De Imágenes Simuladas a Reales: Adaptación de Dominio de DeepLegato para Estimar Parámetros Morfológicos de Galaxias

(From Simulated to Real Images: Domain Adaptation of DeepLegato for Morphological Parameter Estimation of Galaxies)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN DATA SCIENCE

Autor: Fernando Moreno Fernández
Director: Diego Tuccillo
Co-director: Marc Huertas-Company
Julio - 2023

Agradecimientos: En primer lugar quiero agradecer a Diego Tuccillo por dirigir mi tesis y por toda la confianza depositada en mí. También quiero dar las gracias a Marc Huertas-Company por aportar nuevos enfoques que han resultado claves en este trabajo.
Me gustaría dar las gracias a toda mi familia que siempre me ha apoyado y a todos los amigos que he hecho a lo largo de estos años, especialmente a : Alain, Sergio, Fran, Hugo, Duje y Miguel, sin vuestra amistad y ayuda nunca hubiera llegado hasta aquí. Por último dar las gracias a Casa José, por haberme acogido durante la realización de este trabajo.

Abstract:

Las imágenes astronómicas capturadas por los telescopios suelen contener un perfil de luz que representa cómo la intensidad de la luz cambia a través de la imagen. Este perfil de luz puede ser ajustado a un modelo matemático, como el perfil de Sérsic, que permite la caracterización cuantitativa de las propiedades estructurales de las galaxias, tales como el tamaño, la forma, el brillo superficial, etc. El conocimiento de los parámetros estructurales de las galaxias es de fundamental interés para entender su evolución, y como interactúan con la historia del universo. Considerando su interés, en astronomía existen muchos metodos de "light profile fitting", o ajuste del perfil de luz, para determinar las características estructurales de las galaxias a través del análisis de su distribución de luz. Debido al éxito de los métodos de Deep Learning en el tratamiento de imágenes en los últimos años, en astronomía se han explorado con éxito varios métodos basados en estas técnicas para obtener parámetros estructurales a partir de imágenes de galaxias. No obstante, el uso de Deep Learning en este ámbito, presenta el desafío constante de transicionar de estimaciones fundamentadas en simulaciones a las basadas en datos reales. Este Trabajo de Fin de Máster (TFM) explora este desafío, utilizando DeepLegato, el primer método en la literatura que emplea redes convolucionales para estimar parámetros morfológicos de galaxias a partir de imágenes. En este estudio se han desarrollado y comparado sistemáticamente varias técnicas de adaptación de dominio, incluyendo transfer learning, adaptación de dominio adversarial supervisada y semi-supervisada, con el fin de evaluar su efectividad y potencial a futuro.

Palabras Clave: Deep Learning, Domain Adaptation, Galaxies.

Índice general

1. Introducción **2**
1.1. Galaxias . . . 2
1.2. Parámetros estructurales de las galaxias . . . 2
1.3. Large Area Surveys . . . 5
1.4. Machine Learning y Deep Learning . . . 6
1.5. Deep Learning en Astronomía . . . 8
1.6. Domain Shift en el Deep Learning . . . 9
1.7. Objetivos y estructura del TFM . . . 10

2. Metodología **12**
2.1. DeepLegato . . . 12
2.2. Limitaciones de DeepLegato . . . 12
2.3. Técnicas de Adaptación de Dominio . . . 13
2.3.1. Transfer Learning . . . 13
2.3.2. Adversarial Domain Adaptation . . . 13
2.4. Estrategia de estudio . . . 14
2.5. Métricas y evaluación . . . 14

3. Datos **15**
3.1. Conjuntos de Datos . . . 15
3.2. Galaxias Reales . . . 15
3.3. Galaxias Simuladas . . . 16

4. Transfer Learning **19**
4.1. Arquitectura . . . 19
4.2. Entrenamiento y Testeo . . . 19
4.2.1. Prueba en Imágenes Simuladas . . . 19
4.2.2. Prueba en Imágenes Reales . . . 21
4.3. Aplicación de Transfer Learning . . . 21

5. Adversarial Domain Adaptation **27**
5.1. Arquitectura . . . 27
5.2. Entrenamiento y Testeo . . . 27
5.2.1. Supervisado . . . 29
5.2.2. Semi-Supervisado . . . 29

Conclusiones **38**

Apéndice **41**

1. Introducción

1.1. Galaxias

Las galaxias son estructuras astronómicas que albergan entre millones y billones de estrellas, junto con gas, polvo, materia oscura y objetos subgalácticos, como cúmulos estelares, nebulosas y agujeros negros. Las galaxias son una pieza fundamental en la composición del universo; son las principales ubicaciones donde ocurre la formación y evolución de estrellas, y sus interacciones a gran escala han jugado un papel crucial en la estructura del universo tal como lo conocemos hoy.
Las galaxias se han observado durante miles de años, pero no fue hasta el siglo XX que los astrónomos comenzaron a entender realmente lo que son. En la década de 1920, Edwin Hubble demostró que las "nebulosas"que se observaban en el cielo eran en realidad galaxias independientes, más allá de la Vía Láctea. Desde entonces, los avances tecnológicos han permitido observaciones cada vez más detalladas y precisas de las galaxias. Los telescopios modernos pueden detectar la luz en una amplia gama de longitudes de onda, desde el radio hasta los rayos gamma, proporcionando una gran cantidad de información sobre las galaxias y sus componentes.

Figura 1.1: Se muestra una imagen de NGC 4414, una galaxia en espiral a 20 millones de parsec de distancia tomada por el telescopio espacial Hubble. Imagen tomada de [1]

1.2. Parámetros estructurales de las galaxias

Los astrónomos exploran las maneras en las que las galaxias se forman y evolucionan mediante diversos métodos. Uno de estos métodos es el análisis de los espectros de las galaxias, que nos permite entender la composición química a partir de la luz que emiten. Esto es importante porque los elementos químicos presentes en una galaxia pueden indicarnos mucho acerca de su historia y evolución.

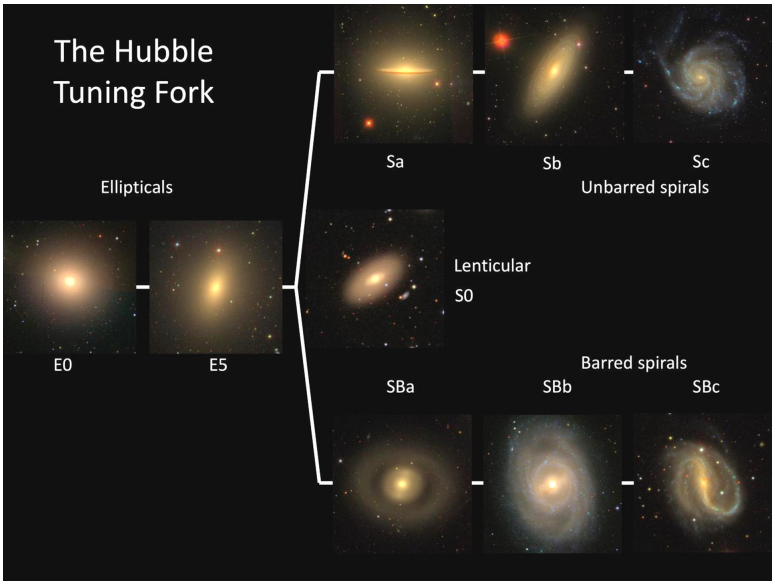

Figura 1.2: Se muestra una imagen del esquema de Hubble, el primer ejemplo de clasificación morfológica cuya nomenclatura sigue usándose a día de hoy. Imagen tomada de [2]

Además, el estudio de la morfología de las galaxias, es decir, la comparación de sus diferentes formas a lo largo de la historia del cosmos, nos permite trazar cómo han llegado a adoptar su aspecto actual. Las galaxias pueden adoptar muchas formas, desde espirales con brazos bien definidos hasta elípticas más uniformes, y estas formas pueden dar pistas sobre los eventos que han afectado a la galaxia a lo largo de su vida, como fusiones con otras galaxias.

Por último, el estudio de los parámetros estructurales de las galaxias es una parte esencial de la astronomía. Estos parámetros incluyen cosas como el tamaño de una galaxia, su forma, su brillo superficial, su perfil de brillo y otros aspectos de su estructura. Los astrónomos pueden utilizar estas medidas para entender una serie de propiedades físicas de las galaxias, desde sus historias de formación estelar hasta sus contenidos de materia oscura. Además, al estudiar cómo estos parámetros cambian con el tiempo, los astrónomos pueden obtener pistas sobre la evolución de las galaxias y el universo en general.

Las imágenes astronómicas capturadas con los telescopios suelen contener un perfil de luz que representa cómo la intensidad de la luz cambia a través de la imagen. Este perfil de luz puede ser ajustado a un modelo matemático, como el perfil de Sérsic, que permite la caracterización cuantitativa de las propiedades estructurales de las galaxias, tales como el tamaño, la forma, el brillo superficial, etc. El conocimiento de los parámetros estructurales de las galaxias es de fundamental interés para entender sus evolución, y como interactúan con la historia del universo.

La luz de las galaxias sigue una distribución de brillo en el cielo nocturno dada por $I(x, y)$ donde I es la intensidad o distribución de brillo superficial medida en unidades de luminosidad por unidad de área en la posición (x,y) donde el área está en unidades físicas (pc^2) o en unidades de área angular en segundos de arco cuadrado. La intensidad suele representarse mediante una función promediada

radialmente, $I(R)$. La luminosidad total de una galaxia es entonces[3]:

$$L_{tot} = 2\pi \int_0^\infty I(R) R dR \tag{1.1}$$

Normalmente el brillo superficial suele expresarse en unidades de magnitud por segundo de arco, esta cantidad denotada por μ representa la magnitud aparente de la luz total equivalente observada en un segundo de arco cuadrado en diferentes puntos de la distribución. Esta cantidad se relaciona con el perfil de luz superficial de la forma[3]:

$$\mu \propto -2{,}5 log_{10} I(R) \tag{1.2}$$

El perfil radial de brillo superficial de una galaxia se describe mediante la función de Sersic dada por[3]:

$$I(R) = I(0) e^{-b_n (\frac{R}{R_e})^{\frac{1}{n}}} \tag{1.3}$$

Siendo R_e el radio efectivo de la galaxia, que representa el radio que contiene la mitad de la luz total emitida por la galaxia, b_n es un parámetro ajustable que surge de las propiedades de integración en R_e y n es el índice de Sersic que describe la curvatura de la intensidad de la luz de la galaxia en función de la distancia al centro R (ver figura 1.3).
Por último, a partir del perfil de luz de una galaxia también se puede obtener su elipticidad, que se define como:

$$elip = \frac{a - b}{a} \tag{1.4}$$

Donde a es el semi-eje mayor y b el semi-eje menor de una elipse.

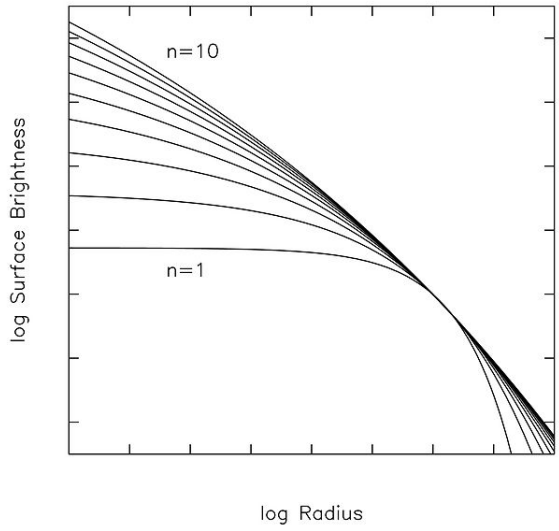

Figura 1.3: Se muestra la curva de intensidad de luz de una galaxia en función de la distancia al centro para diferentes valores del índice de Sersic (n).

Existen muchos métodos de "light profile fitting", o ajuste del perfil de luz. A lo largo de los años se han ido desarrollando multitud de métodos y algoritmos para determinar los parámetros estructurales de las galaxias a partir de su perfil de luz. GALFIT [4] y Gim2d[5] son dos de los algoritmos más usados en astronomía, estos algoritmos requieren de la fijación manual de ciertos parámetros antes de realizar el ajuste y por tanto, a medida que la cantidad de datos disponible ha ido aumentando con el tiempo, se han ido desarrollando nuevos algoritmos que tratan de automatizar este proceso, por ejemplo pyMorph [6] en 2010 y posteriormente ProFit[7] en 2017. A partir de 2018 con el desarrollo de DeepLegato [8], se ha probado la eficacia de los algoritmos de deep learning como las redes convolucionales para llevar a cabo esta tarea y todo a apunta a que las técnicas basadas en deep learning van a seguir ganando popularidad con el paso de los años.

1.3. Large Area Surveys

Uno de los principales enfoques dentro de la astronomía hoy en día para recolectar datos es la realización de large area surveys, como por ejemplo: HSC [9], HST/CANDELS[10],SDSS[11],COSMOS[12], EUCLID [13], SKA [14] etc... Se trata de grandes proyectos de investigación cuyo fin es mapear grandes regiones del cielo conteniendo un gran número de objetos celestes. Para ello se hacen uso de multitud de telescopios, tomando datos en diferentes longitudes de onda: rayos x, infrarrojo, ondas de radio, visible...
De esta manera se analizan extensas regiones del cielo y se clasifican grandes cantidades de objetos astronómicos: galaxias, nebulosas, planetas, estrellas...
Es por ello que el volumen de galaxias disponibles para analizar es cada vez más grande y crece a un

Figura 1.4: Se muestra una imagen del Hubble Space Telescope (HST) usado en surveys como CANDELS. Imagen tomada de [15]

ritmo vertiginoso. En 2014 se publicó un estudio que puede encontrase en [16], donde se menciona que la cantidad de datos astronómicos disponible en aquel entonces, ya se encontraba en la escala

de los petabytes (1000000 Gb). Teniendo en cuenta que no ha dejado de crecer desde entonces, y va a seguir haciéndolo en los próximos años con la realización de nuevos large area surveys, se puede afirmar que el big data y el machine learning son herramientas vitales a día de hoy para poder extraer todo el conocimiento de estos conjuntos de datos masivos.

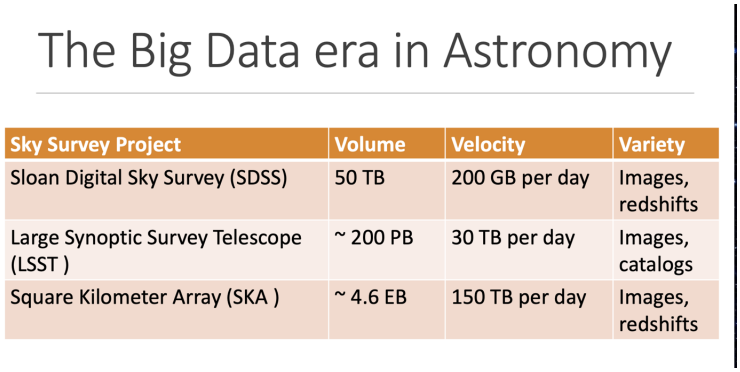

Figura 1.5: Se muestra la comparativa del volumen de datos generado por SDSS, LSST y SKA donde puede verse que el volumen de datos generado crece a un ritmo cada vez mayor. Imagen tomada de [17]

Disponer de grandes cantidades de datos de galaxias permite estudiar su estructura y evolución con más precisión que nunca, minimizando los errores estadísticos asociados al estudio de las distribuciones de parámetros, tipos de galaxias etc...
Sin embargo, aunque se disponga de una gran cantidad de datos, los algoritmos tradicionalmente usados para estimar los parámetros morfológicos de galaxias a partir del perfil de luz como son GALFIT y Gim2d requieren fijar ciertos parámetros manualmente antes de analizar cada imagen por lo que no están pensados para trabajar sobre un volumen de datos tan grande como del que se dispone, mucho menos del que se se dispondrá en los próximos años. En resumen, los enfoques tradicionales que requieran algún grado de supervisión humana ya no son viables para el estudio de los parámetros estructurales de galaxias.
Se han realizado intentos de mitigar este problema con algoritmos como pyMorph, un software pipeline que trata de automatizar la estimación de parámetros morfológicos tomando los datos del perfil de luz de diferentes galaxias y pasándoselo a algoritmos tradicionales que hacen el ajuste, como puede ser GALFIT. Aún así, se requiere de cierta configuración inicial antes de realizar el ajuste, lo que ralentiza el análisis de grandes conjuntos de datos como de los que se dispone.
Por tanto, para poder sacar partido a la gran cantidad de datos de las que se disponen se necesitan nuevos algoritmos capaces de estimar los parámetros morfológicos de manera rápida y totalmente automática.

1.4. Machine Learning y Deep Learning

Los algoritmos de aprendizaje automático (machine learning) se caracterizan porque una vez entrenados, pueden producir los resultados requeridos de manera rápida y automática, es por ello que el interés por estos ha crecido notablemente en los últimos años. A medida que la cantidad

de datos disponibles y la complejidad de los problemas ha ido aumentando la rama del machine learning conocida como deep learning ha ganando una creciente popularidad.
El deep learning es capaz de resolver problemas más complejos que los algoritmos de machine learning tradicionales. La principal diferencia con el machine learning tradicional es que los algoritmos de deep learning extraen las carácteristicas relevantes directamente de los datos, por el contrario, a los algoritmos de machine learning tradicionales hay que especificarles que características son relevantes para una resolver una determinada tarea.
Uno de los algoritmos de deep learning más comunes son las redes convolucionales (CNN). Siguiendo

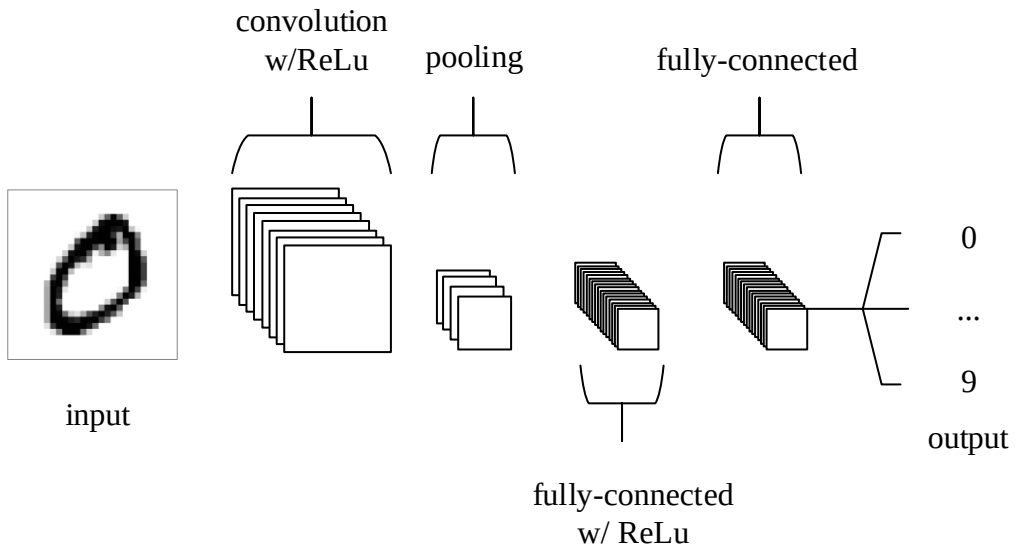

Figura 1.6: Se muestra un esquema de red convolucional para clasificar imágenes del dataset MNIST. Imagen tomada de [18]

el esquema de la figura 1.6, la estructura de una red convolucional consta de 4 partes básicas[18]:

- La **capa de entrada**, que en este caso recibe los valores de los píxeles de las imágenes.
- La **capa convolucional** calculará el producto escalar entre los pesos de las neuronas conectadas a regiones locales de la entrada y los valores de la imagen en estas regiones, esto se traduce en que las capas convolucionales deslizan un filtro por la imagen aprendiendo pequeñas características, como bordes, formas etc... Además a la salida de la capa anterior se le aplica una función de activación no lineal, en este caso ReLu(rectified linear unit).
- La **capa de pooling** o agrupación realiza un muestreo a lo largo del espacio de la imagen (alto y ancho) reduciendo la dimensionalidad espacial de la entrada para reducir el numero de parámetros a ajustar.
- Por último, las **capas densas** se comportan como en los algoritmos tradicionales, reciben la salida de las capas anteriores , aplican la función de activación y ajustan los pesos para poder predecir el número de la imagen en este caso.

1.5. Deep Learning en Astronomía

El deep learning ha visto un increíble crecimiento desde 2012 con la publicación del articulo de Kocewski [19] donde las redes convolucionales ganaron el concurso de clasificación de imágenes de ImageNet. El primer articulo donde se ha aplicado deep learning en astronomía es del 2015 [20], aplicado a clasificación morfológica de galaxias. A partir de entonces se han publicado cada vez más trabajos científicos en astronomía aplicando deep learning en diferentes ámbitos y problemas. Como puede verse en la figura 1.7 , el numero de artículos en astronomía que utilizan la palabra "neural network"ha crecido vertiginosamente desde 2015.

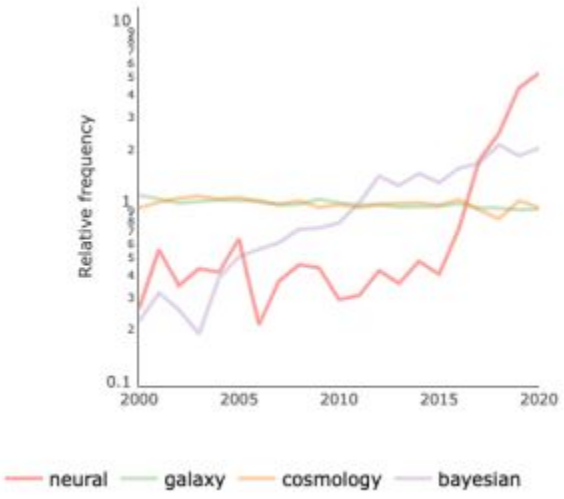

Figura 1.7: Se muestra la frecuencia relativa de artículos de astronomía con diferentes palabras en el abstract a lo largo del tiempo. Imagen tomada de [17]

Sin embargo, antes de poder aplicar estos algoritmos directamente sobre los datos de los surveys hay que tener en cuenta diversos inconvenientes, los más importantes en este contexto son:

- **Costo computacional:** Debido al gran número de parámetros ajustables que poseen este tipo de algoritmos, las fases de entrenamiento suelen ser muy costosas en tiempo y recursos computacionales.
- **Volumen de datos necesarios:** Intrínsecamente relacionado con el anterior, al tener muchos parámetros por ajustar se requieren de grandes cantidades de datos para poder capturar las características generales y la variabilidad del problema.
- **Ground true:** Durante el entrenamiento la red necesita las imágenes de galaxias y los valores de los parámetros de esas galaxias para aprender a predecirlos(aprendizaje supervisado), sin embargo el valor exacto o ground true de los parámetros de galaxias que se observan no se conoce, por tanto, si se usan por ejemplo , los valores de los parámetros obtenidos con GALFIT para un conjunto de galaxias, la red nunca podrá dar mejores resultados que GALFIT ya que

ha aprendido a partir de los valores proporcionados por GALFIT, por tanto su precisión quedaría limitada por las incertidumbres de GALFIT.

- **Domain shift:** El problema del ground true impide usar simplemente un conjunto de imágenes de galaxias reales para el entrenamiento de la red. La solución mas natural es usar galaxias simuladas para el entrenamiento de modo que los valores de los parámetros de las galaxias se conocen con exactitud. Sin embargo, como puede verse en [8] y como se verá más adelante en este trabajo no se obtienen buenos resultados. Esto se debe a que usar imágenes simuladas introduce un nuevo problema conocido como domain shift o desplazamiento de dominio(sección 1.6), el estudio de métodos capaces de resolver este problema es el objetivo principal de este trabajo.

1.6. Domain Shift en el Deep Learning

Debido a la gran cantidad de parámetros que se ajustan en los algoritmos de machine learning se corre el riesgo de perder la capacidad de generalizar. Esta perdida de la generalización se acentúa en el caso de las redes convolucionales, debido a la cantidad de parámetros que se ajustan es más probable que se aprendan detalles muy concretos de los datos. Existen varias posibles fuentes de perdida de generalización:

- **Domain Shift:** Se refiere a la situación en la que la distribución de los datos de entrenamiento difiere de la distribución de los datos de prueba o de los datos en producción. Es un problema común cuando un modelo entrenado se aplica en un contexto ligeramente diferente al del conjunto de entrenamiento. Por ejemplo, puedes entrenar un modelo de detección de objetos con imágenes tomadas durante el día, pero luego tratar de aplicarlo a imágenes tomadas por la noche. En este caso, la distribución de los colores y las texturas puede ser muy diferente, lo que lleva a un rendimiento deficiente. Este problema puede verse ilustrado en la figura 1.8.

- **Model Drift:** Se refiere al cambio en el rendimiento del modelo con el tiempo. Esto suele ser causado por cambios en la distribución subyacente de los datos, lo que hace que el modelo sea menos efectivo a medida que pasa el tiempo. Un ejemplo común es un modelo de predicción de precios de acciones que se entrena con datos de un cierto período de tiempo, pero a medida que pasa el tiempo, las condiciones del mercado cambian y el modelo se vuelve menos preciso.

- **Data Drift:** Es un cambio en la distribución de las variables de entrada del modelo a lo largo del tiempo. Esto puede ser causado por una serie de factores, como cambios en el comportamiento del usuario, cambios estacionales o tendencias a largo plazo. Por ejemplo, los patrones de compra de los clientes pueden cambiar durante las vacaciones, lo que puede llevar a un cambio en la distribución de las variables de entrada de un modelo de recomendación de productos.

- **Concept Drift:** Es un cambio en la relación entre las variables de entrada y la variable de salida a lo largo del tiempo. Por ejemplo, si estás construyendo un modelo de predicción del tiempo, la relación entre ciertas variables climáticas y la temperatura puede cambiar con las estaciones.

Para manejar estos problemas, los investigadores han desarrollado una variedad de técnicas, como el entrenamiento de dominio adversarial para el domain shift, o el monitoreo continuo de los modelos

y re-entrenamiento regular para el model drift. En algunos casos, también puede ser útil recolectar regularmente nuevos datos y re-entrenar tus modelos para mantenerlos actualizados.

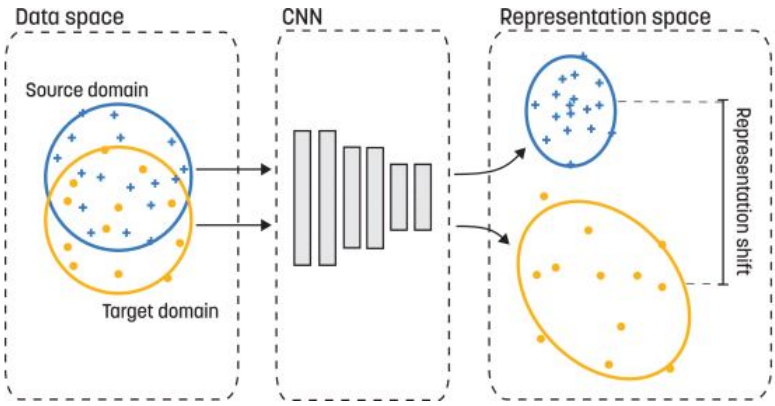

Figura 1.8: Se muestra un ejemplo visual de domain shift. Imagen tomada de [21]

Un claro ejemplo de domain shift se verá más adelante en este trabajo en la sección 4, donde se muestra como una red convolucional entrenada con imágenes simuladas no es capaz de predecir los parámetros morfológicos en imágenes reales, aunque visualmente las diferencias entre las imágenes sea mínima, esas diferencias producen grandes cambios en las representaciones internas del modelo.

1.7. Objetivos y estructura del TFM

Este Trabajo de Fin de Máster busca explorar y estudiar diferentes técnicas de adaptación de dominio, con el fin de descubrir aquellas con un mayor potencial a futuro para resolver el problema de la pérdida de generalización de las redes convoluciones entrenadas con imágenes simuladas cuando tratan de aplicarse sobre imágenes reales. Para ello se parte de DeepLegato[8], el primer método para estimar parámetros morfológicos de galaxias a partir de imágenes y que emplea el transfer learning como método de adaptación de dominio. En este trabajo se replica el método de transfer learning utilizado y se comparan los resultados obtenidos con otras dos técnicas : la adaptación de dominio adversaria supervisada y la adaptación de dominio adversaria semi-supervisada.
En esta primera sección se ha hecho una breve introducción sobre la estimación de parámetros estructurales de galaxias, su importancia, el papel del deep learning y los retos que supone el uso del deep learning para esta tarea. El resto del trabajo se estructura de la siguiente manera:

- En la sección 2 se detalla en que consiste DeepLegato y sus limitaciones, así como se explican más en profundidad las técnicas de adaptación de dominio que se van a usar, la metodología seguida en este trabajo y la métrica que va a usarse para medir la precisión de los diferentes modelos.

- En la sección 3 se describen los conjuntos de datos usados para este trabajo, de donde provienen y las distribuciones de los parámetros en ellos.

- En la sección 4 se describe la arquitectura del modelo utilizado para aplicar transfer learning, se explica el proceso de entrenamiento, como se lleva acabo el transfer learning y los resultados que se obtienen tanto antes como después de aplicar esta técnica.

- En la sección 5 se describe la arquitectura del modelo usado para aplicar la adaptación de dominio adversaria, tanto la supervisada como la semi-supervisada, se explica como se realiza el entrenamiento de cada una de las dos variantes y los resultados que se obtienen con ambas.

- Por último se dan las conclusiones finales de este trabajo a la vista de los resultados obtenidos.

2. Metodología

2.1. DeepLegato

DeepLegato es un método para estimar parámetros morfológicos de galaxias a partir de imágenes usando redes convolucionales descrito en [8]. Se usaron imágenes reales y simuladas de HST/CANDELS, concretamente del filtro F160W. La precisión de DeepLegato en imágenes reales de galaxias aisladas es similar a la de GALFIT pero unas 3000 veces más rápido usando GPU. Además una vez entrenado es capaz de analizar todas las imágenes que sean necesarias sin necesidad de ningún ajuste, permitiendo analizar grandes cantidades de galaxias de manera automática.
DeepLegato emplea una técnica de adaptación de dominio conocida como transfer learning , que consiste en entrenar primero la red con imágenes simuladas y luego realizar un pequeño entrenamiento usando imágenes reales. De esta manera la red aprende primero un conjunto de características de las galaxias a partir de imágenes simuladas y luego se ajusta más concretamente a las características de las galaxias reales. Las arquitecturas usadas en DeepLegato se muestran en la figura 2.1.

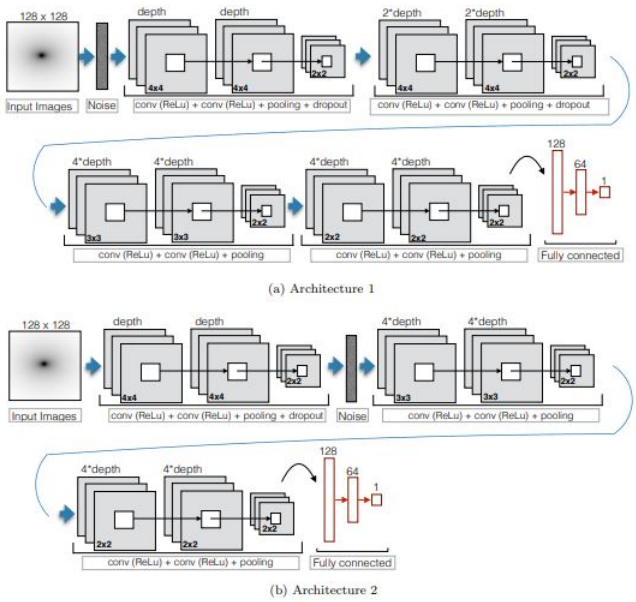

Figura 2.1: Se muestran las dos arquitecturas usadas en DeepLegato. Imagen tomada de [8]

2.2. Limitaciones de DeepLegato

DeepLegato es capaz de predecir los parámetros de galaxias reales a partir de imágenes con la misma precisión y mas rápidamente que GALFIT, sin embargo, cada parámetro requiere del entrenamiento de una red con imágenes simuladas y luego de una segunda ronda de entrenamiento

con imágenes reales, es decir, cada red solo predice un parámetro, además se usan dos arquitecturas diferentes dependiendo del parámetro a predecir y se requieren de los valores de los parámetros de las imágenes reales obtenidos mediante otro algoritmo, GALFIT en este caso.
Por ello, en este trabajo se van a construir redes capaces de predecir los 4 parámetros a la vez y se van a explorar diferentes técnicas de adaptación de dominio con el objetivo de valorar las limitaciones de cada una de cara a futuros trabajos. Emplear técnicas de adaptación de dominio resulta necesario ya que como se mencionó al final de 1.4 resolver el problema del domain shift resulta vital para obtener resultados fiables.

2.3. Técnicas de Adaptación de Dominio

Existen múltiples técnicas de adaptación de dominio, este trabajo se va enfocar en dos técnicas diferentes: transfer learning y adversarial domain adaptation.

2.3.1. Transfer Learning

Como ya se explicó anteriormente el transfer learning consiste en entrenar una red con un conjunto de datos y luego realizar una segunda fase de entrenamiento con otro conjunto de datos diferente pero que tenga características comunes al primero. En este caso, entrenar primero una red con imágenes simuladas permite que la red aprenda una gran cantidad de características diferentes de una gran cantidad de galaxias que no podría aprender solo con imágenes reales porque no se dispone de tanta cantidad, además evita el problema del ground true explicado en 1.4. La segunda fase del entrenamiento con imágenes reales permite que al red se adapte a las particularidades de las imágenes reales de modo que la red aprende detalles específicos de este dominio.

2.3.2. Adversarial Domain Adaptation

El adversarial domain adaptation o adaptación de dominio adversaria consiste en usar una red con dos salidas, por un lado la red extrae las características de las galaxias y realiza la regresión de los parámetros mientras que por otro lado utiliza las características extraídas para tratar de clasificar las galaxias reales y simuladas. La red se entrena con un conjunto de imágenes que contiene galaxias reales y simuladas, usando una función de pérdida conjunta entre la regresión y la clasificación. Por tanto la red intenta aprender a predecir los parámetros a la vez que intenta clasificar cada vez mejor las imágenes. Como lo que se busca es que aprenda características comunes a los dos dominios(real y simulado) se introduce en el clasificador una capa que invierte la dirección del gradiente, de este modo se penaliza cuando la red aprende a distinguir y se fuerza a aprender características comunes a los dos dominios.
En este trabajo se van a explorar dos variantes del adversarial domain adaptation:

- **Supervisado:** Se entrena la red usando un conjunto mixto de imágenes que contiene galaxias reales y simuladas y los valores de sus respectivos parámetros.
- **Semi-Supervisado:** Se entrena la red usando un conjunto mixto de imágenes que contiene galaxias reales y simuladas pero solo se le pasan los parámetros de las galaxias simuladas.

La ventaja que puede tener la variante semi-supervisada es que, si se obtienen buenos resultados, elimina la necesidad de usar otro algoritmo como GALFIT para predecir de antemano los paráme-

tros de las galaxias reales. De esta manera se podrían analizar las galaxias directamente según se observan.

2.4. Estrategia de estudio

Este trabajo toma como base DeepLegato tratando de replicar los resultados obtenidos con transfer learning y comparándolos con los que se obtienen al usar adaptación de dominio adversaria tanto supervisada como semi-supervisada. Para ello se emplea un conjunto de datos similar al que se usó en DeepLegato y se aplica transfer learning de una manera similar a como se hizo en DeepLegato. Sin embargo, se utiliza un modelo diferente a los usados en DeepLegato, más complejo y capaz de predecir los 4 parámetros a la vez, en vez de uno solo como hace DeepLegato. Una vez se entrena el modelo y se aplica el transfer learning se evalúa usando R^2 como métrica tal y como se describe en la sección 2.5. Después se aplica adaptación de dominio adversaria supervisada y semi-supervisada siguiendo los pasos descritos en 5, se evalúan los modelos entrenados y se comparan los resultados con los obtenidos con transfer learning.

2.5. Métricas y evaluación

Para evaluar los resultados obtenidos en este trabajo se va a utilizar como métrica el coeficiente de determinación R^2 que en una regresión mide la precisión con la que se ajusta la predicción del modelo al valor real:

$$R^2 = 1 - \frac{\sum_i^n (y_i - f_i)^2}{\sum_i^n (y_i - \bar{y})^2} \tag{2.1}$$

Siendo f_i el valor predicho por el modelo para un valor real y_i, mientras que $\bar{y}$ es el valor medio del conjunto de n datos. El valor de R^2 puede variar entre $-\infty \leq R^2 \leq 1$. Los valores negativos significarían que una linea recta horizontal en $\bar{y}$ predice mejor los datos que el modelo, mientras que los valores cercanos a 1 indican que las predicciones del modelo se ajustan a los valores reales y_i.
Las predicciones de los modelos se van a comparar con las predicciones de GALFIT para esas mismas galaxias calculando el coeficiente R^2, tomando como mejor modelo aquel que obtenga un R^2 más próximo a 1.

3. Datos

3.1. Conjuntos de Datos

Para este trabajo se han usado dos conjuntos de datos de HST/CANDELS (sección 3.2), uno de 5000 imágenes de galaxias reales y otro de 150,000 galaxias simuladas. En las secciones 3.2 y 3.3 se describen más en detalle cada uno de los dos conjuntos. En la figura 3.2 se visualiza una pequeña muestra de galaxias reales y en figura 3.3 de galaxias simuladas.

3.2. Galaxias Reales

El Cosmic Assembly Near-IR Deep Extragalactic Legacy Survey (CANDELS) es un large area survey realizado por el Telescopio Espacial Hubble o HST por sus siglas en inglés en el que se mapearon 5 regiones del cielo a diferentes longitudes de onda.
La principal fuente de datos de CANDELS proviene de las imágenes tomadas con la cámara de infrarrojo cercano Wide Field Camera 3 (WFC3/IR) junto con la cámara Advanced Camera for Surveys (ACS) que opera en el rango visible. Esta y más información sobre el programa CANDELS puede verse en [10].
Para este trabajo se seleccionó una muestra de 5000 imágenes de galaxias de CANDELS cuyo rango de parámetros se encontrase contenido en el rango de parámetros de las imágenes simuladas (tabla 3.2), resultando en los rangos mostrados en la tabla 3.1. Es importante que los valores de los parámetros de las imágenes reales se encuentren dentro del rango de valores de las simuladas ya que las redes convolucionales son capaces de predecir de manera fiable dentro del espacio de parámetros que han visto durante el entrenamiento, es decir, si durante el entrenamiento solo vio galaxias con magnitudes entre 16 y 27 y se trata de predecir una galaxia con magnitud 14, los resultados no suelen ser fiables. Por ello, al usar un rango más amplio en las imágenes simuladas nos aseguramos de que siempre que vaya a predecir en imágenes reales haya visto con anterioridad galaxias con parámetros similares. Los valores de los parámetros de las galaxias que se usan fueron obtenidos por GALFIT en [22] , donde se centran en los datos obtenidos por WFC3 que contiene tres filtros: F105W, F125W, y F160W, aunque para este trabajo solo se han seleccionado galaxias del filtro F160W, porque GALFIT proporciona resultados con menor incertidumbre en este filtro. En la figura 3.1 se muestran las distribuciones de los 4 parámetros obtenidos por GALFIT para

	Magnitud (AB)	Radio (Píxeles)	Índice de Sersic	Elipticidad
Rango	16-25	1.5-31.6	0.3-6.2	0.1-1.0

Tabla 3.1: Se muestra el rango de valores de los parámetros obtenidos con GALFIT para las imágenes de galaxias reales

este conjunto de galaxias.

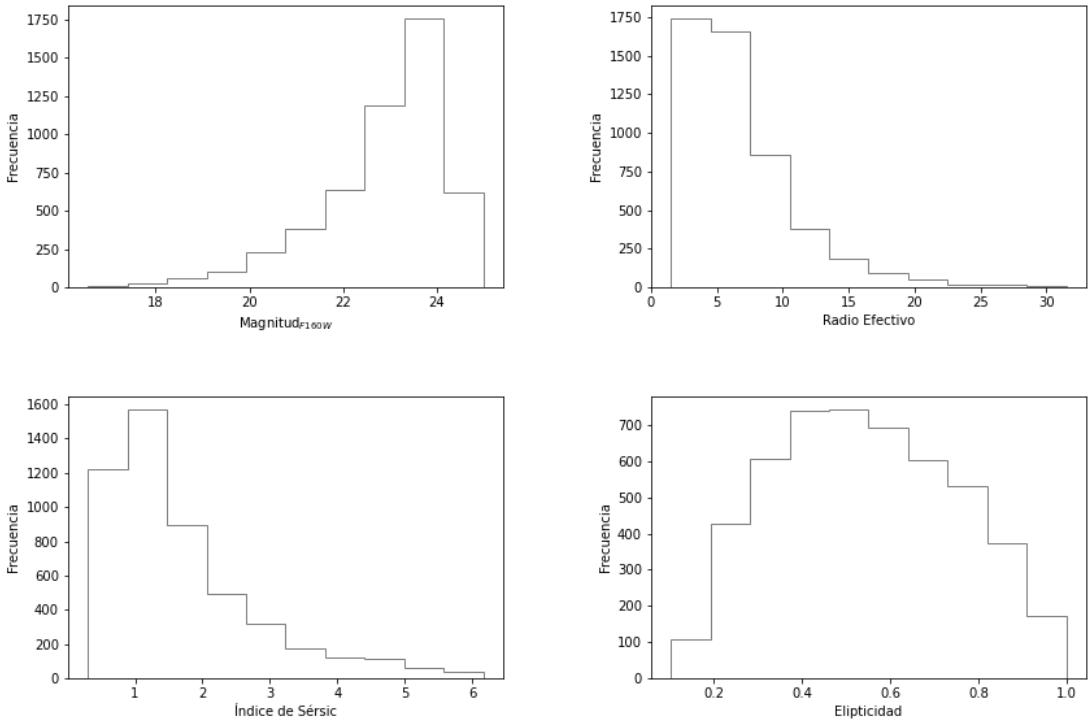

Figura 3.1: Se muestran las distribuciones de parámetros calculadas por GALFIT para el conjunto de galaxias reales

3.3. Galaxias Simuladas

El conjunto de imágenes simuladas se obtuvo usando el algoritmo GalSim versión 1.4 [23]. Para hacer las imágenes mas realistas las simulaciones incluyen ruido real del filtro F160W de HST/CANDELS. El conjunto de imágenes final se obtuvo variando uniformemente los diferentes parámetros entre los rangos mostrados en la tabla 3.2. Los rangos escogidos son aquellos en los que GALFIT proporciona resultados más precisos, de modo que los parámetros de las imágenes reales tengan las incertidumbres más pequeñas posibles. Las galaxias con magnitudes muy bajas son demasiado brillantes para GALFIT, mientras que con magnitudes muy altas son tan poco brillantes que no se aprecia prácticamente nada en al imagen.

	Magnitud (AB)	Radio (Píxeles)	Índice de Sersic	Elipticidad
Rango	16-27	1.3-50.2	0.3-6.2	0-1.0

Tabla 3.2: Se muestra el rango de valores de los parámetros de las imágenes simuladas

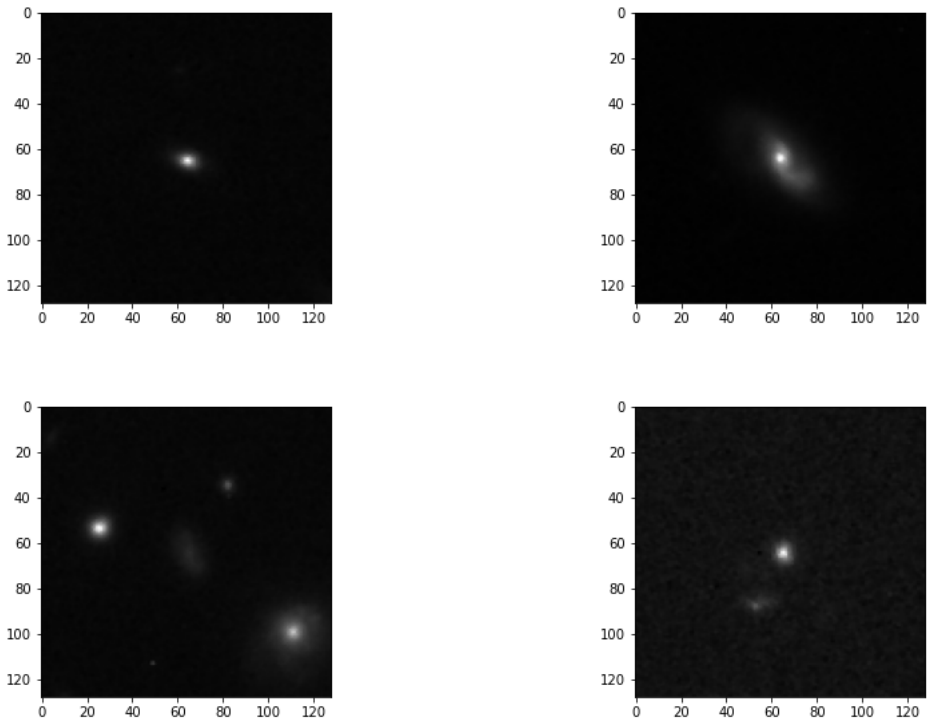

Figura 3.2: Se muestran 4 ejemplos de galaxias reales utilizadas en este trabajo.

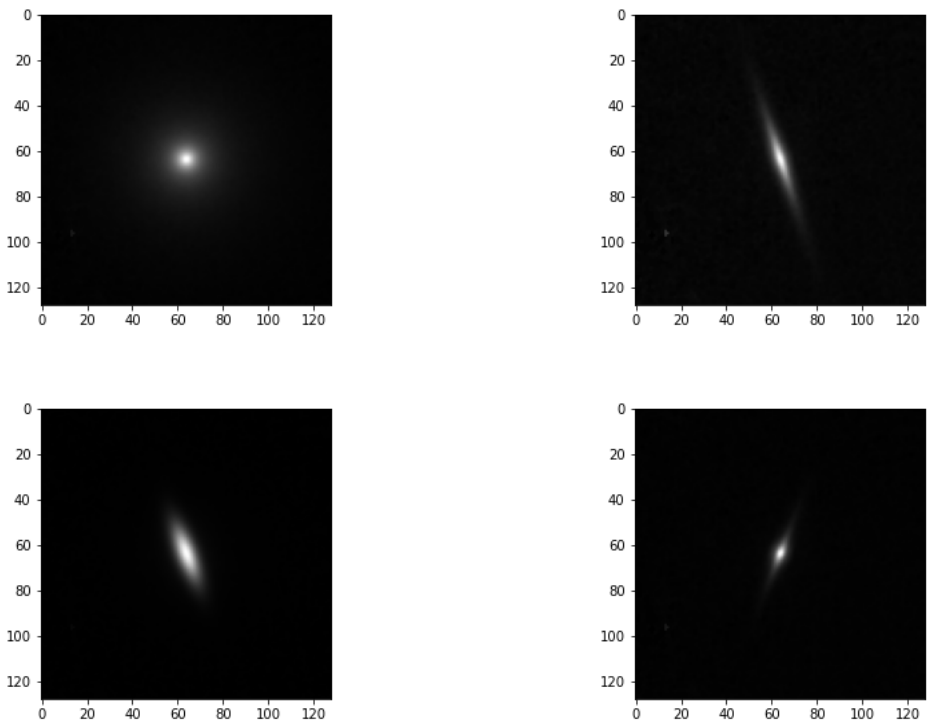

Figura 3.3: Se muestran 4 ejemplos de galaxias simuladas utilizadas en este trabajo.

4. Transfer Learning

4.1. Arquitectura

La arquitectura del modelo usado en esta sección puede verse en el esquema de la figura 4.1. Las imágenes tienen tamaño 128x128 por tanto la capa de entrada tiene el mismo tamaño, después una capa de batch normalization que normaliza la salida de la capa anterior, seguido de varios conjuntos de 2 capas convolucionales (con función de activación relu) y una de max pooling(reduce la dimensionalidad espacial) hasta llegar a las tres capas de neuronas densas con capas de dropout(desactiva neuronas con una cierta probabilidad) intercaladas y una capa de batch normalization justo antes de la capa de salida.Todas las capas convolucionales del modelo presentan regularización L2. La implementación del modelo puede encontrarse en el apéndice.

4.2. Entrenamiento y Testeo

La red neuronal se entrena y prueba usando los datos descritos en 3.2 y 3.3, para ello primero se normalizan las imágenes dividiendo el valor de todos los píxeles entre 255. Se filtran las galaxias en función de su magnitud, eliminando aquellas con $m > 23$ ya que se trata de galaxias muy poco brillantes por lo que la red suele dar resultados poco fiables. Una vez se han filtrado las galaxias por magnitud se normalizan los parámetros de las galaxias usando la función StandardScaler() del paquete sklearn.
Del conjunto de imágenes simuladas se seleccionan aleatoriamente 50,000 imágenes como conjunto de entrenamiento, 10,000 como conjunto de validación y 10,000 como conjunto de testeo. Por otro lado, del conjunto de imágenes reales se seleccionan aleatoriamente 1000 imágenes como conjunto de entrenamiento y 1000 como conjunto de testeo. Es importante destacar que en ningún caso aparece la misma imagen en dos conjuntos diferentes.
La red con la arquitectura descrita en 4.1 se entrena utilizando el método de early-stopping, que consiste en detener el entrenamiento si una métrica que se monitoriza no ha mejorado en un número de épocas denominado paciencia. En este caso la red se entrena usando los conjuntos de entrenamiento y validación de imágenes simuladas durante un máximo de 100 épocas con una paciencia de 10 épocas monitorizando el error absoluto medio del conjunto de validación y usando el error absoluto medio como función de pérdida. Los hiper-parámetros usados en esta parte del entrenamiento pueden verse en la tabla 4.1.

Learning Rate	Batch Size	Tasa de Dropout	L2
0.0001	16	0.0001	0.0001

Tabla 4.1: Se muestra los hiper-parámetros utilizados durante el entrenamiento de la red con imágenes simuladas

4.2.1. Prueba en Imágenes Simuladas

Una vez completado el entrenamiento la red se prueba sobre el conjunto de testeo de imágenes simuladas obteniéndose los resultados mostrados en la tabla 4.2. Los resultados de la tabla 4.2 se

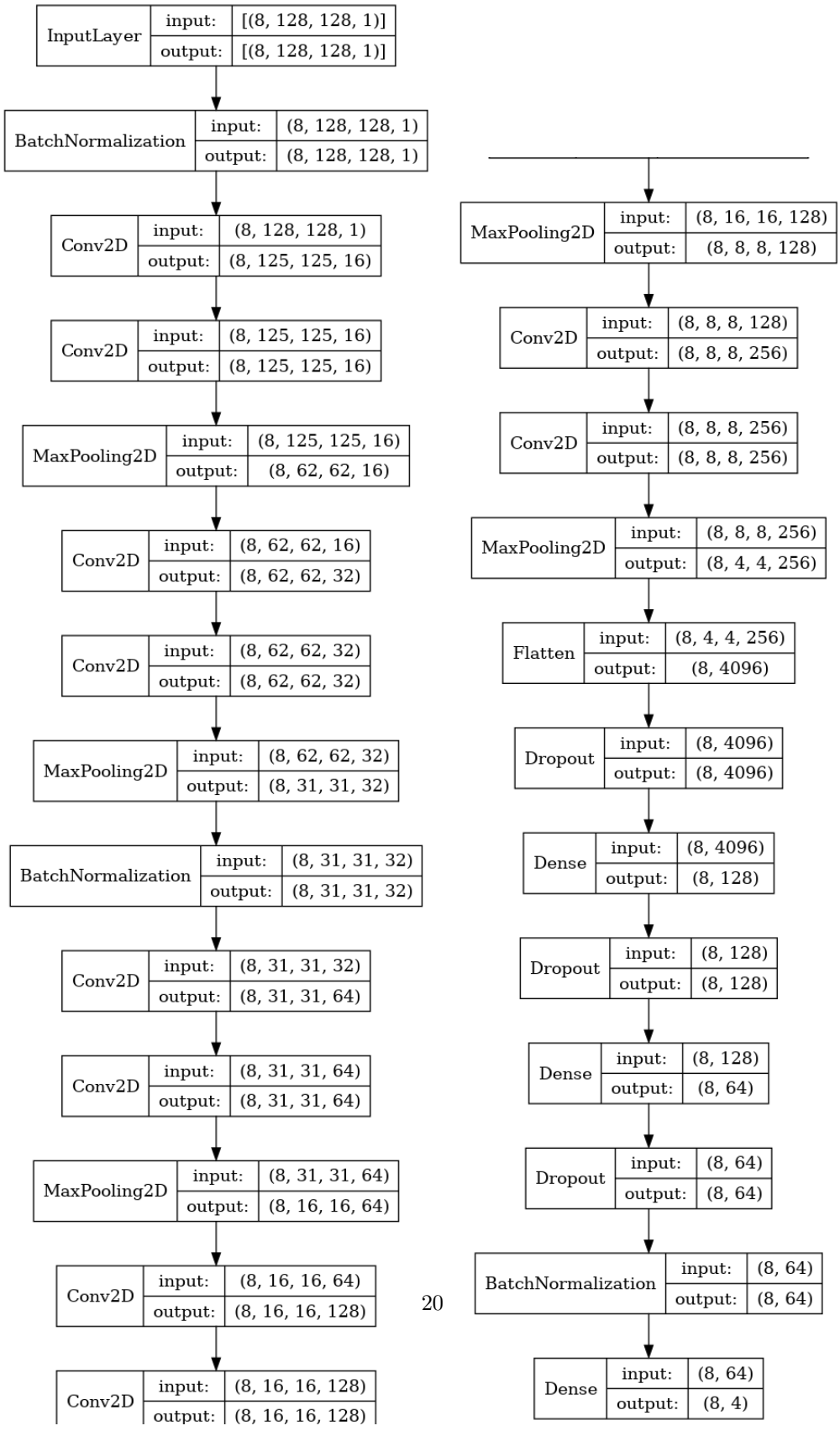

Figura 4.1: Se muestra la arquitectura del modelo utilizado en la sección 4 para aplicar transfer learning.

R^2 Simulado			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
0.9812	0.9487	0.9741	0.9823

Tabla 4.2: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias simuladas

visualizan en la figura 4.2 donde el eje y se corresponde con la predicción del modelo, el eje x el valor real o ground true y la línea discontinua representa el caso y=x, es decir, la línea que se formaría si el modelo y el ground true coincidieran exactamente. Los resultados de la tabla 4.2 muestran que la red es capaz de predecir con bastante precisión los 4 parámetros a la vez en imágenes simuladas. Esto puede comprobarse atendiendo a la figura 4.2 , donde se muestra como la predicción del modelo y el valor real (ground true) se adecuan a la recta y=x que representa el modelo perfecto.

4.2.2. Prueba en Imágenes Reales

Se prueba el modelo entrenado sobre el conjunto de prueba de galaxias reales y se obtienen los resultados mostrados en la tabla 4.3. Como se muestra en esa tabla el modelo capaz de predecir los parámetros en galaxias simuladas no es capaz de predecir en galaxias reales. En la figura 4.3 se muestra nuevamente la predicción frente al ground true, que en el caso de las galaxias reales es el valor obtenido mediante GALFIT. De esta manera se comprueba que el modelo a pesar de dar buenos resultados en galaxias simuladas no es capaz de replicar esos buenos resultados en galaxias reales. Este caso ilustra claramente el problema del domain shift explicado en la sección 1.6.

R^2 Real			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
-0.1226	-0.1102	-0.4023	-1.4060

Tabla 4.3: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de imágenes reales

4.3. Aplicación de Transfer Learning

En esta sección se aplica la técnica de transfer learning descrita en 2.3.1 . Para ello se parte del modelo entrenado con galaxias simuladas y se realiza un segundo entrenamiento usando el conjunto de entrenamiento de galaxias reales. La segunda fase de entrenamiento puede realizarse de dos maneras diferentes: entrenando solo las capas densas o entrenando todas las capas del modelo. En este caso se entrenó el modelo completo ya que los resultados obtenidos fueron mejores. El entrenamiento se llevo acabo durante 15 épocas monitorizando el error absoluto medio del conjunto de validación, en este caso el 20 % del conjunto de entrenamiento, y se guarda el modelo que mejor puntuación obtuvo en esa métrica durante el entrenamiento. Los hiper-parámetros usados en este entrenamiento se muestran en la tabla 4.4. Los resultados de la aplicación del transfer learning (TL) se comparan en la tabla 4.5 con los resultados obtenidos antes de realizar el transfer learning. Resulta evidente que la aplicación del transfer learning ha mejorado significativamente la precisión del modelo.
En la figura 4.4 se vuelve a representar la predicción del modelo para galaxias reales frente al ground

Learning Rate	Batch Size	Tasa de Dropout	L2
0.0001	8	0.0001	0.0001

Tabla 4.4: Se muestran los hiper-parámetros utilizados durante el entrenamiento de la red con galaxias reales

true, obteniendo esta vez resultados mucho mejores que antes del transfer learning(figura 4.3).
Por último en la figura 4.5 se muestra la comparación entre las distribuciones de los parámetros predichas por el modelo y las distribuciones reales, en este caso obtenidas con GALFIT. Se aprecia que ambas distribuciones tienen formas similares aunque el modelo no es capaz de reproducir las distribuciones de forma totalmente precisa.

	R^2 Real			
	Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
PRE TL	-0.1226	-0.1102	-0.4023	-1.4060
POST TL	0.9734	0.8530	0.8318	0.8435

Tabla 4.5: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias reales antes y después del transfer learning(TL).

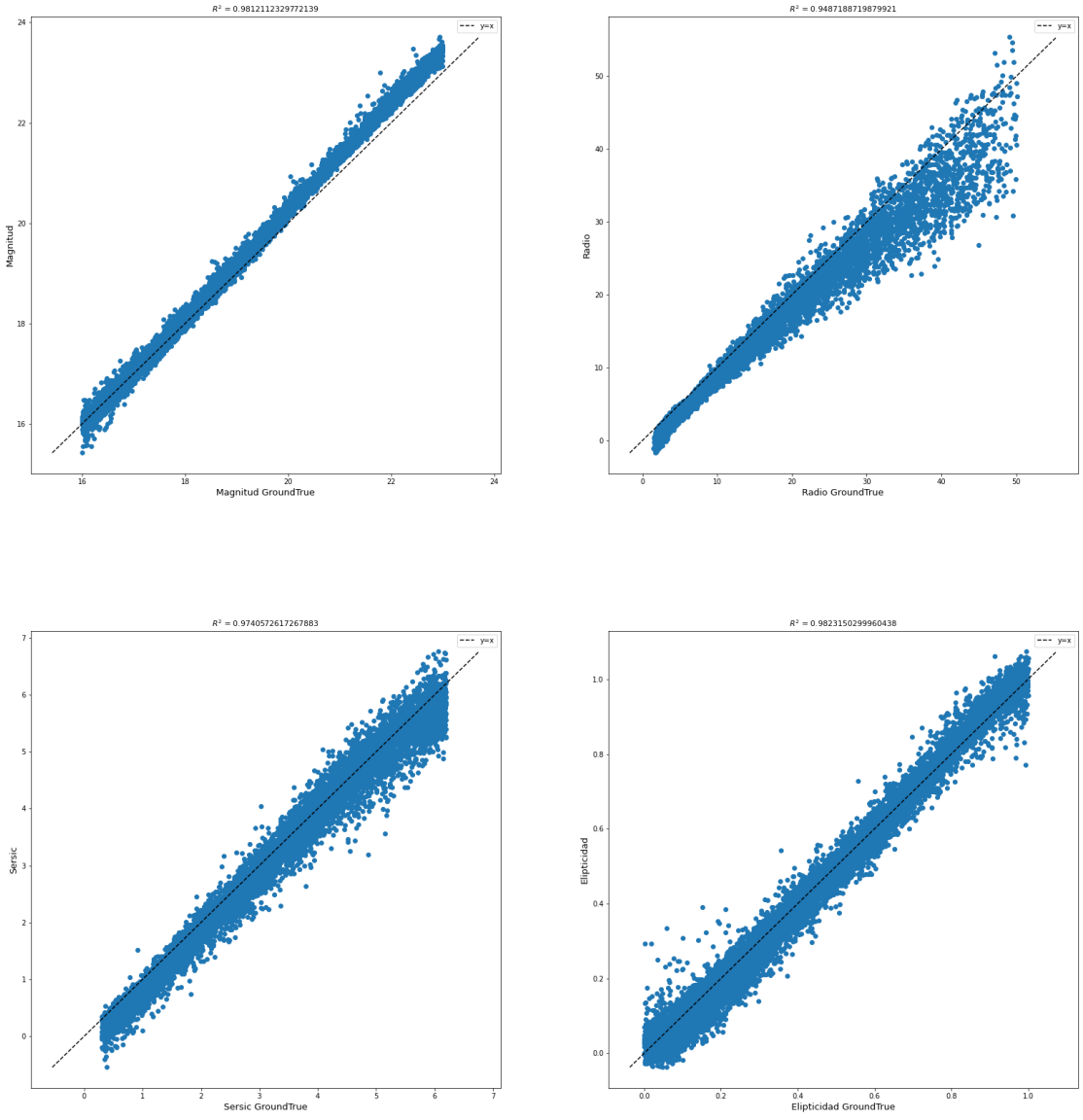

Figura 4.2: Se muestra la predicción vs el ground true en el conjunto de prueba de galaxias simuladas.

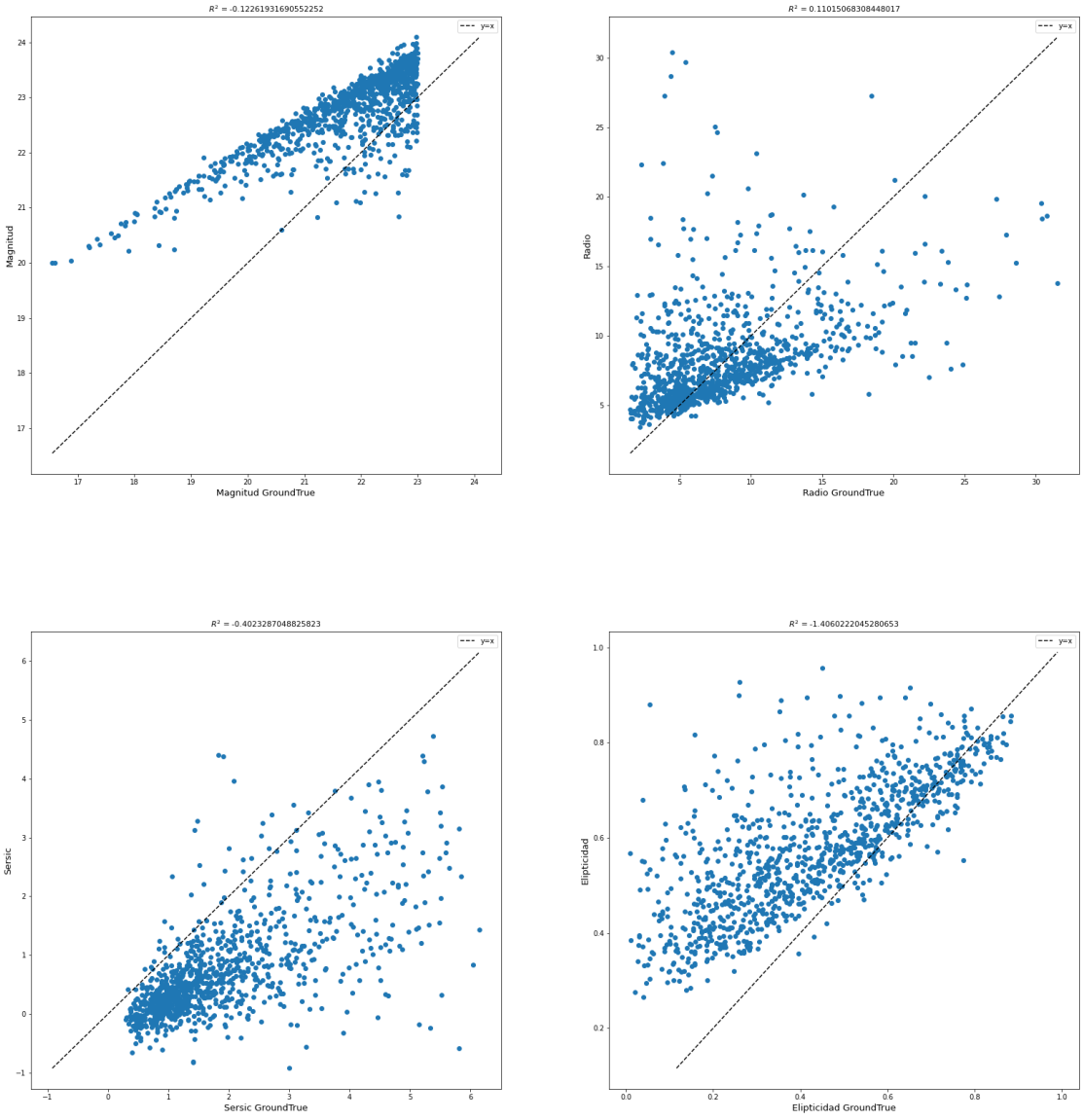

Figura 4.3: Se muestra la predicción vs el ground true (GALFIT) en el conjunto de prueba de galaxias reales antes de aplicar transfer learning.

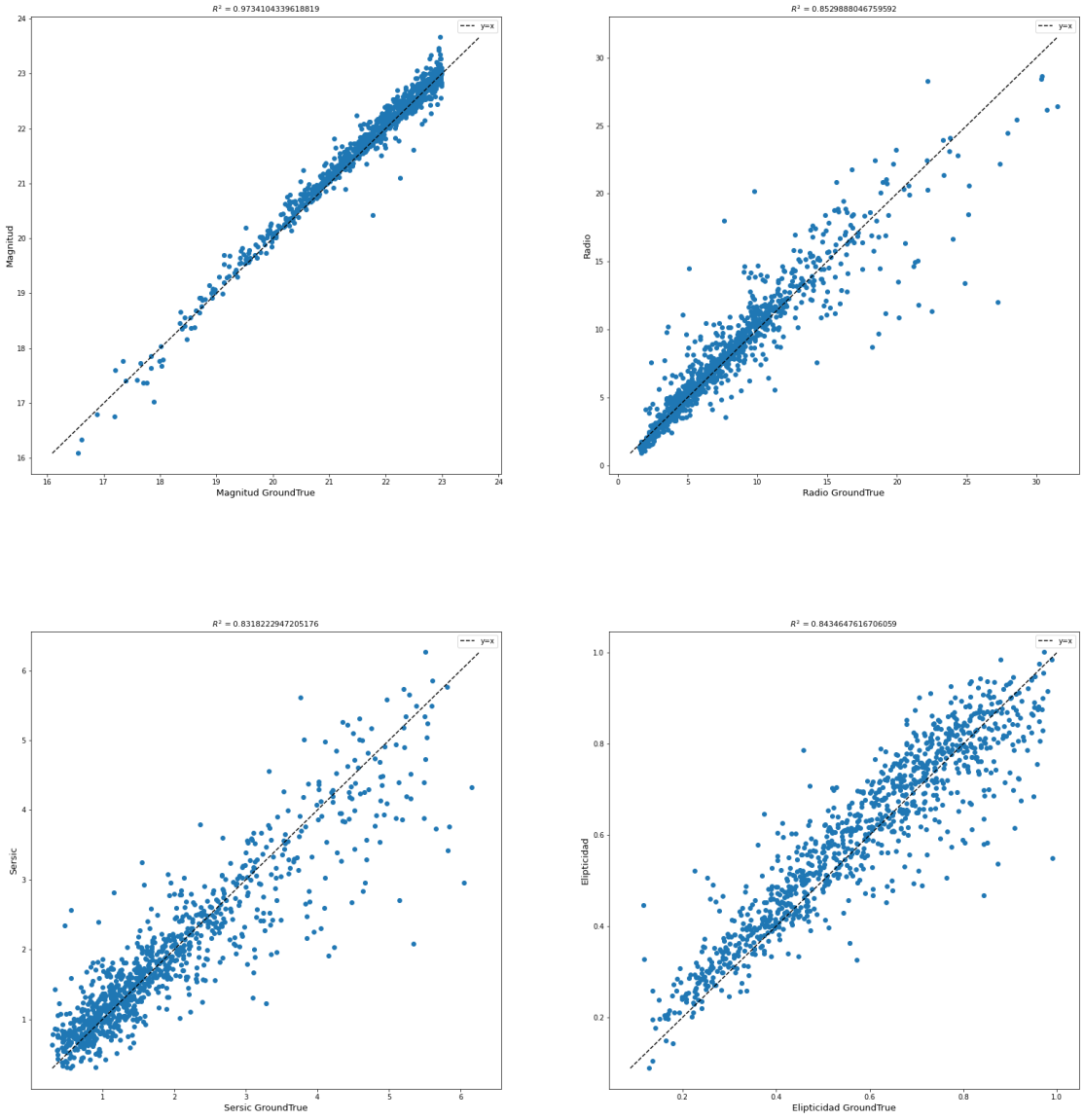

Figura 4.4: Se muestra la predicción vs el ground true (GALFIT) en el conjunto de prueba de galaxias reales después de aplicar transfer learning.

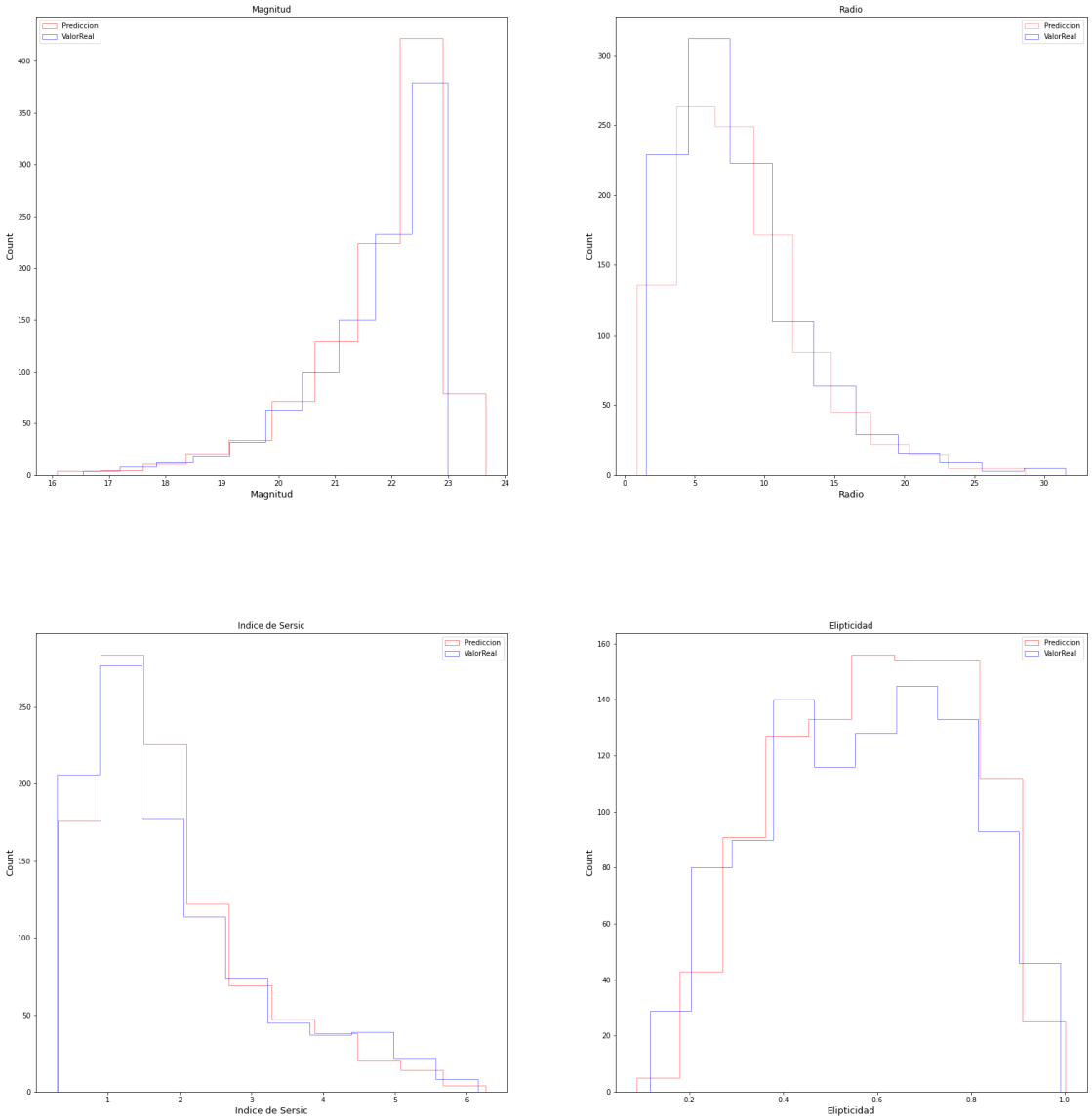

Figura 4.5: Se muestra las distribuciones de los parámetros predichas por el modelo(en rojo) junto a las distribuciones reales(en azul) en el conjunto de prueba de galaxias reales.

5. Adversarial Domain Adaptation

5.1. Arquitectura

La arquitectura del modelo usado en esta sección puede verse en el esquema de la figura 5.1. La capa de entrada tiene el mismo tamñano que las imágenes, 128x128 en este caso, seguido de una capa de batch normalization, varias capas convolucionales (todas con función de activación relu) intercaladas con capas de max pooling hasta llegar a la capa de flatten o aplanamiento donde el modelo se divide en dos partes, por un lado 3 capas de neuronas densas que se encargan de la regresión y por otro lado el clasificador que cuenta con una capa personalizada denominada GradientReversalLayer que se encarga de invertir la dirección del gradiente seguido de 3 capas de neuronas densas que predicen la clase de la imagen(real o simulada). La definición de la capa GradientReversalLayer se muestra en 5.1, mientras que la implementación del modelo puede encontrarse en el apéndice.

Listing 5.1: Se muestra la implementación en Python de la capa personalizada para invertir la dirección del gradiente.

```
class GradientReversalLayer(tf.keras.layers.Layer):
    def __init__(self, **kwargs):
        super(GradientReversalLayer, self).__init__()

    def call(self, inputs, training=None):
        if training:
            return -1.0 * inputs
        return inputs
```

5.2. Entrenamiento y Testeo

El modelo se entrena y prueba usando los conjuntos de datos descritos en 3.3 y 3.2. Primero se normaliza el valor de los píxeles de las imágenes dividiendo entre 255, los parámetros se normalizan usando la función StandardScaler() de la librería sklearn. En este caso no se filtran las galaxias por magnitud, ya que si se hace el total de galaxias reales es demasiado bajo. Se crea un conjunto de entrenamiento compuesto de 4000 galaxias reales y 5000 galaxias simuladas mezcladas aleatoriamente. Como conjunto de prueba se reservan las 1000 galaxias reales restantes y el conjunto de validación será el 20 % del conjunto de entrenamiento.
Se entrena el modelo usando la técnica de early-stoping monitorizando el error cuadrático medio del conjunto de validación con una paciencia de 20 épocas y un máximo de 300 épocas de entrenamiento. En la tabla 5.1 se muestran los hiper-parámetros utilizados en este entrenamiento. El parámetro α que aparece en la tabla 5.1 regula la importancia de cada una de las dos tareas (regresión y clasificación) en la función de coste combinada que se optimiza durante el entrenamiento, en este caso:

$$f = \alpha * mse + (1 - \alpha) * bce \tag{5.1}$$

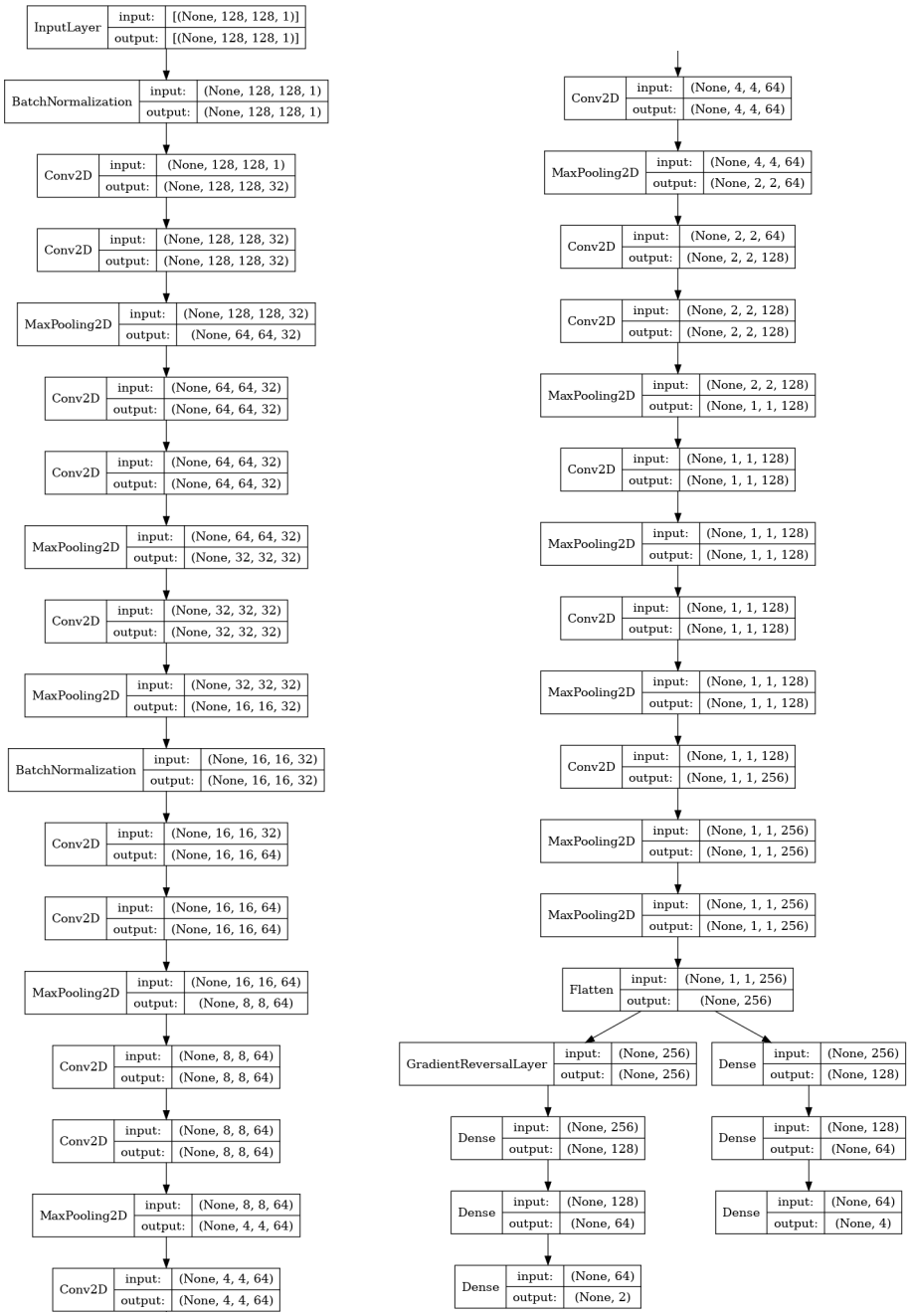

Figura 5.1: Se muestra la arquitectura del modelo utilizado en la sección 5 para aplicar adversarial domain adaptation.

Siendo f la función de coste total, mse el error cuadrático medio que se utiliza en el regresor y bce la entropía cruzada binaria que se utiliza en el clasificador.

Learning Rate	Batch Size	α
0.0001	16	0.9

Tabla 5.1: Se muestra los hiper-parámetros utilizados durante el entrenamiento de la red con imágenes simuladas

5.2.1. Supervisado

Una vez el modelo ha sido entrenado se prueba sobre el conjunto de prueba de galaxias reales obteniendo los resultados mostrados en la tabla 5.2, donde puede verse que se obtienen buenos resultados sobre galaxias reales aunque la precisión es menor que con el modelo de transfer learning (tabla 4.4). En la figura 5.2 se muestran la predicción del modelo frente al ground true, mientras que

R^2 Real			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
0.9425	0.8217	0.7608	0.7743

Tabla 5.2: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias reales

en la figura 5.3 se comparan las distribuciones de los parámetros predichas por el modelo con las distribuciones obtenidas por GALFIT (ground true) para el conjunto de prueba de galaxias reales. En ambas figuras se observa que los resultados siguen siendo buenos aunque presentan algo más de dispersión que el modelo de transfer learning.
Sin embargo, es importante destacar que mientras el modelo de transfer learning se adapta en el segundo entrenamiento a las galaxias reales, el modelo de adversarial domain adaptation aprende directamente características presentes en ambos dominios (real y simulado), por tanto si el mismo modelo que da buenos resultados en galaxias reales se prueba en simuladas debería de ser capaz de dar buenos resultados también. Los resultados de probar el modelo sobre un conjunto de 10,000 galaxias simuladas se muestran en la tabla 5.3. Adicionalmente en la figura 5.4 se representa la predicción frente al ground true sobre el conjunto de galaxias simuladas. De esta manera se comprueba que aplicando adversarial domain adaptation se consigue que el modelo aprenda características comunes a los dos dominios y es capaz de dar buenos resultados en ambos.

R^2 Simulado			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
0.9808	0.8341	0.8247	0.8522

Tabla 5.3: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias simuladas

5.2.2. Semi-Supervisado

En la vertiente semi-supervisada del adversarial domain adaptation se utiliza el mismo modelo que en la supervisada y los mismos hiper-parámetros de la tabla 5.1. La única diferencia es que

en este caso se utiliza una función de coste personalizada para la parte de regresión, la función personalizada calcula el error cuadrático medio y lo multiplica por un vector que vale 1 si la galaxia es simulada y 0 si la galaxia es real, de modo que durante el entrenamiento se anula el gradiente si la galaxia es real, que es equivalente a que la red no sepa cuales son los valores reales de los parámetros de la galaxia reales. La implementación de la función de coste personalizada se muestra en 5.2.

Listing 5.2: Se muestra la implementación en Python de la funcion de coste personalizada para anular el gradiente en galaxias reales.

```
class CustomLoss(Loss):
    def __init__(self, mask, **kwargs):
        super(CustomLoss, self).__init__(**kwargs)
        self.mask = mask

    def call(self, y_true, y_pred):
        mse = K.mean(K.square(y_true - y_pred), axis=-1)
        return mse * self.mask
```

Una vez se ha entrenado el modelo, se prueba sobre galaxias reales y se obtienen los resultados de la tabla 5.4. Exceptuando la magnitud, se aprecia una disminución de la precisión respecto a los resultados supervisados (tabla 5.2), sobre todo en el índice de Sersic. Esto puede deberse a que se trata del parámetro más abstracto de los 4 y generalmente es el más difícil de predecir.

R^2 Real			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
0.9493	0.7064	0.5720	0.7221

Tabla 5.4: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias reales

En la figura 5.5 se representa la predicción del modelo semi-supervisado frente al ground true y se observa como en este caso la dispersión es mayor, sobre todo en el índice de Sersic donde se ve una nube de puntos alrededor de la recta diagonal y=x. En la figura 5.6 se muestra la comparativa de las distribuciones. A pesar de los resultados de la tabla 5.4 el modelo es capaz de predecir las formas de las distribuciones aunque obviamente hay mayores discrepancias que con el método supervisado (5.3).

Una de las posibles razones por las que la precisión del modelo ha disminuido al realizar un entrenamiento semi-supervisado es que la red al poder aprender directamente de galaxias simuladas se ha quedado con características específicas de las galaxias simuladas, para comprobarlo se prueba el modelo en el mismo conjunto de galaxias simuladas en el que se probó el modelo supervisado obteniéndose los resultados de la tabla 5.5. Puede apreciarse que efectivamente el modelo es capaz de predecir con mucha precisión en galaxias simuladas lo que indica que ha aprendido características especificas de este dominio a costa de perder precisión en las galaxias reales.

En la figura 5.7 se muestra la predicción del modelo frente al ground true para galaxias simuladas y se aprecia que efectivamente, el modelo se adecua de manera muy precisa a los valores reales de los parámetros.

R^2 Simulado			
Magnitud	Radio Efectivo	Índice de Sersic	Elipticidad
0.9939	0.9230	0.9231	0.9748

Tabla 5.5: Se muestra el valor de R^2 obtenido sobre el conjunto de prueba de galaxias simuladas

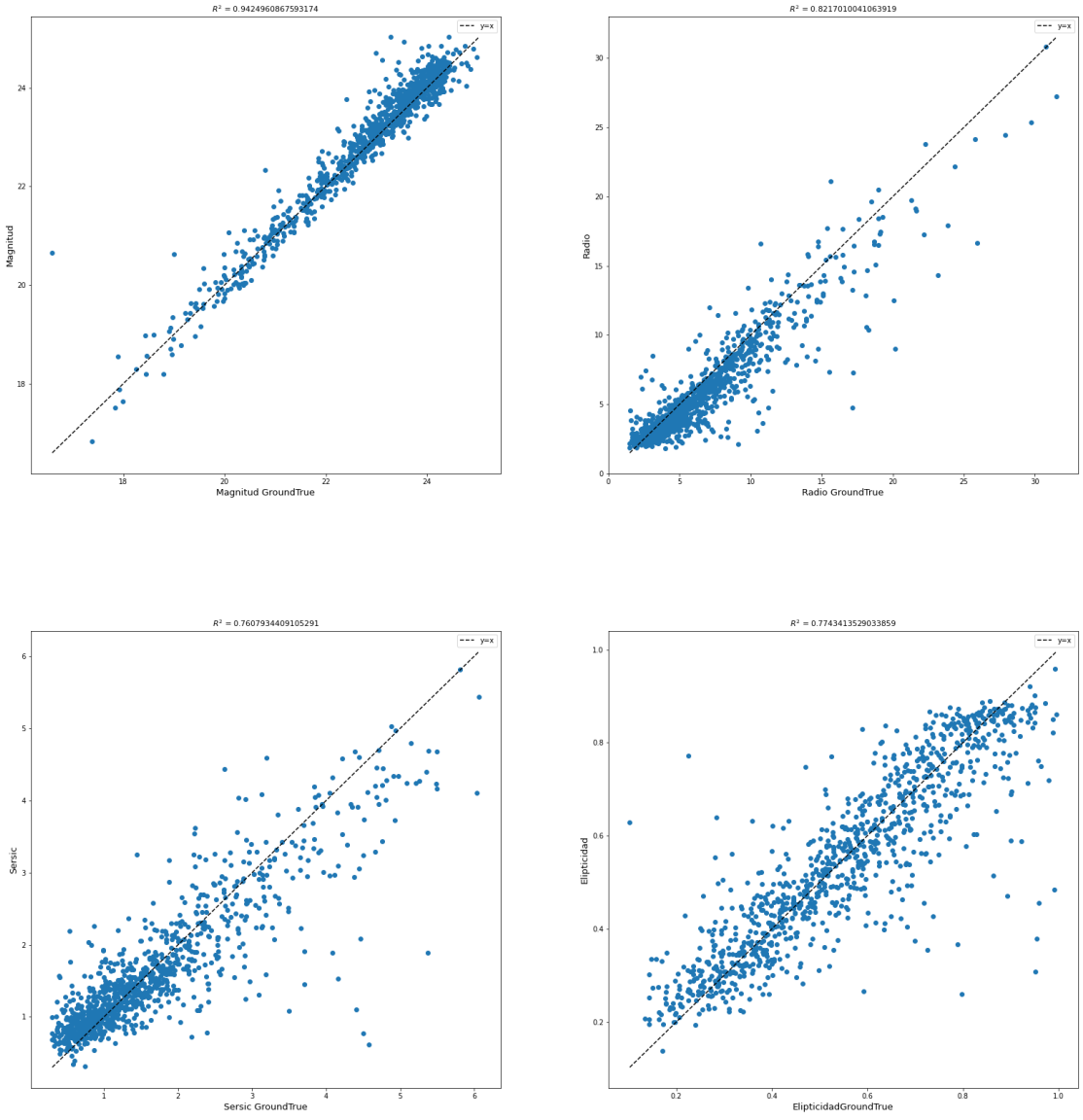

Figura 5.2: Se muestra la predicción vs el ground true en el conjunto de prueba de galaxias reales.

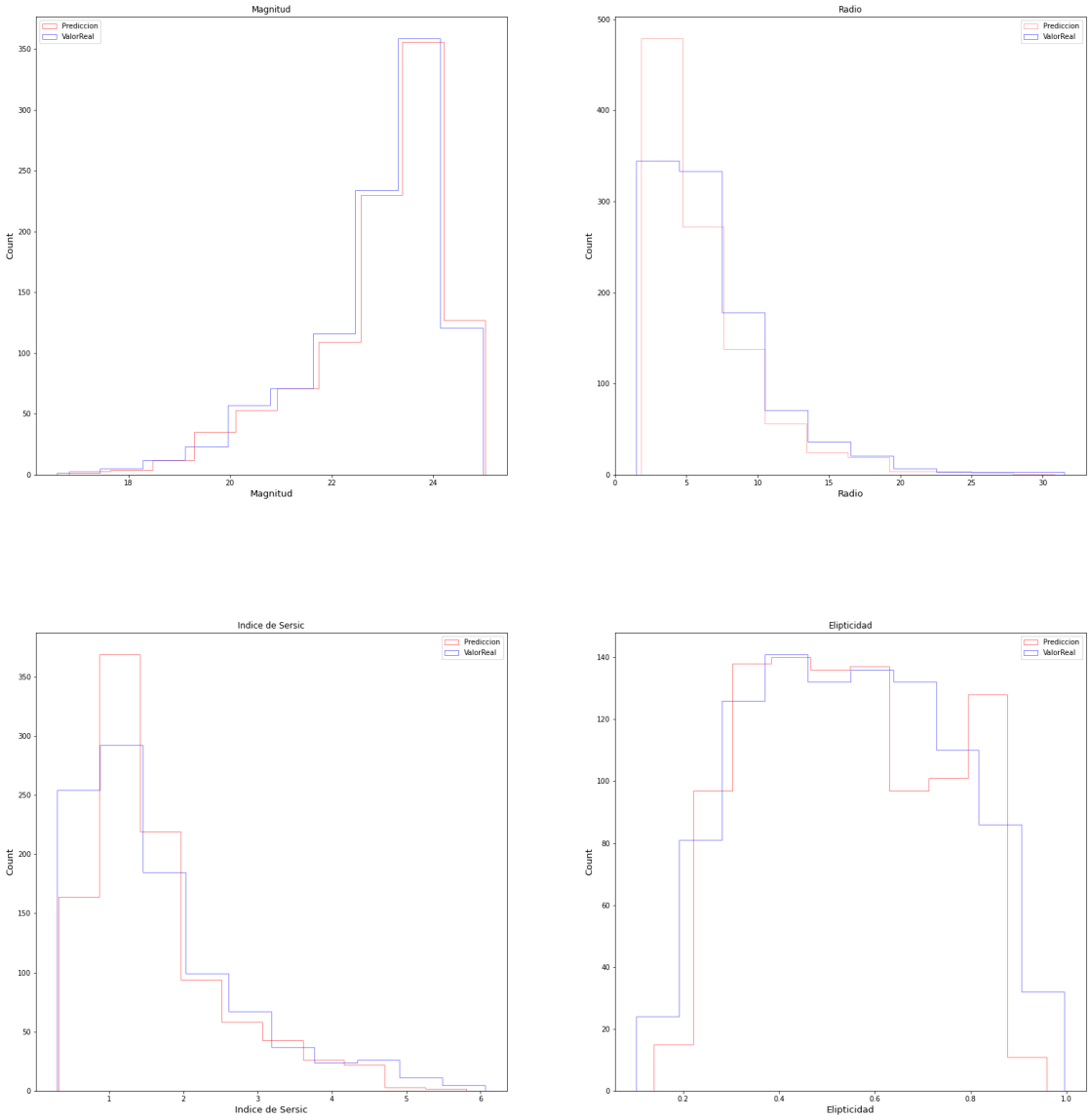

Figura 5.3: Se muestra las distribuciones de los parámetros predichas por el modelo(en rojo) junto a las distribuciones reales(en azul) en el conjunto de prueba de galaxias reales.

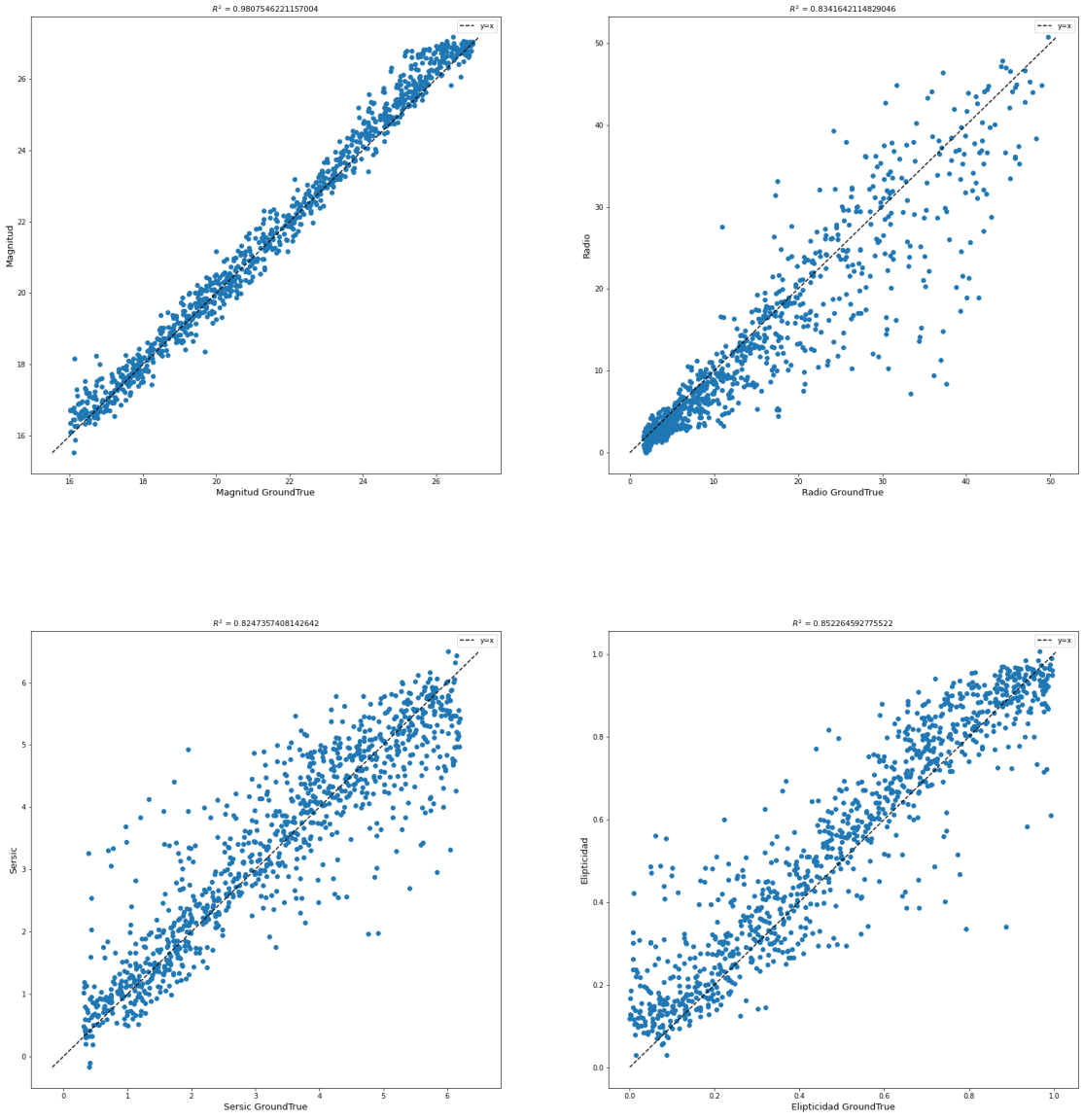

Figura 5.4: Se muestra la predicción vs el ground true en el conjunto de prueba de galaxias simuladas.

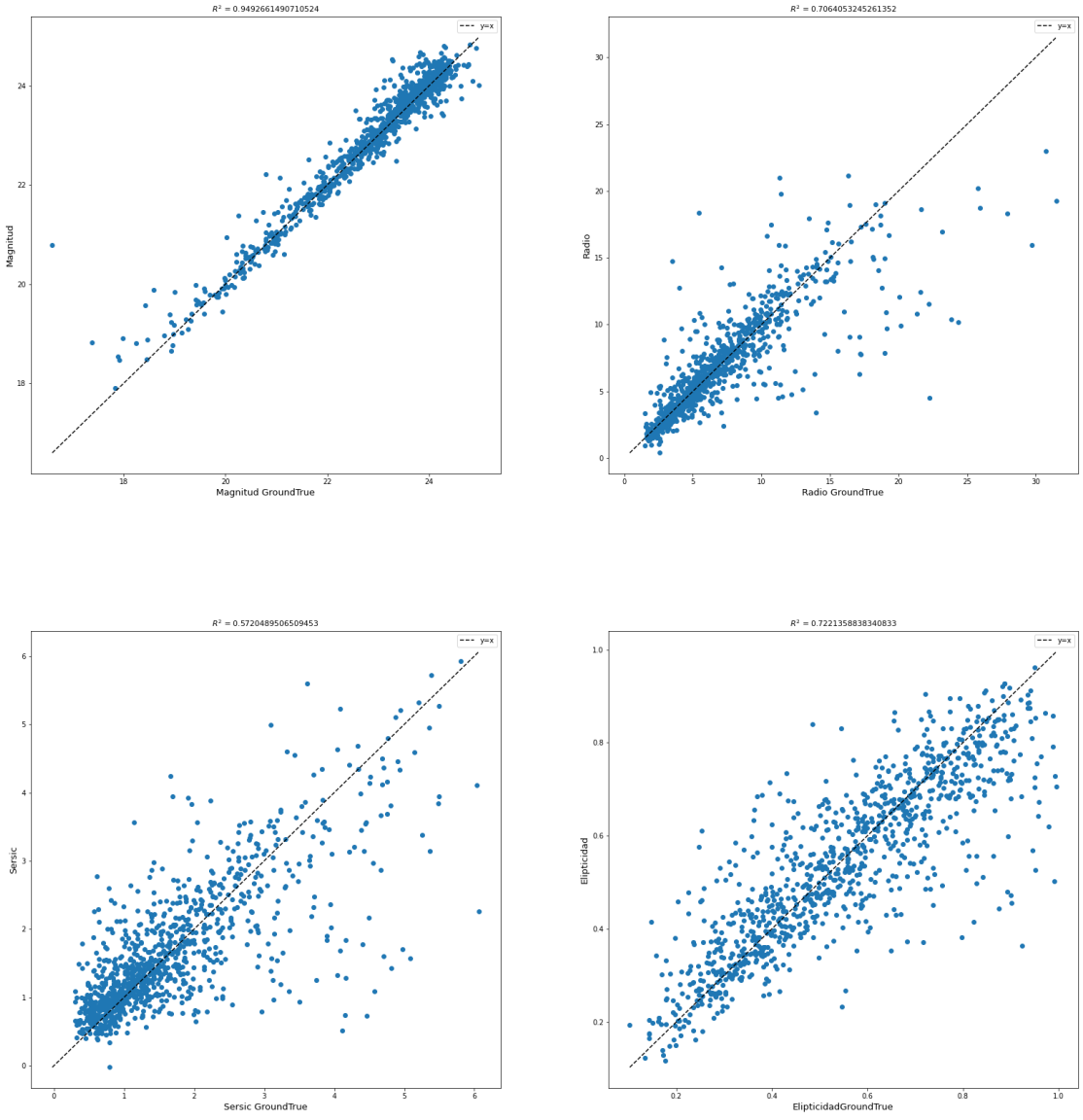

Figura 5.5: Se muestra la predicción vs el ground true en el conjunto de prueba de galaxias reales.

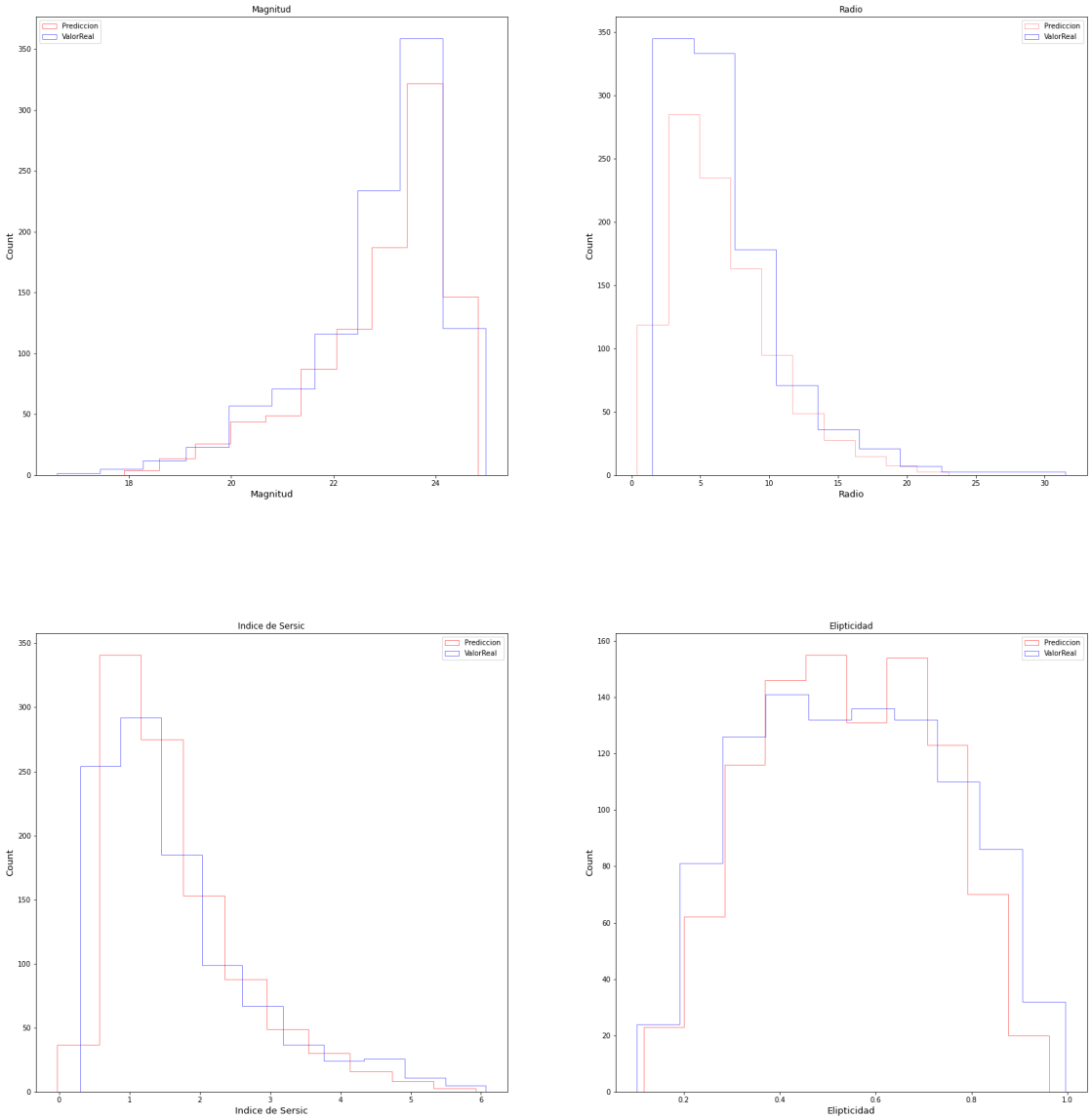

Figura 5.6: Se muestra las distribuciones de los parámetros predichas por el modelo(en rojo) junto a las distribuciones reales(en azul) en el conjunto de prueba de galaxias reales.

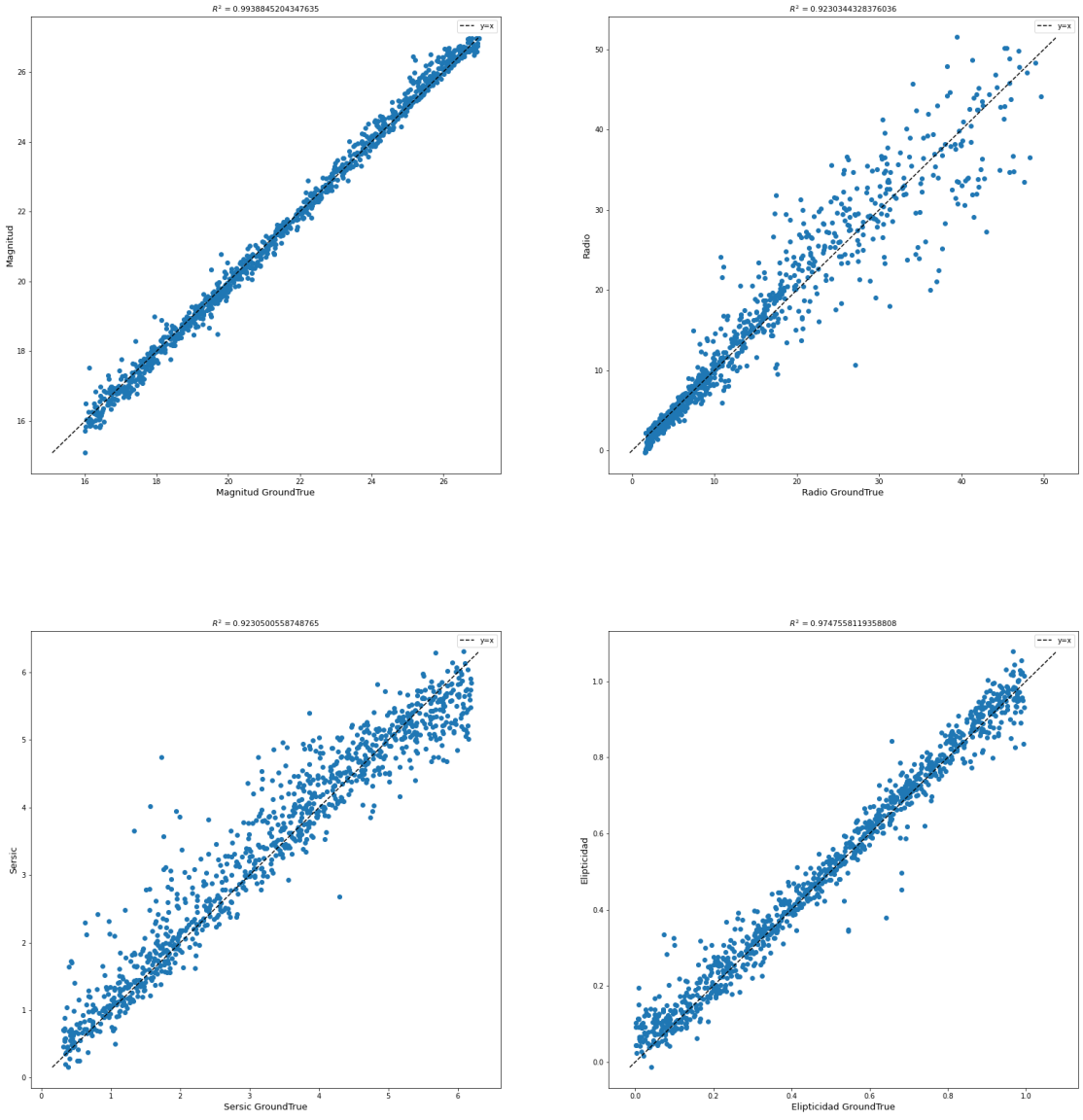

Figura 5.7: Se muestra la predicción vs el ground true en el conjunto de prueba de galaxias simuladas.

Conclusiones

En este trabajo se han estudiado diferentes técnicas de adaptación de dominio, con el objetivo de estudiar si pueden ser útiles de cara a resolver el problema del domain shift. Los mejores resultados sobre galaxias reales se obtuvieron con transfer learning.
Teniendo en cuenta las limitaciones del trabajo, como por ejemplo, usar un conjunto de datos similar al que se usó en DeepLegato, que está optimizado para transfer learning, se puede concluir que el adversarial domain adaptatión es una técnica capaz de dar buenos resultados e interesante de cara a futuros trabajos.
Se ha comprobado que mediante adaptación de dominio adversaria supervisada se puede obtener un modelo basado en características comunes a dos dominios capaz de dar buenos resultados en galaxias reales y simuladas.
Una posible manera de mejorar los modelos de adversarial domain adaptation es utilizar un conjunto de galaxias reales más grande. En este trabajo solo se usaron 4000 galaxias reales para el entrenamiento porque se disponía de 5000 galaxias reales en total. Sería interesante probar con conjuntos de entrenamiento más desbalanceados hacia las galaxias reales, por ejemplo, 10,000 galaxias reales y 5000 galaxias simuladas.
Aunque se obtienen mejores resultados con el modelo supervisado de adversarial domain adaptation que son el semi-supervisado, hay que tener en cuenta que este último elimina la necesidad de utilizar otro algoritmo como GALFIT para predecir de antemano los parámetros de las galaxias reales. Además se ha usado el mismo modelo para los dos casos, probablemente si se optimizara un modelo específicamente para el aprendizaje semi-supervisado se obtendrán mejores resultados. Por ello sería interesante explorar en futuros trabajos, con conjuntos de datos más grandes, hasta donde se puede optimizar esta técnica.
Por último, a la vista de que es posible obtener buenos resultados con métodos semi-supervisados, el siguiente paso es utilizar modelos no supervisados, que idealmente permiten analizar las galaxias según se van observando de manera totalmente automática e independiente de los resultados de otros algoritmos. Sin embargo para ello hay que construir modelos nuevos y totalmente diferentes a los usados en este trabajo por ello se deben estudiar en futuros trabajos dedicados a este tema.

Bibliografía

[1] *Hubble Space Telescope Images*. https://www.nasa.gov/mission_pages/hubble/multimedia/index.html. Accessed: 2023-07-02.

[2] *The Hubble Tuning Fork*. https://cdn.sci.news/images/enlarge6/image_7280e-Hubble-Tuning-Fork.jpg. Accessed: 2023-07-05.

[3] *AST222 Winter 2006 Supplementary Notes: Galaxy Surface Brightness Profiles*. https://www.studocu.com/in/document/central-university-of-rajasthan/introduction-to-electrodynamics/dubinski-sb/34657807. Accessed: 2023-07-02.

[4] Chien Y. Peng et al. "Detailed Structural Decomposition of Galaxy Images". En: *The Astronomical Journal* 124.1 (jul. de 2002), pág. 266. DOI: 10.1086/340952. URL: https://dx.doi.org/10.1086/340952.

[5] Luc Simard et al. "The DEEP Groth Strip Survey. II. Hubble Space Telescope Structural Parameters of Galaxies in the Groth Strip". En: *The Astrophysical Journal Supplement Series* 142.1 (sep. de 2002), pág. 1. DOI: 10.1086/341399. URL: https://dx.doi.org/10.1086/341399.

[6] Vinu Vikram et al. "PyMorph: automated galaxy structural parameter estimation using python". En: *Monthly Notices of the Royal Astronomical Society* 409.4 (sep. de 2010), págs. 1379-1392. DOI: 10.1111/j.1365-2966.2010.17426.x. URL: https://doi.org/10.1111%5C%2Fj.1365-2966.2010.17426.x.

[7] A. S. G. Robotham et al. "ProFit: Bayesian profile fitting of galaxy images". En: *Monthly Notices of the Royal Astronomical Society* 466.2 (nov. de 2016), págs. 1513-1541. ISSN: 0035-8711. DOI: 10.1093/mnras/stw3039. eprint: https://academic.oup.com/mnras/article-pdf/466/2/1513/10867054/stw3039.pdf. URL: https://doi.org/10.1093/mnras/stw3039.

[8] D Tuccillo et al. "Deep learning for galaxy surface brightness profile fitting". En: *Monthly Notices of the Royal Astronomical Society* 475.1 (dic. de 2017), págs. 894-909. DOI: 10.1093/mnras/stx3186. URL: https://doi.org/10.1093%5C%2Fmnras%5C%2Fstx3186.

[9] Hiroaki Aihara et al. "The Hyper Suprime-Cam SSP Survey: Overview and survey design". En: *Publications of the Astronomical Society of Japan* 70.SP1 (sep. de 2017). S4. ISSN: 0004-6264. DOI: 10.1093/pasj/psx066. eprint: https://academic.oup.com/pasj/article-pdf/70/SP1/S4/23692189/psx066.pdf. URL: https://doi.org/10.1093/pasj/psx066.

[10] Norman A. Grogin et al. "CANDELS: THE COSMIC ASSEMBLY NEAR-INFRARED DEEP EXTRAGALACTIC LEGACY SURVE". En: *The Astrophysical Journal Supplement Series* 197.2 (dic. de 2011), pág. 35. DOI: 10.1088/0067-0049/197/2/35. URL: https://dx.doi.org/10.1088/0067-0049/197/2/35.

[11] Kevork N. Abazajian et al. "THE SEVENTH DATA RELEASE OF THE SLOAN DIGITAL SKY SURVEY". En: *The Astrophysical Journal Supplement Series* 182.2 (mayo de 2009), pág. 543. DOI: 10.1088/0067-0049/182/2/543. URL: https://dx.doi.org/10.1088/0067-0049/182/2/543.

[12] N. Scoville et al. "The Cosmic Evolution Survey (COSMOS): Overview*". En: *The Astrophysical Journal Supplement Series* 172.1 (sep. de 2007), pág. 1. DOI: 10.1086/516585. URL: https://dx.doi.org/10.1086/516585.

[13] R. Laureijs et al. *Euclid Definition Study Report*. 2011. arXiv: 1110.3193 [astro-ph.CO].

[14] Anna Bonaldi et al. "SKA weak lensing– II. Simulated performance and survey design considerations". En: *Monthly Notices of the Royal Astronomical Society* 463.4 (ago. de 2016), págs. 3686-3698. ISSN: 0035-8711. DOI: 10.1093/mnras/stw2104. eprint: https://academic.oup.com/mnras/article-pdf/463/4/3686/18518262/stw2104.pdf. URL: https://doi.org/10.1093/mnras/stw2104.

[15] *Hubble Space Telescope NASA*. https://www.nasa.gov/mission_pages/hubble/main/index.html. Accessed: 2023-07-05.

[16] G. Longo et al. *Data Driven Discovery in Astrophysics*. 2014. arXiv: 1410.5631 [astro-ph.IM].

[17] M. Huertas-Company y F. Lanusse. "The Dawes Review 10: The impact of deep learning for the analysis of galaxy surveys". En: *Publications of the Astronomical Society of Australia* 40 (2023). DOI: 10.1017/pasa.2022.55. URL: https://doi.org/10.1017%2Fpasa.2022.55.

[18] Keiron O'Shea y Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[19] Alex Krizhevsky, Ilya Sutskever y Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". En: *Advances in Neural Information Processing Systems*. Ed. por F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[20] Sander Dieleman, Kyle W. Willett y Joni Dambre. "Rotation-invariant convolutional neural networks for galaxy morphology prediction". En: *Monthly Notices of the Royal Astronomical Society* 450.2 (abr. de 2015), págs. 1441-1459. ISSN: 0035-8711. DOI: 10.1093/mnras/stv632. eprint: https://academic.oup.com/mnras/article-pdf/450/2/1441/3022697/stv632.pdf. URL: https://doi.org/10.1093/mnras/stv632.

[21] Karin Stacke et al. "Measuring Domain Shift for Deep Learning in Histopathology". En: *IEEE Journal of Biomedical and Health Informatics* 25.2 (feb. de 2021), págs. 325-336. ISSN: 2168-2208. DOI: 10.1109/JBHI.2020.3032060.

[22] A. van der Wel et al. "STRUCTURAL PARAMETERS OF GALAXIES IN CANDELS". En: *The Astrophysical Journal Supplement Series* 203.2 (nov. de 2012), pág. 24. DOI: 10.1088/0067-0049/203/2/24. URL: https://dx.doi.org/10.1088/0067-0049/203/2/24.

[23] BTP Rowe, M Jarvis, R Mandelbaum et al. "Astronomy and Computing, 10, 121". En: *arXiv preprint arXiv:1407.7676* (2015).

Apéndice

Modelo de Transfer Learning

Listing 5.3: Se muestra la implementación en Python del modelo utilizado para el transfer learning.

```
# Definir la arquitectura del modelo
    model = keras.Sequential(
        [
            layers.BatchNormalization(),
            layers.Conv2D(16, (4, 4), activation="relu", input_shape=(128, 128,
                1),kernel_regularizer=regularizers.l2(0.0001)),
            layers.Conv2D(16, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Conv2D(32, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.Conv2D(32, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.BatchNormalization(),
            layers.Conv2D(64, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.Conv2D(64, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.MaxPooling2D(pool_size=(2, 2), padding='same'),
            layers.Conv2D(128, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.Conv2D(128, (4, 4), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.MaxPooling2D(pool_size=(2, 2), padding='same'),
            layers.Conv2D(256, (3, 3), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.Conv2D(256, (3, 3), activation="relu",
                padding='same',kernel_regularizer=regularizers.l2(0.0001)),
            layers.MaxPooling2D(pool_size=(2, 2), padding= 'same'),
            layers.Flatten(),
            layers.Dropout(0.0001),
            layers.Dense(128, activation="relu"),
            layers.Dropout(0.0001),
            layers.Dense(64, activation="relu"),
            layers.Dropout(0.0001),
            layers.BatchNormalization(),
            layers.Dense(4, activation="relu"),
        ]
    )
```

Modelo de Adversarial Domain Adaptation

Listing 5.4: Se muestra la implementación en Python del modelo utilizado para el adversarial domain adaptation.

```
def build_model(input_shape, num_params, num_classes):
    inputs = Input(shape=input_shape, name='input')
    norm1=BatchNormalization()(inputs)
    conv1 = Conv2D(32, (4, 4), activation="relu",padding =
        'same',kernel_initializer='he_normal')(norm1)
    conv2 = Conv2D(32, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal')(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(32, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (pool1)
    conv4 = Conv2D(32, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (conv3)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv4)
    conv5 = Conv2D(32, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (pool2)
    pool3 = layers.MaxPooling2D(pool_size=(2, 2), padding = 'same') (conv5)
    norm2=BatchNormalization()(pool3)
    conv7 = Conv2D(64, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (norm2)
    conv8 = Conv2D(64, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (conv7)
    pool4 = layers.MaxPooling2D(pool_size=(2, 2), padding= 'same') (conv8)
    conv9 = Conv2D(64, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (pool4)
    conv10 = Conv2D(64, (4, 4), activation="relu",
        padding='same',kernel_initializer='he_normal') (conv9)
    pool5 = layers.MaxPooling2D(pool_size=(2, 2), padding= 'same') (conv10)
    conv11 = Conv2D(64, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal') (pool5)
    conv12 = Conv2D(64, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal') (conv11)
    pool6 = layers.MaxPooling2D(pool_size=(2, 2), padding='same') (conv12)
    conv13 = Conv2D(128, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal') (pool6)
    conv14 = Conv2D(128, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal')(conv13)
    pool7 = layers.MaxPooling2D(pool_size=(2, 2), padding='same')(conv14)
    conv15 = Conv2D(128, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal')(pool7)
    pool8 = layers.MaxPooling2D(pool_size=(2, 2), padding='same')(conv15)
    conv16 = Conv2D(128, (3, 3), activation="relu",
        padding='same',kernel_initializer='he_normal')(pool8)
    pool9 = layers.MaxPooling2D(pool_size=(2, 2), padding='same')(conv16)
    conv17 = Conv2D(256, (3, 3), activation="relu", padding='same')(pool9)
```

```
pool10 = layers.MaxPooling2D(pool_size=(2, 2), padding = 'same')(conv17)
pool12 = layers.MaxPooling2D(pool_size=(2, 2), padding= 'same') (pool10)
features = Flatten()(pool12)

# Cabeza de regresion para los parmetros
dense1 = Dense(128, activation="relu")(features)
dense2 = Dense(64, activation="relu")(dense1)
regression_output = Dense(num_params, activation='linear',
    name='regression_output')(dense2)

# Cabeza de clasificacion para el discriminador (real o simulado)
x_grl = GradientReversalLayer()(features)
dense3 = Dense(128, activation="relu")(x_grl)
dense4 = Dense(64, activation="relu")(dense3)
discriminator_output = Dense(num_classes, activation='softmax',
    name='discriminator_output')(dense4)
optimizer = Adam(learning_rate=0.0001)
modelRegresion = Model(inputs=inputs, outputs=regression_output)
modelRegresion.compile(optimizer=optimizer,loss='mean_absolute_error',metrics=['mean_absolute_error'])
modelDiscriminator = Model(inputs=inputs, outputs=discriminator_output)
modelDiscriminator.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
optimizer = Adam(learning_rate=0.0001)
alpha= 0.9
model = Model(inputs=inputs, outputs=[regression_output, discriminator_output])
model.compile(optimizer=optimizer, loss=[['mean_squared_error'],
    ['binary_crossentropy']],
    metrics=[['mean_squared_error'],['accuracy']],loss_weights=[alpha, 1 - alpha])

return model, modelRegresion, modelDiscriminator
```