Procesamiento de imágenes multiespectrales para el análisis del estado de la vegetación

Sergio Canalejo Ariza
Programa de Estudios de Informática Multimedia y Telecomunicaciones
Inteligencia Artificial

Samir Kanaan Izquierdo
Carles Ventura Royo

Junio de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Procesamiento de imágenes multiespectrales para el análisis del estado de la vegetación*
Nombre del autor:	*Sergio Canalejo Ariza*
Nombre del consultor/a:	*Samir Kanaan Izquierdo*
Nombre del PRA:	*Carles Ventura*
Fecha de entrega (mm/aaaa):	06/2018
Titulación::	*Programa de Estudios de Informática Multimedia y Telecomunicaciones*
Área del Trabajo Final:	*Inteligencia Artificial*
Idioma del trabajo:	***Castellano***
Palabras clave	*Imágenes Multiespectrales, Indices de Vegetación, Red neuronal convolucional*

Resumen del Trabajo (máximo 250 palabras):

El presente trabajo trata de desarrollar un sistema de análisis de imágenes multiespectrales con los valores de las diferentes bandas de espectrales, de manera que sea capaz de procesar dichas imágenes y obtener de forma rápida el valor de un índice de vegetación concreto que nos de una información sobre el estado de la vegetación en el momento de las imágenes.

Para ello se ha diseñado una red neuronal convolucional, actuando como regresor, que sea capaz de procesar los datos y ofrecer en un corto periodo de tiempo el valor del índice de vegetación elegido. Se ha elegido este tipo de sistemas por su gran capacidad y acierto en los tratamientos de imágenes.

Este sistema pretende agilizar la obtención de estos datos por parte del Departament de prevenció d'incendis ya que actualmente depende un sistema externo que da los datos en rangos dos semanas.

Tras el diseño y entrenamiento de la red neuronal convolucional, se ha conseguido obtener unos resultados muy próximos a los resultados reales para un píxel concreto de la imagen de forma inmediata, suponiendo una gran mejora con la manera de obtener esos datos en la actualidad.

Abstract (in English, 250 words or less):

The present work tries to develop a multispectral image analysis system with the values of the different spectral bands, so that it is capable of processing said images and quickly obtain the value of a specific vegetation index that gives us an information about the state of the vegetation at the time of the images.

To do this, a convolutional neuronal network has been designed, acting as a regressor, capable of processing the data and offering the value of the selected vegetation index in a short period of time. This type of systems has been chosen for its great capacity and accuracy in the image treatments.

This system aims to expedite the obtaining of these data by the Department of prevention of incendis since it currently depends on an external system that gives the data in ranges of two weeks.

After the design and training of the convolutional neuronal network, it has been possible to obtain results very close to the actual results for a specific pixel of the image immediately, assuming a great improvement with the way of obtaining that data at present.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

El empleo de sistemas imágenes multiespectrales tomadas por satélite es un método muy extendido que se utiliza para la monitorear el estado de las parcelas vegetales una zona. Actualmente existen una gran variedad de satélites que ofrecen entre sus servicios, además de las obtención de los valores de distintas bandas infrarrojas, productos más avanzados con los que obtener una información de análisis útil.

En este momento, el Departament de prevenció d'incendis está usando imágenes proporcionadas por el satélite MODIS[1] para obtener diferentes índices de vegetación que ayuden al control y prevención de incendios, así como poder valorar el efecto del cambio climático en la vegetación. Para este último punto, se está llevando a cabo una monitorización de la población de la especie de pinus nigra de la comarca de Solsonés, Lérida. El pinus nigra se trata de una especie de especial interés, ya que corresponde a una especie de transición entre los climas mediterráneo y los clima de montaña siendo por eso muy afectada por el cambio climático.

El Departament obtiene los datos que les proporciona MODIS en un plazo de 15 días, que es lo que tarda el servicio MODIS en elaborar los datos necesarios. El sistema a desarrollar pretende hacer que este plazo desaparezca, pudiendo obtener los datos al instante. Otra ventaja derivada del sistema a desarrollar es que se podrían obtener los valores de estos índices en periodos anteriores donde no se disponía de esa información, de manera que se puedan estudiar como afecto a la vegetación hechos pasados importantes, como por ejemplo los grandes incendios del 94 en Cataluña.

Para el desarrollo del sistema se ha optado por usar técnicas de **aprendizaje profundo,** o aprendizaje automático. Dentro de las arquitecturas de aprendizaje profundo están muy de actualidad las llamadas **redes neuronales[2],** y en concreto las redes neuronales convolucionales. Este tipo de redes fueron refinadas en el año 2012 por Dan Ciresen y desde entonces están consiguiendo unos resultados impresionantes y gran popularidad en el mundo de la inteligencia artificial. Debido a la naturaleza de las redes neuronales convolucionales, las hacen especialmente útiles en el análisis de imágenes. La gran popularidad adquirida, junto con el gran avance en los computadores y el poder hacer uso de las GPUs para este tipo de computación, han hecho que las redes neuronales estén en pleno apogeo y que multitud de grandes empresas como Google, Apple o

Tesla, estén trabajando intensamente en ellas para sus productos más innovadores.
En el trabajo se abordarán tres grandes objetivos, siendo el principal de todos desarrollo de un sistema de auto-aprendizaje, basado en la arquitectura de una red neuronal convolucional, que sea capaz de analizar una imagen multiespectral y dar como resultado el valor de un índice vegetal que de una información cuantitativa del estado de la cubierta vegetal de forma inmediata, ofreciendo una fiabilidad suficientemente alta como para poder prescindir del sistema MODIS actual que ofrece dichos resultados en plazos de 15 días.

1.2 Estado del arte

1.2.1 Redes neuronales

Las redes neuronales son un modelo computacional basado en un conjunto de unidades llamadas neuronas o unidad neuronal. Cada unidad neuronal está conectada a muchas otras y los enlaces que pueden incrementar o inhibir el estado de la activación de las neuronas adyacentes. Cada neurona opera de forma individual empleando funciones suma. Son sistemas con capacidad para el aprendizaje automático, lo que es conocido como **Deep Learning.**

Estas redes suelen consistir en varias capas con un diseño de cubo, en el que la señal las atraviesa de adelante hacía atrás. El método de **Propagación hacía atrás** es usado para restablecer los pesos de las unidades neuronales, es decir, es como la red va aprendiendo mientras va funcionando.

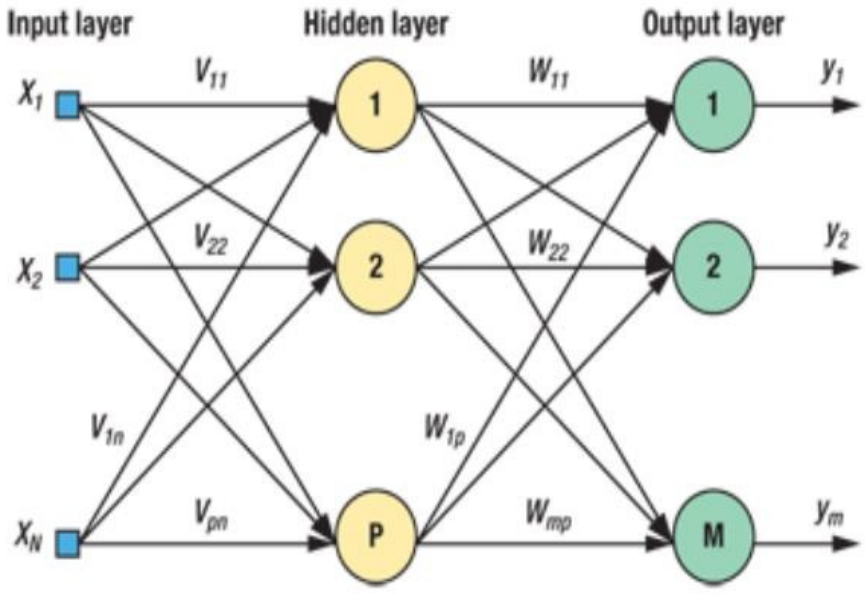

Ilustración 1: Ejemplo de red neuronal

El objetivo de una red neuronal es resolver problemas emulando el funcionamiento del cerebro humano y son usadas para una gran variedad de tareas, tales como la visión por computador, reconocimiento de voz, redes sociales, videojuegos o diagnóstico médico.

Dentro del tipo de redes neuronales, en el ámbito del procesamiento de imágenes o audio, destacan las **redes neuronales convolucionales.** Este tipo de redes están especialmente preparadas para procesar como datos de entrada matrices muy grandes, ya que las redes neuronales normales no escalan bien con este tipo de imágenes. Suponiendo que contamos de entrada con una imagen de 200x200 con colores RGB, usando una red neuronal convencional, si hablamos de una red plenamente conectada, se estaría trabajando con neuronas de 120.000 pesos (200x200x3), y eso hablando de una imagen relativamente pequeña.
En cambio, las redes neuronales convolucionales colocan una capa convolucional al principio de la red. Esta primera capa que da nombre a la red, realiza la tarea llamada **convolución**, que consiste en aplicar algún tipo de *filtro* o *kernel* sobre la imagen original, e tal manera que se extraigan las características más importantes de la imagen reduciendo en gran medida el tamaño de los parámetros.
La operación de convolución también muy útil cuando se trabaja con entradas de tamaño variable.

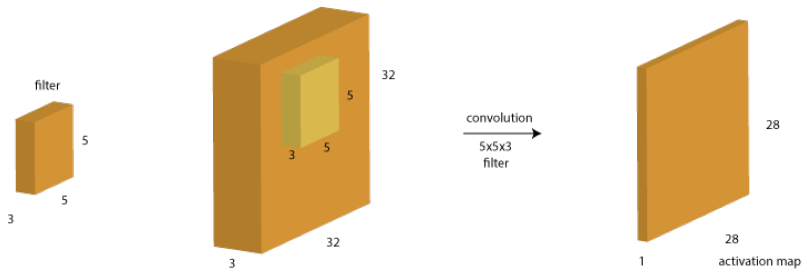

Ilustración 2: Capa de convolucion

Tras la capa convolucional, se suele colocar la capa de **pooling**. La utilidad principal de esta capa es reducir las dimensiones espaciales que se recibe como entrada. Para realizar esto, utiliza la operación llamada *reducción de muestreo*. Como se ha visto durante la asignatura de Inteligencia Artificial Avanzada, reducir los datos conlleva, por pequeña que sea, a una pérdida de la información. Aunque en este tipo de redes esta pérdida de información puede ser beneficiosa debido a la disminución del cálculo en las siguientes capas y la reducción del sobreajuste.
Tras estas dos capas ya se conecta la capa clasificadora totalmente conectada que realiza las operaciones necesarias para procesar los datos.

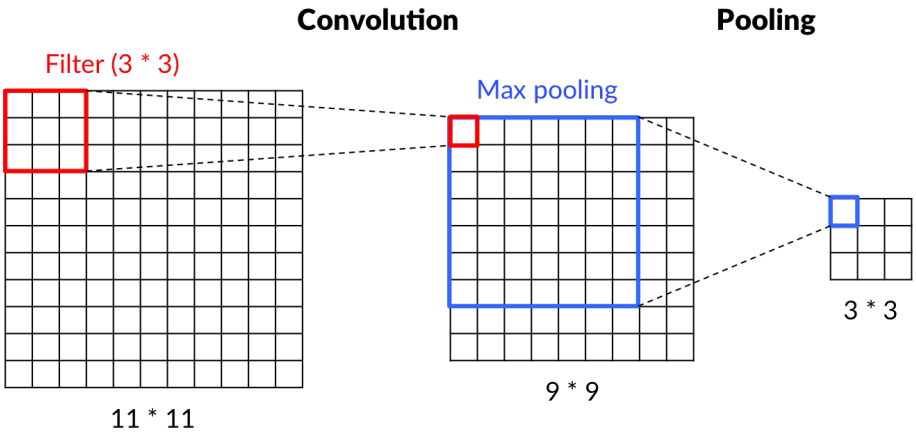

Ilustración 3: Capa de pooling trás convolución

1.2.2 Imágenes multiespectrales

Una imagen multiespectral es aquella en la que se capturan los datos de la imagen dentro de rangos de longitud de onda específicos a través del espectro electrónico magnético.
Una imagen multiespectral divide la luz en pequeño número de **bandas espectrales**, generalmente de 3 a 15 bandas por imagen. Existe un caso especial, llamado **imagen hiperespectral**, en el que se capturan cientos de bandas espectrales de forma contigua.

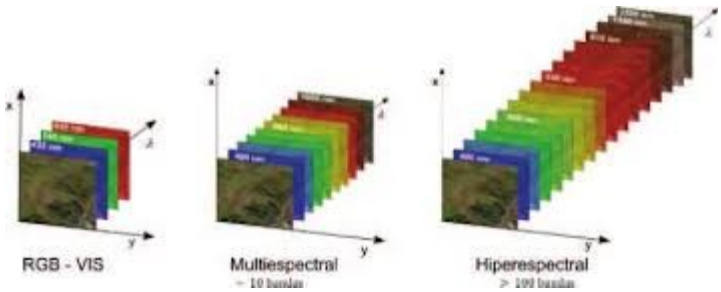

Ilustración 4: Diferencia entre imágen RGB, Multiespectral e Hiperespectral

Este tipo de imágenes son la base para multitud de sistemas de observación remota de la tierra, existiendo una variedad de satélites que proporcionan estas fotos en diferentes bandas, para que los usuarios puedan descargarse las que deseen según la observación que vayan a llevar a cabo. Se pueden destacar los siguientes:

- **Landsat**: Una serie de satélites construidos y puestos en órbita por EE.UU para la observación en alta resolución de la tierra. El último en ser lanzado es el Landsat-8, que da acceso a 9 bandas espectrales.

- **Sentinel**: Sentinel es un proyecto multi-satélite llevado a cabo por la **ESA** (Agencia Espacial Europea) dentro del marco *Programa Copérnico*. De este conjunto de satélites, el Sentinel-2 proporciona imágenes ópticas terrestres de gran resolución. Estas imágenes son imágenes multiespectrales con datos de 13 bandas.
- MODIS Land Team: Moderate-Resolution Imaging Spectroradiameter (MODIS), es un instrumento lanzado en órbita por la NASA. Este satélite ofrece datos como la temperatura del suelo y del océano, color del océano, cartografía de la vegetación etc. Los sensores de MODIS son capaces de ofrecer imágenes multiespectrales en 36 franjas, además de ser capaces de detectar nubes del tipo cirrus que están relacionadas con el calentamiento global. Las bandas de MODIS son sensibles a los incendios, con lo que se pueden a llegar a distinguir llamas de brasas.

De los tipos de servicios de satélites descritos, el Departament está usando el servicio MODIS, por lo que será este tipo de imágenes las que usaremos para el desarrollo del sistema.

1.2.3 Índices de vegetación

Además de las imágenes multiespectrales, en las tareas de observación remota se han desarrollado una serie de índices de vegetación o índices verdes. Estos índices son transformaciones que implican efectuar una combinación matemática entre las diferentes niveles que se aprecian entre una o más bandas espectrales de una imagen multiespectral.

El desarrollo de estos índices obedeció a la observación de la consistencia de la respuesta a la reflectancia de la luz roja e infrarroja de la vegetación verde. A mayor clorofila, mayor observación de la luz incidente roja, a mayor volumen foliar, mayor reflectancia de la luz infrarroja cercana.

Los índices se aplican en análisis cualitativos o cuantitativos. Empleados cuantitativamente, permiten determinar rápidamente el estado relativo de la vegetación de una zona. Se pueden nombrar los siguientes índices:

- **Normalized Difference Vegetation Index[3]**: NDVI es el índice de vegetación más conocido. Es usado para estimar la cantidad, calidad y desarrollo de la vegetación en base a la intensidad de la radiación de ciertas bandas que la vegetación refleja. Los valores de este índice fluctúan entre el -1 y el 1.
- **Soil Adjusted Vegetation Index[4]**: SAVI el índice que se utiliza para minimizar el efecto del suelo sobre otros índices de

vegetación, como en el NDVI. Se suele usar en los primeros estadios de un cultivo, donde la cubierta vegetal es parcial, y por lo tanto reflectancia provocada por el suelo puede producir errores.

- **Green Normalized Difference[5]**: GNDVI es el índice que reemplaza en la formula del NDVI la banda roja por la banda verde, permitiendo así tener un indicador del nivel de sanidad de la vegetación.
- **Moisture Stress Index[6]:** MSI es el índice que relaciona la información referida a la producción de biomasa de la vegetación, captada por el espectro rojo e infrarrojo cercano, y la referida a la humedad en planta, detectable en el infrarrojo medio. Este indicador se basa en que a medida que la vegetación se seca, produce como resultado aumentos en la zona del rojo y en el infrarrojo. Los valores del índice también fluctúan entre el -1 y el 1, siendo los valores 0 o negativos referentes al suelo. Los valores mayores de 0,7 son considerados extremos.

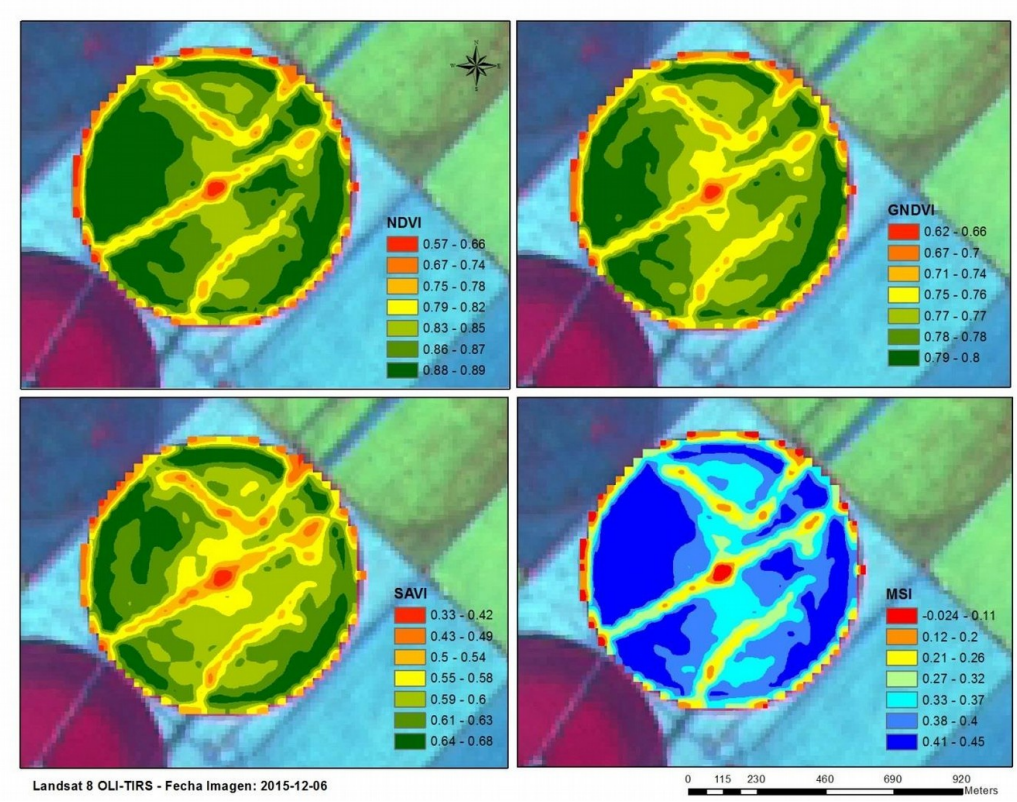

Ilustración 5: Representación gráfica de los diferentes índices de vegetación

Además de esos los índices descritos anteriormente, está el **Fraction of Absorved Photosynthetically Active Radiation[7]** o fAPAR, el cuál es comúnmente aceptado como un parámetro biofísico fundamental en el contexto de las disciplinas relacionadas con el cambio climático global. Nuestro sistema se va a centrar en el estado

de la especie pinus nigra, descrita en la introducción, para el monitoreo del cambio climático. Es por esta razón por la que nuestro sistema dará como resultado a una imagen multiespectral dada el valor de este índice en un píxel dado.

1.3 Objetivos del Trabajo

1.3.1 Objetivos generales

Los objetivos generales que se desean alcanzar con este Trabajo de Fin de Máster son los siguientes:

- **OG01 –** Investigar y establecer bases del estado del arte de las redes neuronales convolucionales.
- **OG02 –** Diseñar y programar un modelo para estimar el índice fAPAR de una imagen multiespectral.
- **OG03 –** Evaluar los modelos desarrollados y analizar y comparar los valores obtenidos.

1.3.2 Objetivos específicos

Los objetivos generales descritos posteriormente se pueden descomponer en una serie de objetivos más específicos. La siguiente tabla muestra la relación entre los objetivos generales y los objetivos específicos:

OG01	**OE01** - Investigar sobre redes neuronales.
	OE02 – Investigar sobre las imágenes multiespectrales y el uso que se puede dar a los datos obtenidos de las mismas.
	OE03 - Buscar y leer trabajos previos relacionados.
OG02	**OE04** – Construcción de la red neuronal.
	OE05 – Selección de los conjuntos de datos a usar.
	OE06 – Entrenamiento de la red neuronal con el conjunto de datos seleccionado.
OG03	**OE07** – Analizar y diseñar las pruebas a realizar.
	OE08 – Analizar los resultados obtenidos.

Tabla 1: Objetivos específicos

1.4 Enfoque y método seguido

Para conseguir los objetivos mencionados, se va a desarrollar un producto nuevo, puesto que no en la actualidad no existe ningún otro sistema parecido que de la funcionalidad que se está buscando.

La naturaleza del sistema que se busca, el cuál tiene como entrada de datos una serie de imágenes multiespectrales, lo hace idóneo para las arquitecturas basadas en redes neuronales convolucionales. Lenguajes de programación como Python o R ofrecen unas librerías y frameworks muy útiles para poder crearlas desde 0.

Las redes neuronales realizadas en estos lenguajes de programación son un producto portable y sin costos, que ofrecen unos resultados con gran acierto en tiempos muy bajos gracias al uso tanto de los nuevos tipos de procesadores de los equipos modernos, como de la GPU para sus cálculos, siendo posible utilizarlas en equipos de uso cotidiano sin ningún problema.

1.5 Planificación del Trabajo

1.5.1 Tareas

La duración de las tareas se indicá en la última columna y viene expresado en días.
Con los objetivos específicos ya definidos, las tareas que se obtienen son estas.

OG01	**OE01**	**T01** – Análisis de redes neuronales.	4
	OE02	**T02** – Análisis de los diferentes índices vegetales. **T03** – Redacción de situación de partida. **T04 –** Análisis de imágenes multiespectrales	3 6 3
	OE03	**T05** – Análisis de trabajos previos.	4
OG02	**OE04**	**T06** – Selección de las técnicas a usar para la construcción de la red neuronales. **T07** – Selección lenguaje/librería a usar **T08** – Construcción de la red neuronal.	2 5 10
	OE05	**T09** – Selección del conjunto de datos.	2
	OE06	**T10** – Entrenamiento y refinamiento de la red neuronal sobre el conjunto de datos obtenidos.	10
OG03	**OE07**	**T11** – Definición de pruebas. **T12** – Ejecución de las pruebas.	10 16
	OE08	**T13**– Análisis de los resultados.	16

Tabla 2: Objetivos y tareas

A estas tareas se incluyen una serie de tareas necesarias para la correcta realización del Trabajo de Fin de Máster, aunque no forman parte de ningún objetivo en concreto.

T14 – Lectura de la documentación del Trabajo Fin de Máster.	2
T15 – Elección de temática y objetivos.	4
T16 – Enfoque y metodología.	4
T17 – Redacción plan de trabajo.	10
T18 – Redacción de la memoria.	11
T19 – Elaboración de la presentación.	7

Tabla 3: Tareas sin objetivos asociados

1.5.2 Calendario

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Pr
✓		**Trabajo Fin de Máster**	**73 días**	**mié 21/02/18**	**vie 01/06/18**	
✓		**Planificación**	**10 días**	**mar 06/03/18**	**lun 19/03/18**	
✓		Lectura Documentación	2 días	mié 21/02/18	jue 22/02/18	
✓		Temática y Objetivos	4 días	jue 22/02/18	mar 27/02/18	
✓		Metodología	4 días	mar 27/02/18	vie 02/03/18	
✓		Redacción plan de trabajo	10 días	mar 06/03/18	lun 19/03/18	
✓		**Investigación previa**	**15 días**	**mar 20/03/18**	**lun 09/04/18**	
✓		Análisis redes neuronales	4 días	mar 20/03/18	vie 23/03/18	
✓		Análisis trabajos previos	4 días	mar 20/03/18	vie 23/03/18	
✓		Análisis indices vegetales	3 días	mar 20/03/18	jue 22/03/18	
✓		Análisis imágenes multiespectrales	3 días	jue 22/03/18	lun 26/03/18	
✓		Redacción situación de partida	6 días	lun 02/04/18	lun 09/04/18	
✓		**Diseño del Regresor**	**15 días**	**lun 09/04/18**	**vie 27/04/18**	
✓		Selección de las técnicas a usar	2 días	lun 09/04/18	mar 10/04/18	
✓		Selección lenguaje/librería a usar	5 días	mar 10/04/18	lun 16/04/18	
✓		Selección del conjunto de datos	2 días	mar 10/04/18	mié 11/04/18	
✓		Construcción de la red neuronal	10 días	lun 16/04/18	vie 27/04/18	
✓		Entrenamiento	10 días	lun 16/04/18	vie 27/04/18	
✓		**Ejecución de pruebas**	**24 días**	**mar 17/04/18**	**vie 18/05/18**	
✓		Diseño de pruebas	10 días	vie 27/04/18	jue 10/05/18	
✓		Ejecución de pruebas	16 días	vie 27/04/18	vie 18/05/18	
✓		Análisis de los resultados	16 días	vie 27/04/18	vie 18/05/18	
✓		**Preparación entrega final**	**11 días**	**vie 18/05/18**	**vie 01/06/18**	
✓		Redacción memoria	11 días	vie 18/05/18	vie 01/06/18	
✓		Elaboración presentación	7 días	vie 18/05/18	lun 28/05/18	

Ilustración 6: Planificación temporal

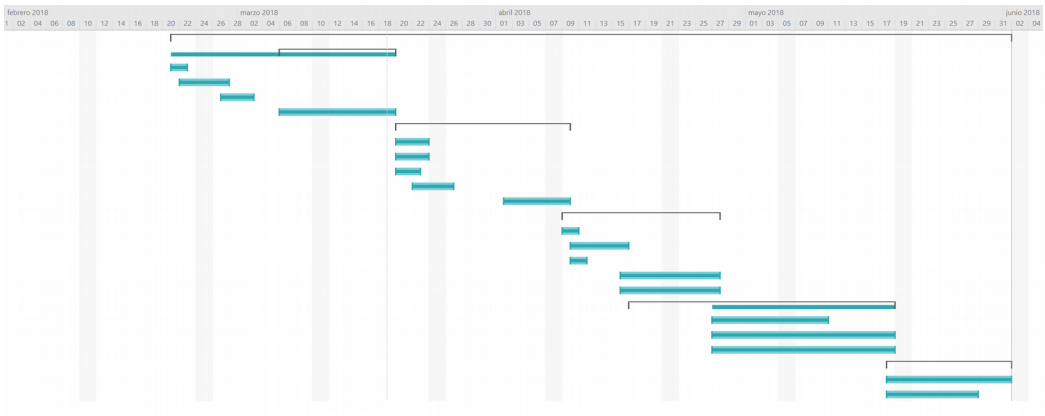

Ilustración 7: Diagrama de Gantt del proyecto

1.6 Hitos

En este caso, los hitos se relacionan con entregables, de manera que coincidan con las entregas de las PEC establecidas en el calendario del Trabajo Fin de Máster.

Los hitos a destacar son:

- Plan de trabajo (documento)
- Redacción de la situación de partida (documento)
- Red neuronal (código)
- Definición de pruebas (documento)
- Memoria (documento versión final)
- Presentación (vídeo)

1.7 Breve descripción de los otros capítulos de la memoria

A continuación se pasará a describir las tareas más importantes que se han tenido que llevar a cabo para cumplir los objetivos propuestos para el proyecto.

Primero se describirá la tecnología usada, describiendo tanto el equipo físico donde se han llevado a cabo las pruebas, como el software y librerías utilizadas.

Seguidamente se describirá la arquitectura de la red neuronal diseñada, explicando las diferentes capas que las componen y sus parámetros. También se explicará el tipo de pruebas diseñadas y se mostrarán y comentarán los resultados obtenidos.

Finalmente se expondrán las conclusiones que se obtienen tras el análisis de los resultados obtenidos de los experimentos desarrollados.

2. Diseño de la red neuronales

2.1 Descripción del equipo de pruebas

Un aspecto importante en el producto a desarrollar es que no se requiera de unos equipos de procesamiento muy potentes pero que aun así se obtuvieran resultados en un tiempo rápido. El producto se desarrolló teniendo esto en mente y se ha realizado y probado en equipos de uso cotidiano para poder demostrar la viabilidad del mismo.

El equipo de desarrollo del producto ha sido el siguiente:

Procesador	Intel I7 4510U 2.0 GHz.
Memoria RAM	8 GB Ram DDR3.
Disco Duro	500 GB HD.
Tarjeta Gráfica	Intel HD Graphics 4440, 1792 MB memoria dedicada.
Sistema Operativo	Ubuntu 18.04 64 bits.

Tabla 4: Descripción del equipo de desarrollo

En el equipo descrito arriba es donde se desarrolló y se realizaron los experimentos que describirán en capítulos posteriores. Así mismo, tras la realización de estos experimentos, el producto obtenido se probó también en otro equipo con características distintas para poder corroborar que funcionaba de forma igualmente correcta.

El equipo donde se realizaron las pruebas posteriores tiene las siguientes características:

Procesador	Intel I5 3470 3.2 GHz.
Memoria RAM	16 GB Ram DDR3 .
Disco Duro	1 TB HD.
Tarjeta Gráfica	AMD Radeon R9, 3 GB GDDR5 de memoria dedicada.
Sistema Operativo	Windows 10 Profesional 64 bits

Tabla 5: Descripción del equipo de pruebas

2.2 Descripción de las tecnologías usadas

Existen numerosas alternativas de lenguajes de programación en las que se pueden desarrollar redes neuronales. Debido a la naturaleza de las mismas, que tal y como se describieron en los capítulos anteriores, no son otra cosa que un conjunto de nodos que aplican unas operaciones matemáticas a unos valores de entrada según una serie de parámetros. Es decir, las redes neuronales tienen una fuerte base matemática en su núcleo. Con esto en mente, era fácil saltar directamente a lenguajes de programación con especializados en este tipo de tareas.

Así pues, de estos lenguajes de programación fuertemente orientados a operaciones matemáticas, podemos destacar algunos como MatLab, R o Python. De todos estos, Python[8] se ha escogido como lenguaje usado por varios motivos de peso.

1. Python es un lenguaje multiplataforma con licencia GNU que es ampliamente usado, existiendo una increíble base de conocimiento al respecto.
2. Durante la asignatura de Inteligencia Artificial se ha usado este lenguaje para el desarrollo de otras aplicaciones, por lo que existe una base que permite empezar a desarrollar la aplicación de forma rápida.
3. Tal y como se vio durante la asignatura, Python cuenta con excelentes librerías matemáticas que facilitan mucho la realización de complicadas operaciones matemáticas. Además también cuenta con opciones para trabajar con elementos de tipo array y/o matrices de forma intuitiva, siendo estos tipos de elementos un punto clave en las operaciones que se realizan en las redes neuronales convolucionales.

Una vez elegido el lenguaje de programación que se va a usar, se ha buscado información sobre distintas librerías que serán necesarias para el desarrollo del producto.

Para empezar a construir el entorno de desarrollo idóneo, se ha instalado Anaconda. Anaconda[9] es un sistema de gestión de paquetes libre y abierta ampliamente usada en proyecto científicos para Python y R. Anaconda ofrece grandes facilidades a la hora de instalar todo lo necesario para empezar proyectos de carácter científico, ofreciendo desde librerías, hasta entornos de desarrollo a un sólo click.

Para facilitar el tratamiento de estos datos, se ha hecho uso de la librería scikit-learn[10]. Potente librería que ya se usó durante la asignatura de Inteligencia Artificial y que ofrece grandes prestaciones en operaciones matemáticas y de tratamiento de matrices.

En primer lugar se descubrió la existencia de una librería desarrollada por Google, llamada TensorFlow[11], con un gran uso para el desarrollo

de aplicaciones de aprendizaje automático, como es el caso de la construcción de a una red neuronal. El uso de esta librería se descartó ya que es una biblioteca de bajo nivel, por lo que el trabajo de desarrollar una red neuronal convolucional de las características deseadas conllevaría un gran trabajo de programación. Para la red neuronal que se desea construir, es deseable una biblioteca de alto nivel, de manera que con pocas instrucciones se pueda declarar y poner en funcionamiento la red neuronal al completo.

Para esto, se escoge la librería Keras[12] para el desarrollo de la red neuronal. Como se ha dicho, Keras es una librería de alto nivel que ofrece una API para python para la construcción de redes neuronales.

Keras permite hacer un fácil prototipado de la red neuronal que se quiera construir. Da soporte tanto a redes neuronales recurrentes, como a redes neuronales convolucionales. Para las operaciones matemáticas a bajo nivel utiliza otras librerías como CNTK, Theano o, como para nuestro caso, TensorFlow, aprovechando toda la potencia de estas librerías. Keras además permite usar para sus operaciones tanto la CPU como la GPU, de manera que la red neuronal será mucho más rápida.

Además, dado el índole de los datos de entrada con la que vamos a trabajar, en formato HDF4, será necesaria una librería capaz de trabajar con dichos datos. Aunque Keras ofrece utilidades para trabajar con ficheros en formato HDF5, no así con ficheros en HDF4. Para poder trabajar con ellos, se ha instalado la librería PyHDF[13] que ofrece las utilidades necesarias para operar con los ficheros de entrada.

Lenguaje de programación	Python 3.6.4
Gestor de paquetes	Anaconda
IDE	Visual Studio
Librerias usadas	scikit-learn
	numpy
	Keras
	TensorFlow
	PyHDF

Tabla 6: Tabla resumen de software utilizado

2.3 Naturaleza de los datos

Los datos de entrada de nuestro sistema son ficheros en formato .HDF. Este tipo de formato se corresponde a datos de carácter científico. Estos ficheros estructuran los datos como en paquetes, de manera que se pueda acceder a los que queramos. Las datos se obtienen del producto MOD09GA de MODIS[14]. De este producto se puede obtener una imagen jpeg de la zona que ha fotografiado el satélite, una fichero en formato .xml con la descripción de los datos obtenidos y finalmente un fichero .HDF con los datos que se obtienen del radar. Los datos referente a la zona de estudio son los que están etiquetados con como h18v04.

De todos los datos que contiene este fichero, los que nos interesan son los tipos de objetos denominados sur_refl_b0x_1, siendo x un número del 1 al 7 correspondiente al número de banda a la que hace referencia. Para definir el valor de la reflectancia, el producto MODIS usa 16 bit integer signed, que van desde el valor -100 hasta el 16000. De esta manera, tenemos el valor de las 7 bandas de reflectancia que capta el satélite. Estos datos son matrices de 2400 filas por 2400 columnas. La zona de estudio, correspondiente a parcelas con pinus nigra, es la que está en la posición x= 450 y = 500.

Se ha procedido a la descarga masiva de estos ficheros correspondientes a periodos 2000 a 2002.

Como datos de salida para entrenar a nuestra red también tenemos un fichero con formato .HDF. Esta vez son ficheros correspondientes al producto MOD15A2H[15] que proporciona datos de índices avanzados. Como se ha indicado en la instrucción, MODIS ofrece estos datos con cierto retraso, no cada día. Dentro de los diferentes datos que tiene el fichero, nos interesan los objetos denominados Fpar_500m, que es una matriz de tamaño 2400 por 2400 con los valores del índice fPAR de la zona fotografiada por el satélite. De nuevo, nos interesan los ficheros etiquetados como h18v04. MODIS en este caso ofrece los datos usando 8 bit unsigned, ofreciendo un rango de valores de 0 a 255.

Se ha procedido a la descarga masiva de estos ficheros correspondientes a periodos de 2000 a 2002.

La obtención de estos datos ha supuesto un gran trabajo en el transcurso del proyecto. Ha sido totalmente necesaria la colaboración con el Departament de prevenció d'incendis de la Generalitat para la identificación tanto de los productos MODIS a utilizar como de la zona a estudio y la comunicación con ellos ha sido lenta, por lo que han existido periodos de espera para poder continuar con el trabajo, ya que a la hora de la construcción de la red neuronal, uno de los puntos más importantes es conocer la naturaleza tanto de los datos de entrada como de los datos de salida.

Una vez que se ha tenido la certeza de los datos necesarios para el entrenamiento de la red neuronal, se ha descubierto que el sitio web de los productos MODIS no dispone de ninguna utilidad de descarga masiva de datos, por lo que ha sido necesaria la descarga manual de cientos de ficheros bastante pesados para empezar a poder trabajar con ellos. Al terminar estos pasos, conociendo ya la naturaleza de los datos, se pudo comenzar el trabajo del diseño de la red neuronal y su entrenamiento.

2.3 Arquitectura de la red neuronal desarrollada

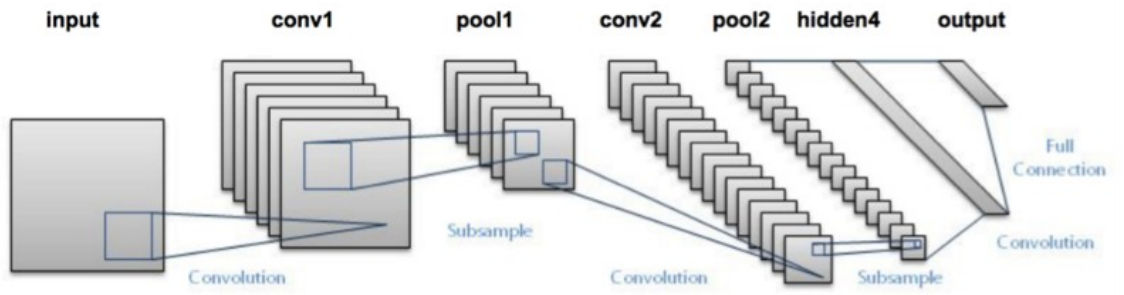

Ilustración 8: Vista genera de la red neuronal a diseñar

La red neuronal convolucional, a partir de ahora **CNN**, diseñada corresponde a un regresor, ya que disponiendo de unos valores de entrada, en este caso los valores de las diferentes bandas de una imagen multiespectral, se debe obtener el valor del índice **fAPAR** (*Fraction of Absorbed Photosynthetically Active Radiation*). Como se ha explicado anteriormente, este índice ofrece un valor cuantitativo muy interesante, ya que está directamente relacionado con la productividad primaria de la fotosíntesis y se puede utilizar para estimar la asimilación del dióxido de carbono por la vegetación, por lo que se usa como parámetro para el estudio del cambio climático. Estos valores son obtenidos con el formato descrito en el apartado anterior. La red diseñada permitiría obtener las medidas actualizadas cada día sin tener que esperar por que el servicio MODIS los elabore o reconstruir el estado de la vegetación en fechas anteriores en los que no existía el producto MODIS que ofrece estos datos.

Aunque los datos finalmente son en formato tabular, en realidad estamos tratando con imágenes, donde cada celda corresponde al valor de una banda para un píxel concreto. De manera que es evidente que existirá algún tipo de relación espacial entre ellos que será interesante de tratar con redes convolucionales.

Para formar los datos de entrada, se ha pasado a recorrer los ficheros descargados y formar una matriz de 3 dimensiones, donde en el índice Z se corresponde a cada una de las 7 bandas de reflectancia obtenida. Como datos de salida tenemos un array simple, con el valor del índice fAPAR para el píxel de estudio. Además no se va coger toda la matriz de datos de cada banda, sino una submatriz que corresponda al píxel del

que se desea averiguar su índice fAPAR y unos vecinos contiguos a este píxel.

Una de las ventajas que ofrece Keras es que se puede desarrollar una red neuronal de forma modular, definiendo paso a paso cada una de las capaz que se van a utilizar y el internamente se encarga de realizar las conexiones pertinentes entre las diferentes capas. Para ello, Keras ofrece el modelo sequential en el que las capas, o layers, que se vayan definiendo las conecta una a una, de forma que la creación de la red sea fácil e intuitiva.

La primera capa que definimos es una capa convolucional en 2D de 32 filtros. Esta capa leerá directamente el input de datos y realizará la operación de convolución a ellos. Al ser la primera capa, hay que indicarle el formato de las variables de inputs esperadas. En una capa convolucional 2D, los formatos que se esperan es una matriz de 3 dimensiones, donde se indica el alto y el ancho de la matriz de datos, así como el número de canales. En fotos normalmente este número es 3, correspondiente a los canales R G B, pero en nuestra red el número de canales de los datos de entrada es 7, ya que cada canal corresponde a una banda de reflectancia que se ha leido en los ficheros de datos.

Ya al declarar está primera capa se ve puede apreciar la potencia de Keras a la hora de definir redes neuronales, ya que en comparación con la definición de una capa de convolución en TensorFlow nos hace falta definir muchos más parámetros que con Keras no son necesarios. Keras internamente configura todos lo necesario del tensor de forma automática por nosotros.

Seguida de está capa, se realiza una normalización de los datos, para eso Keras nos permite aplicar un BatchNormalization. Al añadirse esta capa después de la anterior y estar trabajando con un modelo secuencial, no hace falta que de ahora en adelante definamos el formato de los datos de entrada, ya que Keras automáticamente definirá como formatos de datos de entrada el formato de los datos de salida de la capa anterior.

Con los datos normalizados y ya una vez realizada la operación de convolución, se pasa a realizar la operación de pooling añadiendo una capa de MaxPoling2D. Dicho de otra manera, estamos ampliando el número de parámetros de los inputs de manera que podamos establecer mejor las relaciones entre ellos.

Aunque la imagen original es de 2400 por 2400, nuestra red va a trabajar con una submatriz más chica de esa, por lo que la imagen con la que se trabaja no es muy grande. Para poder encontrar más fácilmente las relaciones espaciales entre ellas, se repiten las 3 primeras capas de nuevo, aunque cambiando algunos parámetros. A lo obtenido de la capa de MaxPoling 2D anterior, volvemos a realizar una convolución, sólo que ahora aplicamos 64 filtros. De nuevo se vuelven a normalizar los

resultados y finalmente se aplica otra capa de pooling para reducir los parámetros.

Estas 2 series de convolución→normalización→pooling de nuestra red neuronal hacen las veces de «ojo» de nuestra red. Leeran los datos entrada buscando relaciones espaciales entre ellos de manera que pueda inferir algo. Estos datos son servidos a una serie de nodos densamente conectados que terminarán ofreciendo el valor deseado.

Los resultados de una convolución siempre son datos en formato de 4 dimensiones, donde cada dimensión corresponde al número de muestras, filas, columnas y canales. Nuestra CNN como resultado debe dar un único valor, correspondiente al valor del índice fAPAR para el píxel deseado. Es por esto que ahora se añade una capa Flatten. Esta capa realiza una disminución de las dimensiones a trabajar, de manera que ahora podamos usar capas de 2 dimensiones como son las Dense, que realizarán la lógica del regresor.

Una vez que hemos realizado la reducción de la dimensión, podemos añadir nuestras capas de nodos completamente conectados. En nuestro caso definimos 3 capas de nodos plenamente conectados. Primero una capa de mil nodos, con una activación de tipo «softmax». Tras esta colocamos una capa del mismo número de nodos que los parámetros de salida del flatten, plenamente conectada con activación «relu». Finalmente una capa de un nodo con activación tipo «linear» que nos dará el resultado esperado.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d 1 (Conv2D)            (None, 6, 6, 32)          928

batch normalization 1 (Batch (None, 6, 6, 32)          128

max pooling2d 1 (MaxPooling2 (None, 3, 3, 32)          0

conv2d 2 (Conv2D)            (None, 2, 2, 64)          8256

batch normalization 2 (Batch (None, 2, 2, 64)          256

max pooling2d 2 (MaxPooling2 (None, 1, 1, 64)          0

flatten 1 (Flatten)          (None, 64)                0

dense 1 (Dense)              (None, 64)                4160

dense 2 (Dense)              (None, 32)                2080

dense 3 (Dense)              (None, 1)                 33
=================================================================
Total params: 15,841
Trainable params: 15,649
Non-trainable params: 192
```

Ilustración 9: Sumario de la CNN diseñada

3. Experimentos y análisis de resultados

3.1 Diseño de los experimentos

Como se ha explicado previamente, la CNN esta diseñada para que se le pase una ventana de píxeles y ella como resultado devuelva el valor del índice fAPAR del índice del medio de la ventana.

Se han diseñado distintos tipos de tamaños de ventana, de mayor y menor tamaño, para ver como afectan las zonas colindantes al resultado calculado del índice del medio. Tal y como se explicó en la introducción, se está trabajando con una zona que hace límite entre dos tipos de climas, por lo tanto es interesante ver las relaciones espaciales entre estos dos climas y cómo pueden afectar a la precisión de los resultados obtenidos por la CNN.

Como se están trabajando con volúmenes de datos muy grandes, la primera vez que se ejecuta el programa se guardan las matrices con los datos que nos interesan en ficheros de extensión .npy. Estos tipos de ficheros son ficheros que genera numpy con matrices muy ligeros, por lo que para poder realizar múltiples operaciones se leen directamente esos ficheros y no los ficheros .HDF muy pesados que hace que el entrenamiento de la CNN sea muy lenta en cada experimento.

Para poder medir la precisión de los resultados se hace uso del RMSE[17] para ver como según se van haciendo iteraciones el error que se va obteniendo de la red neuronal. Se están ejecutando 150 epochs, ya que se han observado que más de esas supone un gasto de computación inútil, ya que apenas hay variación del RMSE obtenido.

Para cada ventana de datos se realizan varias ejecuciones y imprime una gráfica con la media del RMSE obtenido en cada epoch. De esta manera se podrá ver sí el RMSE resultante para cada ventana es aceptable o no. Aunque se realizan varias ejecuciones para cada tamaño de ventana, el píxel al que estamos estimando su índice fAPAR siempre ha sido el mismo, el que se corresponde con la zona señalada por el Departament de prevenció d'incendis.

Los resultados del índice fAPAR según indica la descripción del producto oscila entre 0 y 255, por lo tanto tenemos 256 valores posibles.

Para la realización de los experimentos se han descargado un total de 200 imágenes de los distintos productos. Es decir, 200 imágenes con los valores de las 7 bandas de reflectancia y 200 imágenes con los valores del índice fAPAR. Estas 200 imágenes se han dividido en datos de train/test con la función train_test_split de sklearn. La proporción utilizada se ha dejado la de por defecto, es decir el 25% de los datos serán para validar mientras el 75% para entrenar. Esto hace que tengamos 150 imágenes de entrenamiento y 50 para validar. El RMSE que se calcula

en cada experimento es sobre los datos de test, para comprobar la validez del entrenamiento realizado.

Se ha especificado un tamaño de batch_size de la CNN de tamaño 10, así como 150 epochs. Estos valores se han obtenido tras unos primeros experimientos en la que se se detecta que con estos valores se empiezan a estimar unas resultados correctos y sin excedernos en el tiempo de cómputo. Como se podrá ver en los resultados de los experimentos, los resultados obtenidos se mantienen estables sin seguir mejorando a partir de la epoch número 100, por lo que tenía poco sentido aumentar este número.

3.2 Resultados de los experimentos

3.2.1 Ventana de 7x7

Se comienza con un tamaño de ventana pequeño, para ver como afectan los píxels vecinos en la estimación del índice. Este tamaño está considerando los vecinos con distancia hasta 3 píxels del píxel central al que se le va a calcular el fAPAR. Con este tamaño de ventana la gráfica del RMSE que se obtiene es la siguiente:

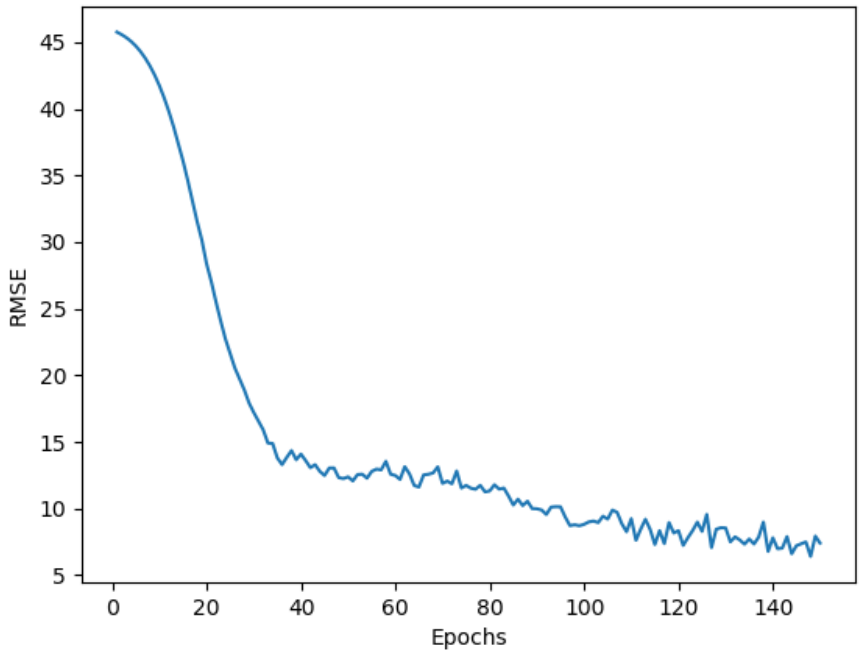

Ilustración 10: RMSE con tamaño de tamaño de ventana 7x7

Tal y como se puede ver, aunque hay una gran reducción del error en las primeras iteraciones, al llevar a la iteración 40 el error empieza a ser más estable, incluso hay pequeñas subidas. El valor también parece bastante alto hasta la iteración 120 donde ya empieza a estar por debajo de 10, llegando a estar cercano al 5 al final de las 150

iteraciones, el cual teniendo en cuanto el número de valores posibles con el que se trabaja, 256, se considera un valor aceptable.

3.2.2 Ventana de 15x15

Ampliamos el tamaño de la ventana, hasta traer los píxels vecinos de hasta distancia 7 con el píxel central. Con este tamaño de ventana, tras realizar de nuevo varias ejecuciones, se obtiene la siguiente gráfica:

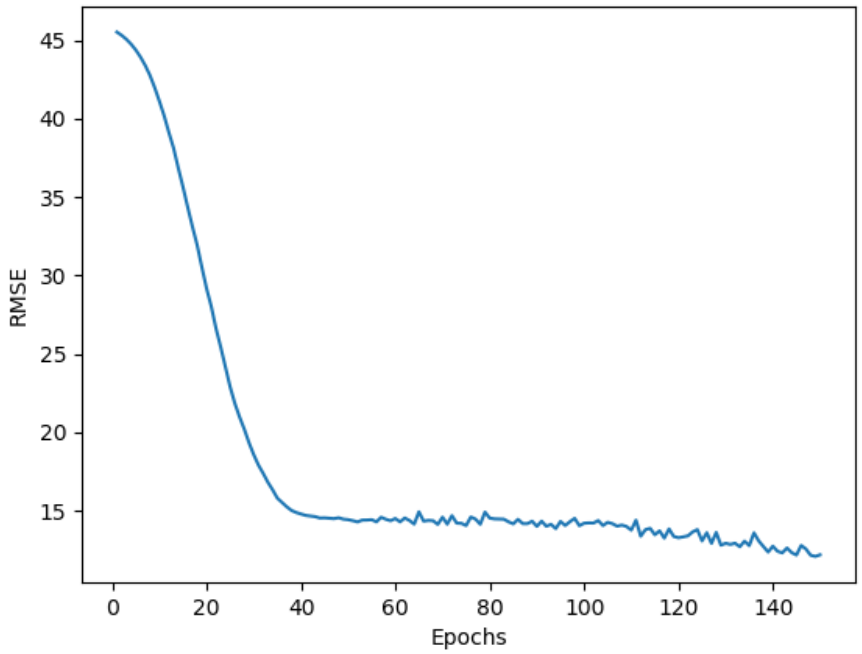

Ilustración 11: RMSE con tamaño de tamaño de ventana 15x15

Aquí el error se muestra mucho más estable y además baja a niveles aceptables mucho antes que en la ventana anterior. Mientras que en la ventana 7x7 con 40 iteraciones aún se tenía un RMSE con valor entorno al 40, en este caso con 40 iteraciones prácticamente estamos ya con un RMSE de valor 15. Lo que ocurre que ya se queda estable en este valor y baja de manera muy gradual, llegando a estar con valor por debajo al 10 en 150 iteraciones. Parece que la ventana anterior, aunque más lenta y con picos, al final alcanzaba un RMSE más bajo.

3.2.3 Ventana de 31x31

Se vuelve a ampliar la ventana. Esta vez estamos cogiendo vecinos con distancia a 15 píxels del píxel central. Tras varias ejecuciones de

la red con este tamaño de ventana se obtiene la siguiente gráfica del RMSE:

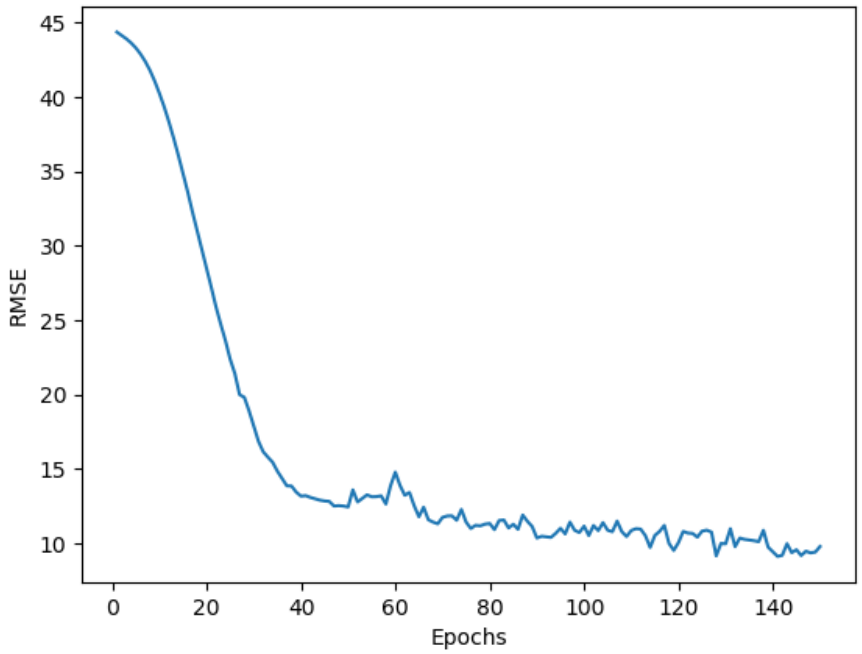

Ilustración 12: RMSE con tamaño de tamaño de ventana 31x31

Los resultados mostrados son mejores que los de la ventana 15x15, siendo parecidos a los de la ventana inicial. De nuevo hay una gran bajada entorno a la iteración 40, que deja el RMSE ya por debajo del 15. Es curioso ver como luego hay una subida relativamente importante, ya que se alcanzan valores cercanos al 20 de RMSE durante varias iteraciones, pero aproximándose a iteración 120, aun con picos más pronunciados, el error vuelve a quedarse cerca del 10.

3.2.4 Ventana de 51x51

Tamaño de ventana mayor de todos los experimentos llevados a cabo. Se cogen los vecinos con una distancia de 25 píxels del píxel central. Como en todos los casos anteriores, tras varias ejecuciones de la CNN se obtiene la siguiente gráfica:

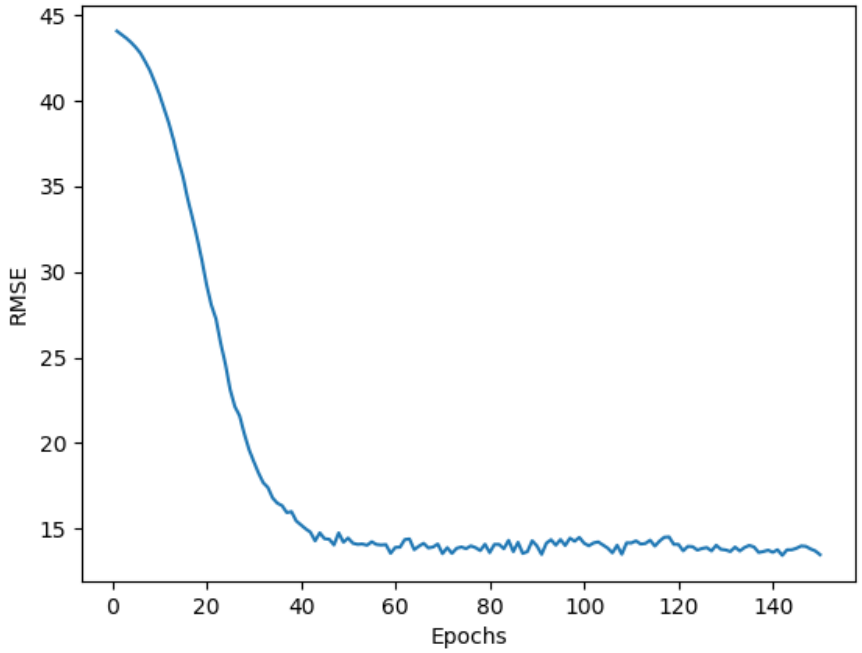

Ilustración 13: RMSE con tamaño de tamaño de ventana 51x51

Se vuelven a tener resultados muy parecidos al segundo caso. El RMSE baja rápidamente entorno al valor 15, pero se mantiene estable en ese valor, no acercándose ni siquiera a 10 al final de las iteraciones. Además el desempeño de la CNN con esta configuración ha empeorado bastante. No sólo el valor del RMSE se mantiene más alto que con tamaño de ventanas menores, es que también el tiempo que pasa en ejecutar cada iteración ha pasado del segundo o incluso menos de las ventanas más chicas, a tardar casi 20 segundos por iteración de este tamaño.

Por lo tanto estamos ante unos peores resultados y tiempo de ejecución mayor, por lo que es claramente el peor de los casos vistos hasta ahora

3.2.5 Ventana de 5x5

A la vista de los experimentos anteriores, se decide realizar otro experimento con un tamaño de ventana menor que los anteriores. En este caso se cogen los píxels con una distancia de 2 del píxel central. La gráfica del RMSE que se obtiene con este tipo de ventanas es la siguiente:

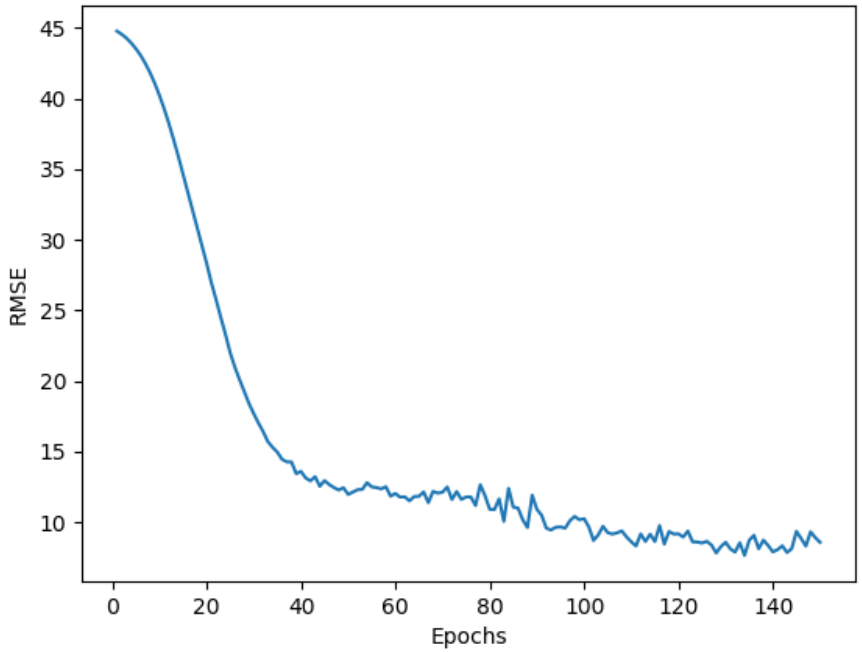

Ilustración 14: RMSE con tamaño de tamaño de ventana 5x5

Con el tamaño de ventana más pequeño se obtienen resultados parecidos a los mejores, aunque con una menor precisión que el caso de ventana de 7x7. El RMSE tarda más en bajar hasta valores del cercanos al 10 que en otros casos, con picos pronunciados, pero se puede ver que finalmente se llega a valores de RMSE de 10, mientras que por ejemplo en en el caso de ventana de 51x51 este valor se mantiene en 15.

3.3 Análisis de los resultados

Tras la realización de los 5 experimentos se concluye que el producto resultante es viable. Teniendo en cuenta que el índice fAPAR oscila entre 0 y 255, hemos conseguido unos valores de errores que oscilan entre el 3.5% de los mejores casos, al 4.5% de los peores. Curiosamente parece que al ampliar el tamaño de la ventana, el desempeño de la CNN empeora, resultando en procesamientos más largos y en precisiones más pobres.

Tamaño Ventana	RMSE Medio	Mejor RMSE	Tiempo medio
5x5	8,9 (3,94%)	8,4 (3,2%)	1,05 segundos
7x7	8,7 (3,4 %)	7,3 (2,8%)	1,15 segundos
15x15	12,3 (4,8 %)	10,1 (3,9%)	1,89 segundos
31x31	13,1 (5,11%)	8,9 (3,47%)	4,25 segundos
51x51	14,5(5,66 %)	12,5 (4,88%)	18,3 segundos

Tabla 7: Resumen de los resultados de los experimentos realizados

Esto puede ser debido a que aunque existen relaciones espaciales, no a grandes distancias, de manera que el los índices de reflectancia de una zona lejana al píxel que estamos analizando en realidad no está aportando información útil para el cálculo del su índice fAPAR.

Así pues, parece que el experimento con mejor desempeño ha sido el primero, ofreciendo un RMSE que quedaba por debajo de 10, es decir, un error por debajo del 3.5% , teniendo un tiempo de procesamiento bastante rápido y además estos valores eran más estables conforme pasaban las épocas. Realmente la tasa de error no ha sido muy alta en ninguno de los experimentos, pero si ha aumentado exponencialmente el tiempo de cálculo.

La CNN diseñada aun tiene mucho margen de mejora, empezando por tener una mayor cantidad de datos para el entrenamiento que mejore la precisión de la misma, pero a la vista de los resultados obtenidos parece un producto muy a tener en cuenta que puede suplir los resultados obtenidos por el producto MODIS.

4. Conclusiones

A continuación se pasan a detallar las conclusiones que se han podido obtener durante el desarrollo de este proyecto, las cuales pretenden dar respuesta a los objetivos iniciales de éste.

- Queda demostrado que es viable la construcción de una CNN que sea capaz de estimar el valor de un índice vegetal a partir de los valores de las distintas bandas de reflectancia de infrarrojo obtenidos por satélite.
- Durante todo el proceso de elaboración de este proyecto se han alcanzado una serie de conocimientos que no se tenían previamente, tales como:
  1. Se ha adquirido conocimientos sobre las técnicas de teledetección por satélite básicas, así como el uso de los distintos índices vegetales más importantes.
  2. Se ha profundizado en el uso y manejo de Python así como de las librerías de carácter científico usadas durante la asignatura.
  3. Se ha adquirido conocimientos sobre redes neuronales en general, y en particular en el funcionamiento y manejo de redes neuronales convolucionales y sus distintos usos.
- Con el transcurso de los distintos experimentos se ha visto que en este caso concreto, cuanta más distancia se tenga del píxel central que se desea calcular el desempeño de la CNN empeora, por lo que las relaciones espaciales parecen que a grandes distancias no tienen demasiada importancia.
- Se ha desarrollado con éxito un prototipo de CNN capaz de dar resultados sobre un índice en concreto para un píxel. En caso de querer tener el valor de varios píxels, se tendría que ir desplazando la ventana que se introduce como input a la CNN para que de los distintos valores.

Teniendo en mente todo lo expuesto, se considera que se han cumplido todos los objetivos que se habían propuesto al principio del proyecto. También se ha adquirido un conocimiento que no se tenía al principio del proyecto tanto sistemas de **deep learning** en general, como de redes neuronales en particular. Para terminar se ha conseguido diseñar con éxito un modelo para detectar el detectar el índice fAPAR de forma inmediata y que puede funcionar en cualquier equipo, de forma que no se dependa del sistema MODIS, pudiendo disponer de este índice en periodos que MODIS no contempla, finalizando con un análisis sobre los resultados de los distintos experimentos obtenidos, llegando a una conclusión sobre ellos.

Para llegar a la finalización del proyecto, ha sido imprescindible seguir la planificación y metodología de trabajo definidas al principio del mismo. Durante el transcurso del proyecto la planificación ha sufrido de algunos imprevistos, sobre todo en lo referentes a la obtención de los datos que supuso un cuello de botella grande en las últimas etapas del mismo, pero gracias a que se hizo una planificación realista, con una estimación de tiempos y esfuerzos correcta que se siguió desde el primer día, estos

retrasos fueron absorbibles y se puedo llegar a la conclusión del proyecto en los tiempos estimados.
El proyecto fue definido para que se pudiera trabajar en forma paralela en varios momentos del mismo, por lo tanto aunque en algunos momentos existió el retraso por los problemas de comunicación, en realidad el proyecto nunca es paró y se pudo seguir avanzado en otros aspectos.

Creo la planificación y la metodología escogidas han significado un gran paso hacía el éxito del proyecto, que de otro modo pudiera haberse detenido.

Para finalizar, hay ciertos puntos en los que este proyecto no aborda y que sería interesantes de llevar a cabo:

- Actualmente el sistema sólo devuelve el valor del índice para un sólo píxel. Una futura mejora sería diseñar un sistema que te devuelva el valor del píxel para toda una zona, de manera que de una sola ejecución tengas varios valores analizados.
- También se está trabajando con una salida única, es decir el devuelve el valor de un sólo píxel. Se podría trabajar para que devolviera el valor de varios índices.
- Para mejorar el entrenamiento de la CNN sería muy interesante contar con un mayor tamaño de datos de entrada. Actualmente se están usando datos de un periodo de 2 años, pero hay registros de mucho más tiempo. Se podría diseñar un sistema que automatización de obtención dichos datos para poder tener un entrenamiento mejor.
- No se está haciendo uso de la potencia de la GPU para las ejecuciones de la CNN, un punto de mejora sería optimizar el código para además de la CPU también use la GPU.
- Otro dato que no se ha tenido en cuenta a la hora de diseñar la red han sido otros datos aparte de los valores de las bandas de reflectancia. Se considera que hay otros datos que pueden influir en el valor del índice y que se podrían añadir como parámetros de input de la CNN. Por ejemplo se podía considerar tener como input además de los ficheros HDF, un histórico con los datos de las precipitaciones de la fecha, de forma que también se encuentre una relación entre las lluvias el fAPAR.

Es decir, este es un primer paso en la construcción de un modelo de procesamiento de imágenes multiespectrales para la obtención de datos cuantitativos en que nos den una imagen sobre el estado de la vegetación de la zona, pero esta versión aun tiene gran margen de mejora que lo convertiría en una gran herramienta para la teledetección y control de las parcelas vegetales.

5. Glosario

CNN: Acrónimo de Convolutional Neural Network, Red neuronal convolucional. Es un tipo de red neuronal que al comienzo de la red usa unas capas encargadas de hacer operaciones de convolución y pooling. Estas redes se han demostrado especialmente útiles en trabajos de visión artificial y tratamiento de ficheros de audio.

Convolución: Función matemática que aplicada sobre una imagen devuelve un “mapa” de las características de la imagen.

fAPAR: Acrónimo de Fraction of Absorved Photosynthetically Active Radiation, Fracción de radiación fotosintéticamente activa absorbida. Es un índice vegetal fundamental en las disciplinas relacionadas con el cambio climático.

MODIS: Acrónimo de Moderate-Resolution Imaging Spectroradiameter. Es un instrumento científico de la NASA que se encuentra orbitando por la Tierra. Ofrece una serie de productos que son de ayuda en la teledetección y medición de diferentes índices vegetales, así como de temperatura, estado de los océanos y concentración de aerosoles entre otras cosas.

Pooling: Conjunto de operaciones matemáticas realizadas sobre un conjunto de datos para reducir las dimensiones de los mismos. La capa de pooling se suele colocar después de una capa de convolución para tener un mejor ajuste en los nodos de la red.

Red Neuronal: Es un modelo computacional inspirado en las neuronas físicas. Consiste en una serie de nodos, o neuronas artificiales, conectadas entre sí que realizan una serie de operaciones a los datos que reciben y devuelven el resultado a la siguiente neurona. Existen distintos tipos de neuronas, como las neuronas convolucionales.

RMSE: Acrónimo de Root-mean-square deviation, Raíz del error cuadrático medio. Medida de desempeño cuantitativa utilizada comúnmente para evaluar métodos de pronostico. Consiste en la raíz cuadrada del sumatorio de los errores al cuadrado.

6. Bibliografía

[1] Web Oficial de MODIS Land. Última vez consultado el 3 de Junio de 2018: https://modis-land.gsfc.nasa.gov/

[2] James A Anderson, An Introducction To Neural Networks, MIT Press, 1995

[3] Paper de la Nasa sobre NDVI. Última vez consultado el 3 de Junio de 2018: https://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_2.php

[4] Web wiki Landscape Toolbox. Última vez consultado el 3 de Junio de 2018:http://wiki.landscapetoolbox.org/doku.php/remote_sensing_methods:soil-adjusted_vegetation_index

[5] Paper sobre diferentes índices vegetales. Última vez consultado el 3 de Junio de 2018: http://www.mdpi.com/2072-4292/7/4/4026/htm

[6] Canopy Water Content. Última vez consultado el 3 de Junio de 2018: http://www.harrisgeospatial.com/docs/CanopyWaterContent.html

[7] Martínez, B.* , Albargues, E. , Camacho, F. , Moreno, A. , Gilabert, M.A. Estimación de la fAPAR sobre la Península Ibérica a partir de la inversión del modelo de transferencia radiactiva 4SAIL2. Revista de Teledetección. Página 61 a página 77.

[8] Web oficial de Python. Última vez consultado el 3 de Junio de 2018: https://www.python.org/

[9] Web oficial de Anaconda. Última vez consultado el 3 de Junio de 2018: https://www.anaconda.com/

[10] Web oficial de Scikit-learn. Última vez consultado el 3 de Junio de 2018: http://scikit-learn.org/stable/index.html

[11] Web oficial de TensorFlow. Última vez consultado el 3 de Junio de 2018: https://www.tensorflow.org/

[12] Web oficial de Keras. Última vez consultado el 3 de Junio de 2018: https://keras.io/

[13] Web oficial de la librería PyHDF. Última vez consultado el 3 de Junio de 2018: https://hdfeos.org/software/pyhdf.php

[14] Web oficial de MODG09GA. Última vez consultado el 3 de Junio de 2018: https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod09ga_v006

[15] Web oficial de MOD152AH. Última vez consultado el 3 de Junio de 2018: https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod15a2h_v006

[16] Wikipedia sobre RMSE. Última vez consultado el 3 de Junio de 2018: https://en.wikipedia.org/wiki/Root-mean-square_deviation

7. Anexos

7.1. Código fuente de la CNN.

```
import numpy as np
import matplotlib.pyplot as plt
import os
import glob
import keras
from pyhdf.SD import SD, SDC
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras import backend
from pathlib import Path

class AccuracyHistory(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.acc = []

    def on_epoch_end(self, batch, logs={}):
        self.acc.append(logs.get('rmse'))

def rmse(y_true, y_pred):
        return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))


epochs = 150
batch_size = 10

dirname = os.path.dirname(__file__)

"""
El pixel que vamos a buscar se va a encontrar en la posición X = 450, Y = 500
Hacemos una ventana de distancia 2, por lo que será [448:553, 498:503]
"""

"""
Obtener el array de los datos de entrada. Obtenemos los datos de las 8 bandas.
"""

X = []
x_file = Path(dirname+'/x_2.npy')
if x_file.is_file():
    X = np.load(dirname+'/x_2.npy')
```

```
else:
    dir = sorted(glob.glob(dirname+"/Inputs/*.hdf"))

    for file in dir:
        bands = []
        hdf = SD(file, SDC.READ)

        for i in range(1,8):
            sdsObj = hdf.select('sur_refl_b0'+str(i)+'_1')
            aux = sdsObj.get()
            band = np.array(aux[448:453, 498:503])
            bands.append(band)

         data = np.dstack((bands[0], bands[1], bands[2], bands[3], bands[4],
bands[5], bands[6]))
        X.append(data)

    X = np.array(X)
    np.save(dirname+'/x_2.npy', X)
"""
Obtener el array de datos de salida
"""
y = []
y_file = Path(dirname+'/y.npy')
if y_file.is_file():
    y = np.load(dirname+'/y.npy')
else:
    dir = sorted(glob.glob(dirname+"/Outputs/*.hdf"))

    for file in dir:
        hdf = SD(file, SDC.READ)
        sdsObj = hdf.select('Fpar_500m')
        aux = sdsObj.get()
        data = aux[450, 500]
        y.append(data)

    y=np.array(y)
    np.save(dirname+'/y.npy', y)

X_train, X_test, y_train, y_test = train_test_split(X, y)

model = Sequential()
model.add(Conv2D(32,            kernel_size=(2,            2),activation='relu',
input_shape=X_train.shape[1:]))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(1, 1), strides=(1, 1)))

model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(1, 1)))
```

```
model.add(Flatten())
model.add(Dense(1000, activation='softmax'))
model.add(Dense(256, activation='relu'))
model.add(Dense(1))

model.compile(loss='mse', optimizer='adam', metrics=[rmse])

history = AccuracyHistory()

model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs,
validation_data=(X_test, y_test), callbacks=[history])
score1 = model.evaluate(X_test, y_test, batch_size=batch_size)

plt.plot(range(1,151), history.acc)
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.show()
print(score1)
```