# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA Y ARQUITECTURA

UNIDAD TICOMÁN

# DETECCIÓN DE ESTRUCTURA EXTERNA DE MAREAS GALÁCTICAS CON REDES NEURONALES CONVOLUCIONALES

T E S I N A

QUE PARA OBTENER EL TÍTULO DE INGENIERO TOPÓGRAFO Y FOTOGRAMATRISTA

P R E S E N T A:

JUAN GUILLERMO JOSÉ HERNÁNDEZ

DIRECTORES DE TESINA:

ING. ARIADNA ORTEGA ESPINOSA
ING. JULIÁN MARES VALVERDE
ING. JOSÉ OZIEL GUZMÁN HERNÁNDEZ
DR. HÉCTOR MANUEL HERNÁNDEZ TOLEDO
DR. JOSÉ ANTONIO VÁZQUEZ MATA

CIUDAD DE MÉXICO, DICIEMBRE 2019

# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA Y ARQUITECTURA

UNIDAD TICOMÁN

# DETECCIÓN DE ESTRUCTURA EXTERNA DE MAREAS GALÁCTICAS CON REDES NEURONALES CONVOLUCIONALES

T E S I N A

QUE PARA OBTENER EL TÍTULO DE INGENIERO TOPÓGRAFO Y FOTOGRAMATRISTA

P R E S E N T A:

JUAN GUILLERMO JOSÉ HERNÁNDEZ

DIRECTORES DE TESINA:

ING. ARIADNA ORTEGA ESPINOSA
ING. JULIÁN MARES VALVERDE
ING. JOSÉ OZIEL GUZMÁN HERNÁNDEZ
DR. HÉCTOR MANUEL HERNÁNDEZ TOLEDO
DR. JOSÉ ANTONIO VÁZQUEZ MATA

CIUDAD DE MÉXICO, DICIEMBRE 2019

Autorización de uso de obra

**INSTITUTO POLITÉCNICO NACIONAL**

P R E S E N T E

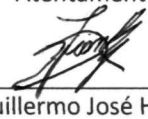
Bajo protesta de decir verdad el que suscribe: **José Hernández Juan Guillermo**, manifiesto ser autor titular de los derechos morales y patrimoniales de la obra titulada **"DETECCIÓN DE ESTRUCTURA EXTERNA DE MAREAS GALÁCTICAS CON REDES NEURONALES CONVOLUCIONALES"**, en adelante "la Tesina" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el **artículo 27 fracción II inciso b) de la Ley Federal del derecho de autor**, otorgo al **INSTITUTO POLITÉCNICO NACIONAL**, en adelante, **"EL IPN"** autorización no exclusiva para comunicar y exhibir Públicamente total o parcialmente en medios digitales (Publicación en línea) "la Tesina" por un periodo de **TIEMPO INDEFINIDO** contando a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a " EL IPN" de su terminación.

En virtud de lo anterior **"EL IPN"** deberá reconocer en todo momento mi calidad de autor de "La Tesina".

Adicionalmente, y en mi calidad de autor titular de los derechos morales y patrimoniales de "La tesina" manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito prospecto de "La tesina", por lo que deslindo de toda responsabilidad "Al IPN" en caso de que el contenido de "la Tesina" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México, a 12 de diciembre de 2019

Atentamente



Juan Guillermo José Hernández

E.S.I.A.
UNIDAD TICOMAN
RECIBIDO
23 AGO 2019 14:00
DIRECCION

"2019, Año del Caudillo del Sur, Emiliano Zapata"
60 años de la Unidad Profesional Adolfo López Mateos
70 Aniversario del CECyT No. 3 "Estanislao Ramírez Ruiz"
60 años de XEIPN Canal Once, orgullosamente politécnico
60 Aniversario del CECyT No. 4 "Lázaro Cárdenas"

**Folio:** DES/4548/2019

**Asunto:** Impartición de Seminario

Ciudad de México, a 20 de agosto de 2019

SUBDIRECCION ACADEMICA
E.S.I.A.
RECIBIDO
23 AGO 2019 15:51

**DR. ARTURO ORTIZ UBILLA**
**DIRECTOR DE LA ESCUELA SUPERIOR DE INGENIERÍA Y ARQUITECTURA (ESIA), UNIDAD TICOMÁN DEL INSTITUTO POLITÉCNICO NACIONAL**
**P R E S E N T E**

Con fundamento en el Artículo 44, Fracción VII del Reglamento Orgánico; Artículo 5, Fracción III del Reglamento General de Estudios; Artículo 12 del Reglamento de Titulación Profesional del Instituto Politécnico Nacional; en atención a su oficio DET/1532/2019, le comunico que se autoriza la impartición del Seminario de Actualización con Opción a Titulación:

**"GEOMÁTICA"**

| | |
|---|---|
| Registro: | DES/ESIA-TIC/S/010-09/2011-2019 |
| Vigencia del seminario: | 22 de marzo de 2018 al 22 de marzo de 2020 |
| Duración: | 172 horas. |
| Periodo de Impartición: | Del 03 de septiembre al 12 de diciembre de 2019. |
| Horario: | martes y jueves de 18:00 a 22:00 horas y sábados de 10:00 a 14:00 horas. |
| Sede: | ESIA-TIC. |
| Expositores: | Ing. Ariadna Ortega Espinosa, Ing. Julián Mares Valverde e Ing. José Oziel Guzmán Hernàndez. |

Debiendo observar lo siguiente:

- Enviar la lista inicial oficial de participantes, firmada y sellada por el Coordinador del Seminario y el Subdirector Académico dentro de los primeros diez días hábiles posteriores a la fecha del inicio del seminario.
- Dar a conocer a los participantes el folio de autorización correspondiente, para trámites de titulación ante la Dirección de Administración Escolar.

"2019, Año del Caudillo del Sur, Emiliano Zapata"
60 años de la Unidad Profesional Adolfo López Mateos
70 Aniversario del CECyT No. 3 "Estanislao Ramírez Ruiz"
60 años de XEIPN Canal Once, orgullosamente politécnico
60 Aniversario del CECyT No. 4 "Lázaro Cárdenas"

- Al concluir el programa del seminario enviar la relación de asistencia, de evaluación final y de trabajos finales, en un plazo no mayor a 20 días hábiles, para la emisión de las constancias a los participantes.

Cabe señalar que tanto la información emitida para la autorización de vigencia, como los datos de los participantes utilizados en la emisión de constancias, está sustentada en los anexos adjuntos al oficio enviado por usted, por lo que solicito verificarla a detalle previamente a su trámite.

Sin otro particular, le envío un cordial saludo.

**ATENTAMENTE**
**"La Técnica al Servicio de la Patria"**

PODER EJECUTIVO FEDERAL ESTADOS UNIDOS MEXICANOS MEXICO CDMX

**M. EN C. ROSALÍA MARÍA DEL CONSUELO TORRES BEZAURY**
**DIRECTORA**

INSTITUTO POLITÉCNICO NACIONAL
Dirección de Educación Superior

c.c.p Dr. Jorge Toro González. - Secretario Académico del IPN.
Lic. Marisela Cabrera Rojas – Directora de Administración Escolar DAE.

T-14744

**RMCTB/ECGV/ymvn**

**RESUMEN**

El estudio de las galaxias, tiene muchos beneficios a largo plazo, en el desarrollo tecnológico se ha presionado constantemente por instrumentos, procesos y software contribuyendo en áreas como la óptica y la electrónica, por ejemplo: los primeros sensores fueron desarrollados para capturar imágenes astronómicas, el lenguaje informático fue creado para ser utilizado por el telescopio Kitt Peak, el GPS depende de objetos astronómicos, como los quásares y las galaxias distantes para determinar posiciones precisas, la síntesis de apertura o imágenes sintéticas desarrollada por el astrónomo Martin Ryle, se utiliza para generar imágenes de resonancia magnética, el software para procesar imágenes satelitales tomadas desde el espacio ahora está ayudando a los investigadores médicos a establecer un método simple para implementar la detección a gran escala de la enfermedad de Alzheimer (ESA, 2013), entre otros.

Además, en 1949 se descubrió que el desarrollo científico y tecnológico de un país o región está estrechamente relacionado con su índice de desarrollo humano, una estadística que es una medida de la esperanza de vida, la educación y los ingresos. Desde una perspectiva en nivel más apremiante este tipo de estudios sirve para adquirir nuevos conocimientos como requisitos para prolongar la supervivencia de la especie humana. En este trabajo se construyó una red neuronal convolucional que detecta estructuras de mareas.

Antes de empezar con la construcción de la red neuronal convolucional, fue necesario obtener las imágenes del proyecto DESI DECaLS (The Dark Energy Camera Legacy Survey), la ubicación para acceder a las bases de datos de galaxias residuales fue proporcionada por el Instituto de Astronomía de la UNAM, así como el catálogo de galaxias Nair and Abraham (2010).

Una vez descargadas las imágenes se consideraron tres tipos de marea galáctica, las que se generan después de una interacción entre galaxias, las que tienen marea fuerte y las que a su alrededor tienen capas de marea. El total de imágenes detectadas con marea fue de 1414. Durante el proceso se usaron varias técnicas que permitieron mejorar el comportamiento de la red neuronal.

La construcción de la red neuronal convolucional se realizó a partir de imágenes de entrada, imágenes de validación e imágenes de prueba. Es decir, las 1414 imágenes con presencia de marea se dividieron en esos tres sectores. A su vez cada apartado fue dividido en galaxias con marea y galaxias sin marea.

Desde un navegador se abrió un cuaderno Jupyter para la creación de la red. Se importó Keras y TensorFlow, además se agregó la ubicación de las carpetas que contenían las imágenes de galaxias y se imprimieron los números correspondientes para asegurar que fueran reconocidas y se importaron los primeros filtros de la red convolucional. Los cuales corresponden a operaciones de convolución, en este caso, con la función de activación ReLU. Para la última capa siempre se agregó la función sigmoide para que los resultados fueran probabilísticos.

Se realizaron dos métodos de entrenamiento, una con una red preentrenada y otra sin ella. Todas las imágenes fueron redimensionadas antes de aplicarles los filtros. Se utilizaron filtros de convolución Conv2D y MaxPooling, ambos de 2x2 y 3x3. La red preentrenada que se utilizó es VGG19 con las dimensiones de imágenes de 224x224 a color.

El optimizador que primero se utilizó es RMSProp, y SGD para la red de la base preentrenada. Las funciones de pérdida utilizadas fueron: cros-entropia binaria y mse. La métrica fue la misma para ambas: acc.

También se usó la técnica del aumento de datos, para aumentar el número de imágenes de entrenamiento y que sea más fácil para la red aprender, ya que teóricamente las redes se desempeñan mejor cuando el número de datos de entrada es grande.

Al final de cada entrenamiento, se generaron gráficas que mostraban el comportamiento de la red con cada iteración respecto a la precisión y la pérdida.

Las imágenes de prueba se utilizaron para verificar el comportamiento de la red neuronal con imágenes que nunca había visto, redimensionándolas también como a las demás para disminuir el consumo de recursos computacionales.

En general, la arquitectura de las capas sigue la secuencia de filtros de convolución y un aplanado de datos, es decir, todos los datos correspondientes a imágenes se generalizan en un vector con neuronas que puedan ser reconocidas. Enseguida están las capas densas que van después de la capa de entrada y antes de la última capa, a la cual solo se le dio una neurona de salida.

Con el procedimiento empleado para la construcción de la red neuronal convolucional se obtuvieron dos resultados, cada uno con una arquitectura diferente, pero con el mismo objetivo, así que solo se tomó en cuenta el resultado final, es decir, el modelo VGG19 preentrenado dio un resultado del 85% en precisión para la clasificación binaria de las mareas galácticas.

## ABSTRACT

The study of galaxies, has a lot of long-term benefits, in technological development has been constantly pressed by instruments, processes and software contributing in areas such as optics and electronics, for example: the first sensors were developed to capture astronomical images, the computer language was created to be used by the Kitt Peak telescope, the GPS depends on astronomical objects, such as quasars and distant galaxies to determine precise positions, the opening synthesis or synthetic images developed by astronomer Martin Ryle, it is used to generate magnetic resonance imaging, software to process satellite images taken from space is now helping medical researchers establish a simple method to implement large-scale detection of Alzheimer's disease (ESA, 2013), among others.

In addition, in 1949 it was discovered that the scientific and technological development of a country or region is closely related to its human development index, a statistic that is a measure of life expectancy, education and income. From a more pressing level, this type of study serves to acquire new knowledge as requirements to prolong the survival of the human species. In this work was constructed a convolutional neural network that detects tidal structures.

Before starting with the construction of the Convolutional Neural Network, it was necessary to obtain the images of the DESI DECaLS project (The Dark Energy Camera Legacy Survey), the location to access the residual galaxies databases was provided by the Instituto de Astronomía UNAM, as well as the galaxies catalog Nair and Abraham (2010).

Once the images were downloaded, three types of galactic tide were considered, which are generated after an interaction between galaxies, those with strong tides and those around them have tidal layers. The total number of images detected at tide was 1414. During the process several techniques were used that allowed to improve the behavior of the neural network.

The construction of the convolutional neural network was made from input images, validation images and test images. Namely, the 1414 images with a tidal presence were divided into these three sectors. At the same time, each section was divided in galaxies with tide and galaxies without tide.

From a browser, a Jupyter notebook was opened for the creation of the network. Keras and TensorFlow were imported, also, the folder directories with the images of galaxies were added and the corresponding numbers were printed to ensure that they were recognized and the first filters of the convolutional network were imported. Which correspond to convolution operations, in this case, with the ReLU activation function. For the last layer the sigmoid function was always added so that the results were probabilistic.

Two training methods were performed, one with a pre-trained network and one without it. All images were resized before applying filters. Conv2D and MaxPooling convolution filters were used, both 2x2 and 3x3. The pre-trained network that was used is VGG19 with the dimensions of 224x224 color images.

The optimizer that was first used is RMSProp, and SGD for the pre-trained base network. The loss functions used were: cros-binary entropy and mse. The metric was the same for both: acc.

The data augmentation technique was also used, to increase the number of training images and make it easier for the network to learn, since theoretically the networks perform better when the number of input data is big.

At the end of each training, graphs were generated to show the behavior of the network with each iteration regarding accuracy and loss.

The test images were used to verify the behavior of the neural network with images that it had never seen, also resizing them as the anothers to reduce the consumption of computational resources.

In general, the architecture of the layers follows the sequence of convolution filters and a flattening of data, that is, all the data corresponding to images is generalized in a vector with neurons that can be recognized. Next are the dense layers that are after the input layer and before the last layer, it was only given an output neuron.

With the procedure used for the construction of the convolutional neural network, two results were obtained, each with a different architecture, but with the same objective, so only the final result was taken into account, that is, the pretrained VGG19 model gave a result of 85% in accuracy for the binary classification of galactic tides.

## AGRADECIMIENTOS

En este apartado quiero agradecer al Instituto Politécnico Nacional por permitirme abordar este tema muy interesante para titulación, que no hubiera sido posible sin la ayuda de varios asesores internos y externos. Estoy agradecido también con el doctor José Antonio Vázquez Mata, investigador del Instituto de Astronomía de la UNAM, por asesorarme todo este tiempo en la elaboración de la tesina. Al doctor Héctor Manuel Hernández Toledo, por mostrarme siempre su apoyo cuando lo necesitaba. También al amigo Luis Carlos Mascherpa por aclararme algunos conceptos y algoritmos del proyecto.

También quiero agradecer al ingeniero José Oziel Guzmán Hernández por darme la idea de trabajar con un tema de astronomía durante el curso, con el cual me sentí muy comprometido y al mismo tiempo contento por tratarse de un área que me apasiona. Al ingeniero Julián Mares Valverde por transmitirme confianza cuando me revisaba mis avances de proyecto, así como por estar siempre a disposición de mis compañeros dándonos ánimos para poder lograr nuestro objetivo.

En el IPN conocí a muy buenos profesores, por ellos ahora veo la vida algo diferente, compartieron siempre sus experiencias de trabajo, los errores cometidos, los sueños cumplidos, lo cual para nosotros fue como adquirir experiencias sin siquiera vivirlas. También en la escuela conocí a muy buenos compañeros que a pesar de que tenían muchas responsabilidades siempre mostraron interés por la carrera y muchas ganas de seguir aprendiendo.

Agradezco a mi familia por creer en mí, a mis hermanos, a mis padres: Claudia Hernández y Meinardo José, quienes siempre me aconsejaron durante mi estancia en la universidad, porque, aunque estaba lejos de casa, sentí que ellos estaban cerca de mí.

Gracias o todos los maestros que me aconsejaron para llegar hasta aquí, a personas que sin gustarles la escuela me incitaban a seguir estudiando, animándome cuando les parecía que estaba dudando de mí, y gracias a Dios por estar siempre conmigo y con todos.

# ÍNDICE

# I. INTRODUCCIÓN

## 1.1 Estado del Arte

En los últimos 25 años se han creado varias bases de datos de galaxias detectadas por los telescopios que forman parte de diversos proyectos con el propósito de descubrir la formación de las galaxias observadas, para decodificar la información de las imágenes astronómicas se han usado técnicas del Aprendizaje Profundo.

Los astrónomos clasifican las galaxias por sus formas o morfología. Desde la década de 1930, las morfologías se han detectado mediante inspección visual, pero con el aumento de datos astronómicos es imposible analizar todas las galaxias con el ojo humano.

Es así que, en 2015, se presentó un catálogo de morfologías visuales de aproximadamente 50000 galaxias obtenidas de GOODS (The Great Observatories Origins Deep Survey), un proyecto de observaciones extremadamente profundas de la NASA, de la ESA y de otras instalaciones terrestres, donde se utilizó el aprendizaje profundo y la estimación con redes neuronales convolucionales. Marc Huertas-Company y otros investigadores de distintas instituciones del mundo presentaron un modelo de clasificación que cubre las probabilidades de que cada galaxia pueda tener un disco, presentar una irregularidad, ser compacta o no clasificable. El resultado fue una clasificación con fracción errónea menor al 1%.

Las imágenes astronómicas y los espectros obtenidos por los telescopios se van actualizando con el tiempo, lo que hace que se usen diferentes técnicas y métodos del aprendizaje profundo con tal de obtener mejores precisiones.

En el año 2018, la astrofísica Helena Domínguez Sánchez y los investigadores Huertas-Company, M., Bernardi, M., Tuccillo, D., Fischer, J. L., presentaron un catálogo morfológico de aproximadamente 670000 galaxias SDSS (Sloan Digital Sky Survey), el cual es el catálogo más grande y preciso en la actualidad. Las clasificaciones se obtuvieron con algoritmos de aprendizaje profundo utilizando redes neuronales convolucionales. Para la creación del modelo se utilizaron los catálogos GZ2 y Nair & Abraham (2010) con imágenes a color en las que se encontraron discos, barras, abultamientos y fusiones. Asimismo, el modelo permite la separación entre galaxias elípticas puras (E) de galaxias lenticulares (S0) obteniendo una precisión mayor al 97%; al aplicar galaxias de otras bases de datos obtiene una precisión de más del 90%.

En la actualidad se siguen usando redes neuronales convolucionales aplicadas a la astronomía para distintos fines, pero la mayoría coincide con la técnica del aprendizaje supervisado y la clasificación. Como en cada base de datos las características de las imágenes son diferentes, se sigue mejorando la clasificación morfológica con algoritmos de alto nivel, incluso ya se ha demostrado con el trabajo de Domínguez Sánchez, H., Huertas Company, M. y Bernardi, M. a principios del año 2019 que los algoritmos pueden mejorar la clasificación visual de galaxias para imágenes SDSS.

El estudio de las galaxias u objetos extra-galácticos también se hace con la finalidad de explicar el origen del universo, para esto se ha creado el Instrumento Espectroscópico de Energía Oscura (DESI) para observar aproximadamente 30 millones de galaxias con el objetivo de construir un mapa 3D que abarque el universo cercano a 11 mil millones de años luz. Con las imágenes obtenidas se podrán realizar mediciones de materia oscura en las galaxias.

En el proyecto internacional DESI participan casi 500 investigadores adscritos a 75 instituciones en 13 países, incluyendo cerca de 40 científicas y científicos mexicanos apoyados por el Consejo Nacional de Ciencia y Tecnología (Conacyt). Entre las instituciones mexicanas participantes se encuentran el Instituto de Física, el Instituto de Astronomía y el Instituto de Ciencias Nucleares de la UNAM, el Instituto Nacional de Investigaciones Nucleares (ININ), el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (Cinvestav), la Universidad de Guanajuato, así como estudiantes, posdoctorantes y dos jóvenes catedráticos del Conacyt.

Las imágenes de galaxias DESI brindan información sobre la estructura de mareas con las que se puede explicar la evolución del universo, estas estructuras normalmente pueden alterar la morfología de las galaxias y destruir a las galaxias satélite enanas.

Aunque fue a partir de 1970 cuando se consideró que las mareas galácticas estaban relacionadas con la evolución del universo, el estudio por su morfología continua, cada vez se crean nuevos catálogos desde la investigación observacional que corresponden a un estudio más detallado. Desde entonces se han detectado muchas galaxias con fuerzas de marea producidas por la actividad del material interno de la propia galaxia o factores externos, como las interacciones.

En este proyecto, fueron tomadas 12000 galaxias DESI para encontrar estructura de marea mediante el Aprendizaje Profundo, para acceder a las bases de datos de galaxias residuales la ubicación fue proporcionada por el Instituto de Astronomía de la UNAM, así como el catálogo de galaxias Nair (2010).

Las redes neuronales artificiales han permitido un gran avance tecnológico aplicable a muchas ciencias de investigación, pero en este caso particular, a detectar mareas en las galaxias seleccionadas del catálogo Nair (2010). Para ser más específico, creando una red neuronal convolucional se puede hacer una clasificación de imágenes.

Una red neuronal convolucional es uno de los algoritmos más populares del Aprendizaje Profundo, el cual es un tipo de aprendizaje automático en el que un modelo aprende a clasificar imágenes, videos, textos o sonidos. En el caso de las imágenes las redes neuronales convolucionales aprenden de imágenes de entrenamiento y detectan patrones que cada vez se hacen más complejos conforme se vayan añadiendo más capas ajustándose automáticamente en cada iteración, por eso se dice que la red aprende.

El trabajo de detectar marea de miles galaxias visualmente, es muy complicado en cuestiones de tiempo, actualmente existe la posibilidad de crear una red neuronal para la necesidad que se requiera, esto puede ser complicado, por eso existen varias herramientas del Aprendizaje Profundo que ayudan a hacerlo más fácil.

Keras y TensorFlow fueron las herramientas para desarrollar este proyecto en Jupyter, utilizando el lenguaje de programación Python. También se tomó en cuenta una red preentrenada, es decir, una red que fue entrenada con distintas imágenes, pero que puede servir de base para una nueva clasificación de imágenes.

## 1.2 Objetivo General

Hacer un censo de galaxias con mareas y desarrollar una red neuronal convolucional para identificarlas dentro del catálogo DESI (Dark Energy Spectroscopic Instrument) con millones de galaxias de todo tipo. Este trabajo podrá contribuir para encontrar una explicación del origen y evolución de estas galaxias, así como la evolución de la estructura jerárquica del universo.

### 1.2.1 Objetivos Particulares

- Clasificar visualmente 12000 imágenes DESI de acuerdo al tipo de marea que presenten.
- Desarrollar un código Python en el Cuaderno Jupyter y crear la CNN o ConvNet (Convolutional Neural Network).
- Realizar el entrenamiento y validación de la CNN usando las imágenes clasificadas previamente, hasta encontrar el mejor modelo para detectar galaxias con estructuras de marea.
- Obtener resultados de probabilidad con imágenes de prueba de mareas galácticas para comprender cómo se comportaría la red neuronal con datos que no había visto.

## 1.3 Hipótesis

Las redes neuronales convolucionales pueden encontrar estructuras en imágenes que pueden ser funcionales para varios proyectos de investigación. Aunque en muchos casos puede ser complicado construir una red tan precisa, los resultados pueden ser de gran ayuda, es decir, es imposible que los resultados arrojen una precisión del cien por ciento, pero sí pueden estar muy cerca. Conociendo las técnicas empleadas para la creación de redes neuronales convolucionales, es posible crear una que pueda detectar la estructura deseada, para este caso, detectar estructuras de marea como colas, capas o grandes estructuras provocadas por la fuerza de marea, las cuales dependiendo la forma en que se presenten están relacionadas al origen y evolución del universo.

## 1.4 Alcance

El presente proyecto considera imágenes de galaxias de 800 x 800 pixeles, de las cuales un porcentaje importante presenta mareas, todas dentro de la base de datos de DESI. Las galaxias fueron clasificadas solo para considerar las mareas fuertes, mareas en forma de capas (shells) y las producidas por la interacción de dos o más galaxias.

## II. ANTECEDENTES

### 2.1 Mareas Galácticas

Por definición, las mareas son un efecto diferencial de la gravitación. Las explosiones nucleares a menudo asociadas con las fusiones de galaxias no son inducidas directamente por las fuerzas de marea. Además, no todos los desechos de colisión encontrados alrededor de las fusiones son de origen de marea.

Las mareas gravitacionales han sido consideradas como la causa principal de la creación de filamentos en las galaxias que interactúan. Es por eso que tales características a menudo se denominan estructuras de marea. Este avance del año 1970 también se produjo por el nacimiento de las computadoras, las cuales permitieron hacer varios experimentos numéricos que en un principio mostraron que las mareas breves pero intensas surgen durante el encuentro de dos galaxias de disco que se deforman y pueden producir estructuras muy largas y delgadas llamadas colas de marea. Al paso del tiempo estas estructuras se desvanecen lentamente en el medio intergaláctico o son recuperados por su galaxia durante varios $10^9$ años.

Cabe mencionar que uno de los factores que puede llegar a producir estructuras de marea es la presencia de halos de materia oscura, lo cual aumenta la fricción dinámica favoreciendo la fusión de galaxias.

Aislada, una galaxia mantiene su material, que está hecho de materia oscura, estrellas, gas y polvo, unido gracias a la gravitación. Sin embargo, cuando se mueve en un potencial externo, creado por ejemplo por galaxias vecinas, puede experimentar fuerzas gravitacionales que son diferentes de un lado a otro de la galaxia. En otras palabras, la galaxia se hunde en un campo de mareas. Como resultado, su material sufre efectos deformantes que reorganizan los componentes individuales de la galaxia. Si el material se distribuye inicialmente de forma aleatoria y no se comparte un patrón de velocidad común durante la interacción, el efecto de marea no se visualiza como un cambio en toda una región de la galaxia, por lo tanto, tales mareas son difíciles de detectar, pero si existen patrones regulares en la distribución del material galáctico, por ejemplo en un disco, todas las estrellas se ven afectadas de la misma manera y por lo tanto, el efecto es mucho más visible. En términos generales significa que un campo de marea es más fácil de detectar cuando afecta una distribución regular y organizada de la materia, que cuando se aplica a estructuras isotrópicas.

Esta es la razón por la cual las características de las mareas, como las colas y los puentes, son bien visibles alrededor de las galaxias de disco donde el movimiento está bien organizado y es simplemente inexistente en elípticas, que producen distribuciones isotrópicas mucho más altas de posiciones y velocidades.

El halo de materia oscura o halo galáctico, principalmente los que son ligeros están asociados a colas de mareas largas y masivas, mientras que el potencial profundo creado por otras más masivas evitaría la creación de estructuras extendidas, ya que un halo denso parece ser más eficiente en retener el componente estelar unido.

Cabe aclarar los halos masivos pueden permitir el crecimiento de colas, siempre que la energía cinética del material del disco sea lo suficientemente alta como para equilibrar la profundidad del potencial gravitacional del halo de materia oscura masiva.

Cabe destacar que, el color óptico promedio de las colas de marea es consistente con que la mayor parte de su población estelar es más antigua que la interacción y originalmente nació en el disco de las galaxias originales. Sin embargo, también albergan componente estelar joven formando nudos de formación estelar con emisión ultravioleta, mientras que el polvo calentado causa la emisión infrarroja de colas de marea.

Las colas de marea y, en general, las estructuras finas que rodean a las galaxias (corrientes estelares, anillos, puentes, capas) se encuentran entre las señales menos ambiguas de la evolución de las galaxias, la formación de filamentos estelares solo puede explicarse por una colisión pasada entre galaxias.

Además, las colas de marea no solo pueden dar información del contenido de las galaxias originales y de cómo reaccionaron al ambiente, sino también para conocer la estructura y distribución de la materia oscura, el cual es el componente más masivo de las galaxias.

Para finalizar es necesario aclarar que:

- No todas las colisiones y eventos de acumulación masiva producen colas de marea.
- Las características de las mareas se desvanecen con el tiempo, ya sea porque vuelven a caer sobre sus progenitores o se evaporan en el medio intergaláctico.
- La tasa de destrucción de las características de las mareas depende del medio ambiente. Los entornos densos, como los cúmulos de galaxias, contribuyen a borrar los restos colisionados.

## 2.2 Inteligencia Artificial

La inteligencia artificial nació en la década de 1950, cuando un puñado de pioneros del incipiente campo de la ciencia de la computación comenzó a preguntar si se podía hacer que las computadoras "pensaran", una pregunta cuyas ramificaciones aún se siguen explorando hoy. Una definición concisa del campo sería la siguiente: el esfuerzo por automatizar las tareas intelectuales que normalmente realizan los humanos. Como tal, la IA es un campo general que abarca el Aprendizaje Automático y el Aprendizaje Profundo, pero que también incluye muchos más enfoques que no implican ningún aprendizaje. Los primeros programas de ajedrez, por ejemplo, solo involucraban reglas codificadas elaboradas por programadores, y no calificaban como aprendizaje automático.

Durante un tiempo bastante largo, muchos expertos creyeron que la inteligencia artificial a nivel humano podría lograrse haciendo que los programadores elaboren un conjunto suficientemente grande de reglas explícitas para manipular el conocimiento. Este enfoque se conoce como IA simbólica, y fue el paradigma dominante en IA desde los años cincuenta hasta finales de los ochenta. Alcanzó su máxima popularidad durante el auge de los sistemas expertos de la década de 1980.

Si bien la IA simbólica demostró ser adecuada para resolver problemas lógicos bien definidos, como jugar al ajedrez, resultó difícil resolver reglas explícitas para resolver problemas más complejos y difusos, como la clasificación de imágenes, el reconocimiento de voz y traducción de idiomas. Surgió un nuevo enfoque para tomar el lugar simbólico de la IA: el Aprendizaje Automático.

## 2.3 Aprendizaje Profundo y Redes Neuronales

En la época victoriana de la historia del Reino Unido, Charles Babbage, inventó el motor analítico: la primera computadora mecánica de propósito general conocida diseñada en las décadas de 1830 y 1840. Después, en 1843, Ada Lovelace, amiga y colaboradora de Babbage comentó sobre la invención*: "El motor analítico no tiene pretensiones de originar nada. Puede hacer lo que nosotros sabemos como ordenar que se ejecute ... Su función es ayudarnos a poner a disposición lo que ya conocemos ".*

Este comentario fue citado luego por el pionero de IA Alan Turing como *"la objeción de Lady Lovelace"* en su artículo histórico de 1950 *"Computing Machinery and Intelligence"*, que introdujo la prueba de Turing y los conceptos clave que darían forma a la IA. Turing estaba citando a Ada Lovelace mientras reflexionaba sobre si las computadoras de uso general podían ser capaces de aprender y ser originales, y llegó a la conclusión de que podían hacerlo.

Dos años después, en 1952, Arthur Samuel escribió el primer programa de ordenador capaz de aprender. El software era simplemente un programa que jugaba a las damas y que podía aprender de sus errores partida tras partida.

Frank Rosenblatt desarrolló el Perceptrón (Rosenblatt 1958), la primera red neuronal artificial especificada con toda precisión y orientada computacionalmente.

Como era una máquina que podía aprender y demostrar comportamiento adaptativo complejo, atrajo de inmediato la atención de los investigadores. Su procedimiento de convergencia de aprendizaje fue un avance definitivo sobre la teoría de Hebb. Asimismo, Rosenblatt desechó el enfoque de teóricos anteriores, que veían al cerebro como una computadora lógica. En vez de ello, lo consideró como un asociador y clasificador, cuya misión era asociar respuestas de clasificación a estímulos específicos.

En 1962 Rosenblatt publicó su libro *Principles of Neurodynamics* (Rosenblatt 1962) en el que presentó formalmente el Perceptrón como modelo para construir Redes Neuronales Artificiales. Los perceptrones se aplicaron rápidamente a resolver problemas tales como la predicción climatológica, la interpretación de electrocardiogramas y otros.

Tal parecía que se había hallado la clave para comprender el funcionamiento cerebral, emulando las Redes Neuronales naturales mediante redes complejas de perceptrones.

Sin embargo, pronto se comprobó que las redes con una capa de perceptrones eran incapaces de resolver problemas tan simples como la simulación de una compuerta lógica de tipo O exclusivo y, tras una investigación sobre las limitaciones de los perceptrones, Minsky y Pappert publicaron el libro *Perceptrons* (Minsky & Pappert 1969) donde se hacían patentes estas limitaciones.

### 2.4 Logros Del Aprendizaje Automático

Para el año 1996, el computador Deep Blue de IBM vence una partida de ajedrez a Gary Kaspárov, campeón del mundo vigente, aunque al final Kaspárov ganó 3 partidas más, derrotando a Deep Blue. Para mayo de 1997, se vuelven a enfrentar, pero esta vez con una nueva versión de computador llamado Deeper Blue, esta vez se jugaron 6 partidas siendo el vencedor el computador.

Ya en el 2006, Geoffrey Hinton presenta el concepto de Deep Learning o Aprendizaje Profundo. Con este concepto se explicaron los nuevos algoritmos que permiten que los computadores distingan diversos objetos y textos tanto en imágenes como en videos.

Conforme pasaron los años se fueron creando cada vez algoritmos más complejos, en donde ahora participan grandes empresas como Google, Amazon y Microsoft. En particular, el aprendizaje profundo ha logrado los siguientes avances, todo en áreas históricamente difíciles de aprendizaje automático:

1. Clasificación de imágenes a nivel casi humano
2. Reconocimiento de voz a nivel casi humano
3. Transcripción de escritura a mano a nivel casi humano
4. Traducción automática mejorada
5. Conversión de texto a voz mejorada
6. Asistentes digitales como Cortana, Alexa o Siri
7. Conducción autónoma casi humana
8. Mejora en las recomendaciones de Netflix, YouTube, Spotify, etc.
9. Resultados de búsqueda mejorados en la web
10. Capacidad para responder preguntas en lenguaje natural
11. Superhumano en juegos de mesa como el ajedrez o damas chinas

## III. MATEMÁTICAS EN EL APRENDIZAJE PROFUNDO

Para comprender las acciones de una red neuronal es necesario conocer las técnicas matemáticas involucradas durante el proceso, en el que se consideran matrices, vectores, derivadas y tensores, los cuales tienen un reconocimiento específico en el Aprendizaje Profundo.

## 3.1 Matrices

Una matriz es un arreglo rectangular con $m$ filas y $n$ columnas, donde sus $mn$ componentes son números reales, se llama matriz de orden o tamaño $m\ x\ n$**.**

Las matrices, por lo general, se denotan por las letras mayúsculas $A$, $B$. Cuando se nombra o escribe el tamaño u orden de la matriz primero se coloca la cantidad de filas y después la cantidad de columnas. El arreglo rectangular de una matriz se presenta entre paréntesis.

Por ejemplo, si:

$$A = \begin{pmatrix} 2 & -2 \\ -1 & -3 \\ 4 & 5 \end{pmatrix}$$

Entonces la fila 1 de A es $(2\ \ -2)$, la fila 2 es $(-1\ \ -3)$ y la fila 3 es $(4\ \ 5)$, mientras que las columnas son:

$$1\begin{pmatrix} 2 \\ -1 \\ 4 \end{pmatrix} y \quad 2\begin{pmatrix} -2 \\ -3 \\ 5 \end{pmatrix}$$

Para referirse a las componentes de una matriz se hace con la letra minúscula correspondiente a la letra que representa a la matriz, acompañada de dos subíndices que denotan la fila y columna correspondientes en la matriz. En general, en la matriz $A$ la componente de la fila i y columna j se representa por $a_{ij}$; entonces, si la matriz es de orden m y n tenemos:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

La suma entre dos matrices es posible si las dos pertenecen al mismo orden y a los números reales. En cambio, un producto entre matrices $A$ y $B$, solo se puede realizar cuando la cantidad de columnas de la matriz $A$ coincida con la cantidad de filas de la matriz $B$, entonces la matriz del producto tendrá el mismo número de filas de $A$ y el mismo número de columnas de $B$. Esto es, si $A$ es de orden $p\ x\ n$ y $B$ es de orden $n\ x\ q$ el orden de la matriz producto es $p\ x\ q$, el cual se obtiene sumando los productos de los elementos de la fila de $A$ por los elementos de la columna de $B$.

El producto de una matriz por un escalar, se obtiene al multiplicar cada uno de los elementos de la matriz por el escalar, si el escalar es -1 entonces se obtendrá la transpuesta de la matriz.

## 3.2 Derivadas

Isaac Newton desarrolló los principios del cálculo diferencial en su obra Methodus Fluxiorum et Serierum Infinitorum (1671). En ese trabajo, da los pasos precisos alrededor de los conceptos de función y de límite, que le permiten plantear matemáticamente cuando las cantidades varían infinitesimalmente y, de esta forma, describir el movimiento de un punto que traza una curva, situación que expresa de la siguiente forma:

*Por última proporción de cantidades evanescentes debemos entender el cociente de estas cantidades, no antes de que desvanezcan, ni después, pero tal como se van desvaneciendo. La parte infinitesimal pequeña en la que un fluente se incrementa por unidad de tiempo cero es el momento del fluente.*

El texto reseñado puede analizarse en dos partes; la primera habla de cantidades evanescentes, es decir que se desvanecen o tienden a ser infinitamente pequeñas. Esto se puede contrastar directamente de la definición de límite*: "[...] se dice que 'L' es el límite de la función '$f(x)$' cuando la variable 'x' tiende al valor 'a' si la diferencia entre '$f(x)$' y '$(L)$' puede hacerse tan pequeña como se desee…".* En un cociente de cantidades con estas características, completa el texto.

Mientras la segunda parte habla de una parte infinitesimalmente pequeña que incrementa las variables (el fluente indica la letra) por unidad de tiempo.

En este escenario se observa que, sobre la curva descrita por el movimiento, hay un par de puntos y una diferencia entre estas posiciones en una proporción de cantidades que se hace cada vez más pequeña (evanescentes tanto como se desee, según el concepto de límite) por unidad de tiempo; a partir de esta discusión, se puede plantear la ecuación que describe esa acción de movimiento:

$$\frac{f(x+\Delta)-f(x)}{(x+\Delta)-(x)}$$

Al observar la ecuación, destaca la diferencia entre las posiciones (no en vano el término cálculo diferencial) con la parte infinitesimal en que se incrementa, numerador, tiempo (unidad de tiempo) y su correspondiente incremento en el denominador. El significado de la ecuación, es que ésta corresponde con la pendiente de la recta definida por los dos puntos del movimiento estudiado por Newton; de esta manera, se observa la formación de una recta secante (por definición, que corta a la curva en dos puntos).

Adicionalmente a la interpretación matemática de pendiente, se tiene que en física esta misma relación define la velocidad promedio a que se mueve un objeto en la trayectoria planteada (cambio de posición entre tiempo). Pero ahora es interesante la propuesta de Newton: hacer que "Δ" sea infinitesimalmente pequeña. Esto tiene dos implicaciones; la primera es que, al acercarse así, $f(x+\Delta)$ y $f(x)$, la recta cortará a la curva sólo en punto y se convertirá en una tangente (por definición) a la curva. La segunda implicación es que, al acercarse el tiempo dado por $(x+\Delta)$ y $(x)$ en la misma condición, se tendrá ahora un instante. De esta forma, la propuesta de Newton plantea que su aproximación en el límite será la velocidad del móvil en un instante, esto es, velocidad instantánea.

Dicho de otra forma, será posible describir el comportamiento del punto en movimiento punto a punto, instante a instante.

De esta manera, es posible decir que la derivada de una función representa la recta tangente a una curva y es una herramienta que permite evaluar la forma en que se presenta el cambio en una función.

Existen varias reglas para calcular derivadas, como las reglas aritméticas o la regla de la cadena, la cual es una fórmula que indica cómo derivar la composición de funciones:

Sean $y = f(u), u = g(x)$, es decir:

$$y = f\big(g(x)\big) = (f \circ g)(x)$$

Si $g$ es diferenciable en $x$ y $f$ diferenciable en $g(x)$, entonces $(f \circ g)$ es diferenciable en $x$ y se cumple:

$$(f \circ g)'(x) = f'\big(g(x)\big)g'(x)$$

Esta expresión es llamada regla de la cadena (porque se deriva en una secuencia, en una cadena). Dicha fórmula también se expresa así:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

En la práctica, una función de red neuronal consiste en muchas operaciones de tensor encadenadas, cada una de las cuales tiene una derivada simple y conocida.

### 3.3 Vectores y Tensores

El vector es una cantidad que posee magnitud, dirección y sentido. Sin embargo, no es esta la noción matemática más general del vector. Los vectores son elementos de conjuntos denominados espacios vectoriales, si los elementos se combinan con números reales en una operación de multiplicación, se dice que el espacio vectorial es real.

En muchas disciplinas se identifican o diseñan objetos matemáticos cuyas reglas naturales de operación algebraica corresponden a las de un espacio vectorial real. A estos objetos no necesariamente se les asignan propiedades de magnitud, dirección y sentido.

Existen espacios vectoriales, cuyos elementos (los vectores) son las oscilaciones de un péndulo doble, los coeficientes de los polinomios de grado $n$, algún conjunto de polígonos en el plano, entre otros.

Todos los vectores de un espacio vectorial real pueden generarse a partir de unos cuantos vectores básicos mediante las operaciones de suma vectorial y producto de vectores por números reales.

Por otro lado, cuando se habla de un tema como la relatividad, rápidamente surge un concepto: el tensor. El tensor es un objeto matemático muy utilizado en física.

Un escalar es un objeto matemático definido por una variable. Por ejemplo, la temperatura es un escalar. Si un objeto lo definimos en tres dimensiones, tenemos entonces un vector con componentes $x, y, z$, mediante el cual le asignamos a un punto una dirección en el espacio.

Sin embargos no es suficiente con usar escalares y vectores para definir matemáticamente el mundo en el que vivimos. En 1899, el físico alemán Woldemar Voigt, presentó el concepto de tensor, aplicado a las tensiones en un cuerpo.

## IV. TENSORES EN EL APRENDIZAJE PROFUNDO

En general, todos los sistemas actuales de aprendizaje automático utilizan tensores como su estructura básica de datos. Los tensores son fundamentales para el campo, tan fundamentales que el TensorFlow de Google lleva su nombre.

En esencia, un tensor es un contenedor de datos, casi siempre datos numéricos. Entonces, es un contenedor de números y las matrices son tensores 2D. Por lo tanto, los tensores son una generalización de matrices a un número arbitrario de dimensiones, cabe resaltar que, en el contexto de los tensores, una dimensión a menudo se denomina eje.

Un tensor que contiene solo un número se llama escalar (o tensor escalar, tensor de 0 dimensiones, o tensor 0D). El número de ejes de un tensor también se denomina rango.

A un arreglo de números se le llama vector o tensor 1D y dicho tensor tiene exactamente un eje. Por ejemplo: un vector 5D tiene solo un eje y cinco dimensiones a lo largo de su eje, mientras que un tensor 5D tiene cinco ejes (y puede tener cualquier cantidad de dimensiones a lo largo de cada eje).

Un arreglo de vectores es una matriz o tensor 2D. Una matriz tiene dos ejes (a menudo referidos a filas y columnas). Se puede interpretar visualmente una matriz como una cuadrícula rectangular de números, donde las entradas del primer eje se llaman filas, y las entradas del segundo eje se llaman columnas.

Al empacar los tensores 3D en una matriz, puede crear un tensor 4D, y así sucesivamente. En el aprendizaje profundo, generalmente se manipulan tensores de 0D a 4D, aunque puede subir a 5D si se procesan datos de video.

### 4.1 Atributos De Tensor

Un tensor se define por tres atributos clave:

- Número de ejes o rango: por ejemplo, un tensor 3D tiene tres ejes y una matriz tiene dos ejes. Esto también se llama $ndim$ del tensor en las bibliotecas de Python como Numpy.

- Forma: esta es una orden de enteros que describe cuántas dimensiones tiene el tensor a lo largo de cada eje. Un vector tiene una forma con un solo elemento, como $(5,)$, mientras que un escalar tiene una forma vacía, $( )$.
- Tipo de datos (generalmente denominado $dtype$ en las bibliotecas de Python): este es el tipo de datos contenidos en el tensor; por ejemplo, el tipo de tensor podría ser float32, uint8, float64, etc. Se debe tener en cuenta que los tensores de cadena no existen en Numpy (o en la mayoría de las otras bibliotecas), porque los tensores viven en segmentos de memoria contiguos preasignados: y las cadenas, que son de longitud variable, impedirían el uso de esta implementación.

La selección de elementos específicos en un tensor se denomina corte de tensor.

En general, el primer eje (eje 0, porque la indexación comienza en 0) en todos los tensores de datos que se encuentran en el aprendizaje profundo será el eje de muestras (a veces denominado dimensión de muestras). Además, los modelos de aprendizaje profundo no procesan un conjunto de datos completo a la vez; más bien, dividen los datos en pequeños lotes.

Cuando se considera tal tensor de lote, el primer eje (eje 0) se denomina eje de lote o dimensión de lote.

### 4.2 Datos Manipulables En Redes Neuronales

Los datos que se manipulen casi siempre se incluirán en una de las siguientes categorías:

- Datos vectoriales: tensores 2D de forma (muestras, características)
- Datos de series temporales o datos de secuencia: tensores 3D de forma (muestras, pasos temporales, características)
- Imágenes: tensores de forma 4D (muestras, altura, ancho, canales) o (muestras, canales, altura, ancho)
- Video: tensores de forma 5D (muestras, cuadros, altura, ancho, canales) o (muestras, cuadros, canales, altura, ancho)

En el caso de las imágenes, suelen tener tres dimensiones: altura, ancho y profundidad de color. Los tensores de imágenes siempre son 3D. Así, un lote de 128 imágenes en escala de grises de tamaño 256 × 256 podría almacenarse en un tensor de forma (128, 256, 256, 1), y un lote de 128 imágenes en color podría almacenarse en un tensor de forma (128, 256, 256, 3).

Existen dos convenciones para las formas de los tensores de imágenes: la última convención de canales (utilizada por TensorFlow) y la convención de primeros canales (utilizada por Theano). El marco de aprendizaje automático TensorFlow, de Google, coloca el eje de profundidad de color al final: (muestras, altura, ancho, profundidad de color). Mientras tanto, Theano coloca el eje de profundidad de color justo después del eje del lote: (muestras, profundidad de color, altura, anchura). Mientras el marco de Keras proporciona soporte para ambos formatos.

## V. ESTRUCTURA DE LA NEURONA BIOLÓGICA

Desde que se empezó a conocer la anatomía y estructura del tejido nervioso, a partir de los trabajos de Ramón y Cajal (1911) en España, los investigadores trataron de conocer la forma en que este tejido y los órganos que constituye, especialmente el cerebro, procesan la información que reciben de los órganos receptores, para dar una respuesta adecuada a

sus estímulos. Aunque aún se está lejos de comprender el funcionamiento y la estructura del sistema nervioso, se conoce con cierto detalle la estructura de la neurona, como elemento básico del tejido nervioso y la forma como se estructura la corteza cerebral.

La neurona, como toda célula, consta de una membrana exterior $M$, que la limita y le sirve de órgano de intercambio con el medio exterior, de un citoplasma C, que es el cuerpo principal de la célula donde radica el grueso de sus funciones y de un núcleo $N$, que contiene el material genético de la célula.

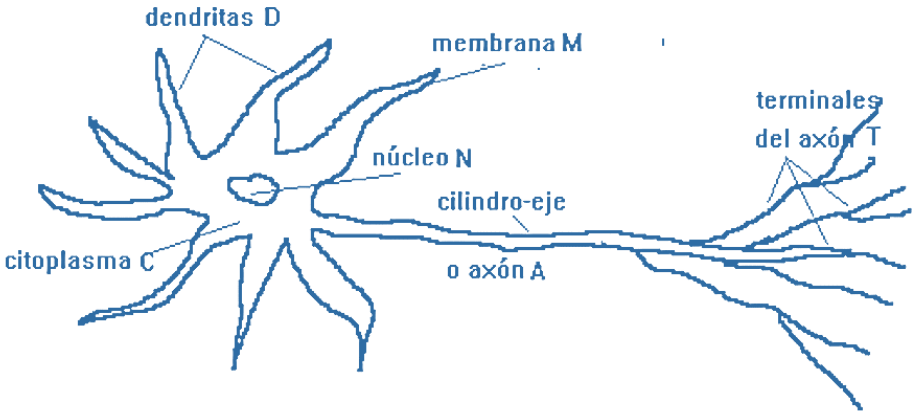



*Ilustración 5.1 Representación de una neurona biológica. Obtenida de http://conceptos.sociales.unam.mx/conceptos_final/598trabajo.pdf*

El citoplasma presenta unos alargamientos $D$, llamados dendritas, que son órganos de recepción. En las dendritas termina un gran número de fibras F que son conductores que llevan la señal o impulso nervioso de los receptores o de otras neuronas hacia la neurona. Estas fibras terminan en un pequeño corpúsculo llamado sinapsis, que constituye un relevador bioquímico y que sirve para transferir la señal de una neurona a otra. Existen dos clases de sinapsis: actuadoras, que favorecen el disparo de la neurona receptora e inhibidoras, que dificultan éste. Cuando se presenta un cierto desbalance entre las sinapsis actuadoras y las inhibidoras activas, la neurona dispara un impulso de salida, que constituye la respuesta de la neurona. Este impulso nervioso de salida es conducido por una prolongación cilíndrica alargada (hasta de varios decímetros de largo) de la neurona, que se llama cilindro eje o axón A, que en su extremo se divide en varias fibras para comunicarse con otras neuronas o con órganos efectores o motores como glándulas o músculos. El citoplasma de las neuronas forma la masa gris de los centros nerviosos y el conjunto de cilindros ejes forma la masa blanca de aquéllos.

En el cerebro humano la corteza cerebral contiene la mayor parte de las neuronas de este órgano y constituye, por ese hecho, la red neuronal natural más compleja. La corteza cerebral tiene un espesor promedio de 3mm y una superficie de unos 2000 $cm^2$.

En ella se sitúan unos cien mil millones de neuronas y de cien a mil billones de sinapsis. Es el centro superior de la memoria, del procesamiento de sensaciones, de la regulación motora, del lenguaje, de los afectos y de los mecanismos de inferencia. Está situada en la periferia del cerebro, formando pliegues llamados circunvoluciones cerebrales. En la corteza hay zonas especializadas en determinado trabajo, llamadas zonas o localizaciones cerebrales. Aunque su definición aún está en estudio, se tienen identificadas claramente las zonas motoras, las zonas del lenguaje, así como las zonas sensitivas correspondientes a los diferentes sentidos.

Conjuntamente con los descubrimientos de los neurofisiólogos y neuroanatomistas, algunos psicólogos comenzaron a desarrollar modelos neuronales de aprendizaje. Uno de los que tuvo mayor reconocimiento fue el de Donald O. Hebb, (1949) que propuso su ley de aprendizaje, que posteriormente fue el fundamento de los algoritmos de aprendizaje de las Redes Neuronales Artificiales. Hebb postuló que, si por alguna circunstancia dos neuronas se activan simultáneamente, se fortalecen las conexiones entre esas dos neuronas, haciendo que para una de ellas sea más fácil disparar, si la otra dispara. Esto explicaría la formación de reflejos condicionados en los animales. Esta teoría de Hebb dio lugar a nuevas teorías sobre la memoria y el aprendizaje. La parte del tejido nervioso más plástica es precisamente la sinapsis, donde la conductividad sináptica puede ser alterada por cambios en la composición química, en el tamaño y en la forma de dicha sinapsis. De esta forma puede explicarse la memoria como un conjunto de cambios en las sinapsis entre neuronas, en cuyo caso la información queda codificada por el número de interconexiones entre neuronas y el grado diferencial de conductividad de las sinapsis correspondientes.

En general, un modelo neuronal consta de:

- Un conjunto de entradas $x_1, \ldots x_n$.
- Los pesos sinápticos $w_1, \ldots w_n$, correspondientes a cada entrada.
- Una función de agregación, $\Sigma$.
- Una función de activación, $f$.
- Una salida, $y$.

En cambio, en una neurona biológica se pueden reconocer las siguientes partes:

- El cuerpo central, llamado soma, que contiene el núcleo celular.
- Una prolongación del soma, el axón.
- Un conjunto de ramificaciones terminales, las dendritas.
- Zonas de conexión entre una neurona y otra, conocidas como sinapsis.

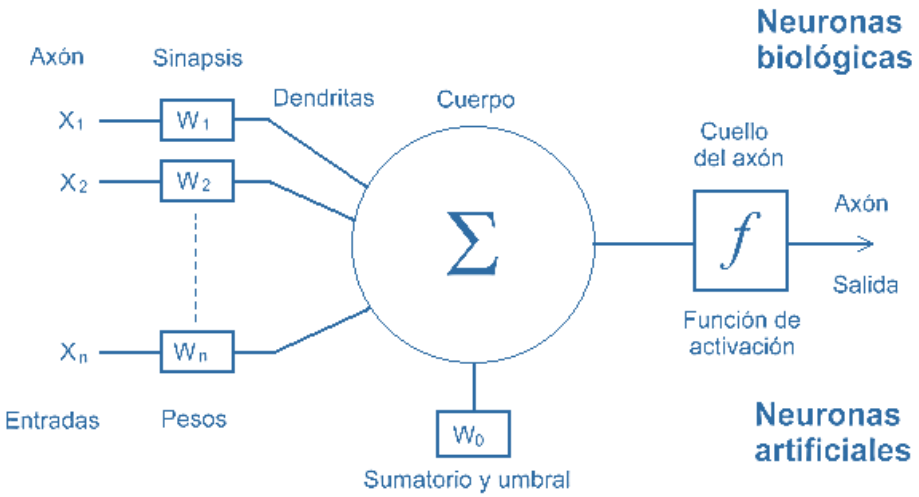



*Ilustración 5.2 Comparativa entre una red neuronal biológica y una artificial. Obtenida de http://www.cs.us.es/~fsancho/?e=72*

## VI. TÉCNICAS DE APRENDIZAJE EN EL APRENDIZAJE PROFUNDO

El Aprendizaje Profundo emplea dos tipos de técnicas: el aprendizaje supervisado, que entrena un modelo con datos de entrada y salida conocidos para que pueda predecir salidas futuras, y el aprendizaje no supervisado, que encuentra patrones ocultos o estructuras intrínsecas en los datos de entrada.

### 6.1 Aprendizaje Supervisado

Un algoritmo de aprendizaje supervisado toma un conjunto conocido de datos de entrada y respuestas conocidas para estos datos (salidas) y entrena un modelo con objeto de generar predicciones razonables como respuesta a datos nuevos.

El aprendizaje supervisado emplea las siguientes técnicas para desarrollar modelos predictivos.

- **Regresión.** La técnica es utilizada para clasificar valores continuos, la clase más común es la regresión lineal que ayuda a predecir cambios de temperatura.
- **Clasificación.** La técnica es usada para clasificar valores discretos, por ejemplo, clasificar imágenes de galaxias, solo si los datos se pueden etiquetar, categorizar o dividir en grupos o clases concretos.

## VII. EL PERCEPTRÓN

El perceptrón es la unidad básica de inferencia, usado para la clasificación binaria. El precursor del perceptrón fue la Unidad Lógica de Umbral (TLU) desarrollada por McCulloch y Pitts en 1943, que fue lograr de aprender a realizar las operaciones lógicas AND y OR. El algoritmo de entrenamiento del perceptrón es considerado aprendizaje supervisado.

El perceptrón fue inventado en el Laboratorio Cornell Aeronáutico, en Búfalo (Nueva York), por Frank Rosenblat, en 1957. Una de las primeras versiones, el Mark I, fue diseñada con el objetivo de que fuera capaz de reconocer imágenes para uso militar de la US Navy.

La unión sumadora previa a la función de activación que se encarga de sumar los valores de las entradas en proporción a los pesos sinápticos, se expresa de la siguiente manera:

$$\sum_{i=1}^{n} x_i \cdot w_i + b$$

donde:

$w$ es el vector de valores reales con los pesos de cada conexión
$w \cdot x$ es el producto escalar del peso de la conexión con el valor de entrada
$n$ es el número de entradas
$b$ es el sesgo o bias

El perceptrón es un modelo lineal de clasificación binaria que envía la suma de las $n$ entradas y sus pesos asociados a una función de activación. La salida de esta función será 0 o 1 dependiendo si el resultado es mayor o menor que 0.5, esto es debido a que la función de activación en el perceptrón es una función escalonada o función de umbral, la cual considera los siguientes criterios:

$$f(x) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$$

Con lo cual se obtiene la siguiente gráfica:

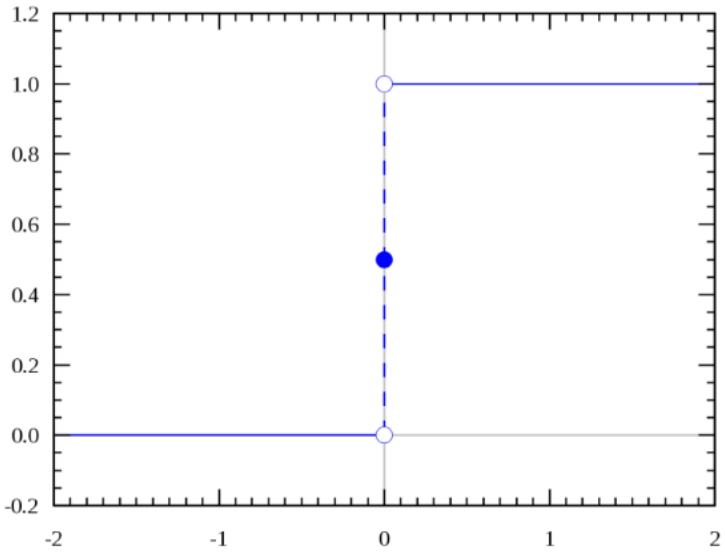



*Ilustración 7.1 Gráfica de la función escalonada del perceptrón. Obtenida del libro Deep Learning with Python de FRANÇOIS CHOLLET, 2018.*

El parámetro sesgo, en caso de ser negativo, obliga a los pesos a ser mayores para obtener un 1 como salida. No obstante, los valores de entrada no afectan al sesgo, pero este es aprendido a través del algoritmo del perceptrón. Primeramente, se inicializan los pesos del vector con valores reales aleatorios muy pequeños. El perceptrón calcula la salida de la clasificación y la compara con el valor de la etiqueta, si esta coincide, no se hacen cambios. En caso de que los valores no coincidan con la etiqueta, se ajustan los pesos proporcionalmente.

## VIII. FUNCIONES DE ACTIVACIÓN

Tanto en las redes neuronales artificiales como en las biológicas, una neurona no sólo transmite la entrada que recibe. Existe un paso adicional, una función de activación, que es análoga a la tasa de potencial de acción disparando en el cerebro. La función de activación utiliza la misma suma ponderada de la entrada anterior, es decir:

$$z = b + \sum_{i=1}^{n} w_i x_i$$

Y la transforma una vez más como salida.

### 8.1 Función Sigmoide

Históricamente, la función sigmoide es la función de activación más antigua. Se define como:

$$f(x) = \frac{1}{1 + e^{-x}}$$

$e$ denota la constante exponencial, que es aproximadamente igual a 2.718281828. Una neurona que utiliza la sigmoide como función de activación se le llama neurona sigmoide. Primero se establece que la variable $z$ equivale a la suma ponderada de entrada y después se pasa a través de la función sigmoide, quedando de la siguiente manera:

$$z = b + \sum_{i=1}^{n} w_i x_i$$

$$f(z) = \frac{1}{1 + e^{-z}}$$

Aunque la ecuación parece complicada y arbitraria, en realidad tiene una forma bastante simple:

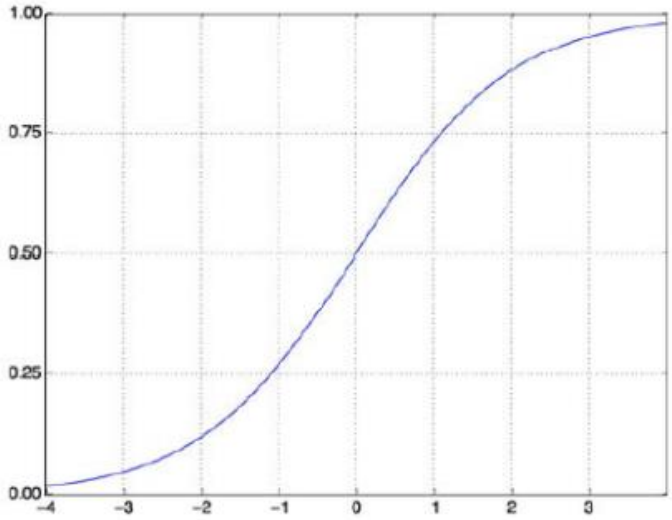



*Ilustración 8.1 Gráfica de la función sigmoide como función de activación. Obtenida del libro Deep Learning with Python de FRANÇOIS CHOLLET, 2018.*

Se puede ver que $f(z)$ actúa como función "aplastadora", comprimiendo la salida a un rango de 0 a 1. En el centro, donde $z = 0$, $f(0) = \frac{1}{1+e^0} = \frac{1}{2}$. Para valores negativos grandes de $z$, el término $e^{-z}$ en el denominador crece exponencialmente, y $f(z)$ se aproxima a 0. Al contrario, valores positivos grandes de $z$ reducen $e^{-z}$ hacia 0, y $f(z)$ se aproxima a 1.

La función sigmoide es continuamente diferenciable, y su derivada convenientemente es $f'^{(z)} = f(z)(1 - f(z))$.

En general, las características de la función sigmoide son las siguientes:

- Lenta convergencia.
- No está centrada en el cero.
- Esta acotada entre 0 y 1.
- Buen rendimiento en la última capa.

### 8.2 Función ReLU

La función de activación llamada rectified linear unit o ReLU (Unidad lineal rectificada) está destinada a poner a cero los valores negativos, mientras que un sigmoide "aplasta" los valores arbitrarios en el intervalo [0, 1]; generando algo que puede ser interpretado como una probabilidad y definido como $f(x) = \max(0, x)$, es decir:

$$f(x) = \begin{cases} 0\ para\ x < 0 \\ x\ para\ \ x \geq 0 \end{cases}$$

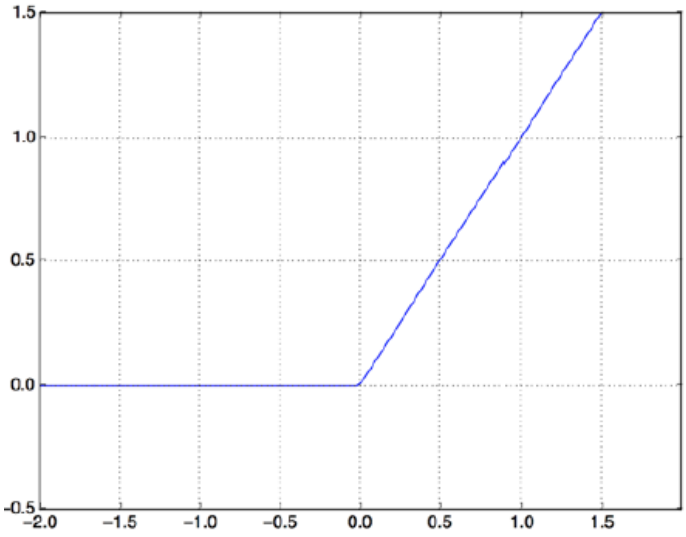



*Ilustración 8.2 Gráfica de la función ReLU, Obtenida del libro Deep Learning with Python de FRANÇOIS CHOLLET, 2018.*

Sin una función de activación como ReLU (también llamada no linealidad), la capa densa consistiría en dos operaciones lineales: un producto punto y una adición. Por lo tanto, la capa solo podría aprender transformaciones lineales de los datos de entrada: el espacio de hipótesis de la capa sería el conjunto de todas las posibles transformaciones lineales de los datos de entrada.

Tal espacio de hipótesis sería demasiado restringido y no se beneficiaría de múltiples capas de representaciones, porque una pila profunda de capas lineales aún implementaría una operación lineal, agregar más capas no ampliaría el espacio de hipótesis.

Para obtener acceso a un espacio de hipótesis mucho más rico que se beneficie de representaciones profundas, se necesita una no linealidad o función de activación. ReLU es la función de activación más popular en el aprendizaje profundo, pero hay muchos otros candidatos. A continuación, se enlistan las características importantes de esta función:

- Solo se activa si son positivos.
- No está acotada.
- Se pueden morir demasiadas neuronas.
- Se comporta bien con imágenes.
- Buen desempeño en redes convolucionales.

## IX. BIAS

También llamado sesgo, consiste en un valor aleatorio, aunque generalmente tiene un valor de 1, que permite cambiar la función de activación a la izquierda o a la derecha, y puede ser crítico para el éxito del aprendizaje.

Como se mencionó anteriormente, la salida de la red se calcula multiplicando la entrada $x$ y el peso $w_0$ y el resultado pasa a través de algún tipo de función de activación, pero si al momento de hacer la sumatoria en las neuronas no se añade el sesgo $b$, entonces todos los distintos valores que adquiera $w$, solo modificarán la inclinación de las funciones de activación sin poder desplazarla.

En cambio, si se añade un sesgo a la red, entonces las gráficas de las funciones tendrán la forma de su función de activación correspondiente. Además, esto implica que no se produce aprendizaje.

Una manera más simple de entender el sesgo, es de alguna manera similar a la constante $b$ de una función lineal:

$$y = ax + b$$

Permite mover a la línea arriba y abajo para el ajuste de la predicción con los datos mejor.

Sin b la línea siempre pasaría por el origen (0, 0).

## X. ALGORITMO DE PROPAGACIÓN HACIA ATRÁS

Este algoritmo también conocido como backpropagation o retropropagación se introdujo originalmente en la década de 1970, pero su importancia aumentó cuando un famoso artículo fue publicado por David Rumelhart, Geoffrey Hinton y Ronald Williams, permitiendo que las redes neuronales funcionaran más rápido y pudieran resolver problemas que en ese tiempo eran insolubles.

Para minimizar el error de una red neuronal siempre se busca ajustar los pesos de cada neurona, backpropagation indica la cantidad de error cometido por cada una de ellas, se le llama algoritmo de propagación hacia atrás porque del error cometido en la última capa lo va propagando hacia las capas anteriores.

El algoritmo exige poder derivar la función de activación, por eso se buscarán funciones de activación continuas cuya derivada exista y sea fácil de calcular.

## XI. GRADIENTE

Un gradiente es la derivada de una operación de tensor. Es la generalización del concepto de derivadas a funciones de entradas multidimensionales: es decir, a funciones que toman tensores como entradas.

El método de descenso por gradiente, es uno de los algoritmos de optimización más populares en aprendizaje automático, particularmente por su uso extensivo en el campo de las redes neuronales, es un método general de minimización para cualquier función.

Al algoritmo se le conoce por varios nombres, sobretodo en la literatura en inglés (vanilla gradient descent, batch gradient descent). A veces se les da el nombre de steepest descent, pero este término es más propio para aproximación analítica de integrales.

El algoritmo también tiene una versión gemela bizarra que en lugar de buscar por un mínimo busca el punto máximo de una función.

### 11.1 Gradiente Descendente

El método del Gradiente Descendente, también conocido como método de Augustin Louis Cauchy (1847), trata de encontrar un mínimo de una función, sea local o no. El método se basa en el uso del gradiente negativo $-\nabla f$, pues en esa dirección se obtiene el descenso máximo en los valores de la función.

Para explicar el gradiente conviene hacer uso las matemáticas y en concreto del cálculo diferencial. Si se tiene la función $f(x)$, el gradiente de la función en un valor particular de $x$, es la tasa de cambio de $f(x)$ cuando se cambia $x$, que puede aproximarse a $\frac{\Delta y}{\Delta x}$ para $\Delta x$ pequeño. Se puede escribir como:

$$\frac{\partial f(x)}{\partial x} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Que se conoce como la derivada parcial de $f(x)$ con respecto a $\Delta x$.

Cuando se modifique el resultado de la función $f(x)$, es evidente que se debe cambiar el valor de $x$, como esto depende del gradiente de $f(x)$ en el valor actual de $x$, se pueden encontrar tres casos:

- Si $\frac{\partial f}{\partial x} < 0$ entonces $f(x)$ disminuye a medida que $x$ aumenta.
- Si $\frac{\partial f}{\partial x} = 0$ entonces $f(x)$ está en un máximo o mínimo, no deberíamos cambiar $x$.
- Si $\frac{\partial f}{\partial x} > 0$ entonces $f(x)$ aumenta a medida que aumenta $x$.

En resumen, se puede disminuir $f(x)$ cambiando $x$ por un valor, es decir:

$$\Delta x = x_{nuevo} - x_{antiguo} = -\eta \frac{\partial f}{\partial x}$$

Donde:

- $\eta$ es una constante positiva que especifica cuánto cambia $x$. Es importante considerar que $\eta$ debe ser un valor adecuado para la función, si se toma un valor muy pequeño el cálculo del gradiente descendente podría ir muy lento, pero si se toma uno muy grande, se podría pasar del mínimo, es conocida como tasa de aprendizaje o tamaño del paso.
- La derivada $\frac{\partial f}{\partial x}$ indica cuál es la dirección que se sigue.

El método del gradiente descendente ha sido, y es, uno de los métodos más utilizados en el entrenamiento de las RNA (Redes Neuronales Artificiales). Existen dos formas de aplicarlo, por Lotes y por SGD (Stochastic gradient descent).

## XII. OPTIMIZADOR

Los optimizadores son fundamentales para el funcionamiento de las redes neuronales, ayudan a minimizar o maximizar una función objetivo implementando la propagación hacia atrás, permitiendo ajustar los pesos en la dirección que reduzca la pérdida del modelo.

En otras palabras, el optimizador mide el rendimiento de la red neuronal para encontrar una actualización de pesos que la optimice.

En este caso, se utilizó el optimizador SGD y RMSprop, los cuales se mencionarán a en los siguientes apartados.

### 12.1 Gradiente Descendente Estocástico (SGD)

Los primeros prototipos de los algoritmos SGD (Stochastic Gradient Descent) son los publicados por Herbert Robbins y Sutton Monro en 1951 con el título de “A Stochastic Approximation Method” y el posterior de Jacob Wolfowitz y Jack Kiefer publicado en 1952 como “Stochastic Estimation of a Regression Function”.

El término estocástico se refiere al hecho de que cada lote de datos se extrae al azar (estocástico es un sinónimo científico de aleatorio), de esta manera pueden existir dos formas para realizar las iteraciones (versiones):

- *Descenso del gradiente estocástico:* se introduce una única muestra aleatoria en cada iteración. El gradiente se calculará para esa muestra concreta, lo que supone la introducción de la deseada aleatoriedad, dificultando así el estancamiento. El problema de esta versión es su lentitud, ya que necesita de muchas más iteraciones, y además no aprovecha los recursos disponibles.
- *Descenso del gradiente (estocástico) en mini-lotes*: en lugar de alimentar la red con una única muestra, se introducen N muestras en cada iteración; conservando las ventajas de la segunda versión y consiguiendo además que el entrenamiento sea más rápido debido a la paralelización de las operaciones.

Además, existen múltiples variantes de SGD que difieren teniendo en cuenta las actualizaciones de peso anteriores al calcular la próxima actualización de peso, en lugar de solo ver el valor actual de los gradientes.

Hay Adagrad, RMSProp, SGD con momento, el cual recuerda el incremento aplicado a las variables en cada iteración y determina la siguiente actualización como una combinación lineal entre el gradiente y el incremento anterior, es decir aplica a los incrementos cierta inercia de forma que varíen más lentamente, y existen muchos otros.

### 12.2 RMSProp

Corresponde al Root Mean Square Propagation, en realidad es una variante del optimizador AdaGrad, en la que en lugar de mantener un acumulado de los gradientes, se consideran sólo los gradientes más recientes, además mantiene una media móvil del gradiente al cuadrado para cada peso de la red neuronal, dividiendo el gradiente.

## XIII. FUNCIÓN DE PÉRDIDA

También llamada función objetivo, es una función $L(z, y) \in R$ que toma como entrada el valor predicho $z$ y el valor real esperado $y$, dando como resultado qué tan diferentes son ambos. Esto quiere decir que la pérdida es un número que indica qué tan incorrecta fue la predicción del modelo en un solo ejemplo. Si la predicción del modelo es perfecta, la pérdida es cero; de lo contrario, la pérdida es mayor.

### 13.1 Error Cuadrático Medio (ECM)

Se refiere al promedio de la pérdida al cuadrado de cada ejemplo. Para calcular el ECM, primero se suman todas las pérdidas al cuadrado de los ejemplos individuales, es decir:

$$(observación - predicción(x))^2 = (y - y')^2$$

y, luego, se divide por la cantidad de ejemplos:

$$ECM = \frac{1}{N} \sum_{(x,y)\in D} (y - predicción(x))^2$$

donde:

$(x, y)$, es un ejemplo en el que:

$x$ es el conjunto de atributos que el modelo usa para realizar las predicciones.
$y$ es la etiqueta del ejemplo.
$predicción(x)$ es un atributo de las ponderaciones y las ordenadas al origen en combinación con el conjunto de atributos $x$.
$D$ es el conjunto de datos que contiene muchos ejemplos etiquetados, que son los pares $(x, y)$.
$N$ es la cantidad de ejemplos en $D$.

### 13.2 Cros-entropía Binaria

Entropía es una medida de la incertidumbre asociado con una distribución dada, entonces si se conoce la verdadera distribución de una variable aleatoria se puede calcular su entropía.

La cros-entropía binaria es una función de pérdida utilizada para problemas de clasificación de dos clases, del cual forma parte este proyecto de detección mareas, ya sea producto de interacciones visibles o no.

## XIV. CICLO DE ENTRENAMIENTO

El aprendizaje automático se trata de asignar entradas (como imágenes) a objetivos (o etiquetas), lo cual se hace al observar muchos ejemplos de entradas y objetivos. Las redes neuronales profundas realizan este mapeo de entrada a destino a través de una secuencia profunda de transformaciones de datos simples (capas) y estas transformaciones de datos se aprenden mediante la exposición a ejemplos.

La especificación de lo que una capa hace a sus datos de entrada se almacena en los pesos de la capa, que en esencia son un montón de números.

En términos técnicos, se dice que la transformación implementada por una capa está parametrizada por sus pesos, los cuales también se denominan a veces como los parámetros de una capa. En este contexto, aprender significa encontrar un conjunto de valores para los pesos de todas las capas en una red, de modo que la red asigne correctamente entradas de ejemplo a sus objetivos asociados. Una red neuronal profunda puede contener decenas de millones de parámetros, encontrar el valor correcto para todos ellos puede parecer una tarea desalentadora, dado que modificar el valor de un parámetro afectará el comportamiento de todos los demás.

Para controlar algo, primero se tiene que poder observarlo. Para controlar la salida de una red neuronal, se debe poder medir qué tan lejos está esta salida de lo que se esperaba. Este es el trabajo de la función de pérdida de la red.

La función de pérdida toma las predicciones de la red y el verdadero objetivo (lo que quería que produjera la red) y calcula un puntaje de distancia, capturando qué tan bien la red ha funcionado en el ejemplo específico.

El truco fundamental en el aprendizaje profundo es usar esta puntuación como señal de retroalimentación para ajustar un poco el valor de los pesos, en una dirección que disminuya la puntuación de pérdida para el ejemplo actual. Este ajuste es el trabajo del optimizador, que implementa lo que se llama el Algoritmo de Propagación hacia atrás: el algoritmo central en el aprendizaje profundo.

Inicialmente, a los pesos de la red se les asignan valores aleatorios, por lo que la red simplemente implementa una serie de transformaciones aleatorias. Naturalmente, su rendimiento está lejos de lo que debería ser idealmente, y el puntaje de pérdida es, en consecuencia, muy alto.

Pero con cada ejemplo que procesa la red, los pesos se ajustan un poco en la dirección correcta y el puntaje de pérdida disminuye.

Este es el ciclo de entrenamiento, que, repetido un número suficiente de veces (generalmente decenas de iteraciones en miles de ejemplos), produce valores de peso que minimizan la función de pérdida. Una red con una pérdida mínima es aquella para la cual las salidas están lo más cerca posible de los objetivos: una red capacitada. Una vez más, es un mecanismo simple que, una vez escalado, termina pareciendo mágico.

## XV. ARQUITECTURA DE UNA RED NEURONAL CONVOLUCIONAL

La topología o arquitectura de una red consiste en la organización y disposición de las neuronas en la red. Las neuronas se agrupan formando capas, la capa de entrada, las capas de convolución, las capas de reducción y las capas completamente conectadas.

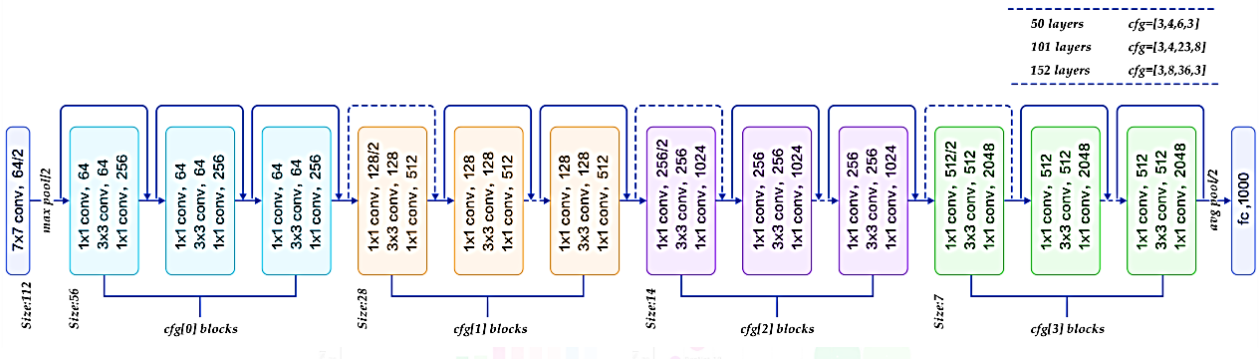



*Ilustración 15.1 Ejemplo de la arquitectura de una red neuronal convolucional. Obtenida de https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5?*

## XVI. TIPOS DE CAPAS

### 16.1 Capas de Convolución

Las capas de convolución son de mucha importancia en las arquitecturas de las redes neuronales convolucionales. Transforman los datos de entrada usando el producto escalar entre la región de las neuronas en la capa de entrada y los pesos asignados. Generalmente, la salida tiene unas dimensiones espaciales menores o iguales a las de su entrada, pero en ocasiones se incrementan el número de elementos en la tercera dimensión de la salida (profundidad).

#### 16.1.1 Convolución

Una convolución es una operación matemática que describe cómo fusionar dos conjuntos de información. La operación de convolución, es conocida como el detector de características de una CNN (Convolutional Neural Network). La salida de una convolución puede ser datos en crudo o un mapa de características salida de otra convolución, enseguida se muestra como el núcleo se desplaza a través de los datos de entrada para producir la capa convolucionada, en cada paso, el núcleo es multiplicado por los valores de entrada:

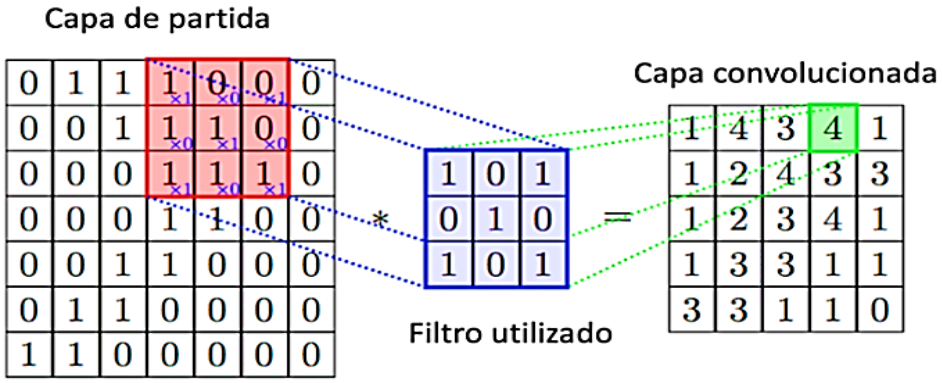



*Ilustración 16.1 Operación convolucional que genera un mapa de características. Imagen obtenida de Towards Data Science*

En la convolución se realizan operaciones de productos y sumas entre la capa de partida y los n filtros (o kernel) que genera un mapa de características. Las características extraídas corresponden a cada posible ubicación del filtro en la imagen original.

La ventaja es que el mismo filtro sirve para extraer la misma característica en cualquier parte de la entrada, con esto se consigue reducir el número de conexiones y el número de parámetros a entrenar en comparación con una red multicapa de conexión total.

Después de aplicar la convolución se les aplica a los mapas de características una función de activación. Para entrenar los parámetros de las capas de convolución, se utiliza el algoritmo del gradiente descendente. Cabe mencionar que las capas de convolución tienen varios componentes, los cuales se mencionarán en los apartados siguientes.

### 16.1.2 Elementos de las Capas de Convolución

#### 16.1.2.1 Filtros

Los filtros son una función que tienen una anchura y altura menor a las de la entrada. Los filtros, como pueden ser las convoluciones, son aplicados a lo largo del ancho y largo del volumen de entrada. El cálculo de la salida del filtro se realiza mediante el producto escalar del filtro y de la región de entrada.

#### 16.1.2.2 Mapas de Activación

Cuando un filtro se activa, quiere decir que el filtro deja pasar la información a través de sí. Si un filtro pasa a través de las dimensiones espaciales del volumen de entrada, se producirá una salida bidimensional llamada mapa de activación de ese filtro. Después, se crea una salida tridimensional para la capa de convolución apilando los mapas de activación en la dimensión profundidad de la salida.

### 16.1.2.3 Hiperparámetros Específicos de Cada Capa

Los hiperparámetros que definen la disposición espacial y el tamaño del volumen de salida de una capa de convolución son:

- Tamaño del kernel o del filtro: Se refiere a la anchura y altura del filtro.
- Profundidad de la salida: Controla el número de neuronas en la capa convolucional que están conectadas a la misma región del volumen de entrada.
- Paso: Es el hiperparámetro con el que se controla cuanto se va a desplazar el filtro por cada aplicación.

## 16.2 Capas de Reducción

Habitualmente insertadas entre capas de convolución sucesivas, las capas de convolución se encargan de reducir progresivamente el tamaño espacial de la representación de los datos. Con esta reducción se busca controlar el sobre entrenamiento. Las dos operaciones de submuestreo más comunes son Max Pooling y Average Pooling. La operación Max Pooling toma el mayor valor numérico del área filtrada, en cambio, la operación Average Pooling toma la media de los números del área filtrada.

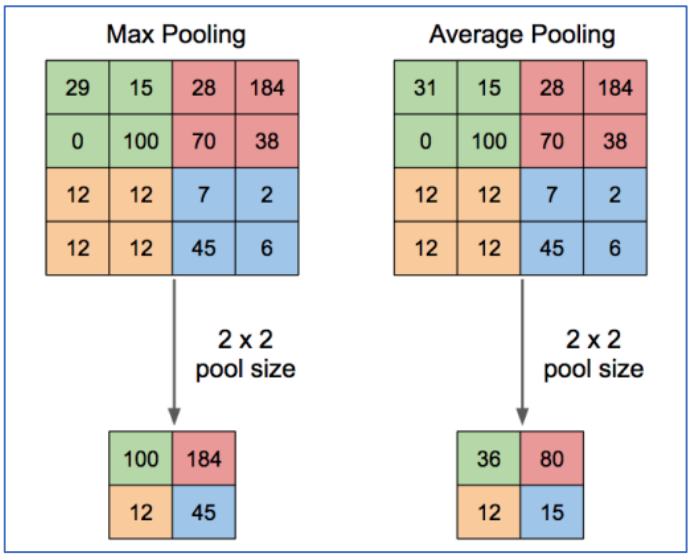



*Ilustración 16.2 Comparación de resultados entre Max Pooling y Average Pooling producto de operaciones de matrices. Obtenida del proyecto de Clasificación de Obras de Arte por Estilo Artístico Usando Redes Neuronales Convolucionales, de Ismael Pérez Roldán y Fernando Ortega Requena, 2018.*

### 16.2.1 Dropout

La capa de exclusión o dropout consiste en abandonar aleatoriamente (establecer en cero) una serie de características de salida de la capa durante el entrenamiento y la tasa de abandono es la fracción de las características que se ponen a cero; generalmente se establece entre 0.2 y 0.5, pero el parámetro toma los valores de 0 a 1, si los valores son cercanos a 0 se desactivarán menos neuronas, si es cercano a 1 se desactivarán más neuronas. Cabe aclarar que para las capas de entrada se suele excluir con un valor muy alto.

Esta exclusión de neuronas ayuda a reducir el sobreajuste, ya que las neuronas cercanas suelen aprender patrones específicos con los datos de entrenamiento, con dropout esta dependencia entre neuronas es menor en toda la red neuronal, de esta manera las neuronas trabajan de forma solitaria y no dependen de las relaciones con las neuronas vecinas.

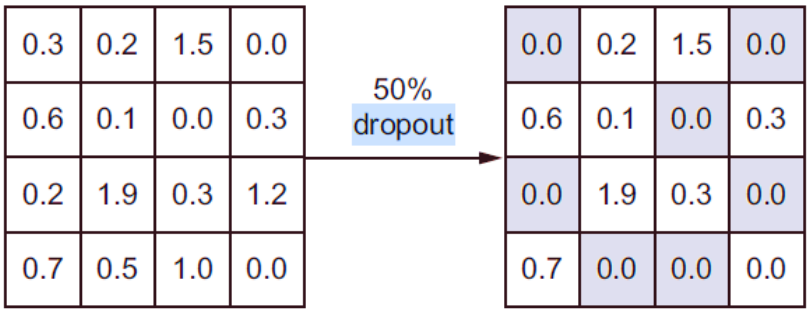



*Ilustración 16.3 Representación de las neuronas apagadas con Dropout en un 50%. Obtenida del libro Obtenida del libro Deep Learning with Python de FRANÇOIS CHOLLET, 2018.*

## 16.3 Clasificación De Las Capas A Nivel General

En un nivel más amplio las capas se pueden dividir en tres tipos:

*Capa de entrada:* corresponde a la primera capa en conectarse con las siguientes capas, en este caso los datos corresponden a los valores numéricos de las neuronas una vez ya aplanados.

*Capa intermedia o capas oculta:* se refiere a todas las capas que están entre la capa de entrada y la capa de salida, cuando existen dos o más capas intermedias es cuando se habla de Aprendizaje Profundo. Estas capas son las que van a procesar los los datos de las imágenes y pueden trabajar con tantas neuronas como se desee. Si a las capas no se les aplica alguna técnica de reducción entonces se habla de capas densamente o completamente conectadas, o simplemente capas densas, es decir, todas las neuronas están conectadas con todas las neuronas de la siguiente capa.

*Capa de salida:* las operaciones realizadas en esta capa son las mismas que en las ocultas, la diferencia es que estas no utilizan los datos de entrada, sino las capas ocultas. Generalmente la capa de salida tiene unas cuantas neuronas o una neurona que mediante una función de activación predice los resultados.

## XVII. HERRAMIENTAS DE DESARROLLO DEL APRENDIZAJE PROFUNDO

Para el desarrollo de una aplicación de Aprendizaje Profundo existen una gran cantidad de herramientas que se pueden utilizar, a continuación, se citan algunas de las más utilizadas:

- TensorFlow, Theano y CNTK (The Microsoft Cognitive Toolkit): Plataformas principales para desarrollar aplicaciones; Google, la Universidad de Montreal, Microsoft, son los creadores respectivamente.
- Keras: Herramienta que facilita el desarrollo sobre las plataformas principales.
- CUDA y cuDNN: Plataformas para acelerar el entrenamiento mediante la GPU.
- Cuaderno Jupyter: Herramienta para realizar experimentos de aprendizaje profundo.

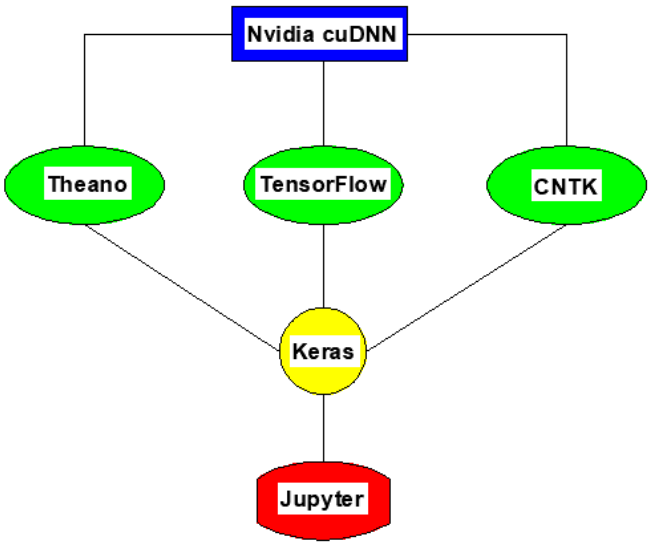



*Ilustración 17.1 Diagrama jerárquico de las herramientas del Aprendizaje Profundo. Creación propia.*

### 17.1 NVIDIA cuDNN

La biblioteca de red neuronal profunda NVIDIA CUDA (cuDNN) es una biblioteca acelerada por GPU (Unidad de Procesamiento Gráfico) de primitivas (tipos de datos originales de un lenguaje de programación) para redes neuronales profundas.

cuDNN proporciona implementaciones altamente ajustadas para rutinas estándar como convolución hacia adelante y hacia atrás, agrupación, normalización y capas de activación.

Los investigadores de aprendizaje profundo confían en cuDNN para la aceleración de GPU de alto rendimiento.

### 17.2 TensorFlow

TensorFlow es una biblioteca de código abierto programado en C++ y Python y dirigida al Aprendizaje Automático a través de una serie de tareas. Ha sido desarrollado por Google para satisfacer las necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Actualmente es utilizado tanto para la investigación como para la producción de productos de Google, remplazando el rol de su predecesor de código cerrado, DistBelief. TensorFlow fue originalmente desarrollado por el equipo de Google Brain para uso interno en Google antes de ser publicado bajo la licencia de código abierto Apache 2.0 el 9 de noviembre de 2015.

### 17.3 Keras

Etimológicamente Keras (κέρας) significa cuerno en griego. Es una referencia a una imagen literaria de la literatura griega y latina antigua, encontrada por primera vez en la Odisea.

Keras se desarrolló inicialmente como parte del esfuerzo de investigación del proyecto ONEIROS (Sistema Operativo de Robot Inteligente Neuroelectrónico Abierto).

Se trata de una API (Interfaz de Programación de Aplicaciones) de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow, CNTK o Theano. Fue desarrollado con el enfoque de permitir la experimentación rápida, es decir, para poder pasar de la idea al resultado con el menor retraso posible, lo cual es clave para hacer una buena investigación.

Las ventajas de usar Keras es que:

- Permite la creación de prototipos fácil y rápida (a través de la facilidad de uso).
- Admite redes convolucionales y redes recurrentes, así como combinaciones de las dos.
- Se ejecuta sin problemas en CPU y GPU.

### 17.4 Cuadernos Jupyter

El cuaderno Jupyter forma parte del proyecto Jupyter, el cual desarrolla software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación.

Los cuadernos Jupyter son ampliamente utilizados en las comunidades de ciencia de datos y aprendizaje automático, son una excelente manera de realizar experimentos de aprendizaje profundo. Un cuaderno es un archivo generado por la aplicación Jupyter Notebook que puede funcionar en un navegador. Combina la capacidad de ejecutar código Python, además, también permite dividir los experimentos largos en piezas más pequeñas que se pueden ejecutar de forma independiente, lo que significa que se puede ir ejecutado por partes y si ocurre algún error, se puede volver a ejecutar desde donde salió mal el experimento.

## XVIII. IMÁGENES DESI

Las imágenes que se utilizaron para realizar el entrenamiento, validación y prueba de la red neuronal son de DESI (Dark Energy Spectroscopic Instrument), un instrumento montado en el Telescopio Mayall de 4 metros, ubicado en el Observatorio Nacional de Kitt Peak en Arizona, el cual forma parte de un proyecto para descifrar los misterios de la energía oscura en el universo.

Los sistemas de datos de DESI corresponden a los objetivos para los cuales fueron creados, además varían en su procesamiento de datos y transferencia, así como la distribución de los mismos. De esta manera, se producen datos de los surveys de Legacy Imaging, WISE y Gaia Satellites.

Los surveys consisten en emplear grandes telescopios que permiten estudiar una parte del cielo para producir enormes catálogos de información. Algunos surveys más destacados en la actualidad son SDSS (Sloan Digital Sky Survey) y DESI.

Los surveys DESI Legacy Imaging son en realidad una combinación de tres proyectos públicos: DECaLS (The Dark Energy Camera Legacy Survey), BASS (The Beijing-Arizona Sky Survey) y MzLS (The Mayall z-band Legacy Survey), que tienen el objetivo de crear una imagen conjunta de aproximadamente 14000 grados cuadrados del cielo extragaláctico visible desde el hemisferio norte en tres bandas ópticas: g, r, z, utilizando telescopios en el Observatorio Nacional de Kitt Peak y el Observatorio Interamericano del Cerro Tololo.

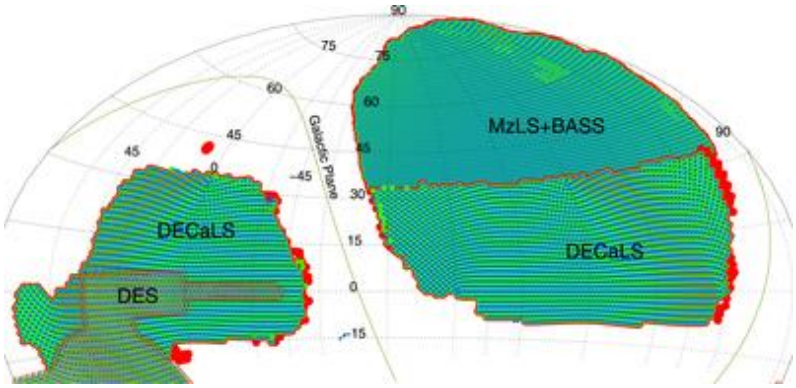



*Ilustración 18.1 Partes de cielo estudiadas mediante proyectos DESI. Obtenida de http://www.astroexplorer.org/details/ajab089df2.*

Las galaxias que se usaron en este trabajo pertenecen al catálogo de Nair (2010), un catálogo hecho con imágenes del proyecto SLOAN para obtener su clasificación morfológica. El catálogo está conformado por 15000 galaxias, de las cuales 12000 se clasificaron visualmente para este proyecto usando imágenes de DESI con la finalidad de encontrar estructura de marea en ellas, para esto, se tomaron en cuenta los siguientes criterios de clasificación: marea fuerte, marea en forma de capas (shells), y marea producto de la interacción entre galaxias. Por esa razón solo

## XIX. CREACIÓN DE LA RED NEURONAL

EL tamaño que tienen las imágenes es de 800x800 pixeles a color, en las bandas: r, g, z, los cuales son filtros que solo perciben la luz en un rango determinado.

En este trabajo también se tomaron en cuenta las imágenes astronómicas residuales para mejores resultados, pero estas no se han utilizado para el proceso de aprendizaje de esta red, pero se utilizarán en un trabajo futuro. Algunos ejemplos de las imágenes consideradas para el entrenamiento de la red neuronal son las siguientes:

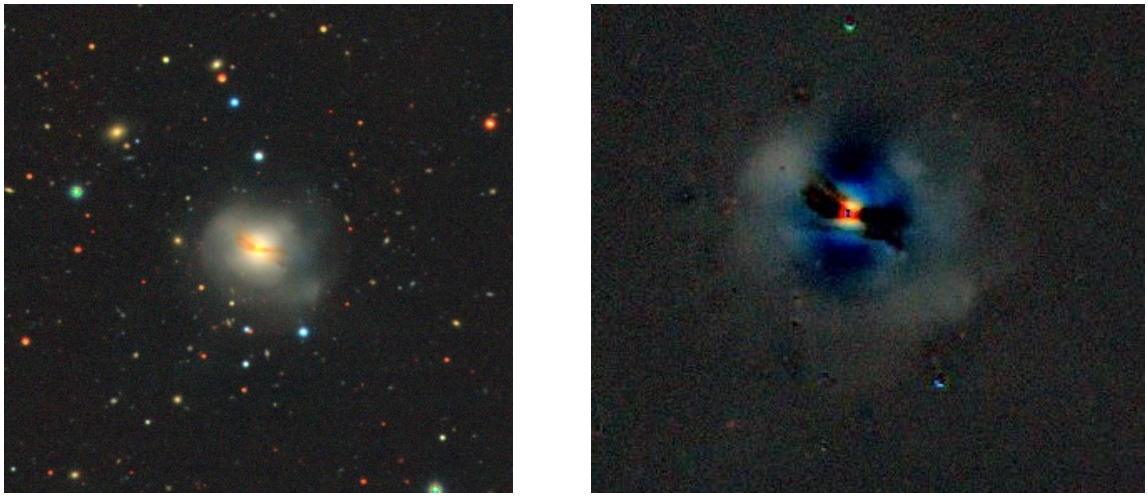


*Ilustración 19.1 Izquierda: Galaxia J083125.79+405722.3 RGB. Derecha: Residual de la galaxia J083125.79+405722.3. Obtenidas de DESI Legacy Imaging Survey.*

Como se aprecia en las imágenes anteriores, hay un anillo residual en el núcleo elongado, pero en este caso, solo importa la estructura de marea que presenten las galaxias y el residual, que permite distinguir otras formas de marea incluso relativamente alejadas alrededor de las galaxias.

Las siguientes imágenes son un ejemplar de las galaxias que se consideraron de interacción con marea:

*Ilustración 19.2 A la izquierda se muestra la Galaxia J102546.26+134300.6 interactuando con galaxia cercana al NE. A la derecha, en el residual se aprecia mejor la interacción con galaxias pequeñas al SW. Obtenidas de DESI Legacy Imaging Surveys.*

A continuación, otra de las galaxias utilizadas para el entrenamiento de la red neuronal con capas a su alrededor:

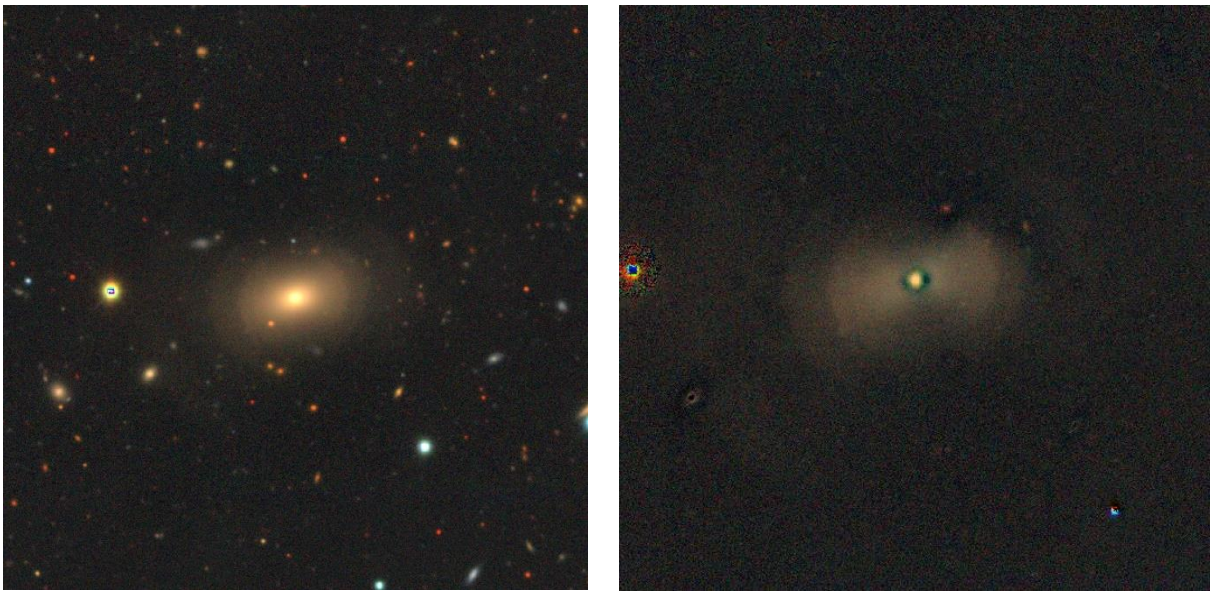
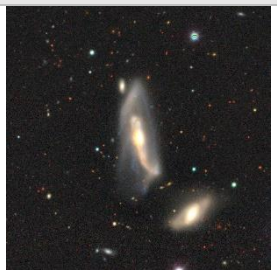
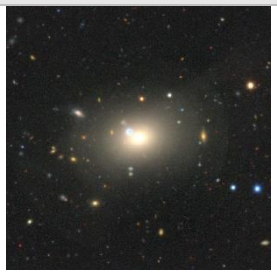
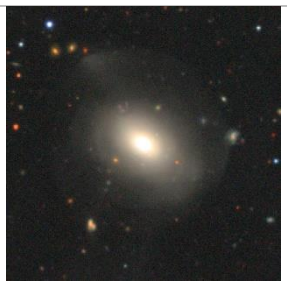
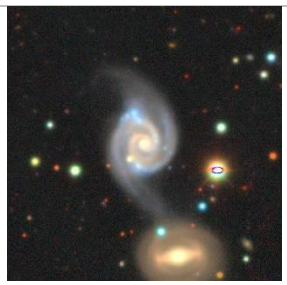


*Ilustración 19.3 Izquierda: Galaxia J094552.53-000534.1 con shells al E y W. Derecha: Residual con shells externos ligeramente más visibles que en la imagen RGB. Obtenidas de DESI Legacy Imaging Surveys.*

Con características similares a las de las anteriores galaxias de entrenamiento se formaron agrupaciones de datos de validación y prueba, algunos ejemplares son los siguientes:

| Conjunto de datos | Galaxias con marea fuerte | Galaxias con shells | Galaxias interactuando con marea |
|---|---|---|---|
| Validación |  *J020349.49+144416.9* |  *J015127.09-083019.3* |  *J014448.00-102726.9* |
| Prueba |  *J233841.20+155716.6* |  *J222614.63+004003.9* |  *J212859.44+112257.0* |

En este punto cabe aclarar que las imágenes de prueba también se clasificaron visualmente, ya que esas se utilizaron para evaluar el rendimiento predictivo de la red neuronal, así como las de validación que permiten ajustar el modelo.

Los resultados obtenidos de la anterior clasificación fueron en total 414 galaxias con marea, el número de datos agrupados para la red neuronal se muestra en la siguiente tabla:

| Galaxias | Entrenamiento | Validación | Prueba |
|---|---|---|---|
| Con marea | 331 | 42 | 41 |
| Sin marea | 800 | 100 | 100 |

En total se utilizaron 1131 imágenes de entrenamiento, 142 imágenes de validación y 141 imágenes de prueba, el número de imágenes para cada categoría fue elegido de acuerdo al porcentaje generalizado que se les ha asignado a otros modelos de redes neuronales convolucionales para obtener buenos resultados. En este caso, el 80% de los datos son datos de entrenamiento, el 20% se divide entre los datos de validación y prueba. Como solo se encontraron 414 imágenes de galaxias con marea, se escogieron aleatoriamente 800 imágenes de galaxias sin marea para balancear el número de las dos categorías y comenzar a hacer el entrenamiento.

Para el entrenamiento se utilizó Keras en TensorFlow, ambos descritos anteriormente, y se establecieron los directorios correspondientes para cada agrupación de datos, se comenzó a crear la red neuronal desde el principio, estableciendo las capas Conv2D, MaxPooling2D, las capas densas y las funciones de activación.

Las imágenes fueron redimensionadas al aplicar los filtros, pero también fueron reducidas antes de aplicar los filtros para no saturar el trabajo de la GPU, de cualquier manera, las imágenes siguen teniendo la misma información para ser entrenadas.

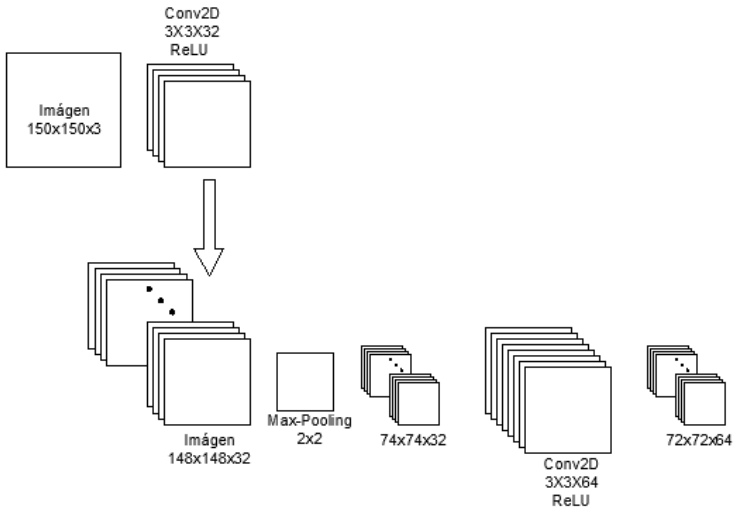



*Ilustración 19.4 Las imágenes son reducidas en una columna y fila de pixeles al aplicar el filtro de la capa convolucional, los pixeles son reducidos a la mitad al aplicar Max Pooling. Creación propia.*

Con la convolución se pueden detectar patrones como líneas o curvas y entre más capas convolucionales se vayan agregando, patrones más complejos podrán ser detectados. Para este proceso la convolución se apoyó de varios kernels con la finalidad de extraer las características más importantes de las imágenes, con lo cual resulta un mapa de características o mapa de activación, durante el proceso se pueden obtener varios mapas de características. Cabe aclarar que el número de filtros resultó de $2^n$.

Después de haber extraído las características, ahora se aplica la función ReLU, la cual va a cambiar a ceros todos los resultados del mapa de activación que sean negativos. La imagen resultante tendrá la profundidad que tenía el filtro, en este caso los kernels forman el filtro.

Ahora es cuando conviene aplicar la capa Max Pooling, que también permite extraer las características importantes, pero reduce las imágenes a la mitad en este caso, porque el filtro utilizado es de 2x2, de nuevo se aplica otra capa de convolución, cabe aclarar que los kernels de la capa de convolución conviene aplicarlos de 3x3 porque se ha probado que se obtienen mejores resultados, al mismo tiempo el número de kernels se fue modificando porque teóricamente se pueden extraer más características importantes.

Como se puede ver en la imagen anterior los filtros van redimensionando las imágenes, así como su número. En la siguiente imagen se muestra la continuación de la arquitectura de la red propuesta:

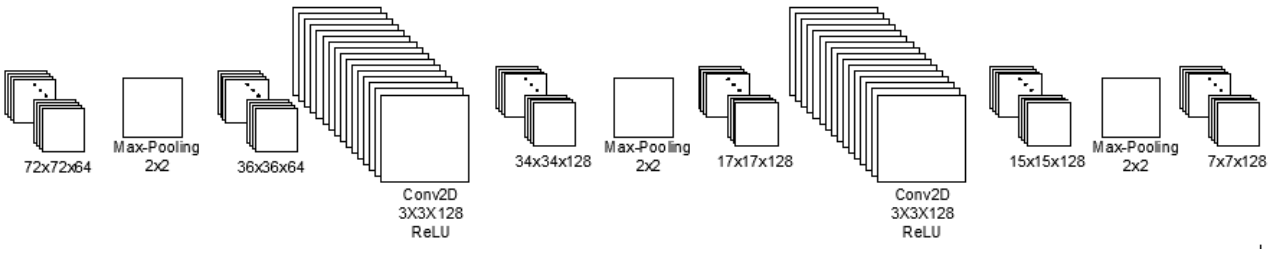



*Ilustración 19.5 A medida que se iban agregando más capas de convolución y Max Pooling las imágenes se fueron reduciendo. Creación propia.*

Aunque la imagen se está haciendo más pequeña, las propiedades o características importantes se siguen conservando, lo cual ayuda para que el entrenamiento no consuma muchos recursos computacionales.

Una vez que se obtuvo la última capa con imágenes de resolución 7x7 pixeles y una profundidad de 128 por los mapas de características, falta aplanar los resultados, es decir, que la imagen deje de ser tridimensional y constituya un vector o una imagen de una dimensión, que serán los datos de entrada o la capa de entrada. Una vez listas las neuronas se añadió una capa densa de 512 neuronas con la activación ReLU, y otra capa con una salida y función de activación sigmoide que diera el resultado del entrenamiento.

El modelo cuenta con una función de pérdida binary crossentropy y el optimizador RMSprop con una tasa de aprendizaje de $1x10^{-4}$ y una métrica accuracy.

Lo que hace la métrica utilizada es sumar los verdaderos positivos y los verdaderos negativos para después dividirlos entre el número total de ejemplos. Los verdaderos positivos corresponden a los ejemplos en los que la red predijo correctamente la clase positiva, y el verdadero negativo cuando se predijo correctamente la clase negativa. Si este proyecto no se tratara de una clasificación binaria entonces la clasificación de clases múltiples en accuracy o precisión resultaría de la división de las predicciones correctas entre el número total de ejemplos.

Las imágenes son normalizadas, es decir, los valores de los pixeles de las imágenes en RGB que van de 0 a 255 pasan al rango de 0 a 1, esto es conveniente para el preprocesamiento, porque la mayoría de los algoritmos de Aprendizaje Automático tienen mejor rendimiento cuando se trabaja en la misma escala. También como se había especificado antes, todas las imágenes se redimensionan a 150x150 pixeles, en este caso se trabaja con una clasificación binaria, por lo tanto, también se especifica. Se aplica lo mismo para los datos de validación y arroja un resultado de 1131 imágenes de entrenamiento y 142 imágenes de validación, ambas para predecir en dos clases. Las imágenes se agrupan en lotes de 20 para el entrenamiento.

El marco de aprendizaje de TensorFlow representa entonces la convención propuesta (20, 150, 150, 3), lo cual quiere decir que el entrenamiento se va a realizar en lotes de 20 con imágenes de 150x150 pixeles y profundidad de color, en este caso RGB.

Ahora corresponde a entrenar la red neuronal, para esto se especificaron los pasos por época, de los que se dieron 50 igual que el número de épocas, y 7 pasos de validación.

Las épocas se refieren al número de iteraciones sobre los datos de entrenamiento, los pasos de validación están relacionados a los lotes de validación e imágenes de validación, teóricamente el número de pasos son el resultado del número total de imágenes de validación entre los lotes de validación, por eso se consideraron 7 pasos de validación, los pasos por época son en realidad los pasos del descenso de gradiente.

El resultado generado del entrenamiento se guardó en un archivo .h5, y una vez hecho esto, se procedió a imprimir las gráficas, que demuestran cómo se está comportando la red, visualmente no se apreciaba bien la tendencia en la precisión y pérdida de los datos de entrenamiento y validación. Uno de los problemas de típicos cuando se trabaja con pocas imágenes de entrenamiento, se produce un sobreajuste.

El sobreajuste es causado por tener muy pocas muestras de las cuales aprender, lo cual hace que el modelo solo sea funcional para los datos que se le han dado, pero no es funcional cuando se agregan nuevos datos. En cambio, si las muestras dadas a la red fueran demasiadas, la red nunca se sobreajustaría.

### 19.1 Mejorando La Red Con El Aumento De Datos

Con el aumento de datos se pueden generar más datos de entrenamiento a partir de datos ya existentes, esto se logra con una serie de transformaciones aleatorias que se ejecutan con comandos en la plataforma Jupyter, lo que sucede es que el modelo no ve la misma imagen dos veces, así ayuda a generalizar mejor la red neuronal.

En este apartado las imágenes fueron rotadas 180 grados, se les aplicó un acercamiento, un desplazamiento ligero y una vuelta (flip), esto es también muy conveniente para que la red no memorice los datos de entrenamiento.

El nuevo modelo se cambió a una arquitectura con más capas densas después del aumento de datos quedando de la siguiente manera:

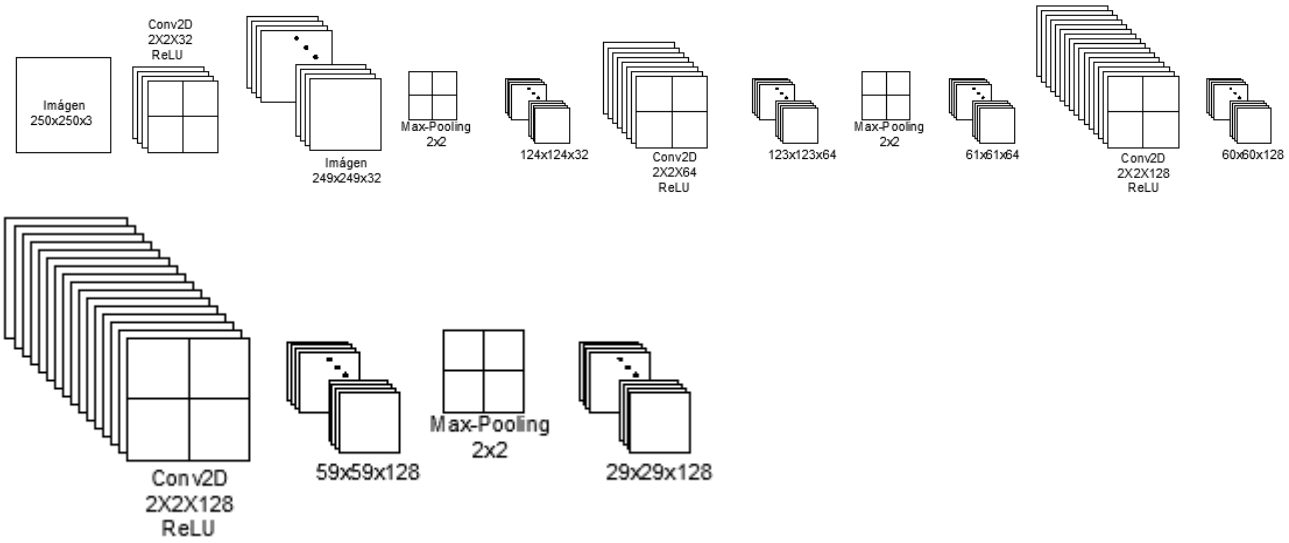



*Ilustración 19.6 Arquitectura elegida después del aumento de datos con imágenes de entrada de 250x250 pixeles. Creación propia.*

El aplanado de los datos dio lugar a 107648 neuronas de entrada para una imagen, es por eso que enseguida se les aplicó el dropout de 0.5, es decir, se apagaron la mitad de las neuronas, enseguida se aplicaron otros filtros, en este caso, capas densas con funciones de activación ReLU y la función sigmoide para la última capa.

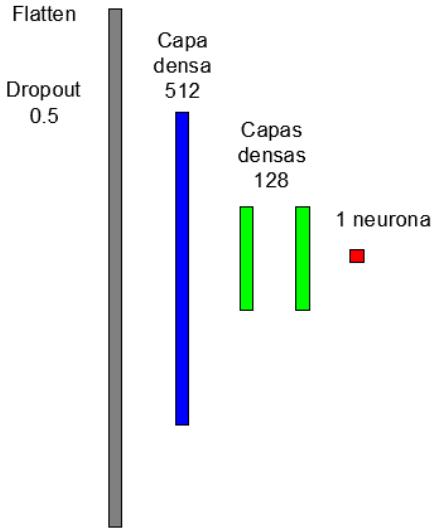



*Ilustración 19.7 Representación de la arquitectura de la red neuronal de acuerdo al número de capas. Creación propia.*

Se especificaron de nuevo los siguientes datos para compilar el modelo:

| | |
|---|---|
| Función de pérdida | Binary crossentropy |
| Optimizador | RMSprop |
| Métrica | Accuracy |

Las reglas exactas que rigen el uso del descenso de gradiente están definidas por el optimizador RMSprop (Root Mean Square Propagation), un método de tasa de aprendizaje adaptativo y que puede dividirla por un promedio exponencialmente decreciente de gradientes cuadrados.

Enseguida se realiza nuevamente el entrenamiento, pero ahora con 100 pasos por época, 7 pasos de validación y 50 épocas con 20 lotes.

En esta parte es cuando se toman en cuenta los datos de prueba, con los cuales la red neuronal va a predecir las mareas en las galaxias. Los resultados fueron los siguientes:

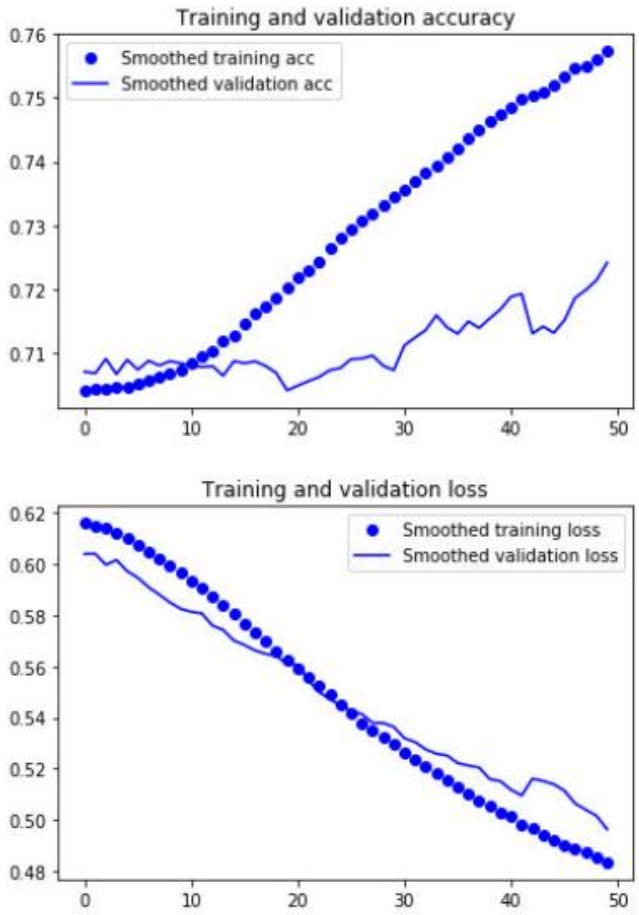



*Ilustración 19.8 Gráficas que describen el comportamiento de los datos de entrenamiento y validación. Obtenida desde un cuaderno Jupyter como resultado del entrenamiento, validación y prueba.*

Como se observa en las gráficas, existe una tendencia para cada una, lo que significa que la red sí aprende. En la primera gráfica se muestra en el eje horizontal el número de épocas y en el eje vertical la precisión y la pérdida respectivamente. Los puntos azules pertenecen a los datos de entrenamiento, las líneas azules se refieren al comportamiento de las imágenes de validación, la precisión no es tan buena, se puede ver que la red no estaba aprendiendo durante las primeras épocas, pero la pérdida cada vez se hacía más pequeña en la segunda gráfica.

Ya obtenidas las gráficas anteriores, es momento de comprobar qué tanto aprende la red, para eso se usaron las imágenes de prueba, imágenes que la red no había visto, se les especificó un tamaño de 250x250 pixeles y un tamaño de lote de 20, también como para los otros grupos de imágenes: binary, la clase que se utilizó para la clasificación, dando como resultado una precisión de 0.778, lo que significa que la red puede clasificar imágenes, pero solo con esa precisión.

Como consecuencia del resultado arrojado por la red, se hizo conveniente hacer la clasificación con una red preentrenada, con la cual ya no se comenzaría la red desde cero.

## 19.2 Red Preentrenada

La efectividad de una red preentrenada es muy importante al tratar con problemas de datos pequeños. Se trata de una red guardada que ya ha sido entrenada generalmente en clasificación de imágenes a gran escala, por lo tanto, también se ha generalizado para actuar como modelo en problemas diferentes de visión por computadora a pesar de que los nuevos problemas a resolver traten de clases muy distintas, por lo cual es una ventaja para el Aprendizaje Profundo.

### 19.2.1 VGG19

VGG-19 es una red neuronal convolucional preentrenada con más de un millón de imágenes de 224x224 pixeles a color de la base de datos ImageNet. La red tiene 19 capas de profundidad y puede clasificar imágenes en 1000 categorías de objetos, como teclado, mouse, lápiz, y muchos animales. Como resultado, la red ha aprendido representaciones de características para una amplia gama de imágenes.

En el proceso, la red preentrenada fue importada de Keras usando TensorFlow y estableciéndola como base, dicha red consta de 5 bloques entre capas convolucionales y Max Pooling. Una vez importada la red se establecieron los directorios a los cuales tendrá acceso y se agregó después un padding o relleno, el cual hace que a las imágenes se les agregue una capa de pixeles alrededor para que la información contenida en ellas no se pierda al ocupar el filtro de la capa convolucional.

Una vez hecho lo anterior se creó el modelo, esta vez con una arquitectura más pequeña, con menos capas que antes, pero con VGG19 como base.

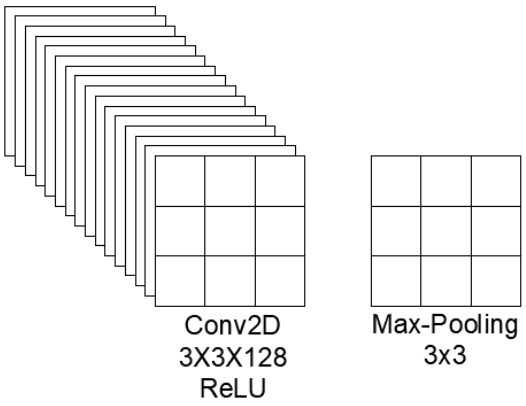



*Ilustración 19.9 Capa de convolución y Max Pooling añadidos a la base convolucional. Creación propia.*

Después de añadir los filtros se aplanaron los datos que se llevaban hasta ahora y se añadieron las siguientes capas densas:

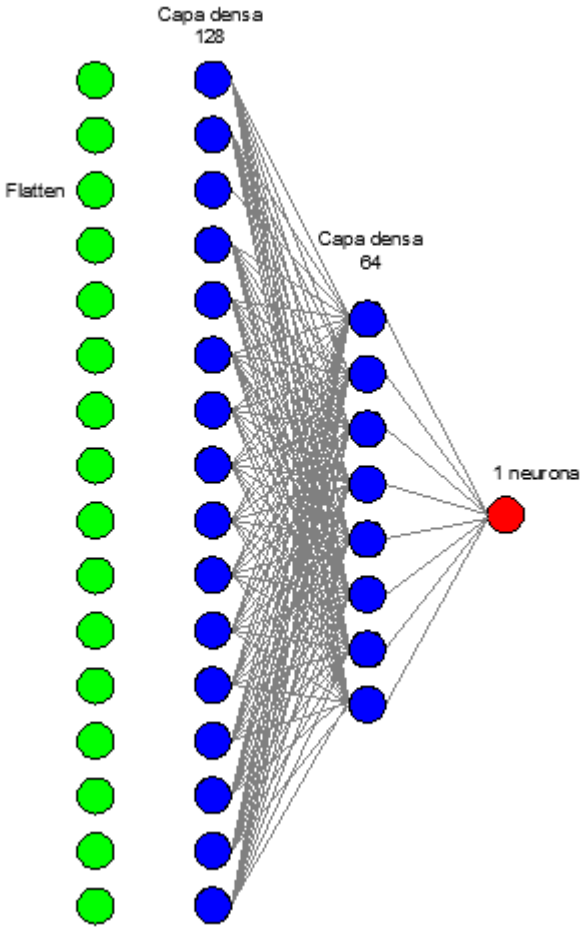



*Ilustración 19.10 Capas densas agregadas después del flatten. Creación propia.*

A las capas densas se les dio la función de activación ReLU y a la última capa la función sigmoide, como en las anteriores.

Enseguida se usó la técnica del aumento de imágenes, pero ahora para la base convolucional, lo cual va aumentar los datos durante el entrenamiento y a ejecutarlo de extremo a extremo en las entradas, con la finalidad de realizar una buena extracción de características. Por lo tanto, las modificaciones en las imágenes fueron las siguientes:

| Imágenes | |
|---|---|
| Rotación | 180° |
| Desplazamiento de anchura | 0.2 |
| Desplazamiento de altura | 0.2 |
| Corte | 0.2 |
| Zoom | 0.2 |
| Voltear horizontal y verticalmente | |

Después, todas las imágenes de entrenamiento y validación fueron reescaladas por 1/255, y fueron redimensionadas al igual que la base convolucional, por 224 pixeles con un tamaño de lote de 20 estableciendo el modo de clasificación como binary porque se usa la función de pérdida binary crossentropy.

Al imprimir, la forma queda como (20, 224, 224, 3). Teóricamente los primeros filtros que se añaden a las redes convolucionales, extraen características generales con ciertos patrones, y las últimas capas extraen características más específicas de las imágenes con las que se hizo la red preentrenada VGG19, por eso, en este caso, no se consideraron los valores preentrenados a partir del bloque 4, porque las imágenes de galaxias son muy diferentes a las de la red preentrenada.

Por tal motivo se hace un reentrenamiento del bloque 4 y 5, el cual después de aplicarle el filtro Max Pooling quedó de 14x14 pixeles con profundidad de 512.

Lo que sigue ahora es compilar el modelo, para esto se usó la función de pérdida mean squared error, el optimizador sgd y la métrica acc.

El modelo compilado ya es entrenable. Se tomaron en cuenta 50 épocas, 100 pasos por época y 50 pasos de validación. Después del procedimiento el modelo ya se puede guardar y graficar.

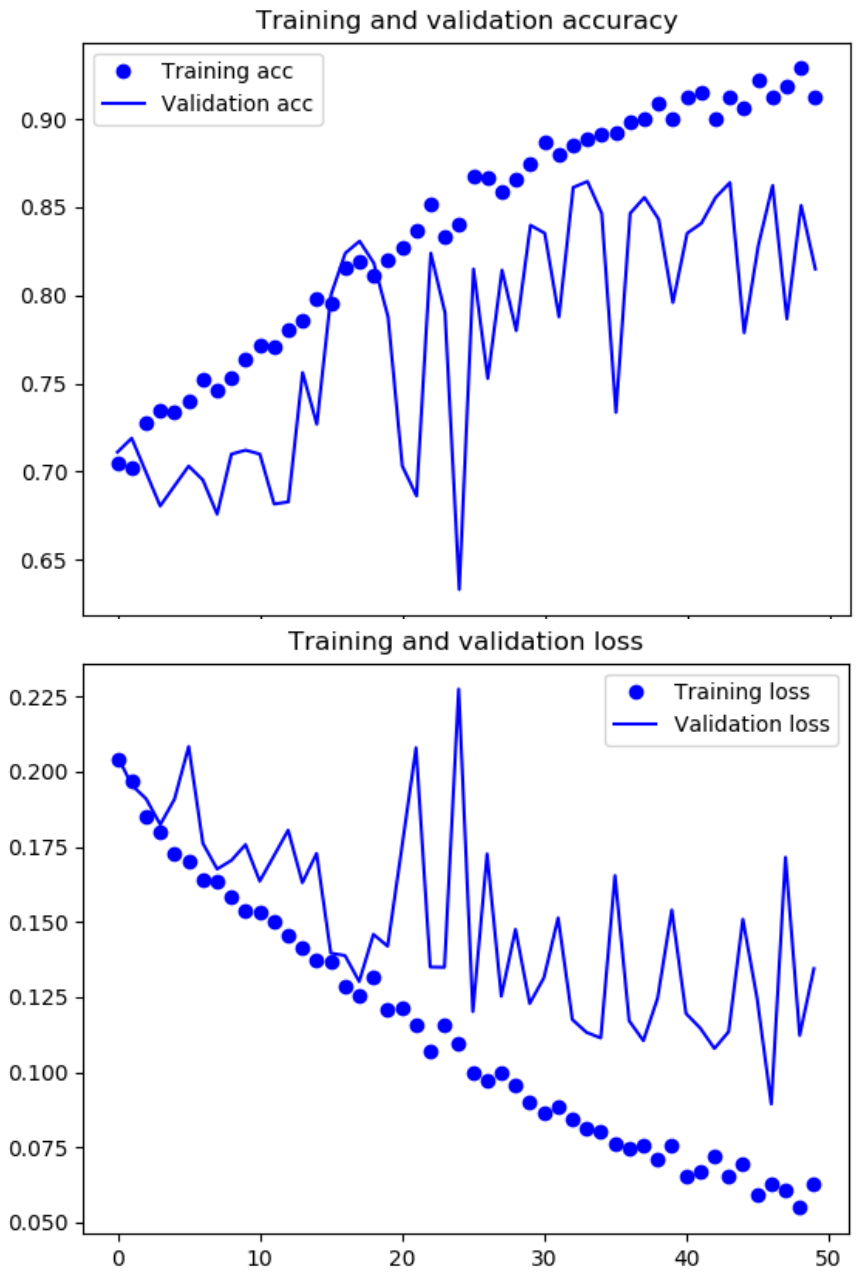



*Ilustración 19.11 Gráficas de precisión (arriba) y pérdida (abajo) respectivamente. Obtenida desde un cuaderno Jupyter como resultado del entrenamiento, validación y prueba.*

Aunque la presencia de ruido es más evidente, la gráfica muestra una mayor precisión respecto a la anterior. Con esta gráfica la red neuronal convolucional está completa. Para verificar su precisión se tomaron los datos de prueba en 224x224 pixeles, ahora con tamaño de lote de 30, y con la clasificación binaria especificada dando una precisión de 0.85.

## CONCLUSIÓN

El mejor resultado obtenido fue con la red preentrenada VGG19, es una de las ventajas de usar redes que han entrenado miles de imágenes de distintos objetos. Para la detección de imágenes con estructura de marea se tiene una precisión del 85%, es sin duda un modelo limitado a proyectos donde no se requiera tanta precisión. Pero a diferencia de otros proyectos es que este requirió menos tiempo para su ejecución, ya que otros pueden llevar años desde la recolección de datos hasta la obtención de resultados.

Uno de los problemas sin duda fue la cantidad de imágenes entrenadas, pero se lograron los resultados esperados con la cantidad de imágenes disponibles en este proyecto. Se puede decir que el modelo sirve para detectar mareas de un conjunto de imágenes con estructuras de shells, colas de marea y mareas fuertes.

Los investigadores H. Domínguez Sánchez (2018) y M. Huertas-Company (2015) consideraron en su clasificación morfológica de galaxias aproximadamente el 75% de sus datos como datos de entrenamiento, con lo cual obtuvieron una precisión de más del 95%. Esto sirvió de base para preparar el número de datos en cada categoría de esta clasificación de mareas. Sus trabajos también demuestran que se necesitan de al menos 10000 galaxias para una adecuada clasificación.

Si de las 12000 galaxias clasificadas visualmente se encontraron mareas en aproximadamente el 4%, es decir, solo se trabajó con 414 galaxias con marea fuerte más las que no tienen marea, podría significar que en DESI encontraría aproximadamente 1 millón de imágenes de las 30 millones de galaxias que conforman el proyecto. Sin duda DESI, proporciona información muy importante de las galaxias a través de espectros, con lo cual se obtienen imágenes de esta calidad para detectar las mareas a través de este tipo de redes.

Este modelo será sin duda de gran ayuda para comenzar proyectos futuros relacionados a la detección de mareas galácticas.

Las redes neuronales convolucionales sí son capaces de detectar estructuras en imágenes cuando se ocupan las técnicas de aprendizaje adecuadas. El Aprendizaje Profundo es indispensable para el avance científico, así como en muchas ramas de la ingeniería. El plan: hacer que la máquina piense algunas cosas de forma similar a un humano es un gran logro.

## BIBLIOGRAFÍA


Chollet F.. (2018). *Deep Learning with Python*. Shelter Island, NY: Manning Publications.

Pérez I. & Ortega F.. (2018). *Clasificación de Obras de Arte por Estilo Artístico Usando Redes Neuronales Convolucionales*. Madrid, España: Universidad Politécnica de Madrid.

Gutierrez E. & Ochoa S.. (2014). *Álgebra lineal y sus aplicaciones*. Ciudad de México: Patria.

Becerra J.. (2005). Matrices y Determinantes. En *Matemáticas VI Área III* (30). Ciudad de México: Colegio de Matemáticas de la ENP-UNAM.

Rodríguez A., Becerrilo H., Falcón N. (2000). *Campos vectoriales y tensoriales*. Ciudad de México: División de Ciencias Básicas e Ingeniería.

Villa J.. (2015). Cálculo Diferencial. Aguascalientes: Dirección general de difusión y vinculación.

Lara F.. (1998). *Fundamentos de Redes Neuronales Artificiales*. Ciudad de México: Centro de Instrumentos UNAM.

Nair P.. (2009). *The Morphology of Local Galaxies and The Basis of The Hubble Sequence*. Toronto: University of Toronto.

Duc P. & Renaud F.. (2013). Tides in Astronomy and Astrophysics. Berlin: Springer-Verlag.


## CIBERGRAFÍA


CUAED UNAM. (2017). Derivada. 2017, de REA UNAM Sitio web: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1146/mod_resource/content/1/contenido/index.html

MathWorks. (2014). ¿Qué es una red neuronal?. 2019, de The MathWorks Sitio web: https://es.mathworks.com/discovery/neural-network.html

Sancho F.. (2009). Redes Neuronales Una visión superficial. 2019, de The CulturePlex Lab Sitio web: http://www.cs.us.es/~fsancho/?e=72

Vinhas A.. (2018). The magic behind the perceptron network. 2019, de Medium Sitio web: https://towardsdatascience.com/the-magic-behind-the-perceptron-network-eaa461088367

UFV Madrid. (2019). Gradiente Descendente. 2019, de Para computar Numerentur Sitio web: http://numerentur.org/gradiente-descendente/

Durán J.. (2019). Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales. 2019, de Medium Sitio web: https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78

Das S.. (2017). CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more. 2019, de Medium Sitio web: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5?

Ortego D.. (2017). ¿Qué es Tensorflow?. 2019, de OpenWebinars Sitio web: https://openwebinars.net/blog/que-es-tensorflow/

La función del Consejo Directivo de Jupyter. (2014). Jupyter. 2019, de Project Jupyter Sitio web: https://jupyter.org/

nvidia Developer. (2019). NVIDIA cuDNN. 2019, de NVIDIA Corporation Sitio web: https://developer.nvidia.com/cudnn

Keras. (2014). Keras: The Python Deep Learning library. 2019, de Keras Documentation Sitio web: https://keras.io/

Burrueco D.. (2014). RMSPROP. 2019, de InteractiveChaos Sitio web: https://www.interactivechaos.com/manual/tutorial-de-machine-learning/rmsprop

Rodríguez V.. (2018). Dropout y Batch Normalization. 2019, de Programing and artificial intelligence Sitio web: https://vincentblog.xyz/posts/dropout-y-batch-normalization

DECaLS and MzLS Principal Investigators. (2019). DESI Legacy Imaging Surveys. 2019, de Legacy Survey Sitio web: http://legacysurvey.org/

Team DESI. (2018). Data systems. 2019, de DESI Sitio web: https://www.desi.lbl.gov/data-systems/

The Astronomical Journal. (2019). Overview of the DESI Legacy Imaging Surveys. 2019, de The American Astronomical Society Sitio web: http://www.astroexplorer.org/details/ajab089df2

The MathWorks. (2014). vgg19. 2019, de MathWorks Sitio web: https://www.mathworks.com/help/deeplearning/ref/vgg19.html;jsessionid=83a0c4d789222804d00c596392ed#bvmdok9.mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe

Google. (2004). Glosario sobre aprendizaje automático. 2019, de Google Developers Sitio web: https://developers.google.com/machine-learning/crash-course/glossary?hl=es-419

Burrueco D.. (2014). BACKPROPAGATION. 2019, de InteractiveChaos Sitio web: https://www.interactivechaos.com/manual/tutorial-de-machine-learning/backpropagation

Domínguez Sánchez, H., Huertas-Company, M., Bernardi, M., Tuccillo, D., Fischer, J. L. (2018). Improving galaxy morphologies for SDSS with Deep Learning. 2019, de Astrophysics data system Sitio web: https://ui.adsabs.harvard.edu/abs/2018MNRAS.476.3661D/abstract

Huertas-Company M.. (2018). Deep Learning and Galaxy Classification. 2019, de American Scientist Sitio web: https://www.americanscientist.org/article/deep-learning-and-galaxy-classification

M. Huertas-Company, R. Gravet, G. Cabrera-Vives, P.G. Pérez-González, J.S. Kartaltepe, G. Barro, M. Bernardi, S. Mei, F. Shankar, P. Dimauro, E.F. Bell, D. Kocevski, D.C. Koo, S.M. Faber, D.H. Mcintosh. (2015). A catalog of visual-like morphologies in the 5 CANDELS fields using deep-learning. 2019, de Cornell University Sitio web: https://arxiv.org/abs/1509.05429

Space Telescope Science Institute. (2009). GOODS: The Great Observatories Origins Deep Survey. 2019, de The Association of Universities for Research in Astronomy Sitio web: http://www.stsci.edu/science/goods/

Domínguez Sánchez, H., Huertas-Company, M., Bernardi, M., Tuccillo, D., Fischer, J. L.. (2018). Improving galaxy morphologies for SDSS with Deep Learning. 2019, de Astrophysics data system Sitio web: https://ui.adsabs.harvard.edu/abs/2018MNRAS.476.3661D/abstract

Domínguez Sánchez, H., Huertas Company, M., Bernardi, M.. (2019). Deep Learning for morphological classification of galaxies. 2019, de Astrophysics data system Sitio web: https://ui.adsabs.harvard.edu/abs/2019hsax.conf..214D/abstract

Rosenberg M., Russo P., Lindberg L., Bladon G. . (2013). Astronomy in Everyday Life. 2019, de International Astronomical Union Sitio web: https://www.iau.org/public/themes/why_is_astronomy_important/

González L.. (2018). Historia de Machine Learning. 2019, de Aprende todo sobre inteligencia artificial Sitio web: http://ligdigonzalez.com/historia-de-machine-learning/

CreativeCommons. (2017). Papel de Sesgo en las Redes Neuronales. 2019, de RSTOPUP Sitio web: https://rstopup.com/papel-de-sesgo-en-las-redes-neuronales.html

Calvo D.. (2015). Función de activación – Redes neuronales. 2018, de Universidad de Valladolid Universidad de Valladolid Sitio web: http://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

Alvarado I.. (2018). Redes Neuronales. 2019, de GitHub Sitio web: https://ml4a.github.io/ml4a/es/neural_networks/