UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE POSGRADO EN ASTROFÍSICA

INSTITUTO DE ASTRONOMÍA

ESTUDIO DE PROCESOS FÍSICOS DE FORMACIÓN ESTELAR CON REDES NEURONALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE

MAESTRO EN CIENCIAS (ASTROFÍSICA)

PRESENTA:

JOSÉ CARLOS CARVAJAL GARCÍA

TUTOR:

DR. MIGUEL ÁNGEL ARAGÓN CALVO

INSTITUTO DE ASTRONOMÍA, UNAM.

Ensenada, Baja California, México. Mayo 2022

Resumen

En este trabajo se utilizaron técnicas de inteligencia artificial para entender aspectos específicos de los procesos físicos que actúan en la formación de estrellas en galaxias. Estos procesos son en general difíciles de estudiar debido a las escalas donde estos ocurren, limitando los estudios a regiones individuales de formación estelar en la Vía Láctea. Por lo que se hace uso de redes neuronales convolucionadas aprovechando la capacidad de éstas para identificar patrones y características en imágenes, como propuesta para hacer un estudio de los procesos de formación estelar en galaxias simuladas. Se generaron conjuntos de datos de imágenes sintéticas de mapas de temperatura, masa y entropía del gas en 7881 galaxias a partir de la simulación hidrodinámica Illustris TNG 50. Para esto, se hicieron un total de 10 conjunto de datos de entrenamiento con variaciones en propiedades, escalas físicas y estructuras internas, con la finalidad de determinar las propiedades físicas de las cuales la red está obteniendo la información de la formación estelar al compararlos con los valores obtenidos directamente de la simulación. Los resultados de utilizar esta técnica fueron satisfactorios, en condiciones iguales de tiempos de entrenamiento y arquitectura de la red neuronal, se encontró que mapas de densidad de masa y temperatura, en combinación con las estructuras de escalas pequeñas en galaxias es donde la red obtiene mejores resultados, sugiriendo que procesos físicos en el gas a escalas menores de ~ 1 kpc$/h$ juegan un papel relevante en la formación de estrellas.

Abstract

In this work, artificial intelligence techniques were used to understand specific aspects of the physical processes that act in the formation of stars in galaxies. These processes are generally difficult to study due to the scales on which they occur, limiting studies to individual star-forming regions in the Milky Way. In order to solve this limitation convolutional neural networks were used, taking advantage of their ability to identify patterns and characteristics in images, as a proposal for a study of star formation processes in simulated galaxies. Datasets of synthetic images of temperature, mass and gas entropy maps were generated for 7881 galaxies from the Illustris TNG hydrodynamic simulation. From these galaxies, a total of 10 training datasets were made, which have variations in physical properties, scales and structures, in order determine the physical qualities from which the network is obtaining the star formation information by comparing them with the values obtained directly from the simulation. The results of using this technique were satisfactory, in equal conditions of training times and neural network arquitecture, it was found that mass density and temperature maps, in combination with the small scale structures in galaxies is where the network obtains better results. This suggest that physical process of scales smaller than ~ 1 kpc$/h$ play a role in the process of star formation.

Agradecimientos

A mi asesor, Dr. Miguel Angel Aragon Calvo. Por todo su apoyo, conocimiento, tiempo, dedicación y paciencia a lo largo del desarrollo de este proyecto.

A los miembros de mi comité sinodal: Dr. Alfredo Montaña Barbano, Dr. Christophe Morisset, Dra. Aida Nava de Wofford, Dr. Benjamín Hernández Valencia y al Dr. Octavio Valenzuela Tijerino por su tiempo, sus comentarios y observaciones en la revisión de esta tesis.

Al Concejo Nacional de Ciencia y Tecnología (CONACyT) por la beca recibida, que sin ésta no habría podido realizar una maestría.

Y la Dirección General de Asuntos del Personal Académico Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (DGAPA- PAPIIT) IA102020.

Índice general

Índice de figuras

CAPÍTULO I

INTRODUCCIÓN

La observación de objetos celestes en astronomía ha ido evolucionando en el tiempo, pasando de recolectar información de un objeto para su estudio detallado, a recabar información mediante telescopios robóticos de miles de objetos al mismo tiempo, por ejemplo el "Sloan Digital Sky Survey" (SDSS) produce cada noche cerca de 200Gbs de información [5]. A su vez, debido a los grandes avances que se han logrado en el ámbito de la computación, las simulaciones en la rama de la astronomía son cada vez más complejas y robustas, tal es el caso del proyecto IllustrisTNG con más de 1.1 PB de información [6], datos de los cuales hablaremos más adelante. Como vemos, conforme pasa el tiempo, la cantidad de datos que se tiene que analizar crece exponencialmente, esto hace poco viable la tarea de revisarlos manualmente, por lo que las técnicas de aprendizaje automático o machine learning (ML por sus siglas en inglés) han comenzando a tener mayor relevancia y con esto nuevas formas de analizar datos e imágenes han ido surgiendo.

En este trabajo se aprovecha la cualidad de las técnicas de ML, en particular de las redes neuronales para identificar patrones, y se hace un estudio con el fin de determinar de dónde es que la red obtiene la información de la formación estelar a partir de imágenes sintéticas de densidad, temperatura y entropía del gas de Galaxias de la simulación Illustris TNG, y con esto poder inferir los procesos físicos que se ven involucrados según los entrenamientos de las redes neuronales y los resultados que nos arrojan.

I.1. Contexto Histórico

I.1.1. Aprendizaje Automático (ML)

El Aprendizaje Automático o Machine Learning (ML) se refiere a un conjunto de algoritmos los cuales mediante la experiencia, sean capaces de aprender y mejorar por si mismos. El ejemplo más sencillo de ML es un modelo de Red Neuronal Artificial o Artificial Neuronal Network (ANN) propuesto por primera vez en 1943 por McCulloch y Pitts [7] simulando el funcionamiento de las neuronas animales Fig 1.1. Este primer trabajo se basa en operaciones simples que contienen parámetros libres, estas a su vez se interconectan unas con otras en una secuencia. Lo que hace esta red es recibir valores de entrada que se ven afectados en estas neuronas y nos arrojan un valor de salida. Entonces, ajustando los parámetros libres se busca, mediante la prueba y error, encontrar la combinación correcta de parámetros libres en que esta red afecte a los valores de entrada de modo que obtengamos en los valores de salida lo que buscamos.

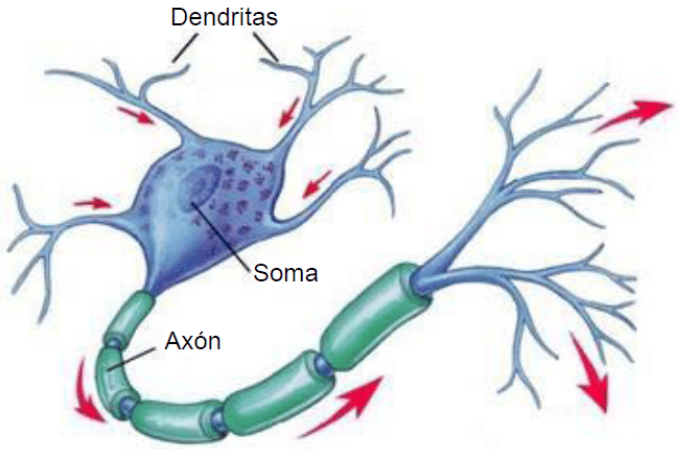

Figura 1.1: Esquema simple de los componentes esenciales en una neurona animal [1]. En las ANN la función de las dendritas son los parámetros de entrada hacia la neurona, la función de activación sustituye a la Soma, y el Axón es la salida de cada neurona que se activa y envía la señal al resto de neuronas.

La problemática principal a la que se enfrentan las redes neuronales, es que requieren un procesamiento de datos y a una repetición de tareas extremadamente costosa computacionalmente [8], ya que encontrar esta combinación de parámetros libres que nos resuelvan el problema que queremos, es una tarea ardua, para la cual es necesario primero que nada, una gran cantidad de datos previamente etiquetados y equipos de computo con la capacidad de hacer estas pequeñas tareas con una gran velocidad.

Alrededor de los años 2000's, el surgimiento de las Unidades de Procesamiento Gráfico (GPU's por sus siglas en inglés) y los grandes conjuntos de datos [9], dieron paso a la adopción masiva de métodos de ML para el análisis de estos [10], que posteriormente dieron paso a técnicas más complejas que se catalogaron como Aprendizaje Profundo o Deep Learning (DL) del cual hablaremos más adelante.

I.1.2. ML en Astronomía

Desde el surgimiento de las GPU's el número de artículos en astronomía que utilizan ML, ha crecido exponencialmente. Por ejemplo del 2014 al 2019 ha crecido por un factor de 4 los artículos que mencionan Machine Learning, mientras que el término Deep Learning lo ha hecho en un factor de poco más de 3 [11]. El ML se utiliza para distintas tareas en la astronomía y surgen nuevas maneras de utilizarlas, en lo particular hay dos áreas en las cuales son muy utilizadas, la clasificación y la identificación de patrones.

Clasificación

En particular para catálogos con un número de objetos celestes del orden de $10^6 - 10^8$ o superior, la clasificación por tipos de objetos, es por necesidad, automática. Hasta antes del desarrollo de ML ya se tenían catalogados por tipos un gran número de objetos celestes, sus imágenes, y espectros, por lo que una de las primeras aplicaciones de estas técnicas, fue, el entrenar redes para identificar entre estrellas, galaxias, y otros objetos (como supernovas o quásares) [12]. A este tipo de entrenamiento donde se tiene un con-

junto de datos previamente catalogados se le conoce como entrenamiento supervisado. También destacan el uso de ML para la clasificación entre tipo de Galaxias [13, 14, 15], o para la clasificación de tipos espectrales de estrellas [16, 17].

Gran parte de la eficiencia de los algoritmos basados en ML se basa en qué tan buenos son los datos de entrenamiento y su etiquetado previo. Por ejemplo, el famoso proyecto Galaxy Zoo [18], en donde ayudaron más de 100,000 personas a catalogar una muestra de 893,212 galaxias para un total de 40 millones de clasificaciones en distintas características de las imágenes de la muestra, tomó poco más de 6 meses. En contraste, una Red Neuronal Convolucional o Convolutional Neuronal Network (CNN) bien entrenada (con los mismos datos) puede lograr la misma clasificación en unos pocos minutos [19] con una confiabilidad de más del 90 % . Actualmente, con redes neuronales más sofisticadas, se llegan a alcanzar confiabilidades mayores al 98 % [20].

Identificación de Patrones

La identificación de patrones de forma automática ha sido otra rama donde se ha utilizado de manera exitosa el ML, por ejemplo para detectar la presencia de cráteres nuevos en cuerpos del sistema solar [21, 22], en el análisis de los datos de la misión espacial Kepler [23, 24, 25, 26, 27], para la detección de posibles candidatos a exoplanetas, la identificación de estrellas variables y su clasificación, identificación de actividad estelar. Tomando como ejemplo la misión espacial Kepler, las observaciones de un gran número de objetos por noche son cada vez más comunes, por lo que contar con algoritmos automáticos basados en ML para un pre-análisis de los datos es estrictamente necesarios para el aprovechamiento de estos en su totalidad con un grado de confiabilidad aceptable. Estos nuevos algoritmos están teniendo gran auge, sustituyendo a técnicas menos robustas y más complejas computacionalmente, utilizadas anteriormente.

Otras aplicaciones de ML

Otras áreas de la astronomía donde destaca la aplicación del ML con fines científicos son la estimación de corrimientos al rojo a partir de datos fotométricos [28, 29, 30] y de la tasa de formación estelar [31, 32], así como la aplicación para el análisis de simulaciones robustas como el caso de Illustris TNG [33, 34], entre muchas otras aplicaciones.

I.2. Objetivos, hipótesis y estructura de la tesis

Se utilizan redes neuronales para obtener información física de procesos de formación estelar no con el fin específico de poder estimar estos valores, sino de encontrar las características y variables que influyen directamente a estos procesos, de una forma práctica e indirecta a partir de los datos. Para esto, se generan imágenes sintéticas a partir datos de masa, temperatura y entropía del gas en galaxias así como de su tasa de formación estelar, obtenidos de la simulación IllustrisTNG. Y aprovechando la cualidad de las redes neuronales para encontrar patrones, se busca determinar las características de las imágenes de las cuales la ANN obtiene la información necesaria para obtener los valores de formación estelar. Esto partiendo de la hipótesis que ciertas características de las imágenes tienen más información de la tasa de formación estelar que otras, y con esto poder inferir o limitar los procesos físicos que se ven involucrados. Una vez terminado este trabajo y con la prueba de que la técnica funciona, se puede generalizar a campos magnéticos, campos de velocidades o pasar a trabajar con información mas detallada de las estrellas por ejemplo, con el fin de hacer un estudio más detallado.

En el capítulo **II** se describen las bases de los algoritmos de aprendizaje automático utilizados en esta tesis, seguido de un resumen del projecto Illustris TNG junto con una breve descripción de la elección de los datos de donde se obtendrán las imágenes sintéticas de galaxias. En el capítulo **III** se describen los algoritmos utilizados para la generación de imágenes base a partir de los datos públicos del proyecto. Se enlistan los distintos

pre-procesamientos aplicados a las imágenes utilizadas en el estudio. Se describe la rutina principal para el entrenamiento de la red neuronal así como la estructura de la red utilizada para llevar acabo dicho entrenamiento. En el capítulo **IV** se analizan los resultados obtenidos, haciendo énfasis en las diferencias obtenidas en cada entrenamiento y lo que significa esto físicamente. Finalmente en el capítulo **V** se presentan las conclusiones del trabajo, así como otras posibles aplicaciones de la técnica para estudiar distintos procesos.

CAPÍTULO II

MARCO TEÓRICO

En esta sección se introducen los términos y conceptos más importantes de aprendizaje automático y de la simulación Illustris TNG necesarios para comprender a detalle el trabajo realizado en esta tesis.

II.1. Aprendizaje Automático

En su descripción más general, el Aprendizaje Automático es el proceso mediante el cual un conjunto de datos iniciales, un modelo con parámetros libres, una función de error y un algoritmo de optimización, trabajan en conjunto mediante una rutina automática para adaptar los parámetros libres (pesos) de modo que minimicen la función de error. Si los datos iniciales de entrenamiento están previamente categorizados, entonces a este se le conoce como *Aprendizaje Supervisado*, y se busca minimizar las diferencias entre la categorización previa y la obtenida mediante el aprendizaje automático. Por otro lado, si los datos no están categorizados, el algoritmo tratará de aprender por si mismo propiedades importantes o encontrar patrones que ayuden a caracterizar los datos. A este tipo de aprendizaje se le conoce como *Aprendizaje No-Supervisado*.

II.1.1. Redes Neuronales Artificiales

Una red neuronal es una colección de neuronas (o nodos), usualmente organizadas en capas, que toman un modelo lineal con parámetros libres llamados pesos, y se conectan entre ellas mediante el uso de funciones de activación no lineales Fig. 2.1, siendo esto último la característica principal que hace posible la solución de problemas no lineales. Las funciones de activación permiten capturar relaciones no lineales entre entradas y salidas.

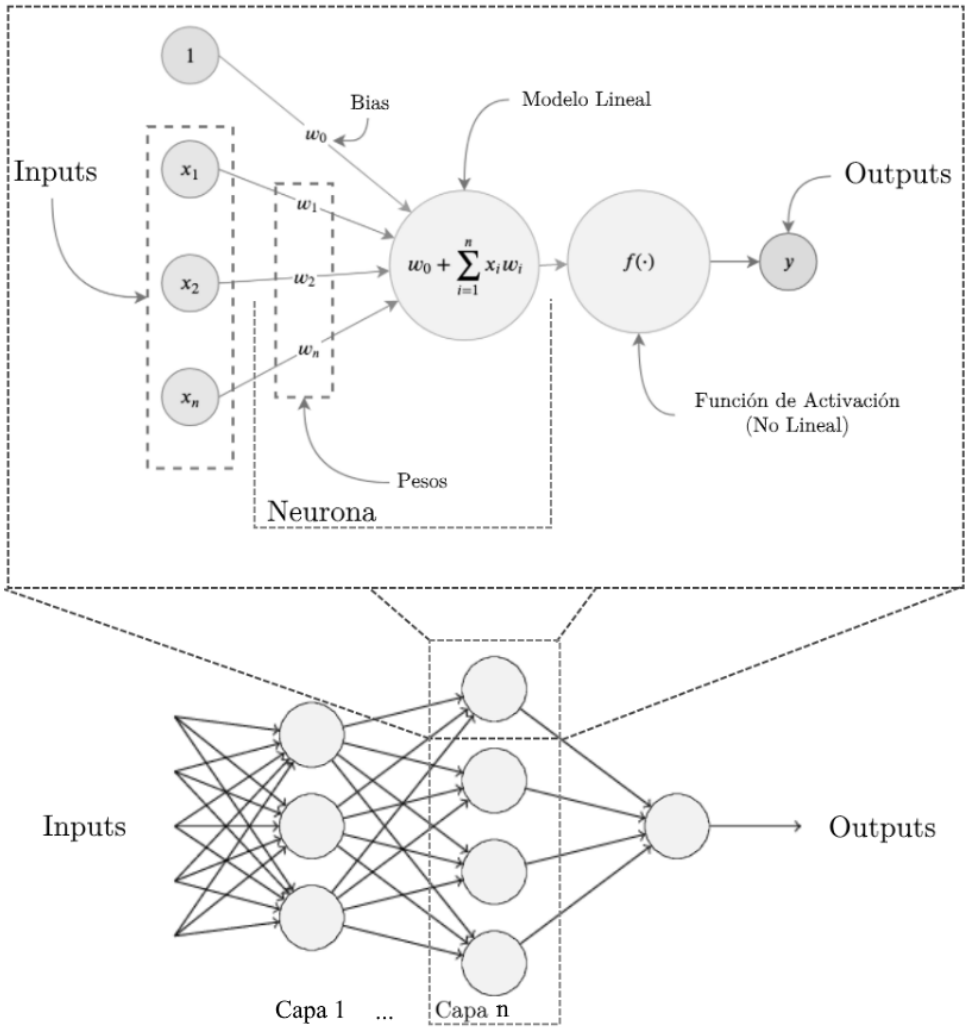

Figura 2.1: Esquema gráfico de los componentes principales en una red neuronal artificial multicapa simple, la cual está compuesta por neuronas y conexiones entre estas (pesos). Estas neuronas se organizan en capas, una capa de entrada, una capa oculta (la cual puede tener a su vez varias sub-capas), y una capa de salida.

Sí encadenamos múltiples transformaciones no lineales entre cada capa, aumentamos así, la flexibilidad y capacidad de expresión de la red neuronal. Lo que define una red neuronal es el número de capas y de neuronas en cada una de estas capas, así como la función de activación no lineal que se utiliza en éstas.

II.1.2. *Capas de una Red Neuronal*

Matemáticamente una capa en una red neuronal se expresa de la siguiente manera:

$$y_l = f_l(z_l(y_{l-1})) = f_l(w_0 + \sum_{i=1}^{n} w_i \cdot y_{l-1}), \tag{2.1}$$

donde y_{l-1} es la salida de la capa $l-1$ y por subsecuente la entrada de la capa y_l, siendo entonces y_0 las entradas de la red neuronal. Los pesos de la matriz de aprendizaje son representados por w_i, el sesgo por w_0 el cual hace de constante análogo al de b en una función lineal ($f(x) = ax + b$), el modelo lineal o suma está expresada como z_l, y por último la función de activación no lineal está representada por f_l.

En la práctica, la función de activación no lineal más utilizada es la Unidad Lineal Rectificada (ReLU), ya que produce buenos resultados y tiene la ventaja de ser computacionalmente poco costosa comparada con otras funciones de activación [35]. La función de activación ReLU está definida como:

$$f(t) = \begin{cases} t, & \text{para } t \geqslant 0 \\ 0, & \text{para } t < 0 \end{cases} \tag{2.2}$$

II.1.3. *Redes Neuronales Convolucionadas*

En el caso de una red neuronal simple, el orden de los valores de entrada no es importante, pero cuando se analiza una imagen bidimensional representada por píxeles, es claro que el orden en el que se analizan los datos de entrada es importante para poder determinar las características de esta, ya que los píxeles contiguos a otro píxel le dan contexto para poder obtener la información espacial que tratan de representar mediante dicha imagen. Para estos casos se utilizan un tipo particular de redes neuronales llamadas Convolutional Neuronal Networks.

Una CNN recibe como entrada una imagen bidimensional (digamos $m \times m$), tenemos un kernel de convolución o filtro, de dimensiones $n \times n$ con $n < m$, con n^2 pesos, que opera ordenadamente a un conjunto de píxeles de igual tamaño en nuestra imagen de entrenamiento. Los resultados pasan por una función de activación y se reordenan en un nuevo arreglo bidimensional llamado imagen de características. La figura 2.2 representa de manera esquemática la estructura y flujo de una CNN.

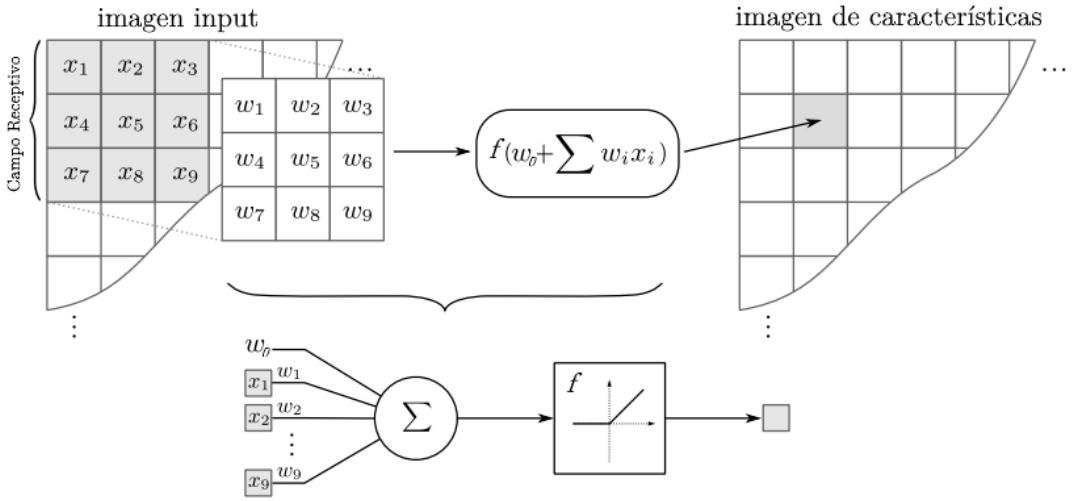

Figura 2.2: Esquema gráfico de una Red Neuronal Convolucionada. Obtenida de: [2]

Al arreglo de distintos kernels cada uno con sus tamaños y pesos, lo llamamos capa convolucionada. Las características extraídas después de una primera capa convolucionada suelen ser patrones sencillos, como gradientes, lineas verticales, horizontales y diagonales, y conforme aumenta el número de capas, los patrones son cada vez más complejos y específicos Fig 2.3.

II.1.4. Capas de Agrupación

En conjunto con las capas convolucionadas, están las capas de muestreo o agrupación, que se encargan de reducir la dimensionalidad de las imágenes de características. Esto normalmente se hace agrupando píxeles, operando de distintas maneras sobre estos, por

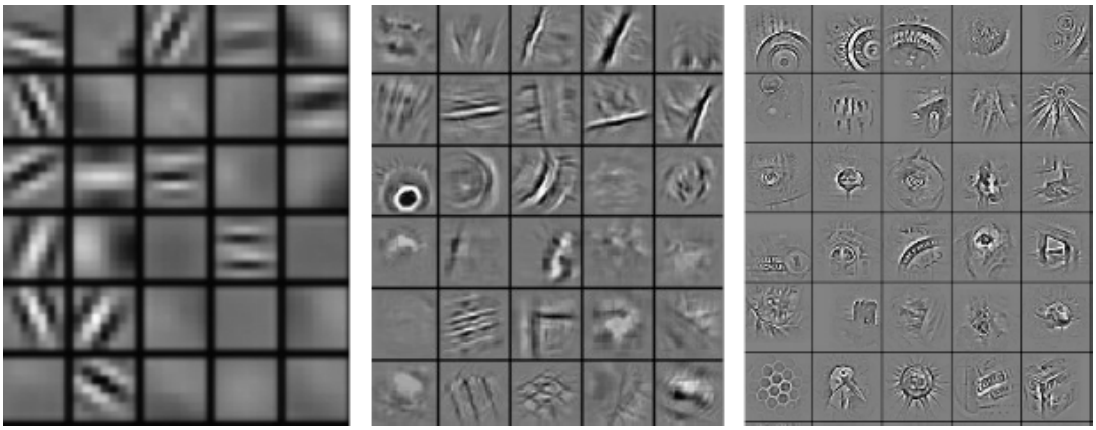

Figura 2.3: Imágenes de características de activaciones máximas después de haber pasado por los kernels de un grupo de capas convolucionadas, de izquierda a derecha aumenta la cantidad de capas. Obtenida de: [3].

ejemplo dejando el valor mínimo, el máximo, o el promedio en su defecto. A estas capas se les conoce como MinPooling, MaxPooling y AveragePooling. Esto se hace con el fin de poder sintetizar la información obtenida mediante las capas convolucionadas, hasta el punto de sintetizar esta información en un vector de $n \times 1$ que pueda a su vez conectarse con una red neuronal clásica Fig 2.4.

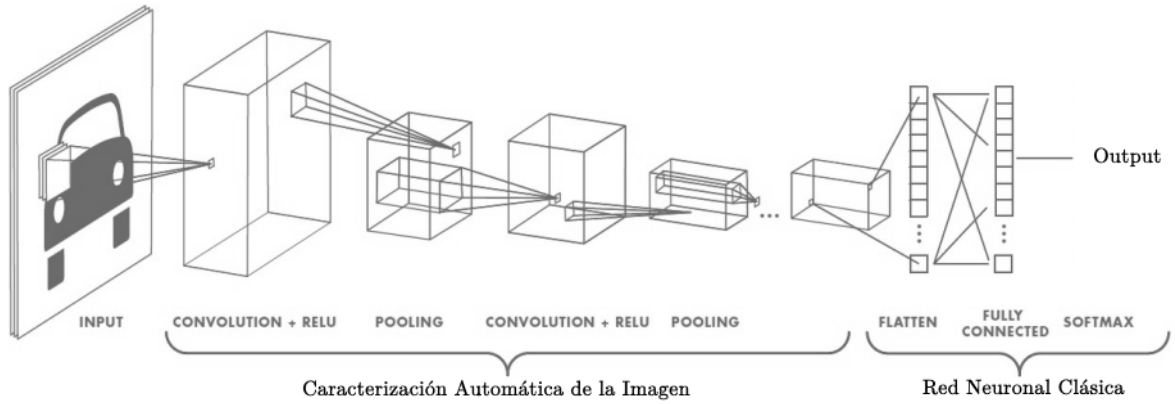

Figura 2.4: Estructura clásica utilizada en ML para el reconocimiento de patrones en imágenes . Obtenida de: [3].

II.1.5. Entrenamiento de una Red Neuronal

Ahora que tenemos todas las partes de nuestra red neuronal necesitamos que aprenda automáticamente, a partir de datos previamente catalogados para el caso de un aprendizaje supervisado. Para esto se define una función de error, la cual pondera la salida de la red neuronal con el valor real. Durante el entrenamiento, se busca mediante la modificación de los pesos w_i y los bias w_0 de las neuronas, que esta función de error se minimice. La función de error más utilizada es el error cuadrático medio o MSE por sus siglas en inglés. Recordemos que el cálculo de este se hará una gran cantidad de veces, por lo que la simplicidad a veces es la mejor opción. Al proceso que mide qué tanto afecta cada una de las neuronas y sus pesos al valor final, con el fin de determinar cómo modificar estos pesos automáticamente en cada paso, se le conoce como algoritmo de propagación hacia atrás del error (backpropagation).

La mayoría de los algoritmos de optimización para buscar mínimos, están basados en el descenso del gradiente:

$$\beta^{(t+1)} = \beta^{(t)} - \tau_t l^{(t)}, \tag{2.3}$$

donde los pesos β de la siguiente iteración $t+1$, se modifican de acuerdo al valor de la iteración actual t, el aprendizaje actual τ_t y el error del gradiente asociado al peso actual $l^{(t)}$.

II.1.6. Transferencia de Aprendizaje

Una técnica muy importante en el aprendizaje de máquinas, y que ha permitido avances importantes en esta rama, es la conocida como *transferencia de aprendizaje*, que consiste en utilizar una red neuronal pre-entrenada que comparta similitudes con el problema que se va a resolver [36], en este trabajo la similitud es que utilizaremos imágenes de dimensiones iguales al de esta red. Al igual que el comportamiento de nuestro cerebro,

el cual cuando aprende algo nuevo se basa en el conocimiento previo que ya se posee, no comienza desde cero al enfrentarse a una nueva tarea.

En este trabajo, utilizaremos esta técnica con la red pre-entrenada $VGG-16$ [37] compuesta por 13 capas convolucionales, 5 capas tipo Max-Pooling y 3 capas Densas o capas totalmente conectadas Fig 2.5. Esta red fue entrenada con el fin de clasificar imágenes de la base de datos ILSVRC-2012 [38] con más de 1.3M de imágenes separadas en más de 1000 clases distintas, por lo cual posee una gran capacidad de reconocer patrones generales en cualquier tipo de imágenes. La ventaja es que los pesos de esta red están disponibles públicamente, por lo que no es necesario hacer un entrenamiento desde cero. Anteriormente se ha utilizado satisfactoriamente esta red con problemas de clasificación en astronomía [39, 40, 41].

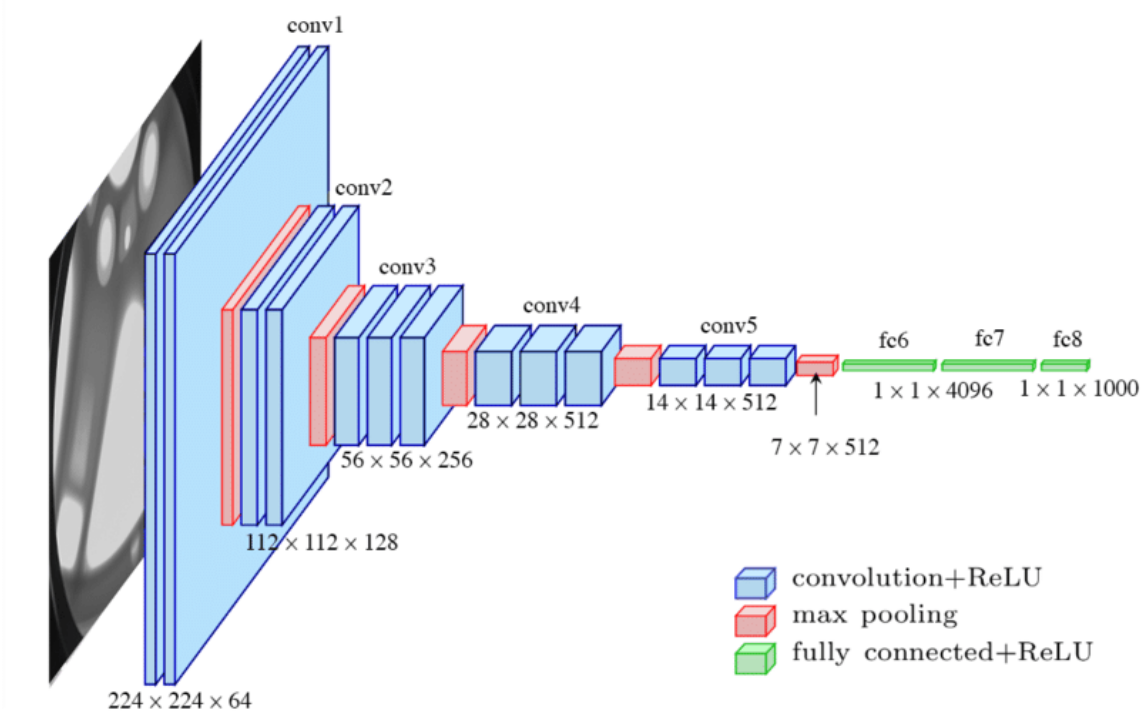

Figura 2.5: Estructura de la red pre-entrenada VGG-16 utilizada para la clasificación de imágenes. Imagen obtenida de Google Images.

II.2. Formación Estelar

El estudio de la formación estelar se puede catalogar en dos vertientes: el estudio de la microfísica y la macrofísica [42]. La primera se encarga, entre otras cosas, de estudiar la forma en que nacen las estrellas individuales y todos los procesos físicos que intervienen [43], incluyendo información acerca de discos, jets y en general flujos asociados a estrellas jóvenes y también su evolución dinámica (e.g.[44]). Por otro lado, la macrofísica se centra en el estudio de la formación de sistemas de estrellas (e.g.[45]), cúmulos de estrellas (e.g.[46]) y estructuras en galaxias (e.g.[47]), etc. En general, esta área estudia las características que propician la formación estelar de las llamadas *nubes moleculares*, cómo afectan la composición de estas, o el tipo de galaxia y el tipo de medio interestelar en que ocurre la formación de estrellas [48]. Un problema importante es que os procesos físicos clave que determinan cómo las nubes moleculares se contraen y se fragmentan puede ser sondeado de cerca sólo en la Vía Láctea, y solo se tiene estudios de regiones individuales de formación estelar. Para otras galaxias cercanas, se integra sobre regiones grandes, o galaxias completas para un estudio más general [49]. Podemos resumir las principales interrogantes del estudio de la formación estelar son: *¿Qué determina la tasa de formación estelar? ¿Qué proceso determina la función inicial de masa estelar?* y *¿Cómo se agrupan las estrellas?*. En particular, en esta tesis nos centramos en buscar cuales características en las galaxias son importantes para obtener la tasa de formación estelar, ya que se analizaran imágenes de gas en galaxias donde se pueden observar estructuras con resolución para observar nubes moleculares gigantes.

El proceso para la formación de estrellas comienza con una nube molecular compuesta por gas y polvo, la cual puede ser alcanzada por una onda de choque provocada por el entorno que la rodea, esto puede ser debido a estrellas calientes (OB), turbulencia en la misma nube, una explosión de supernova o incluso una colisión con otra nube. La nube molecular debido a esta interacción, sufre fragmentación con zonas donde ocurren

contracciones de caída libre que a su vez forman protoestrellas, las cuales en combinación con un disco protoestelar de la misma nube contrayéndose, finalmente forman estrellas. Es importante notar que este proceso se retro-alimenta, dependiendo el tipo de estrellas que se formen. Por ejemplo, las estrellas calientes supermasivas tienden a barrer el gas a su alrededor provocando a su vez formación estelar en las regiones aledañas debido a acumulación con el gas más externo. Finalmente la nube se divide en asociaciones de diferente combinación de estrellas calientes tipo OB (rojas y azules), cefeidas variables, estrellas supergigantes, y estrellas con propiedades dinámicas similares a la de la nube progenitora [50, 51].

Las estrellas se forman a partir de gas interestelar neutro en nubes moleculares gigantes de una masa de 10^4 a $10^{6.5} M_\odot$ donde la densidad de H_2 es de $n(H_2) \sim 10^2$ a $10^5 \mathrm{cm}^{-3}$ en zonas de temperatura baja (T_{gas} $10 - 30K$)[52, 53]. Las observaciones de galaxias del tipo espirales muestran que la mayor parte del gas molecular está concentrado en regiones de sus brazos espirales, incluyendo la Vía Láctea [54, 55, 56], dado que son las zonas con una mayor densidad de gas en este tipo de galaxias. Por otro lado desde hace tiempo se sabe que las galaxias de tipo elípticas en general tienden a tener menos formación estelar, por lo que, en principio, se puede establecer también una relación entre la estructura general de la galaxia y su formación estelar.

Existen propiedades físicas que nos pueden dar información directa de la formación estelar, como lo son el campo magnético asociado al gas, las velocidades de éste [57], la masa del gas y su distribución [58]. También como se mencionó anteriormente la temperatura juega un papel importante. Para este trabajo utilizamos la información de masa, temperatura y entropía, esperando que la entropía nos sirva para mapear corrientes de gas frío entrando a las galaxias mediante filamentos, los cuales se ha encontrado que son importantes en galaxias con alta formación estelar [59], asociando cada uno a un canal por separado de las imágenes que utilizamos para los entrenamientos. Esto con la idea de hacer un primer caso de estudio donde sabemos que los canales contienen información de

la formación estelar, mostrando cómo la técnica aplicada en esta tesis nos da información relevante sobre la importancia de los parámetros de entrada en función de los resultados que se obtienen.

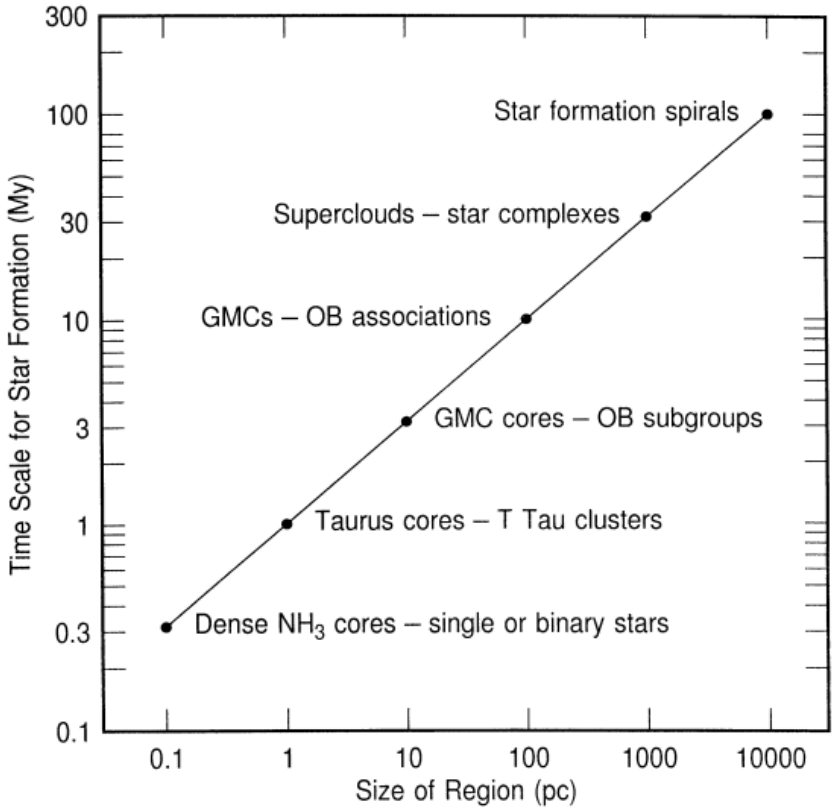

Figura 2.6: Diagrama de escalas de tiempo correspondientes para agrupaciones de estrellas en zonas de formación estelar. Obtenida de: [4]

Prosiguiendo con la idea de una formación estelar jerárquica [4, 60, 61], que relaciona escalas de tiempo con la evolución de escalas espaciales cada vez mayores en zonas de galaxias, pasando desde la formación particular de estrellas (0.3 My), hasta la formación de complejos estelares (30 My) (ver Figura 2.6), para nuestro estudio, la resolución de la simulación es suficiente para mostrar escalas de hasta aproximadamente 200 h^{-1}pc. Por otro lado, para escalas grandes y mediante la relación de la entropía con la densidad y la

temperatura $K = log(T/\rho^{2/3})$ se busca encontrar información necesaria para identificar flujos de corrientes de gas frío entrando a centro de la galaxia que anteriormente han sido relacionados con altas tasas de formación estelar y que usualmente se logran ver en escalas comparables al tamaño de las galaxias [59, 62].

II.2.1. *Illustris the Next Generation*

La simulación de Illustris TNG [63, 64, 65, 66, 67] consiste en un grupo de simulaciones cosmológicas hidrodinámicas que modelan un rango amplio de procesos importantes en la formación de galaxias. Los datos se dividen en 3 simulaciones: TNG50, TNG100 y TNG300, siendo el número que los acompaña el tamaño en megapársec del un lado de la caja de la simulación Fig 2.7. En particular en esta tesis nos interesa la información de estructuras dentro de galaxias, por esto seleccionamos la simulación TNG50 [68, 69] ya que, aunque el tamaño de la simulación es más pequeña, ésta cuenta con mayor resolución y contiene un aproximado de $\sim$ 700 galaxias de más de 10^8 masas solares.

Al día de hoy, TNG50 es la simulación con más resolución del proyecto Illustris. Incluye información de distribución de materia oscura, gas, estrellas, agujeros negros, y campos magnéticos, en un cubo de 51.7^3 Mpc, con una longitud de caja de $35c$ kpc h^{-1} en coordenadas comóviles y con condiciones a la frontera periódicas, conteniendo un total de 2×2160^3 partículas de materia oscura y gas, con resolución espacial promedio de entre 70 y 140 h^{-1}pc en las zonas de formación estelar [68].

II.2.2. *Descripción de los Datos*

La simulación TNG50 se divide en 100 snapshots a distinto valor de corrimiento al rojo desde 20.05 hasta 0, en donde cada uno tiene información de las partículas de gas, partículas de materia oscura, estrellas, partículas de viento estelar, gases trazadores y agujeros negros. De cada uno de los tipos de partículas se pueden obtener distintos campos, como lo son posiciones, masas, velocidades, energía interna entre otros. Una descripción

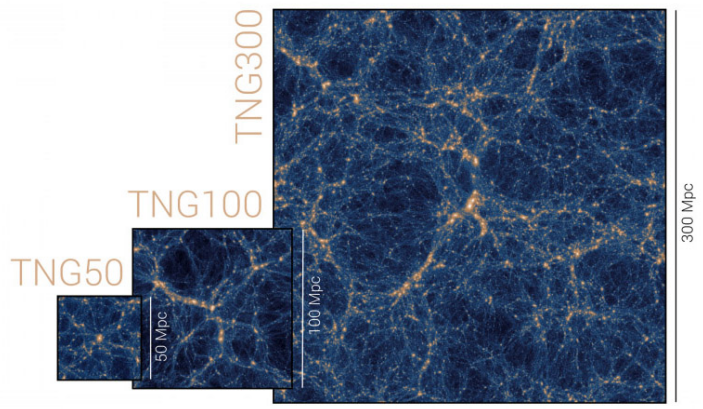

Figura 2.7: Tamaño de las 3 simulaciones del proyecto Illustris TNG . Obtenida de: `www.illustris-project.org`

completa de cada uno de los campos en cada categoría de partícula se encuentra en: `www.tng-project.org/data/docs/specifications/`.

En nuestro estudio nos centramos en las partículas de gas. La información de posición, masa y densidad de éstas se obtiene directamente de la simulación, obtenemos el valor de la tasa de formación estelar especifica como $sSFR = SFR/M_\star$, siendo $M_\star$ la masa total de las estrellas en la galaxia, una representación de la distribución de masa estelar y formación estelar de la simulación se muestra en figura 2.8. El modelo utilizado para la formación estelar en la simulación TNG sigue el trabajo de [70], donde las partículas de gas se convierten estocásticamente en partículas estelares una vez que la densidad local es mayor a $n_H = 0.1\text{cm}^{-3}$ en un tiempo de escala que empíricamente reproduce la relación de Kennincutt-Schmidt [71].

La temperatura se obtiene indirectamente de la energía interna u y la densidad de electrones x_e, a través de las siguientes relaciones:

$$T = (\gamma - 1) \cdot u/k_B \cdot \frac{[UE]}{[UM]} \cdot \mu \tag{2.4}$$

$$\mu = \frac{4}{1 + 3X_H + 4X_H x_e} \cdot m_p \tag{2.5}$$

donde, $\gamma = 5/3$ es el índice adiabático, k_B es la constante de Boltzmann en sistema CGS, $X_H = 0.76$ es la fracción de masa de Hidrógeno. UM y UE son las unidades de masa y energía del código en CGS, donde $UE = UM \cdot UL^2/UT^2$, donde $UL = 1$ kpc y $UT = 1$Gyr, por lo que la razón $[UE]/[UM]$ es 10^{10}. Finalmente, con la temperatura se obtiene la entropía, mediante la definición astrofísica usual de entropía para una partícula de gas:

$$S \equiv \log K = \log \left(\frac{k_B T}{n_e^{2/3}} \right) \tag{2.6}$$

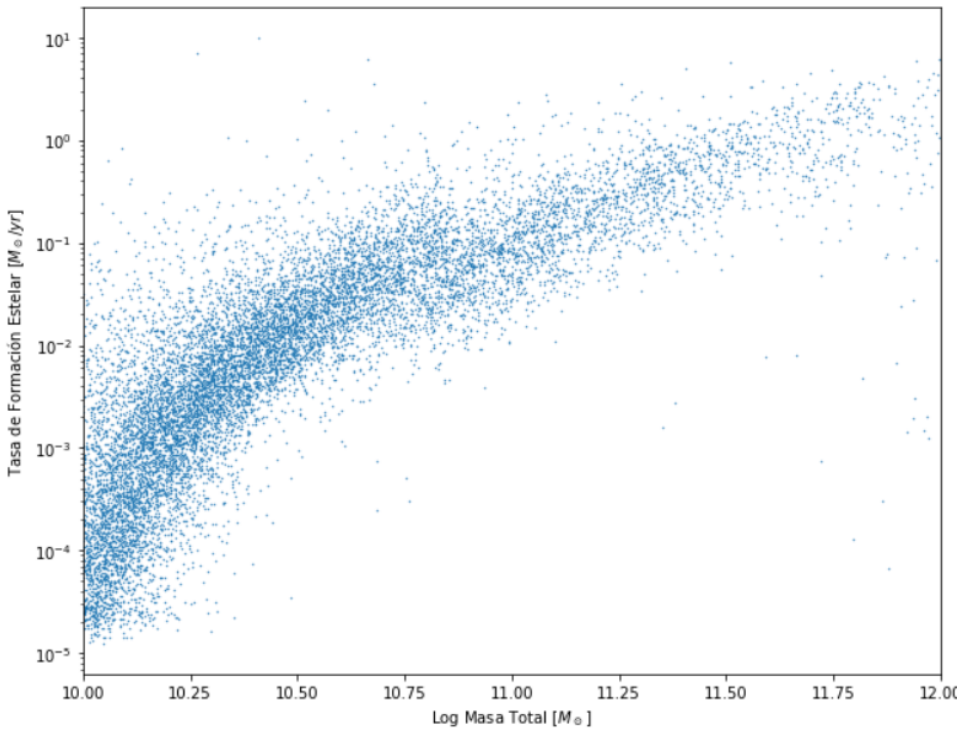

Figura 2.8: Tasa de formación estelar como función de la masa total de la galaxia, para una muestra de aprox. 14,000 galaxias de Illustris TNG-50.

CAPÍTULO III

METODOLOGÍA

III.1. *Conjunto de Datos de Entrenamiento*

En este capítulo se describen los pasos seguidos para la obtención de los datos, los distintos pre-procesamientos aplicados a las imágenes para generar los conjuntos de datos de entrenamiento, y finalmente se describe la estructura de la red neuronal a utilizar durante el proceso de aprendizaje automático.

III.1.1. *Obtención de los Datos*

Seleccionamos como origen de los datos la simulación TNG50 de Illustris, por su resolución óptima para generar imágenes de galaxias con gran detalle en las zonas de formación estelar. Para obtener la información de los distintos tipos de partículas de una forma sencilla se utilizó la librería desarrollada para este fin, una descripción y ejemplos de cómo utilizarla puede verse en: `www.tng-project.org/data/docs/scripts/`

```
import illustris_python as il
PATH_ILL = '../sims.TNG/TNG50-1/output/'
```

Para hacer la selección de los datos primero se eliminan subhalos que no se consideran en sí galaxias mediante el campo $'SubhaloFlag'$. Se seleccionaron galaxias en un rango entre $0.5 \times 10^{10} M_{\odot}$ y $1 \times 10^{12} M_{\odot}$ de masa total de estrellas, con una masa total menor a $2 \times 10^{12} M_{\odot}$ reproduciendo las características típicas de estudios a $z = 0$ [72, 73], y que el grupo de galaxias al que pertenecen tenga una masa menor a $5 \times 10^{12} M_{\odot}$ con el fin de que efectos de interacción entre galaxias no afecten nuestros resultados [74, 75].

```
subhalo = il.groupcat.loadSubhalos(PATH_ILL, SNAPSHOT, fields = ['
    SubhaloSFR', 'SubhaloMass', 'SubhaloMassType', 'SubhaloGrNR', '
    SubhaloFlag'])
```

Snapshot	Redshift (z)	No. Galaxias	sSFR Promedio ($\times 10^{-12}$)	sSFR Mín. ($\times 10^{-17}$)	sSFR Máx. ($\times 10^{-10}$)
99	0.000	715	5.68	2.69	1.00
98	0.009	710	5.61	2.03	0.66
97	0.023	713	6.03	11.75	1.43
96	0.033	716	6.75	2.05	4.74
95	0.048	712	6.54	3.54	1.70
94	0.058	708	8.22	5.33	10.8
93	0.073	713	7.98	225.20	6.33
92	0.083	712	9.23	2.66	6.92
91	0.099	726	7.51	6.88	4.00
90	0.100	726	7.99	15.19	6.23
89	0.125	739	7.73	6.72	2.99

Tabla **III**.1: Snapshots seleccionados con su corrimiento al rojo, y el número de galaxias de la muestra correspondiente a cada uno de estos, así como los valores promedio, mínimo y máximo para cada sub muestra.

```
halos = il.groupcat.loadHalos(PATH_ILL, SNAPSHOT,
        fields = ['GroupMass'])
valid.append( np.where(
    (subhalos['SubhaloMassType'][:,4] > 0.5) *
    (subhalos['SubhaloMassType'][:,4] < 100) *
    (subhalos['SubhaloFlag'] == True) *
    (subhalos['SubhaloMass'] < 200) *
    (halos['GroupMass'][subhalos['SubhaloGrNr']] < 500) )[0] )
```

Dadas las condiciones anteriores, en promedio se obtuvieron $\sim$ 700 galaxias con estas características por cada snapshot de la simulación. Por tanto, se extendió la selección a los últimos 10 snapshots (99, 98, ..., 89) obteniendo en total 7881 galaxias. La descripción de cada snapshot así como el número de galaxias obtenido se describe en la Tabla **III**.1.

El valor de la tasa de formación estelar especifico para las galaxias válidas lo obtenemos y guardamos para utilizarlo posteriormente en nuestro entrenamiento. También se guardan los "ID" para las galaxias seleccionadas. En la Fig. 3.1 se muestra la distribución de tasa de formación estelar de la muestra de galaxias seleccionadas.

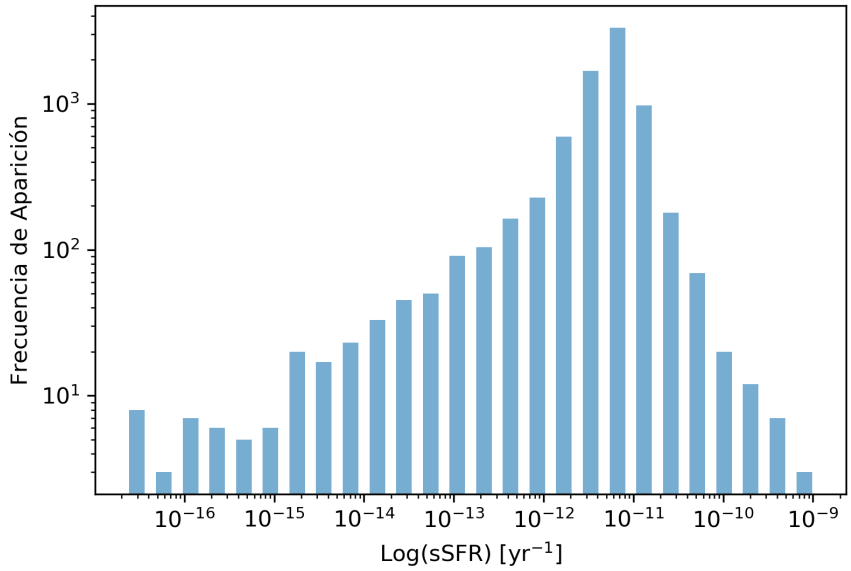

Figura 3.1: Distribución de la tasa de formación estelar específica para las galaxias seleccionadas.

```
import pickle
ssfr = subhalos['SubhaloSFR'] / (subhalos['SubhaloMass']*1e10)
validsfr = []
for i in valid:
    validsfr.append(ssfr[i])
file_name = "allsfr.pkl"
open_file = open(file_name, "wb")
pickle.dump(validsfr, open_file)
open_file.close()
file_name = "valid.pkl"
open_file = open(file_name, "wb")
```

```
pickle.dump(valid, open_file)
open_file.close()
```

Una vez definida la muestra de galaxias válidas, se procedió a obtener la información de interés, esto es, las posiciones, masa, temperatura y entropía de las partículas de gas de cada galaxia. En este punto se utilizaron rutinas previamente desarrolladas por el Dr. Miguel Aragón y pueden ser obtenidas en la siguiente liga Google Drive (`t.ly/W1Xj`).

```
from halos import *
gas = get_galaxy_gas( PATH_ILL, 35000, haloId,
                 POS_SUBHALO, SNAPSHOT, FoF=True)
```

Con esta función obtenemos posición, masa, temperatura, formación estelar, densidad y entropía, todo esto por cada partícula de gas. A estas partículas de gas, primero se rotan las coordenadas de posición para obtener la galaxia alineada con su eje de rotación usando la función *compute_ normal()* que hace uso de las posiciones y velocidades para calcular el vector de rotación. Se usa el tiempo de formación de estrellas con el fin de identificar aquellas estrellas jóvenes que se mueven aún con velocidades similares a la nube de la cual se formaron y así determinar el eje de rotación del disco, la galaxia para poder obtener imágenes de las galaxias vistas de frente ("face-on") y observar la estructura interna de la galaxia en su totalidad.

```
stars  = get_galaxy_stars2(PATH_ILL, 35000, haloId,
                     POS_SUBHALO, SNAPSHOT, FoF=False)
disk_normal = compute_normal(stars[0], stars[2], stars[3])
gas[0][:,0],gas[0][:,1],gas[0][:,2] =
        rotate_halo(gas[0][:,0],gas[0][:,1],gas[0][:,2], disk_normal)
```

Para generar las imágenes se hicieron proyecciones 2D de la distribución de gas por medio de histogramas 2D con 256 bins ($60h^{-1}$kpc) por lado (1 píxel $\approx 234\ h^{-1}$pc). Para los campos de entropía, masa y temperatura se implementó una rutina en C (también disponible en la carpeta de Google Drive) (`t.ly/W1Xj`) llamada desde Python, ya que

realizaba la tarea con una velocidad superior a las alternativas disponibles en las librerías de Python (p.e Numpy es aprox. 3 veces más lento).

```
!gcc -shared -o ../C_Code/histogram_2d.so -fPIC ../C_Code/
   histogram_2dandimp.c
import ctypes
from numpy.ctypeslib import ndpointer
lib = ctypes.CDLL('../C_Code/histogram_2d.so')
histogram_2d = lib.histogram_2d
histogram_2dimp = lib.histogram_2dimp
def histogram_2d_fast(_x,_y, _w, _n, _ng, _xr,_yr):
    lib = ctypes.CDLL('../C_Code/histogram_2d.so')
    histogram_2d = lib.histogram_2d
    histogram_2d.restype = None
    histogram_2d.argtypes = [ndpointer(ctypes.c_float),
                         ndpointer(ctypes.c_int),ndpointer(ctypes.c_int),
                         ndpointer(ctypes.c_float),
                         ctypes.c_int,]
    val = np.where((_x >= _xr[0]) * (_x < _xr[1]) *
                   (_y >= _yr[0]) * (_y < _yr[1]) )[0]
    _x2 = ( (_x[val]-_xr[0])/(_xr[1]-_xr[0])*0.99*_ng ).astype(np.int32)
    _y2 = ( (_y[val]-_yr[0])/(_yr[1]-_yr[0])*0.99*_ng ).astype(np.int32)
    _w2 = _w[val].astype(np.float32)
    vol = np.zeros((_ng,_ng), dtype=np.float32)
    histogram_2d(vol, _x2,_y2, _w2, len(val), _ng)
return vol
```

III.1.2. Normalización de los Datos

Para que la muestra fuera uniforme en intensidad, se realizaron los histogramas para las primeras ~ 700 galaxias del primer snapshot. De este, se obtuvieron los valores máximos en los bines de los histogramas, con el fin de identificar un máximo global en cada uno

de los canales y normalizar todas las galaxias con respecto a este valor (dividiendo todo el canal sobre el valor máximo encontrado). Los resultados obtenidos para máximos en los histogramas son $6.21 \times 10^3[KeVcm^2], 1.56 \times 10^{-2}[M_\odot], 1.25 \times 10^4[K]$, para entropía, masa y temperatura respectivamente.

Con el histograma normalizado utilizando el máximo obtenido anteriormente lo siguiente es el tratamiento para generar las imágenes. En este caso utilizamos la siguiente subrutina, basada en el trabajo de Lupton et al. [76] con el fin de disminuir el rango dinámico y hacer más evidentes las estructuras en las imágenes.

```
def myasin(x, beta):
    return np.log( np.abs(x/beta) + np.sqrt( np.power(x/beta,2) + 1.0) )
def fx(_x, mini, maxi, beta):
    return myasin(_x - mini, beta)/myasin(maxi-mini, beta)
def lupton(r,g,b, beta = 0.001):
    I = (r + g + b)/3.0
    R = r
    G = g
    B = b
    with np.errstate(invalid='ignore', divide='ignore'):
        R = r*np.divide(fx(I, 0,1, beta),I, where = (I != 0) )
        G = g*np.divide(fx(I, 0,1, beta),I, where = (I != 0) )
        B = b*np.divide(fx(I, 0,1, beta),I, where = (I != 0) )
    return np.dstack([R,G,B])
```

Las imágenes para las 7881 galaxias en sus tres canales pueden verse en el siguiente Google Drive (`t.ly/2TuE`). Una representación típica de las galaxias con su valor de formación estelar especifico puede verse en la Fig. 3.3.

III.1.3. Datos de Entrenamiento y Datos de Prueba

Es necesario separar nuestras galaxias de forma que una parte de la muestra se utilice para entrenar la red (datos de entrenamiento), y otra parte de la muestra que la red nunca

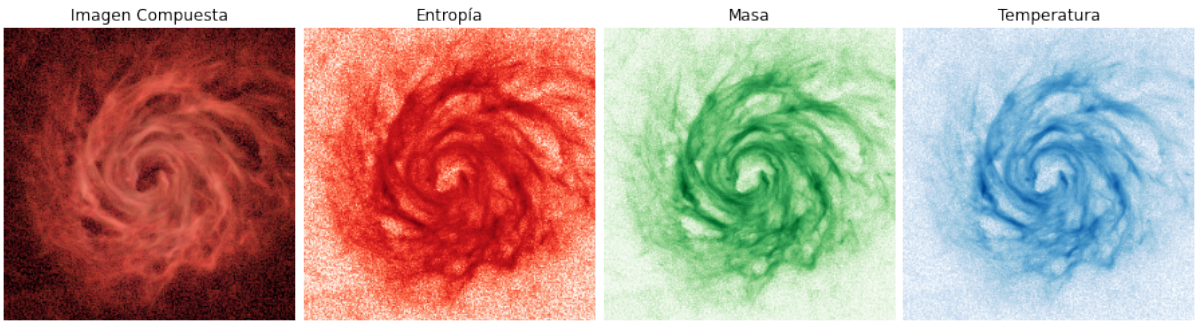

Figura 3.2: Ejemplo de una imagen compuesta y su división en 3 canales correspondientes a entropía, masa y temperatura. Cada lado de la imagen representa $60h^{-1}$kpc.

haya visto para validar la red (datos de prueba). El código "Split Data" para separar los datos se encuentra en la sección de Anexos **V.1**. Con este dividimos nuestra muestra en un 90 % de las galaxias de entrenamiento (aprox. 7100), y el resto en galaxias de prueba 10 % (aprox. 700 galaxias).

III.1.4. Casos de Entrenamiento

Se realizó un pre-procesamiento sobre las imágenes para tener distintos casos de estudio relevantes y así determinar las características de las imágenes que nos interesan y que suponemos importantes para determinar la formación estelar. Estos casos se describen a continuación.

Datos con propiedades físicas independientes

Se generaron 4 conjuntos de datos de entrenamiento, el primero con los 3 canales en conjunto (entropía, masa, temperatura), el segundo, tercero y cuarto, con los canales por separado buscando encontrar cual de estos tiene más información. Un detalle a tener en cuenta es que en el caso de muestras con un solo canal, este se copió en los tres canales de la imagen para mantener la misma estructura de entrada de la red neuronal.

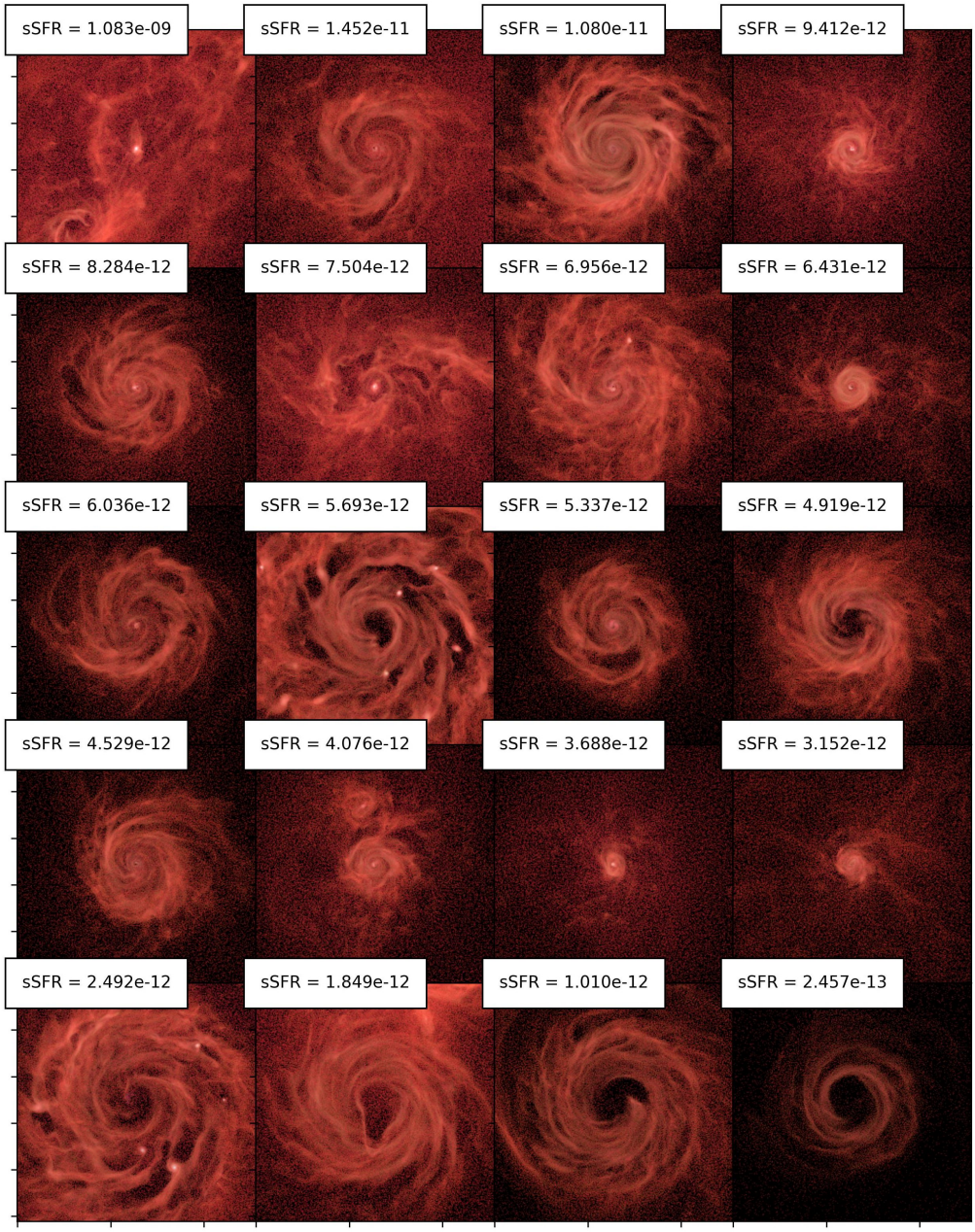

Figura 3.3: Muestra típica de las galaxias de entrenamiento y su valor de formación estelar especifica, mostrando densidad, entropía y temperatura en los canales r,g,b. Las unidades de la formación estelar especifica están dadas en yr^{-1}. Cada lado de la imagen representa $60h^{-1}\mathrm{kpc}$.

```
### Canal de Entrenamiento, 0 = Entropia, 1 = Masa, 2 = Temperatura.
    Copia el canal seleccionado en todos los canales.
ichanel = 0
for i in [0, 1, 2]:
  xtrain[:,:,:,i] = xtrain[:,:,:, ichanel]
  xtest[:,:,:,i] = xtest[:,:,:, ichanel]
```

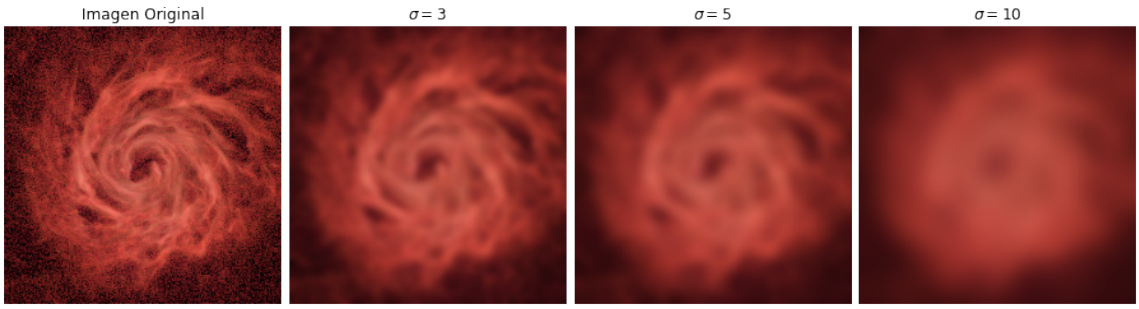

Figura 3.4: Ejemplo de mapas de densidad y aplicando un filtro Gaussiano con un valor de σ igual a 3, 5 y 10 píxeles que corresponden a $700, 1170$ y 2340 h^{-1}pc de radio de suavizado. Cada lado de la imagen representa $60h^{-1}$kpc.

Datos con suavizado Gaussiano

La siguiente muestra de imágenes de entrenamiento se realizó aplicando un filtro Gaussiano a las imágenes compuestas por los tres canales con una escala de suavizado σ igual a $3, 5$ y 10 pixeles correspondiendo a $700, 1170$ y 2340 h^{-1}pc, con la finalidad de eliminar información espacial en las estructuras de las galaxias a escalas menores (ver figura 3.4).

```
from PIL import Image, ImageFilter
### Ejemplo para cargar y aplicar filtro Gaussiano a imagen.
image = np.asarray(PIL.Image.open(PATH_IMAGE))
gaussianimage3 = PIL.Image.open(PATH_IMAGE).filter(ImageFilter.
    GaussianBlur(radius=3)))
gaussianimage5 = PIL.Image.open(PATH_IMAGE).filter(ImageFilter.
    GaussianBlur(radius=5)))
```

```
gaussianimage10 = PIL.Image.open(PATH_IMAGE).filter(ImageFilter.
    GaussianBlur(radius=10))))
```

Datos con máscaras de enfoque: estudio de estructuras

Por último, se realiza la operación inversa al suavizado, a las imágenes compuestas por los tres canales se les aplica la técnica de máscaras de enfoque (unsharp masking), para resaltar solamente las estructuras de escala menor que el filtro Gaussiano aplicado. Las imágenes que se obtienen muestran solamente la resta entre la imagen original y una imagen suavizada, por lo que la información de intensidad se pierde en el proceso (ver figura 3.5). Durante este trabajo llamaremos *estudio de estructuras* al uso de esta técnica, creando así los últimos 2 conjuntos de datos de entrenamiento.

```
### Haciendo uso del c digo anterior obtenemos las estructuras.
structureimage3 = image - np.asarray(gaussianimage3)
structureimage5 = image - np.asarray(gaussianimage5)
structureimage10 = image - np.asarraygaussianimage10)
```

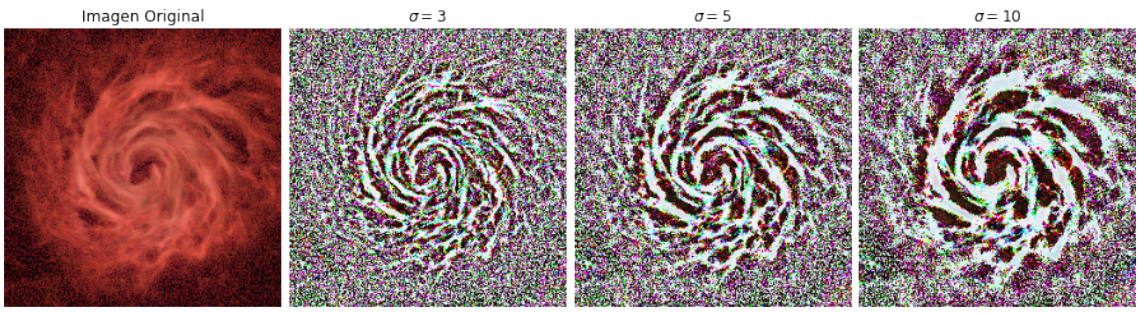

Figura 3.5: Ejemplo de mapas de densidad mostrando la diferencia entre la imagen original y una imagen con una máscara de enfoque a distintos valores de suavizado, ver Fig. 3.4. Cada lado de la imagen representa $60h^{-1}$kpc.

Con estas muestras de entrenamiento tendríamos listos los distintos 10 casos con los cuales se pondrá a prueba nuestra red neuronal: 1 set de imágenes compuestas, 3 sets

correspondientes a los canales de entropía, masa y temperatura, 3 sets con suavizados gaussianos, y 3 sets con imágenes de contraste (i.e. estudio de estructuras).

III.2. Arquitectura de la Red Neuronal

Como red neuronal se utilizó la red pre-entrenada VGG16 haciendo uso de la técnica de transferencia de aprendizaje **II.1.6**, congelando los parámetros de la red convolucionada, y agregando capas completamente conectadas ("Dense") sobre las cuales se hará el entrenamiento.

```
model = VGG16(weights='imagenet', include_top=False, input_shape = (256,
    256, 3))
for layer in model.layers:
    layer.trainable = False
x = model.output
x = Conv2D(1, (1, 1),  activation='relu')(x)
x = Flatten()(x)
x = Dense(64, activation="relu")(x)
x = Dense(128, activation="relu")(x)
x = Dense(256, activation="relu")(x)
x = Dense(512, activation="relu")(x)
x = Dense(256, activation="relu")(x)
x = Dropout(0.2)(x)
out_val = Dense(1, activation = "relu", name="out_val")(x)
model = Model(inputs=model.input, outputs=out_val)
```

Esta red recibe como entrada imágenes en formato RGB de dimensiones 256x256x3, y devuelve como resultado un solo valor, el cuál se comparará con los valores de formación estelar específica, obtenidos de la simulación durante la selección de galaxias de Illustris. Dada esta estructura, la red neuronal cuenta con ∼15 millones de parámetros, de los cuales ∼14.7 millones no entrenables son de la red convolucionada pre-entrenada VGG16,

y los ∼300,000 restantes son parámetros entrenables de las capas densas de nuestra red.

Como se mencionó anteriormente, la misma estructura se utilizó para los 10 casos distintos de entrenamiento. Se usó el optimizador *Adadelta*, en un total de 1500 épocas (1 época es un ciclo completo de entrenamiento con todos los datos) en cada caso, reiniciando los pesos de la red cada que un nuevo caso de estudio era analizado.

```
model.compile(loss='mse', optimizer='Adadelta',metrics=['mse'])
model_checkpoint = ModelCheckpoint("%sWeights.h5"%(namee), monitor='loss
    ', save_best_only=True)
model.fit(xtrain[:,:,:,:3], ytrain, epochs = 1500, batch_size = 100,
    verbose = 1, callbacks=[model_checkpoint])
ypred = model.predict(xtest[:,:,:,:3])
np.save('YValues.npy', np.array([ytest, ypred[:,0]]))
```

Finalmente se obtienen las predicciones de las galaxias de prueba, y los valores de la sSFR se guardan junto al valor real obtenido de la simulación para su posterior análisis. También obtenemos los pesos de la red ya entrenada para cada caso de estudio, que nos sirven de base en caso de querer realizar un entrenamiento más profundo.

Un ejemplo completo del código empleado para hacer el entrenamiento del conjunto de datos original con los 3 canales puede verse en la sección de Anexos **V.1** como "Pipeline".

CAPÍTULO IV

ANÁLISIS DE RESULTADOS

En esta sección se discuten los resultados del entrenamiento de la red neuronal, así como el procesamiento y análisis realizados sobre estos.

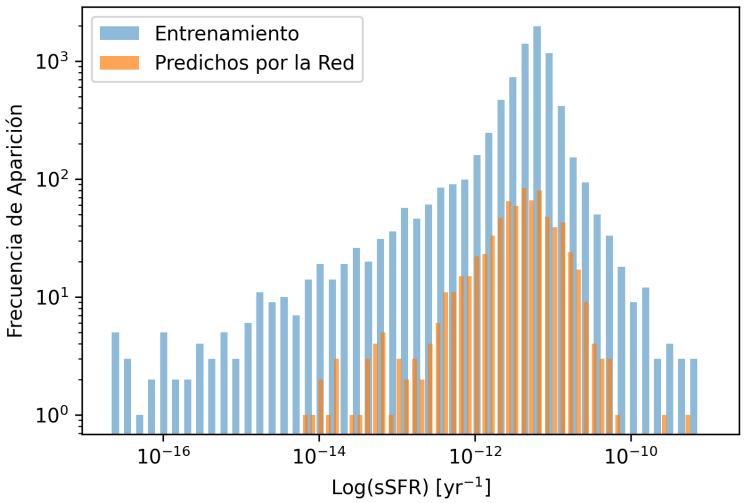

Figura 4.1: Distribución de la tasa de formación estelar específica para las galaxias seleccionadas (Azul) y también para los valores predichos por la red de los datos de prueba (Naranja). Esto para el primer caso de entrenamiento con los 3 canales correspondientes a entropía, masa y temperatura.

IV.1. Valores Reales vs Valores Predichos

Para cada uno de los casos de entrenamiento se realizaron mapas de color donde se comparan los valores de la tasa de formación estelar específica de la simulación contra los obtenidos por la red neuronal para las galaxias de prueba (Fig. 4.2). También en cada gráfica se muestra el valor del coeficiente de correlación de Pearson. En el histograma de la Figura 4.1, se muestra la distribución de los valores de la tasa de formación estelar

especifico para las galaxias de entrenamiento, y el valor predicho por la red para los datos de prueba para el primer caso de entrenamiento con 1500 épocas.

En este punto es importante recordar que la finalidad de este trabajo no es encontrar la mejor combinación de imágenes-red neuronal para la predicción de valores exactos de formación estelar específica, sino aprovechar la capacidad de las redes para identificar patrones, y encontrar las cualidades de las imágenes de entrenamiento de las cuales la red está aprendiendo para obtener estas predicciones. De esta manera esperamos obtener información de los procesos físicos involucrados en la formación de estrellas.

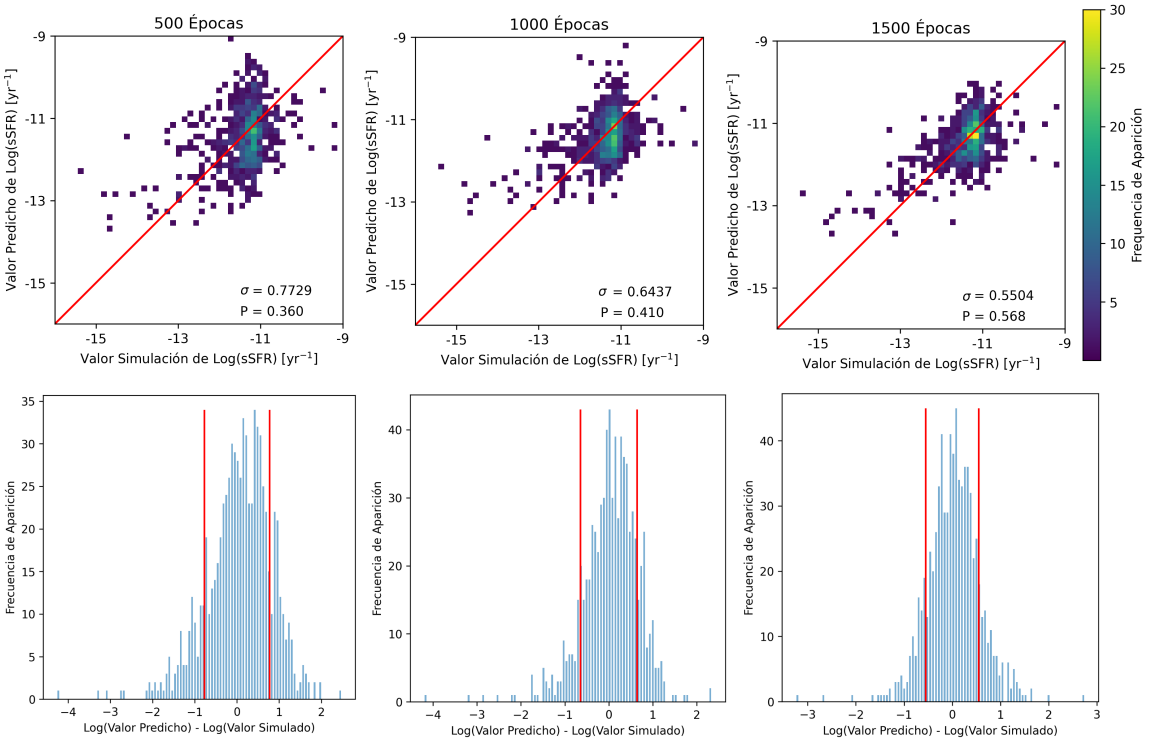

Figura 4.2: Comparación de los valores simulados vs valores predichos por la red para el caso de imágenes compuestas con los canales correspondientes a entropía, masa y temperatura, durante 500, 1000 y 1500 épocas de entrenamiento. Los histogramas muestran la distribución de la diferencia entre los valores simulados y predichos en escala logarítmica, las líneas marcadas rojas, representan los límites marcados por la desviación estándar.

IV.1.1. Análisis de Imágenes Compuestas

En este primer caso se hizo un estudio para definir el número de épocas ideal a utilizar en todos los entrenamientos, el cual finalmente se fijo en 1500 épocas. Se adoptó este valor ya que para esta instancia del entrenamiento, se puede observar una convergencia entre los valores simulados y los predichos por la red suficiente para determinar que la red está obteniendo la información necesaria de las imágenes para realizar un entrenamiento satisfactorio. Se puede observar en la Fig. 4.2 cómo para 1500 épocas la frecuencia de aparición de las predicciones empiezan a converger hacía la relación 1 a 1 comparados con los valores obtenidos de la simulación. Es claro que dándole a la red el número de épocas suficiente las predicciones, así como el valor de correlación de Pearson, serán cada vez mejores. En esta misma imagen se observan los histogramas de las diferencias del valor simulado y el predicho en escala logarítmica, las líneas rojas representan la desviación estándar de estas diferencias, donde podemos encontrar una misma convergencia hacia valores cada vez más pequeños en este caso.

Los tiempos típicos de entrenamiento para 1500 épocas rondan las 7 horas para cada caso de estudio en un equipo con un Intel Xeon CPU E5-2620 v4 2.10 GHz, 2 GPU Nvidia Titan-X 12GB y 64 Gb de Ram. Es importante resaltar que, aunque los ajustes no son los mejores en cuanto a valores de correlación y dispersión de los datos, lo importante es encontrar esta convergencia y tendencia del valor de correlación de Pearson a crecer. A mayor número de épocas terminaríamos encontrando, claro está, mejores resultados, pero en los casos dónde no haya información, por más que se aumente el número de épocas de entrenamiento no encontraría la información necesaria para converger hacia valores cada vez mejores.

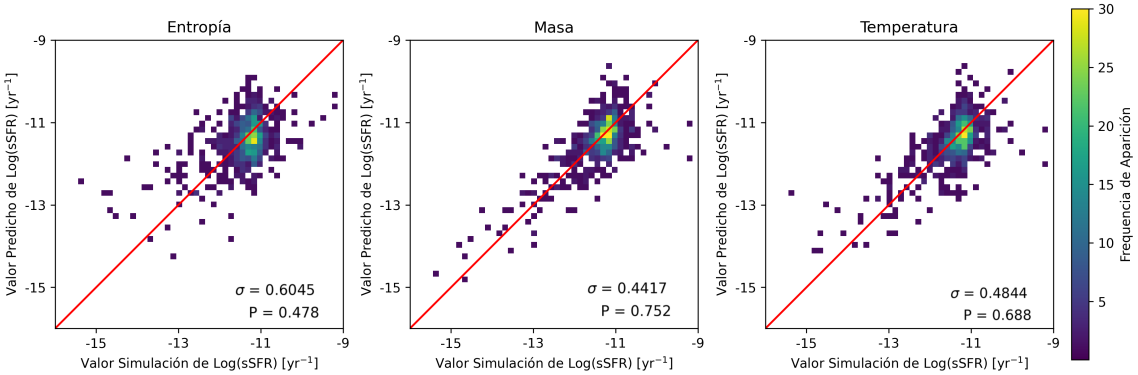

Figura 4.3: Comparación de los valores simulados vs valores predichos por la red para el caso de imágenes con sólo un canal, entropía, masa y temperatura de izquierda a derecha, durante 1500 épocas de entrenamiento.

IV.1.2. Análisis de Entropía, Masa y Temperatura

El siguiente análisis se realizó entrenando la red neuronal con los canales de entropía, masa y temperatura por separado (ver figura 4.3). Nuestros resultados muestran que para la red es más fácil obtener información del canal de masa obteniendo un coeficiente de correlación de Pearson de 0.752 además que en este caso se observa menos dispersión para el rango de $10^{-17}\ yr^{-1} < sSFR < 10^{-12}\ yr^{-1}$, es decir para galaxias con menor formación estelar. El caso del canal de temperatura también muestra resultados interesantes, ya que muestra un ajuste mejor que los obtenidos del entrenamiento de los 3 canales juntos. Esto sugiere que el concentrar un mayor número de neuronas en un solo canal es mejor para el entrenamiento comparado con pasarle los canales en conjunto a la red. En el caso del canal de entropía se obtienen resultados muy similares a los de los 3 canales juntos. En resumen, podemos decir que según los resultados obtenidos del entrenamiento, parece existir una relación entre la distribución de masa estelar y la distribución de temperatura en la galaxia y su formación estelar específica, también para la entropía pero en menor medida. Podemos concluir que es mejor analizar canal por canal ya que en general se

obtienen mejores resultados.

IV.1.3. Análisis con Filtro Gaussiano

Al aplicar el filtro Gaussiano (ver sección **III.1.4**) sobre las imágenes los resultados (ver figura 4.4) nos muestran que nuestra red neuronal está obteniendo información importante de la intensidad total de la imagen, ya que para el caso del suavizado con un radio de píxel de 10, en el que no se aprecian estructuras en las imágenes de las galaxias, el ajuste es mejor que en los dos casos con radio 5 y 3 píxeles. En general los resultados son muy parecidos a los de las imágenes originales, por lo que podemos inferir que suavizar la imagen no compromete la información de donde la red obtiene la información referente a la tasa de formación estelar. Este resultado es muy importante, ya que podríamos esperar en un trabajo a futuro en estudios con galaxias con menos resolución aún resultados importantes, donde quizá no podamos determinar regiones concretas de la galaxia, pero si hacerlo con galaxias en distintas etapas de su evolución.

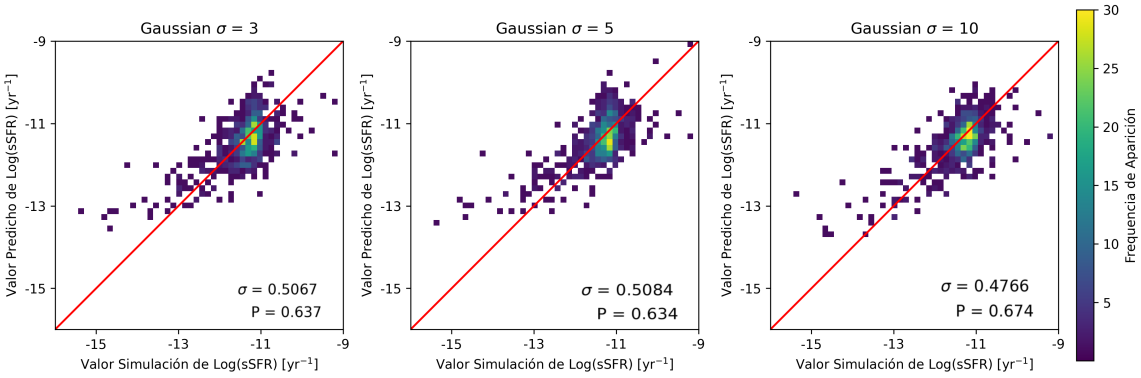

Figura 4.4: Comparación de los valores simulados vs valores predichos por la red para el caso de imágenes aplicando un filtro gaussiano de radio 3, 5 y 10 píxeles que corresponden a $700, 1170$ y $2340\ h^{-1}$pc de izquierda a derecha, durante 1500 épocas de entrenamiento.

El filtro Gaussiano es aplicado a los canales por separado, de modo que la información

de los canales no se ve comprometida, por lo que este estudio es similar a obtener formación estelar directamente del flujo integrado sobre toda la galaxia. Sería necesario un estudio más detallado y una arquitectura de red distinta para determinar de cual de los canales está obteniendo más información al usarse los 3 canales en conjunto.

Los resultados de este estudio se deben revisar con cuidado, recordemos que anteriormente se ha demostrado que las redes neuronales son muy buenas para mejorar la resolución de imágenes e incluso recuperar imágenes desenfocadas [77, 78].

IV.1.4. Análisis de Estructuras

Finalmente, en el estudio de estructuras (ver sección **III.1.4**) encontramos que a escalas grandes la red no encuentra información importante para obtener buenos resultados, mientras que las estructuras de pequeña escala (menores a $\sim 700h^{-1}$pc) contienen información relevante a procesos de formación estelar (ver figura 4.5). Es importante notar que las imágenes procesadas con el filtro de enfoque prácticamente no contienen información de intensidad en ninguno de los canales, por lo que la información se obtiene directamente de las estructuras internas de la galaxia y su distribución en las imágenes.

El uso de esta técnica, funciona cómo un mapeo de estructuras a distintas escalas, que se pueden relacionar a su vez con zonas de éstas escalas físicas, como son nubes moleculares, complejos estelares, y estructura de brazos espirales (Fig. 2.6).

IV.1.5. Resumen de Resultados

Dados los resultados obtenidos podemos afirmar que la distribución de masa y temperatura, así como la intensidad global de estos canales están fuertemente relacionados con los valores de formación estelar específicos. Al mismo tiempo, en cuestión de escalas espaciales, la parte más importante para obtener información de formación estelar se encuentra en escalas menores a $\sim 700\ /h^{-1}$pc. Basándonos en esta escala física los procesos que buscamos asociados a nubes gigantes moleculares Sec. **II.2** y 2.6. Las figuras 4.6,

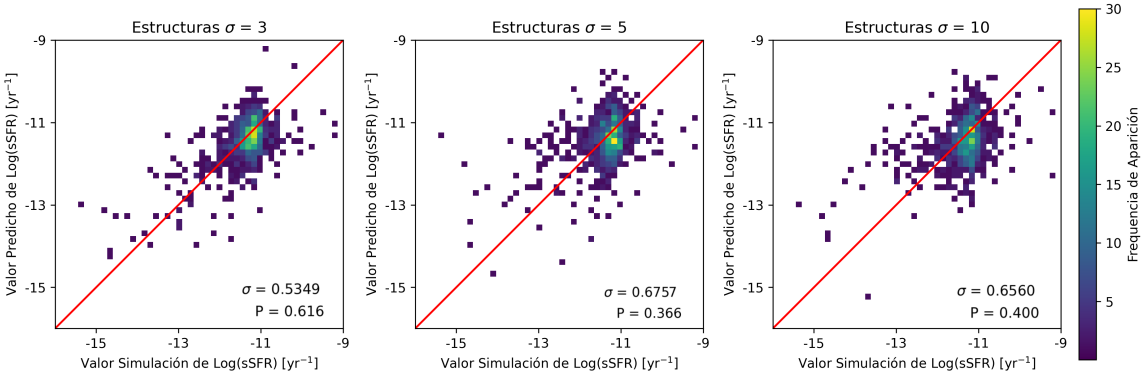

Figura 4.5: Comparación de los valores simulados vs valores predichos por la red para el caso de imágenes que muestran la estructura de la galaxia a diferentes escalas, aplicando máscaras de enfoque de radio 3, 5 y 10 píxeles que corresponden a $700, 1170$ y 2340 h^{-1}pc de izquierda a derecha a la galaxia original, durante 1500 épocas de entrenamiento.

4.7, 4.8, presentan una muestra típica de galaxias, para cada uno de los casos donde se obtuvieron los mejores resultados. Es evidente que visualmente, no se podrían inferir tan fácilmente los patrones de las relaciones con los valores de formación estelar específico que se obtienen al realizar el entrenamiento con la red neuronal.

Para todos los casos de estudio es evidente la falta de robustez de nuestros datos para galaxias de baja formación estelar, esto es, para el rango de 10^{-17} $yr^{-1} < sSFR < 10^{-12}$ yr^{-1}, ya que en esta zona los valores predichos tienen una mayor dispersión en la mayoría de los casos.

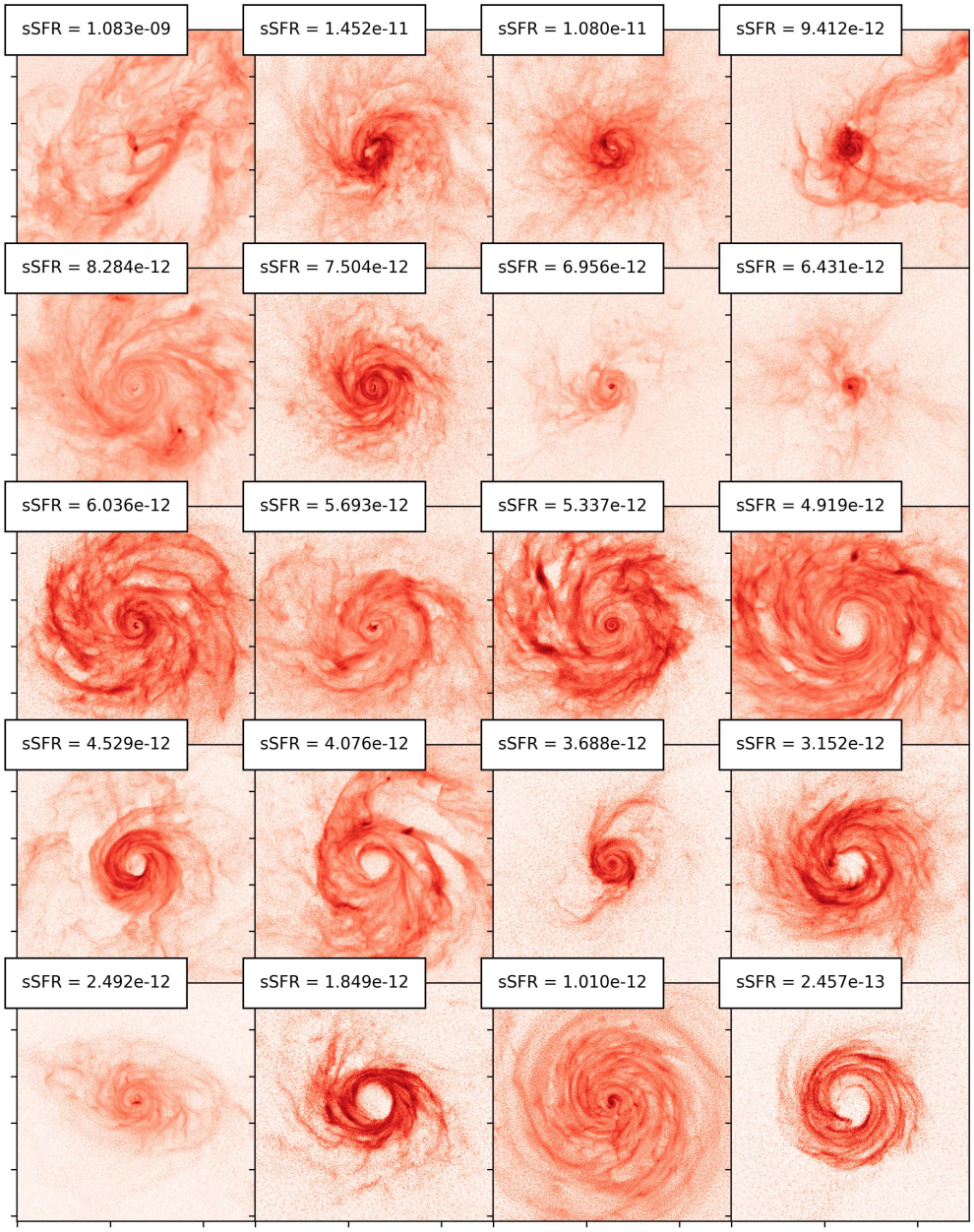

Figura 4.6: Muestra típica de las galaxias de entrenamiento en el canal de distribución de masa y su valor de formación estelar especifica. Las unidades de la formación estelar especifica están dadas en yr^{-1}. Cada lado de la imagen representa $60h^{-1}\mathrm{kpc}$.

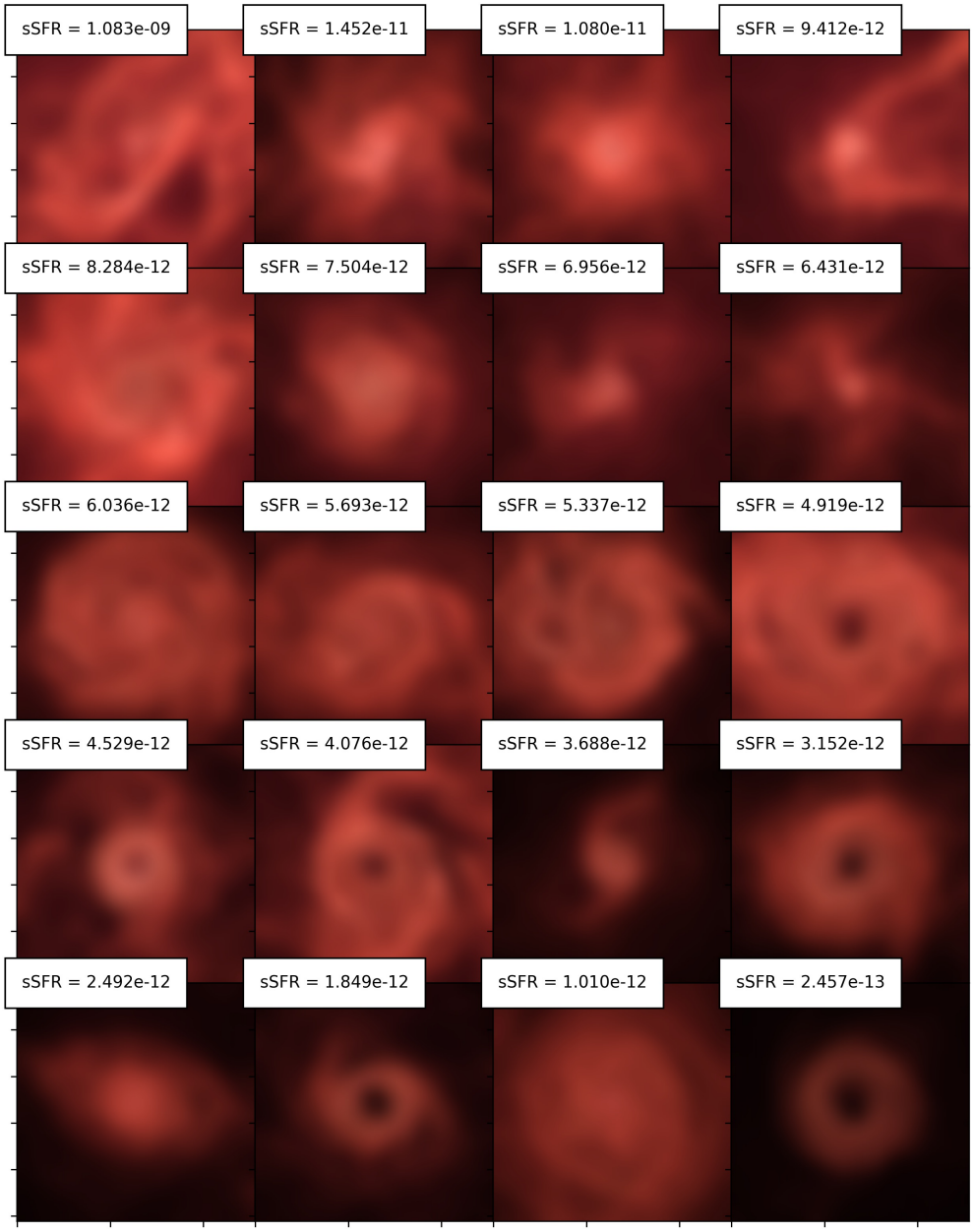

Figura 4.7: Muestra típica de las galaxias de entrenamiento con un suavizado gaussiano con radio de 10 píxeles y su valor de formación estelar específica. Las unidades de la formación estelar especifica están dadas en yr^{-1}. Cada lado de cada imagen representa $60h^{-1}\mathrm{kpc}$.

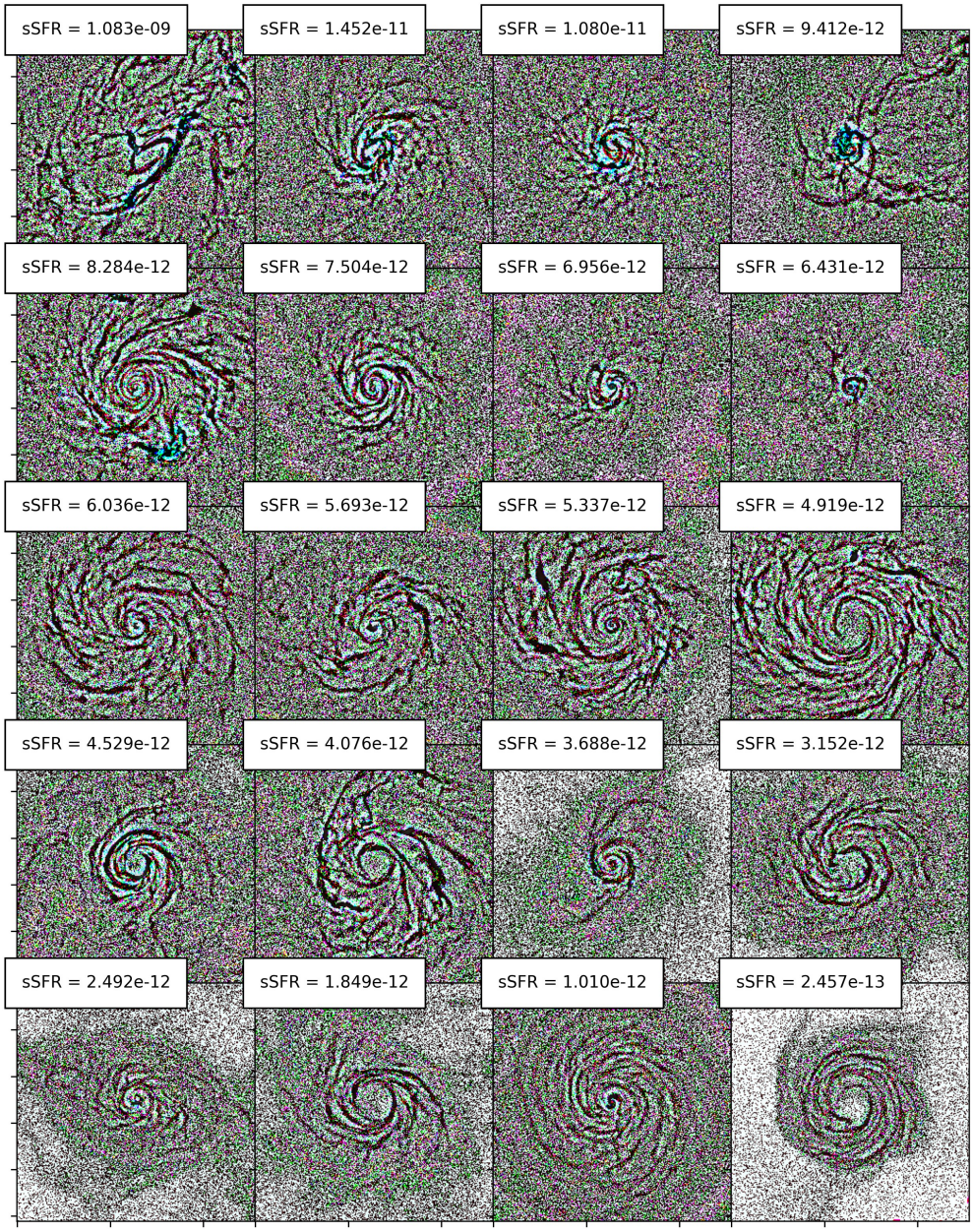

Figura 4.8: Muestra típica de las galaxias de entrenamiento en estructuras con radio de 3 píxeles y su valor de formación estelar específica. Las unidades de la formación estelar especifica están dadas en yr^{-1}. Cada lado de cada imagen representa $60h^{-1}$kpc.

CAPÍTULO V

CONCLUSIONES

El método utilizado en este trabajo muestra su eficacia para identificar patrones en distintos sets de imágenes, y con esto es posible analizar las propiedades de los conjunto de datos que ayudan a la red neuronal a determinar los valores de interés. En este caso se puede concluir que los mapas de distribución de masa y temperatura, así como su intensidad, y las escalas pequeñas en las galaxias, son importantes para determinar los procesos que intervienen en la formación estelar. De cierta manera, podemos decir dado que la combinación de estas características están muy asociadas a las de las nubes moleculares gigantes (teniendo en cuenta que estamos trabajando solo con el gas), que lo que está realizando la red neuronal es detectar estas zonas en nuestras imágenes, ya que como sabemos de la literatura descrita en el capitulo **II**, estás son las zonas en donde se suelen encontrar estrellas jóvenes, y formación estelar activa.

Se pueden hacer estudios más detallados, por ejemplo para explorar la información disponible en estructuras más pequeñas, o analizar los distintos canales de entrenamiento. Sin embargo, como se mencionó, la finalidad de este trabajo es demostrar que la aplicación de la técnica es factible y funcional. Con base en el trabajo realizado, será posible, generando distintos datos de entrenamiento (variando canales y resoluciones) con los datos disponibles en Illustris, para poder determinar cuáles son las propiedades más importantes para la determinación de la formación estelar en galaxias.

Este trabajo muestra un ejemplo de otra aplicación práctica del uso de redes neuronales, aprovechando sus cualidades para poder identificar patrones donde visualmente son difíciles de encontrar, y con esto determinar relaciones directas entre propiedades de interés y procesos físicos. Se optó por datos de simulación de Illustris en lugar de datos observacionales por el fácil acceso a distintos campos a los cuales se puede acceder y por tanto se puede ampliar el estudio a éstos, además de la posibilidad de alinear galaxias para

poder analizar la mayor parte de su estructura. En un intento por ir más a detalle incluso se puede generalizar en lugar de imágenes 2d, hacerlo con cubos 3d, donde se tendría que cambiar la estructura de la red neuronal utilizada, y los tiempos de entrenamiento incrementarían, pero en principio uno esperaría mejores resultados, dado que la red tiene más información disponible.

V.1. Trabajo a Futuro

En este primer trabajo se eligieron los canales convenientemente para el análisis, ya que se sabe que la masa, temperatura, están fuertemente relacionados con la formación estelar. Lo novedoso sin embargo, es la estrategia implementada, ya que puede extenderse a estudiar los campos de velocidad, la distribución de materia oscura, campos magnéticos, entre otras propiedades físicas que posee la simulación de Illustris, y con esto poder determinar si de alguna forma se relacionan con la formación estelar en las galaxias. Incluso se puede realizar un estudio con las 8 bandas de magnitudes estelares disponibles en la simulación, esto con la intención de encaminar junto con una generalización a galaxias con distinta orientación para en un futuro poder aplicarlo a datos observacionales.

Esto, así como explorar la información en escalas espaciales más pequeñas y distintos canales de entrenamiento, son parte del trabajo que se plantea a futuro. Además, este tipo de estrategias se puede extender a otras áreas de estudio, ya que no se limitan a la astronomía o a Illustris, sino a cualquier área con una base de datos suficientemente robusta para generar conjunto de datos de imágenes con distintas propiedades físicas del problema de estudio.

ANEXOS

Split Data

Este código se utiliza para hacer una separación de las imagenes de galaxias, en datos de entrenamiento, y datos de prueba.

```
import os
import random
import numpy as np
from shutil import copyfile
PATH = '/home/'
try:
  os.mkdir(PATH + 'training')
  os.mkdir(PATH + 'testing')
except OSError:
    pass
def split_data(SOURCE, TRAINING, TESTING, SPLIT_SIZE):
    files = []
    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")
    training_length = int(len(files) * SPLIT_SIZE)
    testing_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0:training_length]
    testing_set = shuffled_set[-testing_length:]
    for filename in training_set:
        this_file = SOURCE + filename
```

```
26          destination = TRAINING + filename
27          copyfile(this_file, destination)
28      for filename in testing_set:
29          this_file = SOURCE + filename
30          destination = TESTING + filename
31          copyfile(this_file, destination)
32  SOURCE_DIR = PATH + 'Images/'
33  TRAINING_DIR = PATH + 'training/'
34  TESTING_DIR = PATH + 'testing/'
35  split_size = .9
36  print('Split Data...')
37  split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)
```

Pipeline

Código de entrenamiento completo. Para utilizarlo es necesario tener los archivos guardados *allsfr.pkl* y *valid.pkl* descritos en la sección **III**, así como tener las imagenes de las galaxias separadas en datos de entrenamiento y datos de prueba.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import random
import pickle
import PIL
from scipy.stats import gaussian_kde
from shutil import copyfile
import tensorflow as tf
import tensorflow.compat.v1 as tf
# Admitir el acceso ilimitado a memoria del GPU
physical_devices = tf.config.experimental.list_physical_devices('GPU')
tf.config.set_visible_devices(physical_devices[0:1], 'GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.layers import Input, Conv2D, Lambda, Dense, Flatten,
    Concatenate, MaxPooling2D, GlobalAveragePooling2D, Dropout
from keras.layers import Layer
from keras.models import Model
from keras.callbacks import ModelCheckpoint
PATH_DATA = '/home/'
```

```
26 TRAINING_DIR = '/home/training/'
27 TESTING_DIR = '/home/testing/'
28 file_name1 = '/home/allsfr.pkl'
29 file_name2 = '/home/valid.pkl'
30 open_file1 = open(file_name1, "rb")
31 open_file2 = open(file_name2, "rb")
32 sfr = pickle.load(open_file1)
33 valid = pickle.load(open_file2)
34 nsfr = []
35 nid = []
36 nsnap =[]
37 IDTRA = []
38 SFRTRA = []
39 IDTES = []
40 SFRTES = []
41 for i in range(len(sfr)):
42   nsfr = np.append(nsfr, sfr[i])
43   nid= np.append(nid, valid[i])
44   nsnap= np.append(nsnap, np.full(len(sfr[i]), 99-i))
45 for i in range(len(nsfr)):
46   if os.path.exists(PATH_DATA + 'training/' + 'Snap%dID%d.png'%(nsnap[i
     ], nid[i])):
47     SFRTRA = np.append(SFRTRA, nsfr[i])
48     IDTRA = np.append(IDTRA, 'Snap%dID%d.png'%(nsnap[i], nid[i]))
49   if os.path.exists(PATH_DATA + 'testing/' + 'Snap%dID%d.png'%(nsnap[i],
      nid[i])):
50     SFRTES = np.append(SFRTES, nsfr[i])
51     IDTES = np.append(IDTES, 'Snap%dID%d.png'%(nsnap[i], nid[i]))
52 dftra = pd.DataFrame({'filename' : IDTRA, 'sfr': SFRTRA}, columns = ['
    filename', 'sfr'])
53 dftes = pd.DataFrame({'filename' : IDTES, 'sfr': SFRTES}, columns = ['
    filename', 'sfr'])
```

```
xtrain = []
ytrain = []
xtest =[]
ytest = []
print("Cargando Imagenes para Entrenamiento")
for i in range(len(dftra['filename'])):
  if dftra['sfr'][i] != 0:
    xtrain.append( np.asarray(PIL.Image.open(PATH_DATA + 'training/' +'%s'%dftra['filename'][i])))
    ytrain.append( dftra['sfr'][i])
for i in range(len(dftes['filename'])):
  if dftra['sfr'][i] != 0:
    xtest.append( np.asarray(PIL.Image.open(PATH_DATA + 'testing/' +'%s'%dftes['filename'][i])))
    ytest.append( dftes['sfr'][i])
xtrain = np.array(xtrain, dtype='uint8')
ytrain = np.log10(1E15*np.array(ytrain))
xtest = np.array(xtest, dtype='uint8')
ytest = np.log10(1E15*np.array(ytest))
print(ytest)
model = VGG16(weights='imagenet', include_top=False, input_shape = (256, 256, 3))
#-------------------------------------------------------
#       Congelamos los pesos de la red VGG16
# para solamente entrenar los pesos de las capas densas.
#-------------------------------------------------------
for layer in model.layers:
    layer.trainable = False
#Agregamos capas densas a la salida de la red pre-entrenada
x = model.output
x = Conv2D(1, (1, 1),  activation='relu')(x)
x = Flatten()(x)
```

```
x = Dense(64, activation="relu")(x)
x = Dense(128, activation="relu")(x)
x = Dense(256, activation="relu")(x)
x = Dense(512, activation="relu")(x)
x = Dense(256, activation="relu")(x)
x = Dropout(0.2)(x)
out_val = Dense(1, activation = "relu", name="out_val")(x)
model = Model(inputs=model.input, outputs=out_val)
model.summary()
model.compile(loss='mse', optimizer='Adadelta',metrics=['mse'])
model_checkpoint = ModelCheckpoint('Test.h5', monitor='loss',
    save_best_only=True)
model.fit(xtrain[:,:,:,:3], ytrain, epochs = 1500, batch_size = 100,
    verbose = 1, callbacks=[model_checkpoint])
ypred = model.predict(xtest[:,:,:,:3])
np.save('YValues.npy', np.array([ytest, ypred[:,0]]))
```

APÉNDICE

Artículo Publicado

Como trabajo previo al desarrollo del trabajo central de la tesis, y cómo parte de investigación previa en el uso y diseño de redes neuronales, se trabajó con redes convolucionadas en forma de un tipo especifico conocido como "Autoencoders" en un estudio de imágenes sintéticas de perfiles de galaxias.

En este artículo, se propone un Autoencoder Semántico, en la cual mediante un "Encoder" que es un tipo especial de red neuronal que reduce el número de neuronas en cada capa, con el fin de obtener una representación de las imágenes de entrada con un número de parámetros reducido. Lo que se propone es que estos parámetros que se obtienen sirvan como parámetros de un modelo físico definido, que a su vez genera una nueva imagen que nos sirve para evaluar la calidad de nuestro entrenamiento comparando la imagen original contra la generada por el modelo físico.

Como ejemplo práctico se generan 30,000 imágenes 2D de perfiles exponenciales de galaxias con cierta inclinación, cuya parametrización se puede hacer en 3 variables, esto es $I \propto exp(-r'(A, e, \theta))$, las cuales cumplen el modelo propuesto en la Sección 3 de este artículo, las cuales se usan para entrenar el "Encoder" siguiendo la idea descrita anteriormente, obteniendo unos resultados satisfactorios, salvo el caso ambiguo de la transición de $180^\circ < \theta < 0^\circ$ al ser discontinua. La técnica muestra excelentes resultados, incluso en imágenes con ruido sintético.

Un importante resultado de esta técnica es la rapidez con la que se pueden obtener parámetros para modelos previamente estudiados, siendo de apenas un par de segundos (una vez entrenada la red). Particularmente, esto puede ser de gran utilidad para encontrar valores iniciales a modo de complemento al utilizar rutinas de ajuste de parámetros.

Mon. Not. R. Astron. Soc. **000**, 1–?? (2002) Printed 2 March 2022 (MN LATEX style file v2.2)

Self-supervised Learning with Physics-aware Neural Networks I: Galaxy Model Fitting

M.A. Aragon-Calvo[1], J. C. Carvajal[1] ⋆
[1]*Instituto de Astronomía, UNAM, Apdo. Postal 106, Ensenada 22800, B.C., México*

ABSTRACT

Estimating the parameters of a model describing a set of observations using a neural network is in general solved in a supervised way. In cases when we do not have access to the model's true parameters this approach can not be applied. Standard unsupervised learning techniques on the other hand, do not produce meaningful or *semantic* representations that can be associated to the model's parameters. Here we introduce a self-supervised hybrid network that combines traditional neural network elements with analytic or numerical models which represent a physical process to be learned by the system. Self-supervised learning is achieved by generating an internal representation equivalent to the parameters of the physical model. This semantic representation is used to evaluate the model and compare it to the input data during training. The *Semantic Autoencoder* architecture described here shares the robustness of neural networks while including an explicit model of the data, learns in an unsupervised way and estimates, by construction, parameters with direct physical interpretation. As an illustrative application we perform unsupervised learning for 2D model fitting of exponential light profiles and evaluate the performance of the network with noisy images.

Key words: Cosmology: large-scale structure of Universe; galaxies: kinematics and dynamics; methods: data analysis, N-body simulations

1 INTRODUCTION

Fitting a model to a given set of observations is a basic paradigm in science. Model fitting allows us to estimate the value of the parameters defining the model and also serves as a tool for testing its validity. Traditionally, this task has been approached as an optimization problem where we wish to minimize the error between the model G and a set of observations $\boldsymbol{x}$ as done in regression techniques (Legendre A. M., 1805). Model fitting can be a challenging task as often the result is highly sensitive to the set of initial parameters, noise and artifacts in the data.

In recent years, new techniques developed in the field of Machine Learning (ML) have been successfully applied to problems where the complexity of the data prevented the use of traditional techniques. One of the fields that has seen the largest advances is image processing, where Deep Neural Network (DNN) architectures have now reached and even surpassed human performance (LeCun Y., Bengio Y., Hilton G. 2015; Dieleman, Willett & Dambre 2015). At the core of the ML revolution is the ability of DNN to generate an internal hierarchical representation of the patterns in the data. However, they do so at the cost of a lack of transparency. Typical DNN systems contain millions of trainable parameters with no direct physical interpretation, effectively making them black boxes. This lack of transparency in the inner workings of DNN still limits their full application in quantitative sciences. Currently, DNN are mainly used in astronomy as workhorses for classification, segmentation and regression tasks (Dieleman, Willett & Dambre 2015; Soumagnac, et al. 2015; Kim & Brunner 2017; Tuccillo D. et al. 2018; Domínguez Sánchez, Huertas-Company, Bernardi, Tuccillo & Fischer 2018; Ribli, Dobos & Csabai 2019). However, their use as tools for insight extraction of physical processes remains poorly exploited. The work presented here aims to be one step forward in this direction by explicitly introducing physical models in a regular neural net architecture.

1.1 Autoencoders

Autoencoders belong to a family of neural networks that are trained to copy its input to its output. While replicating the input is a trivial task the utility of autoencoders stems

⋆ E-mail:maragon@astro.unam.mx

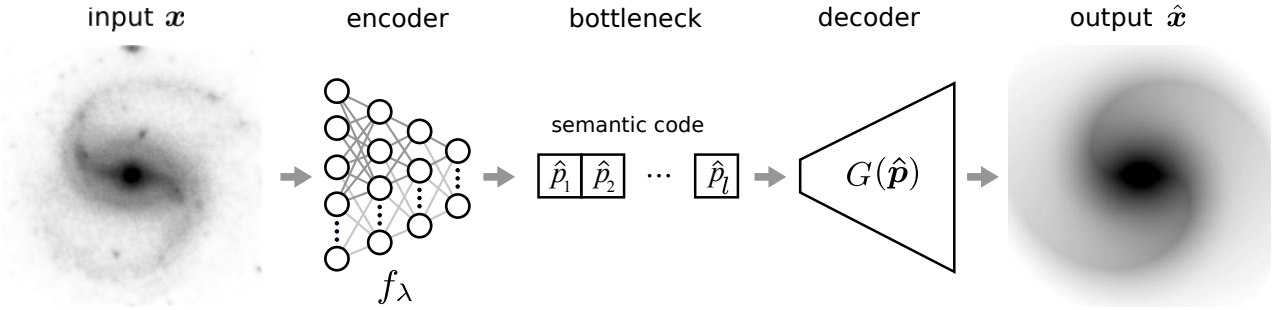

Figure 1. Semantic autoencoder. The input image $\boldsymbol{x}$ represents a measurement taken from some process Ψ. The encoder f_λ consists of a standard deep neural network that maps the input into the semantic parameters $\hat{\boldsymbol{p}}$. The decoder model G evaluates $\hat{\boldsymbol{p}}$ producing the reconstructed output $\hat{\boldsymbol{x}}$. The decoder G is a model that can be analytically or numerically solved and has not trainable parameters. Note that the actual output of interest of this architecture is the semantic code at the bottleneck, not the output image.

form their ability to map the high-dimensional input into a low-dimensional compressed internal representation (LeCun Y., 1987; Bourlard H., Kamp Y., 1988; Hinton G., Salakhutdinov R., 2006).

A standard autoencoder is composed of an encoder and a decoder joined at a bottleneck (see Fig. 1). The encoder is a neural network f_λ (described by a set of trainable parameters λ consisting of weights and biases) that maps the input $\boldsymbol{x} \in \mathbb{R}^m$ into a code:

$$\boldsymbol{c} = f_\lambda(\boldsymbol{x}) \in \mathbb{R}^l \tag{1}$$

The code is then fed into a decoder neural network g_λ that "decompresses" it back to input space producing a reconstruction of the input:

$$\hat{\boldsymbol{x}} = g_\lambda(\boldsymbol{c}) = g_\lambda(f_\lambda(\boldsymbol{x})) \in \mathbb{R}^m. \tag{2}$$

Learning the optimal λ is performed by minimizing the loss function $L(\boldsymbol{x}, \hat{\boldsymbol{x}})$ chosen to quantify the difference between the input and the reconstructed output.

In order for autoencoders to produce a compressed representation they are, by construction, undercomplete i.e. the dimensionality of the code is lower than that of the input ($m > l$). One consequence of this is that the reconstruction will not be an exact copy of the input data. In fact, it can be desirable that the reconstructed output is a simplified version of the input, meaning that only the relevant aspects of the input image are encoded (Vincent P. et al. 2010).

Autoencoders are attractive architectures for data encoding due to their ability to create a compressed representation at their bottleneck. This representation is in general not semantically meaningful (although it has been shown that sparcity can lead to semantical encoding for simple physical models (Iten R., et al., 2018)). In general sparsity is a required but not sufficient condition for semantic encoding. This is also the case of other dimensionality reduction techniches such as Principal Component Analysis, for which the linear autoencoder produces equivalent encodings (Bourlard H., Kamp Y., 1988).

2 SEMANTIC AUTOENCODERS

In this section we present the motivation and description of the basic hybrid neural network architecture introduced in this paper.

The lack of semantic encoding in regular autoencoders discussed in the previous section is a consequence, in part, of their generality as both the encoder and decoder are based on universal trainable systems (Cybenko G., 1989) with no predefined internal structure. One way of enforcing semantic encoding would be to impose meaningful constraints on the autoencoder. Consider a physical process Ψ from which we take a set of measurements $\boldsymbol{x}$. The process Ψ can be described or approximated by a (mathematical) model G expressed as a function of a set of parameters $\boldsymbol{p}$:

$$\Psi(\boldsymbol{p}) \rightarrow \boldsymbol{x} \approx G(\boldsymbol{p}). \tag{3}$$

We now feed $\boldsymbol{x}$ to an autoencoder that will encode $\boldsymbol{x}$ into the parameters $\hat{\boldsymbol{p}} = f_\lambda(\boldsymbol{x})$ at its bottleneck and then decode it back to produce the reconstructed input $\hat{\boldsymbol{x}} = g_\lambda(\hat{\boldsymbol{p}})$. If we somehow force $\hat{\boldsymbol{p}}$ and $\boldsymbol{p}$ to be semantically equivalent ($\boldsymbol{p} \Leftrightarrow \hat{\boldsymbol{p}}$) then the decoder g_λ will effectively learn to mimic the behavior of the process Ψ.

We can then replace the trainable decoder g_λ by the model G as they are functionally equivalent ($g_\lambda \Leftrightarrow G$). The constraints imposed on the autoencoder by G force the encoder f_λ to generate a semantic encoding of the input. Note that the model G can be any function such as a modeling equation, a numerical code like an N-body solver, or even an actual physical system such as a mechanical arm. Semantic autoencoders do not require to be provided the "true" code $\boldsymbol{p}$ during training (we may not even have access to the code). Instead the rely on the validity of the model G in order to generate reconstructed observations $\hat{\boldsymbol{x}}$ and learn in a self-supervised way.

The *semantic autoencoder* described here is a system combining trainable with non-trainable elements. In this approach we consider neural networks as special cases of a larger family of differentiable functions with trainable or non trainable parameters that can learn via gradient descent Wengert, R. E, (1964); Bischof C. H., (1996).

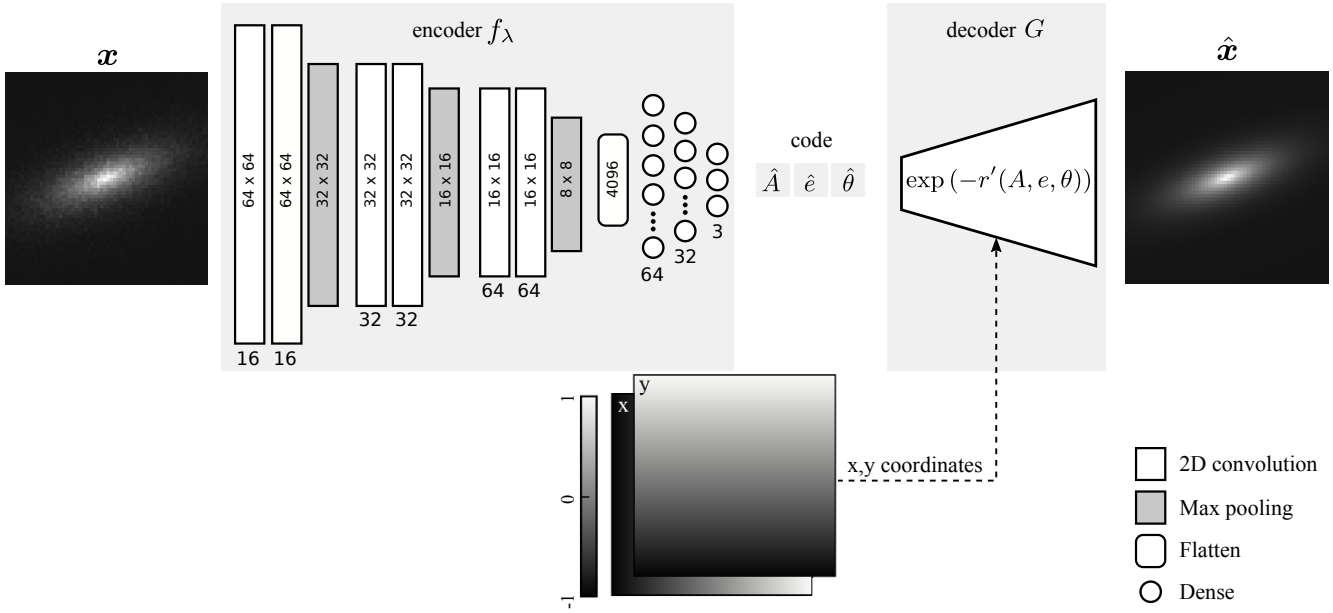

Figure 2. Implementation of a semantic autoencoder trained to estimate the parameters A, e and θ defining a 2D exponential light profile. The input (noisy) image $\hat{\boldsymbol{x}}$ is feed to an encoder f_λ consisting of a deep convolutional neural network and a dense neural network. At the bottleneck we obtain the parameters $\hat{A}, \hat{e}, \hat{\theta}$ which are then used to evaluate the model G (here corresponding to eq. 4) in the decoder, yielding the reconstructed output image. The numbers under the layers indicate the number of neurons/kernel in the layer. The horizontal numbers indicate the activation image size in pixels. During training we compare, via the loss function, the input and output images. Note the auxiliary input consisting of a two-channel 2D array containing the x and y coordinates used to expand the model's parameters to a 2D image.

3 APPLICATION: SELF-SUPERVISED EXPONENTIAL PROFILE FITTING

In this section we present an illustrative application of a semantic autoencoder trained to estimate the parameters defining an exponential profile commonly used to describe the light distribution of late-type galaxies (Sérsic 1963; Peng C. et al. 2002; Simard, Mendel, Patton, Ellison & McConnachie 2011; Barden, Häußler, Peng, McIntosh & Guo 2012). Model fitting can be addressed as a standard supervised problem providing both the input image and the model's parameters (Tuccillo D. et al. 2018; Ribli, Dobos & Csabai 2019). In this approach the quality of the training is only as good as the training data itself. If the parameters used for training were computed from other methods (as it is commonly done) then any potential bias from this method will propagate into the training process. Without a model to compare it is not possible to asses the true accuracy of the trained network. In this work instead we train a network in a self-supervised way, i.e. without providing information on the true model's parameters. Here the quality of the training depends on the ability of the model G to describe the observations.

A general exponential profile of a galaxy observed with some inclination and position angle can be expressed as:

$$I(r) = I_0 \exp(-r') \tag{4}$$

where the radius r' is computed as:

$$r' = \sqrt{(x'/A)^2 + (y'/B)^2}. \tag{5}$$

The major and minor semi-axis A, B of the ellipsoid are related to the ellipticity as $e = 1 - B/A$. The rotated coordinates x' and y' are given by:

$$x' = \cos\theta(x - x_0) - \sin\theta(y - y_0) \tag{6}$$
$$y' = \sin\theta(x - x_0) + \cos\theta(y - y_0), \tag{7}$$

where θ corresponds to the position angle of the ellipsoid. For simplicity we set in our analysis $I_0 = 1$ and $x_0, y_0 = 0$.

3.1 Network Implementation

The full model was implemented using the Keras library (Chollet F. et al., 2015) with the tensorflow backend (Abadi, M. et al., 2015). Keras offers a clean API for boilerplate code such as setting up a convolutional layer while allowing direct access to tensorflow for non standard tasks.[1]

Figure 2 shows the implementation of the architecture proposed here. In practice this particular network has three inputs: the image we wish to encode and two 2D arrays containing x and y coordinates used to evaluate the 2D exponential profile. Coordinates could be computed on the fly using the *eager mode* introduced in tensorflow 2.0. However, passing the pre-computed coordinates may be computationally more efficient at the expense of increased memory consumption. For the application presented here this is not an issue.

[1] The code used in this paper can be found in the following repository: https://github.com/miguel-aragon/Semantic-Autoencoder-Paper

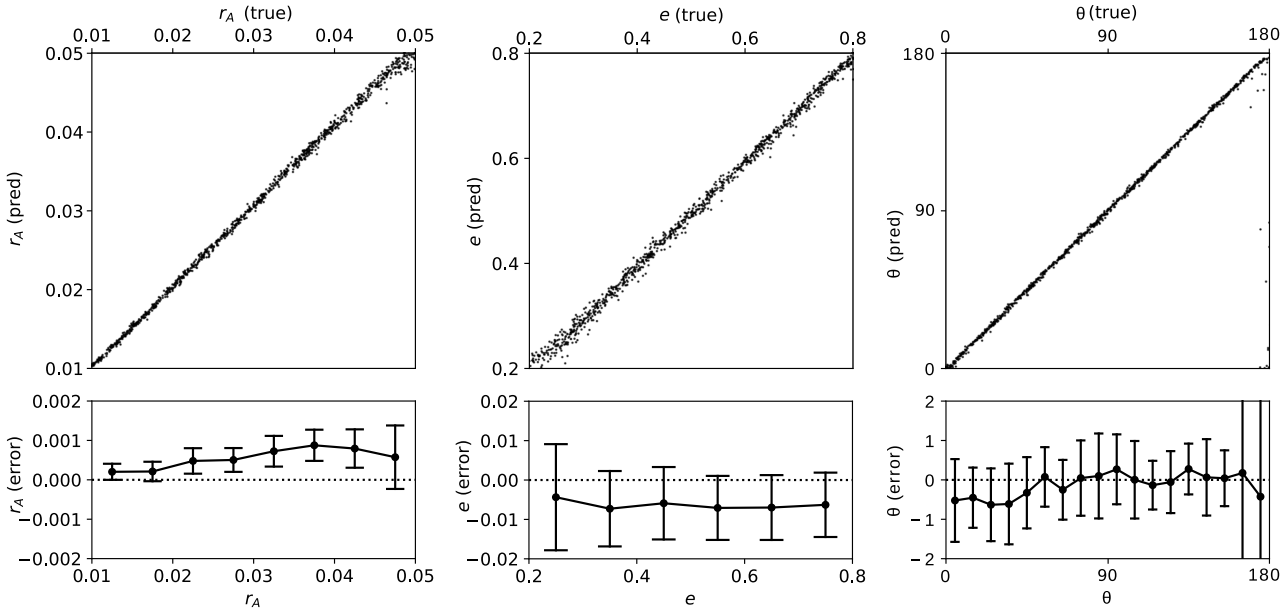

Figure 3. Top panels: true vs. predicted major semi-axis A, ellipticity e and position angle θ (left, center and right panels respectively) computed from 1000 test images without added noise. The dots correspond to individual predictions. Bottom panels: difference between the true and predicted parameters. The solid lines indicate the median error inside each bin and the error bars are computed as the standard deviation of the errors inside each bin. Note the failed predictions near the discontinuous interval in position angle $0° \sim \theta \sim 180°$.

3.1.1 Encoder

The encoder consists of 3 *convolutional blocks* containing each two convolutional layers with rectified linear unit (ReLU) activations followed by a 2×2 max-pooling operation. The number of kernels in the layers of each of the three convolutional blocks is $32, 64$ and 128 respectively. The output of the last convolutional block has 128 channels of 8×8 pixel activation images. This layer is flattened to a 8192 vector which is connected to three dense layers with sigmoid activations containing 64, 32 and 3 neurons respectively. The last three neurons of the encoder output the values of the predicted parameters $\hat{A}, \hat{e}, \hat{\theta}$ (one neuron per parameter). We use dense layers as a convenient way to generate any desired number of output parameters. The encoder contains all the trainable parameters in the network with 812,963 parameters of which 526,531 correspond to the dense network.

The above architecture was chosen after performing a basic manual hyperparameter exploration in order to find the smallest network that would be able to accurately encode the input images. Larger networks produce lower errors at at higher computational cost.

3.1.2 Decoder

The output of the encoder, the parameters $\hat{A}, \hat{e}$ and $\hat{\theta}$, are evaluated by the 2D exponential model in eq. 4 acting as a decoder. In order to "expand" the model's parameters into a 2D image we evaluate the exponential function with the predicted parameters $\hat{A}, \hat{e}, \hat{\theta}$, and the two auxiliary arrays (of the same size as the original image) containing the x and y coordinates with origin at the center of the image and running in the range (-1,1). Note that by construction this network is not translational invariant. The center of the profile could be added to a more complex model and learned during training. The decoder was implemented as a custom tensorflow layer with three inputs: the list of parameters $(\hat{A}, \hat{e}, \hat{\theta})$ coming from the encoder and the 2D arrays containing the coordinates x and y (see Fig. 2). The output of the decoder layer is the reconstructed image $\hat{\boldsymbol{x}}$. The exponential profile in eq. 4 was implemented using native tensorflow functions in order to allow tensorflow to automatically compute derivatives during backpropagation at training.

3.1.3 Training and test data

Training data consisted of a set of 10,000 synthetic images of exponential profiles with a resolution of 64×64 pixels. The parameters A, e, θ of each image were sampled from uniform distributions in the ranges $0.01 < A < 0.05$, $0.2 < e < 0.8$ and $0° < \theta < 180°$. The ellipticity lower limit was chosen to avoid the ill-posed problem of computing a position angle from a circular or nearly circular profile and the higher limit was chosen to avoid unrealistically thin profiles. The position angle θ was normalized in the range $(0, 1)$ to bring it inside the range of the sigmoid functions used as activations in the neurons leading to the bottleneck. For simplicity we set $I_0 = 1$ for all models but this could easily be added as another parameter in the model. The test data consisting of 1000 images was generated with the same parameter intervals as the training data but using a different random seed generator.

3.1.4 Training procedure

Training was done by feeding the model with images in batches of 20 samples for 500 epochs and using 9000 images for training and 1000 for validation. Weights were updated using the ADADELTA optimizer (Zeiler 2012) with the default values of initial learning rate $l_r = 1$ and Adadelta

decay factor $\rho = 0.95$. We used the mean absolute error (*mae*) between the input and reconstructed images as the loss function. Note that this loss function is pixels-based and therefore is potentially prone to instabilities in the case of images with a high dynamic range in which case either a more robust loss function could be used or the images could be rescaled with a non-linear function. However, due to its linear behaviour the *mae* function still produces values that are less sensitive to the high dynamic range of the exponential profiles compared to the more commonly used mean squared error. Training took $\sim$ 2 hours on an nvidia 1060 GPU card.

3.2 Results

After training the network we tested it with 1000 images not included in the training process. Predictions took a few seconds to compute. The network produces two outputs: the reconstructed 2D profile and the parameters $\hat{A}, \hat{e}$ and $\hat{\theta}$ at the network's bottleneck. The parameters correspond to the activations at the last three neurons in the decoder. Figure 3 shows the comparison between the true and predicted parameters as well as their errors. The three parameters are recovered with excellent overall agreement. However the predicted scale is slightly overestimated by $\Delta A = 0.00052$ and the error in the predictions increases with the scale of the profile. This may reflect the effect of the finite size of the image as the receptive field cover less of the profile. On the other hand, the ellipticity is underestimated by $\Delta e = -0.0069$. The errors in the estimated ellipticity are larger for smaller profiles. The error in the position angle estimates is roughly constant across the full range of angles with a median value of $\theta = 0.20^\circ$. This is to be expected as the network should be insensitive to the position angle of the profile. However, the network had difficulty predicting position angles close to 0° and 180° (~ 1 percent of the images) due to the discontinuous nature of the position angle as the network tries to interpolate values close to angles 0° and 180°.

The dispersion of the errors between the true and predicted parameters is $\sigma_A = 0.00049$, $\sigma_e = 0.0096$ and $\sigma_\theta = 1.13^\circ$. We computed σ_θ using only points in the interval $10^\circ < \theta < 170^\circ$ in order to avoid the ambiguous predictions at angles close to 0° and 180°

3.2.1 Effect of noise in parameter prediction

In order to have a more realistic setting and to test the denoising properties of the semantic autoencoder (since the architecture can still be considered a denoising autoencoder) we ran an additional set of training runs as described in Sec. 3.1.3 but this time adding Gaussian noise with dispersion $\sigma_G = 0.05$ and $\sigma_G = 0.1$. While Gaussian noise is a simplification it is sufficient for our purposes. In order to compensate for the addition of noise we trained with 30,000 images divided in 27,000 training images and 9000 validation images. From the trained networks we computed predictions as described in Sec. 3.2.

Figure 4 shows the comparison between the true and predicted parameters computed from the samples with the two noise levels. The dispersion of the errors increase with the noise level. The three parameters show a similar behaviour in the two samples with a small over-estimation of the scale r_A of the order of $\Delta r_A \sim 0.0005$, and an underestimation of the ellipticity e of the order $\Delta e \sim 0.005$. The dispersion in the errors increase with the noise level for the three parameters and follow the same trend as the clean sample with increasing errors with increasing scale and ellipticity values. The errors in the position angle show no trend with angle. We note that the errors in the position angle around $\theta = 180^\circ$ actually decrease in the higher noise sample although this may be an spurious effect.

It is interesting to note that both the mean error and their dispersion is lower in the case of the same with noise compared to the clean sample. The effect is small but consistent and may be the result of noise allowing the network to generalize better. It has been observed that by adding noise to an autoencoder during training it improves its encoding ability by forcing the autoencoder to focus on the relevant aspects of the image and ignore the noise, leading to a more robust network.

Figure 6 shows a visual comparison between noisy images feed to the semantic autoencoder and the images reconstructed from the predicted parameters A, e, θ. Visually, the original images without noise are indistinguishable from the reconstructed images as already indicated by the tight correlation between true and predicted parameters in Fig. 3. The residual of the reconstructed and the input image ($\hat{\boldsymbol{x}} - \boldsymbol{x}$) shows mostly the Gaussian noise applied to the original images. In a sense, the semantic autoencoder presented here acts as an ideal denoising autoencoder provided that the model G encodes the relevant features in the input images.

3.2.2 Network error as a function of number of training parameters

In order to quantify the performance of the network as a function of its size we ran 6 different networks with $\frac{1}{4}$, $\frac{1}{16}$, $\frac{1}{24}$, $\frac{1}{32}$, $\frac{1}{48}$ and $\frac{1}{64}$ of the number of parameters compared to the original network described in Sec. 3.1.1. We kept the same number of layers and only changed the number of neurons in each layer (see table 1). The results are shown in Fig. 5 where we see that the mean errors in the predictions of the network (measured as the mean value of the errors in the predicted parameters) decrease with network size. The trend is more pronounced in the $1/64 - 1/16$ cases with larger networks having a slightly better performance afterwards. Figure 5 can be used as a guideline to determine the optimal network size.

4 DISCUSSION AND FUTURE WORK

We presented the Semantic Autoencoder, a hybrid Deep Learning architecture that combines standard neural network elements with an explicit model describing the measured input data. We applied this architecture to the problem of self-supervised 2D exponential profile fitting with excellent results. The network is able to accurately estimate the parameters of the model with the only exception of position angles close to the ambiguous case of $0^\circ \sim \theta \sim 180^\circ$. This is the result of the network trying to interpolate in the discontinuous interval.

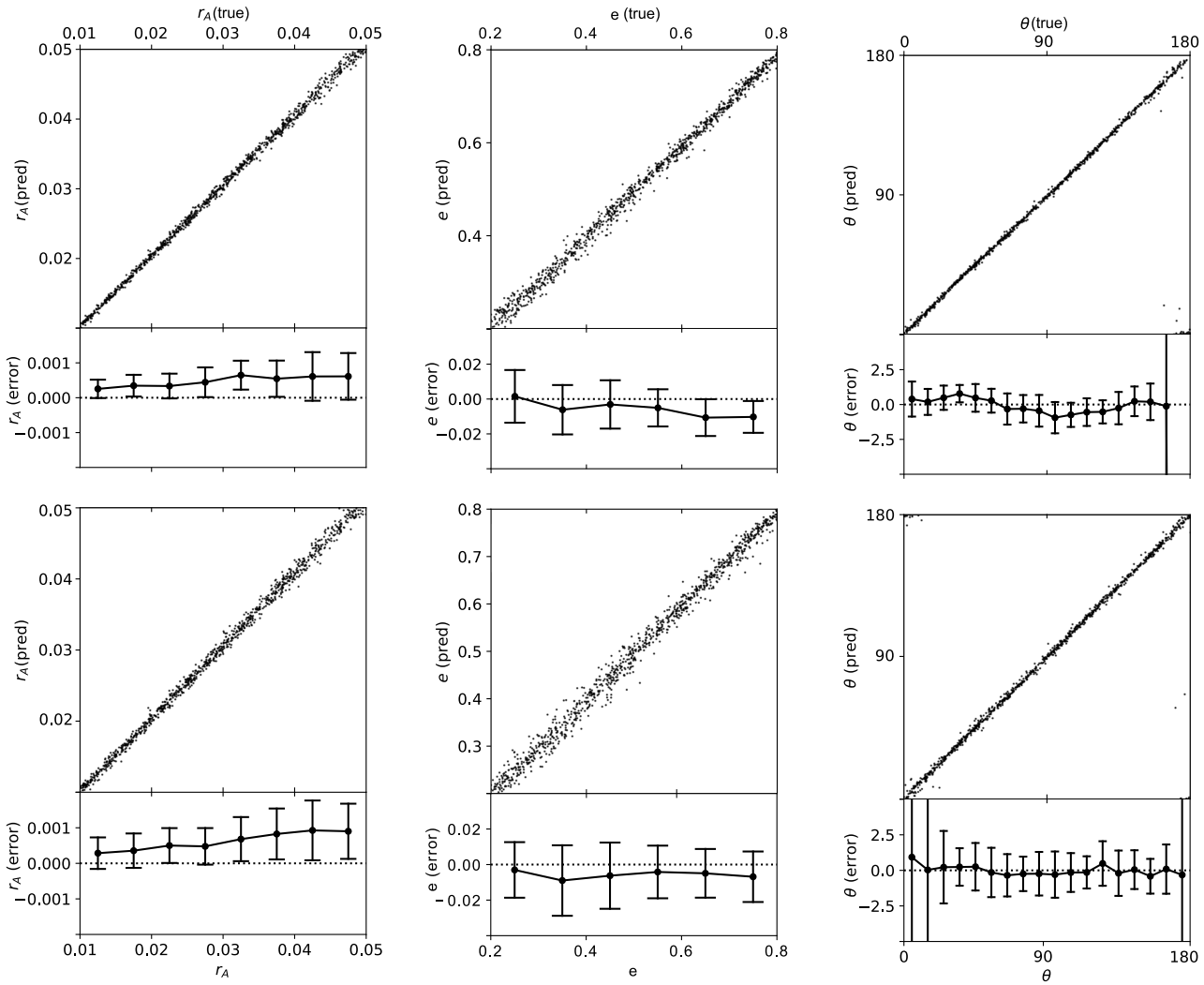

Figure 4. Effect of noise in the predicted parameters. See Fig. 3 for details. The two top figures correspond to the run with $\sigma_G = 0.05$ while the two bottom figures correspond to the run with $\sigma_G = 0.1$ (see text for details). Note that the scale of the error bars is larger in this figure compared to Fig. 3.

	Block 1		Block 2		block 3		Block 4		Block 5
No. Parameters	**2D Conv**	**2D Conv**	**2D Conv**	**2D Conv**	**2D Conv**	**2D Conv**	**Dense**	**Dense**	**Dense**
812963	320	9248	18496	36928	73856	147584	524352	2080	99
203475	160	2320	4640	9248	18496	36928	131104	528	51
50981	80	584	1168	2320	4640	9248	32784	136	27
33567	60	330	715	1534	3068	6110	21645	84	21
24041	50	230	506	1100	2200	4378	15499	60	18
17093	40	148	333	738	1558	3268	10953	40	15
12807	40	148	296	584	1168	2320	8200	36	15

Table 1. Number of parameters used in each layer. The first column indicates the total number of trainable parameters in the network and the following columns indicate the number of parameters per layer. The first row corresponds to the original network and the following rows correspond to $\frac{1}{4}$, $\frac{1}{16}$, $\frac{1}{24}$, $\frac{1}{32}$, $\frac{1}{48}$ and $\frac{1}{64}$ of the original number of parameters respectively.

Semantic autoencoders act as ideal denoising autoencoders, in the sense that they reconstruct the input image as a function of the features we consider most relevant. As such, their efficacy fully depends on the model used as decoder. This "Bayesian" twist on the standard autoencoder can be exploited for model testing, assuming that the encoder is sufficiently complex to learn to estimate the model's parameters.

The multi-column architecture presented here is the simplest architecture but many others are possible depending on the particular problem. For instance, for the problem at hand one could have a common network dedicated to extracting the simplest image features (assuming that all

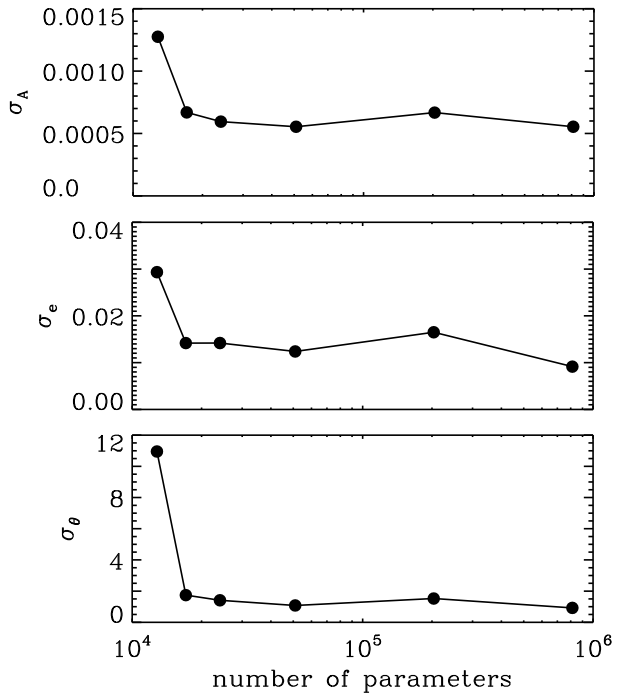

Figure 5. Parameter mean errors as a function of network size for 6 different ratios (see Table 1). The top, middle and bottom plots correspond to $r\sigma_A, \sigma_e$ and σ_θ respectively. The network used in the main analysis contains 812963 parameters and is corresponds to the point at the right of the figures.

parameters may be computed from such features) and then several specialized branches.

Other applications of Semantic Autoencoders, besides self-supervised parameter fitting, include solving inverse-PDE problems (Pakravan, et al. 2020) and self-supervised reconstruction of the initial conditions of complex systems. In a future work we will present an application of an autoencoder for the reconstruction of initial density fields from the observed galaxy distribution. The decoder in this case will be a (fast) N-body code and gradients will be numerically computed for each element in $\hat{\boldsymbol{p}}$ during training.

In general, problems that can be solved with Semantic Autoencoders can also be solved with standard recursive optimization methods. What the proposed architecture offers is a model that can solve, in an unsupervised way, for all the possible cases (within the region in feature space covered by the training sample) in a fast and efficient way. A machine learning approach trades network complexity and long training time for robustness and speed at prediction time.

The work presented here highlights the possibility of constructing of more general neural networks that combine standard artificial neurons with general algorithms or even physical systems. The decoder function G could be any process that is differentiable including analytic functions, numerical algorithms (Li T.-M. et al. 2018) and even real-world physical systems.

While this work was being independently developed google released the TENSORFLOW GRAPHICS deep learning framework (Valentin, J., et al. 2019) which is an implementation of the basic architecture presented here applied to computer graphics.

5 ACKNOWLEDGMENTS

The author would like to thank Mark Neyrinck for useful comments on the manuscript. This work was funded in part by "Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica" grant DGAPA-PAPIIT IA104818.

REFERENCES

Abadi, M. et al., 2015, TensorFlow: Large-scale machine learning on heterogeneous systems, tensorflow.org.
Barden M., Häußler B., Peng C. Y., McIntosh D. H., Guo Y., 2012, MNRAS, 422, 449
Bischof C. H., 1996, IUTAM Symposium on Optimization of Mechanical Systems, Springer Netherlands, 41-48.
Bourlard H., Kamp Y., 1988, Biol. Cybern., 59, 291
Chollet F. et al., 2015, https://github.com/fchollet/keras
Cybenko G., 1989, MCSS, 2, 303
Dieleman S., Willett K. W., Dambre J., 2015, MNRAS, 450, 1441
Hinton G., Salakhutdinov R., 2006, Science (New York, N.Y.), 313, 504
Domínguez Sánchez H., Huertas-Company M., Bernardi M., Tuccillo D., Fischer J. L., 2018, MNRAS, 476, 3661
Iten R., Metger T., Wilming H., del Rio L., Renner R., 2018, arXiv e-prints, arXiv:1807.10300
Kim E. J., Brunner R. J., 2017, MNRAS, 464, 4463
LeCun Y., 1987, PhD thesis, Universite de Paris VI
LeCun Y., Bengio Y., Hilton G., 2015, Nature, 521,436
Legendre A. M., 1805, Nouvelles méthodes pour la détermination des orbites des cometes, Sur la Méthode des moindres quarrés, Paris: F. Didot.
Li T.-M., Aittala M., Durand F., Lehtinen J., 2018, ACM Trans. Graph. (Proc. SIGGRAPH Asia), 37, 222:1
Peng C. Y., Ho L. C., Impey C. D., Rix H.-W., 2002, A&A, 124, 266
Pakravan S., Mistani P. A., Aragon-Calvo M. A., Gibou F., 2020, arXiv, arXiv:2001.03608
Ribli D., Dobos L., Csabai I., 2019, MNRAS, 489, 4847
Sérsic, J. L. 1963, Boletin de la Asociacion Argentina de Astronomia La Plata Argentina, 6, 41
Simard L., Mendel J. T., Patton D. R., Ellison S. L., McConnachie A. W., 2011, ApJS, 196, 11
Soumagnac M. T., et al., 2015, MNRAS, 450, 666
Tuccillo D., Huertas-Company M., Decenci'ere E., Velasco-Forero S., Domınguez Sanchez H., Dimauro P., 2018, MNRAS, 475, 894
Valentin, J., Keskin, C., Pidlypenskyi, P., Makadia A., Sud, A., and Bouaziz, Sofien, 2019, TensorFlow Graphics. https://github.com/tensorflow/graphics.
Vincent P., Larochelle H., Lajoie I., Bengio Y., Manzagol P.-A., 2010, J. Mach. Learn. Res., 11, 3371
Wengert, R. E, 1964, Communications of the ACM., 7, 463-464
Zeiler M. D., 2012, arXiv, arXiv:1212.5701

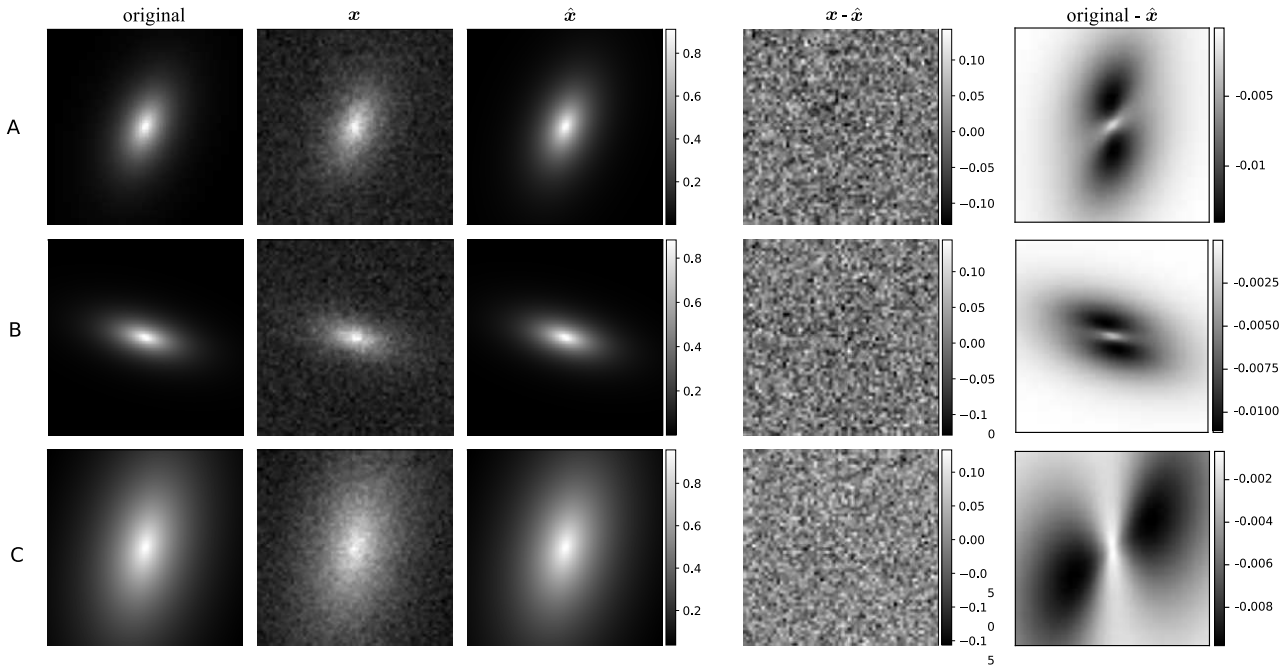

Figure 6. From left to right: (original) profile images, original images with added Gaussian noise with $\sigma = 0.05$ ($\boldsymbol{x}$), reconstructed output $\hat{\boldsymbol{x}}$, and residual images ($\boldsymbol{x} - \hat{\boldsymbol{x}}$) and (original $-\hat{\boldsymbol{x}}$) for three random combinations of the parameters A, e and θ, A, B and C rows respectively. The original images are shown here only for comparison. Both training and testing were done using the images with added Gaussian noise $\boldsymbol{x}$.

Bibliografía

[1] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, "An introduction to neural networks," 1993.

[2] M. A. Aragon-Calvo, "Classifying the large-scale structure of the universe with deep neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 484, no. 4, pp. 5771–5784, 2019.

[3] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[4] Y. N. Efremov and B. G. Elmegreen, "Hierarchical star formation from the time—space distribution of star clusters in the large magellanic cloud," *Monthly Notices of the Royal Astronomical Society*, vol. 299, no. 2, pp. 588–594, 1998.

[5] M. Khalil, M. Said, H. Osman, B. Ahmed, D. Ahmed, N. Younis, B. Maher, M. Osama, and M. Ashmawy, "Big data in astronomy: from evolution to revolution," *International Journal of Advanced Astronomy*, vol. 7, no. 1, pp. 11–14, 2019.

[6] D. Nelson, V. Springel, A. Pillepich, V. Rodriguez-Gomez, P. Torrey, S. Genel, M. Vogelsberger, R. Pakmor, F. Marinacci, R. Weinberger, *et al.*, "The illustristng simulations: public data release," *Computational Astrophysics and Cosmology*, vol. 6, no. 1, pp. 1–29, 2019.

[7] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[8] D. Steinkraus, I. Buck, and P. Simard, "Using gpus for machine learning algorithms," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pp. 1115–1120, IEEE, 2005.

[9] E. D. Feigelson and G. J. Babu, "Big data in astronomy," *Significance*, vol. 9, no. 4, pp. 22–25, 2012.

[10] Y. Bengio, Y. Lecun, and G. Hinton, "Deep learning for ai," *Communications of the ACM*, vol. 64, no. 7, pp. 58–65, 2021.

[11] A. Siemiginowska, G. Eadie, I. Czekala, E. Feigelson, E. Ford, V. Kashyap, M. Kuhn, T. Loredo, M. Ntampaka, A. Stevens, *et al.*, "Astro2020 science white paper: The next decade of astroinformatics and astrostatistics," *arXiv preprint arXiv:1903.06796*, 2019.

[12] E. Vasconcellos, R. De Carvalho, R. Gal, F. LaBarbera, H. Capelato, H. F. C. Velho, M. Trevisan, and R. Ruiz, "Decision tree classifiers for star/galaxy separation," *The Astronomical Journal*, vol. 141, no. 6, p. 189, 2011.

[13] A. Gauci, K. Z. Adami, and J. Abela, "Machine learning for galaxy morphology classification," *arXiv preprint arXiv:1005.0390*, 2010.

[14] P. Barchi, F. da Costa, R. Sautter, T. Moura, D. Stalder, R. Rosa, and R. de Carvalho, "Improving galaxy morphology with machine learning," *arXiv preprint arXiv:1705.06818*, 2017.

[15] X.-P. Zhu, J.-M. Dai, C.-J. Bian, Y. Chen, S. Chen, and C. Hu, "Galaxy morphology classification with deep convolutional neural networks," *Astrophysics and Space Science*, vol. 364, no. 4, pp. 1–15, 2019.

[16] T. Kuntzer, M. Tewes, and F. Courbin, "Stellar classification from single-band imaging using machine learning," *Astronomy & Astrophysics*, vol. 591, p. A54, 2016.

[17] T. A. Hinners, K. Tat, and R. Thorp, "Machine learning techniques for stellar light curve classification," *The Astronomical Journal*, vol. 156, no. 1, p. 7, 2018.

[18] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, *et al.*, "Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 389, no. 3, pp. 1179–1189, 2008.

[19] M. Banerji, O. Lahav, C. J. Lintott, F. B. Abdalla, K. Schawinski, S. P. Bamford, D. Andreescu, P. Murray, M. J. Raddick, A. Slosar, *et al.*, "Galaxy zoo: reproducing galaxy morphologies via machine learning," *Monthly Notices of the Royal Astronomical Society*, vol. 406, no. 1, pp. 342–353, 2010.

[20] M. Reza, "Galaxy morphology classification using automated machine learning," *Astronomy and Computing*, vol. 37, p. 100492, 2021.

[21] M. Munje, I. Daubar, G. Doran, K. Wagstaff, and L. Mandrake, "Large-scale automated detection of fresh impacts on mars using machine learning with ctx observations," *LPI Contributions*, vol. 2251, p. 2065, 2020.

[22] T. Stepinski, W. Ding, and R. Vilalta, "Detecting impact craters in planetary images using machine learning," in *Intelligent Data Analysis for Real-Life Applications: Theory and Practice*, pp. 146–159, IGI Global, 2012.

[23] L. Walkowicz, A. Howe, R. Nayar, E. Turner, J. Scargle, V. Meadows, and A. Zee, "Mining the kepler data using machine learning," in *American Astronomical Society Meeting Abstracts# 223*, vol. 223, pp. 146–04, 2014.

[24] D. J. Armstrong, J. Gamper, and T. Damoulas, "Exoplanet validation with machine learning: 50 new validated kepler planets," *Monthly Notices of the Royal Astronomical Society*, vol. 504, no. 4, pp. 5327–5344, 2021.

[25] G. Bass and K. Borne, "Supervised ensemble classification of kepler variable stars,"

Monthly Notices of the Royal Astronomical Society, vol. 459, no. 4, pp. 3721–3737, 2016.

[26] S. D. McCauliff, J. M. Jenkins, J. Catanzarite, C. J. Burke, J. L. Coughlin, J. D. Twicken, P. Tenenbaum, S. Seader, J. Li, and M. Cote, "Automatic classification of kepler planetary transit candidates," *The Astrophysical Journal*, vol. 806, no. 1, p. 6, 2015.

[27] K. Vida and R. M. Roettenbacher, "Finding flares in kepler data using machine-learning tools," *Astronomy & Astrophysics*, vol. 616, p. A163, 2018.

[28] I. Sadeh, F. B. Abdalla, and O. Lahav, "Annz2: photometric redshift and probability distribution function estimation using machine learning," *Publications of the Astronomical Society of the Pacific*, vol. 128, no. 968, p. 104502, 2016.

[29] N. M. Ball, R. J. Brunner, A. D. Myers, N. E. Strand, S. L. Alberts, D. Tcheng, and X. Llorà, "Robust machine learning applied to astronomical data sets. ii. quantifying photometric redshifts for quasars using instance-based learning," *The Astrophysical Journal*, vol. 663, no. 2, p. 774, 2007.

[30] S. Heinis, S. Kumar, S. Gezari, W. Burgett, K. Chambers, P. Draper, H. Flewelling, N. Kaiser, E. Magnier, N. Metcalfe, *et al.*, "Of genes and machines: application of a combination of machine learning tools to astronomy data sets," *The Astrophysical Journal*, vol. 821, no. 2, p. 86, 2016.

[31] M. Delli Veneri, S. Cavuoti, M. Brescia, G. Longo, and G. Riccio, "Star formation rates for photometric samples of galaxies using machine learning methods," *Monthly Notices of the Royal Astronomical Society*, vol. 486, no. 1, pp. 1377–1391, 2019.

[32] V. Bonjean, N. Aghanim, P. Salomé, A. Beelen, M. Douspis, and E. Soubrié, "Star for-

mation rates and stellar masses from machine learning," *Astronomy & Astrophysics*, vol. 622, p. A137, 2019.

[33] M. Huertas-Company, V. Rodriguez-Gomez, D. Nelson, A. Pillepich, C. Bottrell, M. Bernardi, H. Domínguez-Sánchez, S. Genel, R. Pakmor, G. F. Snyder, *et al.*, "The hubble sequence at z 0 in the illustristng simulation with deep learning," *Monthly Notices of the Royal Astronomical Society*, vol. 489, no. 2, pp. 1859–1879, 2019.

[34] L. Ferreira, C. J. Conselice, K. Duncan, T.-Y. Cheng, A. Griffiths, and A. Whitney, "Galaxy merger rates up to z 3 using a bayesian deep learning model: A major-merger classifier using illustristng simulation data," *The Astrophysical Journal*, vol. 895, no. 2, p. 115, 2020.

[35] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[36] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI global, 2010.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[39] S. Jia-hui, F. Li, Y. Han-qing, Z. Yan, Z. Xian, G. Wei-qun, L. Hui, and H. Yu, "Au-

tomatic coronagraph image classification with machine learning methods," *Chinese Astronomy and Astrophysics*, vol. 44, no. 4, pp. 507–518, 2020.

[40] J. A. Levy, "Using deep learning to detect galaxy mergers," 2019.

[41] D. Castillo, A. Shukla, and T. Wright, ""convolutional neural networks for galaxy morphology classification" machine learning (course 1dt071) uppsala university–spring 2018," 2018.

[42] K. P. Hall, "The micro-and macro-physics of giant molecular cloud formation," *Ph. D. Thesis*, no. 28718059, 2021.

[43] R. B. Larson, "The physics of star formation," *Reports on Progress in Physics*, vol. 66, no. 10, p. 1651, 2003.

[44] A. Goodman, P. Benson, G. Fuller, and P. Myers, "Dense cores in dark clouds. viii-velocity gradients," *The Astrophysical Journal*, vol. 406, pp. 528–547, 1993.

[45] C. D. Matzner and C. F. McKee, "Efficiencies of low-mass star and star cluster formation," *The Astrophysical Journal*, vol. 545, no. 1, p. 364, 2000.

[46] C. J. Lada and E. A. Lada, "Embedded clusters in molecular clouds," *Annual Review of Astronomy and Astrophysics*, vol. 41, no. 1, pp. 57–115, 2003.

[47] H. Gerola and P. E. Seiden, "Stochastic star formation and spiral structure of galaxies," *The Astrophysical Journal*, vol. 223, pp. 129–135, 1978.

[48] N. Murray, E. Quataert, and T. A. Thompson, "The disruption of giant molecular clouds by radiation pressure & the efficiency of star formation in galaxies," *The Astrophysical Journal*, vol. 709, no. 1, p. 191, 2009.

[49] R. C. Kennicutt Jr and N. J. Evans, "Star formation in the milky way and nearby galaxies," *Annual Review of Astronomy and Astrophysics*, vol. 50, pp. 531–608, 2012.

[50] F. Elias, E. J. Alfaro, and J. Cabrera-Caño, "Hierarchical star formation: stars and stellar clusters in the gould belt," *Monthly Notices of the Royal Astronomical Society*, vol. 397, no. 1, pp. 2–13, 2009.

[51] B. G. Elmegreen and Y. N. Efremov, "An extension of hierarchical star formation to galactic scales," *The Astrophysical Journal*, vol. 466, p. 802, 1996.

[52] P. Solomon, A. Rivolo, J. Barrett, and A. Yahil, "Mass, luminosity, and line width relations of galactic molecular clouds," *The Astrophysical Journal*, vol. 319, pp. 730–741, 1987.

[53] C. F. McKee and E. C. Ostriker, "Theory of star formation," *Annu. Rev. Astron. Astrophys.*, vol. 45, pp. 565–687, 2007.

[54] T. J. Moore, J. Urquhart, L. Morgan, and M. Thompson, "The effect of spiral arms on star formation in the galaxy," *Monthly Notices of the Royal Astronomical Society*, vol. 426, no. 1, pp. 701–707, 2012.

[55] D. Eden, T. Moore, J. Urquhart, D. Elia, R. Plume, A. Rigby, and M. Thompson, "Star formation scales and efficiency in galactic spiral arms," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 1, pp. 289–300, 2015.

[56] S. Ragan, T. Moore, D. Eden, M. Hoare, J. Urquhart, D. Elia, and S. Molinari, "The role of spiral arms in milky way star formation," *Monthly Notices of the Royal Astronomical Society*, vol. 479, no. 2, pp. 2361–2373, 2018.

[57] M. R. Krumholz and C. Federrath, "The role of magnetic fields in setting the star formation rate and the initial mass function," *Frontiers in Astronomy and Space Sciences*, vol. 6, p. 7, 2019.

[58] Y. Gao and P. M. Solomon, "The star formation rate and dense molecular gas in galaxies," *The Astrophysical Journal*, vol. 606, no. 1, p. 271, 2004.

[59] A. Dekel, Y. Birnboim, G. Engel, J. Freundlich, T. Goerdt, M. Mumcuoglu, E. Neistein, C. Pichon, R. Teyssier, and E. Zinger, "Cold streams in early massive hot haloes as the main mode of galaxy formation," *Nature*, vol. 457, no. 7228, pp. 451–454, 2009.

[60] N. Bastian, M. Gieles, Y. N. Efremov, and H. Lamers, "Hierarchical star formation in m 51: star/cluster complexes," *Astronomy & Astrophysics*, vol. 443, no. 1, pp. 79–90, 2005.

[61] N. Bastian, B. Ercolano, M. Gieles, E. Rosolowsky, R. Scheepmaker, R. Gutermuth, and Y. Efremov, "Hierarchical star formation in m33: fundamental properties of the star-forming regions," *Monthly Notices of the Royal Astronomical Society*, vol. 379, no. 4, pp. 1302–1312, 2007.

[62] J. Sánchez Almeida, B. G. Elmegreen, C. Munoz-Tunón, and D. M. Elmegreen, "Star formation sustained by gas accretion," *The Astronomy and Astrophysics Review*, vol. 22, no. 1, pp. 1–60, 2014.

[63] F. Marinacci, M. Vogelsberger, R. Pakmor, P. Torrey, V. Springel, L. Hernquist, D. Nelson, R. Weinberger, A. Pillepich, J. Naiman, *et al.*, "First results from the illustristng simulations: radio haloes and magnetic fields," *Monthly Notices of the Royal Astronomical Society*, vol. 480, no. 4, pp. 5113–5139, 2018.

[64] V. Springel, R. Pakmor, A. Pillepich, R. Weinberger, D. Nelson, L. Hernquist, M. Vogelsberger, S. Genel, P. Torrey, F. Marinacci, *et al.*, "First results from the illustristng simulations: matter and galaxy clustering," *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 1, pp. 676–698, 2018.

[65] J. P. Naiman, A. Pillepich, V. Springel, E. Ramirez-Ruiz, P. Torrey, M. Vogelsberger, R. Pakmor, D. Nelson, F. Marinacci, L. Hernquist, *et al.*, "First results from the illustristng simulations: A tale of two elements–chemical evolution of magnesium

and europium," *Monthly Notices of the Royal Astronomical Society*, vol. 477, no. 1, pp. 1206–1224, 2018.

[66] A. Pillepich, D. Nelson, L. Hernquist, V. Springel, R. Pakmor, P. Torrey, R. Weinberger, S. Genel, J. P. Naiman, F. Marinacci, *et al.*, "First results from the illustristng simulations: the stellar mass content of groups and clusters of galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 1, pp. 648–675, 2018.

[67] D. Nelson, A. Pillepich, V. Springel, R. Weinberger, L. Hernquist, R. Pakmor, S. Genel, P. Torrey, M. Vogelsberger, G. Kauffmann, *et al.*, "First results from the illustristng simulations: the galaxy colour bimodality," *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 1, pp. 624–647, 2018.

[68] A. Pillepich, D. Nelson, V. Springel, R. Pakmor, P. Torrey, R. Weinberger, M. Vogelsberger, F. Marinacci, S. Genel, A. van der Wel, *et al.*, "First results from the tng50 simulation: the evolution of stellar and gaseous discs across cosmic time," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 3, pp. 3196–3233, 2019.

[69] D. Nelson, A. Pillepich, V. Springel, R. Pakmor, R. Weinberger, S. Genel, P. Torrey, M. Vogelsberger, F. Marinacci, and L. Hernquist, "First results from the tng50 simulation: galactic outflows driven by supernovae and black hole feedback," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 3, pp. 3234–3261, 2019.

[70] V. Springel and L. Hernquist, "Cosmological smoothed particle hydrodynamics simulations: a hybrid multiphase model for star formation," *Monthly Notices of the Royal Astronomical Society*, vol. 339, no. 2, pp. 289–311, 2003.

[71] R. C. Kennicutt Jr, "The star formation law in galactic disks," *The Astrophysical Journal*, vol. 344, pp. 685–703, 1989.

[72] R. Davé, "The galaxy stellar mass–star formation rate relation: evidence for an evolving stellar initial mass function?," *Monthly Notices of the Royal Astronomical Society*, vol. 385, no. 1, pp. 147–160, 2008.

[73] A. K. Weigel, K. Schawinski, and C. Bruderer, "Stellar mass functions: methods, systematics and results for the local universe," *Monthly Notices of the Royal Astronomical Society*, vol. 459, no. 2, pp. 2150–2187, 2016.

[74] P. Di Matteo, F. Combes, A.-L. Melchior, and B. Semelin, "Star formation efficiency in galaxy interactions and mergers: a statistical study," *Astronomy & Astrophysics*, vol. 468, no. 1, pp. 61–81, 2007.

[75] D. F. Woods and M. J. Geller, "Minor galaxy interactions: Star formation rates and galaxy properties," *The Astronomical Journal*, vol. 134, no. 2, p. 527, 2007.

[76] R. Lupton, M. R. Blanton, G. Fekete, D. W. Hogg, W. O'Mullane, A. Szalay, and N. Wherry, "Preparing red-green-blue images from ccd data," *Publications of the Astronomical Society of the Pacific*, vol. 116, no. 816, p. 133, 2004.

[77] R. Dian, S. Li, A. Guo, and L. Fang, "Deep hyperspectral image sharpening," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5345–5355, 2018.

[78] J. Yang, X. Fu, Y. Hu, Y. Huang, X. Ding, and J. Paisley, "Pannet: A deep network architecture for pan-sharpening," in *Proceedings of the IEEE international conference on computer vision*, pp. 5449–5457, 2017.