Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

Trabajo Fin de Máster

Desarrollo de un prototipo de delimitación automática de parcelas a partir de imágenes de satélite

Presentado por Willow Maui García Moreno
en Universidad de Burgos — 19 de julio de 2024
Tutores: Dr. José Francisco Díez Pastor

Universidades de Burgos, León y Valladolid

Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. José Francisco Díez Pastor, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Willow Maui García Moreno, con DNI 71706742X, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado Desarrollo de un prototipo de delimitación automática de parcelas a partir de imágenes de satélite.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 19 de julio de 2024

Vº. Bº. del Tutor:

Dr. José Francisco Díez Pastor

Resumen

En el contexto de la agricultura, la detección precisa de los límites de las parcelas agrícolas es fundamental para una amplia gama de aplicaciones, como la gestión eficiente de los cultivos, el monitoreo de la producción y los rendimientos, el cumplimiento normativo y la solicitud de subvenciones, el análisis de cambios en el uso del suelo, y la gestión de la propiedad de la tierra.

En este trabajo, los autores del artículo "AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photography" presentan un dataset compuesto por imágenes satelitales del satélite Sentinel-2 y ortofotos de varios países europeos, con el objetivo de abordar el reto de la detección automática de los límites de las parcelas. Utilizando este conjunto de datos, se entrenaron diversos modelos de inteligencia artificial, con especial énfasis en una red neuronal convolucional de tipo U-Net, para evaluar su eficacia en la tarea de delimitación de los campos de cultivo.

Los resultados obtenidos en este estudio sientan las bases para futuras investigaciones en este campo de gran relevancia para el sector agrícola, al demostrar el potencial de las técnicas de aprendizaje automático para automatizar la detección de los límites de las parcelas a partir de imágenes satelitales y aéreas.

Descriptores

Visión artificial, *Big Data*, Machine Learning, Redes neuronales convolucionales, Inteligencia artificial, Detección de bordes.

Abstract

In the context of agriculture, the precise detection of agricultural field boundaries is crucial for a wide range of applications, such as efficient crop management, production and yield monitoring, regulatory compliance and subsidy application, analysis of land use changes, and land property management.

In this work, the authors of the article "AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photography" present a dataset composed of Sentinel-2 satellite images and orthophotos from several European countries, with the aim of addressing the challenge of automatic detection of field boundaries. Using this dataset, various AI models were trained, with a particular emphasis on a U-Net convolutional neural network, to evaluate their effectiveness in the task of delineating croplands.

The results obtained in this study lay the groundwork for future research in this highly relevant field for the agricultural sector, by demonstrating the potential of machine learning techniques to automate the detection of field boundaries from satellite and aerial imagery.

Keywords

Computer vision, *Big Data*, Machine Learning,Convolutional neural networks, Artifical Intelligences, border detection.

Índice general

Índice de figuras

Índice de tablas

Memoria

Capítulo 1

Introducción

En la actualidad, en el contexto de la agricultura, los límites de las parcelas son un aspecto muy relevante, ya que nos permiten conocer la extensión de cada tipo de cultivo y su ubicación. Además, estos límites son fundamentales para cuestiones relacionadas con la propiedad de los terrenos y los posibles impuestos asociados.

El problema radica en que estos límites son cambiantes por diversos motivos, como pueden ser los cambios de propiedad o la unión de varios campos de cultivo el mismo tipo. Si bien existen registros de la propiedad que dividen estas parcelas en muchos países, estos no siempre representan la realidad de las mismas, no las cubren en su totalidad a lo largo del mundo y pueden ser complejos de consultar y analizar de forma automática.

Para abordar este problema, los autores del artículo *"AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photography"* [18] recopilaron una serie de imágenes satelitales, las clasificaron y realizaron una separación de los terrenos en base a las bases de datos regionales de cada uno de los países involucrados. Estas imágenes proceden de Austria, España (concretamente de Cataluña), Francia, Luxemburgo, Países Bajos, Eslovenia y Suecia, y se han obtenido a partir del satélite Sentinel 2 [14] y de ortofotos extraídas del "Geospatial Aid Application (GSAA)" [2]. En total, se cuenta con 7.831 muestras, cada una de ellas acompañada de la delimitación correspondiente.

En este trabajo, se han utilizado las imágenes del satélite Sentinel 2, las cuales están compuestas por 5 bandas espectrales: las bandas RGB (rojo, verde y azul), una banda infrarroja y la banda NDVI (Normalised Difference Vegetation Index), que muestra la cantidad de vegetación visible.

A partir de estos datos, se ha procedido a procesar la información y entrenar diversos modelos de inteligencia artificial para comprobar su eficacia en la tarea de detección de los límites de las parcelas. Se ha puesto especial énfasis en el entrenamiento de una red neuronal convolucional de tipo U-Net, debido tanto a los resultados obtenidos en trabajos previos como a la concepción teórica de que es la más adecuada para el contexto del problema.

1.1. Motivación y relevancia del problema

La detección precisa de los límites de las parcelas agrícolas es fundamental para una amplia gama de aplicaciones en el sector agrícola. Algunos de los principales beneficios y aplicaciones de esta tecnología incluyen:

- **Gestión eficiente de los cultivos**: Conocer con precisión los límites de las parcelas permite a los agricultores y gestores de tierras planificar y administrar de manera más eficiente los diferentes cultivos, optimizando el uso de recursos como agua, fertilizantes y pesticidas.
- **Monitoreo de la producción y los rendimientos**: La delimitación de las parcelas facilita el seguimiento de la producción y los rendimientos de cada cultivo, lo que permite tomar decisiones informadas sobre la gestión de la explotación agrícola.
- **Cumplimiento normativo y subvenciones**: En muchos países, los límites de las parcelas son fundamentales para el cumplimiento de las normativas agrícolas y la solicitud de subvenciones y ayudas gubernamentales.
- **Análisis de cambios en el uso del suelo**: La detección de los límites de las parcelas a lo largo del tiempo permite analizar los cambios en el uso del suelo, lo que es crucial para la planificación y la toma de decisiones en el ámbito de la agricultura y el desarrollo rural.
- **Catastro y gestión de la propiedad**: La delimitación precisa de las parcelas es esencial para los registros catastrales y la gestión de la propiedad de la tierra. Dada la importancia de este problema, el desarrollo de soluciones eficientes y automatizadas para la detección de los límites de las parcelas agrícolas a partir de imágenes satelitales es un área de investigación de gran relevancia y con un gran potencial de impacto en el sector agrícola.

1.2. Estructura del documento

Este trabajo de fin de máster se estructura de la siguiente manera:

- **Introducción**: Se presenta el contexto y la relevancia del problema de la detección de los límites de las parcelas agrícolas, así como los objetivos del trabajo.
- **Objetivos**: Se detallan los objetivos principales y específicos del proyecto.
- **Conceptos teóricos**: Se abordan los conceptos teóricos relevantes para el desarrollo del proyecto.
- **Técnicas y herramientas**: Se describen las técnicas y herramientas utilizadas en el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto**: Se presentan los detalles más relevantes del proceso de desarrollo.
- **Trabajos relacionados**: Se revisan los trabajos previos relacionados con la detección de bordes de parcelas a partir de imágenes satelitales.
- **Conclusiones y líneas de trabajo futuras**: Se resumen las principales conclusiones del trabajo y se plantean posibles líneas de investigación futuras.

Capítulo 2

Objetivos

En esta sección se detallarán los objetivos a cumplir en este trabajo.

Objetivo Principal

El objetivo principal de este trabajo de fin de máster es desarrollar un modelo de detección de bordes de parcelas eficiente y preciso que pueda ser aplicado a imágenes satélite de alta resolución procedentes del satélite Sentinel 2. Se buscará entrenar un modelo de inteligencia artificial capaz de diferenciar de manera eficiente y acertada los bordes de las parcelas en este tipo de imágenes.

Objetivos específicos

Para lograr el objetivo principal, se plantean los siguientes objetivos específicos:

- Entrenar un modelo capaz de realizar una separación suficiente entre parcelas y sus bordes en base a una imagen satelital.
- Probar diferentes métodos y configuraciones de cara a comprobar si realmente la red neuronal convolucional es el método más potente o pueden ser más interesantes otras metodologías que impliquen que no tener en cuenta el contexto de los píxeles.
- Exponer los conocimientos extraídos en un notebook de Python, lo cuál facilitará la trasmisión y la manipulación de estos.

- Generar un repositorio dónde almacenar dicho notebook con la estructura definida para una poder ejecutar de forma simple el cuaderno.
- Tratar de optimizar en la medida de lo posible los parámetros del modelo para maximizar su eficiencia.

Capítulo 3

Conceptos teóricos

En este apartado se explicaran el grueso de los conceptos necesarios para la correcta comprensión del trabajo y el material. Teniendo tanto en cuenta el funcionamiento de los mecanismos internos como los conceptos abstractos.

3.1. Visión e imágenes

Una imagen es una representación visual de un objeto, escena o perspectiva, ya sea de forma física (como una fotografía) o digital (como una imagen digital). Para comprender cómo se compone una imagen, es crucial explicar el funcionamiento de la visión humana.

La visión [1] es un proceso complejo que se produce cuándo determinados fotorreceptores en el interior de nuestros ojos, llamados conos y bastones, reciben la luz reflejada de los objetos, esta luz reflejada para que sea visible por nuestros ojos tiene que venir reflejada entre los valores de frecuencia de 460 nanómetros (nm) y 780 nm, correspondiente al espectro visible para el ojo humano.

Dentro de este rango, existen tres tipos de conos, cada uno sensible a una parte del espectro: los conos largos, que detectan las frecuencias del rojo, con una longitud de onda entre 618 y 780 nm; los conos medios, que detectan las frecuencias del verde, con una longitud de onda entre 529 y 497 nm; y los conos cortos, que detectan las frecuencias del azul, con una longitud de onda entre 460 y 482 nm. Estas tres señales se combinan en nuestro cerebro, permitiéndonos percibir una amplia gama de colores.

Así los colores que nosotros somos capaces de percibir son una conjunción de los distintos grados de activación de esos tres tipos de sensores, es decir, los distintos colores que interpretamos son una combinación de Rojo, Verde y Azul, por lo tanto el modelo de color que contempla esta composición se llama RGB, por los nombres en inglés de *Red, Green and Blue*.

Esto es relevante para este trabajo ya que nos permite comprender como las imágenes digitales se representan, ya que estas se componen de tres capas o canales, cada uno correspondiente a una de las bandas del espectro RGB. La combinación de estos tres canales permite reproducir la gama de colores que percibimos en la realidad.

Aunque estas capas solo son una selección de las frecuencias reflejadas por como funciona el ojo humano, pero realmente existen otros tipos de radiación electromagnética que se encuentran fuera de este espectro visible.Algunas de estas radiaciones pueden ser captadas por sensores especializados y utilizadas en diversos campos, como la teledetección y la investigación científica.

Concretamente en este trabajo se utiliza una banda más, de una longitud de onda mayor que el rojo, es decir una frecuencia infrarroja. El satélite Sentinel 2 es capaz de recoger hasta 13 bandas [15], con distintas resoluciones y longitudes de onda, de las cuales se utilizaran 4 y una composición entre varias de ellas:

- **Banda 2**: Esta banda es la correspondiente al color azul, con una frecuencia de onda de 490 nm.
- **Banda 3**: Esta banda es la correspondiente al color verde, con una frecuencia de onda de 560 nm.
- **Banda 4**: Esta banda es la correspondiente al color rojo, con una frecuencia de onda de 665 nm.
- **Banda 8**: Esta banda es la correspondiente al una frecuencia infrarroja de 842 nm. Esta banda es relevante ya que a esa frecuencia de onda la luz no se refleja correctamente en la clorofila, por tanto te permite saber el estado de la vegetación de la zona.
- **Banda NDVI**: Esta es una combinación de bandas, NDVI son las siglas para *Normalized Difference Vegetation Index*, esta es muy útil para diferenciar zonas en función de la vegetación que poseen, la formula es:

  $NDVI = \frac{B8-B4}{B8+B4}$

Estas bandas, al ser combinadas y procesadas, permiten obtener información valiosa sobre las características y el estado de la superficie terrestre, lo cual es fundamental para los objetivos de este trabajo.

3.2. Inteligencia artificial

Dentro de esta sección se explicará algunos modelos de inteligencia artificial y su funcionamiento además de conceptos teóricos básicos para entender esta materia. La inteligencia artificial [20] consiste en programas capaces de usar algoritmos para aprender de unos datos y ser capaz de tomar decisiones en base a ellos.

La inteligencia artificial se usa en la actualidad para multitud de tareas:

- Reconocimiento de imágenes estáticas, clasificación y etiquetado.
- Mejoras del desempeño de la estrategia algorítmica comercial.
- Procesamiento eficiente y escalable de datos de pacientes.
- Distribución de contenido en las redes sociales
- Protección contra amenazas de seguridad cibernética

El aprendizaje automático o *Machine Learning* [6] es uno de los principales enfoques de la inteligencia artificial, este utiliza algoritmos para aprender de conjuntos de datos. Este se divide en tres categorías de manera principal:

- **Aprendizaje supervisado**: En este tipo de aprendizaje los algoritmos usan datos previamente etiquetados, es decir que ya tienen una categoría asignada. Esto hace que el modelo a entrenar extraiga las distintas características que convierten el dato de entrada en la categoría de salida. Este tipo de algoritmos se utilizan para la predicción y clasificación de datos. Este es el que se ha utilizado para este trabajo. El deep learning se encuadra dentro de este apartado. Algunos de los métodos más utilizados de este tipo de aprendizaje son las redes neuronales, los clasificadores bayesianos, la regresiones lineales, los bosques aleatorios y las máquinas de vectores de soporte.
- **Aprendizaje no supervisado**: En este tipo de aprendizaje se utilizan algoritmos para analizar y agrupar datos no etiquetados. En estos los

algoritmos descubren patrones ocultos a simple vista y agrupaciones sin necesidad de intervención humana. Se utilizan para clasificación y agrupación por lo tanto. Entre sus usos se encuentra el análisis exploratorio de datos, la segmentación de datos, el reconocimiento de patrones e incluso la reducción de la dimensionalidad. Algunos métodos utilizados son el análisis de componentes principales, la descomposición en valores singulares, El algoritmo de K-medias y métodos de agrupación probabilística.

- **Aprendizaje semisupervisado**: Este es una variante intermedia a los anteriores, dónde en el entrenamiento se utilizan datos etiquetados sobre un conjunto de datos mayor dónde el resto están sin etiquetar, este te ayuda a etiquetarlos.

También cabe destacar el Aprendizaje por refuerzo, el cual funciona como un mecanismo de condicionamiento operante, dónde al algoritmo se le entrena sobre la marcha validando si ha realizado la predicción correcta o no y ajustando este su algoritmo interno para aprender en función de esto.

Es relevante comprender para el contexto de este trabajo que en la clasificación de imágenes binaria, o expresado de otra manera una segmentación de la imagen, las etiquetas consisten en una capa de la imagen, llamada *ground truth*, con los valores de 0 y 1 para cada pixel en función de la salida esperada.

Métricas de aprendizaje supervisado

La evaluación de estos modelos de aprendizaje supervisado se puede hacer con diversos métodos, aunque las medidas más básicas son las que se detallaran en este apartado.

La base de todas las métricas [13] que se van a detallar es la **matriz de confusión**. Esta se compone por cuatro valores en una matriz 2 x 2 al haber dos clases entre las que segmentar. Estos valores son:

- True Positives: Este está en la primera posición de la matriz, se corresponde con el número de predicciones que han sido predichas como positivas de manera correcta. En adelante se llamará TP.
- True Negatives: Este está en la última posición de la matriz, se corresponde con el número de predicciones que han sido predichas como negativos de manera correcta. En adelante se llamará TN.

- False Positives: Este está en la segunda posición de la matriz, se corresponde con el número de predicciones que han sido predichas como positivas de manera incorrecta
- False Negatives: Este está en la primera posición de la matriz, se corresponde con el número de predicciones que han sido predichas como negativas de manera

Ahora se explicarán el resto de medidas, que se calculan sobre los datos de la matriz de confusión.

Accuracy

Esta es la principal medida. La precisión representa el porcentaje total de valores correctamente clasificados sobre el total de valores. Esta medida es útil cuando los datos están balanceados. La formula por tanto es:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Precisión

Esta medida prioriza el valor de los TP, ya que lo que nos plantea es que porcentaje de valores han sido correctamente calificados como positivos sobre todos los valores calificados como positivos. La fórmula por tanto es:

$$Precisión = \frac{TP}{TP + FP} \tag{3.2}$$

Recall

La métrica de recall, es el ratio de verdaderos positivos y sirve para saber cuántos de los valores positivos han sido correctamente clasificados. La formula por tanto es:

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

F1 Score

Esta métrica es útil en conjuntos de datos desbalanceados. Esta combina el recall y la precisión. La formula es:

$$F1Score = 2 * \frac{recall * precision}{recall + precision} \tag{3.4}$$

Estás son las métricas que se han utilizado para medir este proyecto, aunque hay otras médidas muy usadas que también definiré.

Curva ROC

La curva ROC (Receiver Operating Characteristic) es un gráfico muy utilizado para problemas de clasificación. Esta representa el porcentaje de verdaderos positivos contra el ratio de falsos positivos. Lo que diferencia a esta métrica de las demás es que el umbral por el que se clasifica un elemento como 0 o 1 se va modificando para generar todos los valores de la gráfica. El valor óptimo del umbral será el que más cercano se encuentre respecto a la esquina izquierda superior del gráfico.

AUC

Esta métrica es derivada de la anterior, la métrica AUC (Area Under Curve) o área bajo la curva. El valor de esta métrica está entre 0 y 1 siendo 0 una clasificación aleatoria y 1 una clasificación perfecta. Esta se calcula mediante integrales.

3.3. Algunos modelos de Inteligencia artificial

Ahora se explicarán en profundidad algunos de los algoritmos previamente mentados, concretamente los relacionados con este trabajo.

KNN

KNN [7] son las siglas para *K Nearest Neighbours*, es decir K vecinos más cercanos, este clasificador de aprendizaje supervisado utiliza la proximidad para hacer clasificaciones o predicciones, aunque generalmente se suele utilizar para clasificación.

KNN forma parte de la familia de algoritmos de aprendizaje perezoso, ya que en la fase de entrenamiento no realiza cálculos, sino que almacena las instancias, realizándose estos cálculos en la fase de predicción.

También por su funcionamiento se le considera basado en instancias.

Es un algoritmo popular debido a su simplicidad y precisión pero no escalan bien sus capacidades con el crecimientos del conjunto de datos.

Este asigna una etiqueta en función de un calculo de distancia con las instancias del modelo, por lo que uno de los conceptos más relevantes dentro de este modelo son las métricas de distancia, que definirán cuánto se de lejos están unas instancias a otras.

Las ventajas y desventajas de este algoritmo son:

Ventajas:

- Facilidad de implementación: Como se ha visto el algoritmo es relativamente sencillo.
- Gran adaptabilidad: Ya que al introducir siempre nuevas instancias da una mayor versatilidad.
- Pocos hiperparámetros: Ya que los unicos parámetros que requiere son el valor de K y la medida de distancia.

Desventajas:

- Mal escalado: Como ya se ha comentado, cuándo el número de instancias crece ralentiza la predicción del modelo y hace que requiera más recursos, al tener que cargar en memoria las instancias.
- Mal funcionamiento para alta dimensionalidad: Aumentando el número de errores acorde a la dimensionalidad.
- Propensión al sobreajuste: Por su funcionamiento tiende a sobreajustarse en función del valor K, en especial si es muy bajo, pero en caso de tener un K muy alto tiende a ajustarse mal alos datos

Su funcionamiento consiste en calcular la distancia de la nueva instancia con el resto de las del modelo y entre las K más cercanas ver que clase es la mayoritaria, esta será la clase predicha.

Algunas de las métricas mas usadas para el cálculo de distancias son:

- **Distancia euclidiana**: Está es de las más utilizadas. Se calcula la distancia de la línea recta entre dos puntos mediante la formula:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2} \tag{3.5}$$

  n = número de dimensiones del punto.

- **Distancia Manhattan**: Esta es una medida más sencilla que mide la distancia entre dos puntos tomando como trayectoria los vértices del eje. Su formula es:

$$d(a,b) = \sum_{i=1}^{n} |a_i - b_i| \tag{3.6}$$

  n = número de dimensiones del punto.

- **Distancia Minkowski**: Esta es la utilizada en el experimento. Es la generalización de las dos anteriores, siendo la p en Manhattan 1 y en la euclídea 2. La formula por tanto es:

$$d(a,b) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}} \tag{3.7}$$

  n = número de dimensiones del punto.

- **Distancia de Hamming**: Esta técnica es utilizada de forma normal para vectores booleanos o cadenas, buscando identificar igualdades i diferencias. Es el sumatorio de los items diferentes.

Se utiliza entre otros usos para:

- Preprocesamiento de datos
- Finanzas
- Cuidado de la salud
- Reconocimiento de patrones

Random Forest

Random Forest [4] o árboles aleatorios es un algoritmo supervisado basado en la combinación de diversos árboles de decisión binarios para alcanzar un único resultado.

Es un algoritmo popular debido a su flexibilidad y facilidad de uso.

Para comprender el Random Forest es necesario comprender antes los árboles de decisión simples, ya que este es una composición de ellos.

Los árboles de decisión binarios se basan en respuestas binarias a preguntas, dónde separan en ramas u hojas los caminos en función de si esa

repuesta lleva a otra pregunta (rama) o a una clasificación final (hoja). De esta manera, a base de condiciones se van clasificando las distintas instancias que le llegan al árbol. Las ramas se pueden configurar para definir cuales son las condiciones que se consultan en cada rama, aunque son propensos a problemas como sesgos y sobreajustes. Estos suelen cubrir todo el espacio de posibilidades.

Es un método de aprendizaje por multiclasificador, lo que significa que se compone de varios clasificadores que se agregan para identificar el resultado más popular. Estos se entrenan de manera independiente y después se promedian los resultados.

Ahora, el algoritmo del Random Forest consiste en muchos arboles binarios con características aleatorias, es decir con condicionantes aleatorios. Estos por lo tanto no tienen porque cubrir todo el espacio de probabilidades.

El algoritmo del Random Forest tiene tres hiperparámetros principales, el tamaño del nodo, la cantidad de arboles y la cantidad de características muestreadas.

Dependiendo del tipo de problema, la determinación de la predicción variará. Para una tarea de regresión, se promediarán los árboles de decisión individuales, y para una tarea de clasificación, un voto mayoritario, es decir, la variable categórica más frecuente, arrojará la clase predicha.

Las ventajas y desventajas de este modelo son:

Ventajas:

- Bajo riesgo de sobreajuste: si bien anivel interno se pueden sobreajustar los arboles de decisión al haber gran cantidad de ellos el modelo no se ajusta demasiado al modelo, ya que el promedio de árboles no correlacionados reduce la varianza general y el error de predicción.
- Gran flexibilidad: es un metodo popular porque funciona de manera eficiente tanto para regresiones como clasificaciones.
- Facilita la obtención de la importancia de las características: la evaluación de la importancia o contribución de las variables al modelo. Hay algunas formas de evaluar la importancia de las características. La importancia de Gini y la disminución media de impurezas (MDI) se utilizan generalmente para medir cuánto disminuye la precisión del modelo cuando se excluye una variable determinada.

Desventajas:

- Puede tardar: Para grandes conjuntos de datos puede ser relativamente lento.
- Requiere de recursos: Para conjuntos de datos grandes puede requerir de recursos para almacenarlos.
- Relativamente opaco: La predicción de un árbol pouede ser mucho más simple de interpretar que la de un bosque

Se utiliza entre otros usos para:

- Comercio electrónico
- Finanzas
- Cuidado de la salud

Máquinas de vectores de soporte (SVM)

Si bien esta técnica se ha demostrado no ser adecuada para su utilización en este trabajo, al requerir de un tiempo de entrenamiento excesivo, se explicará al haber sido una de las pruebas realizadas.

Una máquina se vectores de soporte [5], en adelante SVM siendo estas las siglas de *Support Vector Machine*, es un algoritmo de aprendizaje automático supervisado que clasifica los datos encontrados en una línea o hiperplano óptimo que maximice la distancia entre cada clase en un espacio N-dimensional. Esto significa que clasifican encontrando lo que diferencia a unas clases de otras dentro de un espacio vectorial en el que se representan las características de unas instancias etiquetadas.

Estos se suelen utilizar en problemas de clasificación. La cantidad de características del modelo marca la dimensionalidad del mismo y por tanto de los propios planos que genera como separación.

Este es capaz de clasificar tanto de manera lineal como no lineal, pero en caso de separar linealmente se suelen utilizar funciones kernel para transformar los datos a un espacio de mayor dimensionalidad para permitir la separación lineal. Se pueden elegir distintos tipos de kernel.

Hay 3 tipos de SVM:

SVM Lineales

Estos utilizan datos linealmente separables, por lo que no se necesitan transformaciones para separarlos en diferentes clases. El limite de decisión ha de tener una separación máxima con las clases.

Hay dos enfoques para calcular este margen, con un margen duro y con un margen blando, diferenciándose por que en la primera opción todos los puntos están separados por el vector, mientra que en la segunda hay un margen de holgura.

SVM no lineales

Normalmente los escenarios del mundo real no son linealmente separables, para poder separarlos se aplica preprocesamiento de datos mediante funciones de kernel para transformarlos en un espacio de características de mayor dimensión. Estas funciones se conocen como "kernel trick". Las hay de diversos tipos como pueden ser kernels lineales, kernels polinomiales, kernels*Radial Basis function*(RBF) o kernels sigmoides. Esto aumenta el riesgo de sobreajuste.

Regresión de vectores de soporte (SVR)

La regresión de vectores de soporte es una extensión de las SVM dedicada a los problemas de regresión. Si las SVM encuentran el vector o plano que separa los datos las SVR tratan de encontrar el vector o plano que siguen los datos.

Redes neuronales

En este apartado se explicarán los modelos que quedan por mentar que siendo estas, las redes neuronales, las redes neuronales convolucionales y finalmente la red U-Net, que es un tipo de red neuronal convolucional.

Una red neuronal [8] es un modelo de machine learning las cuales tienen un fundamento similar a las neuronas biológicas.

Las redes neuronales son un conjunto de neuronas, compuestas por una o varias neuronas en la capa de entrada, una o varias capas ocultas intermedias y finalmente una capa de salida.

El funcionamiento de una neurona es el siguiente:

1. Se asignan una ponderaciones a los distintos valores de entrada de la neurona.
2. Estas ponderaciones se multiplican por el valor de su entrada y se suman
3. La suma se pasa a través de una función de activación dónde se determina la salida de la neurona
4. Si esta salida es superior al valor umbral de la neurona esta se activa y pasa señal a la siguiente capa.

Al entrenar el algoritmo se quiere evaluar su precisión utilizando una función de coste o pérdida, la cual tiene la neurona la misión de minimizar.

El modelo va ajustando su ponderaciones y valores umbral utilizando la función de coste y el aprendizaje reforzado para alcanzar un punto de convergencia.

Las redes neuronales también pueden tener retropropagación, es decir la actualización de los pesos y el umbral viene marcada por las capas posteriores.

Las redes neuronales se entrenan y mejoran su precisión con el tiempo.

Si bien Deep learning y redes neuronales son términos que se usan indistintamente de forma general, pero en verdad solo se consideran deep learning si hay más de 3 capas.

Tipos

Existen distintos tipos de redes neuronales, que se usan para fines diferentes. Aunque aquí se explicaran las más comunes:

- Perceptrón simple:esta es una red neuronal con una capa de 1 neurona de entrada, un salida de 1 neurona y sin capas ocultas o intermedias.
- Redes neuronales de avance: o perceptrones multicapa, se componen de una capa de entrada, una o varias capas ocultas y una capa de salida.
- Redes neuronales recurrentes: Tienen bucles de retroalimentación. Estas se emplean datos de series temporales para hacer predicciones futuras.

- redes neuronales convolucionales : Estás se definirá en más profundidad en el próximo apartado. don redes neuronales hacia adelante (feedforward) y se utilizan para reconocimiento de imágenes Estas utilizan la multiplicación de matrices y los principios del álgebra lineal.

Redes neuronales convolucionales

Como ya se ha mentado, son un subtipo de redes neuronales [9], usualmente utilizadas para tareas de clasificación y reconocimiento de imágenes. Son un enfoque escalable para tareas de clasificación de imágenes y reconocimiento de objetos. Estas pueden requerir altos recursos computacionales.

Estas tienen tres tipos de capas:

- Capa convolucional
- Capa de agrupación
- Capa totalmente conectada

La red siempre empieza por una capa de convolución y acaba por una capa totalmente conectada, aunque después de la primera capa de convolución pueden ir otras capas convolucionales o capas de agrupación. Cada capa del modelo aumenta la complejidad de la red, identificando partes cada vez más grandes. Las primeras capas se centran en características simples como bordes y colores.

Capa convolucional

Esta es el bloque de creación principal de la red y se realizan en ella la mayoría de los cálculos. Requiere de datos de entrada, un filtro y un mapa de características. También hay un detector de característica, el filtro, el cual se encarga de comprobar si esa característica está presente en los datos de entrada.

El detector de características es una matriz, normalmente bidimensional de 3x3, esto determina el tamaño del campo receptivo. Después se aplica este filtro a un área de la imagen calculando el producto escalar entre el filtro y los datos de entrada. Esto va a la matriz de salida y se repite el proceso sobre toda la imagen. La capa resultante se denomina mapa de características, mapa de activación o característica convolucionada. Se puede ver en la figura 3.1.

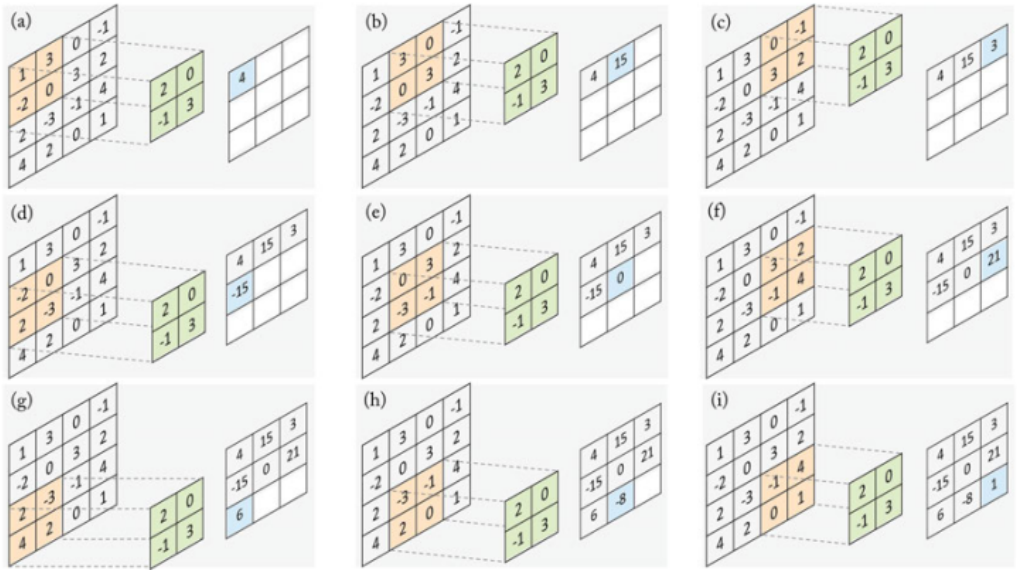

Figura 3.1: Ejemplo aplicación de un filtro [17].

Los pesos del filtro permanecen fijos a lo largo de la misma ejecución del algoritmo anterior aunque durante el entrenamiento se van ajustando a través de un proceso de retropropagación y descenso de gradiente.

Hay tres hiperparámetros que afectan al volumen de salida:

- Número de filtros: afecta a la profundida de la salida, ya que cada filtro produce su mapa de características.
- Stride: es la distancia, o el número de píxeles, que el kernel mueve sobre la matriz de entrada.
- zero-padding: Se utiliza para ajustar los filtros al tamaño de la imagen de entrada. pone a 0 todos los elemnetos que quedan fuera de la matriz de entrada. Hay tres tipos:

  Valid padding: O no padding, en este caso la última convolución se descarta si las dimensiones no se alinean.

  Same padding: Este garantiza que la capa de salida tenga el mismo tamaño que la de entrada. Es el utilizado en este trabajo.

  Full padding: Este aumenta el tamaño de la capa de salida añadiendo ceros en el borde de la entrada.

Después de cada convolución la red aplica una transformación de unidad lineal rectificada (ReLU) al mapa de características introduciendo no linealidad en el modelo. Esto eliminará los números negativos de este. Después de una primera capa convolucional puede ir otra como ya se ha comentado.

Capa de agrupación

La agrupación de capas o submuestreo, permite reducir la dimensión mediante la reducción de los parámetros de la entrada. Este aplica una función de agregación a los valores dentro del campo receptivo y llena la matriz de salida.

Hay dos tipos de agrupación:

- Agrupación máxima: a medida que recorre la entrada manda el pixel con el valor mayor a la matriz de salida.
- Agrupación media: a medida que recorre la entrada manda el valor medio de los pixeles dentro del campo receptivo a la matriz de salida

Aunque se pierde mucha información en la capa de agrupación esto ayuda a reducir la complejidad, mejora la eficiencia y limita el riesgo de sobreajuste.

Capa totalmente conectada

Esta capa realiza la predicción en base a los mapas de características de las capas anteriores contra la imágen original. Suele usar una función softmax que clasifica valores de entre 0 y 1. En el caso de este experimento se utiliza una funcion sigmoid al ser una clasificación binaria.

Red convolucional U-Net

En este apartado explicaremos el funcionamiento de la red U-Net, un tipo de red convolucional muy utilizado para análisis de imágenes, el cual ha sido implementado en este experimento. Este es una mejora a otras redes convolucionales aumentando la precisión y pudiendo hacer clasificaciones más ajustadas.

Para comprender correctamente la arquitectura de la red U-Net [19] es recomendable ver la figura 3.2.

La red U-Net es una red neuronal convolucional qeu se divide en dos partes, el camino de expansión y el camino de compresión. La red empieza aplicando dos capas convolucionales a la imágen, separando en 64 mapas de características las imágenes y pasando a la siguiente capa estas además de pasandolo a la capa reflejada en el camino de expansión. La siguiente capa es una capa de agrupación, la cual se encarga de reducir su dimensionalidad. este proceso se repite en todo el camino de contracción. Una vez acaba el proceso de contracción se hace el proceso inverso para el proceso de expansión, sumando los mapas de características que se han obtenido en el camino de contracción.

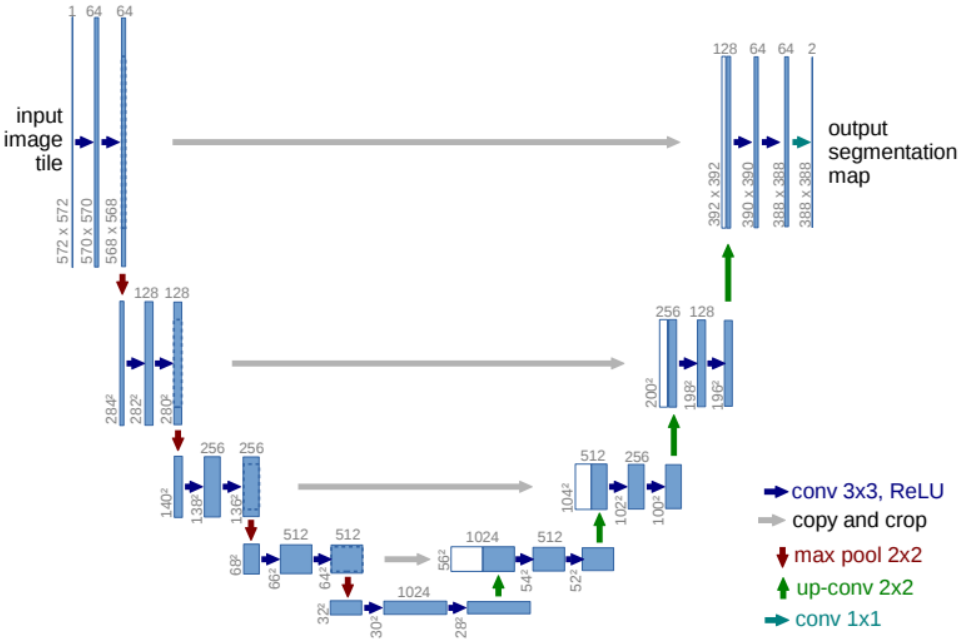

Figura 3.2: Funcionamiento red U-net [19].

Capítulo 4

Técnicas y herramientas

En este apartado se explicarán el grueso de las herramientas utilizadas para este proyecto.

4.1. Python

El lenguaje de programación que se ha utilizado en este proyecto ha sido Python. Este es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "Monty Python" [16]. Su sintaxis se centra en la legibilidad y la limpieza de código.

El lenguaje tiene las siguientes características:

- **Lenguaje interpretado**: Esto significa que para su ejecución requiere otro programa llamado interprete, en lugar de compilar el código a lenguaje máquina y ejecutarlo directamente. Esto lo convierte en más flexible y portable. Aunque este tiene muchas de las características de los lenguajes compilados.
- **Tipado dinámico**: Esto se refiere a que no es necesario declarar los tipos de las variables de manera explícita. Estos tipo se determinarán en tiempo de ejecución. El tipo de las variables puede cambiar con nuevas asignaciones.
- **Fuertemente tipado**: Esto significa que no se puede tratar a una variable como si tuviera un tipo distinto del que tiene, han de hacerse las transformaciones de forma explícita.

- **Multiplataforma**: Este es capaz de correrse en una gran cantidad de sistemas sin cambios significativos.
- **Orientado a objetos**: La orientación a objetos es un paradigma de programación en el cual se abstraen los conceptos del mundo real como clases y objetos. Aunque python permite también la programación imperativa y la orientada a aspectos.

Python no es adecuado para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.

El principal motivo para la selección de este lenguaje es las librerías que hay desarrolladas para el, como Scikit-learn 4.5, NumPy 4.4 o Matplotlib 4.3.

4.2. Jupyter Notebook

Jupyter Notebook [3] es una aplicación cliente-servidor que permite crear, compartir y visualizar archivos en formato JSON que siguen un esquema de celdas de entrada y salida.

Las celdas contienen textos, código, fórmulas matemáticas e incluso contenido multimedia. Los archivos se guqardan en formato .ipynb.

Estos permite compartir códigos, ejecutarlos de manera sencilla, proporcionar una interfaz cómoda para albergar las visualizaciones además de intercalar textos explicativos con el código.

Jupyter Notebooks tiene un conjunto de núcleos y el Dashboard. Cada núcleo o kernel es el que se encarga de recibir las peticiones, procesar las solicitudes y devolver las respuestas. El kernel por defecto es IPyhton, el cual interpreta comandos de Python, pero se pueden instalar otros tipos de núcleos que permitan trabajar con otros lenguajes como C++, R, Java o Scala entre otros.

Es de uso gratuito.

4.3. Matplotlib

Matplotlib [12] es una librería open source de Python que permite visualizaciones de datos. La visualización de datos es una parte clave del análisis de datos, en este proyecto concreto se utiliza para la visualización de las imágenes tanto obtenidas de los datos como generadas por el programa.

Se creó en 2002 por Jhon Hunter como una alternativa a MATLAB para visualizar gráficas. La comunidad lo ha mejorado a lo largo del tiempo.

Permite generar trazados, histogramas y gráficas de gran calidad.

4.4. NumPy

NumPy [11] es un paquete de Python que incluye operaciones complejas como operaciones con vectores de N dimensiones, matrices, integrales, ecuaciones diferenciales , estadísticas y más. Python tiene implementadas por defecto algunas funciones matemáticas pero no adecuadas para matrices y vectores. NumPy permite el uso eficiente de Python para propósitos científicos.

NumPy se especializa en el procesado de vectores multidimensionales, en los que se permiten operaciones elemento a elemento. Se puede usar álgebra lineal si es necesario sin modificar previamente los vectores. Se pueden redimensionar las matrices de manera dinámica. Esto es más rápido y eficiente que en otros lenguajes ya que evita el tener que crear nuevos vectores.

Los vectores de NumPy permiten incrustar código en C/C++/Fortran, lo que aumenta sobremanera su eficiencia.Una operación con ndarray es 25 más rápido que un bucle de python. Los ndarrays solo pueden almacenar un tipo de dato por columna.

4.5. Scikit-learn

Scikit-learn es un modulo de Python que integra un amplio rango de algoritmos para problemas de aprendizaje tanto supervisado como no supervisado. Este se enfoca como un paquete que acerca las técnicas de *machine learning* a personas no especializadas en el campo, de forma general en un lenguaje de alto nivel. Usa una interfaz orientada a las tareas.

Tiene dependencias externas mínimas y se distribuye con una licencia de BDS simplificada, es decir como un software libre.

También incorpora código compilado y librerías en C++ para aumentar la eficiencia. Para reducir las barreras de entrada reducen al mínimo los objetos propios y trata de utilizar los objetos de NumPy para almacenar datos.

Proporciona una documentación muy completa con tutoriales varios.

4.6. TensorFlow y Keras

TensorFlow [21] [10] es uno de los frameworks de inteligencia artificial más conocidos y utilizados. fue desarrollado originalmente por el grupo de Google Brain. Fue diseñado para facilitar la investigación en *machine learning* y realizar más rápida la transición de un prototipo a un sistema de producción.

Keras es una API de alto nivel para redes neuronales. El código se especifica en Pyhton y es capaz de ejecutarse en tres entornos : TensorFlow, CNTK o Theano. Se puede cambiar de motor de ejecución sin alterar el código. TensorFlow ha decidido adoptar Keras como su API principal. Los modelos de Keras se pueden crear mediante APIs secuenciales de 'tf.keras.Sequential' o mediante la API funcional de Keras 'tf.keras.Model'.

TensorFlow proporciona un grupo de paquetes relacionados con la creación de redes neuronales. Estos facilitan la creación y personalización de las capas (tf.keras.layers), datasets (tf.data), métricas (tf.keras.metrics), funciones de perdida (tf.keras.losses) y columnas de características (tf.feature_column).

Capítulo 5

Aspectos relevantes del desarrollo

En este apartado se explicarán los experimetos realizados y las conclusiones obtenidas a lo largo de estos.

5.1. Obtención de los datos de prueba

Como se ha comentado previamente los datos se obtienen del proyecto *"AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photography"* [18], concretamente del repositorio de datos del proyecto.

Como podemos ver se organizan en varias 3 carpetas:

- Orthophotos: Carpeta dónde aparecen las Ortofotos del proyecto. Estas no son utilizadas en este proyecto.
- Sampling: Carpeta dónde se almacenan los csv que relacionan las distintas imágenes con su respectiva url en el ftp sobre el repositorio, tanto la imagen en formato .nc como las máscaras de capa .tif. Esta carpeta tampoco se ha utilizado ya que las imágenes se han descargado para las pruebas en lugar de hacer una carga dinámica.
- Sentinel2: Carpeta que guarda tanto las imágenes como las máscaras además de archivos csv que muestran cuantos píxeles de cada tipo tienen los resultados.

Por la cantidad de datos y los recursos de los que dispone el sistema se ha decidido trabajar solo con un subconjunto de los datos, concretamente los correspondientes a 128 imágenes de España.

5.2. Visualización de los datos y composición de la imagen

Lo primero que se realizó fue una exploración de los datos, para comprender mejor estos y ver sus estructuras, para eso se extrajeron las bandas 4, 3 y 2, correspondientes a la banda Roja, Verde y Azul, se visualizaron en como podemos ver en la figura 5.1 y posteriormente se compusieron en una imágen RGB como podemos ver en la figura 5.2.

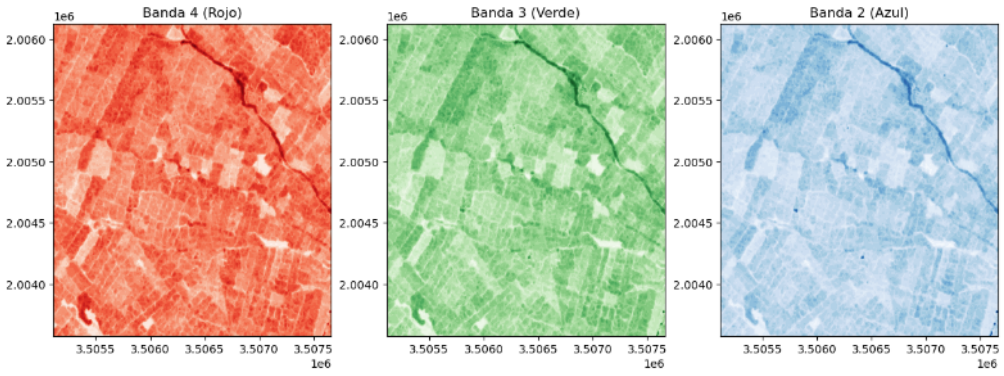

Figura 5.1: Visualización de las Bandas RGB separadas.

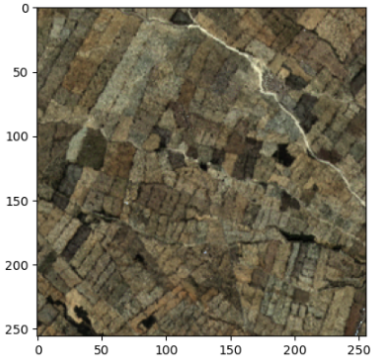

Figura 5.2: Visualización de la imagen compuesta con las bandas RGB.

Una vez visualizada la imagen se comprobó cómo son las máscaras de capa, estas tienen 4 capas distintas la máscara de capa, una mascara de límites,una mascara de distancia y una enumeración de campos. Se ha decidido entrenar los modelos con la primera opción. Como podemos ver en la figura 5.3.

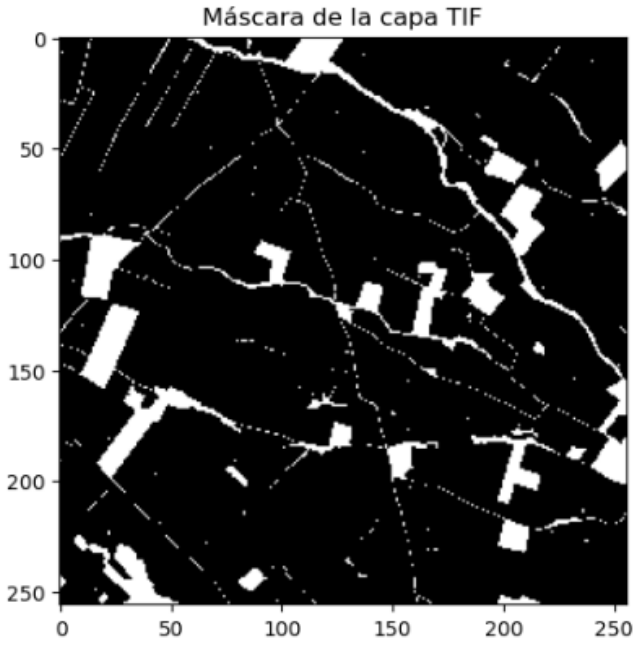

Figura 5.3: Visualización de la máscara de capa de las delimitaciones.

Tras eso se superpuso la capa con la imagen para comprobar cuánto se adecua con la imagen. Como se puede ver en la figura 5.4 registran los grandes límites.

5.3. Modelos sin contexto

En este apartado intentaremos entrenar tres modelos: KNN, SVM y RandomForest. Estos no tienen en cuenta el contexto de los píxeles, predicen unicamente en base a la información de cada píxel por lo que se asumía, cosa que se comprobó que obtienen unos resultados peores.

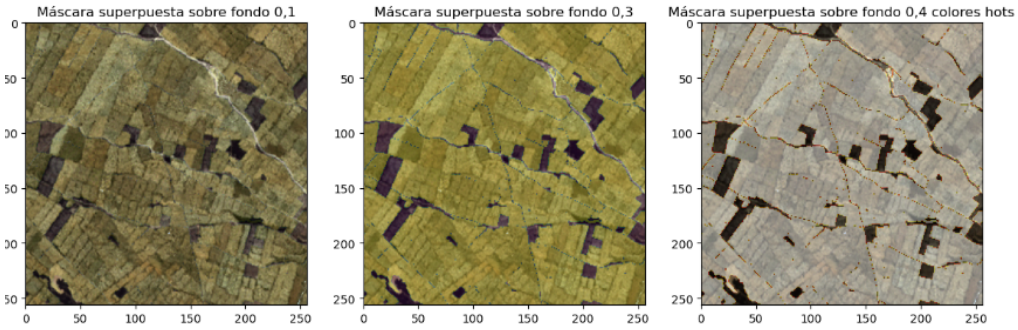

Figura 5.4: Visualización de la máscara de capa superpuesta con la imagen.

KNN

Se empezó por KNN, este tuvo un tiempo de entrenamiento de 57.88 segundos. Se hizo una predicción sobre la imagen del inicio para poder visualizar el resultado. Los resultados son la figura 5.5.

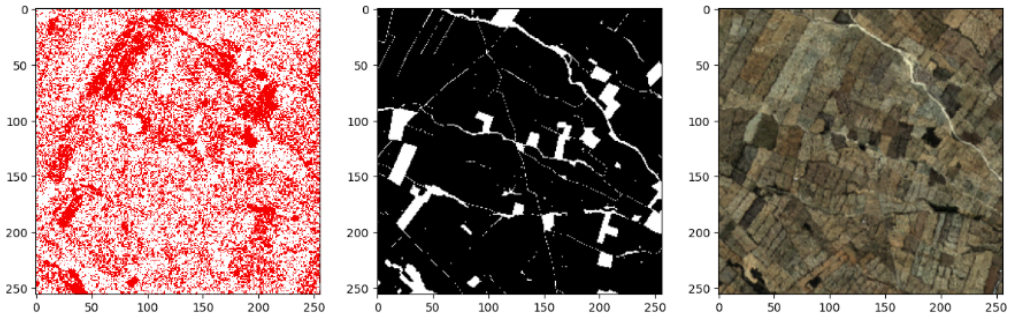

Figura 5.5: Visualización de la predicción sobre la imagen por el modelo KNN comparada con el resultado previsto y la imagen orginal.

```
Accuracy: 0.628
Classification Report:
              precision    recall  f1-score   support

         0.0       0.16      0.64      0.26      6723
         1.0       0.94      0.63      0.75     58813

    accuracy                           0.63     65536
   macro avg       0.55      0.64      0.51     65536
weighted avg       0.86      0.63      0.70     65536

Confusion Matrix:
[[ 4336  2387]
 [21995 36818]]
```

Figura 5.6: Métricas obtenidas para el modelo KNN.

Las métricas obtenidas por el modelo han sido las mostradas en la figura 5.6. Si se analizan los resultados podemos ver que si bien reconoce algunas estructuras y se puede llegar a intuir la imagen en la predicción, hay mucho ruido en la imagen y probablemente el hecho de que haya obtenido unos resultados es derivado de que es un conjunto de datos relativamente desbalanceado.

SVM

El siguiente modelo a probar es SVM, el cual no se pudo llegar a entrenar ya que el tiempo que requirió para entrenarse era claramente excesivo superando las 16 horas antes de que se decidiera pararlo.

Random Forest

Finalmente Se probó Random Forest, el cual se lanzo con todos los núcleos posibles en paralelo, de cara a mejorar su eficiencia. igualmente tuvo un tiempo de entrenamiento de 1273.35 segundos. Se pueden ver los resultados de la predicción en la figura 5.7 y las métricas 5.8.

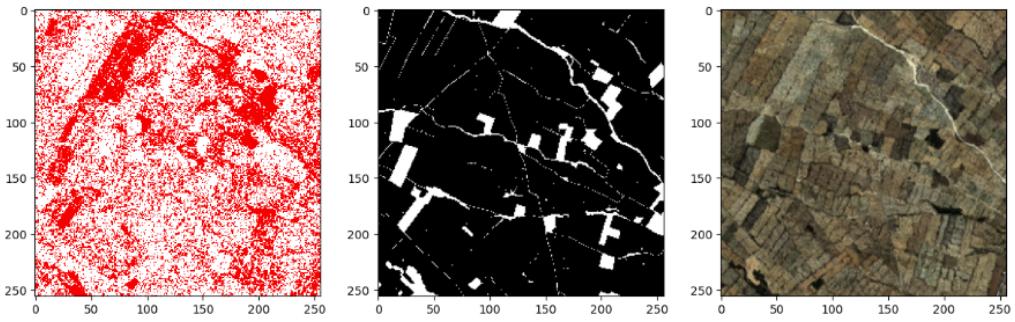

Figura 5.7: Visualización de la predicción sobre la imagen por el modelo Random Forest comparada con el resultado previsto y la imagen orginal.

```
Accuracy: 0.632
Classification Report:
              precision    recall  f1-score   support

         0.0       0.17      0.66      0.27      6723
         1.0       0.94      0.63      0.75     58813

    accuracy                           0.63     65536
   macro avg       0.55      0.64      0.51     65536
weighted avg       0.86      0.63      0.70     65536

Confusion Matrix:
[[ 4411  2312]
 [21800 37013]]
```

Figura 5.8: Métricas obtenidas para el modelo Random Forest.

5.4. Modelo U-Net

Finalmente el grueso de las pruebas se hicieron son las relacionadas con la red convolucional de tipo U-Net. Esta como ya se ha mentado tiene contexto de los píxeles alrededor del que se está analizando. Esto de manera teórica se supone que obtendrá mejores resultados, cosa que se ha probado.

Se empezó por implementar la red neuronal, siguiendo los pasos del paper [19] aunque se implementó una segunda versión en la cual hay Dropout en tre las convoluciones de una misma capa, para evitar el sobreajuste.

En todas las opciones se escogió como optimizador adam, como métricas accuracy y como función de perdida binary_crossentropy. Estos han tenido un tiempo de entrenamiento por época de entre 250 y 300 segundos. Marcando un accuracy según las métricas de Keras de entre 0.75 y 0.86 siendo a partir de 10 épocas normalmente superiores a 0.8.

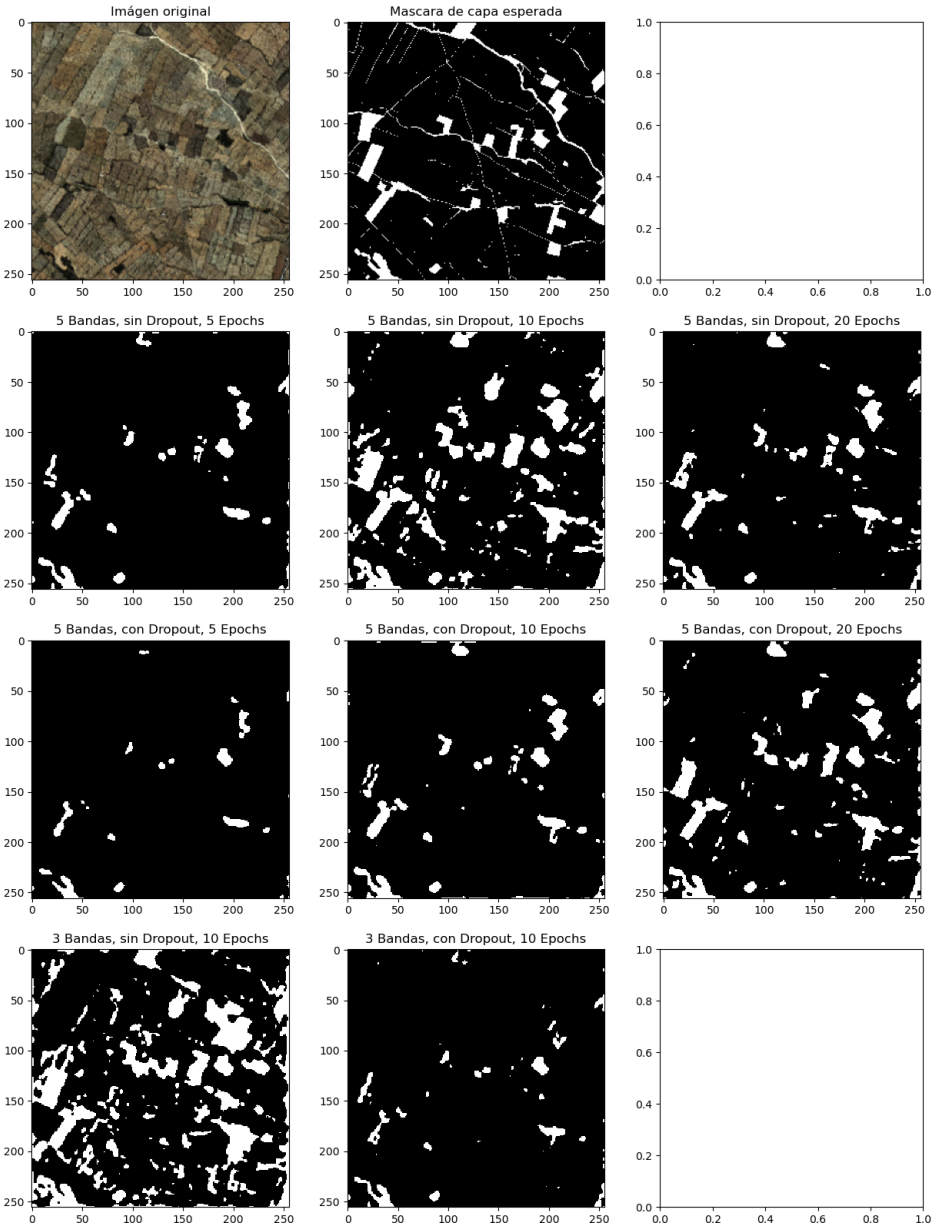

Figura 5.9: Predicciones realizadas por los modelos descritos.

De cara a buscar cual es la mejor configuración posible se probaron en un inicio 8 modelos, los cuales se hizo una predicción sobre la imagen previa para visualmente analizar cual ha sido más adecuado. Se muestra en la figura 5.9.

Finalmente viendo los resultados dados se decidió entrena un modelo más esta vez con 40 épocas y con dropout, que será el modelo a utilizar. Los resultados de este lo podemos ver en la figura 5.10.

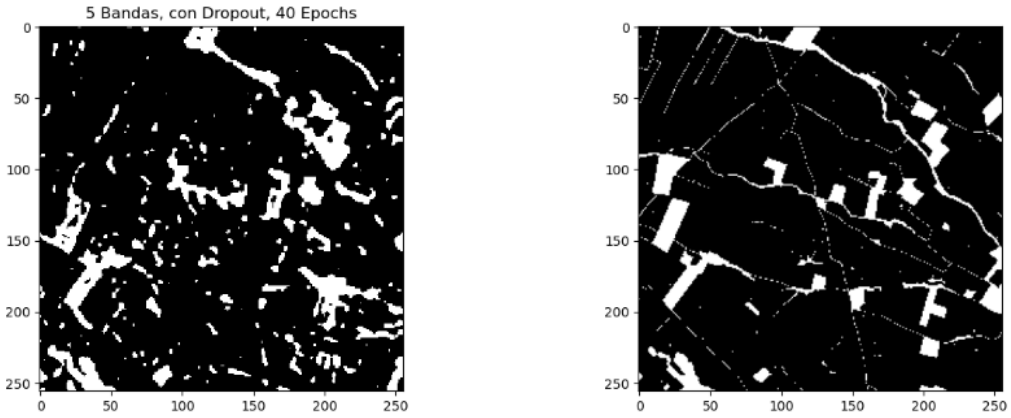

Figura 5.10: Predicción realizada por el modelo final.

Es reseñable que si bien este muestra esa predicción con un umbral de predicción de 0.5, entendiendo que este es el punto a partir del cual en la predicción se considera 1. La predicción sin hacer ese filtrado por umbral muestra una imagen muy interesante que puede ser más interpretable como se puede ver en la figura 5.11.

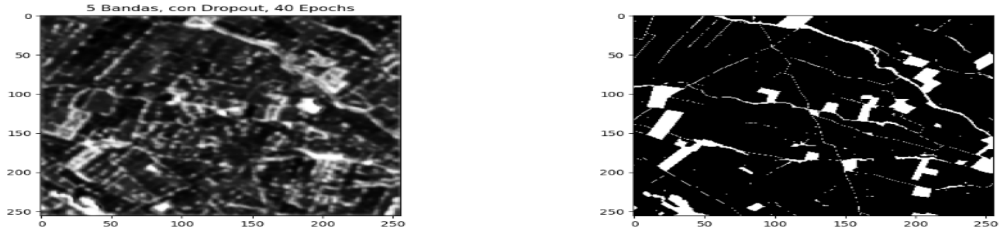

Figura 5.11: Predicción realizada por el modelo final sin binarizar.

Finalmente se evaluaron los modelos y el que mejores métricas consiguió fue el último, tal y como se había predicho. Este tienes las métricas mostradas en la siguiente figura 5.12.

```
Accuracy: 0.758
Reporte de Clasificación:
              precision    recall  f1-score   support

         0.0       0.73      0.67      0.70    879796
         1.0       0.78      0.82      0.80   1217356

    accuracy                           0.76   2097152
   macro avg       0.75      0.75      0.75   2097152
weighted avg       0.76      0.76      0.76   2097152
```

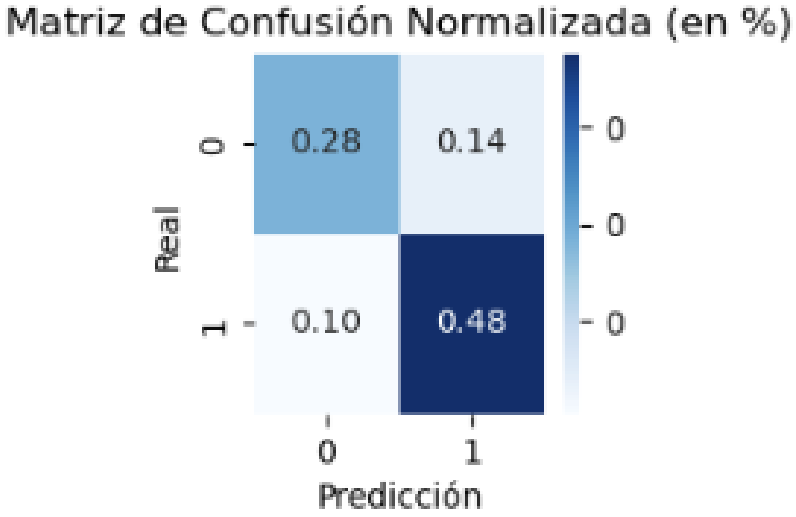

Figura 5.12: Métricas del modelo U-Net con 40 Epochs con dropout.

Analizando estos resultados se puede reseñar que se han obtenido un modelo suficientemente adecuado, aunque puede llegar a ser mejorable.

Capítulo 6

Trabajos relacionados

En este apartado se comentarán algunos de los trabajos relacionados y experimentos similares.

6.1. AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photograph [18]

Este es el artículo sobre el que se basa este trabajo, ya que es el que proporciona los materiales para el entrenamiento de los modelos sin los cuales esta tarea hubiera muchísimo más ardua y probablemente de una calidad muy inferior.

El artículo presenta AI4Boundaries, un conjunto de datos abierto y preparado para el aprendizaje automático que tiene como objetivo facilitar la detección de límites de campos agrícolas utilizando imágenes de satélite Sentinel-2 y fotografías aéreas.

- El conjunto de datos incluye dos componentes principales: Una composición mensual de imágenes Sentinel-2 a una resolución de 10 metros, lo que permite el análisis a gran escala.
- Ortofotografías aéreas a una resolución de 1 metro, lo que posibilita el análisis a escala regional.

Los datos de etiquetado de los límites de campos provienen de fuentes públicas, como el Sistema Integrado de Gestión y Control (SIGC) de varios

países europeos, como Austria, Cataluña, Francia, Luxemburgo, Países Bajos, Eslovenia y Suecia.

El conjunto de datos ha sido diseñado para ser utilizado en el entrenamiento de modelos de aprendizaje automático que puedan realizar la delineación automática de los límites de campos agrícolas.

6.2. U-Net: Convolutional Networks for Biomedical Image Segmentation [19]

Este es el artículo que da la arquitectura del modelo principal de este trabajo, la red convolucional de tipo U-Net. Esta está diseñada para la segmentación de imágenes biomédicas. Se basa en la red convolucional completamente conectada (FCN) y se modifica para funcionar con pocos datos de entrenamiento y producir segmentaciones más precisas.

Este artículo trata de resolver el problema de que el entrenamiento de redes profundas requiere generalmente miles de muestras de entrenamiento etiquetadas. Sin embargo, en tareas biomédicas, como la segmentación de imágenes, es difícil obtener tantos datos etiquetados.

Esto lo soluciona a través de las de dos fases:

- Contracción: Un camino de contracción que captura el contexto mediante la aplicación repetida de convoluciones 3x3, seguidas de ReLU introduciendo no linealidad al modelo y operaciones de pooling máximo de 2x2 con un salto de 2 para la reducción de la información espacial y el aumento de la información de características. A medida que el modelo pasa por estas fases le pasa el mapa de características resultante a la capa correspondiente en la segunda fase, siendo esta la capa de su mismo número de mapas de características.

- Expansión: Un camino de expansión que combina la información de características y espacial a través de la concatenación de características de alta resolución de la ruta de contracción con las características de alta resolución de la ruta de expansión. En esta fase se aplican las mismas que en la anterior pero en orden inverso.

La U-Net representa una herramienta valiosa para la comunidad científica y los profesionales del sector biomédico, al proporcionar una arquitectura de

red que puede ser entrenada de manera eficiente con pocos datos y producir segmentaciones precisas y rápidas.

Este método superó a los métodos anteriores tanto en velocidad como en precisión.

Capítulo 7

Conclusiones y lineas de trabajo futuras

En este apartado se hará una rendición de cuentas respecto a los objetivos marcados, se explicarán las conclusiones que se han extraído y finalmente se comentarán posibles líneas de trabajo futuras.

7.1. Conclusiones

Las conclusiones que podemos obtener de este proyecto son las siguientes:

- Las redes neuronales convolucionales son una herramienta muy útil para la segmentación de imágenes. Si bien hay otros modelos capcaes de realizar la tarea de manera aproximada este es el mejor de los probados.
- Existen técnicas automáticas para la separación automática de parcelas capaces de realizar la tarea satisfactoriamente.

7.2. Rendición de cuentas

Se expondrán en esta sección los objetivos marcados en el episodio previo y se comentará el estado de los mismos.

- Entrenar un modelo capaz de realizar una separación suficiente de parcelas en base a una imagen satelital.

Se da por cumplido este objetivo, si bien es relativo que se entiende por suficiente, se consideran suficientemente buenos los resultados obtenidos.

- Probar diferentes métodos y configuraciones de cara a comprobar si realmente la red neuronal convolucional es el método más potente o pueden ser más interesantes otras metodologías que impliquen que no tener en cuenta el contexto de los píxeles.

  Con respecto a este apartado se puede considerar como comprobado, analizando las métricas de los modelos explorados podemos ver que las mejores métricas obtenidas han sido las de los modelos predichos. Además se han probado diversos modelos, si bien es cierto que SVM no ha sido posible el tiempo de entrenamiento excesivo ya lo hubiera descartado como viable.

- Exponer los conocimientos extraídos en un notebook de Python, lo cuál facilitará la trasmisión y la manipulación de estos.

  El Notebook ha sido realizado con éxito.

- Generar un repositorio dónde almacenar dicho notebook con la estructura definida para una poder ejecutar de forma simple el cuaderno.

  El repositorio ha sido creado, en el apartado 5.1 se explica como han de almacenarse los archivos y la URL del repositorio es: Repositorio

- Tratar de optimizar en la medida de lo posible los parámetros del modelo para maximizar su eficiencia.

  Si bien es cierto que pudieran obtenerse mejores resultados, se consideran suficientes las pruebas realizadas, ahora en este punto es en el que más se puede mejorar, cosa de la que hablaremos en el apartado de Líneas de trabajo futuras 7.3.

7.3. Líneas de trabajo futuras

Respecto a las posibles mejoras y líneas de trabajo futuras:

- Probar a entrenar el modelo con un número mayor de imágenes y una mayor variedad.

  Si bien es cierto que los resultados son suficientes con el número de imágenes reducido que se han utilizado, es posible que al usar una mayor parte del Dataset los resultados puedan ser mejores.

- Probar más parámetros.

  Dentro de los parámetros que se pueden manipular para cambiar los resultados en este trabajo sobre todo se han probado las épocas, el número de bandas y la posibilidad de hacer dropout o no, pero como hemos visto en el apartado 3.3 hay muchos hiperparámetros que se pueden modificar además de alguno más derivado de la definición del modelo o incluso el umbral de corte.

- Hacer un programa que sea capaz de analizar imágenes.

  Puede ser interesante hacer un programa que sea capaz de dada una imagen satélite haga la separación. También se podrían automatizar las descargas de los documentos del ftp para no requerir de almacenarlos en memoria.

Apéndices

Apéndice A

Plan de Proyecto Software

En este apéndice se describirá la planificación que se ha seguido a la hora de organizar el proyecto, tanto desde la perspectiva temporal y cómo se ha distribuido el trabajo como desde un estudio de la viabilidad del mismo.

A.1. Planificación temporal

La organización temporal de este trabajo se ha realizado mediante Sprints de una duración variable entre 1 y 2 semanas, en función de la carga de trabajo de cada uno de ellos.

Sprint 1: 16/04/2024 - 22/04/2024

Este se dedico a la lectura de los artículos científicos y la interiorización de los contenidos expuestos.

Sprint 2: 23/04/2024 - 06/05/2024

En este se Inicio un proceso de familiarización con los datos, tratando de visualizarlos y comprender sus estructuras internas.

Sprint 3: 07/05/2024 - 20/05/2024

En este Sprint se inicio el Notebook y se aplicaron los conocimientos de la fase anterior, componiendo las imágenes y capas.

Sprint 4: 21/05/2024 - 03/06/2024

Este sprint fue dedicado a la creación y entrenamiento de los modelos sin contexto de los pixeles adyacente, es decir KNN, SVM y Random Forest. Además de medir sus métricas.

Sprint 5: 04/06/2024 - 17/06/2024

Este sprint fue dedicado al desarrollo del modelo de la red U-Net.

Sprint 6: 18/06/2024 - 01/07/2024

Este fue dedicado al entrenamiento y visualización de las primeras predicciones.

Sprint 7: 02/07/2024 - 09/07/2024

Este Sprint finalmente fue dedicado a la evaluación de los modelos de la red U-Net.

A.2. Estudio de viabilidad

En este apartado se va a comentar la viabilidad del proyecto tanto de forma económica, calculando el coste total que debería de haber costado el desarrollo del proyecto, y la viabilidad legal de las librerías y herramientas utilizadas.

Viabilidad económica

En este subapartado se encuentran los cálculos económicos del proyecto general. Estos son los derivados del equipo y personal necesarios para la realización del mismo ya que el resto de los materiales son de libre acceso.

Para realizar el cálculo se han divido los gastos en:

- **Coste personal** (tabla A.1): contratación del personal de investigación y desarrollo del proyecto.
- **Coste *hardware*** (tabla A.2): dispositivos *hardware* necesarios en el proyecto.

Concepto	**Coste (€)**
Salario mensual bruto	2.047,78
Seguridad Social (30,04 %)	615,15
Retención IRPF (2 %)	28,65
Salario mensual neto	1.403,97
Total 3 meses	6.143,34

Tabla A.1: Costes de personal.

Concepto	**Coste (€)**
Ordenador de desarrollo (x1)	50
Total	50

Tabla A.2: Costes de *hardware*.

Tipo	**Coste (€)**
Personal	6.143,34
Hardware	50
Total	6.193,34

Tabla A.3: Costes final.

Teniendo en cuenta un precio de 1000€ para el ordenador, con una vida media de 5 años y se va a usar durante 3 meses:

Finalmente se ha realizado un cálculo final del coste del proyecto, tabla A.3.

Viabilidad legal

En este subapartado se van a comentar las distintas licencias que tienen las herramientas y librerías utilizadas en el proyecto, así como se comenta la licencia final que tiene el proyecto.

Las licencias de las librerías y herramientas utilizadas en el desarrollo del proyecto se pueden ver en la tabla A.4.

Teniendo en cuenta estas licencias, las licencias de las herramientas usadas tanto en la aplicación final de rehabilitación y las licencias de las

Librería-Herramienta	**Licencia**
Numpy	BSD 3
Matplotlib	PSF
Seaborn	BSD 3
Python	PSF
Scikit-learn	BSD 3
Tensorflow	Apache 2.0

Tabla A.4: Tabla con las licencias de las librerías y herramientas utilizadas.

herramientas y librerías usadas por el compañero se ha decido utilizar la licencia *GPL v3.0* a partir de las licencias más restrictivas.

Con esta licencia *GPL v3.0* se puede utilizar el *software* desarrollado para uso comercial, se puede modificar las implementaciones realizadas, distribuirlas, realizar patentes sobre ellas y usarlas de forma privada.

Apéndice B

Manual de programador

En este manual se indicarán los pasos a seguir para la ejecución del Notebook.

Lo primero que se ha de hacer tener instalado o instalar Python, preferentemente en su versión 3.11.5 que es en la que se ha desarrollado, aunque seguramente no den problemas versiones más nuevas.

Tras lo cual será necesario instalar Jupyter Notebook. La forma sencilla sería hacer, la guía propia de Jupyter se encuentra en el siguiente enlace: pip install jupyterlab

Una vez instalado se debe iniciar mediante el comando jupyter notebook, al ejecutarlo se proveerá de un enlace para navegar desde el programa.

Se ha de clonar el repositorio de del proyecto "wgm1001 / Trabajo_fin_master_deteccion_bordes" de gitHub o bien descargarlo.

Como se ha comentado previamente los datos se obtienen del proyecto *"AI4Boundaries: an open AI-ready dataset to map field boundaries with Sentinel-2 and aerial photography"* [18], concretamente del repositorio de datos del proyecto.

Para una ejecución similar a la del proyecto y sin tiempos excesivos se recomienda escoger solo los datos que inician por ES.

Una vez descargados Las imágenes .nc de entrenamiento las almacenamos en la carpeta "/data/trainES/", las de test en la carpeta "/data/test/" y las máscaras de capa de los resultados *ground truth* en la carpeta "/data/-maskES/" para las de los datos de entrenamiento y "/data/maskTestEs/" para el test.

También el archivo “ES_126_S2_10m_256” tanto .nc como .tif en la carpeta “data” ya que es el que se ha utilizado para todas las visualizaciones.

Una vez están todos los archivos en su correspondiente localización, desde el navegaro en la aplicación de Jupyter se ha de navegar hasta en archivo ipynb. Ya se pueden correr de manera secuencial todas las celdas menos la del SVM. La primera vez que se ejecuten se han de entrenar los modelos. en posteriores ejecuciones con ejecutar las celdas en las que se cargan debiera bastar para usarlos. No se pueden subir al repositorio los modelos entrenados por su tamaños excesivo.

Bibliografía

[1] El modelo RGB, algunas definiciones básicas | Formación Gráfica | Asesorías y talleres personalizados de diseño — formaciongrafica.net. https://www.formaciongrafica.net/blog/tutoriales-online/el-modelo-rgb-algunas-definiciones-basicas/. [Accessed 11-07-2024].

[2] eurodatacube.com. https://eurodatacube.com/marketplace/data-products/lpis. [Accessed 09-07-2024].

[3] Jupyter Notebook: documentos web para análisis de datos, código en vivo y mucho más — ionos.es. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/. [Accessed 15-07-2024].

[4] What Is Random Forest? | IBM — ibm.com. https://www.ibm.com/topics/random-forest. [Accessed 12-07-2024].

[5] What Is Support Vector Machine? | IBM — ibm.com. https://www.ibm.com/topics/support-vector-machine. [Accessed 12-07-2024].

[6] ¿Qué es el machine learning (ML)? | IBM — ibm.com. https://www.ibm.com/es-es/topics/machine-learning. [Accessed 12-07-2024].

[7] ¿Qué es KNN? | IBM — ibm.com. https://www.ibm.com/mx-es/topics/knn. [Accessed 12-07-2024].

[8] ¿Qué es una red neuronal? | IBM — ibm.com. https://www.ibm.com/es-es/topics/neural-networks. [Accessed 12-07-2024].

[9] ¿Qué son las redes neuronales convolucionales? | IBM — ibm.com. https://www.ibm.com/es-es/topics/convolutional-neural-networks. [Accessed 12-07-2024].

[10] Ekaba Bisong.

[11] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[12] Daniel. Matplotlib: todo lo que tienes que saber sobre la librería Python de Dataviz — datascientest.com. https://datascientest.com/es/todo-sobre-matplotlib. [Accessed 15-07-2024].

[13] Roberto Díaz. Métricas de Clasificación - Aprende a EVALUAR tu modelo — themachinelearners.com. https://www.themachinelearners.com/metricas-de-clasificacion/. [Accessed 12-07-2024].

[14] Copernicus Data Space Ecosystem. Sentinel-2 — dataspace.copernicus.eu. https://dataspace.copernicus.eu/explore-data/data-collections/sentinel-data/sentinel-2. [Accessed 09-07-2024].

[15] GISGeography. Sentinel 2 Bands and Combinations - GIS Geography — gisgeography.com. https://gisgeography.com/sentinel-2-bands-combinations/. [Accessed 09-07-2024].

[16] Raúl González Duque. *Python para todos*. Creative Commons Reconocimiento, 2011.

[17] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. *A Guide to Convolutional Neural Networks for Computer Vision*. Gérard Medioni & Sven Dickinson. Morgan & Claypool, 2018.

[18] D'andrimont R, Claverie M, Kempeneers P, Muraro D, Martinez Sanchez L, and Waldner F. Ai4boundaries: an open ai-ready dataset to map field boundaries with sentinel-2 and aerial photography. *Earth System Science Data*, 15, 2023.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[20] Lasse Rouhiainen. Inteligencia artificial. *Madrid: Alienta Editorial*, pages 20–21, 2018.

[21] Jordi Torres. *Python deep learning: Introducción práctica con Keras y TensorFlow 2*. Alpha Editorial, 2020.