Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Clasificación morfológica de galaxias usando redes neuronales convolucionales

TESIS

Que para optar por el grado de:

Maestro en Ciencia e Ingeniería de la Computación

Presenta:

Gabriel Efraín Condés Luna

Director de Tesis:
Dr. Gibran Fuentes Pineda
IIMAS, UNAM

Ciudad Universitaria, Cd. Mx. Febrero 2021

Agradecimientos

A Brenda Saavedra, que ha sido mi compañera incondicional a lo largo de este maravilloso viaje, y junto a quien he aprendido a disfrutar la vida.

A mi madre y mi hermana, que me han dado su apoyo y ayuda siempre y con las que formo una familia fuerte y llena de amor.

A Gibran Fuentes, a quien admiro y quién ha sido mi guía a través de este camino de aprendizaje. Le agradezco porque siempre me he sentido apoyado y cobijado por él, más en los momentos dificiles. Muchas gracias por todo.

Índice general

Capítulo 1

Introducción

1.1. Morfología de galaxias

En este trabajo de tesis exploramos la posibilidad de emplear técnicas de aprendizaje autosupervisado en redes neuronales para mejorar la clasificación de imágenes de galaxias según su morfología. Además de otras opciones para mejorar el rendimiento sobre dicha clasificación.

Puesto que esta es una tesis para un posgrado en ciencia e ingeniería de la computación, ocuparemos un espacio considerable en este cápitulo para explicar y describir el problema de clasificación de galaxias según su morfología desde el punto de vista de la astronomía.

La raza humana desde sus orígenes ha mostrado curiosidad por los objetos que se revelan en el cielo nocturno. Los esfuerzos por dar explicación y encontrar patrones en los objetos celestes, de la mano con el desarrollo de la ciencia, han evolucionado hasta la actualidad en proyectos de investigación en los cuales se crean catálogos digitales del cielo. Estos datos consisten en sondear el cielo registrando señales que llegan desde el espacio exterior (principalmente radiación, o sea luz), siendo uno de los proyectos de sondeo más recientes e importantes el *Sloan Digital Sky Survey*, al cual nos referiremos de ahora en adelante como SDSS. Las señales recopiladas en estos proyectos pueden provenir de objetos tales como el sol, estrellas, galaxias, planetas fuera de nuestro sistema solar, el medio interestelar, el fondo cósmico de microondas, etc., los cuales son todos objetos de

estudio de distintas ramas de la astronomía y astrofísica.

Dada la imposibilidad de recrear fenómenos astronómicos en laboratorios, tales como la dinámica de galaxias, creación de agujeros negros, etc., la principal fuente de información sobre la naturaleza real del espacio exterior proviene de estos sondeos. En la actualidad, la astrofísica, usa estos datos, no solo para llevar registros y con base en ellos hacer predicciones, como lo hacían varias civilizaciones antiguas, sino también para tratar de describir los fenómenos astronómicos usando las leyes de la física como principios básicos.

Proyectos como el SDSS, no solo necesitan la contribución de los astrónomos y astrofísicos, tareas como el diseño y construcción de los telescopios, el procesamiento de las señales que reciben los telescopios y el análisis de los datos que se recolectan ha provocado que gente afín a las ingenierías y ciencias de la computación colaboren fuertemente en estos proyectos.

El gran auge que ha tenido el aprendizaje máquina en los últimos años ha provocado que las aplicaciones de esta subárea de la inteligencia artificial permeen a otras áreas de la ciencia. Al ser los datos el principal motor de los algoritmos de aprendizaje máquina, los proyectos, como el SDSS, en los cuales se sondea el cielo, representan una gran oportunidad para la aplicación y desarrollo de algoritmos de aprendizaje máquina, debido a que, conforme la tecnología avanza, la cantidad de datos y medidas que obtienen estos sondeos crece de forma vertiginosa y cada vez son más necesarias herramientas para automatizar su procesamiento y análisis.

Uno de los problemas más susceptibles de ser abordado a través del aprendizaje máquina, y de forma más específica desde el aprendizaje profundo, es el de la clasificación de galaxias. Las galaxias se presentan en una gran variedad de formas, o en otras palabras morfologías. Estas morfologías son tales que clasificaciones con base en éstas surgen de forma muy natural. Además, el marco de trabajo, que deriva de una clasificación de galaxias, es tal que facilita e incluso sugiere, en muchos casos, estudios e investigaciones en torno a éstas. Este hecho es de gran relevancia, puesto que en la actualidad siguen existiendo muchas interrogantes sobre el significado físico de la morfología de las galaxias, por lo tanto, el contar con un esquema que permita estudiar a las galaxias de una forma relativamente ordenada, supone una herramienta muy poderosa.

Sin embargo, la clasificación de galaxias sufre de un conjunto de complicaciones muy particulares, problemas que no se presentan en otro tipo de problemas de clasificación en la ciencias como el de clasificar a las especies de animales, a continuación mencionaremos

algunas de ellas. Como ya mencionamos anteriormente, dado que los seres humanos aún no somos capaces de reproducir fenómenos en un laboratorio como la formación y evolución de galaxias, no podemos apreciar a las galaxias desde todos sus ángulos. Esto supone un problema porque las galaxias tienen planos de simetría predilectos. Por ejemplo, en el caso de galaxias que presenten la forma de un disco, su plano de simetría predilecto será el plano en el cual está contenido el disco. Estos planos presentan una orientación aleatoria sobre todas las galaxias existentes en el universo, y dado que, los telescopios están situados en un punto fijo del universo (la tierra), las imágenes que podamos conseguir están limitadas a capturar solo un plano en especifico. Además, las galaxias están distribuidas a lo largo de un rango muy amplio de distancias, por lo que existirán galaxias que podamos ver con más dificultad que otras [1].

Otra situación que puede afectar la apariencia de las galaxias, y por lo tanto complicar la tarea de clasificación, es que el aspecto de una galaxia depende de la longitud de onda en la cual se le vea; las imágenes y fotografías que se toman de galaxias solo capturan una parte del espectro de luz que emiten. Por ejemplo, la mayoría de la luz emitida por las galaxias proviene de las estrellas que en esta residen; las estrellas más jóvenes tienden a tener temperaturas muy altas, lo que deriva en que éstas emiten la mayoría de su luz en el dominio ultravioleta del espectro (longitudes de onda cortas), mostrando colores azulados, mientras que las estrellas viejas, teniendo menores temperaturas, emiten la mayoría de su luz en el dominio infrarrojo (longitudes de onda largas), teniendo colores rojizos. Por lo tanto, las imágenes de las galaxias capturadas en longitudes de onda cortas, tienden a enfatizar a las estrellas más jóvenes y calientes de las galaxias. Por el otro lado, cuando las imágenes se crean capturando longitudes de onda más largas, se resaltan las estrellas más viejas. Esto tiene una consecuencia directa en la apariencia que revelan las imágenes de las galaxias, puesto que las estrellas viejas tienden a estar distribuidas de forma más suave y uniforme que las estrellas jóvenes, si se ve a las galaxias usando filtros azules (que le dan preferencia a las longitudes de onda cortas) la forma de la galaxia será más irregular [1].

La expansión del universo también puede afectar la apariencia de las galaxias. Debido a la constante expansión del universo, el espectro de luz emitido por las galaxias sufre un fenómeno conocido como corrimiento al rojo (muy parecido a lo que ocurre con el sonido en el efecto Doppler). Este efecto consiste en que los espectros de luz sufren un desplazamiento a longitudes de ondas más largas (hacia el color rojo). Esto puede provocar que la apariencia que es proporcionada por su luz en longitudes de onda cortas (colores azules), solo sea visible para nosotros en longitudes de onda largas (colores

rojizos). Esto provoca que sea difícil comparar las imágenes de galaxias que sufren un alto corrimiento al rojo con las de galaxias que sufren un bajo corrimiento al rojo [1].

Además de lo anterior, otro factor que debe ser tomado en cuenta al tratar de esquematizar la galaxias por su forma, es que la morfología de una galaxia no es constante en el tiempo, por el contrario, ésta evoluciona y es afectada por distintos agentes, externos o internos a ella. Algunos de estos podrían ser tales como la constante muerte y nacimiento de estrellas, la rotación de las galaxias, la interacción entre galaxias contiguas, etc. Todos estos agentes conforman un ambiente que está determinando la morfología de las galaxias y, debido a que la velocidad de la luz no es infinita, cuando vemos galaxias muy lejanas, estamos viendo cómo eran esas galaxias en tiempos en los cuales el universo era mucho más joven de lo que es ahora. Por lo tanto el ambiente que está determinando la morfología de esas galaxias, está conformado por condiciones particulares que existieron en esa edad del universo [1].

Daremos ahora una pequeña revisión de la historia de la clasificación de galaxias. De 1781 a 1847, Sir William Herschel y su hijo John, identificaron algunos objetos en el cielo como *nebulosas blancas*; estos objetos más tarde resultaron ser galaxias cercanas. Fue hasta que en 1845 que William Parsons que acuñó el término de espiral a los objetos nombrados, anteriormente por los Herschel, como *nebulosas blancas*. Sin embargo, hasta este momento todas las observaciones se habían hecho de forma visual, fue hasta que se empezaron a usar fotos de galaxias, obtenidas usando telescopios con platos con emulsiones que podían captar luz en la región azul, que se pudo identificar las principales clases de galaxias [1].

En el siglo XX, Edwin Hubble propuso el esquema de clasificación que, con ligeras modificaciones, se usa hoy en día. En la fig.1.1 podemos ver un esquema de la clasificación de Hubble. En un inicio, se hizo la distinción entre tres clases, elípticas, espirales e irregulares. Debido a que el nombre de cada clase sugiere mucho de la apariencia de las galaxias, el lector podría hacerse una idea de cómo se ven las galaxias de cada clase. Por ejemplo, las primeras son entes suaves con forma elíptica, las segundas tienen forma de discos aplanados que tienen artefactos en forma de brazos y las últimas son galaxias que no presentan una forma clara [1]. Cada una de estas tres clases pueden identificarse en la fig.1.1, donde las galaxias que pertenecen a la clase de elípticas, espirales e irregulares están indicadas con las letras E, S e I, respectivamente, al principio de su clasificación.

Posteriormente se fueron haciendo refinamientos de cada una de estas tres clases. A las galaxias elípticas se les agrego un número n después de la E, el cual es conocido como

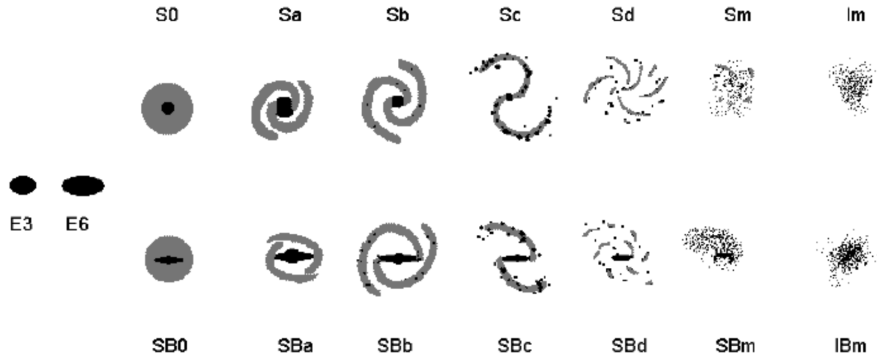

Figura 1.1: Esquema de la clasificación de Hubble, también conocido como secuencia de Hubble (imagen tomada de [1]).

índice de elípticidad e indica qué tan redonda o alargada es cada galaxia, entre mayor sea n más alargada es la galaxia. Las galaxias espirales pueden verse como la composición de varios elementos, un disco el cual contiene los brazos de la galaxia, un bulto esférico en el centro del disco y, en algunos casos, también un artefacto en forma de barra en el centro de la galaxia. Las galaxias que sí poseen la estructura de barra son indicadas poniendo la letra B después de la S. Además de lo anterior, la letra minúscula que aparece hasta el final de las clases, que son espirales, en la fig.1.1 es un indicador de qué tan enroscados están los brazos de la galaxia [1].

Las clases Sm y SBm, son galaxias que parecen ser irregulares, sin embargo en 1950 G. de Vaucouleurs, demostró que un par de galaxias, que se veían al sur del cielo y que se tenían contempladas como galaxias irregulares, presentaban una estructura espiral; a este tipo de galaxias se les conoce también como Nubes de Magallanes, en referencia a las dos galaxias ya mencionadas. Las clases Im y IBm se refieren a las galaxias que se parecen a las Nubes de Magallanes, pero no presentan ningún tipo de estructura espiral.

En 1936, Hubble hizo una modificación a su esquema de clasificación, agregando la clase S0, la cual representa a las galaxias que tienen una estructura de disco, pero que no poseen brazos. La idea detrás de esta modificación es la de simbolizar la transición entre las galaxias elípticas y las espirales.

Uno hecho bien establecido es que la morfología de las galaxias está íntimamente relacionado con la dinámica del universo, por lo tanto el entender la morfología de éstas, aunque sea a través de una clasificación sistemática, puede derivar en información que

dé luz a problemas de gran importancia en la astronomía como la evolución del universo. Sin embargo, y como mencionamos anteriormente, el procesar los datos provenientes de los censos del cielo es un problema, entre ellos las imágenes de galaxias, las cuales no pueden ser clasificadas por astrónomos expertos en su totalidad, debido a la capacidad de los censos del cielos de recabar cada vez más y más datos, se ha pasado de recabar cientos de imágenes de galaxias a cientos de miles e incluso millones.

Aunque el escenario, en el cual astrónomos clasifiquen todas las imágenes de galaxias con las que se cuenta al día de hoy, esté descartado por su inviabilidad, eso no quiere decir que no se hayan realizado esfuerzos para clasificar algunos subconjuntos de todas las galaxias disponibles. Por ejemplo, el proyecto conocido como *Galaxy Zoo 1* [2], en el cual se reclutaron a voluntarios (no necesariamente con una formación científica) para clasificar imágenes de galaxias usando una interfaz web amigable con el usuario, que los iba guiando a través de un árbol de decision al responder varias preguntas. *Galaxy Zoo 1* [2] clasificaba galaxias en tres clases. Posteriormente, debido al éxito de *Galaxy Zoo 1* [2], se realizó una reedición llamada *Galaxy Zoo 2* [2], que seguía la misma dinámica pero tratando de clasificar a las galaxias en una más grande variedad de clases. Sin embargo, este método presenta una debilidad bastante clara, los *astrónomos aficionados*, tienden a elegir opciones intermedias para ciertas características; por ejemplo, los voluntarios solo notifican la presencia de la estructura de barra cuando ésta se presenta de forma muy clara. Por otra parte, muchas galaxias terminan presentando un gran margen de desacuerdo en las clasificaciones hechas por los voluntarios.

Otro esfuerzo importante fue el hecho por Nair [3], quien en 2010 presentó un catálogo de alrededor de 13,000 galaxias clasificadas por la astrónoma Preethi B. Nair. Este catálogo es de gran importancia para este trabajo de tesis, puesto que es el que se usa para entrenar lo modelos que aquí presentaremos.

Sin embargo, las bases de datos con las que se cuentan para poder implementar modelos de aprendizaje máquina, en especial de aprendizaje supervisado, padecen de ciertos defectos que pueden afectar de sobremanera a estos modelos. Por un lado, puede pasar que las etiquetas de los datos no sean de buena calidad, como en el caso del *Galaxy Zoo 1* [2] y *Galaxy Zoo 2* [2]. Por otro lado, que las etiquetas sean de mejor calidad pero que la cantidad de datos sea mucho menor, como en el caso de la base de datos de Nair [3]. Además, puesto que la distribución de galaxias no es uniforme, el que las clases presenten un gran desbalance en estas bases de datos, también es un problema grande que perjudica el rendimiento de los modelos de inteligencia artificial.

En el caso de los datos obtenidos por los censos del cielo, siempre existen muchos más datos no etiquetados que etiquetados. En la actualidad, dentro del área de aprendizaje máquina y aprendizaje profundo, se realizan grandes esfuerzos para lograr que modelos de redes neuronales aprendan a extraer características visuales de alto nivel usando datos no etiquetados; estas técnicas son propias del aprendizaje no supervisado. En este trabajo exploramos la posibilidad de contrarrestar el problema del desbalance de clases y la escasez de datos, usando una técnica conocida como aprendizaje autosupervisado, el cual es una técnica de aprendizaje no supervisado, que ha ganado fuerza en los últimos años. Además, presentamos otro tipo de aproximaciones para tratar estos problemas usando solo aprendizaje supervisado. Habiendo contextualizado más el problema que estamos tratando, procederemos a definir de forma concreta los siguiente puntos.

1.2. Objetivos

El objetivo general de esta investigación es estudiar la posibilidad de disminuir el efecto que tiene la poca cantidad de datos etiquetados y el desbalanceo de clases en la tarea supervisada de clasificación de galaxias según su morfología, a través de técnicas de aprendizaje autosupervisado. También se estudia la posibilidad de emplear otro tipo de técnicas usando solo un esquema aprendizaje supervisado.

Los objetivos específicos de este trabajo de investigación son los siguientes:

- Entrenar un modelo de control para la clasificación multiclase de galaxias solo de forma supervisada. Este modelo servirá para tener un punto de comparación con los demás modelos.
- Implementar las técnicas de aprendizaje autosupervisado conocidas como *jigsaw* y *relative patch location* para extraer representaciones visuales de las imágenes no etiquetadas de galaxias.
- Realizar transferencia de conocimiento de estas dos tareas para la realización de la tarea de clasificación multiclase de galaxias de forma supervisada.
- Entrenar un modelo solo de forma supervisada para la realización de la tarea de clasificación multiclase de galaxias, añadiendo factores que pueden mejorar el rendimiento respecto al modelo de control, tales como mejorar la arquitectura, realizar la técnica de *up-sampling* para el balancear de forma artificial las clases en la base de datos etiquetados e implementar un *learning rate schedule*.

- Estudiar y comparar los efectos de todas les técnicas antes mencionadas sobre el rendimiento de la tarea supervisada de clasificación multiclase de galaxias según su morfología.

1.3. Hipótesis

A través de técnicas de aprendizaje autosupervisado, se puede aprovechar la gran cantidad de imágenes de galaxias no clasificadas, para que los modelos aprendan a extraer características útiles para la realización de la tarea supervisada de clasificación de las galaxias etiquetadas.

1.4. Planteamiento del problema

Como ya hemos mencionado, el problema de clasificación de galaxias conlleva una serie de complicaciones, algunas de ellas son las siguientes:

- Etiquetar datos, en el caso de imágenes de galaxias, es un proceso especialmente costoso, puesto que dicha clasificación solo puede ser hecha por astrónomos especializados en esto. Es por esto que la disponibilidad de imágenes etiquetadas es una gran restricción en este problema en específico.
- El desbalance entre clases es un gran problema que padecen todas las bases de datos de imágenes de galaxias, resultando más afectadas las clases que se encuentran al final de la secuencia de Hubble (fig. 1.1).
- A diferencia de otros problemas de clasificación, la transición entre clases es continua. Por lo tanto, el clasificación de galaxias cuya morfología esté cercana a los puntos de transición entre clases puede ser una tarea ambigua, este factor agrega dificultad a la tarea de clasificación de galaxias.
- Debido a la naturaleza tan particular de las imágenes de galaxias (son imágenes de objetos que se encuentran a millones de años luz), éstas son afectadas por varios factores que ya mencionamos anteriormente, tales como las siguientes:
    - Puesto que solo podemos ver a las galaxias desde un plano específico, no podemos elegir verlas dede su plano predilecto de simetría. Las galaxias que

solo podemos observar de canto, son las que más incertidumbre tienen al momento de ser clasificadas.

- Al estar, las galaxias distribuidas a lo largo de un amplio rango de distancias, existirán galaxias que podamos ver con más dificultad que otras, esto repercute de gran forma en la calidad de las imágenes de muchas galaxias.
- El rango del espectro de luz a través del que se decida ver a las galaxias también es importante; dependiendo de su tipo, las galaxias pueden tener predilección por emitir luz en una zona específica del espectro.
- El aspecto de las galaxias puede ser afectado por el fenómeno físico conocido como corrimiento al rojo. Aunque en la mayoría de los casos éste efecto es despreciable.
- La imagen de una galaxia puede ser contaminada por la luz de objetos cercanos.

Éstos, entre otros, son el tipo de factores que pueden afectar la calidad de las imágenes que se usan para entrenar los modelos.

1.5. Contribuciones

- Como veremos en capítulos posteriores, el problema de clasificación multiclase no se ha tratado de forma exhaustiva, si bien existen trabajos en los cuales se trata, no son lo suficientemente explícitos en los detalles de los resultados, concentrándose más en discutir a fondo solo el problema de clasificación en dos clases. En este trabajo, discutimos y presentamos de la forma más explícita posible los resultados que se obtienen de realizar la clasificación multiclase de galaxias.
- El estudiar la posibilidad de implementar técnicas del estado del arte del aprendizaje profundo al área de la astronomía.
- Estudiar la posibilidad de encontrar un esquema de entrenamiento de modelos de aprendizaje no supervisado, que aproveche la gran cantidad de imágenes de galaxias no etiquetadas que existe.
- Sentar un antecedente sólido para la colaboración entre investigadores del IIMAS y el Instituto de Astronomía de la UNAM para la implementación de técnicas de aprendizaje profundo usando datos astronómicos.

1.6. Organización de la tesis

En este apartado daremos una breve descripción del contenido de los capítulos subsecuentes. En el capítulo 2 detallamos el estado del arte, tanto en la aplicación de técnicas de aprendizaje profundo y aprendizaje máquina para la clasificación de imágenes de galaxias, como en las técnicas de aprendizaje autosupervisado utilizadas para extraer representaciones visuales.

En el capítulo 3 hacemos una pequeña revisión de las técnicas de aprendizaje profundo más usuales para el procesamiento de imágenes. Partimos desde el perceptrón multicapa, después hablamos de las redes neuronales convolucionales y finalizamos con una descripción breve de las arquitecturas usadas para los modelos. Posteriormente hacemos una introducción al aprendizaje autosupervisado y a las técnicas específicas empleadas en este trabajo.

En el capítulo 4 hacemos explícito todos los detalles de los experimentos realizados en este trabajo, tanto las arquitecturas como esquemas de entrenamiento, tanto de los experimentos supervisados como de los autosupervisados.

En el capítulo 5 presentamos los resultados de los experimentos hechos, empezando por el experimento de control, el rendimiento de los modelos encargados de realizar las tareas de aprendizaje autosupervisado y los resultados de hacer transferencia de conocimiento de esas tareas a la tarea de clasificación multiclase de galaxias. Finalizamos presentando los resultados obtenidos de mejorar el modelo que resuelve solo de manera supervisada la tarea de clasificación.

Para finalizar, en el capítulo 6 recapitulamos la discusión hecha entorno a los resultados obtenidos y exponemos los posibles trabajos que podrían derivar del presente trabajo de investigación.

Capítulo 2

Estado del arte

Dado que el trabajo de investigación presentado en esta tesis se realizó en la intersección de dos áreas científicas diferentes, inteligencia artificial y astronomía, en esta sección discutiremos el estado del arte desde las perspectiva de ambas áreas por separado. Dedicaremos un apartado en el cual discutiremos los avances recientes en la aplicación de métodos de inteligencia artificial en la astronomía, en específico para la clasificación de galaxias según su morfología y en otro apartado se presentan los avances propios del área de aprendizaje profundo, los cuales sientan las bases para los experimentos que se realizaron en este trabajo de investigación.

2.1. Inteligencia artificial para la clasificación de imágenes de galaxias

Para hablar del estado del arte de la aplicación de métodos de inteligencia artificial para la clasificación de galaxias nos referiremos principalmente a al trabajo de Barchi [4]. En [4], Barchi hace un comparativo de diversas técnicas, tanto de aprendizaje profundo como de otros algoritmos de aprendizaje máquina *tradicionales*, implementadas para la clasificación de imágenes según su morfología.

La clasificación más sencilla que se le puede dar a las galaxias según su forma, fue propuesta por Hubble y divide a las galaxias en dos clases, galaxias espirales (también conocidas como galaxias tardías) y galaxias elípticas (también conocidas como galaxias

tempranas). El nombre de ambas clases alude a la apariencia de las galaxias (fig. 2.1), las galaxias espirales presentan estructuras en formas de brazos que le dan la forma de una espiral, mientras que las galaxias elípticas muestran una forma elipsoidal. Existen a la par formas más finas de clasificar a las galaxias, llegando a haber catálogos en los cuales se clasifican galaxias en quince clases diferentes [3].

Figura 2.1: Ejemplo de galaxia elíptica (izquierda) y de galaxia espiral (derecha) (imágenes tomadas de [5]).

En [4] se trata principalmente el problema de clasificar galaxias entre espirales y elípticas, para dicha tarea se usa la base de datos del *Galaxy Zoo 1* [2], también se estudia el problema de clasificar galaxias en múltiples clases, para lo cual se usa la base de datos del *Galaxy Zoo 2* [2]. Para atacar estos problemas se implementan algoritmos tanto de aprendizaje profundo como de aprendizaje máquina *tradicionales*. De forma más específica, por el lado del aprendizaje profundo, se reportan los resultados de implementar redes neuronales convolucionales, y por el lado del aprendizaje máquina *tradicional*, se emplean los algoritmos de árbol de decisión, máquinas de vectores de soporte y perceptrón multicapa. Discutamos primero cuál fue el preprocesamiento que se le dio a los datos para entrenar a los segundos.

De forma usual, los datos con los cuales se entrenan modelos de aprendizaje máquina *tradicionales*, como los anteriormente mencionados, están contenidos en una tabla, donde cada renglón de la tabla representa un elemento de la base de datos y este renglón debe contener tanto el campo a predecir como los atributos o variables a partir de los cuales se pretende hacer la predicción. Por ejemplo, se podría querer predecir la clase de una flor, a aprtir de los tamaños de sus hojas [6] o el precio de una casa en función de su tamaño

y materiales [7]. Por lo tanto, es natural que el lector se pregunte cómo es que se podría representar la imagen de una galaxia en forma de un renglón en una tabla. Una forma ingenua de proceder sería el aplanar la matriz de pixeles de la imagen, de tal forma que los valores de los pixeles terminen siendo formados en un único vector, ver fig. 2.2. Este procedimiento puede derivar en distintas complicaciones, una de ellas sería que el número de campos en nuestra base de datos termine siendo excesivamente grande, suponiendo que las imágenes del problema son a escala de grises y de un tamaño $n \times n$ entonces el tamaño de los renglones sería del orden $O(n^2)$, lo cual podría afectar el rendimiento de los modelos de forma desfavorable.

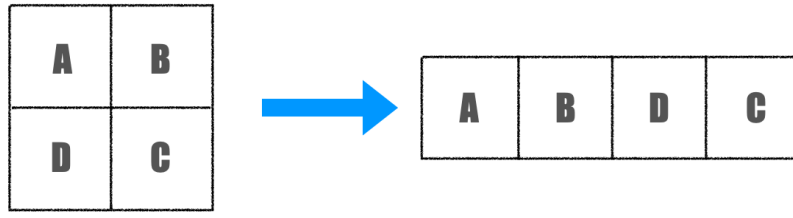

Figura 2.2: Transformación de una matriz a un vector.

Es por esta razón, que en [4] Barchi opta por extraer atributos de las imágenes usando técnicas propias de la astronomía. Estos atributos serán posteriormente utilizados como los campos para entrenar los modelos de aprendizaje máquina. El conjunto de atributos extraídos de las imágenes de galaxias es denominado CyMorph. A continuación, enlistaremos y describiremos brevemente a los atributos que conforman este marco de trabajo:

- Concentración (C): Se define como la cantidad $C = \log_{10}(R_1/R_2)$, donde R_i es el radio del círculo de la imagen que contiene cierta cantidad del flujo de luz de la imagen [8]. En [4] Barchi menciona que estos hiperparámetros fueron fijados en $C = \log_{10}(R_{75\,\%}/R_{35\,\%})$ para la optimización del rendimiento de los modelos.

- Asimetría (A): Esta es una cantidad definida como $A = 1 - s(I^0, I^\pi)$, donde I^0 es la imagen original, I^π es la imagen rotada π radianes y s es una función conocida como el rango de correlación de Spearman [9].

- Grumosidad (S por su inicial en ingles *smoothness*): Esta es una cantidad cuya intención es la de medir qué tanto cambia el gradiente sobre toda la imagen. Ésta

se calcula $A = 1 - s(I^0, I^s)$, donde I^s es el resultado de hacer pasar la imagen original por un proceso de suavizado.

- Análisis del patrón del gradiente (GPA): Es una cantidad que permite estudiar propiedades locales del gradiente de la imagen en un conjunto de puntos [10].

- La entropía de información (H): Es una medida de la distribución de los valores de los pixeles en la imagen [11].

Es importante aclarar que no se pretende dar aquí una revisión exhaustiva de este marco de trabajo; sí el lector quisiera ahondar más en este tema puede referirse al trabajo de Sautter [12].

El cálculo de estas cantidades depende de varios hiperparámetros, de aquí surge el problema de afinar dichos hiperparámetros de tal forma que las cantidades del sistema CyMorph sean las óptimas para resolver el problema de clasificación. La idea que los autores usa Barchi [4] para atacar este problema es el siguiente. Entre mejores sean las variables definidas, éstas deberían ayudar a distinguir mejor entre las galaxias espirales y las elípticas. Para medir qué tan bien separa cada variable estos dos conjuntos se consideran las distribuciones de cada clase a lo largo de cada una de las variables. Para medir la separación entre ambas distribuciones se usa la métrica δ_{GHS}, conocida en inglés como *Geometric Histogram Separation* [10, 13]. Este método se resume en variar los hiperparámetros del sistema CyMorph, de tal forma que se maximice la separación entre las distribuciones de galaxias espirales y elípticas, medida por δ_{GHS}, para cada variable. Para realizar dicha selección de hiperparámetros se tomó una muestra de los datos que consistía de mil galaxias elípticas y mil galaxias espirales. Una vez encontrados los hiperparámetros con los cuales se maximiza δ_{GHS} para cada variable, se calculan los valores del sistema CyMorph para cada una de las imágenes de las galaxias, generando así una representación tabular de la base de datos con la cual se entrenará a los modelos de aprendizaje máquina *tradicional*.

En el caso del modelo de aprendizaje profundo, puesto que no es factible probar todas las posibles arquitecturas y combinaciones para una red convolucional para llegar a la arquitectura óptima del problema en específico, se ocupa la arquitectura GoogleNet [14], que es una red neuronal propuesta por investigadores de Google, la cual es una red convolucional del estado del arte. Siendo una red convolucional, la base de datos con la que se entrenó este modelo consiste simplemente de las imágenes de las galaxias y sus etiquetas asociadas en cada caso.

Una vez definidas las bases de datos con las que se entrenará cada modelo, los autores realizaron distintos experimentos que conllevan a varios resultados, de entre los cuales podemos destacar los siguientes:

- Se trataron los problemas de clasificación binaria (espirales y elípticas) así como los problemas de clasificación multiclase en 3, 7, 9 y 11 clases. En todos los casos el modelo que obtuvo mejor rendimiento fue la red convolucional.
- Se estudió el impacto que tiene el tamaño de la galaxias con las que son entrenados los modelos. Esto se realizó restringiendo la base de datos, con las cuales se entrenaron los modelos, a solo las galaxias cuyo tamaño fuera mayor o igual a cierto valor. Esto a su vez implica que conforme se restringen las galaxias a tener un tamaño cada vez más grande la disponibilidad de estas es cada vez menor, en otras palabras, entre más grandes queramos que sean las galaxias menos galaxias se tendrán para entrenar los modelos. El resultado de este experimento fue que, aunque se tuvieran menos datos, entre más grandes fueran los tamaños de las galaxias mejor era el rendimiento de los modelos en general.
- El desbalanceo de clases es algo que siempre está presente en le problema de clasificación de galaxias. Restringiéndose al problema de clasificación en tres clases, se estudian varios métodos para mitigar este problema, tales como *oversampling*, *undersampling* y *Synthetic Minority Over-sampling Technique*, obteniéndose que la aplicación de cualquiera de estos métodos conlleva a mejoras similares.

A partir de este punto se restringe la discusión al problema de clasificación en dos clases, dejando prácticamente abierto el problema de más de tres clases, puesto que para más de tres clases nunca se reporta la matriz de confusión o cómo el desbalance de las clases afecta el rendimiento de los modelos.

Habiendo discutido lo anterior, los autores proceden a contextualizar los resultados de los modelos de clasificación binaria dentro de la astronomía. Se realizan discusiones como las siguientes:

- Se realizó una clasificación de un subconjunto de galaxias del catálogo del *Galaxy Zoo 1* [2], cuya clasificación está dada cómo *indefinida*, para posteriormente ver como eran las distribuciones de cantidades físicas, tales como la masa estelar $M_{stellar}$, etc., esto con el fin de saber si, usando los modelos obtenidos para la clasificación de nuevas galaxias, se recuperan propiedades de las galaxias bien establecidas. El

resultado de este ejercicio fue que sí se recuperan las propiedades deseadas de la clasificación hecha.

- Se estudia la fiabilidad de las predicciones realizadas por los modelos comparando con otros catálogos, se compara con el catálogo proporcionado por Nair [3] y con otro catálogo proporcionado por Dominguez [15]. Es importante mencionar que estos dos catálogos están basados en una clasificación diferente a la de los catálogos del *Galaxy Zoo 1* [2] y *Galaxy Zoo 2* [2], sin embargo los autores recuperan, con cierto margen de incertidumbre, las etiquetas necesarias para hacer comparaciones con el problema de dos y de tres clases. Analizando las distribuciones de las clases prerecidas con las dadas por los catálogos se concluye que, en general, los modelos arrojan resultados de acuerdo con los otros catálogos. Los autores terminan concluyendo que, aunque los resultados sean positivos, es difícil llevar a cabo este tipo de comparaciones de forma apropiada, debido a que las clases de estos dos catálogos no son del todo compatibles con las clases del *Galaxy Zoo 1* [2] y el *Galaxy Zoo 2* [2].

En [4], se presenta el trabajo derivado de aplicar métodos de aprendizaje máquina y aprendizaje profundo para la clasificación de galaxias según su morfología. Para entrenar los modelos y probar los modelos se usaron las bases de datos del *Galaxy Zoo 1* [2] y del *Galaxy Zoo 2* [2]. Se reportan, en el mejor de los casos, los siguientes rendimientos para las tareas de clasificación:

- Clasificación binaria: 99.5 % de exactitud.
- Clasificación en 3 clases: 82.7 % de exactitud.
- Clasificación en 7 clases: 77.6 % de exactitud.
- Clasificación en 9 clases: 75.5 % de exactitud.
- Clasificación en 11 clases: 65.2 % de exactitud.

Sin embargo, es desafortunado que no podemos tomar los rendimientos para las tareas de clasificación multiclase como referencia para el trabajo presentado en esta tesis, por dos razones fundamentales. La primera es que no se especifica la metodología seguida para obtener las clases mencionadas, haciéndonos imposible reproducir un conjunto de datos que use el mismo esquema de etiquetado. La segunda es que no reportan la matriz

de confusión para ninguna de las tareas de clasificación multiclase, de esta forma, aunque las precisiones totales reportadas parecen ser buenas no podemos saber cuál fue el rendimiento de los modelos sobre cada una de las clases en específico.

Aunque se estudia el rendimiento de los modelos obtenidos haciendo comparaciones con otros catálogos (Nair [3] y Dominguez [15]), puede que esta comparación no sea en estricto valida, puesto que, como se menciono anteriormente, las etiquetas de los últimos no son del todo compatibles con las etiquetas de las bases de datos con las que fueron entrenados los modelos.

Si bien el trabajo presentado por Barchi [4], es el que presenta los experimentos más parecidos a los realizados en este trabajo de tesis, y es por eso que se ha dado una revisión tan profunda de él, existen otros artículos que presentan trabajos que merecen la pena de ser mencionados, y de los cuales haremos una breve reseña.

Un trabajo ya mencionado anteriormente es el de Domínguez Sánchez [15]. Aquí se entrenan varios modelos de redes neuronales convolucionales para la realización de varias tareas que involucran imágenes de galaxias. Por ejemplo, entrenan un modelo de redes neuronales para reproducir las respuestas obtenidas para algunas preguntas del árbol de decisión empleado en el *Galaxy Zoo 2* [2], también se entrenó una red neuronal para distinguir entre las galaxias con clasificación E y S0 (ver fig 1.1), sin embargo el modelo que resulta de mayor interés en nuestro caso es el que trata de resolver el problema de clasificación multiclase en la base de datos de Nair [3]. Se menciona que dicho modelo tiene problemas para clasificar a las clases más a la derecha en la secuencia de Hubble, sin embargo no se reporta ni la exactitud total en esta tarea ni la matriz de confusión para poder conocer cuál es el rendimiento del modelo obtenido sobre cada una de las clases. Tampoco se proporciona el conjunto de prueba empleado en los experimentos, lo que nos impide usar los resultados de este trabajo como punto de referencia.

Otro trabajo reciente que aborda el problema de la implementación de redes neuronales para tareas de clasificación de imágenes de galaxias es el de Vega-Ferrero [16], donde se aborda de nuevo la clasificación binaria de galaxias elípticas y espirales, reportando una exactitud de hasta el 97 %. También se trata el problema de clasificación binaria entre las galaxias *face-on* y las *edge-on*, que es una clasificación que hace referencia a la orientación del plano de simetría preferencial de la galaxias respecto al observador, logrando una exactitud de hasta el 98 %.

En la tabla 2.1 se presenta una resumen de los resultados de los trabajos mencionados en esta sección.

Año	Autor	Método	Resultados
2020	Vega	CNN	Clasificación binaria (elípticas vs. espirales): 97 % de exactitud. Clasificación binaria (face-on vs. edge-on): 98 % de exactitud.
2019	Barchi	CNN	Clasificación binaria (elípticas vs espirales): 99.5 % de exactitud. Clasificación en 3 clases: 82.7 % de exactitud. Clasificación en 7 clases: 77.6 % de exactitud. Clasificación en 9 clases: 75.5 % de exactitud. Clasificación en 11 clases: 65.2 % de exactitud.
2018	Domínguez	CNN	Reproduce un conjunto de respuestas a la encuesta del Galaxy Zoo 2 con más del 95 % de exactitud. Pero no reporta ni la exactitud ni la matriz de confusión obtenida al realizar la tarea de clasificación multiclase en la base de datos de Nair.

Tabla 2.1: Resumen de los resultados encontrados en el estado del arte para la clasificación de imágenes de galaxias usando redes neuronales.

Teniendo todo lo anterior presente, finalizamos esta sección haciendo la siguiente critica a estos trabajos. Puesto que no mencionan cómo reproducir sus conjuntos de prueba y tampoco reportan las matrices de confusión para las tareas de clasificación multiclase es imposible tomar sus resultados como punto de referencia para futuras investigaciones.

2.2. Aprendizaje autosupervisado

El crear sistemas que usen inteligencia artificial para la automatización de ciertas tareas es algo que ha cobrado fuerza en los últimos años. De forma más específica, en el área de visión por computadora, se han ocupado modelos de inteligencia artificial para la automatización de tareas como la detección de objetos, reconocimientos de objetos, etc. Sin embargo, y más en el área del aprendizaje profundo, el éxito de los modelos implementados en estos sistemas siempre está sujeto a la disponibilidad de datos etiquetados.

La generación de bases de datos etiquetados con el volumen suficiente para entrenar redes neuronales profundas para realizar alguna tarea específica por lo general es un proceso muy costoso, sino es que inasequible en muchos casos. Por el contrario, la disponibilidad de un gran volumen de datos no etiquetados, suele ser una situación que se presenta mucho más frecuentemente.

Una alternativa para resolver el problema de falta de datos sería la transferencia de conocimiento, supongamos el escenario en el cual se quiere resolver una tarea para la cual se tienen pocos datos, pero existe algún modelo que fue entrenado exitosamente, probablemente con una base de datos mucho más grande, para una tarea diferente pero con cierta similitud con la tarea de original. El proceso conocido como transferencia de conocimiento consiste en usar el modelo ya entrenado para resolver la tarea en donde se cuentan con pocos datos. Aunque esta idea en papel suena bastante buena, en muchos casos los modelos están diseñados de forma tan específica para realizar la tarea para la cual fueron entrenados, que transferir lo aprendido a otras tareas no deriva en resultados positivos.

Debido a lo anterior, actualmente se realizan muchos esfuerzos en investigación para encontrar técnicas y marcos de trabajo nuevos que permitan a modelos de inteligencia artificial aprender representaciones útiles para varias tareas usando datos no etiquetados. Hoy en día existen diversas propuestas para resolver el problema mencionado, de las propuestas más atractivas está la del aprendizaje autosupervisado. A grandes rasgos, el marco de trabajo del aprendizaje autosupervisado consiste en crear una *tarea pretexto*,

en la cual se crea una forma de etiquetar artificialmente los datos, y se entrena al modelo para predecir estas etiquetas, esperando que lo que las representaciones que aprenda sean de utilidad para otras tareas, posiblemente supervisadas.

En [17] Kolesnikov hace una revisión de las *tareas pretexto* en aprendizaje autosupervisado para la extracción de representaciones visuales. Aunque los resultados principales del trabajo se centran en ver cómo al elección de la arquitectura de red neuronal afecta el rendimiento de los modelos, más que en discutir las características de cada *tarea pretexto* en sí. A pesar de lo anterior, se puede tomar perfectamente como referencia del estado actual del marco de trabajo del aprendizaje autosupervisado.

A continuación, presentamos una lista de las *tareas pretexto* con las que trabaja Kolesnikov en [17], dando una muy breve descripción de cada una. Más adelante, en la sección 3.5, tendremos oportunidad de hacer una revisión más detallada de las técnicas empleadas para este trabajo de tesis.

- **Rotacion**: Esta tarea pretexto fue anteriormente presentada en [18]. Consiste en rotar una imagen por alguno de los ángulos $\{0°, 90°, 180°, 270°\}$ y entrenar a la red neuronal para que prediga dicho ángulo.

- ***Exemplar***: Anteriormente presentado en [19]. Consiste en considerar cada imagen como una clase, es decir cada imagen pertenece a su propia clase. Posteriormente, usando aumentado de datos para crear variaciones de la misma imagen, se entrena a una red neuronal para el problema de clasificación.

- **Jigsaw**: Estudiada con anterioridad en [20]. Consiste en cuadricular la imagen usando una red de 3×3 y posteriormente alterar el orden de los mosaicos de la cuadrícula y entrenar a la red neuronal para que prediga cómo fueron desordenadas las imágenes.

- ***Relative Patch Location***: Tratado anteriormente en [21]. Se hace una cuadrícula similar a la de la tarea de *Jigsaw*, y se entrena una red neuronal para que aprenda a decir la localización relativa entre dos de los mosaicos.

En trabajos previos de aprendizaje autosupervisado para obtener representaciones visuales [18–22] era común que se usara la arquitectura conocida como AlexNet (nombrada así por el investigador que la propuso Alex Krizhevsky) [23] como base para los modelos. Sin embargo, en [17], puesto que es un trabajo enfocado a estudiar los efectos

de distintas arquitecturas en estos métodos, Kolesnikov usa arquitecturas más recientes propias del estado del arte.

En [17] se ocupan distintas arquitecturas como base para los modelos, tales como ResNet (*residual network*) [24], RevNet (*reversible residual network*) [25] y VGG (nombrada así por el grupo de la propuso, el *Visual Geometry Group*) [26] . Las bases de datos ocupadas para los experimentos fueron las siguientes:

- *ImageNet* [27]: Consiste de 1.3 millones de imágenes clasificados en 1000 clases.
- *Places205* [28]: Consiste de alrededor de 2.5 millones de imágenes lugares, clasificados en 205 clases.

El entrenamiento autosupervisado se realizó en la base de datos de *ImageNet* [27] y se usó para evaluar tanto *ImageNet* [27] como *Places205* [28].

El lector podrá pensar que, para evaluar la calidad de las representaciones visuales aprendidas por los modelos en el entrenamiento autosupervisado, se hace transferencia de conocimiento y se entrena una red neuronal para resolución de alguna tarea supervisada, sin embargo, aunque es un esquema válido, en [17] Kolesnikov opta por otra opción frecuentemente usada en estos casos. Para evaluar la calidad de la representación visual adquirida por los modelos a través del entrenamiento autosupervisado se evalúa tomando los *pre-logits*, que es la salida de la capa de la red que va antes de la capa de clasificación, y usándolos como representación de las imágenes se entrena un clasificador lineal para la tarea supervisada, que, en este caso, es clasificar las imágenes de las clases en sus correspondientes clases.

Habiendo fijado el método de evaluación de los modelos, se hizo un estudio de cómo variaba el rendimiento según la tarea de aprendizaje autosupervisado y la arquitectura de la red neuronal. Dentro de los diferentes tipos de redes ya mencionados se hicieron más variaciones, cambiando el número de filtros por capas en las redes neuronales.

Derivado de estos experimentos se encontraron, entre otros, los siguientes resultados:

- El *ranking* de los métodos de aprendizaje autosupervisado, basado en el rendimiento de su representación, no es independiente de la arquitectura de la red neuronal, es decir, el método con el cual se obtuvo el mejor rendimiento cambia dependiendo de la arquitectura. De forma similar, se encontró que el *ranking* de las arquitecturas, basado en su rendimiento, no es independiente del método de aprendizaje

autosupervisado, es decir la arquitectura con la cual se obtuvo el mejor rendimiento cambia según el método de aprendizaje autosupervisado empleado. Dicho de otra manera, el *ranking* de rendimientos depende tanto de la arquitectura como del método de aprendizaje autosupervisado.

- Posteriormente se comparan los rendimientos que se obtienen al usar las representaciones aprendidas a través de los métodos autosupervisados con los rendimientos obtenidos usando representaciones aprendidas por el camino supervisado. Resultando que, aunque se mejoran los rendimientos de trabajos previos, aún no se alcanzan los rendimientos obtenidos por la vía supervisada.

- Incrementar el número de canales en las redes neuronales mejora la calidad de las representaciones aprendidas por métodos autosupervisados. Aunque este es un fenómeno que se presenta también en el caso supervisado, la mejora es más significativa para el caso autosupervisado.

- Un mejor rendimiento en la *tarea pretexto* no implica una mejor calidad de la representación aprendida. Si se fija la arquitectura, el rendimiento en la *tarea pretexto* puede dar un indicio de cuál modelo proporcionará la mejor representación, sin embargo, si se varía la arquitectura esto no sucede. Puede pasar que la representación obtenida por la arquitectura con mejor rendimiento en una *tarea pretexto* sea peor que la representación obtenida por otra arquitectura con peor rendimiento en la misma *tarea pretexto*.

De los resultados anteriores quisiéramos hacer énfasis en el hecho de que las representaciones visuales aprendidas por redes neuronales convolucionales usando métodos de aprendizaje autosupervisado, no se acercan a ser tan buenas como las representaciones aprendidas por medios supervisados. Sin embargo, dado que los datos no etiquetados, que son los que usaremos para realizar las tareas autosupervisadas, exceden por un orden de magnitud a los etiquetados, es por esto que la expectativa de este trabajo de investigación es que esto ayude a que la representación obtenida por medios autosupervisados iguale o mejore a los obtenidos de forma supervisada.

Es importante que el lector tenga este hecho bien presente, puesto que se hará uso de él en la discusión de los resultados obtenidos en esta tesis.

Capítulo 3

Marco teórico

3.1. La neurona artificial

El aprendizaje profundo es un conjunto de métodos matemáticos con los cuales se modelan datos con arquitecturas complejas creadas a partir de combinar diferentes transformaciones no lineales. Los bloques elementales del aprendizaje profundo son las llamadas neuronas artificiales, las cuales son usadas para formar redes neuronales, que son los modelos con los que se pretende reproducir los datos. El nombre *profundo* nace de que al crear una red neuronal, a partir de otras redes neuronales más pequeñas, se obtiene una red neuronal profunda.

Las técnicas del aprendizaje profundo han permitido realizar grandes avances en los campos del procesamiento de imágenes, procesamiento de audio y procesamiento del lenguaje natural. Tareas como el reconocimiento facial, reconocimiento del lenguaje y la clasificación de textos son ejemplos en los cuales los métodos del aprendizaje profundo han tenido un enorme éxito en los últimos años. Las posibles aplicaciones de las redes neuronales son muy variadas, una muestra de esto es que un programa de computadora llamado *AlphaGo*, usando redes neuronales, aprendió a jugar Go y derrotó al campeón mundial de Go en 2016.

Existe una gran variedad de tipos de arquitecturas de redes neuronales, algunas de las más famosas son:

- Perceptrones multicapa, que son las más simples y antiguas.

- Redes neuronales convolucionales, también conocidas como CNN (por sus siglas en inglés), que son especialmente útiles para el procesamiento de imágenes.

- Redes neuronales recurrentes, las cuales se usan especialmente para datos secuenciales, tales como el texto y series de tiempo.

Una neurona artificial es una función $f : \mathbb{R}^n \to \mathbb{R}$ que está definida por un conjunto de coeficientes reales $\boldsymbol{\theta} = (w_1, ..., w_n, b) = (\boldsymbol{w}, b)$, donde a $\boldsymbol{w}$ y b se les conoce como pesos y sesgo respectivamente, y una función, llamada de activación, $\phi : \mathbb{R} \to \mathbb{R}$, la cual debe ser una función no lineal. De modo que si $\boldsymbol{x}$ es un vector de n entradas la neurona artificial le asignará el siguiente valor

$$f(\boldsymbol{\theta}; \boldsymbol{x}) = \phi(a) \qquad \text{donde} \qquad a = \boldsymbol{w} \cdot \boldsymbol{x} + b = \sum_{i=1}^{n} w_i x_i + b. \tag{3.1}$$

al valor a se le conoce comúnmente como activación.

Aunque la neurona artificial podría parecer una estructura matemática bastante rudimentaria, si le dotamos de las herramientas adecuadas, es capaz de realizar uno de los algoritmos de aprendizaje máquinas más conocidos, la regresión logística.

Supongamos que tenemos un problema de clasificación binaria, es decir, tenemos un conjunto de datos $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, donde cada elemento del conjunto de datos representa la información que sabemos de algún tipo de objeto, las etiquetas $y^{(i)}$ nos dicen si el objeto que representan pertenece o no a la clase que nos interesa. Consideremos desde ahora que solo pueden tomar los valores de 0 o 1, tomando el valor 1 cuando pertenecen a la clase objetivo y 0 en caso contrario, y los $\boldsymbol{x}^{(i)}$ representan características relevantes de los objetos ademas de la etiqueta. Un ejemplo de este tipo de problema podría ser uno en el que estemos interesados en identificar a una especie en especifico de flores, de tal forma que las $\boldsymbol{x}^{(i)}$ darían características de las flores tales como el color, la longitud de las hojas y de los tallos, etc. y las $y^{(i)}$ nos dirían si las flores pertenecen a la especie que nos interesa.

Nuestra intención es que la neurona artificial, como la definimos arriba, reproduzca los mejor posible el conjunto de datos antes mencionado, tomando por entrada las $\boldsymbol{x}^{(i)}$ y tratando de predecir las etiquetas $y^{(i)}$. Para lograr esto es necesario dotar a nuestro modelo de tres cosas, la primera, una función de activación apropiada, la segunda, una forma de medir qué tan bien la neurona artificial reproduce los datos y, la tercera, una

forma de modificar los pesos y el sesgo de tal forma que mejoren el desempeño de la neurona.

Si queremos definir la función de activación de nuestra neurona es necesario aclarar primero el significado de su salida. Una forma apropiada de atacar el problema es pidiendo que la salida $\hat{y}^{(i)} = f(\boldsymbol{\theta}; \boldsymbol{x}^{(i)})$ represente la probabilidad de que el objeto representado por la $\boldsymbol{x}^{(i)}$ pertenezca a la clase objetivo, por lo tanto debemos imponer sobre las $\hat{y}^{(i)}$ la condición de tomar valores entre 0 y 1. Una forma de lograr esto es definiendo a la función de activación como la función *sigmoide* o función logística

$$\phi(z) = sigm(z) = \frac{1}{1 + e^{-z}}. \tag{3.2}$$

Para poder evaluar el desempeño de nuestro modelo definimos una función llamada función de error, que es una función escalar de las etiquetas y las predicciones que hace la neurona $\hat{y}^{(i)}$ dadas las $\boldsymbol{x}^{(i)}$, permitámos sacar de la manga función de error que se sabe que funciona en este tipo de problemas, la función de error llamada entropía cruzada binaria

$$E(y, \hat{y}) = -\left[y \ln\left(\hat{y}\right) + (1 - y) \ln\left(1 - \hat{y}\right)\right], \tag{3.3}$$

Analicemos un momento qué está pasando en esta función. Supongamos que la etiqueta tiene un valor de $y = 1$ eso quiere decir que la función tendrá la siguiente forma

$$E(y, \hat{y}) = -\ln(\hat{y}), \tag{3.4}$$

debido al comportamiento de la función del logaritmo natural entre más cercano este $\hat{y}$ del número 1, y por lo tanto más cercano a predecir la probabilidad de que el objeto esté en la clase correcta, la función de error tendrá un valor más cercano a cero, y entre más cercano esté $\hat{y}$ del cero, y por lo tanto más cercano a predecir la clase del objeto de forma errónea, la función de error arrojará un valor más cercano a infinito, puesto que la función logaritmo tiende a menos infinito en el cero. Esta función tendrá un comportamiento totalmente análogo en el caso en que $y = 0$, devolviendo valores cercanos a cero cuando la neurona esté cerca de predecir mejor a la clase y valores más grande entre más *equivocada* sea la predicción. De tal manera que entre más pequeños sean los valores que arroja la función de error mejores serán las predicciones que realiza

la neurona. Así podemos definir al error en todo el conjunto de datos como la suma del error sobre cada elemento

$$\mathcal{L} = \sum_i E(y^{(i)}, \hat{y}^{(i)}). \tag{3.5}$$

Ya que dotamos a nuestra neurona de una función de activación y una función de error apropiada para el problema, debemos encontrar la forma de modificar los pesos y el sesgo de la neurona para mejorar el rendimiento del modelo. Dado el conjunto de datos $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ podemos ver a $\mathcal{L}$ como una función solo de los parámetros $\boldsymbol{w}$ y b. Puesto que, entre menor sea el valor de la función de error mejor serán las predicciones de la neurona, podemos ver ahora a nuestro problema como un problema de optimización, en el cual debemos encontrar el valor mínimo (o algún mínimo local) de $\mathcal{L}$. Una forma de realizar este proceso de optimización es a través del método llamado descenso por gradiente. El gradiente de una función escalar es un vector que apunta en la dirección en la cual la función cambia de forma más rápida en el sentido positivo, es decir en la dirección en la que la función decrece más rápido, por lo tanto si nos moviéramos en el sentido opuesto del gradiente nos moveríamos de la forma más rápida al mínimo de la función.

Teniendo lo anterior en mente el método del descenso por gradiente puede explicarse de la siguiente manera: se inicializan los pesos y el sesgo a valores *pequeños* $\boldsymbol{\theta}^{[0]}$ distintos de cero de forma aleatoria, después se calcula el gradiente de la función de error en el punto $\boldsymbol{\theta}^{[0]}$ y posteriormente se desplazan los coeficientes en el sentido opuesto de éste para encontrar otro valor de $\boldsymbol{\theta}$ que funcione mejor, es decir

$$\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^{[k]} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{[k]}) \qquad \text{para} \qquad k = 1, 2, 3, ... \tag{3.6}$$

Al parámetro α se le conoce como tasa de aprendizaje e indica qué tan grande es el paso que se da en la dirección opuesta del gradiente y es un hiperparámetro del modelo. Este proceso se repite de forma iterativa hasta alcanzar un número preestablecido de iteraciones, que el valor de la función de error pase un umbral preestablecido o que la función de error sea cero. Así hemos podido reproducir el modelo de aprendizaje de máquinas conocido como regresión logística usando la neurona artificial.

3.2. Perceptrón multicapa & retropropagación

Como vimos en la sección pasada la neurona artificial es una estructura matemática con la cual podemos reproducir algunos algoritmos de aprendizaje máquina, sin embargo, el lector no necesita ser muy suspicaz para darse cuenta que la neurona artificial padece de bastantes limitantes, por ejemplo, que la función de activación solo puede tomar como argumento una combinación lineal de las entradas más un desplazamiento, esto limita mucho los patrones que nuestro modelo es capaz de ajustar.

No obstante, y como mencionamos anteriormente, la neurona artificial simplemente es el bloque elemental de arquitecturas más complejas, llamadas redes neuronales, las cuales han demostrado ser capaces de reconocer características de alto nivel en distintos problemas. Un primer intento por crear una estructura más compleja sería el de componer varias neuronas artificiales una tras la otra, también podríamos componer varias neuronas al mismo tiempo a modo de capas, como se muestra en la fig. 3.1, formando así una estructura a modo de red, de aquí el nombre de red neuronal. A las capas que se encuentran antes de la última se les conoce como capas ocultas, debido a que no se conoce el valor *real* que deben de tener su salidas, contrario a la última capa que se sabe que sus salidas deben coincidir con las etiquetas $y^{(i)}$. Este tipo de modelo es conocido como perceptrón multicapa. Las capas de la red mostradas en la fig. 3.1 son conocidas como capas totalmente conexas, debido a que las salidas de una capa están conectadas a todas las neuronas de la siguiente capa.

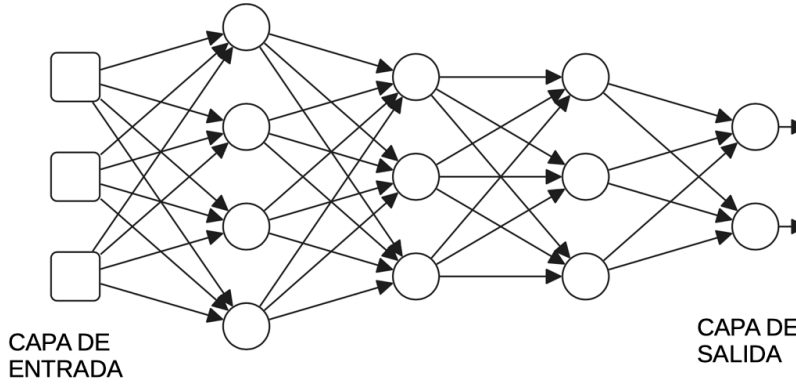

Figura 3.1: Perceptrón multicapa (imagen tomada de [29])

En la sección 3.1 vimos el caso en el que la salida del modelo es un escalar, pero como

podemos ver en la fig.3.1, también podemos crear modelos modelo cuya salida sea un vector, es decir que regrese varios escalares. Además de esto existe una gran variedad de funciones de activación y funciones de error que son ocupadas para distintas tareas, por ejemplo:

- **Clasificación Multiclase**: En este tipo de problemas se debe asociar a los objetos a una clase de entre N clases diferentes, para esto el modelo deberá devolver un vector de N entradas, usar una función de activación llamada función *softmax* y usar una función de error conocida como entropía cruzada categórica. En este modelo cada entrada del vector de salida representa la probabilidad de que el objeto esté en alguna clase. Al solo poder pertenecer a una clase la función de activación es tal que la suma de las entradas del vector de salida debe sumar uno y se clasifica al objeto en la clase a la cual es más probable que pertenezca.
- **Clasificación Multietiqueta**: En este problema, a diferencia del caso anterior, los objetos pueden pertenecer a una o más clases al mismo tiempo de entre un conjunto de N clases. La salida de un modelo que resuelva un problema de clasificación multietiqueta deberá ser un vector de N entradas, cada entrada del vector deberá provenir de una función de activación *sigmoide*, cada entrada indicará la probabilidad de que el objeto pertenezca a una clase. Sin embargo, dado que los objetos pueden pertenecer a varias clases al mismo tiempo, los eventos de que pertenezcan a una u otra clase no son disjuntos, por lo tanto la suma de las probabilidades no necesariamente será uno.

El proceso para que la red neuronal aprenda a reproducir los datos de entrada es totalmente análogo al mencionado en la sección anterior. No obstante, existe un paso que es bastante sútil en el caso del perceptrón multicapa. Calcular el vector gradiente respecto a los parámetros $\boldsymbol{\theta}$ para la neurona artificial es trivial, por ejemplo

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \left(\frac{\partial E}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial w_i}\right) \\ &= \left(\frac{\partial E}{\partial \hat{y}}\right)\left(\frac{\partial \phi}{\partial a}\right)\left(\frac{\partial a}{\partial w_i}\right) \\ &= \left(\frac{\partial E}{\partial \hat{y}}\right)\phi'(a)x_i,\end{aligned} \tag{3.7}$$

La facilidad con la que se calcula esta derivada proviene de que x_i ya no depende

de más variables. Sin embargo, en redes neuronales de más de una capa, calcular las derivadas respecto a los parámetros de las capas ocultas puede ser una tarea bastante tediosa, puesto que si se usara la regla de la cadena para calcular la derivada tendríamos tantos sumandos como caminos del nodo que contiene al parámetro respecto al cual estamos derivando al nodo de la salida del modelo. Computacionalmente esto también representa una desventaja, puesto que el número de sumandos en las derivadas crece de forma exponencial respecto a la longitud de los caminos antes mencionados.

Podemos darle vuelta a este problema usando una técnica llamada retropropagación [11], en la cual se calculan las derivadas respecto a los parámetros de las capas más cercanas a la salida del modelo y con ellas se pueden ir calculando, sin tanto costo computacional, las derivadas de las capas más alejadas. Veamos esto con un ejemplo, supongamos que tenemos una red compuesta por N capas, cada capa consiste de L neuronas y la salida es un escalar; para simplificar la notación ignoraremos los sesgos en este modelo y supondremos que todas las capas tienen la misma función de activación. Llamemos $z_i^{[k]}$ a la componente i-ésima del vector de salida de la k-ésima capa, $a_i^{[k]}$ a la componente i-ésima del vector de activaciones de la k-ésima capa, $w_{ij}^{[k]}$ a la j-ésima componente del vector de pesos de la i-ésima neurona de la k-ésima capa. De esta forma podemos escribir las componentes de la salida de la k-ésima entrada como

$$z_i^{[k]} = \phi(a_i^{[k]}) \qquad \text{donde} \qquad a_i^{[k]} = \sum_j w_{ij}^{[k]} z_j^{[k-1]}. \tag{3.8}$$

La derivada respecto a los pesos de la salida es totalmente análoga a la ec. 3.7

$$\frac{\partial E}{\partial w_j^{[N]}} = \frac{\partial E}{\partial \hat{y}} \phi'(a^N) z_j^{[N-1]}, \tag{3.9}$$

el peso $w_j^{[N]}$ solo tiene un índice porque la última capa solo tiene una neurona. Veamos cómo podemos calcular las derivadas respecto a los pesos de la $N-1$-ésima capa empleando las derivadas respecto a las capas subsecuentes

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{[N-1]}} &= \frac{\partial E}{\partial \hat{y}}\left(\frac{\partial \hat{y}}{\partial a^{[N]}}\right)\left(\frac{\partial a^{[N]}}{\partial w_{ij}^{[N-1]}}\right) \\ &= \sum_l \frac{\partial E}{\partial \hat{y}}\left(\frac{\partial \hat{y}}{\partial a^{[N]}}\right)\left(\frac{\partial a^{[N]}}{\partial z_l^{[N-1]}}\right)\left(\frac{\partial z_l^{[N-1]}}{\partial w_{ij}^{[N-1]}}\right) \\ &= \sum_{l,k} \frac{\partial E}{\partial \hat{y}}\left(\frac{\partial \hat{y}}{\partial a^{[N]}}\right)\left(\frac{\partial a^{[N]}}{\partial z_l^{[N-1]}}\right)\left(\frac{\partial z_l^{[N-1]}}{\partial a_k^{[N-1]}}\right)\left(\frac{\partial a_k^{[N-1]}}{\partial w_{ij}^{[N-1]}}\right),\end{aligned} \tag{3.10}$$

si tenemos que

$$\frac{\partial \hat{y}}{\partial a^{[N]}} = \frac{\partial \phi(a^{[N]})}{\partial a^{[N]}} = \phi'(a^{[N]}) \tag{3.11}$$

$$\frac{\partial a^{[N]}}{\partial z_l^{[N-1]}} = \frac{\partial}{\partial z_l^{[N-1]}}\left(\sum_m w_m^{[N]} z_m^{[N-1]}\right) = w_l^{[N]} \tag{3.12}$$

$$\frac{\partial z_l^{[N-1]}}{\partial a_k^{[N-1]}} = \frac{\partial \phi(a_k^{[N-1]})}{\partial a_k^{[N-1]}} = \phi'(a_k^{[N-1]}) \tag{3.13}$$

$$\frac{\partial a_k^{[N-1]}}{\partial w_{ij}^{[N-1]}} = \frac{\partial}{\partial w_{ij}^{[N-1]}}\left(\sum_n w_{kn}^{[N-1]} z_n^{[N-2]}\right) = \delta_{ik} z_j^{[N-2]}, \tag{3.14}$$

donde δ_{ij} es la delta de Kronecker, entonces

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{[N-1]}} &= \sum_{l,k} \frac{\partial E}{\partial \hat{y}} \phi'(a^{[N]}) w_l^{[N]} \phi'(a_k^{[N-1]}) \delta_{ik} z_j^{[N-2]} \\ &= \sum_l \frac{\partial E}{\partial \hat{y}} \phi'(a^{[N]}) w_l^{[N]} \phi'(a_i^{[N-1]}) z_j^{[N-2]}.\end{aligned} \tag{3.15}$$

Con esto hemos podido calcular las derivadas de los pesos de la penúltima capa usando la regla de la cadena *al revés*, empezando por derivar respeto a los pesos más cercanos a la salida de la red y derivar iterativamente respecto a los pesos de las capas más alejadas. Esta forma de calcular el error puede interpretarse como ver cómo se propagan las variaciones en el error de adelante hacia atrás, de ahí el nombre de retropropagación. Esta forma de derivar permite a las computadoras llevar a cabo el método del descenso por

gradiente de forma eficiente, volviendo posible entrenar modelos de perceptrón multicapa con varias capas de profundidad.

Otro punto que es importante señalar es que esta implementación hace fuerte uso de que podemos ver a las operaciones que realiza cada capa del perceptrón multicapa como una multiplicación de matrices. Notemos que podemos reescribir a las activaciones de la ec. (3.8) como

$$\boldsymbol{a}^{[k]} = \boldsymbol{W}^{[k]} \boldsymbol{z}^{[k-1]^T}, \tag{3.16}$$

donde las entradas de la matriz $\boldsymbol{W}^{[k]}$ son los pesos $w_{ij}^{[k]}$.

3.3. Redes neuronales convolucionales

Las redes neuronales convolucionales, también conocidas como redes convolucionales o por sus siglas en inglés CNN, son redes cuya estructura permite trabajar con datos estructurados en forma de red o cuadrícula, los cuales presentan una gran dependencia espacial de forma local. El ejemplo arquetípico de este tipo de datos son las imágenes, ya sean a blanco y negro (de una dimensión) o a colores (con tres dimensiones), aquí las celdas son los pixeles y estas manifiestan una gran dependencia espacial respecto a sus pixeles adyacentes puesto que al menos en una vecindad pequeña alrededor de cada pixel sus vecinos deben tener valores (colores) parecidos. Otro ejemplo de datos que pueden ser procesados por una red convolucional pueden ser datos en forma de secuencia, tales como texto o series de tiempo. De la misma forma que en el caso anterior, los valores en este tipo de datos representan una fuerte dependencia espacial local, por ejemplo, una palabra en un texto esta evidentemente relacionada con las palabras que van antes y después de ella, y en una serie de tiempo se espera que a partir de datos que sucedieron antes en el tiempo se pueda predecir los valores subsecuentes.

Las redes convolucionales son un claro ejemplo de cómo el crear una arquitectura inspirada en modelos biológicos reales puede llevar a resultados sorprendentes. Las primeras motivaciones para las redes convolucionales derivaron de experimentos realizados por Hubel y Weisel en la corteza visual de los gatos. En estos experimentos se encontró que existen pequeñas regiones de células en la corteza visual que son sensibles a regiones específicas del campo visual. Es decir, si áreas específicas del campo visual son estimuladas, entonces también serán activadas ciertas zonas específicas de la corteza visual. Más

aún, se encontró que bordes verticales activan a cierto conjunto de células neuronales y que los bordes horizontales activan a otro conjunto de células. Las capas convolucionales están inspiradas en esta arquitectura manifestada por las células. También se descubrió que estas células están estructuradas en forma de capas y este hecho condujo a la hipótesis de que los mamíferos usan estas diferentes capas para descomponer las imágenes a diferentes niveles de abstracción. Este fenómeno también ocurre en las redes convolucionales, en las cuales las primeras capas convolucionales captan formas más primitivas en las imágenes y las últimas capas captan formas más complejas [30].

En las siguientes subsecciones daremos los elementos básicos de las redes convolucionales, algunas operaciones que son incluidas en sus arquitecturas y la forma en que se entrenan.

3.3.1. Convolución

La estructura de una CNN puede entenderse como una conjunto de capas ordenadas, donde una capa opera sobre la salida de la capa anterior. Aquí daremos una revisión a los dos tipos de capas más comunes en una CNN, las cuales son las capas de convolución y las capas de *pooling*. En esta subsección en específico nos dedicaremos a dar una revisión de las primeras.

Una imagen tiene una representación matricial (o tensorial). Por el momento restrinjamos nuestra discusión al caso de imágenes en escala de grises, donde cada entrada de la matriz representa un pixel de la imagen. Con lo que hemos aprendido hasta ahora podríamos optar por la opción de aplanar la imagen y pasarla a través de un perceptrón multicapa para realizar alguna tarea de clasificación, regresión, etc. Al fin y al cabo una matriz de $N \times N$ puede verse como una vector de $N \times N$ entradas. Aunque esta aproximación es válida, ha demostrado funcionar solamente en conjuntos de imágenes muy sencillos, además de que conforme las imágenes sean más detalladas, y por lo tanto sean más grandes, el número de pesos también crecerá incrementando el costo computacional del entrenamiento.

La operación de convolución nos ayuda a dar la vuelta a este problema. En una capa convolucional los pesos deben ordenarse en una matriz, a la matriz de pesos se le conoce como *kernel* o filtro. Dada una imagen $\boldsymbol{I}$ y un filtro $\boldsymbol{K}$ el resultado de la convolución de $\boldsymbol{I}$ con $\boldsymbol{K}$ será una matriz que denotaremos por $\boldsymbol{I} * \boldsymbol{K}$, a esta matriz se le conoce también como mapa de características. La forma en la que está definida la componente i, j de la convolución $\boldsymbol{I} * \boldsymbol{K}$ es

$$(\boldsymbol{I} * \boldsymbol{K})_{i,j} = \sum_m \sum_n I_{i+m,j+n} K_{m,n}. \tag{3.17}$$

A primera vista esta ecuación puede parecer muy oscura, por lo que para poner un poco de luz sobre cómo operar realicemos un ejemplo con valores reales. Supongamos que $\boldsymbol{I}$ y $\boldsymbol{K}$ tienen los siguientes valores

$$\boldsymbol{I} = \begin{bmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{y} \quad \boldsymbol{K} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}, \tag{3.18}$$

para implementar la ec. (3.17) debemos usar índices que empiecen desde cero. Calculemos ahora la componente $(\boldsymbol{I} * \boldsymbol{K})_{0,0}$

$$\begin{aligned} (\boldsymbol{I} * \boldsymbol{K})_{0,0} =& I_{0,0}K_{0,0} + I_{0,1}K_{0,1} + I_{0,2}K_{0,2} \\ &+ I_{1,0}K_{1,0} + I_{1,1}K_{1,1} + I_{1,2}K_{1,2} \\ &+ I_{2,0}K_{2,0} + I_{2,1}K_{2,1} + I_{2,2}K_{2,2} \\ =& 3 \cdot 0 + 3 \cdot 1 + 2 \cdot 2 + 0 \cdot 2 + 0 \cdot 2 + 1 \cdot 0 + 3 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 \\ =& 12, \end{aligned} \tag{3.19}$$

calculemos $(\boldsymbol{I} * \boldsymbol{K})_{0,1}$ y $(\boldsymbol{I} * \boldsymbol{K})_{0,2}$

$$\begin{aligned} (\boldsymbol{I} * \boldsymbol{K})_{0,1} =& I_{0,1}K_{0,0} + I_{0,2}K_{0,1} + I_{0,3}K_{0,2} \\ &+ I_{1,0}K_{1,1} + I_{1,2}K_{1,1} + I_{1,3}K_{1,2} \\ &+ I_{2,0}K_{2,1} + I_{2,2}K_{2,1} + I_{2,3}K_{2,2} \\ =& 3 \cdot 0 + 2 \cdot 1 + 1 \cdot 2 + 0 \cdot 2 + 1 \cdot 2 + 3 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 + 2 \cdot 2 \\ =& 12, \end{aligned} \tag{3.20}$$

$$
\begin{aligned}
(\boldsymbol{I} * \boldsymbol{K})_{0,2} =& I_{0,2}K_{0,0} + I_{0,3}K_{0,1} + I_{0,4}K_{0,2} \\
&+ I_{1,2}K_{1,1} + I_{1,3}K_{1,1} + I_{1,4}K_{1,2} \\
&+ I_{2,2}K_{2,1} + I_{2,3}K_{2,1} + I_{2,4}K_{2,2} \\
=& 2 \cdot 0 + 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 2 + 3 \cdot 2 + 1 \cdot 0 + 2 \cdot 0 + 2 \cdot 1 + 3 \cdot 2 \\
=& 17.
\end{aligned} \tag{3.21}
$$

La componente $(\boldsymbol{I} * \boldsymbol{K})_{0,3}$ no puede ser calculada puesto que no tenemos acceso a las componentes $I_{0,5}$, $I_{1,5}$ y $I_{2,5}$. Teniendo lo anterior en cuenta el mapa de características será una matriz de 3×3. Las componentes restantes se pueden calcular de forma análoga a como hicimos las anteriores, de tal forma que el mapa de características es el siguiente

$$
\boldsymbol{I} * \boldsymbol{K} = \begin{bmatrix} 12 & 12 & 17 \\ 10 & 17 & 19 \\ 9 & 6 & 14 \end{bmatrix}. \tag{3.22}
$$

En la fig. 3.2 podemos ver gráficamente cómo se llevan a cabo la operación de convolución.

La idea detrás de aplicar la operación de convolución es que los filtros nos ayuden a transformar a las imágenes en representaciones que sean más fáciles de procesar y que después de cada capa se resalten más las características importantes para realizar la tarea dada.

Existen técnicas que nos permiten extender la operación de convolución de tal forma que esta sea más versátil, aquí mencionaremos solo un par, las técnicas conocidas como *padding* y *striding*:

- *Padding*: Como notamos en la fig. 3.2 para realizar la operación de convolución debemos mover el filtro a lo largo y ancho de la entrada para realizar una operación de multiplicación a cada paso que da. Como hemos definido hasta ahora esta operación, el centro del filtro no pasará por los bordes de la imagen de entrada puesto que no hay nada más allá de la imagen sobre lo cual el filtro pueda operar. Esto puede representar un problema, dado que pueden existir características importantes en los bordes de la imagen que no estén siendo capturadas por el filtro. Para permitir que el filtro pueda pasar de forma más libre sobre los bordes

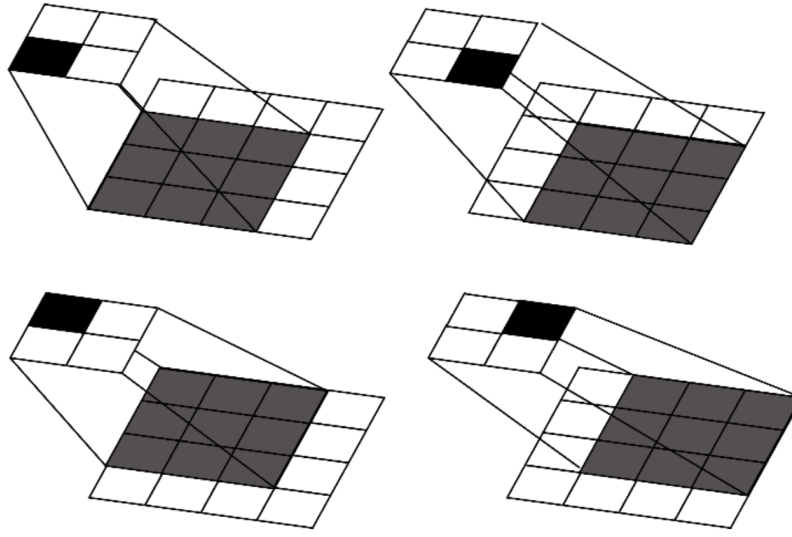

Figura 3.2: Operación de convolución de una entrada de 4×4 y un filtro de 2×2 (imagen tomada de [31])

es que aplicamos ls operación conocida como *padding*, la cual consiste en extender a la imagen original agregando pixeles falsos (por lo general ceros) en los bordes de la imagen. Esto también nos permite tener más control sobre el tamaño de la matriz de salida, ya que podemos hacer *padding* sobre la imagen de entrada de tal forma que la imagen de salida tuviera algunas dimensiones específicas. Siguiendo con el ejemplo anterior, si quisiéramos que la matriz de salida tuviera dimensiones de 5×5 solo tendríamos que concatenar un cero en cada extremo de la imagen de la siguiente manera

$$\boldsymbol{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 & 0 \\ 0 & 3 & 1 & 2 & 2 & 3 & 0 \\ 0 & 2 & 0 & 0 & 2 & 2 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.23}$$

- *Striding*: Otra forma de modular la operación de convolución es cambiando el ta-

maño de los pasos que da el filtro sobre la imagen. En el ejemplo que vimos anteriormente el filtro se movía a pasos de una entrada, tanto horizontal como verticalmente. Sin embargo, podríamos hacer que los pasos fueran más grandes, a esto se le conoce como *striding*. Definir un *stride* de 1 significa que cada paso tiene el tamaño de un pixel, si el *stride* es de 2 quiere decir que cada paso es de dos pixeles. También podemos definir el *stride* por pares, especificando de qué tamaño es el paso en el eje vertical y en el eje horizontal, por ejemplo un *stride* de $(2, 3)$ indica que los pasos en el eje vertical serán de 2 pixeles y los pasos en el eje horizontal serán de 3 pixeles. Un ejemplo gráfico puede verse en la fig. 3.3 donde se realiza la convolución usando un *stride* de 2 en ambas direcciones.

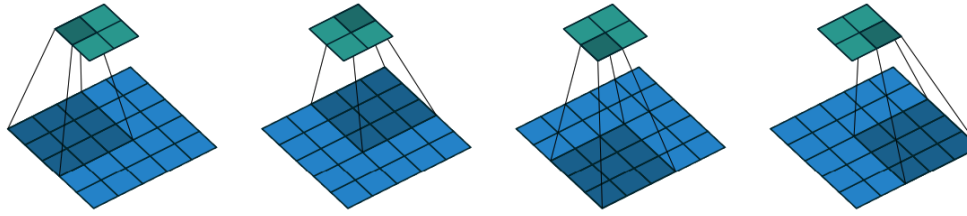

Figura 3.3: Operación de convolución con un *stride* de 2 en ambas direcciones (imagen tomada de [32])

3.3.2. *Pooling*

Como mencionamos anteriormente, otro bloque recurrente en la arquitectura de una CNN son las capas de *pooling*. Las operaciones de *pooling* son operaciones para poder resumir subregiones del mapa de características de tal forma que éste pueda expresarse de forma más pequeña sin perder demasiada información.

Las operaciones de *pooling* pueden entenderse como trasladar una ventana a lo alto y ancho del mapa de características y aplicar al contenido de la ventana operaciones como obtener el máximo o el promedio de dichos elementos, a estas técnicas se les conoce respectivamente como *max pooling* y *average pooling*. En la fig. 3.4 podemos ver un ejemplo de la aplicación de *max pooling*, el lector podrá fácilmente extender este ejemplo al caso de *average pooling*.

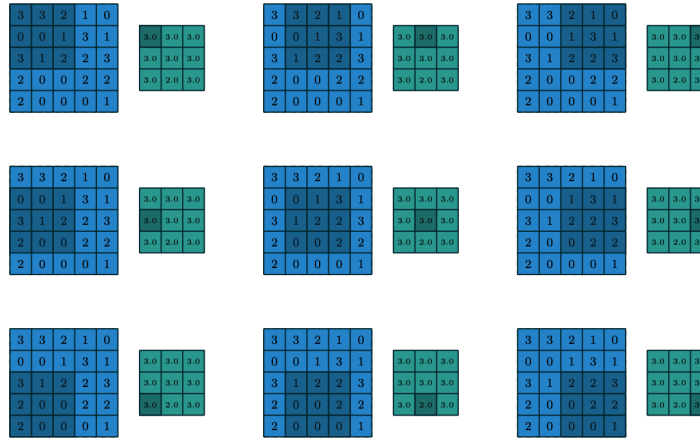

Figura 3.4: Ejemplo de aplicación de *max pooling* (imagen tomada de [32])

3.3.3. Arquitectura de una CNN

En la actualidad existe una exótica variedad de arquitecturas de redes convolucionales. Sin embargo, las arquitecturas más convencionales están formadas principalmente a partir de los siguientes bloques

- Capas convolucionales.
- Capas de *pooling*.
- Capas totalmente conexas.

La forma en la que se apilan comúnmente estas capas es la siguiente: primero se pondría una capa convolucional para generar un mapa de características de la imagen, es común ajustar los *strides* y el *padding* de tal forma que los mapas de características tengan el mismo tamaño en largo y ancho que la entrada, el número de filtros que compongan a la capa convolucional es un parámetro más arbitrario. Después de la capa convolucional sucede una capa de *pooling*, esta capa tiene como fin reducir el tamaño de los mapas de características arrojados por la capa convolucional; si en este punto se usa *max pooling* o *average pooling* también es una decisión de carácter más arbitrario. Posteriormente este patrón agrega una capa convolucional seguida de una capa de *pooling*), es repetido un número determinado de veces para después *aplanar* los mapas de características finales

en un solo vector, lo que se refiere a reorganizar todas las entradas de los mapas de características en un único vector, para después hacer pasar ese vector por una conjunto de capas totalmente conexas. Estas últimas capas realizarán el trabajo de clasificación o regresión según sea el caso.

3.4. Arquitecturas empleadas

En este trabajo de investigación implementamos redes neuronales convolucionales para la realización de tareas de clasificación. El lector ya se imaginará que la cantidad de configuraciones en las cuales se pueden organizar los bloques mencionados anteriormente son infinitas, bajo este escenario vale la pena preguntarse ¿Qué impediría a cada persona interesada en incursionar en el mundo del aprendizaje profundo diseñar una red a la medida de su problema de interés? Sin embargo, la realidad es que el crear una metodología bajo la cual se puedan diseñar redes convolucionales creadas a la medida, a partir de estos bloques, para un problema específico sigue siendo aún un problema abierto. Por el contrario, son contadas las configuraciones que han demostrado ser útiles para tareas tales como la extracción de representaciones visuales y otras tareas.

Los modelos implementados en este trabajo de investigación se basan en dos de estas arquitecturas ya preestablecidas. A continuación, presentamos una breve discusión de estas dos arquitecturas.

3.4.1. AlexNet

La arquitectura AlexNet fue la ganadora en 2010 del concurso *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), que es una competencia en la cual se evalúan los rendimientos de nuevos modelos para la tarea de clasificación de imágenes usando la base de datos de ImageNet. El mejor rendimiento obtenido en esta competencia suele tomarse como el estado del arte en la tarea de clasificación de imágenes cada año.

Los detalles de esta arquitectura fueron presentados por Krizhevsky, Sutskever y Hinton en [23]. Esta arquitectura estandarizó varias elecciones de diseño, las cuales sentaron las bases para arquitecturas posteriores, mencionaremos a continuación algunas de ellas. A partir de AlexNet se normalizó el uso de la función ReLU (*rectified linear unit*) como la elección más usual para las funciones de activación en las redes convolucionales, dejando de lado otras opciones como la función *sigmoide* o tanh. También, AlexNet fue

innovadora en el uso de ciertas estrategias para el entrenamiento de la red, tales como el aumentado de datos y el uso de GPUs (en español unidad de procesamiento gráfico) para el entrenamiento sobre grandes bases de datos, lo que permitió entrenar de forma más rapida la red, puesto que gracias a las GPUs muchos cálculos del entrenamiento pueden paralelizarse. Otras técnicas, como penalización L_2 y el uso de *Dropout*, para evitar el sobreajuste, fueron implementadas. La aplicación del método conocido como *Dropout* en el área de redes neuronales fue propuesta por Srivastava y Hinton, entre otros, en [33]. Esta técnica consiste en apagar ciertos conjuntos de pesos en la red, con cierta probabilidad en varios momentos del entrenamiento (fig. 3.5). Esto simula el efecto de entrenar varios modelos al mismo tiempo.

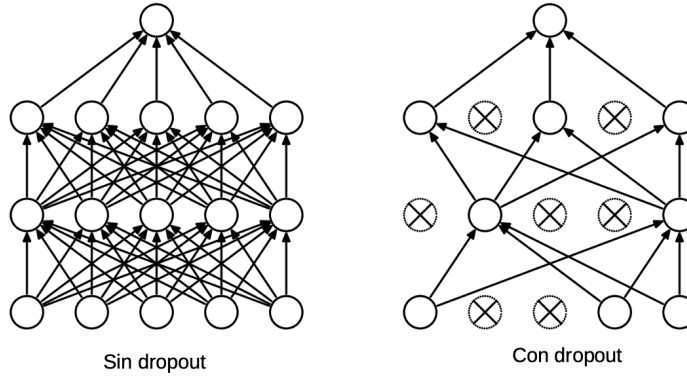

Figura 3.5: Ejemplo de *dropout* (imagen tomada de [33]).

3.4.2. ResNet

La arquitectura conocida como ResNet, la cual originalmente consistía de 152 capas, fue la arquitectura ganadora del ILSVRC en 2015, alcanzando un error top-5 [1] de 3.6 %, siendo el primer modelo en obtener un rendimiento comparable con el que obtendría un humano. Los detalles de la arquitectura fueron por primera vez presentados por He en [24]. Hasta este momento en la historia fue que se empezó realmente a usar redes neuronales profundas (en inglés las famosas *deep neural networks*), puesto que el número de capas de ResNet supera por un orden de magnitud el número de capas de arquitecturas

[1] Para calcular el error top-5 se consideran como predicciones buenas aquellas en las cuales la clase correcta aparece en el top-5 de clases más probables para cada punto de los datos.

previas. El que no se hubieran presentado arquitecturas tan profundas previamente no era una situación fortuita, hasta ese momento no se había podido entrenar exitosamente una arquitectura con tantas capas, debido a que conforme se aumentaban las capas iban apareciendo ciertas complicaciones que impedían entrenar el modelo apropiadamente. Procederemos a discutir brevemente estos problemas y cómo es que la arquitectura ResNet pudo resolverlos.

Como vimos en la sección 3.2, las derivadas respecto a los pesos de una capa pueden ser calculados a partir de productos de las derivadas respecto a los pesos de las capas subsecuentes. Por lo tanto, conforme vamos calculando las derivadas respecto a los pesos de capas más cercanas a la entrada, el número de factores necesarios para calcularlos también aumenta. Estos factores terminan teniendo valores pequeños, que al ser multiplicados todos entre sí producen números aún más pequeños. Esto resulta en que el gradiente que se ocupará para actualizar los pesos de las capas más cercanas a la entrada sea tan pequeño que el cambio de estos después de cada iteración del entrenamiento sea imperceptible, volviendo imposible el entrenamiento de estos pesos y afectando de forma terrible el rendimiento del modelo.

El fenómeno que acabamos de describir es conocido como desvanecimiento del gradiente. Otra forma de entender este fenómeno es la siguiente. Si visualizamos a una red neuronal como una gráfica, como se muestra en la fig. 3.1, el método de retropropagación lo que hará es propagar el error desde la salida a cada nodo de la red, a través de todos los caminos posibles, resultando que el error de las primeras capas es propagado a travez de caminos más largos que los errores de nodos más cercanos a la salida. El desvanecimiento del gradiente se ve reflejado en el hecho de que entre más largo sea el camino por el cual fue propagado el error será cada vez menor.

Pero... ¿Cómo es que la arquitectura ResNet pudo resolver este problema? Como podemos notar en la fig. 3.1 las conexiones entre las neuronas de una capa solo se dan entre neuronas de capas consecutivas, es decir, la salida de una capa solo la recibe la capa que se encuentra inmediatamente después. La característica más importante que tiene la arquitectura ResNet [24] es que no solo contiene conexiones entre capas consecutivas, sino que además permite conexiones entre capas no consecutivas. En la fig. 3.6 se muestra la unidad básica a partir de la cual fue creada la arquitectura ResNet, cómo podemos ver existe una conexión que se salta algunas capas, la cual simplemente toma la salida de la i-ésima capa y la suma a la salida de la $(i+1)$-ésima capa. Este tipo de conexiones provee a la red de dos importantes propiedades.

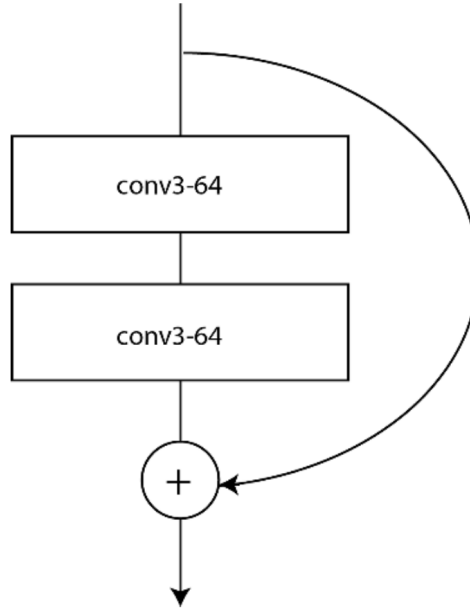

Figura 3.6: Ejemplo de bloque residual (imagen tomada de [31]).

La primera es que estas nuevas conexiones evitan que ocurra el desvanecimiento del gradiente, puesto que ahora el error tiene caminos más cortos, hacia las primeras capas, para propagarse.

La segunda es que, el que existan tanto caminos largos como cortos para que el error se propague a lo largo de las capas, tiene como resultado que la mayor parte del aprendizaje se da por parte de los caminos más cortos, y el aprendizaje que se da con los caminos más largos se interpreta como correcciones más finas (contribución residual). Esta situación le da al algoritmo de aprendizaje la flexibilidad de escoger el nivel de no linealidad que adoptará para cada problema. En problemas en los cuales no se requiera un alto nivel de no linealidad, la mayor parte del aprendizaje se dará a partir de los caminos más cortos (que se saltan muchas conexiones). Si se requiere usar un nivel más alto de no linealidad, las conexiones que conformen caminos más largos tomarán más relevancia en el aprendizaje [30]. Este nuevo esquema de aprendizaje es llamado aprendizaje residual, de ahí el nombre que se le da a la red ResNet (*residual network*).

En [34], Zagoruyko menciona que incrementar la profundidad de la red no siempre implica un mejor rendimiento, puesto que muchos de los caminos más profundos no

terminan siendo relevantes en el momento del aprendizaje. Es por esto que existen variaciones más pequeñas de la arquitectura original de ResNet (con 152 capas). Para este trabajo usamos la implementación de Tensorflow de ResNet de 50 capas, conocida como ResNet50.

3.5. Aprendizaje autosupervizado

Además de saber cómo funcionan las redes convolucionales, para entender el trabajo desarrollado en esta tesis es necesario contar con conocimientos sobre qué es aprendizaje autosupervisado, el cual conforma un subconjunto de técnicas del aprendizaje no supervisado.

La implementación de modelos de aprendizaje profundo ha permitido que las computadoras realicen tareas, tales como reconocimiento de objetos o detección de objetos, casi o igual de bien que los humanos. No obstante, el éxito de estos modelos se debe en gran medida al gran tamaño de los datos etiquetados con los que fueron entrenados. Esto pone sobre la mesa al menos un par de problemas, el primero es que para muchas tareas es imposible o prohibitivamente costoso hacerse de la cantidad necesaria de datos etiquetados para hacer que los modelos de aprendizaje profundo tengan desempeños aceptables. Otro problema es que las características que extrae la red neuronal de los datos son, por lo general, tan específicas de la tarea dada que no son útiles para propósitos más generales, es decir, que los pesos aprendidos por la red neuronal no sirven para hacer transferencia de conocimiento para otras tareas.

En este escenario, y sumando que en muchos casos la cantidad de datos no etiquetados supera por mucho la cantidad de datos etiquetados, se abre una oportunidad para crear técnicas de aprendizaje no supervisado que aprovechen la gran cantidad de datos no etiquetados que existe para poder aprender representaciones que ayuden, en contextos en los cuales la cantidad de datos etiquetados no sea suficiente, a realizar tareas de aprendizaje supervisado.

Es aquí donde entra el aprendizaje autosupervisado. Para realizar una tarea de aprendizaje autosupervisado se requiere tener datos no etiquetados y formular una tarea pretexto cuyo objetivo pueda ser alcanzado de forma no supervisada. La tarea pretexto consiste en encontrar una forma en la que la computadora genere etiquetas para los datos sin ayuda de un humano, es decir autoetiquetar los datos, y entrenar a una red neuronal para predecir dicha etiqueta.

Inspirado en técnicas de aprendizaje autosupervisado en el área de procesamiento del lenguaje natural, en las cuales se usa el contexto para la predicción de palabras [35], se han creado diversas tareas para realizar aprendizaje autosupervisado de representaciones visuales. Un ejemplo de tarea pretexto para la extracción de representaciones visuales sería el siguiente. Supongamos que contamos con un conjunto de imágenes, una forma de autoetiquetar los datos puede ser rotar la imagen original por 0^o, 90^o, 180^o y 270^o (ver fig. 3.7)y asignar como etiqueta a cada imagen rotada el ángulo de rotación correspondiente. Después de esto solo resta entrenar a una red neuronal para realizar la tarea de clasificación del ángulo de rotación de cada imagen [18].

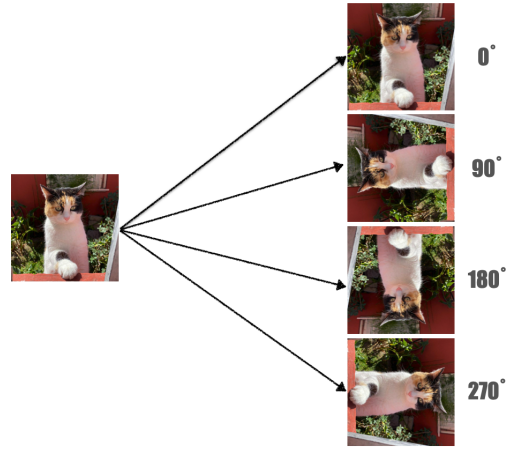

Figura 3.7: Autoetiquetado usando rotaciones para la tarea de pretexto.

Tareas pretexto puede haber una infinidad, pero la idea del aprendizaje autosupervisado es que la red neuronal aprenda a extraer características de los datos que sirvan para más tareas. Es por esto que una buena tarea de pretexto debe ser tal que sea necesario para el modelo extraer conocimiento de alto nivel de los datos para realizar la tarea. En [17], Kolesnikov hace una revisión de técnicas de aprendizaje autosupervisado que han demostrado ser buenas en el sentido anteriormente explicado, y que representan el estado del arte en el área de procesamiento de imágenes. Dichas técnicas son la de rotación, la cual ya discutimos brevemente, *relative patch location*, *jigsaw* y *exemplar*.

El problema que tratamos de resolver en este trabajo de investigación es el de clasificación de galaxias a través de su morfología. Dado que las características morfológicas de una galaxia no dependen de la orientación de esta, la técnica de rotación no es útil

en nuestro caso. Por lo tanto, las posibilidades que quedan son las tareas de *relative patch location*, *jigsaw* y *exemplar*. Aunque, durante la investigación se tenía el plan de implementar todos estos métodos, solo fue posible implementar las técnicas de *relative patch location* y *jigsaw*. A continuación, daremos una revisión de estas dos técnicas.

3.5.1. *Relative patch location*

Esta tarea pretexto consiste en predecir la posición relativa entre dos parches de la imagen [21]. Para realizar esta tarea primero se hace un cuadriculado de la imagen de 3×3 y se etiqueta a los parches del borde de la cuadrícula. Como se muestra en la fig. 3.8, las etiquetas de los parches representan su posición relativa al parche del centro. Posteriormente se usa un par de estos parches como dato de entrada para la red neuronal, dicho par debe constar siempre del parche del centro y cualquier parche del borde y la etiqueta del par es la etiqueta asociada al parche del borde seleccionado. Por lo tanto, la tarea de predecir el *relative patch location* es una problema de clasificación con ocho clases.

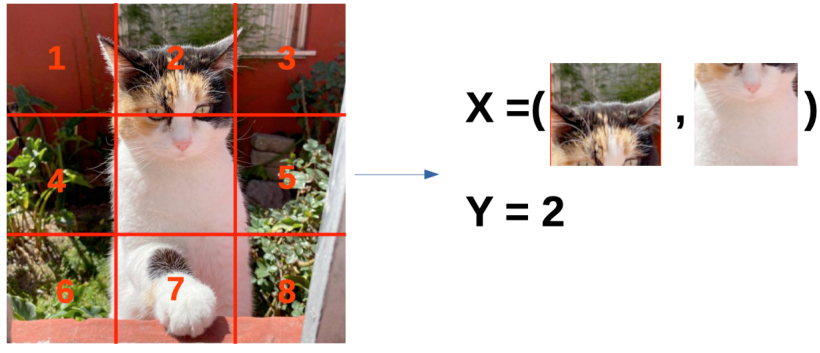

Figura 3.8: Autoetiquetado usando *relative patch location* para la tarea de pretexto.

Hasta el momento no hemos discutido el caso en que una red neuronal convolucional tiene por entrada dos imágenes, los detalles sobre esta implementación se darán en el siguiente capítulo.

3.5.2. *Jigsaw*

El proceso para realizar esta tarea pretexto es el siguiente. Primero se divide la imagen con una cuadrícula de 3, de forma idéntica al del *relative patch location*. Posteriormente se desordena la imagen por cierta permutación, es decir, si el orden original de la imagen es $(1, 2, 3, 4, 5, 6, 7, 8, 9)$ una posible permutación sería $(9, 8, 7, 6, 5, 4, 3, 2, 1)$ (fig. 3.9). Después se entrena a la red neuronal para que prediga la permutación por la cual fue desordenada.

Figura 3.9: Autoetiquetado usando *relative patch location* para la tarea de pretexto.

Existen aún varias sutilezas del método que parecen quedar abiertas en este momento, por ejemplo, puesto que le número total de permutaciones posibles sería 9!, cabe hacer las preguntas ¿En la práctica, se usan todas las permutaciones posibles?, ¿Cómo se ve afectado el modelo en función del subconjunto de permutaciones empleado para el entrenamiento?, ¿Cómo sería la arquitectura de una red que puede resolver este problema?, etc. Puesto que la resolución de estas cuestiones se ve reflejada en la implementación real, abordaremos estos detalles en el siguiente capítulo, donde detallaremos tanto los experimentos como la implementación de los mismos.

Capítulo 4

Propuesta

Antes de describir los experimentos realizados, hagamos una breve recapitulación del contexto en el cual estamos trabajando. La clasificación de galaxias es un problema importante en el área de la astronomía. Clasificar galaxias es un proceso costoso, ya que una persona necesita ser experto en esta área de estudio para poder hacerlo. También, los proyectos como el SDSS han provocado que el flujo de imágenes de galaxias que llegan a los astrónomos sea tan grande que la tarea de clasificar estas imágenes sea imposible para los investigadores. Una posible salida a esta situación es la de automatizar la clasificación de las imágenes de galaxias, siendo el uso de redes neuronales convolucionales la herramienta más recurrente en la actualidad para la realización de esta tarea. No obstante, nos encontramos en un escenario en el cual se tienen pocos datos etiquetados y muchos no etiquetados.

Además, como mencionamos en la sección 3.5, existen técnicas de aprendizaje autosupervisado para la extracción de representaciones visuales, las cuales son un subconjunto de técnicas del aprendizaje no supervisado que permiten aprender a extraer características de alto nivel de las imágenes no etiquetadas.

Bajo este marco, nuestra propuesta es implementar las técnicas de aprendizaje autosupervisado sobre imágenes de galaxias no etiquetadas (a esta etapa la llamamos preentrenamiento autosupervisado) para después usar los pesos y sesgos aprendidos en esas tareas para realizar la transferencia de conocimiento a la tarea de clasificación de galaxias de forma supervisada. Se espera que los pesos y sesgos aprendidos en la etapa autosupervisada nos ayuden a mejorar el rendimiento del modelo para la tarea supervisada, en

comparación a cuando el aprendizaje de la red se lleva a cabo solo de forma supervisada. De forma más especifica, tenemos la expectativa de que, puesto que los modelos serán expuestos a muchas más imágenes de galaxias usando el preentrenamiento autosupervisado, la calidad de la representación aprendida por el modelo sea mejor a la aprendida por el camino puramente supervisado, y por lo tanto sean menos susceptibles de ser afectados por la poca cantidad y el desbalance de clases presente en los datos disponibles para la etapa supervisada.

Cabe señalar que para el momento en que se llevo a cabo la mayoría de la investigación aquí presentada, aún no se habían publicados marcos de trabajo semisupervisados, en los cuales se incluyeran en conjunto técnicas de aprendizaje supervisado y autosupervisado.

Asimismo, realizamos experimentos puramente supervisados, aplicando una serie de técnicas que, aunque no son propias del estado del arte, nos permitirán conocer más sobre la naturaleza de nuestro problema en particular.

A lo largo de este capítulo nos dedicaremos exclusivamente a describir detalladamente cómo se llevaron a cabo los experimentos que acabamos de esbozar.

4.1. Bases de datos

Las bases de datos con las que contamos para realizar los experimentos son las siguientes:

- **Catálogo de Nair et al.**: Contamos con una base datos que consta de 13,711 galaxias que fueron clasificadas por Nair y Abraham [3]. Estas están clasificadas según la clasificación conocida como *T-type*, la cual consta de 17 clases; las etiquetas que usan para referirse a cada una de estas 17 clases son -5, -3, -2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 y 99. El significado de estas etiquetas, en orden, se puede entender en el mismo orden que la secuencia de Hubble que vimos en el capítulo 1, solo que no se considera la estructura de barra, es decir en vez de considerarse dos ramas, en la fig. 1.1, las cuales son caracterizadas por la presencia y ausencia de la estructura de barra, estas dos ramas se aplastan en solo una. Algo que cabe mencionar, es que los valores menores o iguales que cero, corresponden a galaxias esferoidales, mientras que los valores mayores que cero corresponden a galaxias espirales.

  Para los experimentos que realizamos retiramos las galaxias clasificadas en las clases 11, 12 y 99, puesto que en el artículo en el cual reportan el catálogo original [3], no

se define claramente el significado real de las clases 11 y 12 y debido a que la clase 99 se refiere a las galaxias a las cuales no se pudo dar alguna clasificación. Así, el subconjunto final del catálogo de Nair [3] con el que se realizan los experimentos consta de 13,374 galaxias clasificadas en 14 clases.

Una vez hecho este filtro, se hizo una reparametrización de las clases, de tal forma que solo nos quedáramos con 5 clases, dicha reparametrización se hizo agrupando las clases en los siguientes conjuntos: $(-5,-3)$, $(-2,0)$, $(1,2,3)$, $(4,5,6)$ y $(7,8,9,10)$.

Para la realización de los experimentos, este conjunto de datos final fue dividido en los conjuntos de entrenamiento, validación y prueba, los cuales constan del 70 %, 15 % y 15 % de los datos totales respectivamente. En las figuras podemos ver la distribución de clases en cada uno de los subconjuntos. Como podemos ver existe una desbalance claro, se tienen muy pocos ejemplos de la clase 4. En específico se tienen solo 36 ejemplos de la clase 4 mientras que el resto de las clases tienen al menos 316 ejemplos. El efecto de este desbalance de clases lo podremos ver en el siguiente capítulo cuando presentemos los resultados.

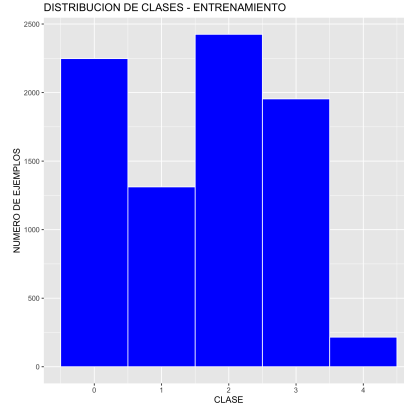

Figura 4.1: Distribución de clases en el conjunto de entrenamiento.

- **Catálogo *Nasa-Sloan-Atlas* (NSA)**: La base de datos de *Nasa-Sloan-Atlas* [36], contiene todas las galaxias vistas por el proyecto SDSS. Esta base de datos no provee de etiquetas a las galaxias. Se tomó un subconjunto de esta base de datos de 574, 406 galaxias para realizar los experimentos autosupervisados. Cabe mencionar que las galaxias de la base de datos de Nair [3] se encuentran contenidas en esta base de datos, por lo tanto nos aseguramos que ninguna galaxia del conjunto de validación

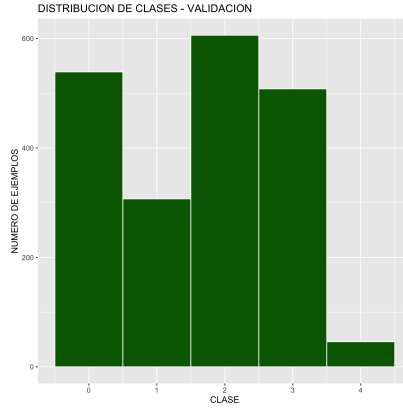

Figura 4.2: Distribución de clases en el conjunto de validacion.

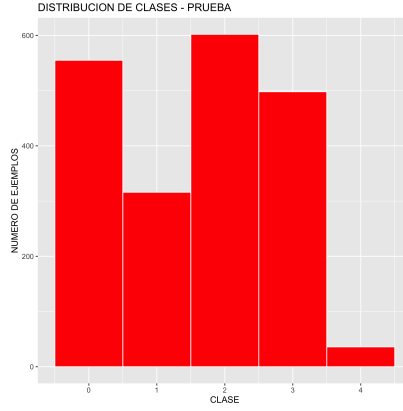

Figura 4.3: Distribución de clases en el conjunto de prueba.

y prueba de la base de datos de Nair [3] fueran incluidas entre las 574, 406 galaxias tomadas de la base de datos del NSA. Los datos fueron finalmente divididos en un conjunto de entrenamiento y validación, constando del 70 % y 30 % de los datos respectivamente.

Cabe resaltar que las bases de datos no constan de las imágenes como tal, sino que estas contienen las coordenadas en el cielo de cada galaxia. Para obtener las imágenes, se hizo una búsqueda de estas usando sus coordenadas astronómicas, en la base de datos del SDSS. También se considero el tamaño aparente de cada galaxia, para que todas

las galaxias ocuparan alrededor del 80 % de la imagen. Aunque el SDSS tiene su propio conjunto de imágenes, las imágenes fueron extraídas de otro proyecto llamado DESI (*Dark Energy Spectroscopic Instrument*) [37], dado que sus imágenes son de mejor calidad.

En la 4.4 se puede ver una pequeña muestra de las imágenes empleadas.

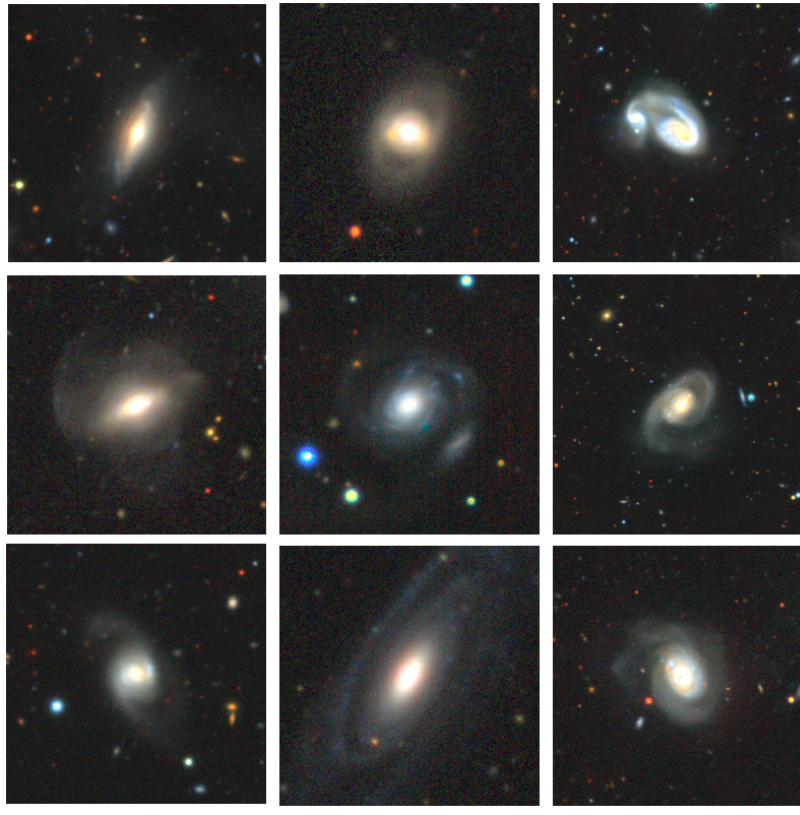

Figura 4.4: Pequeña muestra de las imágenes empleadas.

4.2. Experimentos de aprendizaje autosupervisado

4.2.1. Experimento supervisado de control

Para poder saber qué tan buenos o malos son los resultados obtenidos al usar el preentrenamiento autosupervisado, necesitamos tener un experimento de referencia, este

experimento de referencia simplemente será un modelo entrenado solo de forma supervisada con los datos de Nair [3]. Es importante aclarar que el modelo que proponemos para la tarea de clasificación debe ser tal que nos permita realizar la transferencia de conocimiento adquirido en el preentrenamiento autosupervisado.

Para nuestros experimentos, tanto los supervisados como los autosupervisados, y arquitecturas inspiradas en AlexNet [23]. Podemos ver en la fig. 4.5 la especificación de la arquitectura que empleamos para los experimentos supervisados. Que el valor del *padding* sea igual a *valid* significa que no se aplicó ningún tipo de *padding* y que su valor sea *same* significa que se realizó el *padding* de tal forma que la salida de esa capa tuviera las mismas dimensiones (solo alto y ancho) que la entrada.

ENTRADA
Conv(filters=96, kernel_size=(11,11), strides=(3,3), padding = same)
LRN
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Conv(filters=384, kernel_size=(5,5), strides=(2,2), padding=same)
LRN
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Conv(filters=384, kernel_size=(3,3), strides=(1,1), padding=same)
Conv(filters=384, kernel_size=(3,3), strides=(1,1), padding=same)
Conv(filters=256, kernel_size=(3,3), strides=(1,1), padding=same)
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Dense(4096)
Dropout(0.5)
Dense(4096)
Dropout(0.5)
Dense(1000)
Dense(14, activation = softmax)
SALIDA

Figura 4.5: Especificación de la arquitectura para los experimentos supervisados.

También es necesario hablar sobre el preprocesamiento que se hizo sobre las imágenes de entrada. Se realizaron algunas operaciones de acrecentamiento de datos sobre las imágenes. A continuación mostramos el preprocesamiento por el cual pasan todas las imágenes que se usaron para el entrenamiento antes de entrar a la red:

1. Se volteo la imagen de forma aleatoria respecto al eje horizontal y al eje vertical.
2. Se rotaron las imágenes por un ángulo aleatorio.
3. Se hizo un recorte aleatorio de la imagen de entre el 50 % y el 100 % de la imagen.
4. Se ajustó el tamaño de la imagen a las dimensiones de 330×330, que es un tamaño de imagen usual para usar con redes neuronales convolucionales.

Podemos ver un ejemplo de este proceso en la fig. 4.6. Además, el preprocesamiento que se le dio a las imágenes en la etapa de validación y prueba es la siguiente:

1. Se hizo un recorte respecto al centro de la imagen del 90 %.
2. Se ajustó el tamaño de la imagen a las dimensiones de 330×330.

Se entrenó el modelo para la tarea de clasificación en cinco clases en la base de datos de Nair [3], durante 300 épocas con una tasa de aprendizaje de 1×10^{-5} usando el optimizador Adam. A lo largo del entrenamiento se monitoreó el rendimiento del modelo sobre el conjunto de validación. Se eligió como modelo final al modelo correspondiente a la época en la cual se obtuvo el mejor rendimiento en el conjunto de validación.

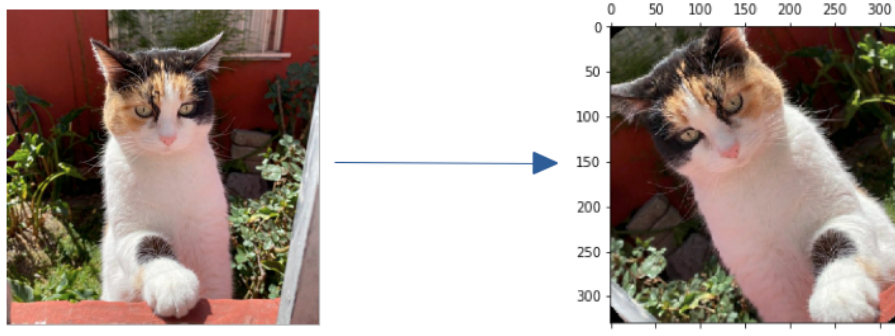

Figura 4.6: Aumentado de datos.

4.2.2. Preentrenamiento autosupervisado

En este apartado hacemos explícitos los detalles de la implementación de cada tarea de aprendizaje autosupervisado para el preentrenamiento de los modelos.

Relative patch location

En la seccion 3.5.1 esbozamos la idea detrás de la tarea de *relative patch location*, a la cual nos referiremos también como RPL, como vimos, debemos crear, a partir de la imagen, una cuadrícula de 3×3 y entrenar un modelo para que prediga la posición relativa de los parches del borde de la cuadrícula respecto al parche del centro.

Discutamos primero cómo es el preprocesamiento de las imágenes antes de entrar a la red convolucional. A continuación enlistaremos en orden los procedimientos a través de los cuales pasan las imágenes. Esta lista incluye tanto el de aumentado de datos como la generación del par de parches y etiqueta para realizar la tarea de *relative patch location*:

1. Se carga la imagen en su tamaño original de 800×800.
2. Se realizan reflexiones aleatorias, tanto de forma horizontal como vertical, además se rota la imagen por un múltiplo aleatorio de $90°$.
3. Se hace un recorte central de entre el $33\,\%$ y el $70\,\%$ de la imagen. Esto tiene el fin de que las imágenes finales sean más propensas a mostrar partes del centro de la imagen original, que es donde se encuentra el objeto de interés.
4. Se hace un recorte aleatorio de 360×360 pixeles.
5. Se reemplazaron aleatoriamente dos canales de la imagen por ruido gaussiano con desviación estándar de 0.1.
6. Se crea la cuadrícula de 3×3, creando un total de 9 parches (*patches*) de tamaño 120×120.
7. Se genera un número aleatorio entero n entre el 0 y el 8, el cual servirá como etiqueta para nuestro par de parches.
8. Se selecciona el parche del centro y el parche correspondiente a la etiqueta generada, según la fig. 3.8.
9. Se extrae una subregión de cada parche de tamaño 96×96.
10. Se pasa a la red el par de parches y el número n como etiqueta asociada al par.

Estos pasos son lo más parecido posible a los dados por Doersch en [21]. El punto 9, el de extraer una subregión de los parches, se implementó para crear una brecha entre los parches. Esto tiene la finalidad de eliminar la continuidad de la imagen en los bordes de los parches, puesto que la red podría aprender a diferenciar las distintas posiciones de los parches simplemente fijándose en esta continuidad sin tener que aprender características de alto nivel de las imágenes [20, 21].

Por otro lado el punto 5, en el cual se reemplazan dos canales de color de las imágenes se implementó para evitar que la red aprenda a diferenciar la posición de los parches

usando la aberración cromática en las imágenes [21]. La aberración cromática es un fenómeno óptico el cual provoca que en algunos puntos de la imagen se capturen más en unos colores que en otros. La red puede usar este atributo en las fotos como atajo para reconocer la posición de los parches. Puesto que existen trabajos similares donde no se considera el efecto de la aberración cromática [20], también se hizo un experimento donde se elimina el paso 5 del procesamiento de datos.

Hablemos ahora de la arquitectura que empleamos para realizar la tarea del *relative patch location*. Como mencionamos anteriormente, tanto los supervisados como los autosupervisados, emplean arquitecturas inspiradas en AlexNet [23]. Esto debido a que los artículos en los cuales se trata de forma individual a cada una de las técnicas de autosupervisión [20, 21], y por tanto las que tratan de forma más extensiva cada método, usan arquitecturas inspiradas en AlexNet. También se tomó esta decisión en favor de tratar de entender lo mejor posible todos los factores involucrados en los experimentos, siendo AlexNet una red mucho más sencilla de tratar y entender que sus parientes más recientes como ResNet.

La arquitectura que empleamos es una red siamesa, que procesa los dos parches con un mismo bloque convolucional, el cual se comparte entre los parches, y después se juntan las salidas correspondientes a las imágenes para pasar por un conjunto de capas completamente conexas para finalmente pasar a la capa de clasificación. En la fig. 4.7 podemos ver la configuración del bloque convolucional. El lector podrá notar que esta incluye también una capa completamente conexa al final, sin embargo, insistimos en llamarla bloque convolucional puesto que esta parte contiene la totalidad de capas convolucionales de la arquitectura. En la fig. 4.8 podemos ver la arquitectura completa.

ENTRADA
Conv(filters=96, kernel_size=(11,11), strides=(2,2), padding = same)
LRN
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Conv(filters=384, kernel_size=(5,5), strides=(2,2), padding=same)
LRN
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Conv(filters=384, kernel_size=(3,3), strides=(1,1), padding=same)
Conv(filters=384, kernel_size=(3,3), strides=(1,1), padding=same)
Conv(filters=256, kernel_size=(3,3), strides=(1,1), padding=same)
MaxPooling(pool_size=(3,3), strides=(2,2), padding=valid)
Dense(4096)
Dropout(0.5)
SALIDA

Figura 4.7: Bloque convolucional.

El bloque convolucional generará dos salidas, una por cada elemento del par generado, antes de entrar a la siguiente parte de la red se promedian entrada a entrada ambas salidas para generar una salida con el mismo tamaño que las anteriores.

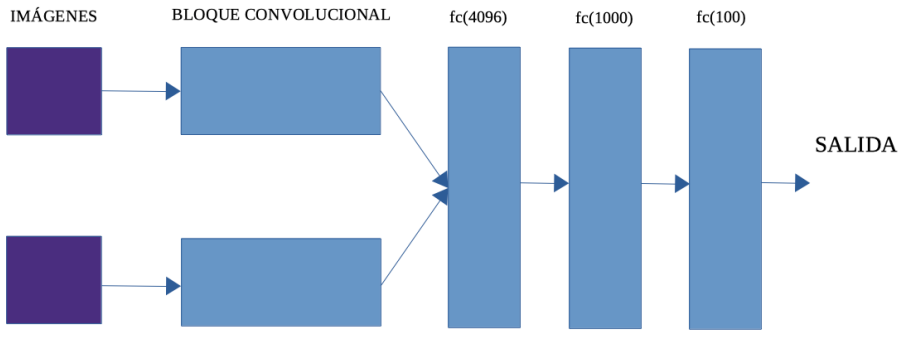

Figura 4.8: Arquitectura para la tarea de RPL.

Entrenamos este modelo por alrededor de 400 épocas para el caso en el cual se ocupan todos los canales de color de las imágenes; para el caso en el que se eliminan dos canales de color por cuestiones de tiempo solo fue posible entrenarlo por 126 épocas, se usó una tasa de aprendizaje de 1×10^{-6} usando el optimizador Adam. A lo largo del entrenamiento se monitoreó el rendimiento del modelo sobre el conjunto de validación. Se eligió como modelo final el modelo correspondiente a la época en la cual se obtuvo el mejor rendimiento en el conjunto de validación.

Jigsaw

Resumiendo lo visto en la sección 3.5.2, para realizar la tarea de *jigsaw* debemos crear una cuadrícula de la imagen dada, de la misma forma en la cual cuadriculamos las imágenes en el *relative patch location*, y reordenar los parches de la cuadrícula según una permutación dada. La red que entrenemos para resolver el *jigsaw*, tendrá que ser capaz de identificar la permutación con la cual fueron reordenados los parches de la cuadrícula.

En la implementación real no se ocupan todas las permutaciones posibles; si tenemos cuadrículas de 3×3, tendríamos que ocupar un total de $9! = 362880$ permutaciones. Siguiendo la discusión hecha por Noroozi en [20], ocuparemos dos conjuntos de permutaciones, uno de 50 permutaciones y otro de 100 permutaciones. De esta forma, el problema de *jigsaw* se vuelve un problema de clasificación de 100 y 50 clases.

El problema de cómo escoger esos subconjuntos de permutaciones aparece ahora. Para entender el algoritmo a partir del cual fue escogido dicho subconjunto debemos entender primero un concepto llamado distancia de Hamming. Dadas dos permutaciones, las cuales representaremos de la siguiente forma $\boldsymbol{x} = (x_1, ..., x_9)$ y $\boldsymbol{y} = (y_1, ..., y_9)$, donde $\boldsymbol{x}$ y $\boldsymbol{y}$ son permutaciones de $(1, 2, 3, 4, 5, 6, 7, 8, 9)$, definimos la distancia de Hamming [38] entre $\boldsymbol{x}$ y $\boldsymbol{y}$ como

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \delta(x_i, y_i) \tag{4.1}$$

donde

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{si} \quad x_i = y_i, \\ 1 & \text{si} \quad x_i \neq y_i \end{cases} \tag{4.2}$$

La idea detrás de aplicar esta función a cada par de permutaciones es el saber qué tanto se parecen entre sí. Entre mayor sea $d_H(\boldsymbol{x}, \boldsymbol{y})$ menos se parecerán entre sí ambas permutaciones. Esto es cierto de forma intuitiva, puesto que la distancia de Hamming entre dos permutaciones simplemente es el número de entradas en las cuales no coinciden dichas permutaciones. Demos un ejemplo de esto, calculemos la distancia de Hamming entre las permutaciones $\boldsymbol{x} = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ y $\boldsymbol{y} = (1, 3, 2, 6, 5, 4, 8, 7, 9)$

$$\begin{aligned} d_H(\boldsymbol{x}, \boldsymbol{y}) =& \delta(1,1) + \delta(2,3) + \delta(3,2) \\ & \delta(4,6) + \delta(5,5) + \delta(6,4) \\ & \delta(7,8) + \delta(8,7) + \delta(9,9) \\ =& 1 + 0 + 0 \\ & 0 + 1 + 0 \\ & 0 + 0 + 1 \\ =& 3. \end{aligned} \tag{4.3}$$

En [20], Noroozi menciona que es conveniente que los elementos del subconjunto de permutaciones, que se ocupan para resolver la tarea del *jigsaw*, se parezcan lo menos posible entre sí. Esto evita que la tarea sea ambigua, entre más se parezcan dos permutaciones entre sí más ambigua será la tarea de distinguir una de la otra. Esto se logra

maximizando la distancia de Hamming entre dichos elementos. En [20], Noroozi propone el siguiente algoritmo para obtener un subconjunto de permutaciones que cumpla las características mencionadas

Algorithm 1 Cálculo del conjunto maximal de permutaciones

procedure MAXPERMSET(n) ▷ n es el tamaño del conjunto a calcular
  $\bar{P} \leftarrow$ todas las permutaciones $[\bar{P}_1, ..., \bar{P}_{9!}]$ ▷ $\bar{P}$ es una matriz de $9 \times 9!$
  $P \leftarrow \emptyset$ ▷ Conjunto maximal de permutaciones
  $j \sim \mathcal{U}[1, 9!]$ ▷ Se obtiene un índice aleatorio entre 1 y 9!
  $counter \leftarrow 1$
  while $i \leq n$ **do**
    $P \leftarrow [P, \bar{P}_j]$ ▷ Agrega la j-ésima permutación a P
    $\bar{P} \leftarrow [\bar{P}_1, ..., \bar{P}_{j-1}, \bar{P}_{j+1}, ..., \bar{P}_{9!}]$ ▷ Remueve $\bar{P}_j$ de $\bar{P}$
    $D \leftarrow d_H(P, \bar{P})$ ▷ D es una matriz de $i \times (9! - i)$
    $\bar{D} \leftarrow \mathbf{1}^T D$ ▷ $\bar{D}$ es un vector de $9! - i$ entradas
    $j \leftarrow \arg\max_k \bar{D}_k$ ▷ $\bar{D}_k$ es la k-ésima entrada de $\bar{D}$
    $i \leftarrow i + 1$
  end while
  return P
end procedure

Usando este algoritmo se obtuvieron los dos subconjuntos de permutaciones mencionados. Posteriormente, se asocia un número del 0 al 49 y del 0 al 100 a las permutaciones, de tal forma que la tarea impuesta sobre la red no es la de predecir todos los números de la permutación, sino solo la etiqueta asociada a cada permutación.

El procesamiento al cual es sometido cada imagen antes de pasar por la red es el siguiente:

1. Se carga la imagen en su tamaño original de 800×800.

2. Se realizan reflexiones aleatorias, tanto de forma horizontal como vertical, además se rota la imagen por un múltiplo aleatorio de 90°.

3. Se hace un recorte central de entre el 33 % y el 70 % de la imagen. Esto tiene el fin de que las imágenes finales sean más propensas a mostrar partes del centro de la imagen original, que es donde se encuentra el objeto de interés.

4. Se hace un recorte aleatorio de 225×225 pixeles.

5. Se crea la cuadrícula de 3×3, creando un total de 9 parches (*patch*) de tamaño 75×75.

6. Se genera un número entero aleatorio entero entre el 0 y el 8, el cual servirá como etiqueta para nuestro par de parches.

7. Se selecciona aleatoriamente una de las permutaciones disponibles a través de su número asociado n.

8. Se reordenan los parches usando la permutación n.

9. Se extrae una subregión de cada parche de tamaño 65×65.

10. Se pasan a la red el arreglo de parches reordenados y el número n de la permutación como etiqueta.

La red empleada para resolver este problema es bastante parecida a la usada para resolver la tarea del *relative patch location*, solo que en vez de pasar 2 parches procesa 9 el bloque convolucional. Una diferencia importante, con la red para resolver el *relative patch location*, es que no promedia las salidas del bloque convolucional, sino que las concatena. Esta diferencia es sutil, puesto que al promediar se pierde el orden en el cual fueron pasados cada uno de los parches por el bloque convolucional, mientras que al concatenar las salidas se conserva el orden, que es lo que está tratando de predecir la red. En la fig. 4.9 podemos ver el diagrama de la red.

El modelo fue entrenado por 400 épocas usando el optimizador Adam con una tasa de aprendizaje de 5×10^{-7}.

4.2.3. Transferencia de conocimiento & *fine tuning*

Los pesos obtenidos en cada uno de los experimentos autosupervisados fueron reutilizados para la tarea supervisada. Se reutilizaron los pesos de las siguientes dos formas:

- **Transferencia de conocimiento**: Se inicializaron los pesos de las capas convolucionales de la red de la fig. 4.5 con los pesos obtenidos en el preentrenamiento autosupervisado. Posteriormente se entrenaron todos los pesos siguiendo el procedimiento planteado en la sección 4.2.1 para resolver la tarea de clasificación de galaxias sobre la base de datos de Nair [3].

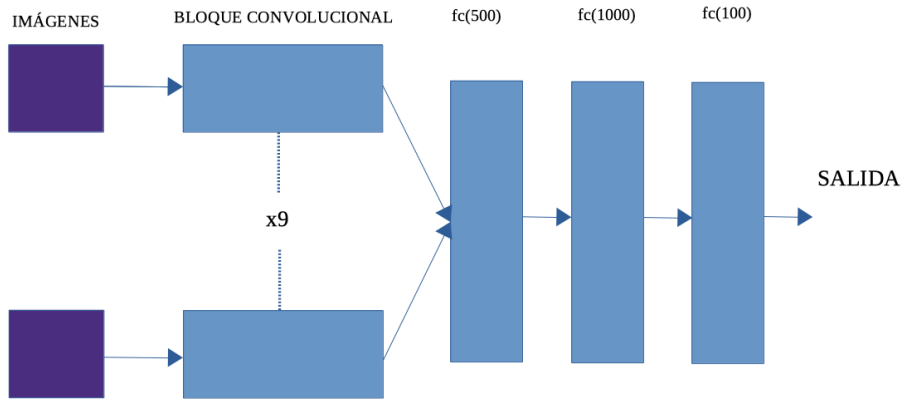

Figura 4.9: Arquitectura para la tarea de *jigsaw*.

- ***Fine Tuning***: Se inicializaron los pesos de las capas convolucionales de la red de la fig. 4.5 con los pesos obtenidos en el preentrenamiento autosupervisado. Posteriormente se congelaron dichas capas, para solo entrenar los pesos de las capas completamente conexas. Finalmente se descongelaron las capas para entrenar todos los pesos. El resto del planteamiento es el mismo descrito en 4.2.1.

4.2.4. Recapitulación

Antes de pasar a la siguiente sección, y para que el lector pueda tener de forma accesible los diferentes experimentos que hemos descrito hasta ahora y cuyos resultados presentaremos más adelante, presentamos en tablas un resumen de los experimentos descritos hasta ahora.

En la tabla 4.1 se encuentran todas las variantes de preentrenamiento autosupervisado. En la tabla 4.2 encontramos los diferentes esquemas a partir de los cuales fueron probados los distintos preentrenamientos autosupervisados para la tarea de clasificación en la base de datos de Nair [3]. Debemos mencionar que la variante de probar los pesos aprendidos con el RPL descartando dos canales de colores usando el esquema de *fine tuning* no fue posible de llevar a cabo por falta de tiempo.

Id	Tarea pretexto	Variación
1	Relative patch location	Realizando la tarea usando todos los canales de color.
2	Relative patch location	Realizando la tarea descartando aleatoriamente dos canales de color para evitar la aberración cromática.
3	Jigsaw	Usando 50 permutaciones.
4	Jigsaw	Usando 100 permutaciones

Tabla 4.1: Variantes de preentrenamiento autosupervisado.

Id preentrenamiento	Descripción preentrenamiento	Esquema bajo el cual fueron usados los pesos preentrenados
1	RPL usando todos los canales de color.	Transferencia de conocimiento.
1	RPL usando todos los canales de color.	Fine tuning.
2	RPL descartando dos canales de color.	Transferencia de conocimiento.
3	Jigsaw con 50 permutaciones.	Transferencia de conocimiento.
3	Jigsaw con 50 permutaciones.	Fine tuning.
4	Jigsaw con 100 permutaciones.	Transferencia de conocimiento.
4	Jigsaw con 100 permutaciones.	Fine tuning.

Tabla 4.2: Experimentos supervisados usando los pesos obtenidos en el preentrenamiento autosupervisado.

Conjuto de entrenamiento	Conjunto de validación
Balanceado	Sin balancear
Balanceado	Balanceado

Tabla 4.3: Experimentos puramente supervisados.

4.3. Experimentos supervisados

Cómo mencionamos anteriormente, una de las principales complicaciones que afectan a los modelos al tratar de resolver el problema de clasificación de galaxias es la del desbalance de clases. A diferencia los experimentos hasta ahora descritos, que buscan atacar este problema a través de técnicas de aprendizaje autosupervisado, lo cual podría considerarse como una forma muy sofisticada de atacar el problema, los experimentos que describiremos a continuación tratan de mejorar el rendimiento de la tarea de clasificación de galaxias de una forma más conservadora.

Entrenamos una ResNet50 por épocas usando el optimizados Adam con una tasa de aprendizaje para realizar la tarea de clasificación de galaxias en la base de datos de Nair [3]. Sin embargo, el detalle más importante y que es la herramienta principal que ocupa este experimento para mitigar el problema del desbalance de las clases, es la técnica conocida como *up-sampling*, la cual consiste en duplicar de forma aleatoria elementos de las clases no mayoritarias hasta que igualen la cantidad de la clase de la cual se tengan más muestras.

Se aplicó el método de *up-sampling* al conjunto de entrenamiento. Aquí se separan dos experimentos, uno en el cual se monitorea el rendimiento del modelo durante el entrenamiento con el conjunto de validación normal y otro en el cual se ocupó un conjunto de validación balanceado para monitorear el rendimiento del modelo. El procesamiento al que son sometidas las imágenes es el mismo descrito en 4.2.1, tanto para las imágenes de entrenamiento como para las imágenes de validación. Con la pequeña modificación de que a las imágenes de validación se les hace un corte central del 70 % en vez del 90 %. Podemos ver resumidos los experimentos supervisados en la tabla 4.3.

Capítulo 5

Resultados

En este capítulo presentaremos los resultados de los experimentos descritos en el capítulo pasado. Organizaremos la presentación de los resultados en el mismo orden que apareció cada experimento en el capítulo anterior.

5.1. Experimentos de aprendizaje autosupervisado

5.1.1. Experimento supervisado de control

El experimento supervisado de control obtuvo el siguiente rendimiento en el conjunto de prueba:

- Exactitud: 69.9551 %
- Pérdida: 0.7299

También presentamos en la fig. 5.1 la matriz de confusión del modelo sobre el conjunto de prueba. Como podemos ver en la matriz de confusión, el modelo solo fue capaz de clasificar 28 % de las galaxias de la cuarta clase en el conjunto de validación, eso significa que solo pudo clasificar correctamente una galaxia de las 36 de esa clase.

Si nuestra hipótesis fuera correcta, el preentrenamiento autosupervisado debería ayudar a mejorar ya sea la exactitud de este modelo o el porcentaje de galaxias en la clase 4 clasificadas correctamente.

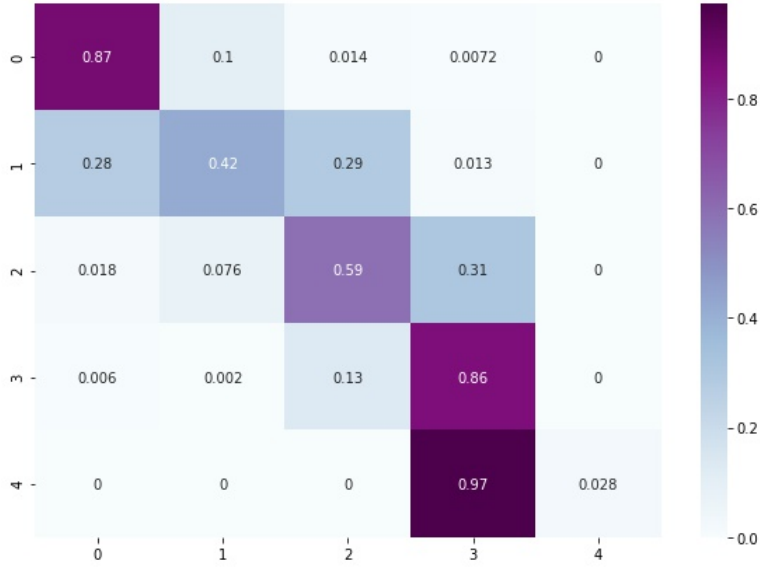

Figura 5.1: Matriz de confusión del experimento supervisado de control.

5.1.2. Preentrenamiento autosupervisado

Aquí presentamos los resultados de los modelos entrenados para resolver las tareas pretexto descritas el capítulo pasado. Los rendimientos presentados a continuación refieren a qué tan bien los modelos pudieron resolver las tareas de aprendizaje autosupervisado. Puesto que nuestro interés específico no es resolver estas tareas, sino más bien usarlas como un medio para mejorar el rendimiento en otra tarea, no hicimos uso de un conjunto de prueba. En cambio, presentamos los rendimientos en los conjuntos de validación y entrenamiento, además de la matriz de confusión en el conjunto de validación en algunos casos.

Relative patch location

Como mencionamos en 4.2.2, implementamos dos versiones del modelo para resolver la tarea de RPL, uno en el cual se dejaban todos los canales de colores de las imágenes y otro en el cual, para evitar la posible presencia de aberraciones cromáticas en las imágenes, se sustituían aleatoriamente dos canales de las imágenes por ruido gaussiano.

Los resultados de dichos experimentos fueron los siguientes:

- **Sin eliminar ningún canal de color:**
  - Exactitud en el conjunto de entrenamiento: 52.6103 %
  - Pérdida en el conjunto de entrenamiento: 1.3586
  - Exactitud en el conjunto de validación: 52.8481 %
  - Pérdida en el conjunto de validación: 1.3561

  Como mencionamos en la descripción del experimento, RPL es una tarea de clasificación en ocho clases, por lo tanto el obtener más del 52 % de exactitud es una señal de que el modelo aprendió a resolver la tarea con relativo éxito; si un modelo calificara las imágenes de forma aleatoria y uniformemente en todas las clases obtendríamos una exactitud del 12.5 %. Para tener una mejor idea de cómo realizó la tarea el modelo en la fig. 5.2 podemos ver la matriz de confusión sobre el conjunto de validación.

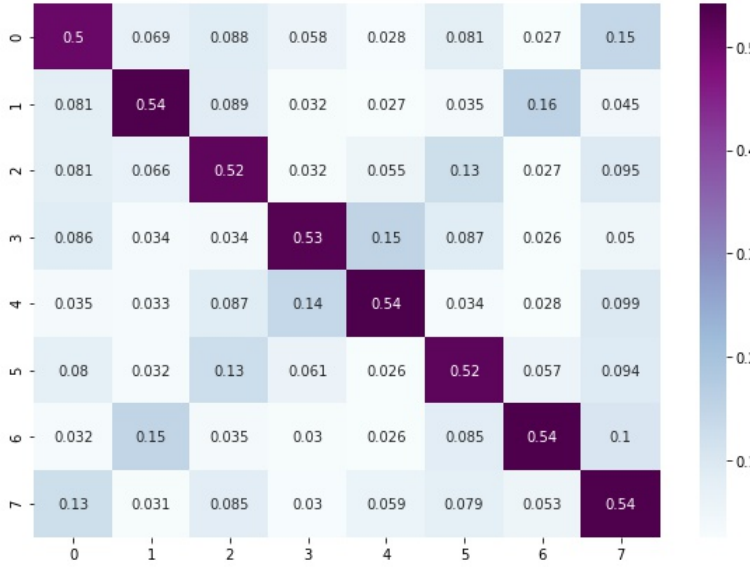

Figura 5.2: Matriz de confusión de la tarea de RPL.

Como podemos ver, al presentarse los colores más oscuros sobre la diagonal de la matriz, podemos afirmar que la tarea de RPL se realizó con éxito.

- **Eliminando dos canales de color de forma aleatoria:**

  - Exactitud en el conjunto de entrenamiento: 50.9157
  - Pérdida en el conjunto de entrenamiento: 1.2771
  - Exactitud en el conjunto de validación: 51.0248
  - Pérdida en el conjunto de validación: 1.2754

Observemos que tanto la exactitud como la pérdida de este modelo son bastante similares a las del experimento en el cual se usaron todos los canales de color. Presentamos en la fig. 5.3 la matriz de confusión también para este experimento. Podemos notar que el comportamiento presentado por esta matriz de confusión es bastante parecido al del experimento pasado, por lo tanto, podemos decir que la tarea de RPL eliminando aleatoriamente dos canales de colores también se logró de forma exitosa.

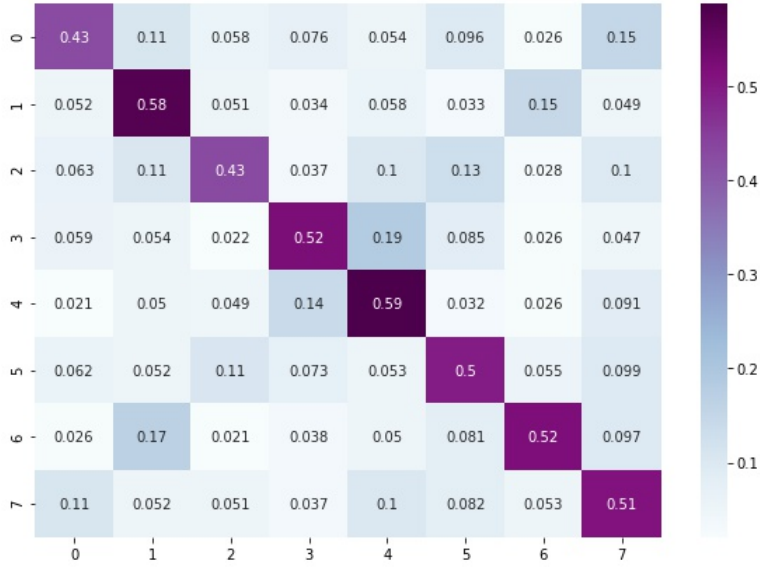

Figura 5.3: Matriz de confusión de la tarea de RPL eliminando dos canales de color.

Jigsaw

Procedemos ahora a presentar los resultados de los modelos obtenidos para resolver la tarea de *jigsaw*. Como mencionamos en la sección 4.2.2, realizamos dos variaciones del experimento, uno en el cual usábamos un subconjunto de 50 permutaciones y otro en el cual usábamos un subconjunto de 100 permutaciones. Procedemos a presentar los resultados de estos dos modelos:

- **50 clases:**
  - Exactitud en el conjunto de entrenamiento: 95.5235
  - Pérdida en el conjunto de entrenamiento: 0.6837
  - Exactitud en el conjunto de validación: 95.5292
  - Pérdida en el conjunto de validación: 0.6829
- **100 clases:**
  - Exactitud en el conjunto de entrenamiento: 93.9410
  - Pérdida en el conjunto de entrenamiento: 0.8742
  - Exactitud en el conjunto de validación: 93.9630
  - Pérdida en el conjunto de validación: 0.8720

Como podemos observar, la exactitud de ambos modelos cae por encima del 90 %; notemos que si cada modelo clasificara las imágenes de forma aleatoria e uniforme sobre todas las clases obtendríamos 2 % y en el otro 1 % de exactitud respectivamente. Por lo tanto, podemos afirmar que los modelos para resolver el problema de *jigsaw* lograron realizar esta tarea de forma exitosa.

Antes de pasar a la siguiente sección, presentamos en la tabla 5.1 los resultados obtenidos.

5.1.3. Transferencia de conocimiento & *fine tuning*

Ahora presentamos los resultados derivados de usar los pesos aprendidos por los modelos en las tareas de aprendizaje autosupervisado para la tarea supervisada de clasificación de galaxias en la base de datos de Nair [3]. Como mencionamos en 4.2.3, los pesos

Id	Tarea pretexto	Variación	Exactitud en la tarea pretexto.
1	Relative patch location	Realizando la tarea usando todos los canales de color.	52.84 %
2	Relative patch location	Realizando la tarea descartando aleatoriamente dos canales de color para evitar la aberración cromática.	51.02 %
3	*Jigsaw*	Usando 50 permutaciones.	95.52 %
4	*Jigsaw*	Usando 100 permutaciones	93.96 %

Tabla 5.1: Resumen de resultados del preentrenamiento autosupervisado.

aprendidos en el preentrenamiento autosupervisado se ocuparon para la tarea supervisada bajo dos esquemas, transferencia de conocimiento y *fine tuning*. Para cada uno de los experimentos de la sección pasada presentamos los resultados derivados de usar sus pesos bajo éstos dos esquemas. Cabe aclarar que, por cuestión de tiempo, no fue posible implementar el experimento que usa los pesos obtenidos de la tarea de RPL eliminando dos canales de colores bajo el esquema de *fine tuning*.

Los resultados fueron los siguientes:

- **Jigsaw con 50 clases usando transferencia de conocimiento**
  - Exactitud: 68.5600 %
  - Pérdida : 0.9121

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.4. Este modelo no mejora la exactitud del modelo supervisado de control, por el contrario es peor. Sin embargo, mejora el rendimiento sobre la clase 4, logrando clasificar correctamente el 11 % de los ejemplos de la clase 4 en el conjunto de validación (4 galaxias de las 36 disponibles en la clase).

- **Jigsaw con 50 usando *fine tuning***
  - Exactitud: 62.7304 %
  - Pérdida : 175.7218

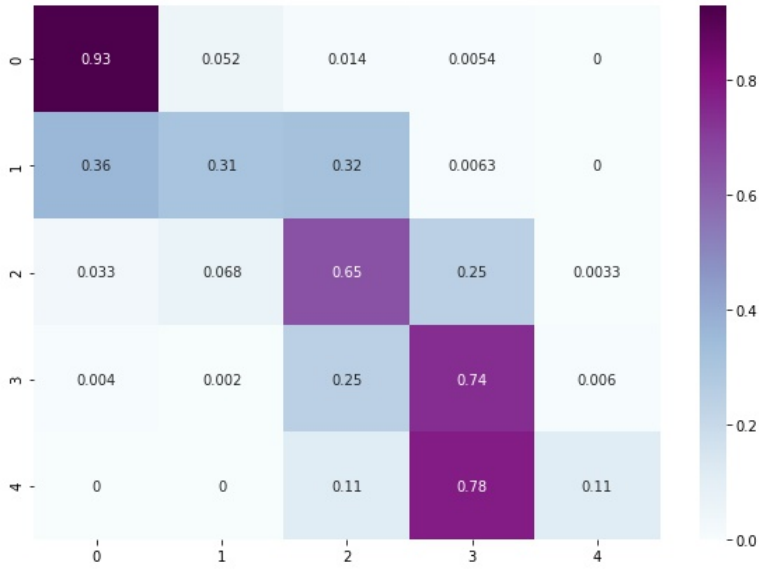

Figura 5.4: Matriz del modelo preentrenado con la tarea de *jigsaw* con 50 clases y usando transferencia de conocimiento.

La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.5. Como podemos ver, este modelo no mejora ni la exactitud en todo el conjunto de prueba ni dentro de la clase 4, por el contrario no es capaz de clasificar correctamente ninguna de las galaxias de esta clase.

- **Jigsaw con 100 clases usando transferencia de conocimiento**
  - Exactitud: 68.2112 %
  - Pérdida : 0.9083

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.6. Este modelo no mejora la exactitud en todo el conjunto de prueba y tiene la misma exactitud dentro de la clase 4 que el experimento de control.

- **Jigsaw con 100 clases usando *fine tuning***
  - Exactitud: 68.6098 %
  - Pérdida : 1.2679

Figura 5.5: Matriz de confusión del modelo preentrenado con la tarea de *jigsaw* con 50 clases y usando *fine tuning*.

La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.7. Este modelo tampoco mejora la exactitud del experimento de control, ni en todo el conjunto de entrenamiento ni dentro de la clase 4.

- **RPL con transferencia de conocimiento**

  - Exactitud: 58.7443 %
  - Pérdida : 200.6112

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.8. La exactitud de este modelo cayó 10 % respecto a la del experimento de control, además no pudo clasificar correctamente ninguna galaxia de la clase 4.

- **RPL con *fine tuning***

  - Exactitud: 70.1046 %
  - Pérdida : 1.2700

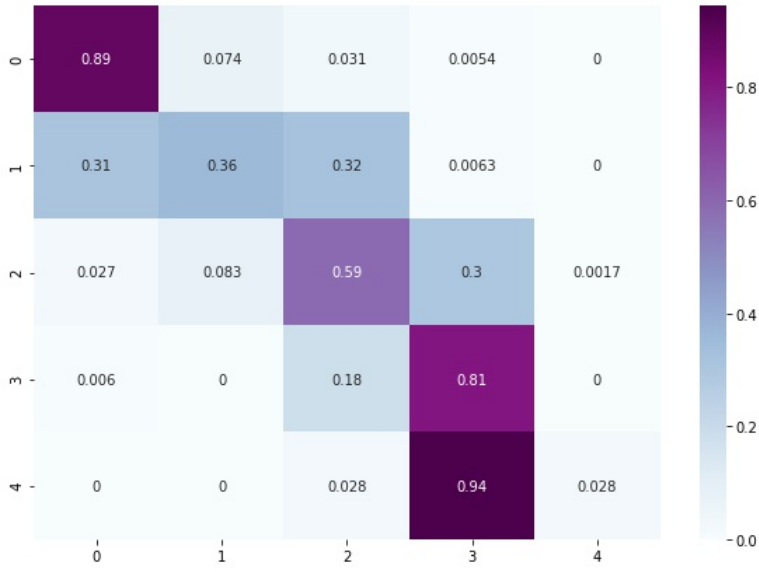

Figura 5.6: Matriz de confusión del modelo preentrenado con la tarea de *jigsaw* con 100 clases y usando transferencia de conocimiento.

La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.9. Como podemos ver, este modelo mejoró el rendimiento del experimento de control, logrando una exactitud de 70.1 %, sin embargo, no logró clasificar correctamente ninguna galaxia de la clase 4.

- **RPL eliminando dos canales de color de forma aleatoria con *transferencia de conocimiento***: Las métricas en el conjunto de prueba son:

  - Exactitud: 60.4882 %
  - Pérdida : 1.1125

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.10. Este modelo tampoco mejora la exactitud sobre el conjunto de prueba, empeorando casi en un 10 % respecto al experimento de control, además no pudo clasificar correctamente ninguna de las galaxias de la clase 4.

Antes de pasar a la siguiente sección resumimos los resultados obtenidos en la tabla 5.2.

Descripción preentrenamiento	Esquema bajo el cual fueron usados los pesos preentrenados	Exactitud total	Exactitud en la clase 4
Sin preentrenamiento (experimento de control)	Ninguno	69.95 %	2.8 %
RPL usando todos los canales de color.	Transferencia de conocimiento.	58.74 %	0 %
RPL usando todos los canales de color.	Fine tuning.	70.1 %	0 %
RPL descartando dos canales de color.	Transferencia de conocimiento.	60.48 %	0 %
Jigsaw con 50 permutaciones.	Transferencia de conocimiento.	68.56 %	11 %
Jigsaw con 50 permutaciones.	Fine tuning.	62.73 %	0 %
Jigsaw con 100 permutaciones.	Transferencia de conocimiento.	68.21 %	2.8 %
Jigsaw con 100 permutaciones.	Fine tuning.	68.6 %	0 %

Tabla 5.2: Resultados de usar los pesos aprendidos en el preentrenamiento autosupervisado para resovler la tarea de clasificación multiclase de galaxias.

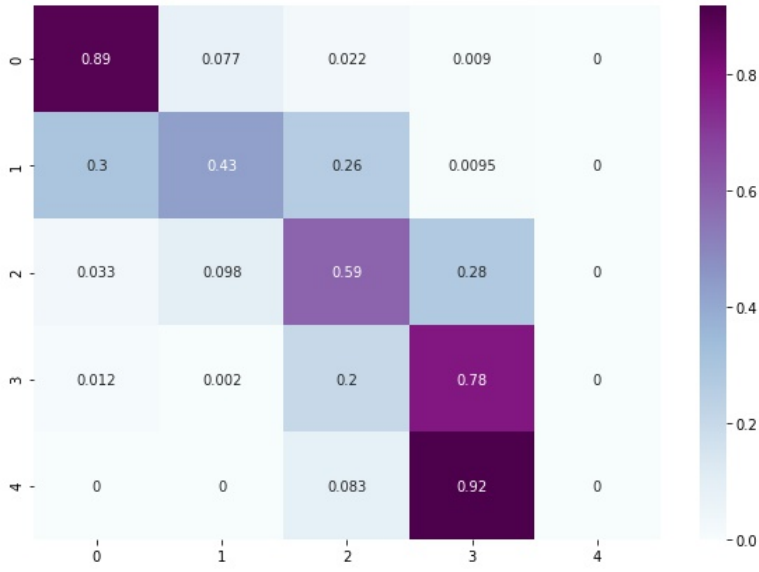

Figura 5.7: Matriz de confusión del modelo preentrenado con la tarea de *jigsaw* con 100 clases y usando *fine tuning*.

5.2. Experimentos supervisados

Los experimentos supervisados, usando la arquitectura ResNet50 y balanceando las clases, obtuvieron los siguientes resultados en el conjunto de prueba:

- **Monitoreando con un conjunto de validación desbalanceado**:
  - Exactitud: 74.6387
  - Pérdida : 0.6307

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.11. Como podemos ver, este modelo obtiene una exactitud de 74.6387 %, siendo el más alto obtenido por los modelos presentados en este trabajo, además logra clasificar un 72 % de las galaxias de la clase 4 correctamente, lo que implica que clasificó correctamente 26 galaxias de las 36 galaxias pertenecientes a la clase 4. Estas métricas muestran una gran mejora respecto a cualquier modelo de la sección anterior.

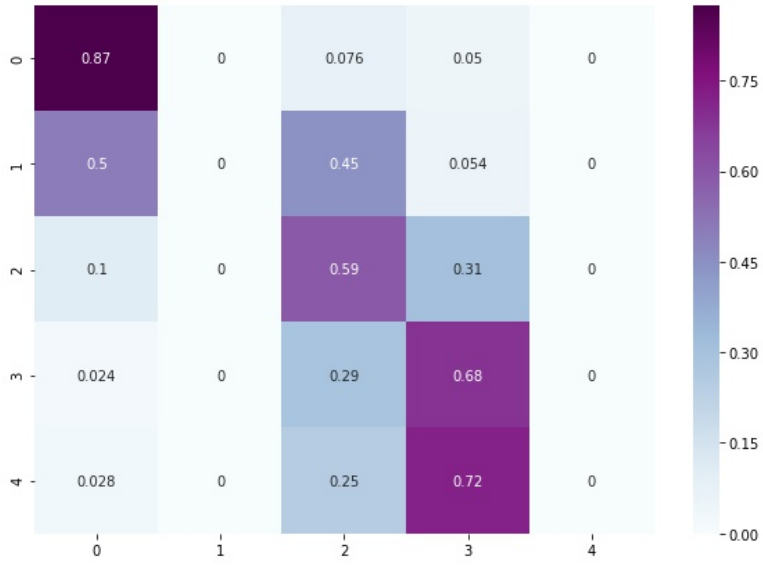

Figura 5.8: Matriz de confusión del modelo preentrenado con la tarea de RPL y usando transferencia de conocimiento.

- **Monitoreando con un conjunto de validación balanceado**:

  - Exactitud: 69.3074
  - Pérdida : 0.7080

  La matriz de confusión para este modelo en el conjunto de prueba puede verse en la fig. 5.12. Aquí podemos notar que la exactitud del modelo sobre todo el conjunto de prueba fue menor que el obtenido en el experimento de control de la sección pasada, sin embargo, algo que es bastante destacable es que este modelo fue el que mejor pudo clasificar a las galaxias de la clase 4, clasificando correctamente al 97 % de estas, lo que significa que clasificó correctamente 35 galaxias de las 36 galaxias de la clase 4.

Antes de proseguir, en la tabla 5.3 encontramos resumidas los resultados para este par de experimentos.

Figura 5.9: Matriz de confusión del modelo preentrenado con la tarea de RPL y usando *fine tuning*.

Conjuto de entrenamiento	Conjunto de validación	Exactitud total	Exactitud en la clase 4
Balanceado	Sin balancear	74.63 %	72 %
Balanceado	Balanceado	69.3 %	97 %

Tabla 5.3: Resumen de resultados de los experimentos puramente supervisados.

5.3. Discusión de resultados

Empecemos hablando de los resultados obtenidos solo en la etapa de preentrenamiento autosupervisado. Se realizaron dos preentrenamientos usando la técnica de RPL, uno usando todos los canales de colores y otro en el cual, cada vez que se procesaba una imagen, se descartaban dos canales de color de la imagen de forma aleatoria. Como observamos en la tabla 5.1 ambos obtuvieron una exactitud de un poco más del 50 %. Este resultado podría parecer negativo a primera vista, sin embargo, y como lo mencionamos anteriormente, este resultado mejora significativamente al modelo que realiza la clasifi-

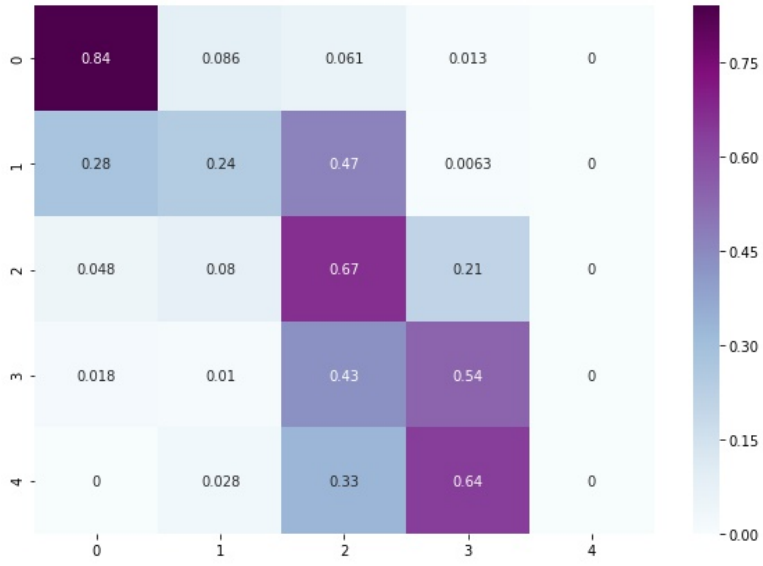

Figura 5.10: Matriz de confusión del modelo preentrenado con la tarea de RPL, descartando dos canales de color, y usando transferencia de conocimiento.

cación de forma aleatoria, cuya exactitud sería del 12.5 %. Además de lo anterior, y en favor de la de nuestros resultados presentamos las matrices de confusión, fig. 5.2 y fig. 5.3, para ambos modelos sobre el conjunto de validación donde podemos apreciar que en ambas matrices los colores más oscuros se presentan en la diagonal, que representa las clasificaciones hechas correctas. Por lo anterior, y de forma reiterativa, concluimos que ambas variantes de la tarea del RPL fueron realizadas exitosamente por sus respectivos modelos.

Por otro lado, tenemos a los modelos que realizaron las dos variaciones de la tarea de *jigsaw*, una que usa 50 permutaciones y otra que usa 100 permutaciones. Como mencionamos anteriormente, si los modelos realizaran la clasificación entre las clases de forma aleatoria entonces los modelos que usa 50 y 100 permutaciones obtendrían una exactitud del 2 % y 1 % respectivamente, pero, como podemos ver en la tabla 5.1, ambos modelos obtuvieron una exactitud mayor al 90 % con esto concluimos que los modelos entrenados para resolver la tarea de *jigsaw*, en sus dos variaciones, resuelven dicha tarea de forma exitosa.

Pasemos ahora a discutir los resultados derivados de emplear el preentrenamiento

Figura 5.11: Matriz del modelo que usa ResNet50 monitoreado con un conjunto no balanceado.

autosupervisado para inicializar los pesos de la red para resolver la tarea de clasificación de galaxias en la base de datos de Nair [3]. De estos experimentos podemos recapitular los siguientes comentarios. El experimento de control, el cual es nuestro punto de referencia para comparar los modelos que fueron preentrenados de forma autosupervisada, obtuvo una exactitud de 69.9551 % y pudo clasificar correctamente una galaxia de la clase 4, que es la que se encuentra desfavorecida por la distribución de clases en la base de datos de Nair [3]. Los modelos que usaron el preentrenamiento autosupervisado solo mejoraron ligeramente el rendimiento del experimento de control en dos casos, ver tabla 5.2, el modelo que usó los pesos aprendidos con la tarea de *jigsaw* con 50 clases bajo el esquema de transferencia de conocimiento pudo clasificar correctamente 4 galaxias de la clase 4, pero no mejoró la exactitud total sobre el conjunto de prueba, por otro lado el modelo que usó los pesos aprendidos por la tarea de RPL bajo el esquema de *fine tuning* obtuvo una exactitud del 70.1046 %, mejorando casi un 0.15 % la exactitud del experimento de control, sin embargo no pudo clasificar correctamente ninguna galaxia de la clase 4. El resto de modelos que usaron preentrenamiento autosupervisado mostraron resultados peores que el experimento de control, tanto en la exactitud total como en la exactitud

Figura 5.12: Matriz de confusión del modelo que usa ResNet50 monitoreado con un conjunto balanceado.

sobre la clase 4.

De lo anterior concluir que la implementación de técnicas de aprendizaje autosupervisado, tales como las tareas de RPL y *jigsaw*, para la extracción de representaciones visuales que permitan aminorar el efecto de la poca cantidad datos, así como el desbalance de clases de estos, sobre los modelos de aprendizaje profundo para la clasificación de galaxias según su morfología, no ha arrojado resultados positivos. O tal vez, no hemos atacado el problema desde la perspectiva apropiada, hablaremos más de esto en la sección de trabajos futuros 6.2.

Comentemos ahora los resultados obtenidos en los experimentos puramente supervisados. Como podemos ver en la tabla 5.3, el entrenar el modelo con una base de datos balanceada a través del método de *up-sampling* y monitoreado con un conjunto de validación no balanceado, es decir que preserva la distribución de clases original, ayuda significativamente a mejorar tanto la exactitud total sobre el conjunto de prueba como la exactitud sobre la clase 4, alcanzando una exactitud total del 74.6387 % y clasificando correctamente 26 galaxias de las 36 pertenecientes a la clase 4. El entrenar un modelo con una base de datos balanceada a través del método de *up-sampling* y monitoreado con un

conjunto de validación también balanceado por el mismo método, no ayuda a mejorar la exactitud total sobre el conjunto de prueba respecto al experimento de control de la tabla 5.2, sin embargo, este modelo fue capaz de clasificar correctamente todas las galaxias de la clase 4 a excepción de una.

Por lo tanto, de lo anterior podemos concluir que la implementación de técnicas más conservadoras para la lidiar con problemas como la escasez de datos y desbalance entre clases, como la técnica de *up-sampling*, resultan un mejor opción que la de emplear el preentrenamiento autosupervisado.

Pasemos ahora a poner nuestros resultados en contexto con los trabajos mencionados en la sección 2.1. Como mencionamos en dicha sección, el problema de clasificación de galaxias por su morfología se ha tratado anteriormente, en especifico de forma muy amplia el problema de clasificación binaria, entre galaxias elípticas y espirales. Sin embargo, como señalamos en la sección 2.1, el problema de clasificación multiclase, aunque se ha tratado anteriormente, se ha dejado bastante abierto por las siguientes razones, no se pone a disposición del lector los conjuntos de prueba con los cuales fueron obtenidos los resultados reportados, no se describe de forma explícita la forma en las cuales fueron etiquetadas las galaxias y, aunque se reporten los valores de las métricas como la exactitud, no se proporcionan las matrices de confusión obtenidas. Estos factores repercuten de varias maneras, la primera es que al no poder reproducir los conjuntos de prueba y las etiquetas empleadas en dichos trabajos, no podemos evaluar nuestros modelos de tal forma que podamos comparar nuestros resultados con los de ellos. Además, pese a que se los trabajos anteriores reportan métricas como la exactitud, es imposible saber que tan bien sus modelos atacan el problema del desbalance de clases, el cual, como hemos visto, es un problema bastante presente en las bases de datos de clasificación de galaxias según su morfología.

Teniendo en cuenta lo anterior, más allá de si las técnicas de aprendizaje autosupervisado presentadas aquí ayudan a la clasificación multiclase de galaxias, consideramos que los resultados presentados en este trabajo mejoran los resultados de trabajos previos por el simple hecho de hacer explícitos los detalles anteriormente mencionados, detallamos claramente como se hizo el etiquetado de nuestras galaxias y mostramos, a través de las matrices de confusión, el rendimiento de los modelos para cada clase en especifico, los cuales son resultados no reportados con anterioridad.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

En esta sección hacemos referencia a las conclusiones e ideas desarrolladas en la ultima sección del capitulo anterior (sección 5.3), solo que de manera más general y compacta. Si el lector quisiera ahondar en el significado de cada uno de los puntos a continuación presentados, le sugerimos referirse a dicha sección.

- Pudimos realizar de forma exitosa los preentrenamientos autosupervisados, tanto la tarea de RPL como la de *jigsaw*, en todas las variaciones propuestas.

- El preentrenamiento autosupervisado ayudó a mejorar el rendimiento de los modelos sobre la tarea de clasificación multiclase en la base de datos de Nair [3] en dos casos respecto al experimento de control. Uno mejorando la exactitud total del modelo de control (pero no mejorando la exactitud sobre la clase 4), obteniendo un 70.1 % de exactitud total y otro mejorando la exactitud sobre la clase 4 (pero no mejorando la exactitud total), que es la más desfavorecida en la base de datos, pudiendo clasificar correctamente 11 % de galaxias de dicha clase. Para tenerlo como referencia, en el experimento de control se obtuvo una exactitud total del 69.95 % y una exactitud del 2.8 % sobre la clase 4.

- El resto de experimentos que usaron el preentrenamiento autosupervisado no mejoraron el rendimiento del experimento de control, ni en la exactitud total ni en la exactitud sobre la clase 4.

- El experimento que obtuvo la mejor exactitud total fue el que empleó una arquitectura ResNet50 entrenada solamente de forma supervisada con un conjunto de entrenamiento balanceado con el método de *up-sampling* y monitoreado con un conjunto de validación sin balancear. Dicho modelo obtuvo una exactitud total del 74.63 % y una exactitud sobre la clase 4 del 72 %.

- El experimento que obtuvo la mejor exactitud sobre la clase 4 fue el que empleó una arquitectura ResNet50 entrenada solamente de forma supervisada con un conjunto de entrenamiento balanceado con el método de *up-sampling* y monitoreado con un conjunto de validación también balanceado por el método de *up-sampling*. Dicho modelo obtuvo una exactitud total del 69.3 % y una exactitud sobre la clase 4 del 97 %.

- Podemos concluir que la implementación de técnicas de aprendizaje autosupervisado, tales como las tareas de RPL y *jigsaw*, para la extracción de representaciones visuales que permitan aminorar el efecto de la poca cantidad datos, así como el desbalance de clases de estos, sobre los modelos de aprendizaje profundo para la clasificación de galaxias según su morfología, no ayudaron a mejorar la tarea de clasificación multiclase en la base de datos de Nair [3] de forma clara. O tal vez, no hemos atacado el problema desde la perspectiva apropiada, hablaremos más de esto en la siguiente sección de trabajos futuros.

- Por el contrario, la implementación de técnicas más conservadoras para la lidiar con los problemas mencionados, como la técnica de *up-sampling*, resultan un mejor camino por el momento.

6.2. Trabajos futuros

- Como mencionamos al final de la sección 2.2, para el momento en que se realizaron la mayoría de los experimentos presentados en este trabajo, las representaciones visuales obtenidas a través del aprendizaje autosupervisadosueprvisado no eran igual de buenas que las obtenidas por el supervisado. Probablemente, esperar que se obtuviera un resultado diferente en nuestro caso fue una suposición bastante

ambiciosa. Si cambiamos el paradigma de querer mejorar el rendimiento para las tareas de clasificación, se abre un mundo de posibilidades. Por ejemplo, probablemente las representaciones aprendidas por las tareas pretexto implementadas en este trabajo no eran del todo adecuadas para la tarea específica de clasificación, sin embargo, eso no quiere decir que no capturaran características importantes de las imágenes de galaxias. Podríamos investigar la naturaleza de éstas características a través de aplicar otros algoritmos de aprendizaje no supervisado sobre las características extraídas de las imágenes de galaxias por los modelos de aprendizaje autosupervisado, podríamos investigar qué tipo de clasificaciones resultarían de aplicar un algoritmo de *clustering* en el espacio de estas características y corroborar con expertos astrónomos que tanto estas clases obtenidas de forma no supervisada reflejan la naturaleza de alguna característica física de las galaxias.

- Bajo el mismo esquema del punto anterior, no solo se podría aplicar métodos de aprendizaje no supervisado sobre el espacio de características extraídas por lo modelos entrenados de forma supervisada, sino también se podría hacer el mismo tipo de investigaciones usando otro métodos de aprendizaje no supervisado para redes neuronales, por ejemplo estudiando la representación obtenida por un *autoencoder*, o básicamente cualquier algoritmo de aprendizaje no supervisado para la obtención de representaciones visuales.

- Por otro lado, mientras los experimentos de este trabajo estaban llevándose a cabo, un conjunto investigadores de Google Research (entre ellos Geoffrey Hinton) presentaron un artículo [39] en el cual no solo se plantea una tarea *pretexto* con la cual se obtienen representaciones visuales cuya calidad equipara a las representaciones obtenidas a partir de medios supervisados, sino que además se plantea un esquema de aprendizaje semi-supervisado en el cual se pueden usar tanto imágenes etiquetadas como no etiquetadas. La implementación de este algoritmo queda abierta para futuros trabajos.

Bibliografía

[1] Ronald Buta. Galaxies: Classification. *The Encyclopedia of Astronomy and Astrophysics*, 2004.

[2] https://data.galaxyzoo.org.

[3] Preethi B. Nair and Roberto G. Abraham. A catalog of detailed visual morphological classifications for 14,034 galaxies in the sloan digital sky survey. *Astrophysical Journal, Supplement Series*, 186(2):427–456, 2010.

[4] P. H. Barchi, R. R. de Carvalho, R. R. Rosa, R. Sautter, M. Soares-Santos, B. A. D. Marques, E. Clua, T. S. Gonçalves, C. de Sá-Freitas, and T. C. Moura. Machine and Deep Learning Applied to Galaxy Morphology – A Comparative Study. 2019.

[5] http://skyserver.sdss.org/.

[6] https://archive.ics.uci.edu/ml/datasets/iris.

[7] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data.

[8] Christopher J Conselice. the Relationship Between Stellar Light Distributions of Galaxies and Their. *ApJS*, 147:1–28, 2003.

[9] S. James Press. *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference*. Courier Corporation, 1982.

[10] R. R. Rosa, R. R. de Carvalho, R. A. Sautter, P. H. Barchi, D. H. Stalder, T. C. Moura, S. B. Rembold, D. R.F. Morell, and N. C. Ferreira. Gradient pattern analysis applied to galaxy morphology. *Monthly Notices of the Royal Astronomical Society: Letters*, 477(1):L101–L105, 2018.

[11] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, Cambridge, U.K., 2011.

[12] Rubens Andreas Sautter. *Gradient Pattern Analysis: New Methodological and Computational Features and Application. Master's thesis. National Institute for Space Research*. PhD thesis, 2018.

[13] R Sautter and P Barchi. pyGHS: Computing Geometric Histogram Separation in Binomial Proportion Patterns. *Journal of Computational Interdisciplinary Sciences*, 8(1):47–51, 2017.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1–9, 2015.

[15] H. Domínguez Sánchez, M. Huertas-Company, M. Bernardi, D. Tuccillo, and J. L. Fischer. Improving galaxy morphologies for SDSS with deep learning. *Monthly Notices of the Royal Astronomical Society*, 476(3):3661–3676, 2018.

[16] J. Vega-Ferrero, H. Domínguez Sánchez, M. Bernardi, M. Huertas-Company, R. Morgan, B. Margalef, M. Aguena, S. Allam, J. Annis, S. Avila, D. Bacon, E. Bertin, D. Brooks, A. Carnero Rosell, M. Carrasco Kind, J. Carretero, A. Choi, C. Conselice, M. Costanzi, L. N. da Costa, M. E. S. Pereira, J. De Vicente, S. Desai, I. Ferrero, P. Fosalba, J. Frieman, J. García-Bellido, D. Gruen, R. A. Gruendl, J. Gschwend, G. Gutierrez, W. G. Hartley, S. R. Hinton, D. L. Hollowood, K. Honscheid, B. Hoyle, M. Jarvis, A. G. Kim, K. Kuehn, N. Kuropatkin, M. Lima, M. A. G. Maia, F. Menanteau, R. Miquel, R. L. C. Ogando, A. Palmese, F. Paz-Chinchón, A. A. Plazas, A. K. Romer, E. Sanchez, V. Scarpine, M. Schubnell, S. Serrano, I. Sevilla-Noarbe, M. Smith, E. Suchyta, M. E. C. Swanson, G. Tarle, F. Tarsitano, C. To, D. L. Tucker, T. N. Varga, and R. D. Wilkinson. Pushing automated morphological classifications to their limits with the Dark Energy Survey. 17(December):1–17, 2020.

[17] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting Self-Supervised Visual Representation Learning. 2019.

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv*, (2016):1–16, 2018.

[19] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in Neural Information Processing Systems*, 1(January):766–774, 2014.

[20] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9910 LNCS:69–84, 2016.

[21] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1422–1430, 2015.

[22] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:645–654, 2017.

[23] Geoffrey E Krizhevsky, Alex and Sutskever, Ilya and Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778, 2016.

[25] Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in Neural Information Processing Systems*, 2017-Decem:2215–2225, 2017.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[27] http://www.image-net.org/.

[28] http://places.csail.mit.edu/downloaddata.html.

[29] John Kelleher. *Deep Learning*. MIT Press, Cambridge, Massachusetts, 2019.

[30] Charu Aggarwal. *Neural Networks and Deep Learning*. Springer, NY, USA, 2018.

[31] Ian Pointer. *Programming PyTorch for Deep Learning.* O'Reilly, 2019.

[32] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. pages 1–31, 2016.

[33] Nitish Srivastava Salakhutdinov, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[34] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *British Machine Vision Conference 2016, BMVC 2016*, 2016-Septe:87.1–87.12, 2016.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

[36] http://www.nsatlas.org/data.

[37] https://www.desi.lbl.gov/.

[38] Hideki Imai. *Essentials of Error-Control Coding Techniques.* Academix Press, Inc., San Diego, California, 1990.

[39] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, (Figure 1), 2020.