UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

Análisis supervisado de imágenes aéreas usando técnicas de aprendizaje profundo aplicado a detección de cultivos tropicales

Supervised analysis of aerial images using Deep Learning applied to tropical crops detection

Realizado por
Cristian Cardas Ezeiza

Tutorizado por
Ismael Navas Delgado
José Manuel García Nieto

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, Junio 2020

UNIVERSIDAD
DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Análisis supervisado de imágenes aéreas usando técnicas de aprendizaje profundo aplicado a detección de cultivos tropicales

Supervised analysis of aerial images using Deep Learning applied to tropical crops detection

Realizado por:

Cristian Cardas Ezeiza

Tutorizado por:

Ismael Navas Delgado

José Manuel García Nieto

Departamento:

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2020

Resumen

En la actualidad, el enorme volumen de datos accesibles sobre diversas áreas de conocimiento y los modelos matemáticos, junto al poder de procesamiento de los computadores, están dando pie a numerosas aplicaciones que aprovechan la información que se puede obtener a partir del estudio de estos datos, para crear e incluso mejorar procesos realizados por seres humanos. En este sentido, el dominio de aplicación de la agricultura no es una excepción, sino que está emergiendo como línea de desarrollo de software centrado en el tratamiento y el análisis de datos. De manera particular, el análisis de datos provenientes de imágenes de dispositivos de Observación de la Tierra y la tele-detección por sensores remotos está proporcionando un marco de trabajo idóneo donde el uso de las técnicas automáticas de aprendizaje máquina están tomando un papel central. Un ejemplo de este tipo de aplicaciones es el proyecto Green-Senti del I Plan Propio Smart Campus de la Universidad de Málaga, mediante el que se analizan las zonas verdes del campus universitario mediante imágenes multi-espectrales para estudiar su evolución, aprovechando los datos abiertos de los satélites Sentinel 2.

Este trabajo de Fin de Grado (TFG) viene motivado por la linea abierta en el proyecto Green-Senti, aunque explorando un enfoque complementario. En concreto, se realiza un servicio de clasificación mediante imágenes aéreas de cultivos, gracias a la generación de un repositorio de imágenes etiquetadas y el entrenamiento de un modelo de Redes Neuronales Convolucionales. El análisis llevado a cabo se ha centrado en los cultivos tropicales del Mango y del Aguacate, de gran importancia en la comarca malagueña de la Axarquía. El objetivo es proporcionar a la industria agroalimentaria de aplicaciones novedosas para la traza de la evolución de estos cultivos, de cara al soporte estratégico.

Con este objetivo en mente, se ha construido un ecosistema web llamado *"Manvocado"*, basado en un modelo de interacción humano-máquina en diferentes fases de captación, etiquetado, entrenamiento de motor de aprendizaje profundo, validación y predicción. El proceso se retro-alimenta mediante la captación de nuevas imágenes, su etiquetado automático y su validación por parte del experto humano.

Cabe destacar que Manvocado ha sido probado por los técnicos de la Cooperativa Trops, S.A.T. 2803, adjuntando informe de valoración positiva y expresando su interés en seguir con el desarrollo de nuevas versiones, además de para su utilización en imágenes atendiendo a diferentes funcionalidades agrícolas (variedades, patologías, etc.).

Palabras clave: Aprendizaje Profundo, Imágenes aéreas, Redes Neuronales, Cultivos Tropicales, Agricultura de Precisión

Abstract

Nowadays, the enormous volume of accesible data on several fields of knowledge and the mathematical models, along with computing power, are leading towards very different kinds of applications that make use of the valuable information that can be extracted from such data, with means to create or even improve processes or actions performed by human beings. In this line of thought, the farming application domain is not an exception, as it is in fact emerging as a line of development of software focused on data analysis. In particular, the data analysis of images from Earth Observation and Remote Sensing devices, are providing a suitable line of work where machine learning techniques are becoming a crucial part. For example, the project Green-Senti del I Plan Propio Smart Campus of the University of Malaga, analyses the campus' green zones using multi-spectral satellite images, collected from the open data provided by the satellite Sentinel 2.

This Final Year Dissertation, is influenced by the line of work from the Green-Senti project, although exploring a complementary approach. Specifically, a classification service with aerial crop images is built, with the help of the generation of a labeled-image repository and the training of a Deep Convolutional Neural Network model. The analysis is focused on tropical crops such as avocado and mango, which are very important in a region from Malaga called "Axarquía". The objective is to provide to the agri-food industry novel applications to monitor the evolution of the crops for the strategic value.

With this objective in mind, an ecosystem called "Manvocado" was built. It is based on a human-machine interactive model for the distinct phases like obtaining data, its classification, the Deep Learning model training, and the validation and prediction of values. The process is fed back with new images, its automatic classification and validation from the human expert's end.

It is worth noting that "Manvocado" has been tested by technicians from Trops, S.A.T. 2803, including a report with positive feedback and expressing their interest in developing further versions, along with its use on other kinds of functionalities, such as varieties, pathologies, etc.

Keywords: Deep Learning, Satellite Imagery, Neural Networks, Tropical Crops, Precision Farming

A mis padres y a mis queridos amigos, por siempre estar ahí para mi.
Gracias.

Índice

1. Introducción

1.1. Contexto y problema

Debido a la alta precisión y gran cobertura que ofrecen los últimos sensores y cámaras de los satélites, el campo del Remote Sensing [Blaschke, 2010] está dando fruto a diversos tipos de nuevas aplicaciones. Nos movemos en un entorno de Big Data, ya que el volumen de datos abiertos accesibles es inmenso, por eso mismo, y gracias al desarrollo y el aumento de la capacidad computacional, están surgiendo diversos estudios denominados de "Observación de la Tierra".

En este contexto, surge el proyecto Green-Senti [1] del I Plan Propio Smart Campus de la UMA, que proporciona un servicio web de monitorización de zonas verdes del campus de la Universidad de Málaga. El estudio se realiza mediante el análisis de imágenes de los satélites Sentinel, que pertenecen al programa Copérnico[2]. Mediante el uso inteligente de esta gran cantidad de información, el número de aplicaciones prácticas que pueden desarrollarse dependen de nuestra imaginación.

Al igual que con Green-Senti, la vista se está alzando sobre aplicaciones para diversos dominios específicos, en busca de la mejora e incluso propuesta de nuevas estrategias en sectores que están comenzando a apostar por el uso de nuevas tecnologías. Así surge el dominio de la Agricultura de Precisión, donde se pretenden desarrollar procesos que, mediante el análisis y la explotación de datos, provean información que favorezca la toma de decisiones y la organización óptima de los cultivos [Fuglie, 2016].

En este Trabajo de Fin de Grado, se pretende explorar el uso de uno de los modelos de computación más populares en la actualidad, la Red Neuronal Convolucional [O'Shea and Nash, 2015]. Este modelo, aplicado a imágenes aéreas, tiene como fin el proveer a la comunidad agroalimentaria de una aplicación novedosa capaz de trazar la evolución de los cultivos de cara al soporte estratégico. Para ello, se ha construido un ecosistema entorno al modelo, para favorecer al desarrollo y a los resultados obtenidos del mismo.

[1] Proyecto Green-Senti: `https://khaos.uma.es/green-senti`

[2] Copernicus Hub: `https://scihub.copernicus.eu/`

1.2. Motivación

Tras el reciente auge del aprendizaje profundo, este tipo de modelos de redes neuronales ha invadido el mundo del aprendizaje computacional, prácticamente reemplazando a otros tipos de algoritmos convencionales al demostrar grandes resultados en la mayoría de campos. Sin embargo, dada la complejidad de la arquitectura y de los principios en los que se basan, las redes neuronales típicas del aprendizaje profundo se utilizan como cajas negras, funcionan pero no se sabe bien porqué. Además, hay siempre un problema en común cuando se utiliza este tipo de modelos, el volumen de datos requeridos para la obtención de buenos resultados es muy elevado, el cual además depende explícitamente de la calidad de los mismos.

Gracias a la gran cantidad de datos abiertos y a su fácil acceso, explorar la aplicación de redes neuronales en este contexto es llamativo e interesante, siendo la Observación de la Tierra, uno de los campos de investigación más prometedores en la actualidad por la realización de aplicaciones con un enfoque medioambiental, que buscan la construcción de un futuro sostenible. No obstante, es necesario pensar en un modelo constructivo para la generación de estos conjuntos de datos y su etiquetado, donde el experto humano pueda incluir y aportar su conocimiento para enriquecer el modelo, de cara a la obtención de predicciones automáticas de mayor precisión.

1.3. Objetivos

El objetivo de este Trabajo de Fin de Grado es realizar un estudio de la aplicación de técnicas de aprendizaje profundo supervisado con imágenes aéreas, con el fin de acercar el análisis de datos y las tecnologías de la información y sus respectivos beneficios a la comunidad agroalimentaria. Para ello, se ha escogido como caso de uso los cultivos tropicales propios de la provincia de Málaga, especialmente de la comarca de la Axarquía.

Para realizar este estudio, se tienen las siguientes metas:

1. Creación de un dataset de un tamaño considerable para el número de clases escogido (Aguacate, Mango, Estructuras), que contenga imágenes representativas del tipo de clase que representan y presenten el mismo formato.

2. Desarrollo de un clasificador de imágenes fiable a partir del dataset propuesto.

3. Acercar la funcionalidad del clasificador a la comunidad agrónoma.

Para lograr las metas, se presentarán:

- Dataset de recortes de árboles de aguacate, mango, estructuras como casas, reservorios de agua; obtenidos de Google Earth y procesados para que sean imágenes PNG de dimensiones 256x256.
- Red Neuronal Convolucional que realiza la clasificación de las imágenes.
- Despliegue de una aplicación web que integre las funcionalidades del modelo mediante una interfaz amigable, para facilitar la interacción con el usuario experto en el dominio de la aplicación.
- Repositorios GitLab con los scripts, instrucciones de uso y casos de test. `https://gitlab.com/thecristian1409/manvocado-web`

1.4. Estructura del documento

Esta memoria esta organizada en 7 capítulos, que describen todo el procedimiento de la realización de este TFG. Seguidamente, se presenta una breve introducción a cada uno de ellos.

- El capítulo 2 consiste en un repaso de las tecnologías y modelos que se han empleado en la realización del trabajo, principalmente Redes Neuronales Convolucionales y tecnologías web. Finalmente, se presenta el estado del arte de los modelos neuronales mostrados y su contribución al campo de la Observación de la Tierra.
- El capítulo 3 muestra en detalle todo el desarrollo software realizado en este TFG.
- El capítulo 4 describe todos los experimentos que se han llevado a cabo durante la realización de este trabajo, comentando en detalle la problemática del uso de este tipo de modelos, que requieren de una clasificación previa.
- El capítulo 5 expone los distintos casos de uso que presenta la aplicación web desarrollada.
- El capítulo 6 presenta una breve discusión tras haber introducido en su totalidad la propuesta realizada y los distintos casos de uso.

- El capítulo 7 se realiza una conclusión al trabajo y a la vez se proponen posibles líneas de trabajo futuro.

2. Conceptos y Estado del Arte

Dentro del mundo del Aprendizaje Computacional existen distintos tipos de aprendizaje.

- **Aprendizaje supervisado**: Disponemos de un conjunto de patrones de entrenamiento para los que conocemos la salida del algoritmo. De esta forma, el objetivo para diseñar la regla de aprendizaje será minimizar el error cometido entre la salida del algoritmo con respecto a la salida esperada que ya conocíamos. De esta manera, se pretende entrenar el modelo de manera guiada para poder predecir nuevos elementos que se introduzcan a modo de prueba, de los cuales no se conozca ninguna salida, i.e: perceptrón, SVM, random forest.

- **Aprendizaje no supervisado**: En este caso, disponemos de un conjunto de patrones de los cuales no conocemos la salida deseada del algoritmo. El algoritmo por sí solo buscará estructuras o prototipos presentes en este conjunto de datos observando el comportamiento de los datos. Esto se llevará a cabo siguiendo cierto criterio o regla de aprendizaje, i.e: aprendizaje competitivo, , clustering, redes de Kohonen, etc.

- **Aprendizaje por refuerzo**: Se basa en un proceso de prueba y error que busca maximizar el valor esperado de una función, la cual se denomina **señal de refuerzo**. Este paradigma surge en la psicología, donde si una acción supone una mejora en el comportamiento, la tendencia a producir esta acción se refuerza y en caso contrario se debilita. Dispondremos de un conjunto de patrones de entrenamiento y de su respectiva evaluación, la cual informa al modelo sobre su comportamiento con respecto a la entrada recibida, i.e: Q-learning, programación dinámica, etc.

Dada la naturaleza del problema que nos concierne, al tratar con imágenes clasificadas, trabajaremos con modelos propios del Aprendizaje Supervisado, donde una de las familias de modelos más utilizadas por su versatilidad y grandes resultados, son las Redes Neuronales Artificiales, las cuales introducimos a continuación.

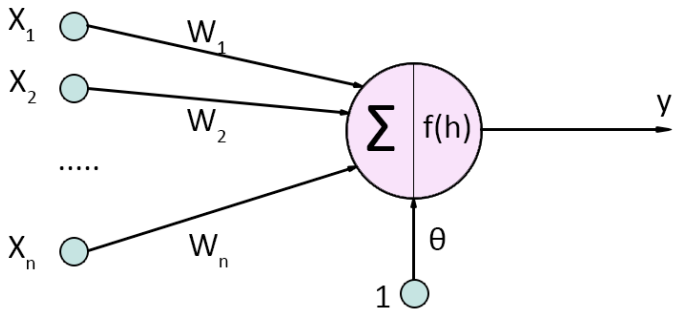
2.1. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son modelos complejos, tanto en estructura, como en técnicas de entrenamiento, es por ello, que para un mejor entendimiento de las mismas, debemos comenzar desde sus inicios, introduciendo el Perceptrón Simple.

2.1.1. Perceptrón simple

El perceptrón simple es la unidad básica de procesamiento de una red de neuronas artificiales, que posee ciertas características de comportamiento inspiradas en el conocimiento que tenemos sobre el funcionamiento de las **neuronas biológicas**. Introducido por [Rosenblatt, 1958], el perceptrón simple se define principalmente mediante sus pesos sinápticos $W_{1..n}$ y su función de transferencia $f(h)$. Como podemos observar en la Figura 1, la unidad recibe varias entradas $x_{1..n}$, sobre las cuales se realiza una suma ponderada por los pesos sinápticos $W_{1..n}$, siendo el valor resultante el potencial sináptico h, que será la entrada a la función de transferencia.

La función de transferencia, en caso del perceptrón simple la función signo, toma el potencial sináptico y mediante un umbral θ, determina la activación o inhibición de la neurona (simulando el comportamiento biológico). Finalmente, tras aplicar la función sobre el potencial sináptico, obtenemos la salida de la neurona que será 1 ó -1.



$$\mathrm{y} = \begin{cases} 1, & \text{si } h > \theta \\ -1, & \text{si } h \leq \theta \end{cases} \qquad \mathrm{h} = w_1 x_1 + ... + w_n x_n$$

Figura 1: Ilustración de los contenidos y estructura perceptrón simple de Rosenblatt

Como hemos comentado anteriormente, estamos realizando aprendizaje supervisado, es decir, en la fase de entrenamiento conocemos la salida deseada del perceptrón para los valores de entrada. Es ahora, que podemos comprobar si la estimación del perceptrón y se corresponde con el valor esperado z, que podemos hablar del concepto de error.

Regla de aprendizaje

Una de las características más importantes de las redes neuronales es la del aprendizaje, el cual es un proceso adaptativo mediante el que se van modificando los pesos sinápticos $W_{1..n}$ de las neuronas para ir corrigiendo el comportamiento de la red. Dada la estructura del perceptrón simple, solo disponemos de los pesos sinápticos que ponderan los datos de entrada, pero en una red neuronal, el proceso es idéntico con los pesos que conectan una neurona con otra.

De manera generalizada, podemos definir la regla de aprendizaje mediante la expresión (1).

$$w_{i,j}(k+1) = w_{i,j}(k) + \Delta w_{i,j}(k) \tag{1}$$

Los pesos sinápticos de la iteración $k+1$ que conectan la neurona i con la neurona j, se definen como la suma de los pesos de la iteración k más un incremento Δ. Donde en el caso del perceptrón (2)

$$\Delta w_i(k) = \eta(k)[z(k) - y(k)]x_i(k) \tag{2}$$

Siendo $\eta(k)$ la tasa de aprendizaje en la iteración k, que dicta cómo de brusco es el cambio que se realiza sobre el peso w_i, es un valor entre 0 y 1.
$z(k) - y(k)$ el error cometido al introducir el patrón de entrada $x_i(k)$. El algoritmo de aprendizaje o entrenamiento acaba cuando tras un número a establecer p de iteraciones, no se han actualizado los pesos.

La corrección del error cometido por el perceptrón se realiza de forma iterativa, consiguiéndose adaptar con el paso del tiempo a los patrones de entrada, sin embargo, la clasificación que aporta una sola unidad de procesamiento nos permite discernir entre 2 clases. Podemos visualizarlo como una división del plano mediante una recta, donde los pesos sinápticos son los coeficientes de la misma. Por lo que este modelo únicamente es

óptimo para problemas donde los datos sean linealmente separables, donde el perceptrón encuentra una solución en un número finito de iteraciones.

2.1.2. Perceptrón multicapa: Gradient Descent & Backpropagation

Para poder aplicar el modelo en problemas donde el conjunto de datos no sea linealmente separable y obtener buenos resultados, debemos ampliar la arquitectura y formar una red de neuronas. El problema es que necesitamos una forma mejor para estimar el error, por lo que debemos actualizar la regla de aprendizaje.

Uno de los problemas del perceptrón era el de su convergencia, ya que no nos garantizaba que se encontrase un balance donde no actualizase sus pesos, de hecho, en problemas con conjuntos no linealmente separables, los pesos del perceptrón oscilan sin encontrar nunca una solución. Es por esto, que requerimos de una regla de aprendizaje que asegure la convergencia, por lo que se empezó a utilizar la técnica de "Gradient Descent", donde aunque no sea al mínimo global, se garantiza la convergencia del modelo.

Como consecuencia de utilizar esta técnica, la regla de aprendizaje debe ser diferenciable, por lo que la función signo que se utilizaba en el perceptrón no se puede utilizar, al ser esta no diferenciable en x = 0 como se observa en la figura 2.

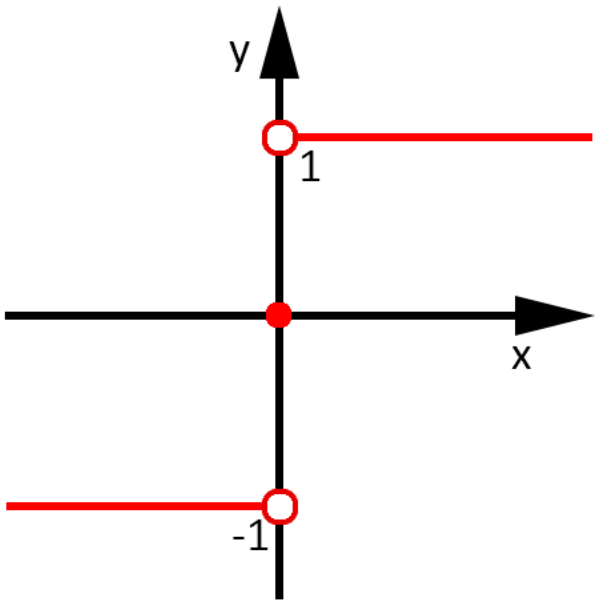


Figura 2: Función signo utilizada por el perceptrón simple

Existen diversas funciones de transferencia diferenciables. Según el tipo de problema,

una proporciona mejores resultados que otras, también dependiendo de la arquitectura y de la estrategia empleada para el aprendizaje de la red. Algunos ejemplos son: tanh, sigmoide, lineal, etc.

Por último, veremos uno de los algoritmos de entrenamiento de redes neuronales más populares debido a su sencillez y eficacia a la hora de modificar los pesos, el algoritmo de *backpropagation* o propagación hacia atrás.

Considerando un perceptrón multicapa o "red neuronal", sabemos que se le introducen unos valores de entrada, también llamados patrones, y seguimos un flujo de izquierda a derecha en la red, donde por cada capa, las neuronas van recibiendo las entradas provenientes de las neuronas de la capa anterior y así hasta el final de la red. Este proceso se denomina *feed forward*, y no es más que para obtener la clasificación final de la red para el patrón de entrada dado.

El algoritmo de backpropagation coge la salida de la red y comienza un procedimiento de derecha a izquierda desde el final de la red, propagando el error cometido (i.e: MSE[3] calculado con respecto a la salida deseada) hacia atrás. Esto se realiza mediante el cálculo de unos δ_l^N para cada neurona N de la capa l como se muestra en la expresión (3) para la última capa y mediante la expresión (4) para las demás.

$$\delta_l^N = (S_l^N)' \cdot [Z_l - S_l] \tag{3}$$

$$\delta_j^N = (\delta_j^N)' \cdot \sum_{i=1}^{M} (w_{i,j}^{n+1} \cdot \delta_i^{n+1}) \tag{4}$$

La propagación del error nos da una "estimación" de cuánto ha contribuido cada neurona al error cometido, siendo las más responsables de esto las neuronas de las últimas capas.

La actualización de los pesos se realiza como en el perceptrón simple, donde en vez de utilizar el error entre la salida esperada y la obtenida, empleamos estos deltas para calcular el incremento Δw.

2.2. Redes Neuronales Convolucionales

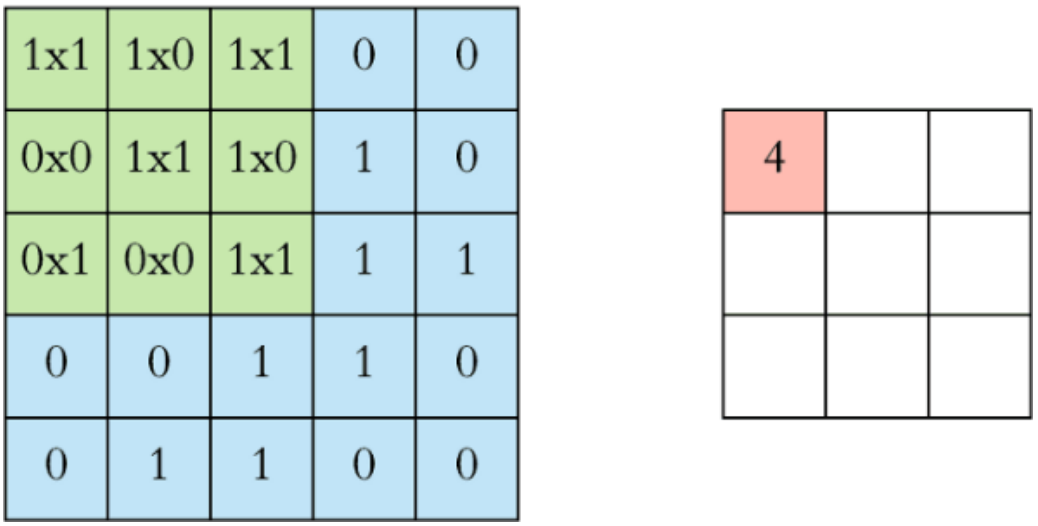
En este Trabajo de Fin de Grado utilizaremos como modelo una DCNN (Deep Convolutional Neural Network), la cual se asemeja a una red de neuronas artificiales común,

[3]MSE: Mean Squared Error

pero que incorpora, además de un número elevado de capas que la hacen denominarse "profunda", capas que realizan una operación llamada convolución.

El mundo de la visión por computador es uno de los campos en los que la Inteligencia Artificial ha supuesto un antes y un después. La primera red neuronal aplicada a imágenes fue creada por [LeCun et al., 1989], que propuso su modelo LeNet1, el cual aplicaba el concepto de convolución. Esta operación se refiere a la combinación matemática de dos funciones para producir una tercera, mezclando ambos conjuntos de información.

En el caso de las redes neuronales, la convolución se realiza en los datos de entrada mediante la aplicación de un **filtro o kernel**, que produce un **mapa de características**. Para esto, se aplica el filtro a toda la entrada, podemos decir, que se va "desplazando" el filtro por la matriz, llevando a cabo multiplicaciones entre los valores de las celdas y sumando en cada localización para obtener este mapa, como puede verse en la figura 3.



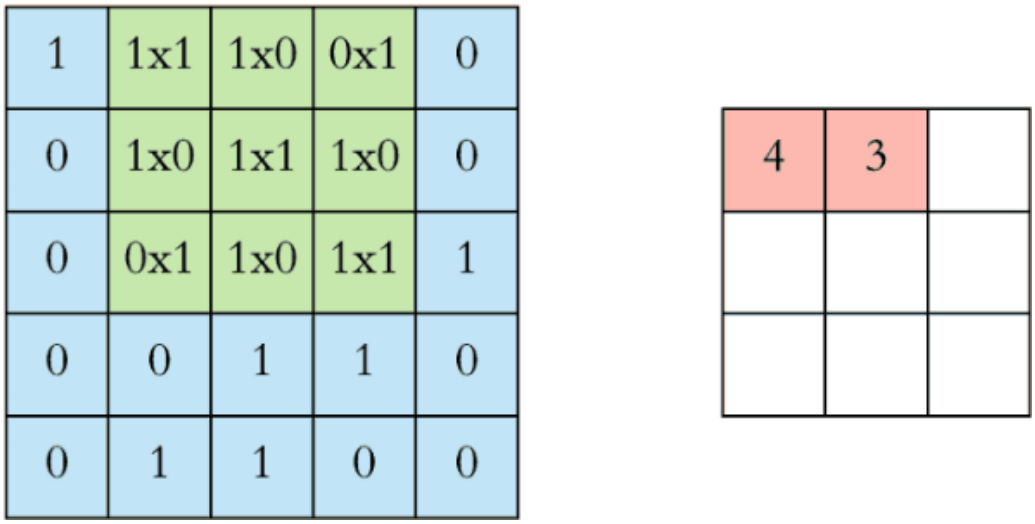


Figura 3: Extracción de características de una matriz con un filtro mediante la convolución. Fuente: `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`

Por cada celda del filtro (representado en verde), realizamos un producto donde el primer operando corresponde con el valor de la celda de la matriz de entrada (representado en azul) y el segundo con el valor de esa celda del filtro. Una vez realizadas todas las multiplicaciones, se suman los valores obtenidos y ese valor se almacena en la matriz resultante. Como observamos en la figura, estamos reduciendo la dimensionalidad de la matriz de entrada, extrayendo los rasgos más característicos de cada región, la cual es dictada por el tamaño y por los valores del filtro.

Volviendo al mundo de las imágenes, podemos representarlas como matrices numéricas con valores definidos entre 0 y 255, con los 3 canales RGB. De esta forma, podemos obtener matrices que modelan imágenes y trabajar con ellas de manera directa. Así, podemos aplicar filtros mediante la convolución, obteniendo información destacable de las imágenes. Es por esto, que este tipo de redes neuronales lidia tan bien con problemas relativos al mundo de la visión por computador.

La arquitectura de una Red Neuronal Convolucional incluirá entre sus capas varias de convolución, donde se aplicarán distintos filtros a los datos de entrada. Por ejemplo consideremos que estamos pasando por la red una imagen RGB de dimensiones 1920x1080, la matriz que la modela tendrá de dimensiones (1920,1080,3). A esta, la primera capa de convolución le aplicará tantos filtros distintos como se especifique, por ejemplo 32. Además, la dimensión de estos filtros será de 256x256, por lo que la salida de la capa será

una matriz de la forma:

$$[(m + 2p - \frac{f_1}{s}) + 1, (n + 2p - \frac{f_2}{s}) + 1, num\_filters] \tag{5}$$

Vamos a razonar la Expresión (5) introduciendo dos conceptos que complementan a la convolución.

El concepto de **stride** o de "paso", define cuantas celdas se desplazará el filtro por cada paso, a mayor sea el desplazamiento, menor será la matriz de características resultante, en la Expresión (5) el stride se representa mediante la constante s. Podemos observar que en la Figura 3 el stride es de 1.

Por otro lado, tenemos el **padding**, que consiste en añadir información adicional a la matriz de entrada, aumentando sus dimensiones. Por ejemplo, si el padding es de 1, se añade un "rectángulo" exterior de 0s a la matriz. Como hemos podido observar en la Figura 3, el mapa de características tiene dimensiones menores que la matriz de entrada, pero gracias al padding, podemos contrarrestar este encogimiento. Pero lo fundamental es que resuelve un problema con los bordes de la imagen, si observamos las esquinas de la matriz, esos valores de la imagen apenas se tienen en cuenta en el mapa de características, ya que el número de veces que se les aplica el filtro es menor que a otras celdas, como las centrales. De esta forma, aumentando el tamaño de la matriz con 0s, el resultado de aplicar el filtro no se ve influenciado por los valores y permite realizar un análisis mayor de los bordes de la imagen. En la Expresión (5) el padding se representa con la constante p, siendo las demás el tamaño de la imagen m, n y el tamaño del filtro f_1, f_2.

Retomando el ejemplo propuesto, asumimos que no utilizamos padding y que el stride es de 1, por lo que la salida de la capa sería una matriz de dimensiones (1665, 825, 32). Hemos sacrificado la imagen de alta resolución inicial, obteniendo 32 imágenes resultantes de haber aplicado los distintos filtros. Existen distintos tipos de filtros, ya que dependiendo de sus valores, se van a extraer unos rasgos u otros, como detección de bordes, líneas horizontales, etc.

Típicamente, tras una capa de convolución, se añade una capa de "**pooling**", la cual se encarga de reducir la dimensionalidad, disminuyendo así la cantidad de parámetros y simplificando el cómputo de manera significativa. Su objetivo es preservar información relevante y fomentar la invarianza a cambios de posición o condiciones lumínicas, frente al ruido, etc.

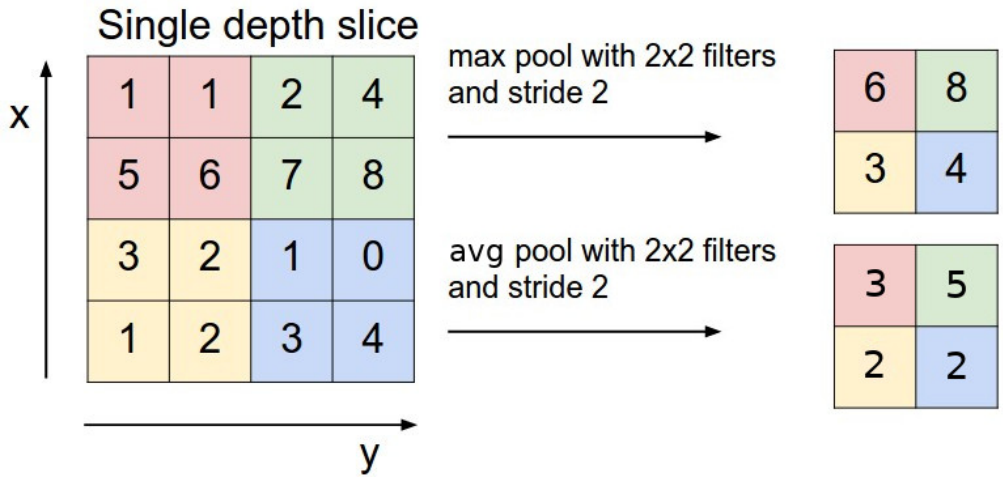


Figura 4: Ejemplo de average y max pooling.

Los 2 métodos más utilizados son average y max pooling. Como podemos observar en la Figura 4, se comporta de manera similar a los filtros, esta vez, realizando la media o escogiendo el máximo de los elementos de la ventana indicada por colores. Es fácil apreciar como al generalizar las regiones cubiertas por la ventana, enfatiza la invarianza ante posibles distorsiones en la imagen.

2.2.1. Aumentación de datos y Validación Cruzada

Numerosos son los estudios y las aplicaciones que utilizan modelos de Aprendizaje Profundo para solucionar problemas, y en todos ellos, siempre hay un inconveniente que es un factor común, el conjunto de datos de entrenamiento.

Para llevar a cabo un entrenamiento con un control óptimo, se requiere particionar el conjunto de datos en 2 partes, el conjunto de entrenamiento y el de testeo. Esta es una cuestión común en el mundo del Aprendizaje Computacional, necesitamos monitorizar el proceso de aprendizaje del modelo, pero si utilizamos los mismos datos para entrenar y para testear, el modelo simplemente clasificaría los datos correctamente al haber entrenado y modificado sus pesos en torno a los mismos. Relacionado con esto, aparece uno de los principales problemas con el entrenamiento de las redes neuronales, el **sobre-entrenamiento**.

El sobre-entrenamiento, del inglés "overfitting", como su nombre sugiere, consiste en entrenar más de lo necesario al modelo, de forma que este se adecúe a los datos con los que entrena, memorizando su estructura, el modelo se vuelve dependiente a los datos de entre-

namiento y no es capaz de generalizar. Para reducir este efecto, reservamos una pequeña parte de los datos para realizar pruebas de forma periódica durante el entrenamiento, con estos NO se modificarán los pesos de la red. De esta manera, podemos realizar una evaluación sobre el rendimiento del modelo en datos que este desconoce y calcular el error para validar su poder de generalización.

Recordemos que el aprendizaje es un proceso iterativo, por lo que la precisión del modelo va variando durante el entrenamiento. Gracias a este conjunto de datos que hemos separado, somos capaces de discernir el punto a partir el cual el modelo ha empezado a empeorar la clasificación de los datos nuevos, por lo que tenemos que parar el entrenamiento para evitar que la red comience a memorizar los patrones de entrenamiento.

El nombre correcto de este proceso de prueba se denomina comúnmente **validación**, ya que el testeo está más reservado a la hora de evaluar el rendimiento del modelo final.

Podemos observar que de nuestro conjunto de datos, una parte considerable no se utilizará para el entrenamiento del modelo, de hecho, realizar esta partición del conjunto de datos puede condicionar los resultados de la evaluación de nuestro modelo, ya que la precisión para un conjunto de test puede variar mucho de la precisión para otro conjunto distinto. Una solución para este problema es el algoritmo de **K-Fold Cross Validation** [Rodríguez et al., 2010], que nos garantiza que cada plegado del conjunto de datos utilizará un conjunto de test distinto, de forma que, si realizamos la media de la precisión de cada pliegue, obtenemos una estimación más fiable de la eficacia del modelo. Podemos ver una representación gráfica de este proceso en la Figura 5.

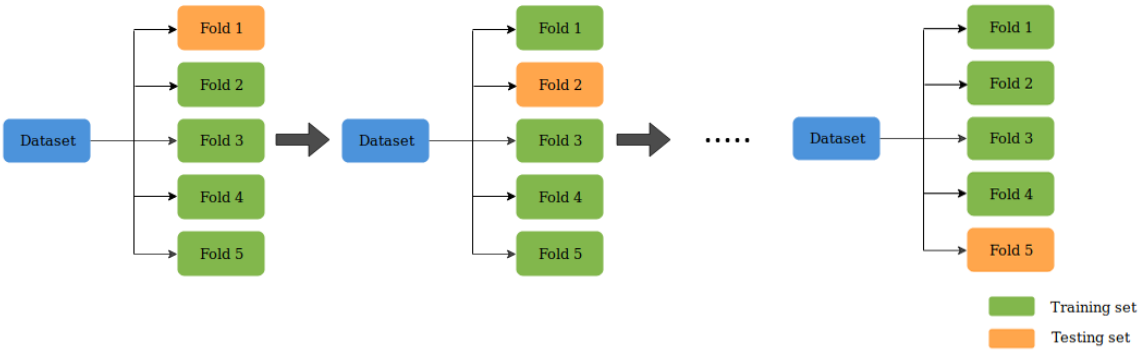


Figura 5: Particionado del conjunto de datos mediante 5-Fold Cross Validation.

Al final, lo más importante de los modelos de Aprendizaje Profundo, son la calidad y cantidad del conjunto de datos. Es por ello que han surgido diversas técnicas para lidiar

con este problema, siendo la más conocida la llamada **aumentación de datos** [Perez and Wang, 2017]. Si imaginamos el funcionamiento de una red convolucional, esta va a obtener características de las imágenes, aprendiendo poco a poco estos rasgos, con el fin de ser capaz de reconocerlos en nuevas imágenes y poder dar una clasificación fiable, por ello, surge la idea de "clonar" una pequeña parte del conjunto de datos, para tener más datos con los que entrenar al modelo. Para entender el porqué esta técnica es de utilidad, vamos a poner un ejemplo con imágenes de gatos y perros.

Tenemos un modelo que tiene que ser capaz de discernir entre imágenes de perros y gatos, una de las diferencias más determinantes es la cabeza, por ejemplo, el perro tiene hocico y la mayoría de razas tiene las orejas significativamente más grandes, algunos perros tienden a tener la lengua fuera, etc. Existen bastantes rasgos que determinan si el animal es un perro o un gato. Si disponemos de un conjunto de datos relativamente pequeño, la red puede no ser capaz de aprender a discernir entre todos estos rasgos y llegar a confundir algunas razas de perro con un gato, sobretodo si tenemos en cuenta ángulos de las fotos, colores del pelaje, etc. Lo que propone la aumentación de datos, es clonar una pequeña parte del conjunto de datos realizando modificaciones a los mismos. Podemos clonar fotos de los animales rotándolos, cambiando los colores de la imagen, añadiendo un pequeño desenfoque, realizar translaciones, etc. Si la red recibe varias imágenes de un mismo gato, pero estas no son físicamente iguales, porque este se encuentre en distintas partes de la foto, su pelaje sea de un tono diferente, se haya añadido ruido, etc. La red no memorizará un patrón, aprenderá y reforzará sus pesos para identificar más tipos de gatos. El problema aparece cuando las imágenes tienen muchos píxeles coincidentes, ya que la red comenzaría a asociar X posiciones o patrones no relevantes a las clases. Podemos apreciar distintos tipos de modificaciones en imágenes en la Figura 6.

A lo largo de los años, han ido apareciendo las distintas técnicas comentadas para incrementar el tamaño del conjunto de datos, y hay diversos estudios que demuestran mejoras de precisión en la clasificación al practicar la aumentación. Sin embargo, hay que mantenerse dentro de unos límites, ya que si el volumen clonado es elevado, la red inevitablemente memorizará estructuras y no será capaz de generalizar como se esperaba en un principio.

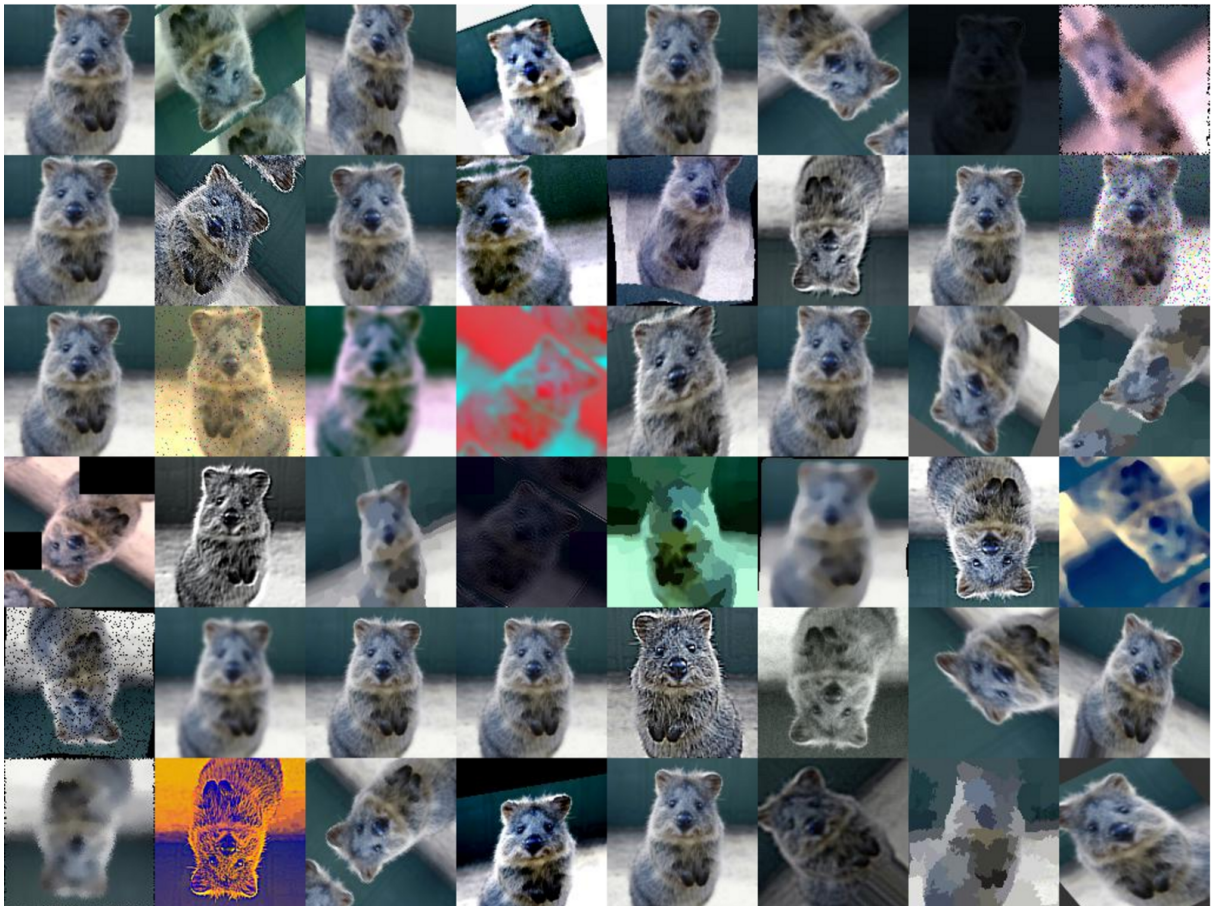


Figura 6: Aumentación de datos mediante la modificación de una misma imagen.

2.3. Estado del arte

La Observación de la Tierra está experimentando una explosión en cuestión al volumen de observaciones disponibles, lo que supone una capacidad sin precedentes hacia la monitorización a escala mundial de procesos naturales y artificiales [Nativi et al., 2015]. Sin embargo, este incremento del volumen, también supone un incremento en la variedad y complejidad de los datos, resultando en que las técnicas de análisis de datos supongan un cuello de botella en el proceso de extracción de conocimientos. Para abarcar este nuevo desafío, se comienzan a emplear técnicas de Machine Learning para automatizar el análisis y facilitar los estudios de Observación de la Tierra [Lary et al., 2016]. A diferencia de las técnicas convencionales de Machine Learning, que primero realizan la extracción de características y entonces aplican técnicas de clasificación superficiales, el Aprendizaje Profundo con sus Redes Neuronales profundas demuestra asombrosas capacidades, sobretodo en cuanto a la extracción automática de características más significativas, eli-

minando la necesidad de identificar características específicas según el problema que se afronta [Chen and Lin, 2014]. No será hasta el año 2014 que la comunidad comience a prestar atención a las Redes Neuronales cuando [Chen et al., 2014] comienzan a conseguir resultados significativos en problemas como clasificación Land Use and Land Cover y detección de objetos gracias a las imágenes multiespectrales.

Según la Cumbre Mundial sobre la Seguridad Alimentaria, en 2050 la población habrá crecido hasta los 10.000 millones, suponiendo una demanda enorme a la agricultura. Este aumento de producción de alimentos, requiere una organización sostenible de los terrenos de cultivo, a nivel de técnicas, y a nivel logístico y administrativo. Bajo este nuevo reto, existe una necesidad en el mundo de la agricultura de comenzar la monitorización de cultivos, para reducir impactos producidos por condiciones climatológicas, plagas, etc.

La monitorización de la agricultura es una aplicación de la Observación de la Tierra que se ha afrontado desde numerosos puntos de vista, predicción y monitorización de cosechas, gestión de riegos, etc. Haciendo uso de diferentes plataformas como satélites, drones, sensores, cámaras aéreo transportadas, etc. Especialmente desde 2013 [Weiss et al., 2020], el interés en aplicaciones de agricultura ha comenzado a crecer de manera exponencial, lo cual refleja el gran progreso en cuanto a tecnología se refiere, que ha emergido estos últimos años, incluyendo sensores con combinaciones novedosas como sensores espaciales, temporales y espectrales (i.e: Sentinel), o inclusive el desarrollo de nuevas plataformas como los UAV (Unmanned Aerial Vehicles).

Las imágenes aéreas y las de satélite son importantes recursos para la investigación en el mundo de la visión por computador, sin embargo, grandes conjuntos de estos datos, proporcionados de manera pública son escasos, como alternativa, [Xing et al., 2019] proponen el uso de plataformas como Google Earth para la obtención de imágenes RGB de alta resolución, suficiente para llevar a cabo un amplio abanico de estudios de interés, como es el caso de [Guirado et al., 2019], que realizan un estudio de las migraciones y el número de ballenas visibles en la superficie mediante imágenes aéreas obtenidas desde Google Earth junto a redes DCNN.

Estas alternativas para la obtención de imágenes de alta resolución, junto a la capacidad computacional actual y a los modelos de Aprendizaje Profundo, deberían de permitir a la humanidad cumplir con las expectativas puestas en las aplicaciones inteligentes.

2.4. Tecnologías empleadas

Este TFG ha sido desarrollado principalmente en el lenguaje de programación Python, el cual es un lenguaje de programación multiparadigma creado en 1991 por Guido van Rossum [Python Software Foundation, 2019]. Es un lenguaje que hace se particulariza por la legibilidad de su código al ser débilmente tipado. Gracias a su flexibilidad, simple sintaxis, rapidez en la iteracción de datos y sobretodo, su abundante cantidad de librerías, es el lenguaje de programación estrella del momento en el ámbito del Aprendizaje Computacional. En concreto, se ha trabajado con librerías como Keras, Tensorflow y OpenCV para la implementación de los modelos y Flask junto a Pymongo y Boto3 para el desarrollo del Backend[4] de la aplicación web, introducimos a estas herramientas a continuación.

Keras y Tensorflow

TensorFlow [Abadi et al., 2015] es un framework de Aprendizaje Computacional de código abierto desarrollado por Google. Es una enorme plataforma que provee a los desarrolladores de todas las herramientas necesarias para crear y desarrollar modelos de Aprendizaje Profundo. Mediante Python podemos acceder a su API para crear nuestras aplicaciones de Aprendizaje Computacional, aunque realmente los cálculos matemáticos se realizan internamente en C++.

A más alto nivel, aparece Keras [Chollet et al., 2015], una API escrita en Python que nos permite trabajar sobre frameworks como CNTK, Theano y TensorFlow. Su objetivo es facilitar la creación y el entrenamiento de los modelos, proporcionando prototipos de redes neuronales ya entrenados, llamar algoritmos de optimización sin tener que implementarlos el usuario, gestionando el entrenamiento del modelo proporcionando estadísticas, etc.

Es importante remarcar que TensorFlow muestra compatibilidad tanto con CPU como con GPU, permitiéndonos aprovechar al máximo nuestro hardware.

OpenCV

OpenCV [Bradski, 2000] es una librería de Visión por Computador y Aprendizaje Computacional de código abierto. Actualmente cuenta con más de 2500 algoritmos de

[4]Backend: parte del desarrollo que se encarga de que la lógica de la web funcione

optimización, incluyendo tanto algoritmos clásicos como del estado del arte. Utilizamos estos algoritmos para procesar las imágenes.

Flask

Flask [PalletsProjects] es un microframework de python para crear aplicaciones web de manera sencilla, a diferencia de otros frameworks como Django, Flask se mantiene simple y minimalista, ya que únicamente tienes que incluir las extensiones con las que realmente vas a trabajar, a diferencia de tener por defecto al crear un proyecto diversas funcionalidades que pueden o no ser de utilidad.

Flask apuesta por la sencillez sin sacrificar el potencial de uso, ya que permite crear aplicaciones robustas y escalables, soportando conectividad con bases de datos, ORMs, integración de migraciones, etc. Es por esto, que se ha escogido como framework para realizar el Backend de la aplicación que presenta este Trabajo de Fin de Grado.

2.4.1. HTML, JavaScript, CSS

A la hora de realizar el Frontend de la aplicación, no se ha utilizado ningún framework, sino que se ha trabajado directamente con las 3 tecnologías principales de desarrollo web. HTML para construir y definir la estructura de la aplicación web, JavaScript para manejar las funcionalidades de la aplicación y CSS para estilizar los contenidos.

2.4.2. MongoDB

MongoDB[5] es una base de datos NoSQL de código abierto orientada a documentos tipo JSON. A diferencia de las bases de datos relacionales, MongoDB almacena estructuras de datos en ficheros BSON[6] de manera dinámica, haciendo que la integración de datos en ciertas aplicaciones sea más fácil y rápida.

Como características destacables, soporta búsqueda por campos, indexación, replicación, balanceo de carga, etc. Además cuenta con un potente framework de agregación que permite realizar consultas avanzadas de forma eficiente.

En este Trabajo de Fin de Grado se ha utilizado MongoDB para almacenar los metadatos de las imágenes que utilizaremos en la aplicación (URL, clasificación, etc).

[5] `https://www.mongodb.com/es`
[6] BSON: Binary JSON

2.4.3. Amazon S3

Amazon Web Services es uno de los gigantes en cuanto a proveedores de servicios web se refiere, y dentro de su repertorio se incluye el denominado S3[7] (Simple Storage Service), el cual es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, rendimiento y seguridad para cualquier tipo de usuarios.

La unidad de almacenamiento en S3 es denominada Bucket, y ahí será donde almacenemos las imágenes de la aplicación que se han comentado anteriormente.

[7] `https://aws.amazon.com/es/s3/`

3. Propuesta y Software

Este TFG propone un ecosistema de aprendizaje profundo, el cual se ha nombrado **"Manvocado"** (de las palabras inglesas Mango y Avocado). Su estructura se puede observar en el diagrama de flujo de la Figura 7, el cual está compuesto por un conjunto de imágenes de cultivos tropicales realizados a mano, un modelo "Generative Adversarial Network" que genere imágenes artificiales para aumentar el conjunto de datos inicial, un modelo estándar de Red Convolucional para llevar a cabo la clasificación de las imágenes, y una aplicación web que, además de integrar el modelo para realizar predicciones de manera interactiva, permita a la comunidad agrónoma etiquetar y aportar imágenes para alimentar el dataset.

Se ha seguido una metodología iterativa incremental [Sommerville] para la realización de este proyecto, comenzando con la recolección de imágenes de forma manual en Google Earth y con la implementación provisional de ambos modelos de aprendizaje profundo. Entonces, se ha desarrollado la aplicación web de manera incremental, en base a las funcionalidades que se han ido necesitando añadir. Finalmente, se ha concluido con la selección de los hiperparámetros de ambos modelos para obtener mejores resultados.

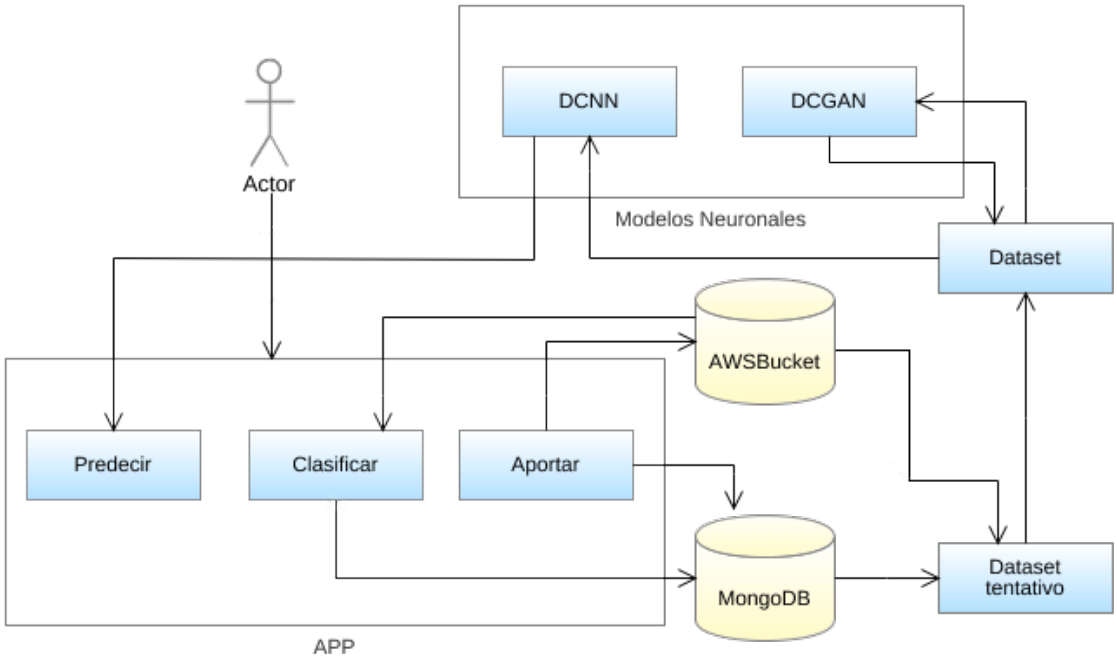


Figura 7: Diagrama de flujo de la aplicación

El diagrama de flujo de la Figura 7 describe el ecosistema propuesto en su totalidad. El usuario experto o actor, tiene contacto con la aplicación, la cual tiene 3 funcionalidades, predecir, clasificar y aportar. Donde la funcionalidad de *predicción* emplea la imagen obtenida por parte del usuario y el modelo DCNN para realizar la predicción. *Clasificar* muestra imágenes que recibe del servidor de AWS, sobre las cuales el usuario realiza el etiquetado, enviando esta información a la base de datos en MongoDB, donde se actualizan los campos relativos a cada imagen. Y por último la funcionalidad de *aportar*, que recibe una imagen por parte del usuario, la cual se guarda en el servidor de AWS y sus metadatos se almacenan en la base de datos MongoDB. Por otro lado y ajeno al usuario, tenemos el conjunto de datos tentativo formado por las imágenes de AWS y sus metadatos de MongoDB, los cuales, bajo un proceso de votos que explicaremos más adelante, son validados y pasan a formar parte del conjunto de datos principal. Finalmente, este conjunto de datos etiquetado y validado, es utilizado por el modelo DCGAN, el cual genera imágenes artificiales con el objetivo de aumentar el conjunto de datos existente.

A continuación pasamos a describir cada uno de los componentes del diagrama de flujo de manera más detallada.

3.1. Deep Convolutional Generative Adversarial Networks: DCGAN

En la teoría de juego clásica, un juego de suma cero representa una situación en la cual la "pérdida/ganancia" de cada jugador está exactamente balanceada con respecto a la de sus rivales, de tal forma, que si sumamos todas las pérdidas y ganancias, estas sumarán cero. Este tipo de juegos se suelen resolver mediante algoritmos minimax[8], los cuales actúan asumiendo que el rival siempre elige la mejor decisión posible en cada turno.

En este contexto, introducimos las Redes Generativas Antagónicas (GAN), modelo formado por dos redes neuronales convolucionales que se "enfrentan" en un escenario similar a un juego minimax de 2 jugadores. Aunque no siempre tenga que ser de la forma de suma-cero, el objetivo de las GAN es alcanzar, en términos de la teoría de juego, el denominado "Nash Equilibrium" [Halpern, 2007].

El modelo está compuesto por una **red generativa**, que genera candidatos de manera

[8]Algoritmos MiniMax: `https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-next-move-8f028bd0e7b1`

artificial, y una **red discriminativa** que se encarga de evaluar el candidato y discernir si es sintético u original. Esta "competición" se realiza sobre distribuciones de datos. La red generativa realiza una asignación a partir de un espacio latente[9] hasta obtener una distribucion de datos de interés. Será entre esta distribución y la distribución de los datos originales, que la red discriminativa llevará a cabo su evaluación. El objetivo de la red generativa es aumentar el ratio de error de la red discriminativa, de manera más coloquial, engañar al discriminante haciéndole creer que los candidatos no son sintéticos.

En nuestro problema, empleamos este modelo con el fin de aumentar el conjunto de datos inicial, además de realizar de aumentación mediante los métodos convencionales como los explicados en la figura 6.

Sin embargo, el entrenamiento de este modelo es una tarea realmente compleja como veremos a continuación.

3.1.1. Inconvenientes del modelo

Fue en el año 2014 cuando [Ian J. Goodfellow] publicó sus resultados con este tipo de modelos, que hasta entonces, no habían terminado de funcionar. Desde entonces, muchas variantes que proponen soluciones a diversos problemas han surgido. Para este trabajo se ha utilizado el modelo WGAN-GP (Wasserstein Generative Adversarial Network with Gradient Penalty) [Gulrajani et al., 2017] debido a su gran mejora en el entrenamiento del modelo.

El problema principal de este tipo de modelos es el entrenamiento, ya que, como se ha comentado previamente, se busca un equilibrio entre ambas redes, deben de ir aprendiendo a la par, lo cual es realmente complejo y según el tipo de problema que se abarque, surgen ciertos problemas para los que a día de hoy todavía no se tiene solución.

Algunos de esos problemas son los siguientes:

- **Desvanecimiento del gradiente [Pascanu et al., 2012]:** A medida que el discriminador aprende, el gradiente del generador se va haciendo cada vez más pequeño, hasta que deja de tener ningún tipo de efecto en la actualización de pesos del generador.

- **"Mode Collapse":** Es un problema frecuente en el entrenamiento de la red genera-

[9]Para más información: `https://ai-odyssey.com/2017/02/24/latent-space-visualization/`

tiva, donde esta colapsa[10], generando siempre la misma imagen independientemente del espacio latente que se le proporcione.

- **Complejidad matemática:** Esto es principalmente un problema debido a su reciente descubrimiento, por los principios en los que se basa, se requiere un alto nivel de conocimiento en estadística y en optimización, sin embargo, también existen diversas técnicas que, aunque implicitamente tienen un modelo matemático, este no se obtiene analíticamente, sino por ensayo, error y mediante la obtención de mejores resultados en la convergencia del modelo.

3.1.2. Arquitectura

La arquitectura seleccionada es una adaptación de la propuesta por [Arjovsky et al., 2017], que consiste en las siguientes 2 redes:

[10]Mode Collapse: `https://developers.google.com/machine-learning/gan/problems#mode-collapse`

- Red Generativa: red convolucional que recibe como entrada ruido de una distribución de probabilidad (generalmente una distribución uniforme) y produce de salida datos que buscan aproximarse a la distribución de los datos reales.

Layer (type)	Output Shape	Parameters
dense_1 (Dense)	(None, 196608)	19857408
reshape_1 (Reshape)	(None, 64, 64, 48)	0
up_sampling2d_1	(None, 128, 128, 48)	0
conv2d_1 (Conv2D)	(None, 128, 128, 128)	98432
batch_normalization_1	(None, 128, 128, 128)	512
activation_1	(None, 128, 128, 128)	0
up_sampling2d_2	(None, 256, 256, 128)	0
conv2d_2 (Conv2D)	(None, 256, 256, 64)	131136
batch_normalization_2	(None, 256, 256, 64)	256
activation_2	(None, 256, 256, 64)	0
conv2d_3 (Conv2D)	(None, 256, 256, 3)	3075
activation_3	(None, 256, 256, 3)	0

Tabla 1: Tipos, tamaños y número de parámetros de las capas de la red generativa

- Red Discriminativa: red convolucional sencilla que se limita a clasificar su entrada como generada o real.

Layer (type)	Output Shape	Parameters
conv_s_n2d_1	(None, 128, 128, 16)	464
leaky_re_lu_1	(None, 128, 128, 16)	0
dropout_1	(None, 128, 128, 16)	0
conv_s_n2d_2	(None, 64, 64, 32)	4672
zero_padding2d_1	(None, 65, 65, 32)	0
batch_normalization_3	(None, 65, 65, 32)	128
leaky_re_lu_2	(None, 65, 65, 32)	0
dropout_2	(None, 65, 65, 32)	0
conv_s_n2d_3	(None, 33, 33, 64)	18560
batch_normalization_4	(None, 33, 33, 64)	256
leaky_re_lu_3	(None, 33, 33, 64)	0
dropout_3	(None, 33, 33, 64)	0
conv_s_n2d_4	(None, 33, 33, 128)	73984
batch_normalization_5	(None, 33, 33, 128)	512
leaky_re_lu_4	(None, 33, 33, 128)	0
dropout_4	(None, 33, 33, 128)	0
flatten_1	(None, 139392)	0
dense_sn_1	(None, 1)	139394

Tabla 2: Tipos, tamaños y número de parámetros de las capas de la red discriminativa

En la Tabla 1 podemos observar como, partiendo del espacio latente y mediante una entrada completamente conectada (*dense_1)*, vamos modificando la forma de las salidas adaptándolas a las dimensiones que requerimos en las imágenes (256x256x3). Resultando en el siguiente número total de parámetros del modelo:

- Total params: 237,970
- Trainable params: 237,281
- Non-trainable params: 689

Mientras que, en el caso del discriminante, se muestra en la Tabla 2 que partimos realizando convoluciones a la imagen de entrada, aplicando los filtros para calcular los mapas de características, que se conectan al final mediante las capas *flatten_1 y dense_SN*, resultando en una clasificación final (comprobamos que la dimensión de la salida es 1) que dicta si la imagen de entrada es real o artificial.

- Total params: 20,090,819
- Trainable params: 20,090,435
- Non-trainable params: 384

El número total de parámetros de la red es mucho menor que el del generador, ya que la capa completamente conectada, Dense, se aplica al final para realizar la clasificación, a diferencia de la red generadora, que la incorpora al principio para procesar el espacio latente.

3.2. Dataset y clasificador mediante Deep Convolutional Neural Network: DCNN

Gracias a la recolección de imágenes en la plataforma de Google Earth, al uso del modelo DCGAN y a las técnicas de aumentación estándar, se ha creado un dataset de 3 clases de entorno a 600 imágenes, en la Figura 8 podemos ver un ejemplo de una imagen por cada clase.

Figura 8: Ejemplo de las distintas clases de imágenes del dataset: Aguacate, Mango y Construcción

Una vez encontrado un conjunto de datos adecuado, comenzamos con la selección de un modelo predictivo que se adecúe a los datos. Cuando se trabaja con un conjunto de datos reducido, como es nuestro caso, la manera óptima de abarcar el problema es recurrir a una técnica denominada **transfer learning** o "transferencia de conocimiento"[11]. Esta técnica aprovecha los modelos entrenados principalmente en competiciones de Kaggle[12], los cuales son modelos con arquitecturas complejas que han sido entrenados con enormes conjuntos de datos, no únicamente en número de imágenes sino también número de clases. Es decir, se aprovechan modelos que ya son capaces de discernir entre un gran número de clases, re-entrenándolos con nuestro conjunto de datos específico.

Cuando la capacidad de un modelo es grande, puede aprender estructuras más complejas, pero a la vez, es más susceptible al **Overfitting**, ya que puede tender a memorizar estructuras en vez de generalizarlas. Por ende, lo más apropiado en nuestro caso es reutilizar algún modelo ya entrenado y no complicar mucho más su arquitectura para no introducir más parámetros que dificulten la generalización.

Cada neurona de una capa de convolución tiene un campo receptivo del tamaño de la entrada que recibe de la capa anterior. De manera intuitiva, cada kernel captura una cierta relación entre entradas cercanas de forma que cada capa se enfoca en cierto tipo de característica. Estos kernels, como se comentó en el capítulo 2, suelen ser pequeños, por lo que al analizar la entrada, son capaces de proveer información localizada de manera más precisa al analizar en mayor detalle. Cuanto más profunda sea la red, el campo receptivo de cada neurona con respecto a la capa anterior se hace más grande, ya que cada vez hay más información. Capas profundas proporcionan características con semántica global y detalles abstractos, que provienen de relaciones de relaciones entre objetos, los cuales somos capaces de generalizar mediante el uso de estos kernels pequeños.

Algunos modelos pre-entrenados populares son VGGNet, Inception, ResNet, etc. Basándonos en lo anterior, una red es capaz de generalizar en mayor detalle si es más profunda, siendo esto más importante que el número de parámetros per se, y esto se puede apreciar parcialmente si comparamos el modelo VGGNet2014 (16 capas con 140M de parámetros) con ResNet2015 (152 capas con 2M de parámetros), habiendo obtenido ResNet mejores resultados.

[11]Transfer learning: `https://machinelearningmastery.com/transfer-learning-for-deep-learning/`

[12]Kaggle: `https://www.kaggle.com/`

Para este Trabajo de Fin de Grado se ha escogido el modelo **ResNet50** como base para realizar transfer learning. A este modelo se le han añadido varias capas para adecuarlo a nuestro conjunto de datos y obtener la salida con el formato correcto. A continuación podemos observar la arquitectura de la red en la Tabla 3.

Layer (type)	Output Shape	Parameters
resnet50	(None, 8, 8, 2048)	23587712
flatten_1	(None, 131072)	0
dropout_1	(None, 131072)	0
dense_1	(None, 3)	393219

Tabla 3: Tipos, tamaños y número de parámetros de las capas de la red predictiva

3.3. Aplicación web

Para mostrar la utilidad de los modelos desarrollados, se ha optado por el despliegue de una aplicación web en la nube. Idealmente, esta aplicación está enfocada a la comunidad agrónoma, debido a la temática del proyecto.

En su desarrollo se ha utilizado el framework de Python, Flask para el backend; HTML, CSS y JS para el frontend y Pytest para la realización y ejecución de casos de prueba.

3.3.1. Requisitos funcionales

La aplicación debe ser capaz de admitir imágenes subidas por los usuarios, manteniendo además un control sobre las mismas antes de realizar cualquier tipo de almacenamiento o procesamiento.

Dichas imágenes, se podrán utilizar para que el modelo entrenado realice una predicción, clasificando la misma en alguna de las clases existentes. Además, se permitirá a los usuarios adjuntar 1 o más imágenes de una clase con el fin de alimentar el dataset, mediante la opción de clasificar manualmente la imagen subida.

Por último, se le permitirá a los usuarios seleccionar entre 1 y 4 imágenes sin clasificar, con el objetivo de que escoja aquellas que se correspondan con la clase especificada.

3.3.2. Implementación del Backend

Las imágenes subidas por los usuarios se alojan a través de una API REST en un S3 Bucket de AWS, y la URL es almacenada en una base de datos no relacional MongoDB. Esta base de datos se encuentra alojada en un cluster Atlas[13], y sus documentos almacenan el enlace, la clase y los votos que ha recibido cada clase.

Adicionalmente, se proporciona un pequeño "juego" que permite a los usuarios clasificar imágenes que serán proveídas por la aplicación. Dichas imágenes son aquellas que los usuarios suben a la aplicación, que se encuentran pendientes de validar, no teniendo todavía una clase asociada. De esta forma, se planea, mediante un criterio estadístico para asegurar fiabilidad de las clasificaciones, ir validando de forma externa la clasificación de imágenes para el crecimiento del dataset.

Aportación de imágenes

Crear un dataset desde 0 es una tarea complicada, es por eso que las empresas y grupos de investigación se apoyan en reutilizar datos de otros proyectos o en grandes colectivos, donde cada persona aporte un poco, obteniendo entre todos un conjunto grande de datos. Por ejemplo tenemos el caso de uno de los conjuntos de datos más populares en el campo de Visión por Computador, el dataset MNIST [Yann LeCun], el cual consiste en números escritos a mano por estudiantes estadounidenses.

Con esta mentalidad, se ha añadido la opción de permitir a los usuarios subir imágenes de cualquiera de las clases del dataset, las cuales se almacenan en un S3 Bucket de AWS y sus metadatos (ver Figura 9) en un cluster atlas de MongoDB. Inicialmente, como medida de seguridad, las imágenes carecen de una clase y se añaden al conjunto de imágenes que están por validar. Cuando el usuario sube las imágenes marcadas tras cierta clase, dicha clase recibe un voto, pero no será clasificada hasta que se alcance cierto grado de confianza, esto se comentará en detalle en el siguiente apartado.

[13]MongoDB Atlas: `https://www.mongodb.com/cloud/atlas`

```
_id: ObjectId("5e81e3ab84421e9d2329a38c")
class: "avocado"
URL: "https://manvocado.s3.eu-west-3.amazonaws.com/image0.png"
votesAvocado: 9
votesMango: 3
votesBuilding: 0
```


Figura 9: Ejemplo de un documento de MongoDB

Para trabajar con S3 y MongoDB se han definido 2 clases gracias a las librerías que proporcionan ambas empresas: *boto3*[14] y *pymongo*[15], cada una contiene los métodos necesarios para realizar las funciones requeridas: guardar y eliminar archivos del Bucket; guardar, eliminar, actualizar y obtener documentos del cluster Atlas, etc.

La aplicación permite subir imágenes de una misma clase en masa ya que las herramientas con las que trabajamos proporcionan soporte para aplicaciones multi-threading. Además, se realiza una comprobación del formato de las imágenes seleccionadas antes de subirlas con JavaScript para evitar la propagación de errores en otras fases del ecosistema.

Etiquetado manual de imágenes de satélite

Como se ha comentado con anterioridad, con fin de alimentar el dataset se ha implementado a modo de juego, una sección de la aplicación donde el usuario recibe varias imágenes correspondientes a recortes de plataformas como Google Earth, teniendo que seleccionar las que se correspondan con la clase que se le especifica. De esta forma, se espera que el usuario etiquete imágenes sin clasificar, esta información se almacena en el cluster de MongoDB, conteniendo así en todo momento, qué imágenes han sido clasificadas con cada clase, así como el número de usuarios que han etiquetado. Será cuando una imagen cumpla un criterio estadístico, que esta pasará a formar parte del dataset, así evitamos introducir ruido a nuestro conjunto de datos.

Este criterio, consiste en evaluar los porcentajes de votos de todas las clases cuando la imagen haya sido clasificada 10 veces, a partir de entonces, sólo se validará como parte del conjunto de datos si alguna de las clases recibe más del 75 % de los votos.

[14] `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`
[15] `https://pymongo.readthedocs.io/en/stable/`

Cada voto que envía el usuario realiza una operación de actualización (update) en el documento correspondiente a la imagen, incrementando en 1 el número de votos de la clase especificada. En el momento que el número de votos de todas las clases sea igual o superior a 10, el Trigger de Stitch[16] que se muestra en Fragmento de código 1 se encarga de aplicar el criterio anterior, de esta forma, cuando este se cumpla, se cambiará la clase de la imagen, eliminándola del conjunto de imágenes en validación, pasando así a formar parte del conjunto de datos.

```
exports = function(changeEvent) {

  const fullDocument = changeEvent.fullDocument;
  const totalVotes = fullDocument.votesAvocado + fullDocument.votesMango +
                     fullDocument.votesBuilding;

  if (fullDocument.class == "pending"){
    const mongodb = context.services.get("Cluster0");
    const manvo = mongodb.db("Images").collection("Manvocado");
    if (totalVotes >= 10) {
      if (fullDocument.votesAvocado/totalVotes >= 0.75) {
        return manvo.updateOne(
          { "_id": fullDocument._id  },
          { "$set": { "class" : "avocado" } }
        );
      } else if (fullDocument.votesMango/totalVotes >= 0.75) {
        return manvo.updateOne(
          { "_id": fullDocument._id },
          { "$set": { "class" : "mango" } }
        );
      } else if (fullDocument.votesBuilding/totalVotes >= 0.75) {
        return manvo.updateOne(
          { "_id": fullDocument._id  },
          { "$set": { "class" : "building" } }
        );
      }
    }
  }
};
```

Fragmento de código 1: Trigger ejecutado cuando se actualiza un documento de la colección *Manvocado*.

[16]MongoDB Stitch: `https://www.mongodb.com/cloud/stitch`

Análisis predictivo de imágenes

El objetivo principal de la aplicación es traer los beneficios que aportan los modelos de Aprendizaje Profundo a los usuarios sin la necesidad de que tengan conocimientos sobre el tema. Esta aplicación integra el modelo DCNN comentado anteriormente para que los usuarios puedan realizar predicciones sobre las imágenes que deseen clasificar. Debido a la naturaleza de estos modelos, la única restricción son las dimensiones de la imagen, que deben de coincidir con las que la red es capaz de tratar, esto es, deben de subirse imágenes de 256x256 píxeles. En este sentido, se aportan también herramientas externas como *Pinetools*[17] para facilitar la captura masiva de imágenes con estas dimensiones.

Al igual que cuando se suben imágenes a la aplicación, está controlado qué tipo de imágenes se suben a nivel de JavaScript, siempre evitando que lleguen al Backend imágenes que no cumplan con las especificaciones necesarias para que sean aceptadas por los modelos. Para realizar las predicciones, se convierte la imagen a un array numpy, se carga el modelo predictivo con los pesos sinápticos y se clasifica la imagen. Como sabemos, las 3 clases que contemplamos son: Aguacate, Mango y Construcción, al realizar la predicción, la aplicación devolverá la clase con mayor probabilidad como el resultado, mostrando el porcentaje de certeza del modelo. En caso de que no se estime una clase con una probabilidad superior a 0.6, la aplicación denotará que dicha imagen no se ha podido clasificar.

3.3.3. Entrega Continua

Para el correcto desarrollo de la aplicación, se han implementado tests para validar la corrección de las funcionalidades descritas en el apartado anterior. Para ello, se ha utilizado la librería *Pytest*.

En el despliegue de la aplicación se ha utilizado la plataforma *Heroku* [Orion Henry and Wiggins, 2007], que nos brinda un sencillo servicio para desplegar aplicaciones de repositorios en un contenedor ligero. Heroku define su propio tipo de contenedor, denominado Dyno[18], que permite a su plataforma aislar y virtualizar contenedores Linux que están diseñados para ejecutar el código personalizado de los usuarios, alojándolos en su web de manera gratuita.

Adicionalmente, como se ha seguido una metodología incremental, se ha realizado

[17] `https://pinetools.com/es/partir-imagenes`

[18] Dynos en Heroku: `https://www.heroku.com/dynos`

Entrega Continua (Continuous Deployment) [Humble et al., 2006] durante el desarrollo de la aplicación con el fin de automatizar los casos de test que se han implementado con el fin de mantener un desarrollo controlado entre las diversas versiones por las que ha pasado el proyecto durante la implementación de todas las funcionalidades, así como la automatización de su despliegue en Heroku. Para esto, se ha hecho uso de las herramientas que provee GitLab[19] en sus repositorios, permitiéndonos automatizar la aplicación de tests tras cada push al repositorio. Se ha utilizado para guardar las credenciales e IDs, un archivo JSON a modo de diccionario, el cual se encuentra público en el repositorio y muestra valores genéricos, pero que gracias a GitLab y a sus variables de entorno privadas que se aprecian en la Figura 10, conseguimos configurar con las claves correctas para su conexión con MongoDB y AWS.

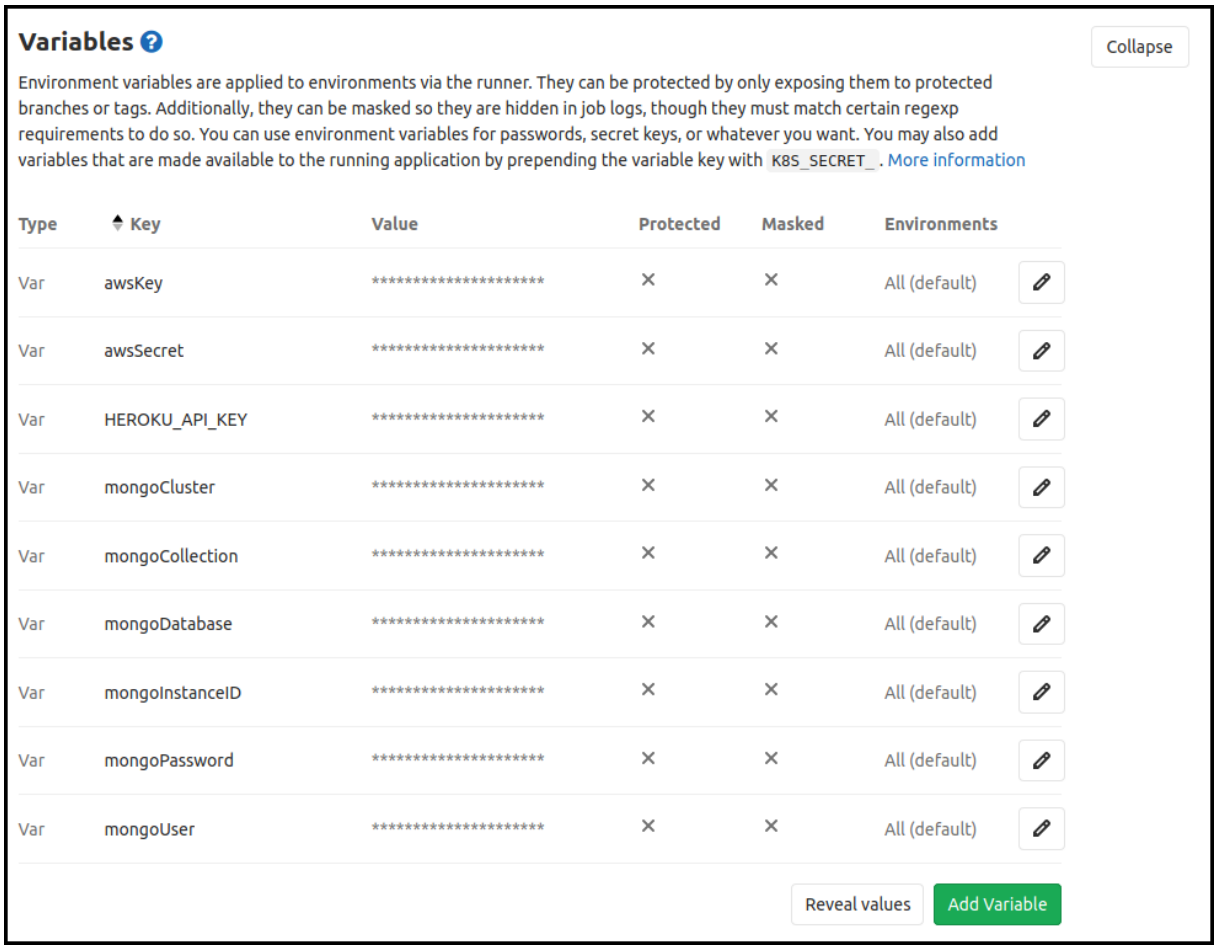


Figura 10: Variables de entorno privadas del repositorio GitLab

Finalmente, en adición a los testeos, gracias a la integración continua (ver Fragmento

[19]GitLab: `https://gitlab.com/`

de código 2) se ha automatizado el despliegue de la aplicación en Heroku, haciendo que, si tras realizar una operación push al repositorio al repositorio, se pasan correctamente los tests, se actualice la versión de la aplicación que está publicada en Heroku, permitiéndonos mantener siempre la versión más actualizada y funcional de nuestro proyecto.

```
image: python:3.7.7-stretch


PIP_CACHE_DIR: "$CI_PROJECT_DIR/.cache/pip"

cache:
  paths:
    - .cache/pip
    - venv/

before_script:
  - python -V  # Print out python version for debugging
  - sed -i -e "s/user/$mongoUser/g"
            -e "s/password/$mongoPassword/g"
            -e "s/cluster/$mongoCluster/g" -e "s/instanceID/$mongoInstanceID/g"
            -e "s/changemongoDBDatabase/$mongoDatabase/g"
            -e "s/changemongoDBCollection/$mongoCollection/g"
            -e "s/changeKey/$awsKey/g"
            -e "s|changeSecret|$awsSecret|g" config.json

  - pip install virtualenv -q
  - virtualenv venv
  - source venv/bin/activate

test:
  script:
    - pip install -r requirements.txt -q
    - pytest

#Push to Heroku to update the web application
    - git add .
    - git config --global user.email "git@gitlab.com"
    - git config --global user.name "Git Lab"
    - git commit -m "Updating web application"
    - git remote add heroku https://heroku:$HEROKU_API_KEY@git.heroku.com/manvocado.git
    - git push -q -f heroku HEAD:master

    - rm config.json
```

Fragmento de código 2: Configuración del fichero gitlab-ci.yml que configura testing y el despliegue en Heroku.

4. Experimentos

Con el fin de maximizar la precisión de los modelos descritos en el apartado anterior, se han llevado a cabo numerosos experimentos de ajuste de hiperparámetros para obtener la configuración más óptima para ambos modelos y se ha probado diferentes infraestructuras para la ejecución de los programas.

Entrenar modelos de Aprendizaje Profundo requiere de una gran capacidad de cómputo. Inicialmente, se empleó una máquina de desarrollo del edificio de investigación Ada Byron[20] debido a su potencia (32 cores, 64 GB RAM) y disponibilidad exclusiva. Sin embargo, al no disponer de unidades de procesamiento gráfico, el rendimiento condicionaba el ritmo de pruebas, resultando en el orden de decenas de horas el entrenamiento de la red GAN. Ante este problema, se decidió realizar los testeos en el cluster de super computación Picasso, localizado en el Parque Tecnológico de Andalucía[21], Málaga.

Picasso es uno de los 7 Super Computadores que se hallan en España, se encuentra en el Parque Tecnológico Andaluz (PTA), y nos ofrece un servicio de alto rendimiento para la ejecución de programas de cálculo intensivos. Sin embargo, debido a problemas de compatibilidades con versiones software y sus unidades GPU, no se pudo dar uso de estas, teniendo que realizar el cómputo sobre CPU.

Para ejecutar los scripts, se ha tenido que realizar un estudio de la herramienta de gestión de carga de trabajo que utiliza Picasso, llamada SLURM[22].
Slurm Worload Manager es una herramienta de código abierto que proporciona un servicio escalable y a prueba de fallos de clustering y planificación de tareas.

Picasso cuenta con esta herramienta, permitiendo al usuario acceder a un entorno virtual Linux reducido mediante SSH (login node) donde almacenar programas y datos para su posterior ejecución en los nodos de ejecución. Se realizan scripts de bash donde el usuario selecciona los recursos que su programa necesitará durante la ejecución, se especificarán parámetros y comandos a ejecutar, archivos de entrada, etc. Será este script el que se envíe al nodo de ejecución, enviándolo a una cola de procesos donde el propio servidor gestiona de manera automática el uso de sus recursos e irá despachando las peticiones de ejecución poco a poco.

[20]Edificio de investigación Ada Byron: `https://www.uma.es/adabyron/`
[21]PTA: `https://www.parquetecnologicodeandalucia.com/`
[22]SLURM Workload Manager: `https://slurm.schedmd.com/`

Desgraciadamente, debido a la no disponibilidad de GPUs y a la gran carga a la que se somete Picasso por parte de diferentes empresas y grupos de investigación, se optó por abandonar el uso de su infraestructura y operar con mi ordenador personal en GPU (CPU: Ryzen 5 2600G, GPU: NVIDIA GeForce 1660Ti, Memoria: 16GB).

A continuación se comentan los experimentos llevados a cabo con los modelos neuronales.

4.1. Técnicas e hiperparámetros para la DCGAN

Debido a su reciente popularidad, las redes GAN constan de escasa literatura, y las técnicas para encauzar su complejo entrenamiento son bastante conocidas, pero limitadas. Como se comentó en el capítulo anterior, nos hemos basado en el modelo WGAN-GP, que incluye la distancia de Wasserstein y un Gradient Penalty para el discriminante.

Al fin y al cabo, las redes GAN se identifican más bien con el aprendizaje no supervisado, ya que se trata de aprender la distribución de probabilidad de las imágenes que se pretenden generar artificialmente. Este aprendizaje se basa en definir familias de funciones de densidad de probabilidad y encontrar la que maximice el "likelihood" en nuestros datos. Se trata de un problema de optimización.

Ya que buscamos maximizar el likelihood, la distancia entre las distribuciones de probabilidad, la de nuestros datos y la de nuestro modelo, debe ser mínima. Basándose en ello, [Arjovsky et al., 2017] propone el análisis de la distancia de Wassertstein o distancia EM (Earth Mover's Distance[23]). La cual se puede describir de una manera bastante intuitiva mediante el siguiente escenario: disponemos de 2 distribuciones de probabilidad que representan 2 formas de amontonar tierra, entonces, la distancia EM es el coste mínimo de convertir una en la otra, siendo el coste la cantidad de tierra que se ha movido por la distancia que se desplaza.

Como resultado de emplear esta métrica, no solo se reduce la aparición de bastantes problemas típicos en el entrenamiento de las redes GAN, sino que también obtenemos una métrica útil (loss function) para correlar la convergencia del generador y la calidad de las imágenes generadas.

Sin embargo, pese a obtener muy buenos resultados y mejorar la estabilidad del entrenamiento de manera considerable, siguen apreciándose comportamientos no deseados,

[23]Earth Mover's Distance: `https://en.wikipedia.org/wiki/Earth_mover's_distance`

especialmente en el discriminante, por ello, [Gulrajani et al., 2017] propone un método distinto para forzar las restricciones requeridas sobre el discriminante, denominado Gradient Penalty.

Utilizando las arquitecturas comentadas en el capítulo anterior y empleando las técnicas descritas obtenemos los siguientes resultados generando imágenes artificiales de árboles de aguacate (ver Figura 11).

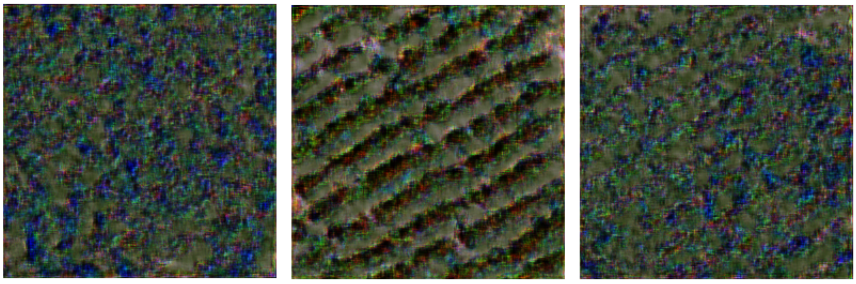


Figura 11: Árboles de aguacate desde vista aérea generados de manera artificial

Como podemos observar en la Figura 12, el error del generador se dispara, el discriminante es demasiado potente y el balance en el entrenamiento se rompe.

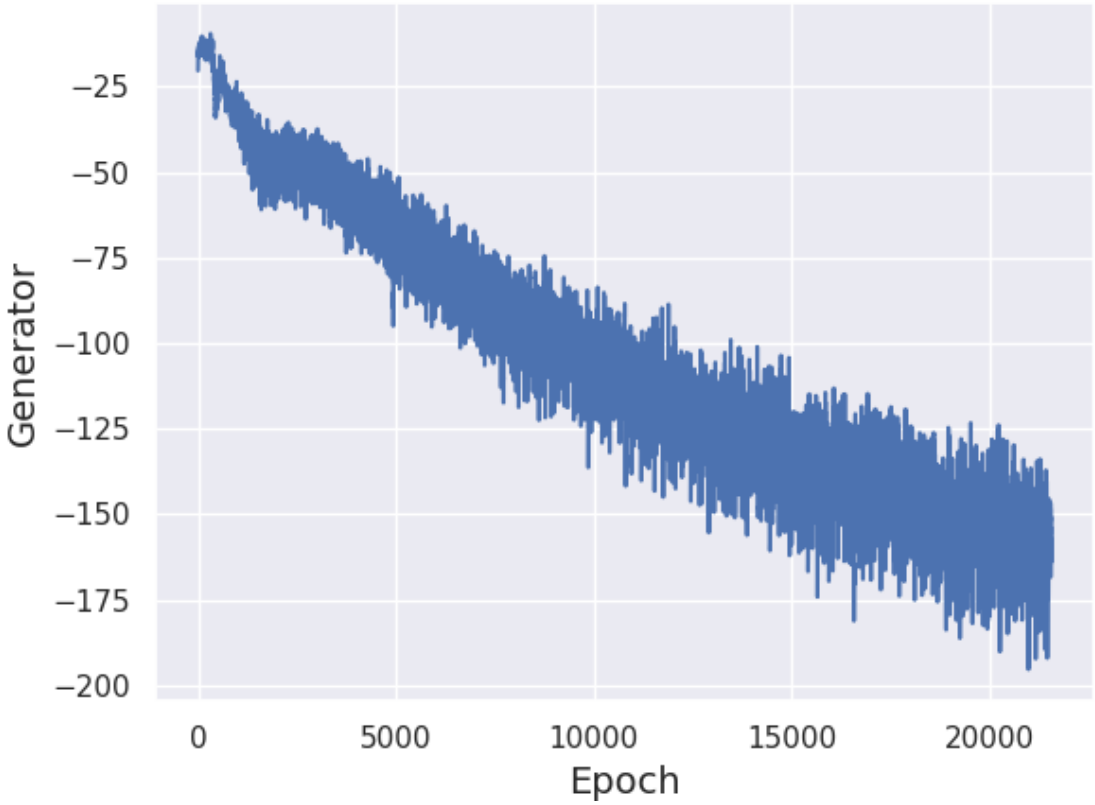


Figura 12: Loss function del generador por cada época

Normalización Espectral (Spectral Normalization)

Se ha podido observar que este campo pese a ser realmente prometedor, es considerablemente reciente y requiere tiempo para establecer criterios sólidos que lleven a la convergencia del entrenamiento de manera maś sencilla. Una de las muchas investigaciones y propuestas que la comunidad de investigación ha desarrollado es la denominada "Spectral Normalization" [Miyato et al., 2018], la cual es un método diferente de normalización de los pesos [Salimans and Kingma, 2016] de las neuronas de la red que se emplea exclusivamente en el discriminante.

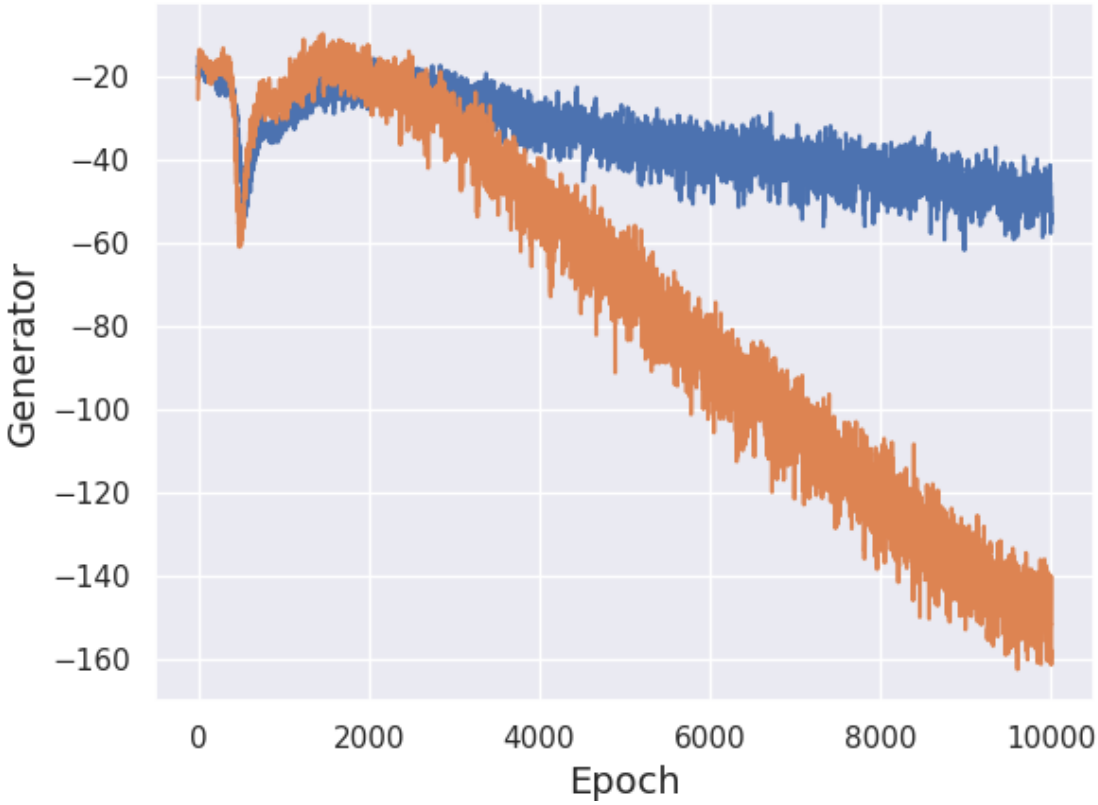


Figura 13: Función de pérdida del generador en naranja vs con Spectral Normalization en azul

En la Figura 13 vemos un gráfico con la misma tendencia que el presentado anteriormente, y podemos apreciar que el error del generador mejora considerablemente, manteniéndose más o menos estable durante el entrenamiento. Es curioso remarcar como el error cometido por el discriminante (ver Figura 14), además de no variar prácticamente al emplear o no Spectral Normalization, se mantiene estable durante todo el entrenamiento, esto sugiere la dominancia que conserva este sobre el generador durante la fase de aprendizaje.

El resultado de aplicar esta técnica, además de notarse en el error del generador, se aprecia en la calidad de las imágenes generadas, las cuales son mucho más realistas (ver Figura 15).

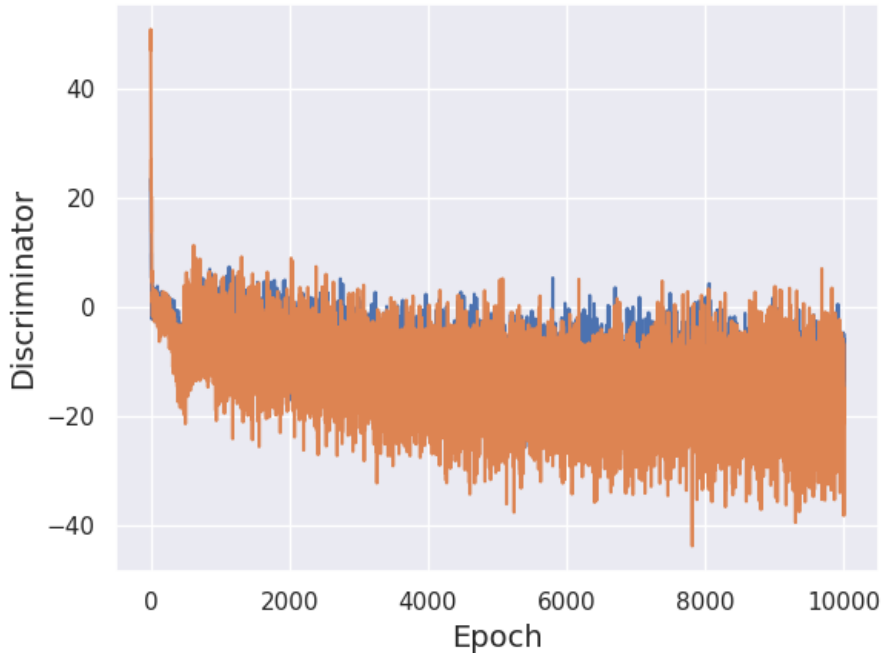


Figura 14: Función de pérdida del discriminante en naranja vs con Spectral Normalization en azul

Figura 15: Árboles de aguacate desde vista aérea generados de manera artificial con Spectral Normalization

Hiperparámetros

Cómo pudimos observar en las tablas 1 y 2 se han utilizado técnicas como Batch Normalization, Dropout, etc. La siguiente configuración de hiperparámetros es la que ha proporcionado los resultados mostrados y es la recomendada en las diferentes publicaciones en las que se ha basado la arquitectura.

- Red Generativa
  - Conv2D: **kernel size = 5**, **padding = same**
  - Batch Normalization: **momentum = 0.8**
  - Activation: **ReLU** y **tanh** en la última capa.
  - RMSprop Optimizer: **learning rate = 0.00001**
- Red Discriminativa
  - ConvSN2D: **kernel size = 3**, **strides = 2**, **padding = same**
  - Activation: **LeakyReLU con α = 0.2**
  - Dropout: **rate = 0.25**
  - Batch Normalization: **momentum = 0.8**
  - RMSprop Optimizer: **learning rate = 0.00004**

4.2. Parámetros para la DCNN

Para el refinado del modelo se han realizado diversas pruebas, a nivel de arquitectura, hiperparámetros, optimizadores, patrones de entrada, etc.

En cuanto a la arquitectura, Keras permite modificar y editar ciertas cosas de los modelos pre-entrenados, como las dimensiones de su entrada, añadirle más capas antes o después, pero también nos permite activar o desactivar el entrenamiento de cada una de sus capas. Esto es una de las características que se ha probado a la hora de hallar una configuración óptima. Cuantas menos capas se permitan entrenar de la red, peores resultados se obtienen, ya que son más capas las que mantienen los pesos sinápticos resultantes del entrenamiento con otro dataset. Sin embargo, tampoco se pueden entrenar demasiadas capas del modelo ResNet, ya que entonces pierde el sentido esa transferencia

de conocimiento. El punto que mejores resultados ha dado ha sido entrenar las últimas 20 capas de la red.

Los hiperparámetros escogidos para el entrenamiento y la construcción del modelo son los siguientes:

- ConvSN2D: **kernel size = 3**, **strides = 2**, **padding = same**
- Activation: **Softmax**
- Dropout: **rate = 0.5**
- RMSprop Optimizer: **learning rate = 0.0005**, **rho = 0.9**, **epsilon = 1e-08**, **decay = 0.1**

Entre ellos, la tasa de aprendizaje (learning rate) se ha probado además con 0.0001 y 0.001, siendo el primero demasiado bajo para terminar de converger durante el entrenamiento y 0.001 demasiado alto e inestable.

Finalmente, tras entrenar el modelo con validación cruzada (90 % de entrenamiento y 10 % de test) en **batches de 8** durante **40 épocas** con el conjunto de datos aumentado, la precisión obtenida alcanza un 82 %, lo cual es un resultado más que decente teniendo en cuenta la varianza que hay entre los distintos tipos de árboles de cultivo de una misma clase y la todavía escasa cantidad de imágenes.

5. Casos de uso

La aplicación web que plantea *Manvocado* aporta las 3 funcionalidades principales descritas en secciones anteriores, las cuales se organizan en pestañas diferenciadas. Esta aplicación se encuentra actualmente disponible en `http://manvocado.herokuapp.com/`, a la cual se puede acceder y utilizar libremente. Pasamos a detallar su uso a continuación.

5.1. Aporte de imágenes por parte del usuario

En la pestaña de Aportar, los usuarios pueden subir tantas imágenes como quieran haciendo clic en el recuadro de input. Tras seleccionar las imágenes que se crean convenientes, se debe seleccionar la clase a la que pertenecen dichas imágenes, sean aguacates, mangos o construcciones. Debajo de las clases se muestran las imágenes seleccionadas y un botón de carga (Upload) para enviarlas a la aplicación.

Figura 16: Ejemplo de la aportación de imágenes en la aplicación web

En la Figura 16 podemos observar como se vería la aplicación al intentar subir 6 imágenes de la clase Aguacate. Como se comenta en la pestaña principal (Home), es importante que estas imágenes sean de dimensiones 256x256 píxeles, sino se mostrará un mensaje de error y no se permitirá subir las imágenes a la aplicación. En caso de que se hayan subido correctamente las imágenes, el usuario será redirigido a una pestaña donde se le agradecerá su contribución con el proyecto, como se muestra en la Figura 17.

Como podemos observar, esta funcionalidad facilita la interacción del humano respecto a la captación de nuevas imágenes y su etiquetado por parte del experto.

Figura 17: Página de agradecimiento por contribuir con el proyecto

5.2. Clasificación de imágenes

La pestaña de Clasificar permite a los usuarios seleccionar entre 4 imágenes, las que crea que pertenezcan a la clase que se especifique. Las imágenes se seleccionan mediante un clic, pudiéndose deseleccionar de la misma manera. Como observamos en la Figura 18, la clase por la que se está preguntando es Aguacate, y en este caso, se ha seleccionado únicamente una imagen, que se puede apreciar en un tono grisáceo. Una vez se hayan seleccionado las imágenes, pulsando el botón de Clasificar, se enviarán los votos a la aplicación, recargando las imágenes que se muestran y haciendo que la aplicación pregunte por otra clase distinta.

En caso de que el usuario no tenga claro qué imágenes puedan pertenecer a la clase que se pide, mediante el botón de Recargar se pueden cambiar las imágenes para hacer el etiquetado más fácil, evitando así posibles errores.

Figura 18: Ejemplo del etiquetado de imágenes de la base de datos en la aplicación web

5.3. Análisis predictivo de imágenes

Por último, la pestaña de Predecir que muestra la Figura 19 permite a los usuarios cargar, igual que en la pestaña de Aportar, una imagen. De nuevo, esta deberá cumplir las restricciones y ser de dimensiones 256x256 píxeles. Una vez subida, se mostrará la imagen que se ha seleccionado y al pulsar el botón Predecir, se mostrará la clase que el modelo ha estimado para dicha imagen. Cuando se muestra el resultado, se muestra tanto la clase a la que pertenece como el porcentaje de confianza con la que se ha realizado esta predicción. En caso de que esta predicción no sea fiable, bien por culpa del modelo, o bien por culpa de la imagen seleccionada, se mostrará que la imagen no se ha podido clasificar. En el ejemplo se muestra una imagen de árboles de aguacate, y el modelo es capaz con una confianza del 94.85 % de clasificarla correctamente como Aguacate.

Figura 19: Ejemplo de predicción en la aplicación web

6. Discusión

Tras la presentación de la aplicación y el ecosistema a nivel teórico y práctico, a modo de discusión se plantea el impacto que este tipo de plataformas tienen así como su potencial a la hora de apoyar aplicaciones de Inteligencia Artificial.

La comunidad agrónoma, uno de los grandes sectores de producción, necesita aprovecharse de los avances tecnológicos no únicamente a nivel técnico y logístico, sino también a nivel estratégico. Gracias a aplicaciones de Machine Learning y al análisis predictivo pueden realizarse estudios novedosos que no son posibles de otra manera, el aprovechamiento de la información puede brindar nuevas estrategias al sector gracias a la obtención de conocimientos previamente imposibles de saber, pudiendo así por ejemplo, prever posibles plagas, mejorar el tratado de los suelos de cultivo, una recogida de frutos eficiente, e infinidad de otras posibles aplicaciones.

El ecosistema presentado se apoya en la idea de las grandes empresas con los conocidos Captcha[24], donde los usuarios, creyendo que únicamente están realizando una verificación para prevenir el uso de procesos automatizados (bots), están realizando también una labor de clasificación de imágenes o sonidos para dichas empresas. Esta es una importante carga de trabajo que las empresas gestionan de manera inteligente, y con esta mentalidad, se plantea un ecosistema autosostenible que se aproveche del trabajo incondicional de los usuarios para validar y hacer crecer el conjunto de datos de nuestro proyecto, con el fin de mejorar la precisión y el rendimiento de nuestros modelos de Aprendizaje Profundo. Estos beneficios son inmediatos, pues permiten generar bases de imágenes etiquetadas de manera transparente al usuario, por lo que se podrán entrenar modelos de manera más eficiente, permitiendo generar estimaciones sobre la superficie real cultivada en una región, ayudando así a la estrategia y control de producciones.

En este sentido, se ha recibido una evaluación muy positiva por parte de la industria agroalimentaria mediante la empresa TROPS.

[24] ¿Qué es un Captcha?: `https://support.google.com/a/answer/1217728?hl=es`

7. Conclusiones y Trabajo Futuro

Gracias a los modelos computacionales, la aplicación web y el dataset desarrollados y publicados en los repositorios, se han cumplido los objetivos establecidos al comienzo del proyecto.

Se han realizado dos modelos de redes neuronales con Keras y TensorFlow, una DCGAN para la aumentación de manera artificial del conjunto de datos original y favorecer al entrenamiento del segundo modelo. Una red DCNN que actúa de clasificador de los 2 tipos de cultivos estudiados, el cual es capaz de discernir entre estos tipos de árboles en una plantación. Como acercamiento al uso de estos modelos de manera sencilla, se ha realizado una aplicación web con Python, la cual no solo integra el clasificador y muestra el poder de predicción del modelo, también permite al usuario, en forma de "juego", etiquetar las imágenes que se han subido a la aplicación, las cuales tras un proceso de validación, pasarán a formar parte del dataset.

El problema más difícil de solventar, como es común en el campo del Aprendizaje Profundo, fue la creación del dataset, que requirió de la ayuda de un profesional y del recorte de decenas de imágenes de forma manual para obtener un punto de partida decente. Además, el proceso de aumentación de datos descrito supuso un enorme reto, debido a que las redes GAN son recientes y la literatura acerca de ellas es escasa, lo que resultó en una importante carga de investigación para su entrenamiento. Por otro lado, dificultades más relacionadas con las imágenes son las siguientes: se requiere de un profesional en el campo de investigación sobre el que se realice la aplicación, en este caso, un experto que sea capaz de discernir entre los tipos de cultivos; la varianza dentro de los cultivos de una misma clase, teniendo en cuenta la disposición de la plantación y el tamaño de los árboles según el tiempo que haya transcurrido desde su plantación, suponen que haya una gran diversidad de imágenes dentro de una misma clase; la calidad de la imagen obtenida, teniendo en cuenta sombras de las nubes, rastros de uso de artefactos para la limpieza de las imágenes, sombras ocasionadas por los propios árboles debido a la posición del sol... Todos estos problemas dificultan la creación de un conjunto de datos de calidad, complicando considerablemente el desarrollo de modelos predictivos de precisión.

Otras dificultades encontradas durante la realización de este trabajo se deben a la infraestructura, teniendo que llevar a cabo el entrenamiento de los modelos en mi ordenador personal. Resultado ser una GPU como la Nvidia GeForce GTX 1660 Ti, insuficiente para

realizar un estudio intensivo de hiperparámetros de los modelos propuestos por motivos de tiempo y recursos como la memoria de la GPU.

Para el correcto entendimiento de los modelos neuronales desarrollados, me ha sido muy útil la asignatura Modelos de la Computación, gracias a la cual he sido capaz de entender el funcionamiento de los modelos y el tratamiento con imágenes en detalle.

Realizar el Trabajo de Fin de Grado formando parte del grupo de investigación Khaos Research[25] de la Universidad de Málaga ha sido de gran ayuda a la hora de pensar en estrategias e ideas durante el proyecto, debido a la gran experiencia de sus miembros en el campo de Machine Learning y el desarrollo de software aplicado a proyectos de biología.

En cuanto a trabajos futuros, se pretende utilizar vehículos UAV para la captación de imágenes de alta resolución, facilitando así el entrenamiento de modelos neuronales específicos, permitiendo obtener resultados precisos a la hora de realizar clasificaciones. Con dichos medios, se pretende realizar un estudio de los terrenos de cultivos, analizando problemas más relevantes como la salud de los árboles o del terreno, factores que dictan la productividad del cultivo y que resultan de gran ayuda a la comunidad agrónoma, pudiendo actuar a tiempo ante futuras adversidades, desplegar un número razonable de trabajadores, etc.

Por otro lado, con respecto a la aplicación web, se plantea automatizar la recogida de imágenes de satélite mediante scripts de Earth Engine[26] para recolectar imágenes de las dimensiones requeridas de manera masiva, exponiéndolas así en el captcha de la aplicación para apoyar el etiquetado y validación humana de imágenes.

[25]Khaos Research Group: `https://khaos.uma.es/`

[26]Earth Engine: `https://earthengine.google.com/`

Referencias

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. URL `https://arxiv.org/abs/1701.07875`.

T. Blaschke. Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(1):2 – 16, 2010. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2009.06.004. URL `http://www.sciencedirect.com/science/article/pii/S0924271609000884`.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Xue-Wen Chen and Xiaotong Lin. Big data deep learning: Challenges and perspectives. *Access, IEEE*, 2:514–525, 01 2014. doi: 10.1109/ACCESS.2014.2325029.

Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2094–2107, 2014.

François Chollet et al. Keras. `https://keras.io`, 2015.

Python Software Foundation, 2019. URL `https://docs.python.org/3/faq/general.html`.

Keith Fuglie. The growing role of the private sector in agricultural research and development world-wide. *Global Food Security*, 10:29 – 38, 2016. ISSN 2211-9124. doi: https://doi.org/10.1016/j.gfs.2016.07.005. URL `http://www.sciencedirect.com/science/article/pii/S2211912416300190`.

Emilio Guirado, Siham Tabik, Marga L. Rivas, Domingo Alcaraz-Segura, and Francisco Herrera. Whale counting in satellite and aerial images with deep learning. *Scientific Reports*, 9, 10 2019. doi: 10.1038/s41598-019-50795-9. URL https://www.nature.com/articles/s41598-019-50795-9.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. URL https://arxiv.org/abs/1704.00028.

Joseph Halpern. Computer science and game theory: A brief survey. *Palgrave Dictionary of Economics*, 04 2007.

Jez Humble, Chris Read, and Dan North. The deployment production line. In *Proceedings of the Conference on AGILE 2006*, AGILE '06, page 113–118, USA, 2006. IEEE Computer Society. ISBN 0769525628. doi: 10.1109/AGILE.2006.53. URL https://doi.org/10.1109/AGILE.2006.53.

Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks. URL https://arxiv.org/abs/1406.2661.

David J. Lary, Amir H. Alavi, Amir H. Gandomi, and Annette L. Walker. Machine learning in geosciences and remote sensing. *Geoscience Frontiers*, 7(1):3 – 10, 2016. ISSN 1674-9871. doi: https://doi.org/10.1016/j.gsf.2015.07.003. URL http://www.sciencedirect.com/science/article/pii/S1674987115000821. Special Issue: Progress of Machine Learning in Geosciences.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018. URL https://arxiv.org/abs/1802.05957.

Stefano Nativi, Paolo Mazzetti, Mattia Santoro, Fabrizio Papeschi, Max Craglia, and Osamu Ochiai. Big data challenges in building the global earth observation system of systems. *Environmental Modelling & Software*, 68:1 – 26, 2015. ISSN 1364-8152. doi: https://doi.org/10.1016/j.envsoft.2015.01.017. URL http://www.sciencedirect.com/science/article/pii/S1364815215000481.

James Lindenbaum Orion Henry and Adam Wiggins. Heroku: Cloud application platform, 2007. URL https://www.heroku.com/.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

PalletsProjects. URL https://palletsprojects.com/p/flask/.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2012. URL https://arxiv.org/abs/1211.5063.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. 12 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf.

Juan Rodríguez, Aritz Pérez, and J.A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32:569 – 575, 04 2010. URL https://www.researchgate.net/publication/224085226_Sensitivity_Analysis_of_k-Fold_Cross_Validation_in_Prediction_Error_Estimation.

Frank F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958. URL https://pdfs.semanticscholar.org/5d11/aad09f65431b5d3cb1d85328743c9e53ba96.pdf.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks, 2016. URL https://arxiv.org/abs/1602.07868.

Ian Sommerville. Ingeniería del software. URL http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI_3242/IngenieriadelSoftware7ma.Ed.-IanSommerville.pdf.

M. Weiss, F. Jacob, and G. Duveiller. Remote sensing for agricultural applications: A meta-review. *Remote Sensing of Environment*, 236:111402, 2020. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2019.111402. URL http://www.sciencedirect.com/science/article/pii/S0034425719304213.

Jifang Xing, Zhang Ruixi, Remmy Zen, Dewa Made Sri Arsa, Ismail Khalil, and Stéphane Bressan. Building extraction from google earth images. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*, iiWAS2019, page 502–511, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450371797. doi: 10.1145/3366030.3368456. URL https://doi.org/10.1145/3366030.3368456.

Christopher J.C. Burges Yann LeCun, Corinna Cortes. Mnist handwritten digit database. URL http://yann.lecun.com/exdb/mnist/.

ANEXO

A. Carta de Apoyo de la empresa TROPS

A continuación se anexa la carta de apoyo de la empresa SAT 2803 TROPS comentada a lo largo de esta memoria.

Vélez Málaga, 17/06/2020

A quien pueda interesar:

Deseamos manifestar nuestro apoyo al **Trabajo Fin de Grado con título "*ANÁLISIS SUPERVISADO DE IMÁGENES AÉREAS USANDO TÉCNICAS DE APRENDIZAJE PROFUNDO APLICADO A DETECCIÓN DE CULTIVOS TROPICALES*", del alumno Cristian Cardas Ezeiza, que se presenta en el GRADO EN INGENIERÍA INFORMÁTICA, DE LA ESCUELA SUPERIOR DE INGENIERIA INFORMÁTICA, DE LA UNIVERSIDAD DE MÁLAGA.**

Desde el departamento de I+D+i y Sostenibilidad de **SAT 2803 TROPS**, hemos probado la herramienta desarrollada y nos resulta muy interesante a nivel estratégico y operacional, al mismo tiempo que estamos interesados en continuar con tal línea de desarrollo.

Quedamos a su disposición en todo lo que pueda requerir de nosotros a este respecto.

Reciban nuestros más cordiales saludos.

JESÚS REGODÓN RUIZ
DIRECTOR I+D+i Y SOSTENIBILIDAD DE SAT 2803 TROPS

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA