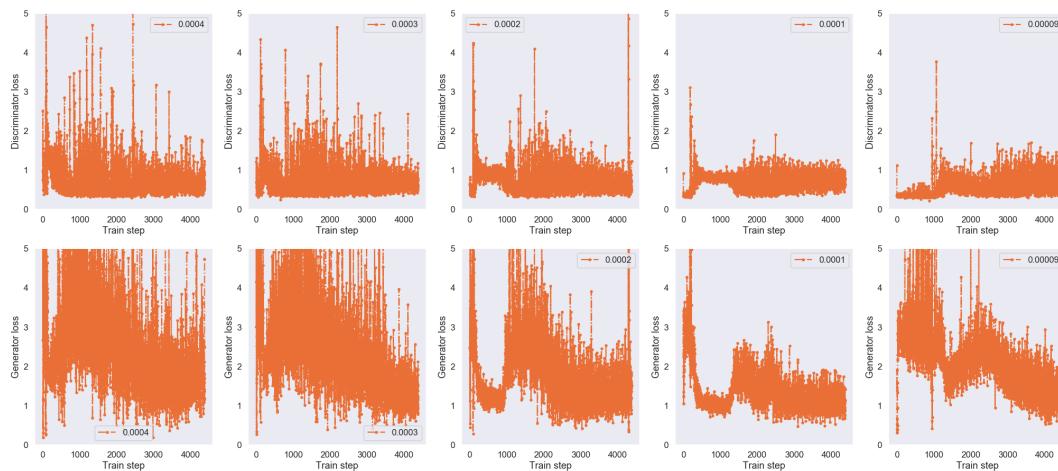


Figura 4.12: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al cambiar el número de filtros de la arquitectura base.

La figura 4.12 muestra el comportamiento de la función de pérdida. Se aprecia que este comportamiento es más compacto para las versiones con menor tasa de aprendizaje, esto tanto en el discriminador como el generador. La situación anterior se observa de mejor manera en la comparación de la figura 4.13. Del mismo modo la disminución de la varianza en el generador de manera más notoria, provocando que la estabilización inicial ocurra en fases más tempranas del entrenamiento.

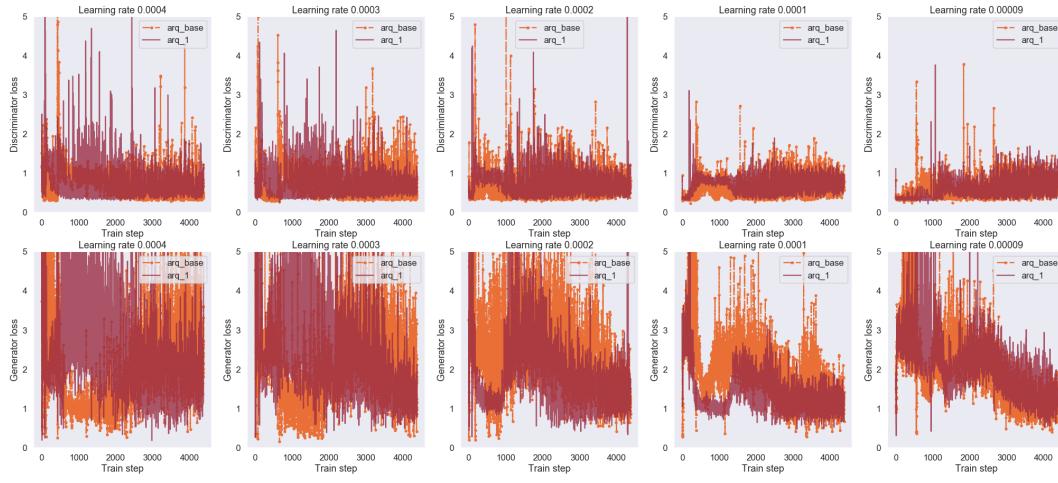


Figura 4.13: Comparación del comportamiento del entrenamiento al cambiar la cantidad de filtros de la arquitectura base.

Las imágenes generadas por esta arquitectura se presentan en la figura 4.14. Un gran cambio que se aprecia es que se generan imágenes con un anillo más definido en la mismas condiciones de entrenamiento que la arquitectura base. El efecto de grilla se presenta en la tercera imagen para el generador con *learning rate* de 0,00009, pero considerando que los otros generadores no lo presentan, o es menos notorio, se presume que con más *epochs* de entrenamiento este efecto debería desaparecer.

Considerando las mejoras en la estabilidad del entrenamiento, la disminución del tiempo de entrenamiento, y la mejora en la calidad de las imágenes, se adopta este cambio arquitectural.

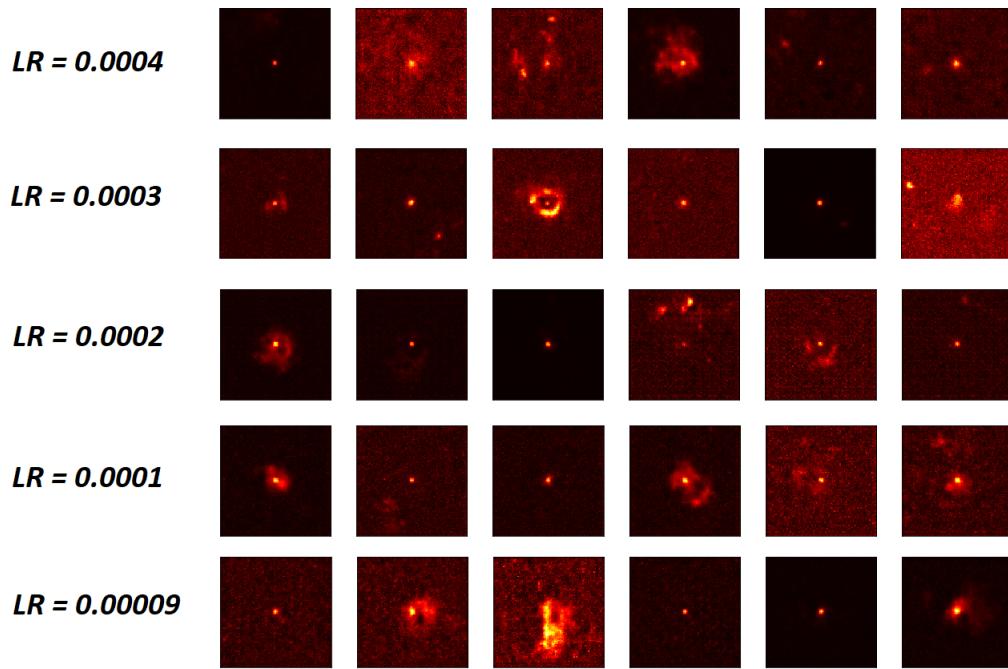


Figura 4.14: Imágenes generadas al cambiar el número de filtros de la arquitectura base.

4.2.2. Disminución de un bloque en el generador

Usualmente el generador de una GAN es menos profundo que el discriminador, por lo que se evalúa el efecto al disminuir un bloque convolucional para provocar este efecto. Al tener menos capas uno esperaría que la cantidad de parámetros entrenables disminuya, pero en este caso este aumenta a $8,4M$ de parámetros aproximadamente, esto debido a que al eliminar un bloque convolucional del generador se realiza un aumento dimensional menor, provocando que la capa densa tenga que tener más neuronas para igualar las dimensiones del primer bloque. Sin embargo, la situación anterior no provoca un aumento del tiempo de entrenamiento.

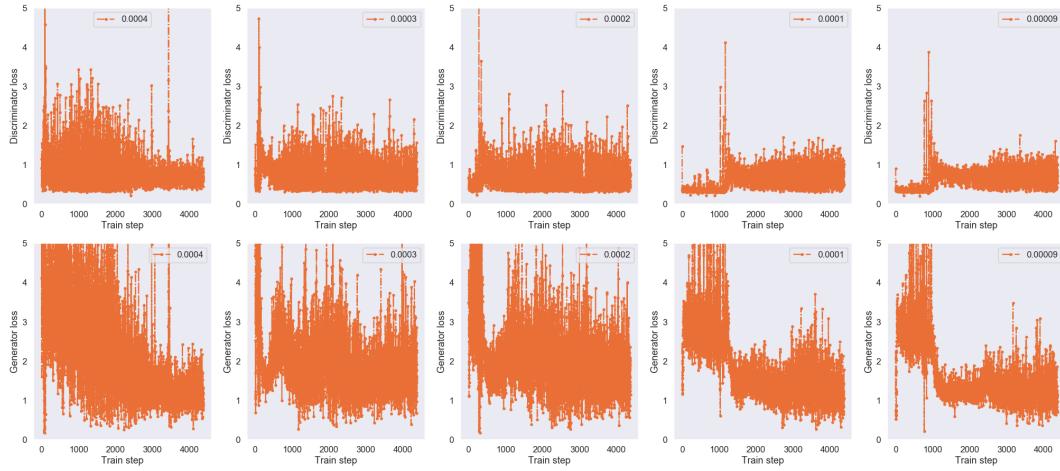


Figura 4.15: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al eliminar un bloque convolucional en el generador.

En términos del comportamiento del entrenamiento, presente en la figura 4.15, se sigue manteniendo la tendencia, donde las versiones más estables a nivel de varianza son los casos con tasas de aprendizaje más bajas. Observando la comparación con la arquitectura anterior, presente en la figura 4.16, se ve que el comportamiento del discriminador no se ve afectado por el cambio del generador, esto a pesar del frágil equilibrio de estos componentes en una red GAN, mientras que el generador mejora su comportamiento para algunas *learning rates* (0,0004,0,0002 y 0,00009) y empeora para otras.

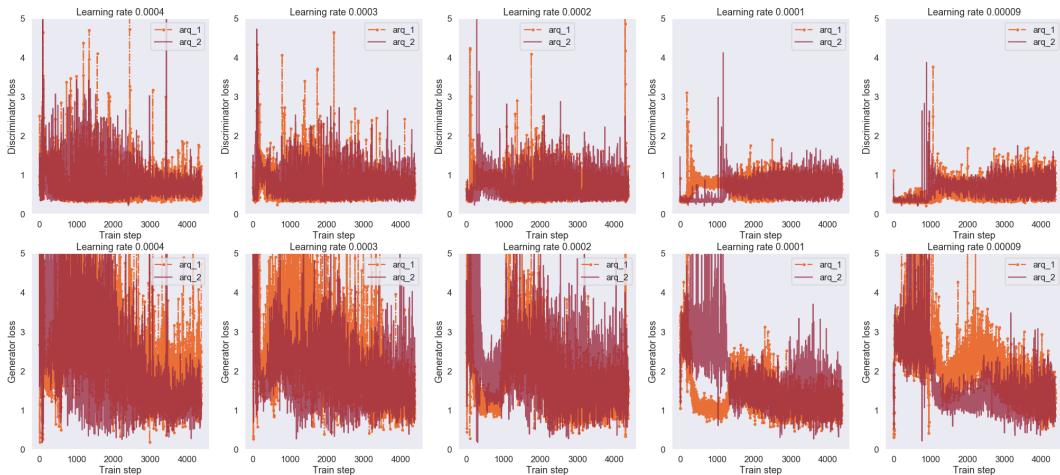


Figura 4.16: Comparación del comportamiento del entrenamiento al eliminar un bloque convolucional en el generador.

En la figura 4.17 se presentan imágenes generadas por esta arquitectura. A simple vista no se observan mayores diferencias con la arquitectura anterior, salvo que el efecto de grilla se hace más notorio que la arquitectura anterior para versiones con tasa de aprendizaje mayor (0,0002 vs 0,00009).

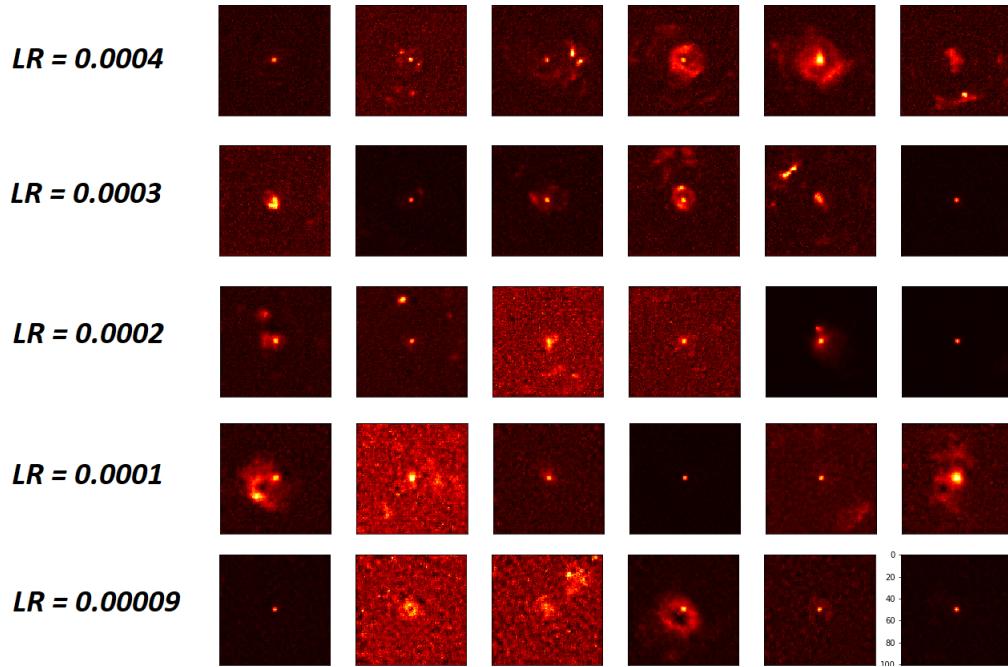


Figura 4.17: Imágenes generadas al cambiar eliminar un bloque convolucional del generador.

Considerando el aumento de los parámetros entrenables de esta arquitectura, la mantención en la calidad de las imágenes, y que las mejorar en la estabilidad del entrenamiento se producen solo para algunas *learning rates*, se decide no adoptar este cambio.

4.2.3. Cambios en el número de capas convolucionales por bloques

Siguiendo las arquitecturas de redes convolucionales como VGG, se experimenta con agregar una capa convolucional más a los bloques que componen el generador, aumentando en 3 el número de capas convolucionales. Este cambio provoca un aumento de parámetros de $2,3M$ a $2,4M$ aproximadamente, lo que en la práctica no representa un

aumento considerable en el tiempo de entrenamiento.

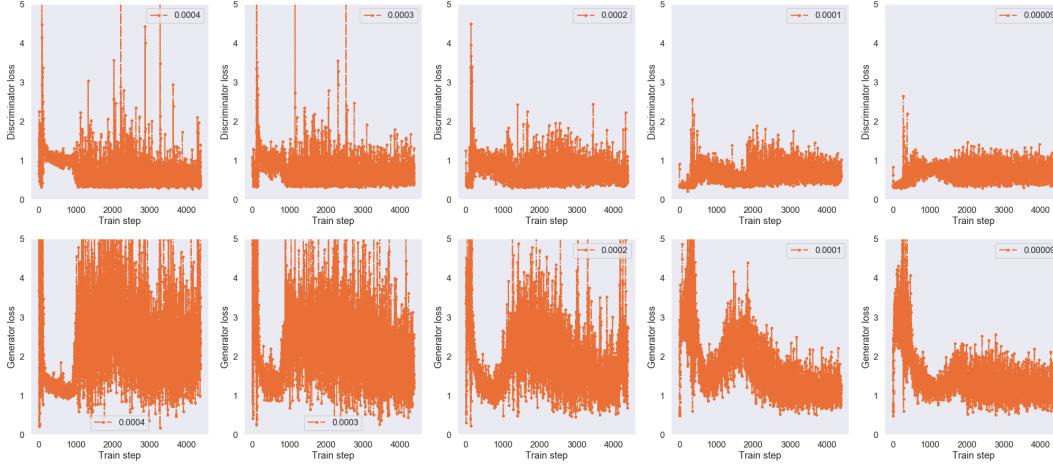


Figura 4.18: Función de pérdida versus pasos de entrenamiento para 5 distintas tasas de aprendizaje al agregar una capa convolucional en los bloques del generador.

Con respecto al entrenamiento, cuyo comportamiento se presenta en la figura 4.18, se observa que se reduce la varianza en el generador de manera notoria para las tasas de aprendizaje 0,0001 y 0,00009. La varianza en el discriminador se reduce levemente, lo que se aprecia de mejor manera en la comparación de la figura 4.19.

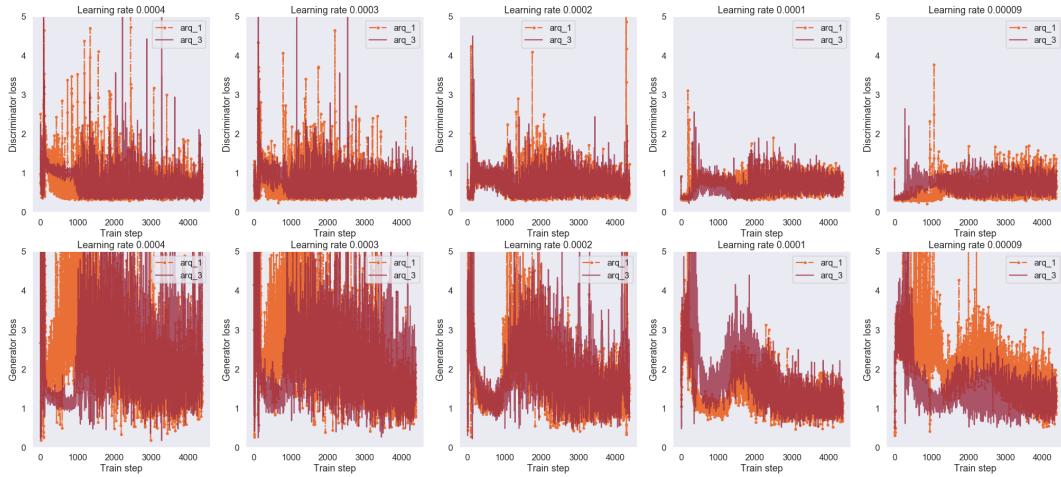


Figura 4.19: Comparación del comportamiento del entrenamiento al agregar una capa en los bloques del generador.

En la misma figura se observa que el generador se demora algunos *epochs* más en

estabilizarse, siendo esto más notorio para las *learning rates* más altas, pero considerando la estabilidad posterior del entrenamiento esta situación es aceptable, sobretodo considerando que para llegar una solución final se aumentará la cantidad de *epochs* del entrenamiento.

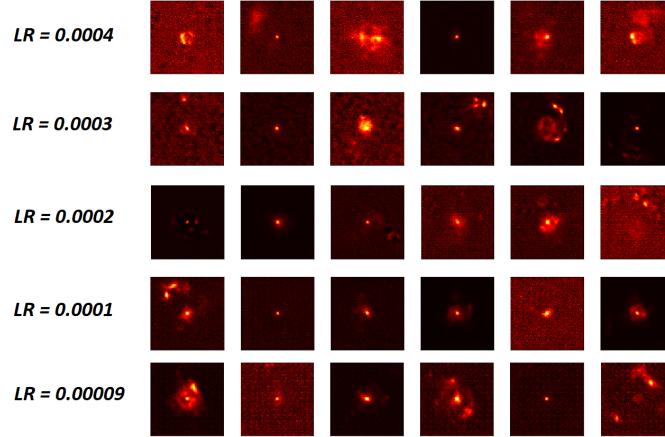


Figura 4.20: Imágenes generadas al cambiar eliminar un bloque convolucional del generador.

Analizando la calidad de las imágenes generadas, presentes en la figura 4.20, se observa que los anillos siguen presentes, pero no son tan notorios como los producidos por la primera arquitectura. Una posible hipótesis para esta situación es que el entrenamiento es más lento con esta versión de la arquitectura, esto reflejado en la demora en la estabilización del generador, por lo que con algunos *epochs* extras los anillos deberían definirse de mejor manera. Otra observación es que el efecto de grilla es más suave.

Se decide adoptar esta arquitectura considerando que la mejora en la estabilización del entrenamiento es lo suficientemente notoria para aceptar la leve disminución en la calidad de las imágenes para una cantidad pequeña de *epochs* de entrenamiento.

4.2.4. Resumen

En la tabla 4.2 se presenta un resumen de los resultados obtenidos para la definición de la arquitectura de la GAN.

Arquitectura	Efecto en entrenamiento	Efecto en imágenes	Conclusión
Arquitectura 1	Reducción de la varianza en la función de pérdida	Anillos más definidos	Se adopta la arquitectura
Arquitectura 2	Mejoras solo en algunas <i>learning rates</i>	Se mantiene la calidad	Se descarta arquitectura
Arquitectura 3	Mayor estabilidad	Leve disminución Suaviza el efecto de grilla	Se adopta la arquitectura

Tabla 4.2: Tabla resumen de los resultados para los experimentos de definición de la arquitectura de la red.

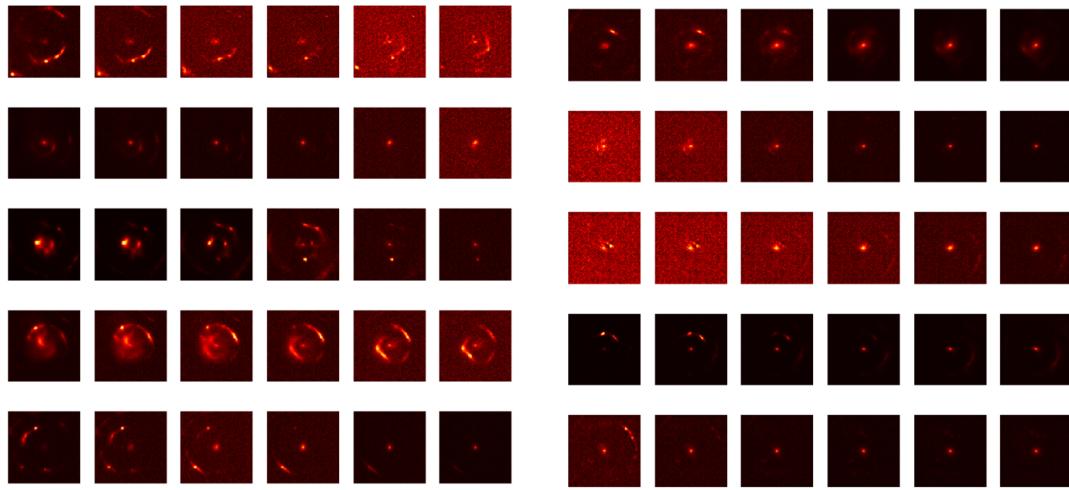
4.3. Validación

Tal como se detalló en el capítulo anterior, para realizar la validación de la solución se utilizan dos redes GAN entrenadas durante 100 epochs, generando así un dataset compuesto por 400 imágenes: 200 imágenes de lentes y 200 imágenes sin lentes. De las imágenes de lentes 100 fueron generadas utilizando las GANS.

El conjunto de imágenes fue etiquetado por el experto Umberto Rescigno, PhD en Física, Astrofísica y Física Aplicada de la Universidad de Milán, considerando 3 clases: **L**, indicando la presencia de lentes gravitacionales; **NL**, indicando que no existe presencia de lentes; y **L-NL** como clase intermedia para denotar incerteza. Las imágenes anotadas como **L-NL** son aquellas donde se detecta la presencia de un lente gravitacional, pero también poseen características que según la teoría no se deberían observar.

4.3.1. Overfitting

Para descartar la presencia de *overfitting* en las GAN utilizadas se realiza una interpolación *Slerp*. Para esto, se obtiene una muestra de dos puntos desde el *prior* para interpolar 6 puntos intermedios. Este proceso se repite 6 veces por cada red, dando origen a las figuras 4.21b y 4.21a. Cada fila corresponde a una interpolación entre puntos aleatorios, mientras que cada columna denota la cercanía a los puntos originales.



(a) Primera red GAN.

(b) Segunda red GAN

Figura 4.21: Interpolación entre 10 puntos aleatorios obtenidos aleatoriamente desde el espacio latente.

Se observa que en ambas redes se producen transiciones suaves, indicando que el mapeo aprendido por ambas redes es significativo. Por ejemplo, en la cuarta fila de la figura 4.21a se observa como se va definiendo el anillo a la vez que el ruido aumenta, mientras que en la primera fila de la figura 4.21b se observa como el tamaño del anillo se va achicando mientras que se reduce el ruido en la imagen.

Si bien, las transiciones suaves en la interpolación son una señal de que no existe overfitting, también se experimenta con las activaciones provenientes de la última capa oculta de la red Lensfinder; por cada imagen generada por GAN, se busca la imagen del conjunto de entrenamiento cuyo vector de activaciones se encuentre a menor distancia. El razonamiento del procedimiento anterior es el siguiente: si la GAN memoriza los datos de entrenamiento, los vectores de activación serán muy similares entre sí, por lo que al comparar la imagen sintética con la imagen del conjunto de entrenamiento, cuyo vector de activación se encuentra a menor distancia, se puede observar si la GAN memorizó ejemplos de entrenamiento o no.

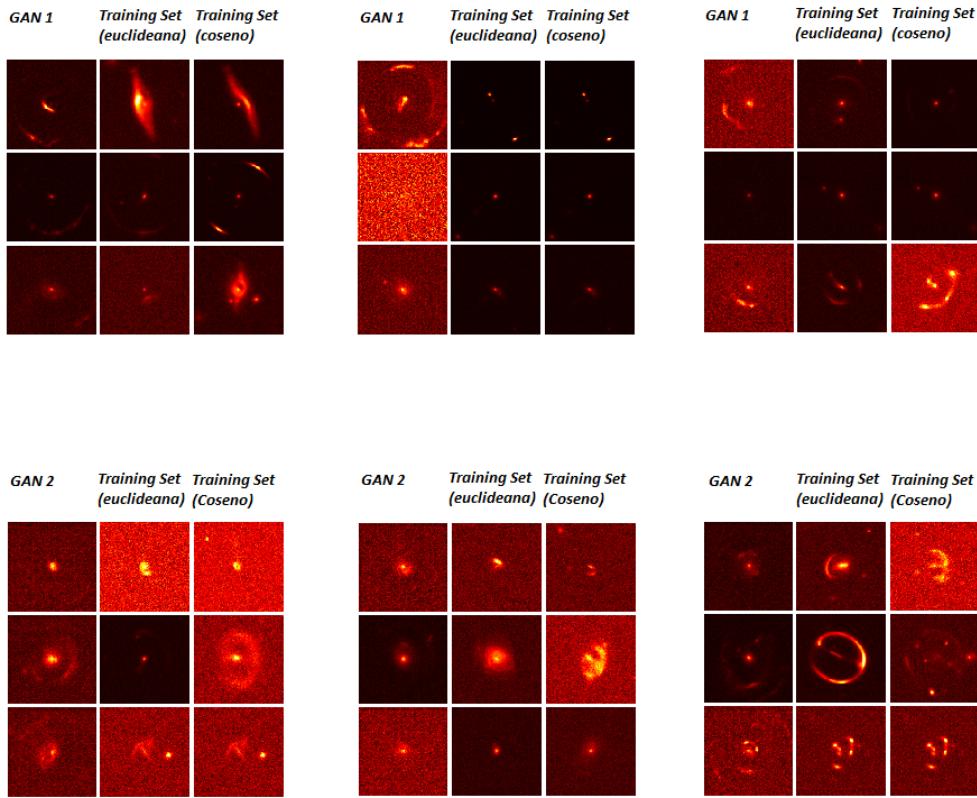


Figura 4.22: Imágenes generadas aleatorias junto a las imágenes más cercanas, según activaciones de la red Lensfinder, del conjunto de entrenamiento.

En la figura 4.22 se muestran 9 imágenes generadas aleatorias para cada GAN², en conjunto con la imagen del conjunto de entrenamiento a menor distancia según las activaciones de Lensfinder, esto considerando tanto distancia euclidiana como similitud de coseno.

Se puede apreciar que las diferencias entre las imágenes generadas, y las provenientes del conjunto de entrenamiento son notorias. Considerando éstas diferencias, sumado a la existencia de transiciones suaves presentes en las interpolaciones del espacio latente, se descarta que las GAN sufran de sobre-ajuste.

²La colección completa de imágenes para cada GAN se presentan en el apéndice 2 (capítulo 6).

4.3.2. Etiquetado manual

El resultado de las anotaciones entregadas por el experto se presenta en la tabla 4.3. En términos de la *accuracy* calculada para cada set de imágenes, se observa que el mejor desempeño alcanzado por el astrónomo se da al etiquetar las imágenes generadas con las GAN, logrando un 82 % de *accuracy*, indicando que la red logra generar imágenes con la suficiente calidad para engañar al experto. Para las imágenes obtenidas del dataset original, el etiquetado presenta un menor desempeño, con un 75 % para las imágenes con lentes y un 82 % para las imágenes sin presencia de este fenómeno.

	L	L-NL	NL	Accuracy
GAN 1	41	5	4	0,82
GAN 2	41	3	6	0,82
Con Lentes	75	14	11	0,75
Sin lentes	66	29	105	0,67

Tabla 4.3: Anotaciones entregadas por el experto en el experimento de validación.

El mejor desempeño del astrónomo en las imágenes generadas por las GAN podría sugerir una mayor compatibilidad entre las imágenes generadas por estos sistemas y aquello que un experto humano efectivamente seleccionaría en la práctica como un candidato para un estudio más profundo. Una hipótesis para la situación anterior es que la red no genera lentes imperfectos o improprios, los cuales generan más dudas y están presentes en el dataset original. Para indagar esta hipótesis, en la figura 4.23 se presentan las primeras 3 discrepancias en las etiquetas de imágenes de lentes.

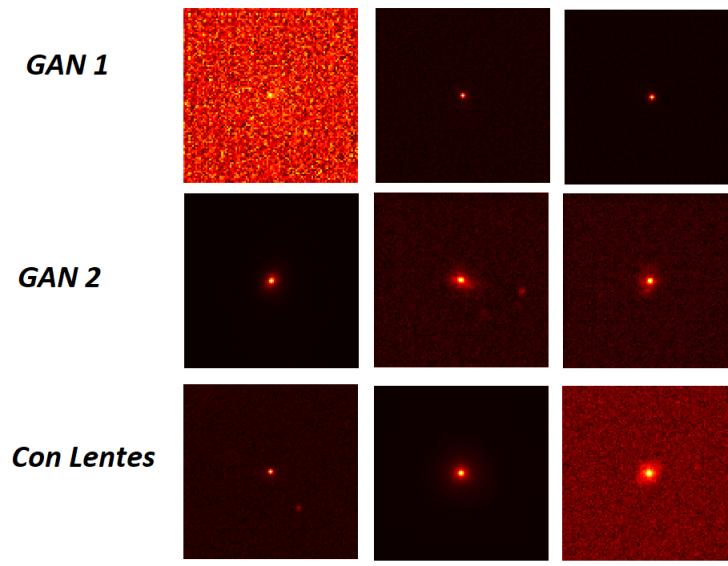


Figura 4.23: Imágenes de lentes gravitacionales donde la etiqueta del *challenge* difiere con la del astrónomo.

Se aprecia que las discrepancias se producen en imágenes similares en los tres set de datos presentados, indicando así que los generadores si logran aprender las características difíciles de detectar para el experto. En particular, las nueve imágenes seleccionadas consideran anillos prácticamente indetectables a simple vista.

En la figura 4.23 se presentan las primeras 9 imágenes sin lentes que fueron etiquetadas como lentes gravitacionales por el astrónomo. A simple vista no se observa mayor diferencia con las imágenes de la figura 4.23, mostrando lo difícil que es la detección de lentes mediante inspección visual en este tipo de casos.

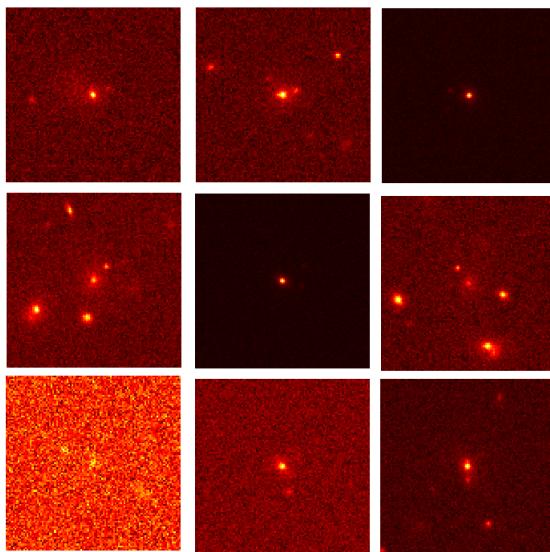


Figura 4.24: Imágenes sin lentes gravitacionales donde la etiqueta del *challenge* difiere con la del astrónomo..

La similitud entre las imágenes de las figuras 4.23 y 4.24 podría explicar la baja en el desempeño al clasificar el conjunto de datos sin lentes, dado que visualmente es más difícil diferenciar entre estos dos tipos de imágenes que cuando los anillos están completamente definidos.

Finalmente, los resultados para el proceso de etiquetación global se presentan en la matriz de confusión de la tabla 4.4³, a partir de la cual se concluye que la clasificación realizada por el astrónomo es sensible al tipo de imagen presentada, ya que es:

- Precisa en un 70,4 %. Es decir, que lo etiquetado como L por el astrónomo, en su mayoría, efectivamente corresponde a un lente.
- Sensible en un 78,5 %. Mostrando que, en su mayoría, los lentes son etiquetados correctamente.
- Específica en un 75,7 %. Lo que da a entender que, en gran proporción, lo etiquetado como NL efectivamente corresponde a imágenes sin lentes.

³Las imágenes etiquetadas como **L-NL** se consideran como **NL**.

		Etiqueta astrónomo	
		L	NL
Etiqueta real	L	157 (TP)	43 (FN)
	NL	66 (FP)	134 (TN)

Tabla 4.4: Matriz de confusión del proceso de etiquetado realizado por el experto.

Lo anterior permite descartar que el juicio del experto sea independiente de la imagen presentada, ya que existe diferencia con las probabilidades marginales de lente (50 %), no lente (50 %) y etiquetado como lente (56 %).

4.3.3. Etiquetado automático

El objetivo detrás de la etiquetación por parte de un experto es analizar si las imágenes generadas por la GAN tienen una calidad similar a las imágenes originales. Este mismo razonamiento se puede aplicar utilizando un clasificador automático entrenado para la detección de lentes gravitacionales, teniendo la ventaja de ser un proceso más rápido y con mejor desempeño [Metcalf *et al.*, 2018]⁴. En este procedimiento, se predice la presencia de lentes utilizando la réplica desarrollada de la red *Lensfinder*, ganadora del *challenge*, cuyos resultados se presentan en la tabla 4.5. Como es de esperar, la *accuracy* alcanzada está por sobre el 90 % en los 4 conjuntos de datos, siendo similar entre sí.

	L	NL	Accuracy
GAN 1	45	5	0,90
GAN 2	47	3	0,94
Con Lentes	92	8	0,92
Sin lentes	13	187	0,94

Tabla 4.5: Anotaciones entregadas por la red *Lensfinder*.

Considerando que el desempeño obtenido para las GANS es similar al desempeño sobre

⁴Uno de los participantes del challenge utilizó inspección visual, sin embargo, la gran mayoría de los métodos automáticos obtuvieron mejor desempeño.

los lentes del *challenge*, sumado a los resultados obtenidos mediante la etiquetación con el experto, se concluye que la calidad de la imágenes es lo suficientemente buena para engañar tanto a la red como al astrónomo, o al menos lo suficientemente buena como para que la red extraiga *features* que permitan su detección como lentes.

En las figuras 4.25b, 4.25a y 4.25c, se presentan histogramas para las predicciones realizadas por Lensfinder sobre los conjuntos de datos con presencia de lentes gravitacionales.

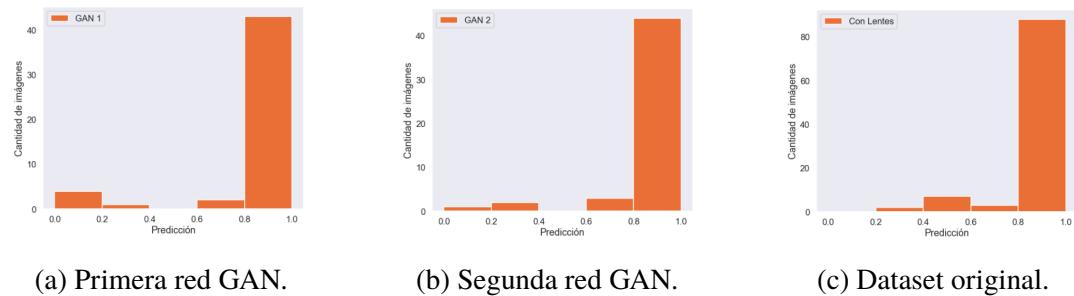


Figura 4.25: Histograma de las predicciones de la red Lensfinder para los 3 conjuntos de datos con lentes del experimento de validación.

Como la función de activación de la capa de salida es sigmoidal, las predicciones de la red se pueden interpretar como la probabilidad de que una imagen tenga presencia de lentes gravitacionales, por lo que los histogramas deberían ser similares entre las imágenes de LG. Considerando lo anterior, y observando las diferencias en las colas de los histogramas, se desprende que aún hay posibilidad de mejora a pesar de lograr generar lentes de calidad.

4.3.4. FID modificado

Modificando la distancia *Frechét-Inception* para obtener las *features* desde Lensfinder, se calcula esta métrica para ambos generadores, obteniéndose un FID de 72,34 y 69,49 para las GAN 1 y 2 respectivamente. La idea de utilizar dos GANS para el experimento era comprobar si esta modificación del FID se correlaciona con el juicio del astrónomo. Sin embargo, tal como se presenta en la tabla 4.3, el desempeño obtenido por el experto es muy similar en ambas redes, por lo que no es posible obtener una conclusión.

Considerando lo anterior, se entrena una nueva GAN con la misma arquitectura durante 5 epochs. Al someterse a un entrenamiento breve se espera que las imágenes generadas sean de baja calidad, obteniendo así un FID de $6,4 * 10^{19}$. Al utilizar una GAN entrenada durante 50 epochs el FID baja a 74,34. La figura 4.26 presenta imágenes de lentes generados con los generadores mencionados anteriormente.

De la misma manera, se calcula el FID con el subconjunto de imágenes del *challenge* (con y sin lentes), ya que éstas cuentan con la misma calidad de imagen, cuyo FID es de 180.16, mostrando que las imágenes generadas por GAN son más cercanas a las imágenes con lentes, lo que se condice con el juicio del experto.

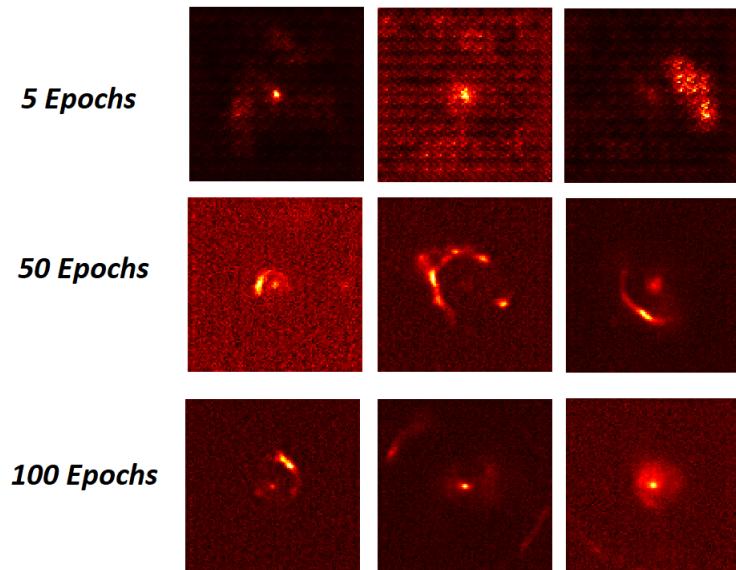


Figura 4.26: Imágenes generadas vía GAN para distintos epochs.

Por otro lado, se calcula esta distancia tanto para las imágenes generadas por GAN, donde el juicio del astrónomo es igual al del *challenge*, y para aquellas donde hay discrepancia en las etiquetas: si el FID se correlaciona con el juicio del experto es de esperar que el primero sea más bajo que el segundo. El FID obtenido para las imágenes con discrepancia es de 84,95, mientras que para las imágenes con igual etiqueta se obtiene un FID de 68,66, mostrando nuevamente su relación con el juicio experto. Si bien, el experimento anterior muestra una correlación con el juicio humano, es necesario realizar un experimento a mayor escala para corroborar esto.

Finalmente, debido a que este trabajo es el primero en generar imágenes de lentes gravitacionales fuertes utilizando GANS, no existe un *benchmark* para comparar los resultados obtenidos. A diferencia del *Inception Score*, donde es posible calcular el score para los datos reales, para el FID se requieren dos *datasets* para su cálculo, por lo que no es posible realizar este ejercicio. Sin embargo, es posible utilizar los datos del conjunto de entrenamiento y de pruebas del *challenge*, lo que permite obtener un valor “para una buena generación”.

Al realizar el cálculo anterior se obtiene un FID de 13,62. De éste resultado se concluye que, a pesar de que las imágenes generadas presentan característica de lentes gravitacionales que permiten engañar a un experto y a una red convolución detectora especializada, aún hay espacio para su mejora mediante el uso de técnicas más avanzadas en la literatura de GANS.

4.3.5. Subclasificación de lentes

Durante el segundo experimento de validación, el experto clasificó las imágenes detectadas como lentes en las siguientes categorías: un anillo (UA), múltiples anillos (MA), *quassarlike* (QL), insual (IN), y otros (O). Los resultados de esta subclasificación se presentan en la tabla 4.6 y la figura 4.27.

	UA	MA	QL	IN	O
GAN 1 (L)	46,34 % (19)	17,07 % (7)	2,44 % (1)	2,44 % (1)	31,71 % (13)
GAN 2 (L)	34,14 % (14)	14,63 % (6)	9,76 % (4)	0 % (0)	41,46 % (17)
Con lentes (L)	41,33 % (31)	0 % (0)	8 % (6)	9,33 % (7)	41,33 % (31)
Sin lentes (L)	0 % (0)	0 % (0)	21,21 % (14)	1,52 % (1)	77,27 % (51)

Tabla 4.6: Proporción de lentes por cada subclase, para los datos previamente etiquetados como L.

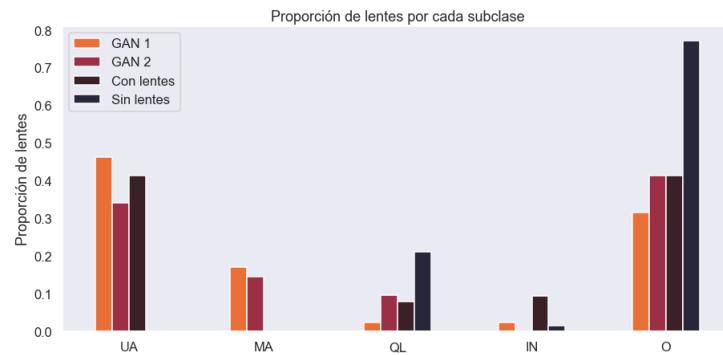


Figura 4.27: Proporción de lentes por cada subclase, para los datos previamente etiquetados como L.

Uno de los resultados más llamativos es que la presencia de lentes con múltiples anillos solo está presente en las imágenes generadas por GAN, siendo un fenómeno que no se presenta en el conjunto de imágenes originales. Una hipótesis para esta situación es que las imágenes con múltiples anillos son generadas al utilizar puntos intermedios, en el espacio latente, entre dos imágenes con un anillo. Para indagar esta hipótesis se generan imágenes con un anillo, para luego interpolar 6 puntos intermedios entre ellas, cuyo resultado se presentan en la figura 4.28.

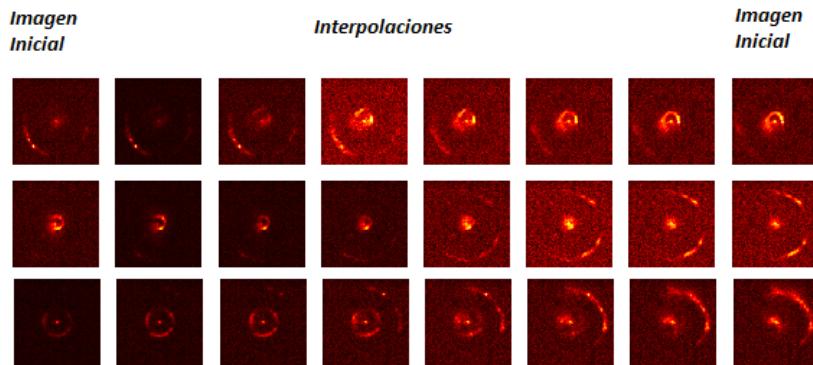


Figura 4.28: 3 interpolaciones entre imágenes con un anillo.

La imagen anterior muestra la generación de imágenes con múltiples anillos entre dos imágenes UA. Esta situación hace patente la necesidad de realizar, a futuro, un estudio profundo del espacio latente, con el objetivo de buscar mejores maneras de muestrear puntos para así solo generar lentes UA. Cabe destacar, que si bien el fenómeno no está

presente en el conjunto original, si es posible encontrar este tipo de lentes, pero se requiere de un lente masivo que envuelve la luz de dos galaxias perfectamente alineadas.

Para las imágenes de lentes con un anillo, solo aparecen para los sistemas que generan lentes gravitacionales, lo que indica que los lentes con esta característica son más afines con lo que un experto seleccionaría para un estudio. Con respecto a la comparación entre las GAN y los datos originales, se observa que la GAN 1 presenta una sobre representación de un 7,19 %, mientras que la GAN 2 tiene una subrepresentación de este tipo de lentes de un 5,01 %, siendo ambas cercanas al conjunto de datos originales.

Con respecto a la clase QL, está presente mayoritariamente en las imágenes sin lentes. Para indagar esta situación, en la figura 4.29 se presentan las imágenes de lentes etiquetadas como QL, como también las primeras 6 imágenes sin lentes etiquetadas de la misma manera.

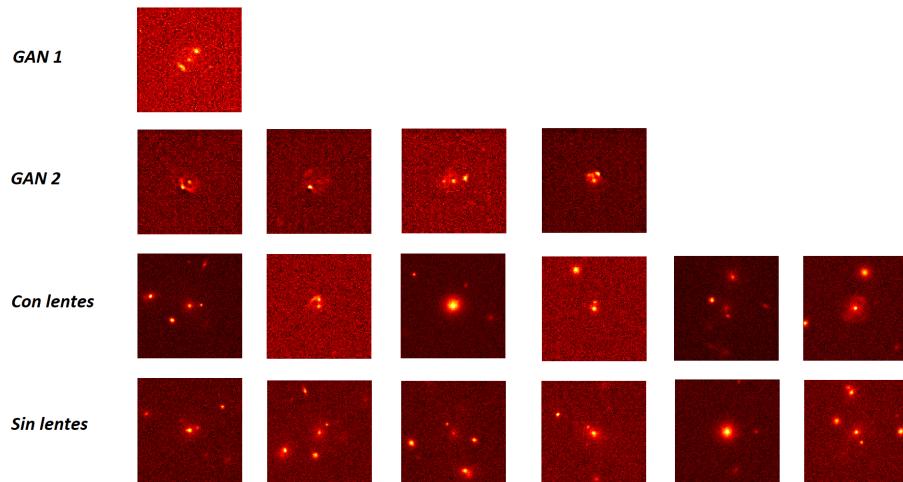


Figura 4.29: Imágenes de lentes etiquetadas como QL, en conjunto a las primeras 6 imágenes NL etiquetadas de la misma manera.

Con respecto a las imágenes generadas por GAN, da la impresión de que son similares a imágenes con anillos incompletos más que a las imágenes QL presentes en el conjunto original, salvo la segunda imagen del conjunto de lentes originales. La baja representación de este tipo de lentes en el conjunto original (8 %) puede explicar el que las GAN no logren generar imágenes de este tipo de la misma calidad.

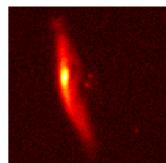
La misma situación anterior se da para la clase IN, donde solo la GAN 1 logra generar un lente de este tipo, indicando que la GAN tiene problemas para la generación de este tipo de imágenes. Para remediar esta situación se podrían utilizar transformaciones para obtener más ejemplos de este tipo de imágenes.

Con respecto a la clase O, la GAN 2 muestra casi la misma proporción que los lentes originales, mientras que en la GAN 1 esta clase está sub-representada. Sin embargo, las proporciones son similares, por lo que las GAN aprendieron a generar este tipo de lentes.

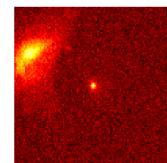
Por otro lado, los resultados muestran que la GAN 2 se encuentra más cercana a la proporción de lentes del conjunto original, lo que se correlaciona con las diferencias entre el FID de ambas GANS.

4.3.6. Descripción cualitativa de L-NL y NL

Con respecto a las características de los lentes clasificados como L-NL, el experto indica que estas presentan características inusuales, no existiendo conexión entre las imágenes provenientes de las GAN. La situación anterior podría explicar el hecho de la no presencia de lentes inusuales en el experimento de subclasiificación. En el caso de las imágenes del conjunto de lentes original, también se presentan casos como los presentes para las GAN, sin embargo, existen casos aún más inusuales, los cuales se presentan en las figuras 4.30a y 4.30b.



(a) Primer caso.



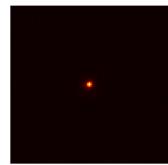
(b) Segundo caso.

Figura 4.30: Imágenes clasificadas como L-NL con características muy inusuales, provenientes del conjunto de lentes originales.

En el caso de la primera figura, se presenta un caso de un lente central no luminoso, que presenta un arco extendido sin forma circular, lo que se asemeja a un “arco recto”, la cual es una característica muy rara que solo es posible en eventos de *clusters* de galaxias.

Con respecto a la segunda figura, presenta un arco muy luminoso con un ancho no despreciable, las cuales son características que no son encontradas en conjunto. Esto indica que en el conjunto de lentes originales existen estos casos asociados a otros tipos de eventos, los que podrían provocar que la GAN aprenda configuraciones no deseadas.

Para los lentes clasificados como NL, el experto no detecta características en el fondo, lo que podría significar que el efecto del lente se encuentra muy cercano a éste. Sin embargo, no se detectan asimetrías al nivel del objeto central, o se detectan al nivel resolución de 2 o 3 píxeles, lo que no es suficiente al encontrarse en la escala del error del telescopio utilizado como referencia. Ejemplos de esta situación se presentan en las figuras 4.31a y 4.31b.



(a) Ejemplo GAN 1.



(b) Ejemplo GAN 2.

Figura 4.31: Imágenes generadas por GAN clasificadas como NL, donde la resolución no permite su detección como lente.

En el conjunto de imágenes de lentes originales también se cuenta con ejemplos donde la resolución de las imágenes no permite su clasificación como lentes, por lo que es de esperar que la GAN haya aprendido a generar dichas imágenes. En la figura 4.32 se presentan ejemplos de esta situación.



Figura 4.32: Imágenes del conjunto de lentes original clasificadas como NL, donde la resolución no permite su detección como lente.

4.3.7. Resumen

- Con respecto al *overfitting*, la presencia de transiciones suaves entre los puntos interpolados, sumado a las diferencias de las imágenes generadas y las imágenes del conjunto de entrenamiento, según las activaciones de la última capa oculta de Lensfinder, se descarta la presencia de overfitting en el generador de la GAN.
- La etiquetación vía experto y la etiquetación automática muestran que los lentes generados por las GANS tienen la calidad suficiente para engañar a ambos anotadores. Las predicciones de la red Lensfinder indican que existe posibilidad de mejora en la calidad de las imágenes generadas.
- La etiquetación del astrónomo no permite comparar la calidad de las redes mediante el uso del FID. Mediante el uso de una GAN con poco entrenamiento, y la comparación entre los lentes generados etiquetados como L y NL⁵, se obtiene que esta modificación se correlaciona con el juicio humano, pero se requiere un experimento de mayor escala para su corroboración.
- La subclasificación entre las imágenes clasificadas como lentes por el astrónomo muestra que las GAN generan imágenes de múltiples anillos que no están presentes en el conjunto original, las cuales se obtienen a partir de puntos intermedios entre dos imágenes con un anillo. Con respecto al resto de las clases, las GAN aprenden a generar lentes del tipo UA y O, mientras que para las clases QL y IN no.
- El análisis cualitativo de las imágenes, donde existe discrepancia entre las etiquetas del *challenge* y el experto, muestra que en el conjunto de datos originales existen imágenes de lentes con características no observables en eventos galaxia-galaxia, como también la existencia de imágenes de lentes donde la evidencia se presenta en una resolución cercana al error del telescopio de referencia, por lo que no es posible su clasificación como lente.

⁵Los lentes clasificados como **L-NL** se consideran como **NL**.

Conclusiones

En este capítulo se presentan tanto las conclusiones del trabajo realizado, como también el trabajo futuro. Las conclusiones se organizan en función de los objetivos propuestos.

4.4. Entrenamiento estable

Con respecto al primer objetivo específico, se experimentó con estrategias de entrenamiento, manteniendo una arquitectura base, y con variaciones de arquitecturas. En ambos casos se experimentó con diversas learning rates.

Los resultados para las estrategias de entrenamiento muestran que el uso de *one-side label smoothing* y *noisy labels* mejoran la estabilidad del entrenamiento. En particular, *one-side label smoothing* permite que el entrenamiento no diverja para todas las tasas de aprendizaje utilizadas para la experimentación, lo que sucedía al utilizar tasas de 0,0002 y 0,00009 al no utilizar esta técnica, como también la reducción en la varianza en las funciones de pérdida, tanto en el generador como en el discriminador, y mejora visual en la calidad de las imágenes generadas. Con respecto a *noisy labels*, se mantiene la calidad de imagen obtenida con la técnica anterior, reduciendo levemente la varianza en las funciones de pérdidas, reduciendo además los *peaks* de ésta. También se experimentó con la inicialización de pesos utilizada en la DCGAN original, distribución normal con media 0 y varianza 0,2, la cual disminuye la varianza en las funciones de pérdida, pero a la vez perjudica la calidad de las imágenes.

Por otro lado, para definir la arquitectura de la red se modifica la arquitectura base,

utilizada para la definición de las estrategias de entrenamiento⁶, considerando 3 tipos de cambios: el primero, aumentar y disminuir el número de filtros en potencias de dos, tal como se acostumbra en modelos clásicos de redes convolucionales; el segundo, la disminución de un bloque convolucional en el generador, de tal manera que el discriminar sea más profundo; el tercero, el uso de una capa convolucional adicional en los bloques convolucionales del discriminador, tal como se acostumbra en arquitecturas modernas de redes convolucionales.

Considerando el cambio en el número de filtros utilizados, un primer cambio es la disminución de parámetros del modelo, lo que lleva a que el tiempo de entrenamiento disminuya a la mitad, apreciando también una reducción en la varianza de la función de pérdida, y la generación de imágenes con anillos más definidos con un leve efecto de grilla, adoptando este cambio arquitectural. Con respecto a la disminución de un bloque convolucional en el generador, se observan mejoras en la estabilidad solo para ciertas *learning rates*, lo que sumado al aumento de los parámetros del modelo, y la mantención de la calidad de las imágenes, se decide no adoptar este cambio. Finalmente, para el último cambio, se obtiene una mayor estabilidad del entrenamiento a costa de una leve disminución en la calidad de las imágenes, por lo que se adopta este último cambio.

Los experimentos anteriores llevan a la definición de la arquitectura de la figura 4.20 (primer apéndice), la cual, en base a los resultados obtenidos, se considera una arquitectura estable.

4.5. Diseño para la evaluación de resultados

Considerando el segundo objetivo, se diseñan dos enfoques para evaluar cuantitativamente las imágenes generadas por GAN. Para el primer enfoque se busca determinar si las imágenes generadas logran “engaños” a un etiquetador experto, determinando si las imágenes generadas son consideradas como lentes gravitacionales. En el segundo enfoque, se busca comparar las imágenes generadas con el conjunto de imágenes original.

Considerando el primer enfoque, se diseña un experimento donde se mide la *accuracy* del etiquetador, ya sea un astrónomo experto o una red convolucional especializada en la

⁶Figura 5.1 primer apéndice.

detección de lentes. Para esto se considera un dataset compuesto por 400 imágenes, 200 sin lentes gravitacionales y 200 con lentes gravitacionales, de las cuales la mitad está compuesta por imágenes generadas por 2 redes GAN.

Para el segundo enfoque, se modifica la Distancia Frechét Inception, métrica clásica en la evaluación de GANS que permite la comparación entre dos conjuntos de imágenes, considerando una réplica de la red Lensfinder para su adaptación al dominio de lentes gravitacionales.

4.6. Evaluación de resultados

Con respecto a la evaluación de resultados, el primer enfoque diseñado en el mecanismo cuantitativo muestra que las imágenes generadas por GAN logran generar lentes gravitacionales: para la etiquetación por parte de un astrónomo experto se obtiene una *accuracy* de 0,82 para las imágenes generadas por GAN, mientras que para las imágenes de lentes originales se obtiene una *accuracy* de 0,75, sugiriendo que las imágenes generadas por la GAN tienen una mayor compatibilidad con aquello que un experto seleccionaría para un estudio más profundo. Por otro lado, se analizan las discrepancias en la etiquetación del astrónomo con la etiqueta propuesta en el *challenge*, mostrando que las imágenes generadas por la GAN donde se produce esta situación son muy similares a las imágenes originales. Finalmente, la precisión, sensibilidad, y especificidad del proceso global de etiquetación muestran que la clasificación realizada es sensible al tipo de imagen presentada, descartando una independencia del juicio frente a la imagen.

Para la etiquetación por parte de una red detectora especializada, se obtiene una *accuracy* de 0,90 y 0,94 en las imágenes generadas por GAN, mientras que una *accuracy* de 0,92 en los lentes originales, lo que sumado a los resultados de la etiquetación vía experto, permite concluir que la calidad de las imágenes es lo suficientemente buena como para engañar tanto a la red como al astrónomo.

Con respecto a la modificación del FID utilizada, se obtienen FID de 72,34 y 69,49 para las GAN utilizadas en la generación de las imágenes. Como los resultados obtenidos en el enfoque anterior no permiten la comparación con estos resultados, se realizan 2 procedimientos para ver si esta métrica se correlaciona con el juicio humano: en el

primero, se calcula esta métrica para GAN entrenadas durante 5 y 50 epoch, donde la primera entrega imágenes de menor calidad mientras que la segunda entrega imágenes con calidad similar, obteniéndose un FID de $6,4 * 10^{19}$ y 74,34; en el segundo, se compara el FID entre las imágenes originales de lentes y no lentes, obteniéndose un FID de 180,16, lo cual muestra que las imágenes generadas por GAN son más cercanas a las imágenes de lentes que aquellas que no lo poseen, lo que se condice con el juicio del experto. Si bien, estos procedimientos presentan evidencia de la correlación de esta métrica con el juicio experto, es necesario realizar un experimento a mayor escala para su corroboración.

Como este trabajo es el primero en generar lentes gravitacionales fuertes mediante la utilización de GAN, no existe *benchmark* para la comparación de los resultados para este tipo de FID. Dado lo anterior, se calcula el FID para las imágenes de lentes del conjunto de entrenamiento y el conjunto de pruebas, logrando un FID de 13,62, por lo que se concluye que, si bien las imágenes generadas permiten engañar a etiquetadores expertos (astrónomo y red detectora), existe espacio de mejora en la calidad de las imágenes.

Con respecto a la subclasiﬁcación de imágenes clasificadas como lentes, se concluye que la GAN logra aprender a generar lentes del tipo UA y O, generando también lentes con múltiples anillos, los cuales no están presentes en el conjunto de imágenes reales, siendo provocados por puntos intermedios, en el espacio latente, entre dos imágenes con un anillo. Con respecto a las clases QL y IN, la GAN no aprendió a generarlas, probablemente debido a la poca presencia de éstas en el dataset original.

Finalmente, mediante un análisis cualitativo de las imágenes clasificadas como L-NL y NL por parte del experto, se encuentran ejemplos de lentes en el conjunto original no observables en los eventos galaxia-galaxia, lo que puede provocar que la GAN aprenda tipos de lentes no deseados. También se encuentran imágenes de lentes donde la resolución de la evidencia está al nivel del error del telescopio de referencia, por lo que no es posible su clasificación como lentes. Esta última situación se encuentra tanto para las imágenes generadas por GAN como para los lentes del dataset original.

4.7. Trabajo futuro

Como trabajo futuro se plantea lo siguiente:

- Experimentar con técnicas más avanzadas en la literatura de GAN, de tal manera de mejorar el FID obtenido para las imágenes generadas con respecto al FID entre lentes de entrenamiento y pruebas. Esto mediante la utilización de técnicas avanzadas de entrenamiento, o la utilización de arquitecturas más modernas de GANS.
- La realización de un experimento de mayor escala para corroborar la correlación del FID con el juicio humano. Este debe considerar una mayor cantidad de imágenes, provenientes de distintos tipos de GAN, y en lo posible la participación de más de un experto en la etiquetación.
- Experimentar con el espacio latente de la GAN, para así obtener muestras desde este espacio de manera más inteligente, y así evitar la generación de imágenes con múltiples anillos.
- Evaluar la utilidad de la GAN como método de aumento de datos en el entrenamiento de métodos automáticos de detección de lentes gravitacionales.

Referencias

- [Abadi *et al.*, 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., y Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- [Arpit *et al.*, 2016] Arpit, D., Zhou, Y., Kota, B. U., y Govindaraju, V. (2016). Normalization propagation: A parametric technique for removing internal covariate shift in deep networks.
- [Barratt y Sharma, 2018] Barratt, S. y Sharma, R. (2018). A note on the inception score.
- [Boylan-Kolchin *et al.*, 2009] Boylan-Kolchin, M., Springel, V., White, S. D. M., Jenkins, A., y Lemson, G. (2009). Resolving cosmic structure formation with the Millennium-II Simulation. *Monthly Notices of the Royal Astronomical Society*, 398(3):1150–1164.
- [Brownlee, 2019] Brownlee, J. (2019). How to develop a gan to generate cifar10 small color photographs. <https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-a-cifar-10-small-object-photographs-from-scratch/>. Accessed: 2020-06-15.
- [Bullock y Boylan-Kolchin, 2017] Bullock, J. S. y Boylan-Kolchin, M. (2017). Small-scale challenges to the λ cdm paradigm. *Annual Review of Astronomy and Astrophysics*, 55(1):343–387.

- [Chollet, 2015] Chollet, F. (2015). Keras. <https://keras.io>.
- [Coates y Ng, 2012] Coates, A. y Ng, A. Y. (2012). Learning feature representations with k-means. En *Neural networks: Tricks of the trade*, páginas 561–580. Springer.
- [Deng *et al.*, 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. En *CVPR09*.
- [Dosovitskiy *et al.*, 2015] Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., y Brox, T. (2015). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747.
- [Geiger, 2017] Geiger, M. (2017). lensfinder-euclid. <https://github.com/mariogeiger/lensfinder-euclid> (2020).
- [Glorot y Bengio, 2010] Glorot, X. y Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. volumen 9 de *Proceedings of Machine Learning Research*, páginas 249–256, Chia Laguna Resort, Sardinia, Italy. JMLR Workshop and Conference Proceedings.
- [Goodfellow *et al.*, 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., y Bengio, Y. (2014). Generative adversarial nets. En Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., y Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 27*, páginas 2672–2680. Curran Associates, Inc.
- [Grillo *et al.*, 2016] Grillo, C., Karman, W., Suyu, S. H., Rosati, P., Balestra, I., Mercurio, A., Lombardi, M., Treu, T., Caminha, G. B., Halkola, A., y et al. (2016). The story of supernova “refsdal” told by muse. *The Astrophysical Journal*, 822(2):78.
- [Guo *et al.*, 2011] Guo, Q., White, S., Boylan-Kolchin, M., De Lucia, G., Kauffmann, G., Lemson, G., Li, C., Springel, V., y Weinmann, S. (2011). From dwarf spheroidals to cD galaxies: simulating the galaxy population in a Λ CDM cosmology. *Monthly Notices of the Royal Astronomical Society*, 413(1):101–131.
- [Habib *et al.*, 2016] Habib, S., Pope, A., Finkel, H., Frontiere, N., Heitmann, K., Daniel, D., Fasel, P., Morozov, V., Zagaris, G., Peterka, T., y et al. (2016). Hacc: Simulating sky surveys on state-of-the-art supercomputing architectures. *New Astronomy*, 42:49–65.

- [He *et al.*, 2015] He, K., Zhang, X., Ren, S., y Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.
- [Heusel *et al.*, 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., y Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500.
- [Ioffe y Szegedy, 2015] Ioffe, S. y Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.
- [Kingma y Ba, 2014] Kingma, D. P. y Ba, J. (2014). Adam: A method for stochastic optimization.
- [Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.
- [Krizhevsky *et al.*, 2012] Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. En *Advances in neural information processing systems*, páginas 1097–1105.
- [Lanusse *et al.*, 2017] Lanusse, F., Ma, Q., Li, N., Collett, T. E., Li, C.-L., Ravanbakhsh, S., Mandelbaum, R., y Póczos, B. (2017). Cmu deeplens: deep learning for automatic image-based galaxy–galaxy strong lens finding. *Monthly Notices of the Royal Astronomical Society*, 473(3):3895–3906.
- [Li *et al.*, 2016] Li, N., Gladders, M. D., Rangel, E. M., Florian, M. K., Bleem, L. E., Heitmann, K., Habib, S., y Fasel, P. (2016). Pics: Simulations of strong gravitational lensing in galaxy clusters. *The Astrophysical Journal*, 828(1):54.
- [Meneghetti *et al.*, 2008] Meneghetti, M., Melchior, P., Grazian, A., De Lucia, G., Dolag, K., Bartelmann, M., Heymans, C., Moscardini, L., y Radovich, M. (2008). Realistic simulations of gravitational lensing by galaxy clusters: extracting arc parameters from mock dune images. *Astronomy & Astrophysics*, 482(2):403–418.
- [Metcalf *et al.*, 2018] Metcalf, R., Meneghetti, M., Avestruz, C., Bellagamba, F., Bom, C., Bertin, E., Cabanac, R., Decencière, E., Flamary, R., Gavazzi, R., Geiger, M., Hartley, P., Huertas-Company, M., Jackson, N., Jullo, E., Kneib, J.-P., Koopmans, L., Lanusse, F., Li, C.-L., y Vernardos, G. (2018). The strong gravitational lens finding challenge. *Astronomy & Astrophysics*.

- [Metcalf y Petkova, 2014] Metcalf, R. B. y Petkova, M. (2014). GLAMER - I. A code for gravitational lensing simulations with adaptive mesh refinement. *Monthly Notices of the Royal Astronomical Society*, 445(2):1942–1953.
- [Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill.
- [Mustafa *et al.*, 2019] Mustafa, M., Bard, D., Bhimji, W., Lukić, Z., Al-Rfou, R., y Kratochvil, J. M. (2019). Cosmogan: creating high-fidelity weak lensing convergence maps using generative adversarial networks. *Computational Astrophysics and Cosmology*, 6(1).
- [Navarro *et al.*, 1996] Navarro, J. F., Frenk, C. S., y White, S. D. M. (1996). The Structure of Cold Dark Matter Halos. *The Astrophysical Journal*, 462:563.
- [Odena *et al.*, 2016] Odena, A., Dumoulin, V., y Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*.
- [Petkova *et al.*, 2014] Petkova, M., Metcalf, R. B., y Giocoli, C. (2014). GLAMER - II. Multiple-plane gravitational lensing. *Monthly Notices of the Royal Astronomical Society*, 445(2):1954–1966.
- [Radford *et al.*, 2015] Radford, A., Metz, L., y Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.
- [Ravanbakhsh *et al.*, 2016] Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J., y Poczos, B. (2016). Enabling dark energy science with deep generative models of galaxy images.
- [Rizzo *et al.*, 2020] Rizzo, F., Vegetti, S., Powell, D., Fraternali, F., McKean, J. P., Stacey, H. R., y White, S. D. M. (2020). A dynamically cold disk galaxy in the early universe. *Nature*, 584(7820):201–204.
- [Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- [Salimans *et al.*, 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., y Chen, X. (2016). Improved techniques for training gans. En *Advances in neural information processing systems*, páginas 2234–2242.

- [Schaap y van de Weygaert, 2000] Schaap, W. E. y van de Weygaert, R. (2000). Continuous fields and discrete samples: reconstruction through Delaunay tessellations. *Astronomy and Astrophysics*, 363:L29–L32.
- [Schaefer *et al.*, 2018] Schaefer, C., Geiger, M., Kuntzer, T., y Kneib, J.-P. (2018). Deep convolutional neural networks as strong gravitational lens detectors. *Astronomy & Astrophysics*, 611:A2.
- [Shoemake, 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254.
- [Simonyan y Zisserman, 2014] Simonyan, K. y Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Springenberg *et al.*, 2014] Springenberg, J., Dosovitskiy, A., Brox, T., y Riedmiller, M. (2014). Striving for simplicity: The all convolutional net.
- [Szegedy *et al.*, 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., y Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.
- [White, 2016] White, T. (2016). Sampling generative networks.
- [Yu *et al.*, 2015] Yu, F., Zhang, Y., Song, S., Seff, A., y Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Capítulo 5

Primer Apéndice

5.1. Arquitecturas

5.1.1. Arquitectura base

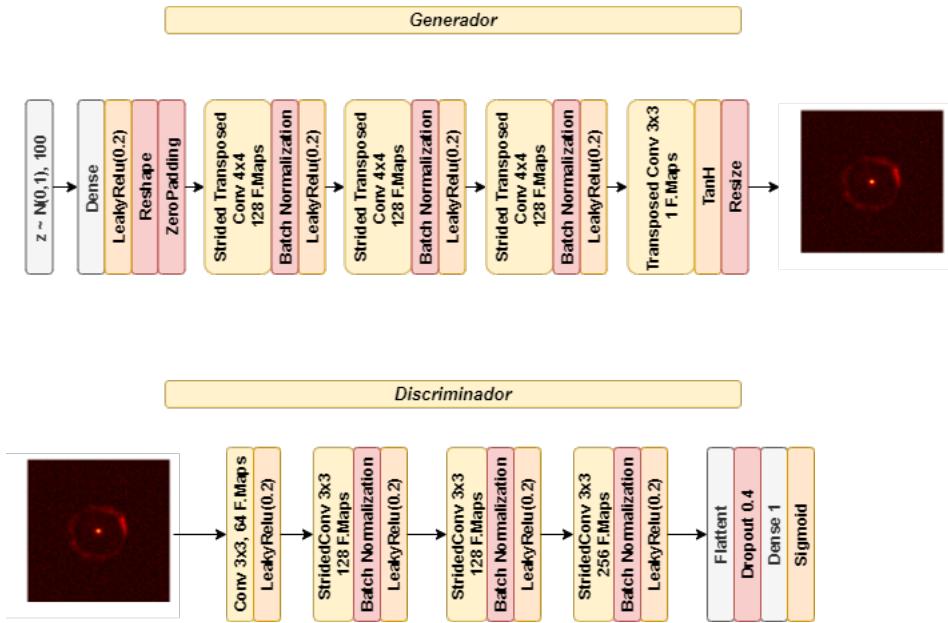


Figura 5.1: Arquitectura utilizada como base para los experimentos. Diseñada para ser utilizada sobre Cifar-10 [Brownlee, 2019].

5.1.2. Arquitectura experimentos

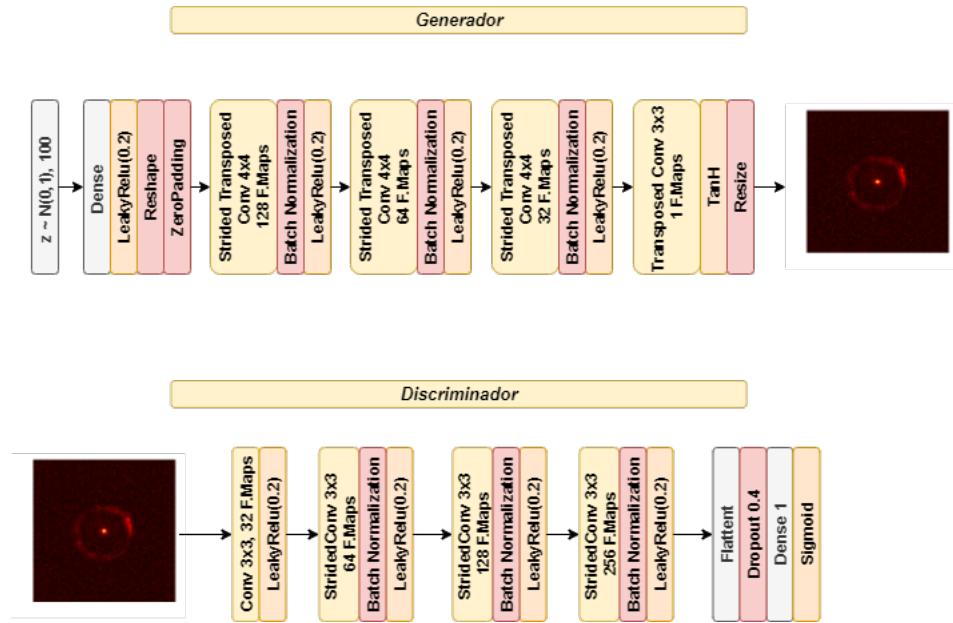


Figura 5.2: Arquitectura utilizada en el primer experimento de definición de arquitectura.

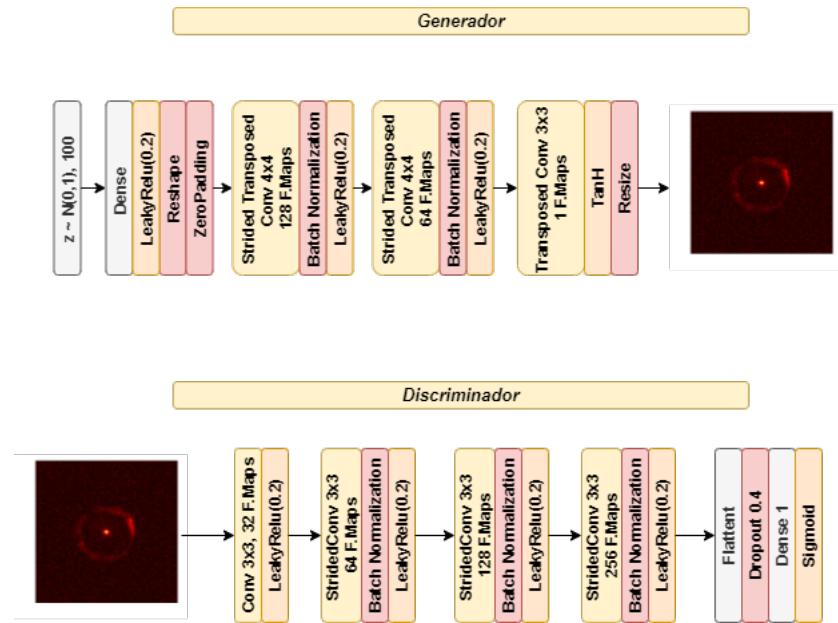


Figura 5.3: Arquitectura utilizada en el segundo experimento de definición de arquitectura.

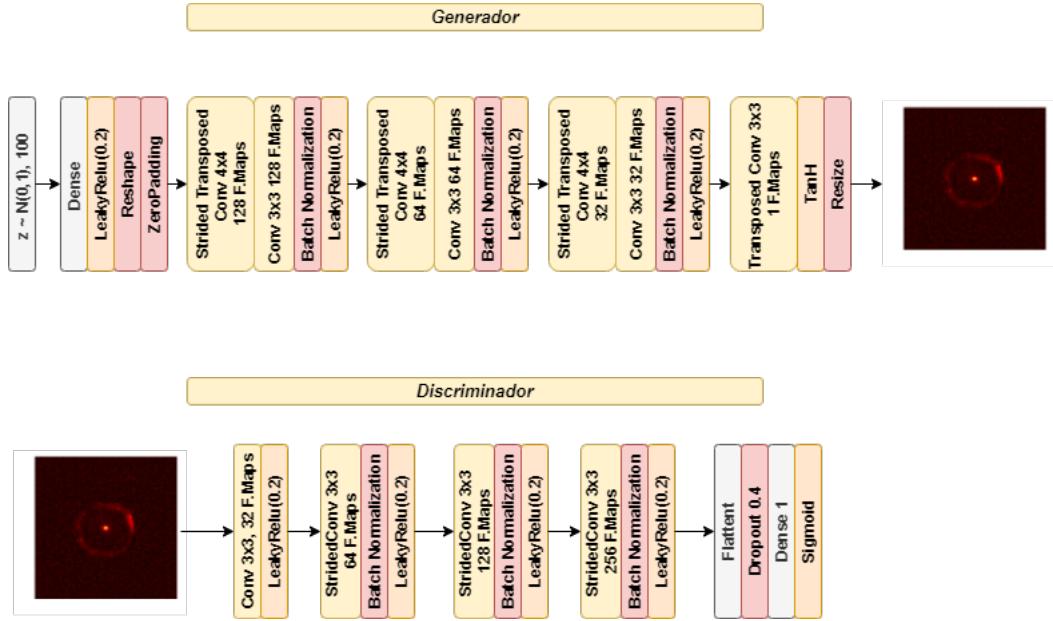


Figura 5.4: Arquitectura utilizada en el tercer experimento de definición de arquitectura. Corresponde a la arquitectura seleccionada.

Capítulo 6

Segundo Apéndice

6.1. Resultados

6.1.1. Overfitting

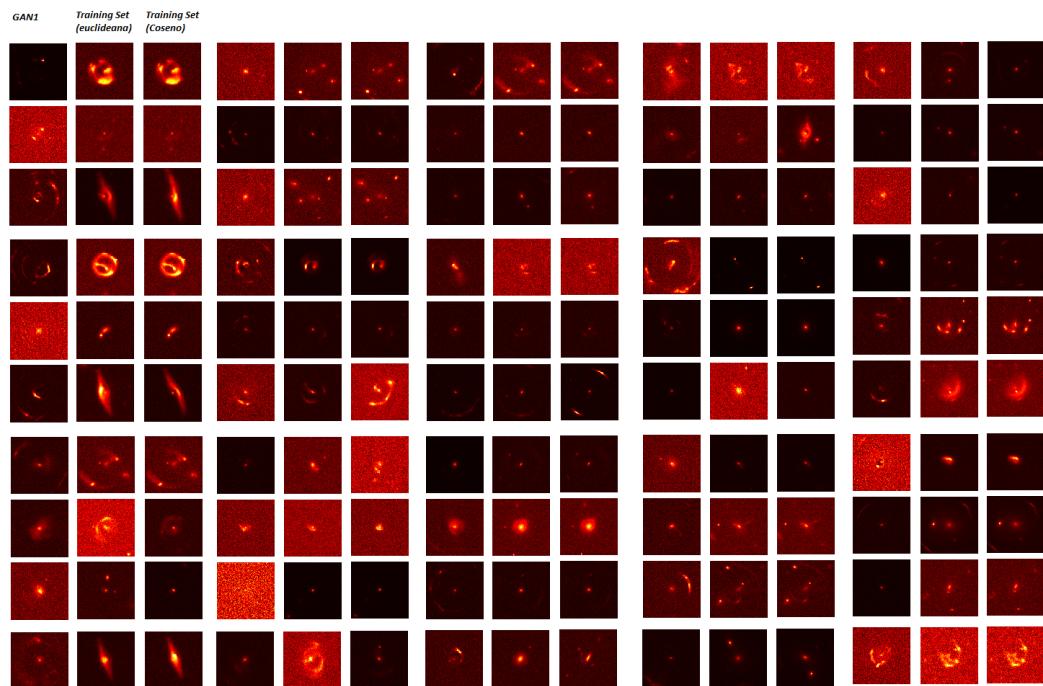


Figura 6.1: Imágenes generadas por la GAN 1, en conjunto con las imágenes más cercanas del conjunto de entrenamiento.

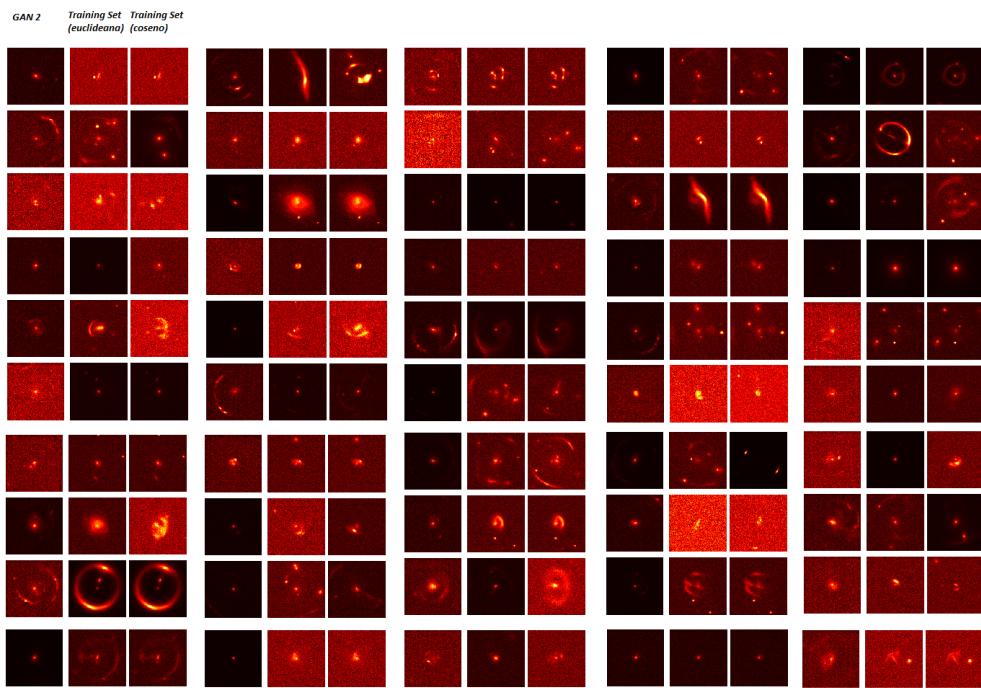


Figura 6.2: Imágenes generadas por la GAN 2, en conjunto con las imágenes más cercanas del conjunto de entrenamiento.