



Universidad de
Oviedo



ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TRABAJO FIN DE MÁSTER

**SEGMENTACIÓN SEMÁNTICA DE IMÁGENES AÉREAS PARA LA
CLASIFICACIÓN DE CULTIVOS**

D. Díaz Pedrayes, Óscar
TUTOR: D. García Martínez, Daniel Fernando
COTUTOR: D. Usamentiaga Fernández, Rubén

FECHA: ENERO 2021

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Alcance	3
1.4. Hardware utilizado en las experimentaciones	3
1.5. Estructura de la memoria	3
1.6. Conceptos generales	4
2. Estado del Arte	7
2.1. Métodos de obtención de imágenes para los conjuntos de datos	7
2.1.1. Sentinel-1	7
2.1.2. Sentinel-2	8
2.1.3. Landsat8	9
2.1.4. SPOT	10
2.1.5. Gaofen-1	11
2.1.6. Drones y helicópteros	12
2.1.7. Aviones	12
2.2. Métodos de clasificación de cultivos	13
2.2.1. Random Forest (RF)	13
2.2.2. Support Vector Machines (SVM)	14
2.2.3. Multilayer Perceptron (MLP)	15
2.2.4. Convolutional Neural Network (CNN)	16
2.3. Tipos de clasificaciones con redes neuronales convolucionales	17
2.4. Redes neuronales convolucionales para la segmentación semántica	19
2.4.1. Introducción	19
2.4.2. Conceptos importantes y entrenamiento de redes	19

2.4.3.	Técnicas de convolución	26
2.4.4.	Redes backbone	30
2.4.5.	UNet	30
2.4.6.	SegNet	33
2.4.7.	DeeplabV3+	34
2.5.	Métricas	38
3.	Información para el etiquetado de las imágenes	44
3.1.	Información de SIGPAC para el conjunto de datos PNOA	48
3.1.1.	Conjunto de datos 20200902	49
3.2.	Información de SIGPAC para el conjunto de datos Sentinel-2	50
3.2.1.	Conjunto de datos 20201209	51
4.	Experimentación con el conjunto de datos PNOA	52
4.1.	Obtención de las imágenes PNOA	52
4.2.	Obtención de los recortes de entrada	53
4.3.	Generación del GroundTruth para los recortes	55
4.4.	Redes neuronales convolucionales utilizables	60
4.5.	Experimentación	61
4.5.1.	Implementación DeepLabV3+ (Matlab)	61
4.5.2.	Implementación DeepLabV3+ (Tensorflow)	67
4.5.3.	Comparaciones	72
4.6.	Servicio de inferencia	72
5.	Experimentación con el conjunto de datos Sentinel-2	76
5.1.	Obtención de las imágenes Sentinel-2	76
5.1.1.	Uso de la API de SentinelHub.	76
5.1.2.	Obtención de hojas MTN50 completas.	79
5.1.3.	Eliminación de nubes.	81
5.2.	Obtención de los recortes de entrada.	81
5.3.	Generación del GroundTruth para los recortes	82

5.4. Redes neuronales convolucionales utilizables	83
5.5. Experimentación con once clases	83
5.5.1. Implementación de UNet en Matlab	83
5.6. Experimentación con tres clases	89
5.6.1. Implementación de UNet en Matlab	89
5.6.2. Implementación de UNet en Keras	94
5.6.3. Comparaciones	98
5.7. Servicio de inferencia	98
6. Conclusiones	104
Anexos	113
Resumen de los Anexos adjuntados	115

Índice de figuras

1.1. Comparación de GFLOPS/s y GB/s proporcionados por GPUs y CPUs. ^[1]	1
2.1. Ejemplo de Random Forest. ^[2]	13
2.2. Ejemplo de SVM. ^[3]	14
2.3. Ejemplo de SVM. Expansión de dos a tres dimensiones. ^[3]	15
2.4. Ejemplo de predicción con MLP. ^[4]	16
2.5. Ejemplo de un filtro de una capa convolucional. Verde=Entrada, Amarillo=Filtro convolucional, Rojo=Mapa de características resultante. ^[5]	17
2.6. Ejemplos de max pooling y average pooling. ^[6]	17
2.7. Ejemplo de los diferentes tipos de clasificación. ^[7]	18
2.8. Ejemplo de sobreajuste. ^[8]	24
2.9. Ejemplo de parada temprana. ^[9]	25
2.10. Ejemplo de dropout. ^[10]	26
2.11. Ejemplo de convolución.	27
2.12. Ejemplo de la variación del parámetro de dilatación. ^[11]	27
2.13. Atrous Convolution. ^[12]	28
2.14. (a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv. ^[13] .	28
2.15. Convolución transpuesta. ^[12]	29
2.16. Ejemplo de ASPP. ^[14]	30
2.17. Estrategia de superposición de baldosas. ^[15]	31
2.18. Arquitectura de UNet. ^[15]	32
2.19. Arquitectura de SegNet. ^[16]	33
2.20. Max-Polling Indices en SegNet. ^[16]	34

2.21. Ejemplo de uso en smartphones Pixel.(a) Imagen sin segmentar ,(b) imagen segmentada. ^[17]	35
2.22. (Izda) Spatial Pyramid Pooling (Ctro) Encoder-Decoder (Dcha) Encoder-Decoder con Atrous Conv. ^[13]	37
2.23. Arquitectura de la red DeepLabV3+. ^[13]	38
2.24. Ejemplo de los diferentes resultados de la predicción de una muestra para clasificación binaria. ^[18]	39
2.25. Cálculo del IoU. ^[19]	40
2.26. Cálculo del F-Score. ^[20]	41
2.27. Ejemplo de matriz de confusión	42
2.28. Ejemplo de representación de cuerdas.	43
3.1. Vista general del visor SIGPAC.	45
3.2. Zoom con visualización de recintos y parcelas del visor SIGPAC.	45
3.3. Ejemplo de hojas MTN50 sobre parte de la geografía española.	48
3.4. Histograma de las clases por su número de píxeles del conjunto de datos 20200902.	50
4.1. Descarga de imágenes en “Ortofoto PNOA Máxima Actualidad”.	52
4.2. Imagen PNOA de la Hoja MTN50 0162 dividida en dos imágenes.	54
4.3. Ejemplo de recortes.	55
4.4. Búsqueda de parcela en el visor de Sigpac.	58
4.5. Capas del visor de Sigpac.	58
4.6. Comprobación del GroundTruth.	59
4.7. Progreso de la GlobalAccuracy y Loss (Matlab E003-01).	65
4.8. Mejor experimento PNOA en DeepLabV3+ (Matlab E003-01). (1) Imagen Original, (2) GroundTruth, (3) ImagenOriginal+GroundTruth, (4) ImagenOriginal+Predicción.	66
4.9. Progreso de Loss (GitHub E010-01).	69
4.10. Recall. Train vs Test (GitHub E010-01).	70
4.11. Precision. Train vs Test (GitHub E010-01).	70
4.12. Mejor experimento PNOA en DeepLabV3+ (GitHub E010-01). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.	71

4.13. Interfaz del servicio web desarrollado.	73
4.14. Petición REST del prototipo web PNOA.	74
5.1. Configuraciones de SentinelHub.	77
5.2. Metadatos de cuadrículas cartográficas MTN25 y MTN50.	80
5.3. Imagen izquierda de la hoja MTN50 número 0162.	81
5.4. Imagen derecha de la hoja MTN50 número 0162.	81
5.5. Ejemplo de recorte de Sentinel-2 (256×256 píxeles).	82
5.6. Progreso de GlobalAccuracy y Loss (Matlab E010-03).	87
5.7. Imágenes de test (Matlab E010-03). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.	88
5.8. Progreso de GlobalAccuracy y Loss (Matlab E011-10).	92
5.9. Imágenes de test (Matlab E011-10). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.	93
5.10. Progreso de GlobalAccuracy (Keras K-001).	96
5.11. Progreso de Loss (Keras K-001).	96
5.12. Imágenes de test (Keras K-001). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.	97
5.13. Ejemplo de máscara predicha de Sentinel-2 poligonizada.	101
6.1. Comparativa entre imágenes RGB de PNOA (inferior) y de Sentinel-2 (superior) sobre la misma región.	105
6.2. Recorte original.	107
6.3. GroundTruth.	107
6.4. Superposición.	107

Índice de tablas

1.1. Componentes de los equipos de pruebas.	3
2.1. Información de las bandas de Sentinel-2A y Sentinel-2B.	9
2.2. Información de las bandas de Landsat8	10
2.3. Información de las bandas de SPOT4	11
2.4. Información de las bandas SPOT5	11
2.5. Información de las bandas Gaofen-1.	12
3.1. Clases disponibles en SIGPAC.	46
3.2. Clases utilizadas de SIGPAC.	47
3.3. Resumen conjunto de datos SIGPAC 20200902 para PNOA.	49
3.4. Resumen conjunto de datos SIGPAC 20201209 para Sentinel-2.	51
4.1. Código de colores de las máscaras GroundTruth.	60
4.2. Código de colores para visualización de las clases.	61
4.3. Parámetros del experimento (Matlab E003-01).	62
4.4. Resultados globales del experimento PNOA DeepLabV3+ (Matlab E003-01).	62
4.5. Métricas de las clases (Matlab E003-01).	63
4.6. Matriz de confusión (Matlab E003-01).	64
4.7. Matriz de confusión normalizada (Matlab E003-01).	64
4.8. Parámetros del experimento (GitHub E010-01).	68
4.9. Resultados globales del experimento PNOA DeepLabV3+ (GitHub E010-01).	68
4.10. Métricas de las clases (GitHub E010-01).	69
4.11. Tiempos de inferencia del prototipo PNOA con CPU.	75
4.12. Tiempos de inferencia del prototipo PNOA con GPU.	75

5.1. Parámetros del experimento E010-03.	84
5.2. Resultados globales del experimento Sentinel-2 UNet (Matlab E010-03).	84
5.3. Métricas de las clase (Matlab E010-03).	85
5.4. Matriz de confusión (Matlab E010-03).	86
5.5. Parámetros del experimento E011-10.	90
5.6. Resultados globales del experimento Sentinel-2 UNet (Matlab E011-10).	90
5.7. Matriz de confusión (Matlab E011-10).	91
5.8. Parámetros del experimento (Keras K-001).	95
5.9. Resultados globales del experimento Sentinel-2 UNet (Keras K-001).	95
5.10. Matriz de confusión (Keras K-001).	95
5.11. Bandas Sentinel-2 de las imágenes a enviar al prototipo web.	99
5.12. Tiempos de inferencia del prototipo Sentinel-2 con CPU.	103

1. Introducción

A lo largo de los últimos años se han producido grandes avances en el campo de la visión por computador mediante redes neuronales convolucionales (CNN), aumentando la complejidad y la precisión de las tareas que se pueden llegar a realizar. Esto empuja a las empresas a competir en una carrera para encontrar nuevos usos para esta tecnología con el objetivo de optimizar los servicios que ya ofrecen o para el diseño de nuevas e innovadoras soluciones para diversos tipos de problemas. Estos avances se deben a la aparición de grandes bases de datos etiquetadas gracias al ‘internet de la cosas’, y al progreso de las GPU o unidades de procesamiento gráfico y con ello la mejora del procesamiento paralelo. Este aumento de potencia en GPU se puede apreciar fácilmente en la figura 1.1, donde se muestra la relación entre las operaciones de punto flotante por segundo y su ancho de banda en las últimas décadas.



Figura 1.1.- Comparación de GFLOPs/s y GB/s proporcionados por GPUs y CPUs.^[1]

1.1.- Motivación

El presente proyecto nace del interés de la detección automática de los diferentes tipos de uso agrícola de la tierra en imágenes aéreas para una obtención temprana y barata, de imágenes etiquetadas con sus respectivos usos de la tierra, al reducir la mano



de obra involucrada en dicho proceso. Para llevar a cabo esta propuesta se evaluará la viabilidad de las tecnologías del estado del arte relativa a las redes convolucionales neuronales (CNN) en el campo de la visión por computador, en concreto, de la técnica conocida como “Segmentación Semántica”. Debido al interés que esta propuesta genera, SERESCO ha lanzado el Proyecto TERRA para la evaluación de estas posibilidades, proyecto del cual nace este TRABAJO DE FIN DE MÁSTER.

1.2.- Objetivos

Los objetivos de este Trabajo de Fin de Máster constan de los siguientes puntos:

- El principal objetivo de este trabajo es el del reconocimiento de los diferentes usos de la tierra en imágenes aéreas mediante técnicas de visión por computador para procesar grandes cantidades de datos con rapidez, de forma automatizada y con bajo costo.
- Estudio de diferentes alternativas de visión por computador para la resolución del problema planteado.
- Estudio de diferentes alternativas para la segmentación semántica.
- Generación de un conjunto de datos válido y realista para la evaluación de las técnicas alternativas de segmentación semántica.
- Estudio de las diferentes métricas de los algoritmos de segmentación semántica y sus resultados para el reconocimiento de tipos de uso de la tierra en imágenes áreas.
- Estudio de los parámetros de entrada de las redes neuronales convolucionales para la segmentación semántica.
- Experimentación con los conjuntos de datos generados.
- Estudio del coste y viabilidad de la puesta en marcha de un servicio de reconocimiento de usos de la tierra basado en imágenes.



1.3.- Alcance

El alcance de este proyecto será el de abordar todos los objetivos presentados en el apartado anterior y presentar los estudios realizados en un Trabajo de Fin de Máster. Además, se pretende asistir a la empresa colaboradora SERESCO mediante la transferencia de conocimientos y proporcionándole toda la asistencia necesaria.

1.4.- Hardware utilizado en las experimentaciones

En esta sección se resumen los equipos utilizados en las experimentaciones realizadas. Los tres equipos utilizados poseen la misma configuración principal, reflejada en la tabla 1.1. Cada equipo ha supuesto un coste unitario de unos 3,800€.

Componente	Modelo
CPU	Intel Core i7-9700K
Placa Base	Gigabyte Aorus Z390
RAM	16 GB DDR4 3000Mhz Vengeance Corsair
Disco SSD NVMe para el SO	1TB SSD NVMe M.2. S970 EVO PLUS Samsung
Disco SSD	4 TB SSD S860 EVO Samsung
Disco HDD	14 TB WD GOLD
GPU	Geforce RTX 2080Ti TURBO 11GB
Caja	ATX F200 ELEMENTS COOLBOX
Fuente de alimentación	Corsair HX1200 1200W Platinum

Tabla 1.1.- Componentes de los equipos de pruebas.

1.5.- Estructura de la memoria

1. Introducción.

Se presenta la temática de este Trabajo Fin de Máster, así como su motivación, objetivos y estructura. También se presentan los equipos utilizados en las experimentaciones.

2. Estado del arte.

Estudio del estado del arte de la obtención de imágenes, diferentes técnicas



de visión por computador y sus métricas. Centrándose en el estudio de la segmentación semántica.

3. Información para el etiquetado de las imágenes.

Se detalla la obtención de los diferentes tipos de uso de la tierra por parcela a partir de SIGPAC así como su uso para la obtención de un GroundTruth en los conjuntos de datos a generar.

4. Experimentación con el conjunto de datos PNOA.

Resumen del procedimiento seguido para la generación de un conjunto de datos con imágenes de PNOA y la presentación de los mejores experimentos con dicho conjunto de datos.

5. Experimentación con el conjunto de datos Sentinel-2.

Resumen del procedimiento seguido para la generación de un conjunto de datos con imágenes de Sentinel-2 y la presentación de los mejores experimentos con dicho conjunto de datos.

6. Conclusiones.

Se comentan las conclusiones obtenidas durante el desarrollo de este Trabajo de Fin de Máster.

7. Referencias.

Se citan todas las referencias utilizadas en este documento.

8. Anexos.

Se resume brevemente el contenido de los diferentes anexos que se adjuntan con este documento.

1.6.- Conceptos generales

A continuación, se explican brevemente una serie de conceptos generales que hay que conocer antes de comenzar la lectura de este documento.

- **Hojas MTN50.** Regiones del Mapa Topológico Nacional a escala 1:50.000. Existen 1116 regiones que mapean la totalidad de la región española.



- **Ground sample distance (GSD).** Se refiere a la resolución en relación a los metros, es decir, un GSD de 0.25 quiere decir que cada píxel tiene un tamaño sobre el terreno de 25 cm de ancho y 25 cm de alto.
- **Formato ECW.** Imágenes en un formato específico para imágenes georeferenciadas de gran tamaño que permiten una gran capacidad de compresión sin pérdida. Una imagen en formato TIF de 20 GB puede llegar a ocupar unos 2 GB en este formato.
- **Formato GEOTIFF.** Imágenes en formato TIF que tienen todos sus píxeles georeferenciados según un sistema CRS (Coordinate Reference System).
- **Imágenes PNOA.** Imágenes del Plan Nacional de Ortografía que siguen las regiones MTN50. Son imágenes en formato ECW con un GSD de 0.25 metros/píxel que tienen como área una región MTN50 completa. Estas imágenes no contienen nubes.
- **Librería GDAL.** La herramienta más utilizada para el tratamiento de imágenes georeferenciadas, se utiliza de backend en herramientas como QGIS. Dispone de enlaces a Python, C++, C y Java.
- **GroundTruth.** Son los datos reales de los que debe aprender el modelo para crear sus predicciones. En este caso consta de imágenes en las que los valores de los píxeles indican el tipo de cultivo de la zona.
- **Recortes.** Cuando se habla de recortes se refiere a las imágenes que se usarán en el conjunto de datos. Estas imágenes son recortes de 256×256 píxeles de las imágenes de tamaño completo. Se decide esta resolución por ser la que mejores resultados ofrece en varias de las redes neuronales testadas.
- **Sistema de Información Geográfica de Parcelas Agrícolas (SIGPAC).** Es una base de datos proporcionada por el Gobierno de España que permite identificar de forma georeferenciada las diferentes parcelas declaradas por los agricultores y ganaderos.
- **Parcela.** Son las regiones identificadas por SIGPAC. A la hora de crear el GroundTruth solamente se tienen en cuenta los tipos de cultivo y no las parcelas de forma individual debido a que es una separación política y no natural, por lo que presenta grandes dificultades para su predicción y, además, se sale del interés del proyecto.



- **Clase.** Las clases a predecir son los diferentes los tipos de cultivo.
- **Imágenes multispectrales.** Imágenes con múltiples bandas fuera del espectro visible.
- **Bandas.** Los diferentes canales o bandas de las imágenes, como por ejemplo, Red, Green, Blue, IR, etc.

2. Estado del Arte

2.1.- Métodos de obtención de imágenes para los conjuntos de datos

En esta sección se detallan los métodos más utilizados para la obtención de imágenes para la generación de conjuntos de datos para el entrenamiento de modelos de predicción de clasificación de tipos de uso de la tierra. Los conjuntos de datos para la predicción de cultivos constan de un conjunto de imágenes y su conjunto de imágenes GroundTruth equivalente en el que se etiquetan las diferentes clases o cultivos, asignando a cada clase un valor RGB o un nivel en una escala de grises, para visualizar fácilmente la clasificación en zonas de una imagen.

2.1.1.- Sentinel-1

Los satélites artificiales de órbita polar Sentinel-1 [21] pertenecen a la ESA (Agencia Espacial Europea) dentro del Programa Copérnico destinado a la monitorización terrestre y de los océanos. Sentinel-1A fue lanzado al espacio el 3 abril de 2014. El 25 de Abril de 2016 se lanzó Sentinel-1B. Las imágenes capturadas por estos satélites son de tipo SAR (Synthetic Aperture Radar).

Las características principales de Sentinel-1 son:

- Repetición del ciclo cada 12 días.
- 175 revoluciones por ciclo.
- 98,6 minutos de periodo orbital.
- Estabilización de altitud de 3 ejes.
- Altitud de 693 km.
- Órbita helio síncrona polar (98.18°).
- 7 años de vida totales (12 años de combustible).



2.1.2.- Sentinel-2

Sentinel-2 [22] es una misión de observación terrestre desarrollada por la Agencia Espacial Europea para el programa Copérnico, para ello utilizan dos satélites idénticos, Sentinel-2A y Sentinel-2B. Esta misión está orientada a la observación de la Tierra para dar servicios de seguimiento de la evolución de la vegetación, cambios en la corteza terrestre y gestión de desastres naturales. Las características principales de Sentinel-2 son:

- Imágenes multispectrales de 13 bandas.
- Bandas en el espectro visible, infrarrojo cercano, infrarrojo de onda corta y de espectro electromagnético (ver tabla 2.1).
- Cobertura global sistemática de las capas de tierra de 56° S a 84°N, aguas costeras y el mar Mediterráneo.
- Actualiza sus imágenes cada 5 días, manteniendo los mismos ángulos de visión. En algunas regiones de latitudes altas se actualizan cada 5 días varias veces desde distintos ángulos.
- Dependiendo de la banda obtiene imágenes con una resolución espacial de 10, 20 o 60 metros por píxel.
- Tiene un campo de visión de 290 km.
- Política de datos libre y en abierto.



Banda	Long. de onda (nm)		Ancho de banda (nm)		GSD (m/píxel)
	S-2A	S-2B	S-2A	S-2B	
B1 Coastal aerosol	442.7	442.2	21	21	60
B2 Blue	492.4	492.1	66	66	10
B3 Green	559.8	559	36	36	10
B4 Red	664.6	664.9	31	31	10
B5 VRE	704.1	703.8	15	16	20
B6 VRE	740.5	739.1	15	15	20
B7 VRE	782.8	779.7	20	20	20
B8 NIR	832.8	832.9	106	106	10
B8A Narrow NIR	864.7	864	21	22	20
B9 Water vapour	945.1	943.2	20	21	60
B10 SWIR Cirrus	1374	1376.9	31	30	60
B11 WIR	1614	1610.4	91	94	20
B12 SWIR	2202	2185.7	175	185	20

Tabla 2.1.- Información de las bandas de Sentinel-2A y Sentinel-2B.

Las siglas “VRE” significan “Vegetation Red Edge”, “NIR” quiere decir “Near InfraRed”, y finalmente “SWIR” es “Short Wave InfraRed”. De color rojo se muestran las bandas con un GSD de 60 m/píxel, de amarillo las bandas de 20 m/píxel y de verde las de 10 m/píxel.

2.1.3.- Landsat8

Landsat 8 [23] es un satélite de observación terrestre estadounidense lanzado el 11 de febrero de 2013. Es el octavo y más reciente satélite del proyecto Landsat operado por la NASA y el Servicio Geológico de los Estados Unidos (USGS). El satélite Landsat 8 transporta dos instrumentos OLI y TIRS, que corresponden a las siglas en inglés de Operational Land Imager (OLI) y Thermal Infrared Sensor (TIRS). El sensor OLI provee acceso a las primeras nueve bandas espectrales que cubren el espectro desde los



0.433 μm a los 2.300 μm , mientras que TIRS registra de 10.30 μm a 12.50 μm para las dos últimas bandas, es decir, las bandas diez y once. Tiene un periodo de 16 días por ciclo. Las características de las diferentes bandas de Landsat8 se encuentran en tabla 2.2.

Nombre	Longitud de onda (μm)	GSD (m/píxel)
B1 Costera - Aerosoles	0.435 - 0.451	30
B2 Azul	0.452 - 0.512	30
B3 Verde	0.533 - 0.590	30
B4 Rojo	0.636 - 0.673	30
B5 NIR	0.851 - 0.879	30
B6 SWIR 1	1.566 - 1.651	30
B7 SWIR 2	2.107 - 2.294	30
B8 Pancromática	0.503 - 0.676	15
B9 Cirrus	1.363 - 1.384	30
B10 TIR 1	10.60 - 11.19	100
B11 TIR 2	11.50 - 12.51	100

Tabla 2.2.- Información de las bandas de Landsat8

2.1.4.- SPOT

Los satélites SPOT (Satellite Pour l'Observation de la Terre: Satélite Para la Observación de la Tierra) son una serie de satélites de teledetección civiles de observación del suelo terrestre. La duración de cada ciclo es de 6 días. Existen varios satélites SPOT pero las versiones mas utilizadas son SPOT4, lanzado el 23 de marzo de 1998 (ver tabla 2.3), y SPOT5, lanzado el 3 de mayo de 2002 (ver tabla 2.4).



Bandas	Longitud de onda (μm)	GSD (m/píxel)
B1 Verde	0,50 - 0,59	20
B2 Rojo	0,61 - 0,68	20
B3 NIR	0,78 - 0,89	20
B4 IRM	1,58 - 1,75	20
B5 Rojo monoespectral	0,61 - 0,68	10

Tabla 2.3.- Información de las bandas de SPOT4

Bandas	Longitud de onda (μm)	GSD (m/píxel)
B1 Verde	0,50 - 0,59	10
B2 Rojo	0,61 - 0,68	10
B3 NIR	0,78 - 0,89	10
B4 IRM	1,58 - 1,75	10
B5 Pancromático	0,48 - 0,71	5
B6 Súper-modo pancromático	0,48 - 0,71	2.5

Tabla 2.4.- Información de las bandas SPOT5

2.1.5.- Gaofen-1

Los satélites Gaofen son una serie de satélites chinos para la obtención de imágenes de alta resolución de la Tierra. El Gaofen-1 [24], lanzado en 2013, es el más utilizado de esta serie. La información de cada banda se puede observar en la tabla 2.5. Las características principales del satélite Gaofen-1 son:

- Imágenes de 8 m/píxel y 16 m/píxel.
- Actualiza su información de un periodo de unos 4 días.
- Sigue la órbita heliosíncrona a una altitud de 645 km con una inclinación de 98°.
- Vida útil entre 5 y 8 años.



Bandas	Longitud de onda (μm)	GSD (m/píxel)
PAN	0.45-90	2
B1 Blue	0.45-0.52	8, 16
B2 Green	0.52-0.59	8, 16
B3 Red	0.63-0.69	8, 16
B4 NIR	0.77-0.89	8, 16

Tabla 2.5.- Información de las bandas Gaofen-1.

2.1.6.- Drones y helicópteros

Para la obtención de conjuntos de datos con un GSD mas pequeño, es decir, mayor resolución, se utilizan drones y/o helicópteros que permiten obtener un GSD de hasta unos pocos centímetros por píxel. Su desventaja principal es que se genera una enorme cantidad de datos para cubrir una zona amplia de cientos de kilómetros, además de su gran costo.

2.1.7.- Aviones

Finalmente, se utilizan aviones, de forma similar a drones y helicópteros, pero que, sin embargo, son capaces de cubrir grandes áreas de terreno. En España se dispone de forma gratuita de las imágenes del Plan Nacional de Ortofotografía Aérea o PNOA, que consta de toda la geografía española fotografiada con un GSD de 0.25 m/píxel en los meses de verano. Su principal desventaja es que no se dispone de bandas multiespectrales y que esta información se actualiza de forma anual, imposibilitando el estudio del crecimiento o diferentes etapas de los cultivos. El costo de la obtención de imágenes con aviones es muy elevado si se decide no utilizar una base de datos gratuita como PNOA.

2.2.- Métodos de clasificación de cultivos

En esta sección se resumen brevemente las técnicas más comunes para la predicción de diferentes tipos de cultivo sobre imágenes aéreas. Se presentan en orden de peores a mejores resultados de precisión en la clasificación, según la literatura de publicaciones como [25] y [26].

2.2.1.- Random Forest (RF)

Este método sigue siendo el más común en predicción de tipos de cultivos, consiste en utilizar un gran número de árboles de decisión individualmente. Cada árbol de decisión devuelve una predicción propia y la clase con más votos se decide como la ganadora (ver figura 2.1).

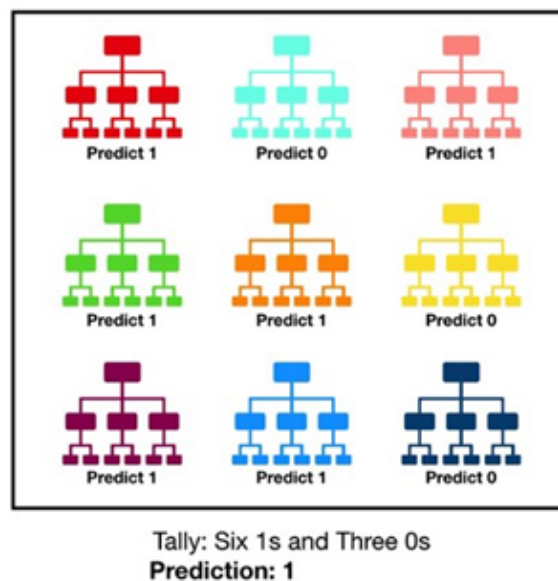


Figura 2.1.- Ejemplo de Random Forest. [2]

Este método se basa en la baja correlación entre los diferentes árboles, y en su combinación para protegerse de errores individuales. Para obtener este efecto se utiliza la técnica de **Bagging**, que consiste en la obtención de forma aleatoria de los elementos de conjunto de datos, evitando que existan dos árboles iguales. Para utilizar este método se necesita que exista algún tipo de señal en las características de los modelos para



evitar una elección aleatoria. Las predicciones y los árboles necesitan tener una baja correlación entre sí.

2.2.2.- Support Vector Machines (SVM)

La técnica SVM consiste en la optimización de la búsqueda de un hiper-plano que separe con el mayor margen posible todos los individuos de las diferentes clases, también conocido como “maximal margin hyperplane”. SVM permite que algunas de las muestras se encuentren dentro del margen o incluso al otro lado del hiper-plano para reducir el sobreajuste de datos y obtener un modelo más robusto (ver figura 2.2).

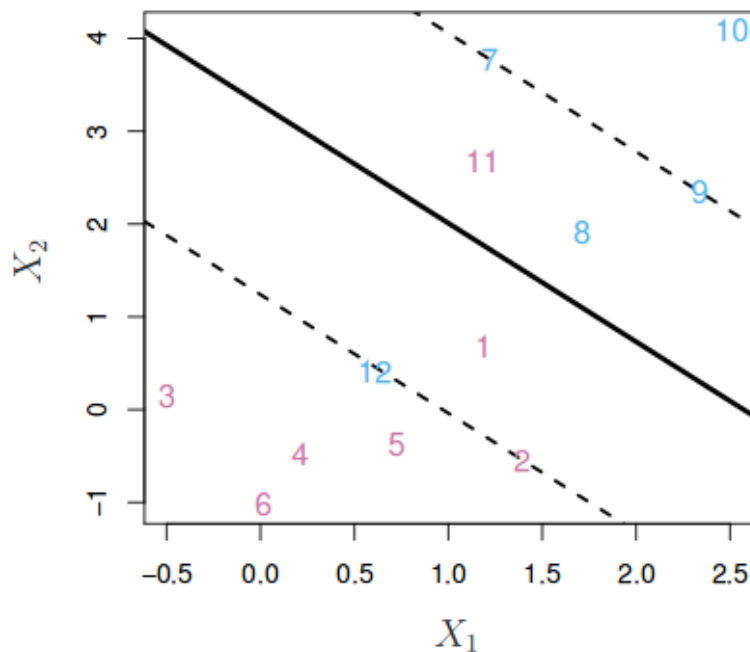


Figura 2.2.- Ejemplo de SVM.^[3]

El proceso de optimización tiene la peculiaridad de que solo las observaciones que se encuentran justo en el margen o que lo violan influyen sobre el hiper-plano. A estas observaciones se les conoce como vectores soporte y son las que definen el clasificador obtenido. Para controlar este margen se utiliza el parámetro C, que al aumentar de tamaño permite que más observaciones influyan en el clasificador. Esta técnica consigue buenos resultados cuando el límite de separación entre clases es aproximadamente lineal. Si no lo es, su capacidad decae drásticamente. Una estrategia para enfrentarse

a escenarios en los que la separación de los grupos es de tipo no lineal consiste en expandir las dimensiones del espacio original (ver figura 2.3).

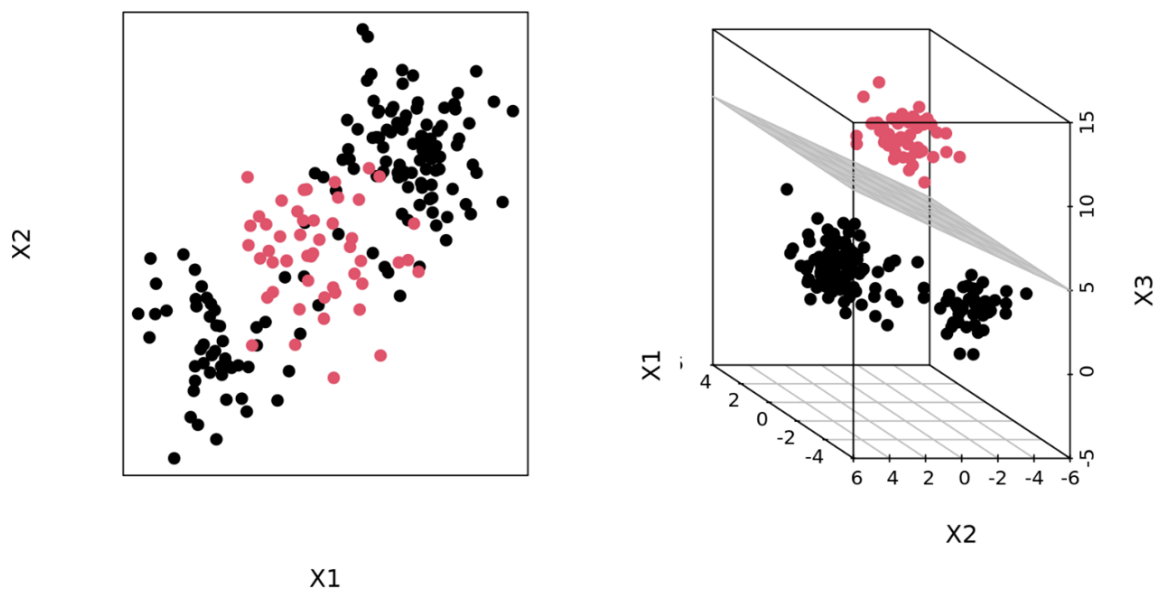


Figura 2.3.- Ejemplo de SVM. Expansión de dos a tres dimensiones.^[3]

2.2.3.- Multilayer Perceptron (MLP)

Estas redes neuronales artificiales consisten en una capa de entrada, una o varias capas ocultas y una capa de salida. Utiliza la técnica conocida como **Backpropagation** para su entrenamiento, alterando los pesos de la red después de cada iteración provocando, en teoría, una mejora en los resultados. Después de cada capa se utiliza una función de activación, excepto en la primera capa de entrada, comúnmente ReLU o LeakyReLU, anteriormente era común utilizar una Sigmoide. Estas redes neuronales consisten en un conjunto de perceptrones. Un perceptrón es lo que se conoce como neurona, es decir, un discriminador lineal a partir del cual se genera un criterio para seleccionar un sub-grupo de componentes. Al utilizar solamente un perceptrón la discriminación se ve limitada a un hiper-plano, sin embargo, al utilizar múltiples perceptrones, como es el caso del Perceptrón Multicapa (MLP) se pueden conseguir discriminaciones mucho más complejas (ver figura 2.4).

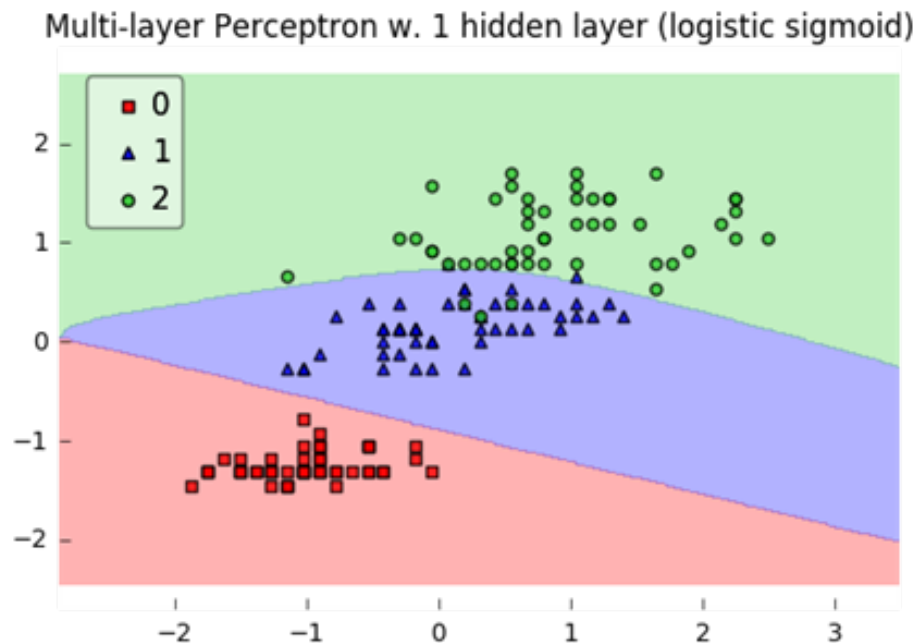


Figura 2.4.- Ejemplo de predicción con MLP.^[4]

2.2.4.- Convolutional Neural Network (CNN)

Estas redes neuronales artificiales son una variación del perceptrón multicapa y, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones, por lo que obtienen mejores resultados que el resto de técnicas resumidas en esta sección. La estructura de las CNNs consiste en múltiples capas de filtros convolucionales (ver figura 2.5) de una o más dimensiones, donde, después de cada capa, generalmente se añade una función de activación como ReLU o LeakyReLU. Al final de la red se encuentran neuronas de perceptrón sencillas para realizar la clasificación final sobre las características extraídas, generalmente, una capa Softmax para la predicción multiclase. Puede contener capas de Pooling (ver figura 2.6) para la reducción del tamaño de las imágenes de entrada y capas densas similares a un perceptrón multicapa, aunque se pueden construir este tipo de redes solamente utilizando capas convolucionales como es el caso de la red UNet.

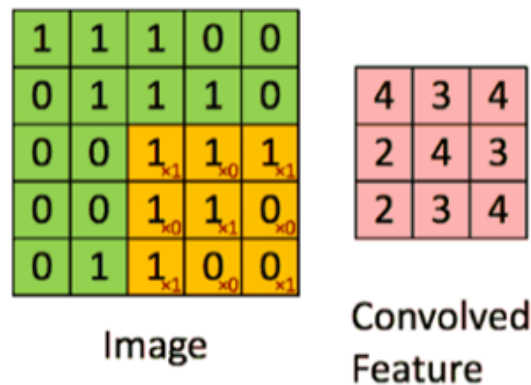


Figura 2.5.- Ejemplo de un filtro de una capa convolucional. Verde=Entrada, Amarillo=Filtro convolucional, Rojo=Mapa de características resultante.^[5]

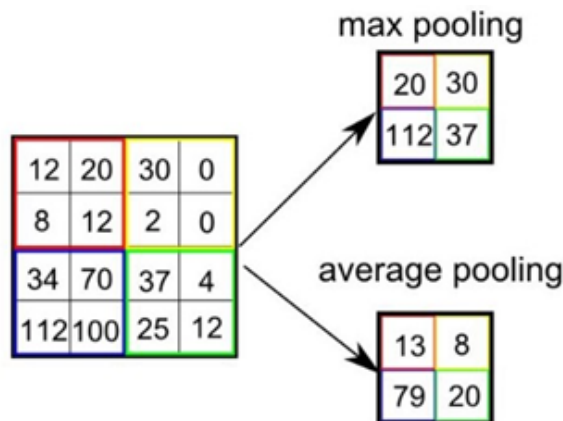


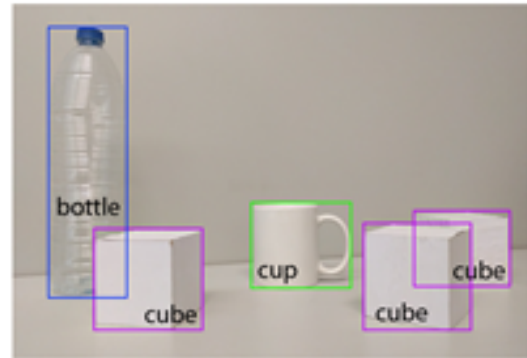
Figura 2.6.- Ejemplos de max pooling y average pooling.^[6]

2.3.- Tipos de clasificaciones con redes neuronales convolucionales

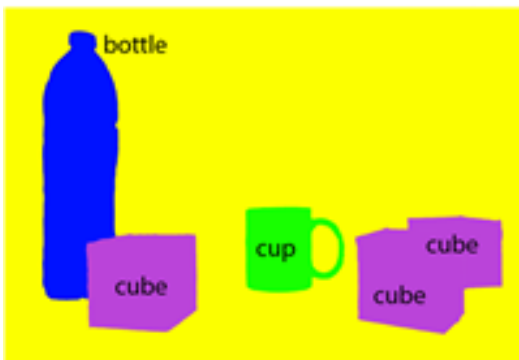
Para la clasificación de objetos en imágenes existen diferentes algoritmos, cada uno tiene una utilidad y complejidad diferente según la tarea que se pretenda resolver. Estos algoritmos se pueden observar se forma gráfica en la figura 2.7.



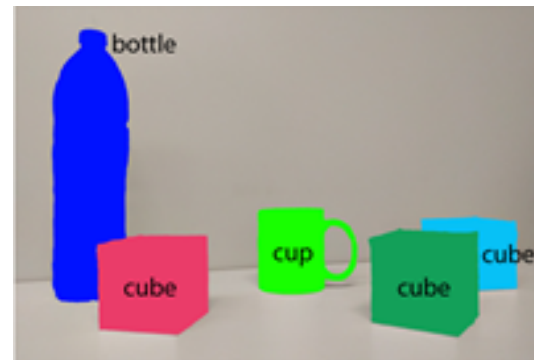
(a) Clasificación de imagen.



(b) Localización de objetos.



(c) Segmentación semántica.



(d) Segmentación de instancias.

Figura 2.7.- Ejemplo de los diferentes tipos de clasificación.^[7]

A continuación, se explicarán brevemente los diferentes algoritmos, por orden creciente de complejidad.

- **Clasificación de imágenes** (figura 2.7a), consiste en obtener la clase o etiqueta de un objeto en una imagen, sin especificar donde se encuentra ni si hay más de uno, es decir de forma binaria. Comúnmente se utilizan imágenes con un solo objeto de gran tamaño y centrado en la misma, aunque esto no es obligatorio.
- **Detección de objetos** (figura 2.7b), consiste en clasificar una imagen de forma similar al algoritmo de “Clasificación de imágenes”, pero en este caso se añade la localización de los objetos etiquetados en la imagen mediante un rectángulo alrededor del objeto a localizar conocido como “Bounding Box”.
- **Segmentación Semántica** (figura 2.7c), consiste en la clasificación de todos los píxeles de la imagen asociando a cada uno su etiqueta correspondiente. De esta forma se obtienen conjuntos de píxeles clasificados con la misma etiqueta.



Con este algoritmo no se puede conocer el número de instancias de los objetos, solamente el número de píxeles etiquetados o clasificados con una etiqueta concreta.

- **Segmentación de instancias** (figura 2.7d), consiste en una variación del algoritmo de “Segmentación semántica” que permite clasificar solamente aquellos píxeles que pertenezcan a objetos a clasificar. Con este algoritmo se puede obtener el número de instancias de los objetos de la imagen, se puede considerar una evolución de la “Detección de objetos” ya que su uso es similar y tiene una mayor precisión al no tener que limitarse a una localización mediante una caja rectangular.

2.4.- Redes neuronales convolucionales para la segmentación semántica

2.4.1.- Introducción

El uso habitual de las redes convolucionales consiste en tareas de clasificación, donde la salida de una imagen es una etiqueta de una clase. Sin embargo, en muchas tareas de visualización, como puede ser en el procesamiento de imágenes biomédicas, la salida deseada incluye la localización exacta, es decir, la etiqueta de la clase debe ser asignada a cada píxel. Por este motivo se decide utilizar redes de segmentación semántica para la resolución de este problema.

2.4.2.- Conceptos importantes y entrenamiento de redes

En esta subsección se resumen brevemente los conceptos mas importantes a tener en cuenta a la hora de entrenar una red neuronal convolucional.

- **Epochs** [27]: Número de veces en los que el conjunto de datos completo es utilizado para entrenar la red neuronal.
- **BatchSize** [27]: Número de imágenes utilizadas en cada lote. Como no se puede pasar el conjunto de datos completo a la red ya que no cabe en memoria, se divide en lotes o batches.



- **LearningRate [28]:** Parámetro que controla la velocidad a la que se ajustan los pesos de la red respecto al gradiente. Un valor bajo provoca un entrenamiento mas lento, que puede caer en un mínimo local. Un valor elevado aumenta la velocidad a la que converge la red pero puede no llegar estabilizarse en un mínimo realizando un entrenamiento menos preciso.
- **Data Augmentation:** Generación de nuevos datos a partir de los originales para aumentar artificialmente el número de muestras del conjunto de datos de entrenamiento.
- **Momentum:** Valor para indicar cuanto contribuye la iteración anterior con la iteración actual en lo que se refiere al descenso del gradiente estocástico cuando se utiliza el solucionador SGDM.
- **Decay [29]:** Periodo de epochs en el que se modifica el learning rate.
- **ValidationData:** Datos para la validación del modelo tras cada epoch.
- **ValidationFrequency:** Número de iteraciones antes de realizar la validación.
- **ValidationPatience:** Número de validaciones con un valor de pérdida igual o menor al obtenido anteriormente antes de parar el entrenamiento.
- **L2Regularization [30]:** La regularización es una técnica para reducir la complejidad de un modelo, para ello se penaliza la función de pérdida. Gracias a esto se consigue reducir el sobreajuste del modelo.
- **Propagación hacia atrás o Backpropagation.** Es el método mediante el cual se entrena el modelo. Se trata de modificar los pesos de los filtros convolucionales una vez obtenida la predicción de la red, calculando su error contra un GroundTruth, tratando de alterar los pesos para reducir ese error. Esta operación se repite tras cada iteración.
- **La función de pérdida o de Loss** devuelve un valor numérico que resume el grado de error de la predicción y se utiliza para hacer el proceso de propagación hacia atrás o backpropagation. Existen multitud de funciones de pérdida diferentes pero en este proyecto siempre se utiliza la función Cross-entropy tras una función Softmax píxel a píxel sobre el mapa de características final, ya que esta es la metodología seguida para la clasificación multiclase y no se recomienda variarla salvo causa mayor. De esta forma se clasifica cada píxel en una de las clases por lo que la idea es que todos los píxeles tengan su propia



clase convirtiendo el problema de la segmentación en un problema de clasificación multiclase.

La **función softmax** es un tipo de función sigmoide que se encarga de asignar probabilidades a múltiples clases, cuya suma debe ser igual a uno. K es el número de clases y z el vector de longitud K , donde se almacenan los valores de las clases a normalizar.

$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

La **función de cross-entropy** es una función de pérdida que se alimenta de dos distribuciones de probabilidades, en este caso la devuelta por la función softmax (q) que se refiere a la distribución de probabilidades predichas (0.47, 0.2, 0, 0.2...), y la obtenida del etiquetado (p) que se refiere a la distribución de probabilidades del etiquetado del GroundTruth (1,0,0,0...). K es el número de clases.

$$C(p, q) = - \sum_{i=1}^K p(i) \log q(i)$$

- **Método de umbral del gradiente (GradientThreshold).** Existen diferentes métodos que escalan el gradiente en función de un umbral con el objetivo eliminar el problema del “exploding gradient” en el que los gradientes aumentan exponencialmente hasta el infinito provocando que la red colapse. Generalmente, no produce ningún tipo de impacto en el rendimiento o precisión de la red siempre que no se utilicen valores muy bajos.
- **Inicialización de los pesos de los filtros convolucionales.** La inicialización de los pesos de las capas de convolución es una parte muy importante de las redes neuronales convolucionales, debido a que si estos pesos no están correctamente inicializados pueden aparecer problemas como el exploding gradient o vanishing gradient provocando que el entrenamiento de la red se vea interrumpido y que



esta no llegue a converger nunca. Existen muchas formas de inicializar estos pesos, pero las más comunes son las explicadas a continuación.

Inicialización de Xavier [31]: Recomendada para su uso con funciones de activación de tipo Sigmoide. Se calcula mediante una distribución normal con la media en cero y la desviación típica de la raíz de uno dividido entre la cantidad de valores de entrada. El valor input es el número de valores de entrada de la capa.

$$N(0, \sqrt{\frac{1}{input}})$$

Inicialización de He Normal (He-et-al) [31]: Recomendada para su uso con funciones de activación de tipo ReLU. Se calcula mediante una distribución normal con la media en cero y la desviación típica de la raíz de dos dividido entre la cantidad de valores de entrada más los de salida. El valor input es el número de valores de entrada de la capa y el valor de output la cantidad de valores de salida.

$$N(0, \sqrt{\frac{2}{input + output}})$$

- **Balanceo de clases.** Si un conjunto de datos esta desbalanceado, es decir, si las clases no tienen mas o menos la misma cantidad de muestras, al entrenar la red se observará que se tiende a clasificar los píxeles como de aquellas clases que tengan mas muestras. Para evitar esto se añade un peso por cada clase por el que multiplicar el valor de la función de pérdida. De esta forma la clase mas abundante se multiplicara por un valor menor que aquella que tenga menos ejemplos, disminuyendo su peso en la propagación hacia atrás o backpropagation y dejando entrenar a clases que tengan un menor número de muestras. Para calcular estos pesos de utilizan los siguientes métodos.



Uniform prior weighting: Utiliza la probabilidad a priori para el cálculo de los pesos, sin tener en cuenta, inicialmente, el valor real de los datos. Donde P es un vector que contiene la cantidad de píxeles de cada clase, y N el número total de clases.

$$prior = \frac{1}{N}$$
$$uniformClassWeights_i = \frac{prior}{P_i}$$

Inverse frequency weighting: Es la inversa de la probabilidad de frecuencia de una clase en el conjunto de datos de entrenamiento. Donde P es un vector que contiene la cantidad de píxeles de cada clase, y T la cantidad total de píxeles.

$$freq_i = \frac{P_i}{T}$$
$$invFreqClassWeights_i = \frac{1}{freq_i}$$

Median frequency weighting: Utiliza la mediana de la frecuencia de las clases sobre la cantidad de píxeles de cada imagen del conjunto de datos del entrenamiento. Donde P es un vector que contiene la cantidad de píxeles de cada clase, e I la cantidad total de píxeles de una imagen.

$$freq_i = \frac{P_i}{I}$$
$$medFreqClassWeights_i = \frac{median(freq_i)}{freq_i}$$

- **Sobreajuste de la red.** Un modelo que tiene sobreajuste es un modelo que ha memorizado el conjunto de datos de entrenamiento, esto quiere decir que, en vez de encontrar las características necesarias para la clasificación de un objeto, simplemente ha memorizado la imagen completa del objeto, lo que provoca

que cuando se utilice una nueva imagen a modo de test, esta se clasifique erróneamente.

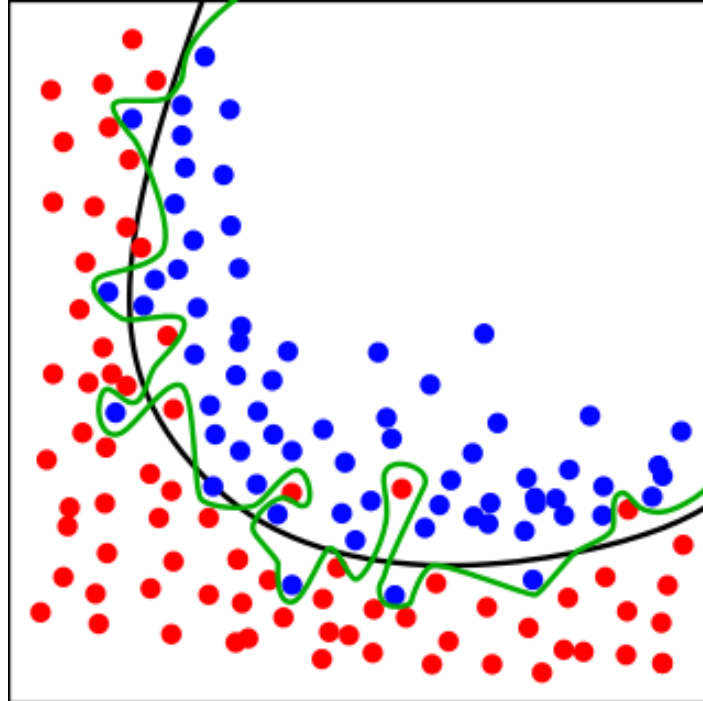


Figura 2.8.- Ejemplo de sobreajuste.^[8]

En la figura 2.8 se observa un ejemplo de sobreajuste, donde se predicen correctamente todos los datos de entrenamiento, pero la curva obtenida (la curva verde) no se parece a la deseada (la curva negra). Esto se entiende como una memorización del ruido de los datos de entrenamiento. Existen diferentes formas de evitar el sobreajuste del modelo, las más comunes son las siguientes.

Simplificar el modelo: si el modelo es excesivamente complejo puede entrar en demasiado detalle y memorizar particularidades en lugar de obtener la tendencia de los datos.

Parada temprana: consiste en parar el entrenamiento cuando se comienza a producir el sobreajuste de la red con el fin de obtener el mejor modelo, ya que si se sobreentrena la red se pierde la generalización del modelo. En la figura 2.9 se muestra un ejemplo de esta técnica. Además, una vez realizada sirve como base para seleccionar el número de epochs que necesita la red para su entrenamiento.

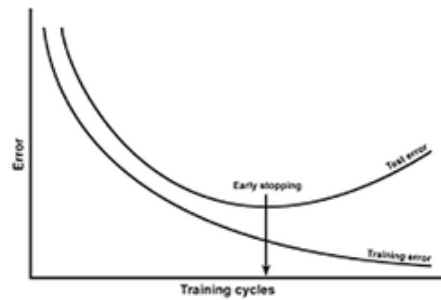


Figura 2.9.- Ejemplo de parada temprana.^[9]

Aumento de datos: Si se añaden más datos de los que puede memorizar la red esta no podrá llegar al sobreajuste. Existen múltiples métodos para aumentar la cantidad de datos disponibles, como, por ejemplo: rotación, escalado, cambios en el brillo o contraste, añadir ruido, reflejar las imágenes en los diferentes ejes, etc.

Uso de regularización: La regularización es una técnica para reducir la complejidad del modelo mediante la penalización de la función de pérdida. Las técnicas más comunes son L1 y L2. La regularización L1 minimiza el valor absoluto de los pesos. La regularización L2 minimiza la magnitud al cuadrado de los pesos de la red. Si los datos de entrenamiento son sencillos se debe utilizar la regularización L1 ya que es más robusta contra valores atípicos. En cambio, si los datos de entrenamiento son más complejos se debe utilizar la regularización L2. En el caso de la visión por computador la regularización L2 suele ser la opción que presenta mejores resultados.

Dropout: es un tipo de regularización que modifica la red, en lugar de la función de pérdida como en las regularizaciones L1 y L2. Esta técnica consiste en la desactivación de un porcentaje de las neuronas de la red de forma aleatoria para que se entrenen “diferentes redes” que produzcan sobreajuste de diferentes formas de manera que al usar la red completa se equilibren. En la figura 2.10 se muestra un ejemplo de dropout.

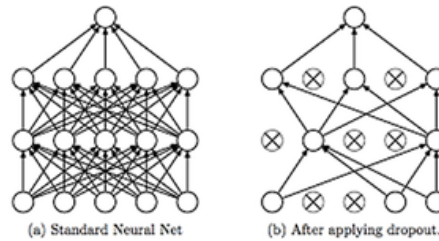


Figura 2.10.- Ejemplo de dropout.^[10]

2.4.3.- Técnicas de convolución

La convolución es la parte principal de las redes convolucionales. La red es capaz de entrenar y memorizar gracias a la variación de los pesos de los filtros o kernels. En esta subsección se resumen los diferentes tipos de convoluciones más usadas en redes de segmentación semántica.

Convolution. La convolución más común y que sirve como base para el resto de variaciones de la misma.

Se calcula de la siguiente forma, donde “O” es el mapa de características resultante de la convolución, “W” la matriz de pesos del filtro y “X” la matriz de entrada (ya sea la imagen original o un mapa de características intermedio). Simplemente se multiplican los pesos del filtro por los valores de la matriz de entrada de forma que se multipliquen aquellos valores en las mismas posiciones.

$$O_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$O_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$O_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

$$O_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Para completar la salida se repite este proceso por toda la matriz de entrada mediante una ventana deslizante (la primera iteración se marca de color rojo en la figura 2.11), si la matriz de entrada tiene múltiples dimensiones también se debe iterar por todas y cada una de las mismas. Este proceso siempre genera una salida de menor tamaño que la matriz de entrada, por lo que, si se quiere mantener el tamaño de entrada,

hay que aplicar padding a la matriz de entrada de forma que se aumente su tamaño lo suficiente para que al finalizar dicho proceso la matriz de salida sea del mismo tamaño que la matriz de entrada original.

$$\begin{array}{|c|c|c|} \hline X_{11} & X_{12} & X_{13} \\ \hline X_{21} & X_{22} & X_{23} \\ \hline X_{31} & X_{32} & X_{33} \\ \hline \end{array} \begin{array}{|c|c|} \hline W_{11} & W_{12} \\ \hline W_{12} & W_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline O_{11} & O_{12} \\ \hline O_{12} & O_{22} \\ \hline \end{array}$$

Figura 2.11.- Ejemplo de convolución.

Atrous Convolution o Convolución Dilatada. Este tipo de convoluciones añade un nuevo parámetro llamado “ratio de dilatación” que define la separación entre píxeles en el kernel (ver figura 2.12).

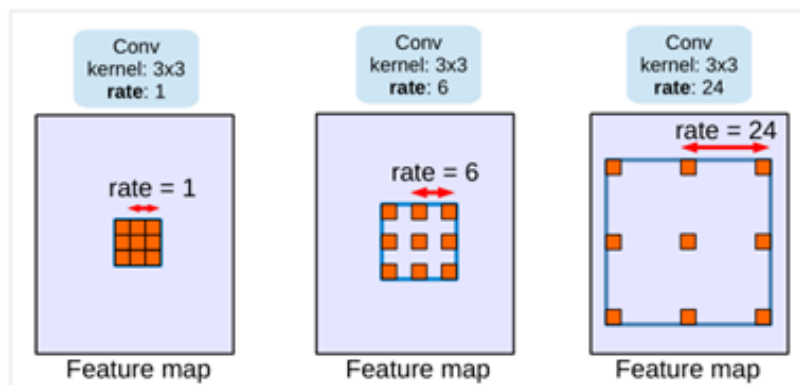


Figura 2.12.- Ejemplo de la variación del parámetro de dilatación.^[11]

El objetivo de la atrous convolution es aumentar el campo de visión sin alterar el coste computacional de la red. En la figura 2.13 se puede observar un ejemplo de como se obtienen los elementos para el cálculo de este tipo de convolución para un ratio de dilatación de seis.

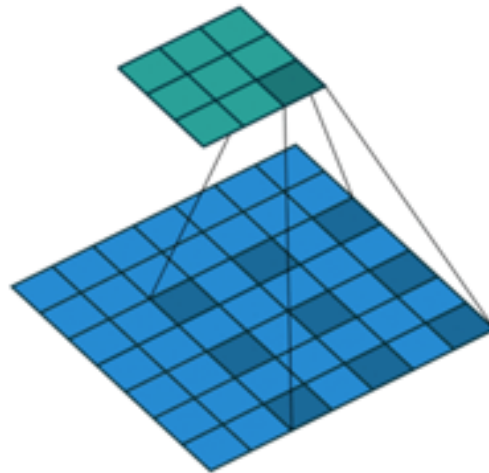


Figura 2.13.- Atrous Convolution.^[12]

Depthwise separable convolution o Convolución separable en profundidad. El objetivo de este tipo de capa convolucional es el de reducir el coste computacional manteniendo los resultados obtenidos. La convolución separable en profundidad se calcula aplicando solamente un filtro por cada canal de entrada, como se puede observar en la figura 2.14 (a) y utilizando una convolución por punto (figura 2.14 (b)), es decir, por píxel, para combinar los mapas de características obtenidos. Esta técnica se puede combinar con la convolución dilatada (atrous convolution) obteniendo finalmente el método de convolución de la figura 2.14 (c).

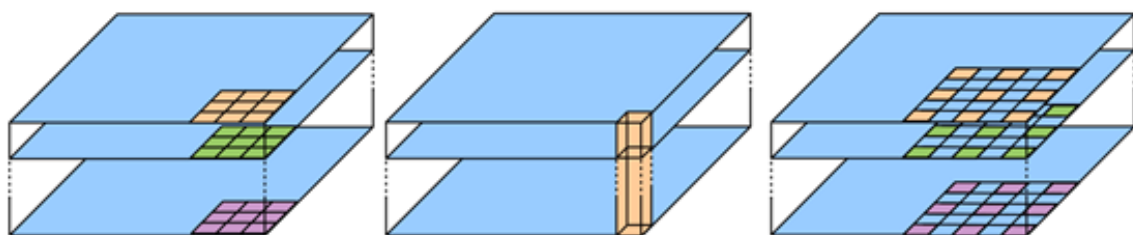


Figura 2.14.- (a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv.^[13]

Convoluciones transpuestas, Up-Convolution o deconvoluciones. En este tipo de convolución se separan los píxeles mediante el uso de padding de la forma mostrada en la figura 2.15. Este proceso sirve meramente para obtener una salida de

mayor tamaño que la entrada, mezclando así el proceso de escalado o upsampling con el de convolución. Este proceso NO es el inverso matemático del visto en la convolución convencional.

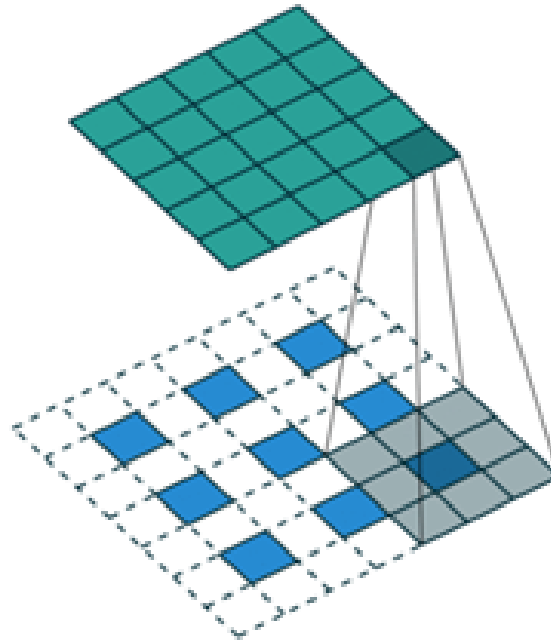


Figura 2.15.- Convolución transpuesta.^[12]

(ASPP) Atrous Spatial Pyramid Pooling. En la figura 2.16 se muestra un ejemplo de ASPP o Atrous Spatial Pyramid Pooling en el que se clasifica un píxel del centro de la imagen, representado de color naranja, utilizando convoluciones dilatadas con un ratio de dilatación de 6, 12, 18 y 24, para obtener diferentes escalas. Estas escalas están representadas cada una de un color diferente en el mapa de características.

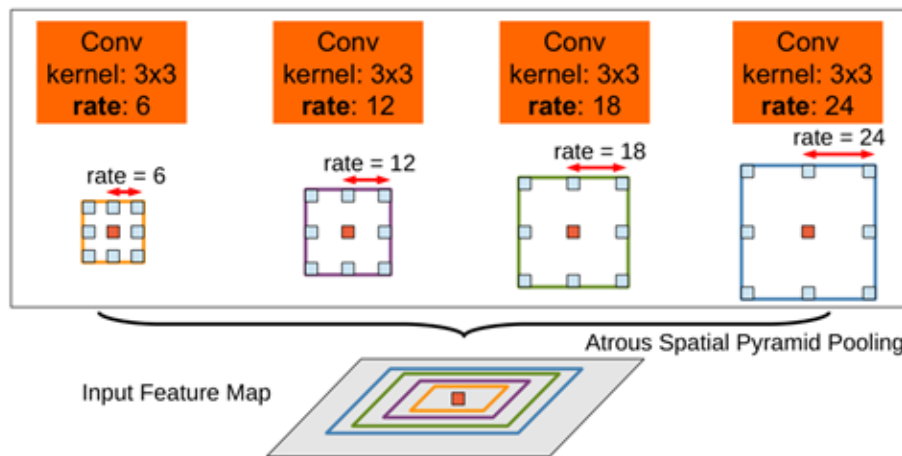


Figura 2.16.- Ejemplo de ASPP.^[14]

La salida de estas convoluciones se concatena y seguidamente se combina mediante un filtro de dimensiones 1×1 , con el objetivo de obtener una mejor representación de las características independientemente de su escala.

2.4.4.- Redes backbone

Las redes neuronales convolucionales mas complejas, como es el caso de las redes para la detección de objetos o las destinadas a la segmentación semántica, generalmente, utilizan como base una red neuronal convolucional de clasificación de imágenes, conocida como red backbone o feature extractor, sobre la que construir el resto de la arquitectura de la red, de forma que, se pueden reutilizar los pesos del entrenamiento de estas redes para ahorrar tiempo de entrenamiento en esta nueva red. Algunas de estas redes son VGG-16, Resnet, Xception, Mobilenet, etc.

2.4.5.- UNet

U-NET ^[15] es una red completamente convolucional, es decir, que no tiene capas densas, especializada en la segmentación semántica. Originalmente creada para la segmentación de células en imágenes biomédicas.

Debido a que no se usan capas densas, y mediante la estrategia de superposición de baldosas mostrada en la figura 2.17, se puede realizar la segmentación en imágenes de tamaños arbitrarios. Para predecir los píxeles de las regiones del borde las imágenes, el contexto faltante se extrapola mediante el reflejo de la imagen de entrada.

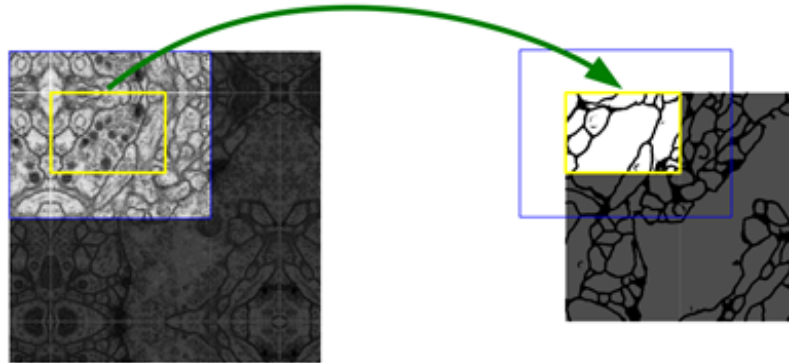


Figura 2.17.- Estrategia de superposición de baldosas.^[15]

El diseño de la red se puede apreciar en la figura 2.18. Este esquema se ha obtenido de la documentación oficial de la publicación [15]. Consiste en un camino de contracción o codificación de características para capturar el contexto, y un camino simétrico de expansión o decodificación que se encarga de obtener la localización precisa. Debido a esta forma descendente y ascendente en forma de “U” se le da el nombre de U-Net. Esta red contiene veintitrés capas convolucionales.

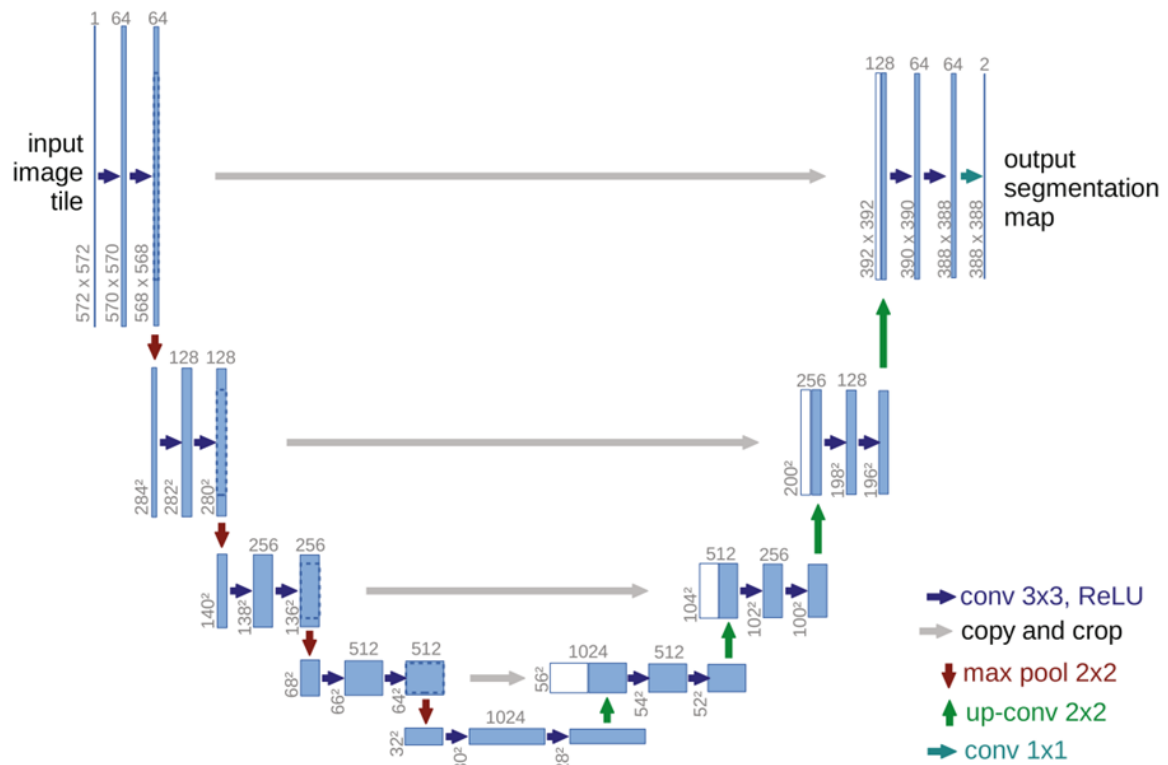


Figura 2.18.- Arquitectura de UNet.^[15]

El camino de contracción o codificación, es similar a la arquitectura clásica de las redes neuronales convolucionales. Consiste en la aplicación de dos capas convolucionales con un tamaño de filtro o kernel de 3×3 , sin padding, cada una de ellas seguida por una función de activación ReLU y una operación de Maxpooling de 2×2 con stride 2 para el escalado de los canales o mapas de características. Con cada escalado el número de canales de características se dobla. La profundidad de la red se mide según la cantidad de operaciones de Maxpooling realizadas, en este caso cuatro, aunque se pueden añadir o quitar niveles de profundidad fácilmente.

Cada nivel de la red consiste en dos capas convolucionales, en donde, al no existir padding el tamaño de la imagen disminuye en 4×4 píxeles. La flecha de color rojo indica el proceso de Maxpooling que divide a la mitad el tamaño de la imagen.

En cada nivel del camino de expansión o decodificación de la arquitectura se dispone de un aumento del escalado del mapa de características mediante una convolución

transpuesta con un filtro o kernel de tamaño 2×2 que reduce a la mitad el número de canales de características, una concatenación con el mapa de características del camino descendente del mismo nivel, y dos convoluciones transpuestas con un filtro o kernel de tamaño $\times 3$, cada una seguida por una función de activación ReLU. El mapa de características obtenido del camino de codificación es necesario debido a la pérdida de los píxeles del borde en cada convolución, se utiliza para ayudar en la localización.

Como capa final se encuentra una convolución con filtro o kernel de tamaño 1×1 usada para mapear cada vector de características al número deseado de clases, para ello utiliza tantos filtros o kernels como número de clases. Esta arquitectura posee un gran número de canales de características que permiten a la red propagar la información del contexto a las capas de resoluciones o tamaños mayores. Como consecuencia, el camino de expansión o decodificación es mas o menos simétrico al de contracción o codificación.

2.4.6.- SegNet

En [32], [33] y [16] se desarrolla SegNet, una red neuronal FCN para la segmentación semántica. El núcleo de esta red está basado en una arquitectura encoder-decoder, por lo que posee un camino de contracción y un camino de expansión y una capa final softmax para la clasificación píxel a píxel.

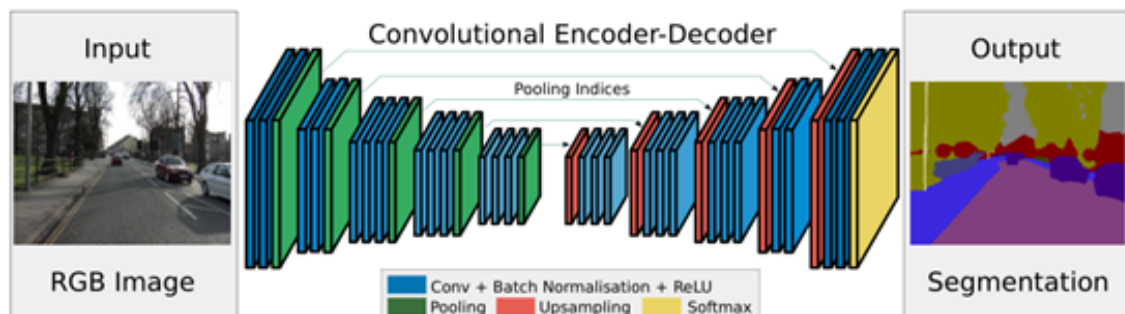


Figura 2.19.- Arquitectura de SegNet.^[16]

La arquitectura de la red codificadora es idéntica topológicamente a las trece capas convolucionales de la red VGG16. El rol del camino de decodificación consiste en mapear el mapa de características de baja resolución del codificador a un mapa de características con la resolución completa de la imagen de entrada para una clasificación píxel a píxel. La novedad de SegNet consiste en la forma en la que el camino decodificador escala los mapas de características. En concreto, se utilizan “pooling indices” (ver figura 2.20) en el proceso de max-pooling para realizar un escalado no lineal, reduciendo la cantidad de memoria necesaria para entrenar esta red.

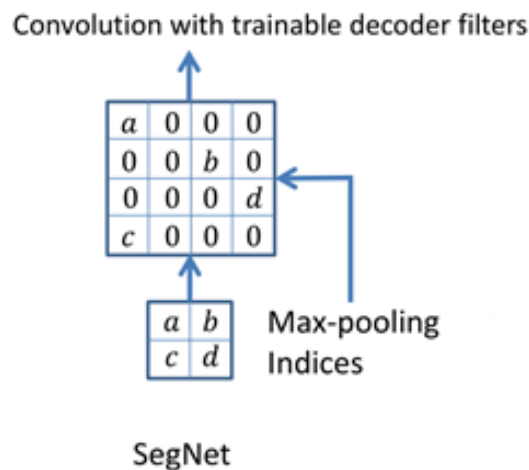


Figura 2.20.- Max-Pooling Indices en SegNet.^[16]

2.4.7.- DeeplabV3+

DeepLabV3+ [13] es una red especializada en la segmentación semántica y desarrollada por Google. Esta red es utilizada en tareas de segmentación en las cámaras de los smartphones “Pixel” de Google. En la imagen (a) de la figura 2.21 se observa la imagen capturada por el smartphone sin ningún tipo de retoque o postprocesado, mientras que en la imagen (b) se puede observar el uso de la segmentación semántica de la red DeepLabV3+ para la separación de la persona y el fondo de la imagen haciendo posible que se pueda aplicar un desenfoque sobre el fondo y estilizar la imagen resultante.



Figura 2.21.- Ejemplo de uso en smartphones Pixel.(a) Imagen sin segmentar
,(b) imagen segmentada.^[17]

DeepLabV3+ es la cuarta versión de la red DeepLab por lo que para entender mejor esta versión se resumen a continuación los cambios mas relevantes de cada versión.

DeepLabV1 [34]: Utiliza convoluciones transpuestas para controlar la resolución a la que se computan las características de la red.

DeepLabV2 [14]: Se utiliza ASPP o Atrous Spatial Pyramid Pooling para segmentar objetos de forma robusta en múltiples escalas utilizando filtros con múltiples tasas de muestreo y campos de visión efectivos.

DeepLabV3 [11]: Se aumenta el módulo ASPP con características a nivel de imagen para capturar un mayor rango de información. También se incluyen parámetros para la normalización de lotes (Batch Normalization) para facilitar el entrenamiento. En concreto, se aplica una convolución transpuesta para extraer las características de la salida o “output” en diferentes pasos o “strides” durante el entrenamiento y la



evaluación, que permite el entrenamiento con normalización de lotes con un stride de 8 o 16, alcanzando un alto rendimiento con stride de 8 durante la evaluación.

DeepLabV3+ [13]: Se extiende DeepLabV3 para incluir un simple, pero efectivo, modulo decodificador para refinar los resultados de la segmentación, especialmente sobre las fronteras de los objetos. Además, en esta estructura codificador-decodificador se puede controlar de forma arbitraria la resolución de las características extraídas por el decodificador utilizando convoluciones transpuestas para balancear precisión y tiempo de ejecución.

AutoDeepLab [35]: Mediante la técnica conocida como AutoML o AutoMachineLearning, se obtiene una arquitectura de la red Deeplab, basada en DeepLabV3+, optimizada para el conjunto de datos concreto que se va a utilizar para el entrenamiento. Su coste computacional es muy elevado.

Comúnmente las redes de segmentación semántica utilizan una arquitectura ASPP, o Encoder-Decoder (figura 2.22 (Izda) y (Ctro) respectivamente). Mientras que las arquitecturas basadas en ASPP obtienen mejores resultados en objetos de diferentes escalas, la precisión de las fronteras de los objetos disminuye. Así mismo, las arquitecturas basadas en Encoder-Decoder obtienen peores resultados en objetos de diferentes escalas, pero su precisión en la clasificación de las fronteras de los objetos aumenta. Por estos motivos la implementación de DeepLabV3+ combina ambas arquitecturas para obtener los beneficios del reconocimiento de objetos en múltiples escalas sin perder calidad en la clasificación de las fronteras. Esta arquitectura se puede observar en la figura 2.22 (Dcha) bajo el nombre de “Encoder-Decoder with Atrous Convolution”.

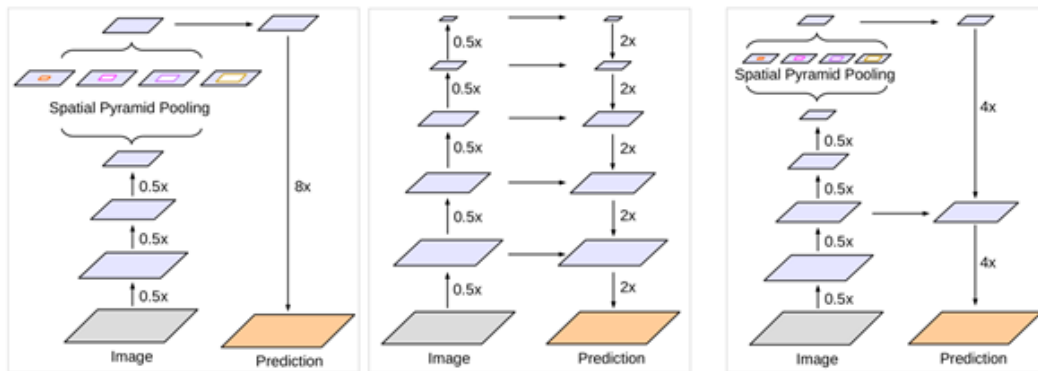


Figura 2.22.- (Izda) Spatial Pyramid Pooling (Ctro) Encoder-Decoder (Dcha)
Encoder-Decoder con Atrous Conv.^[13]

Además, como se puede observar, el camino de expansión o decodificador es más simple que en las arquitecturas Encoder-Decoder convencionales, esto se justifica como un intento para reducir el impacto computacional que sufre la red al añadir el módulo ASPP.

DeepLabV3+ integra una arquitectura Encoder-Decoder sobre la arquitectura ASPP existente de DeepLabV3 (ver figura 2.23). El módulo de Encoder se encarga de codificar la información contextual multi-escala aplicando capas de “atrous convolution” o convoluciones dilatadas, en múltiples escalas, mientras que el módulo Decoder refina los resultados de la segmentación al aumentar la nitidez de las fronteras de los objetos mediante el aumento de la resolución de la imagen. Esta parte de la red es similar a las redes convoluciones utilizadas comúnmente para tareas de clasificación, por esta razón, se utilizan redes existentes con una pequeña modificación al final de las mismas para eliminar las capas densas y concatenar el mapa de características de la salida con el módulo de Decoder o de decodificación. Como se explicaba en apartados anteriores, las redes utilizadas en este contexto se les conoce como redes “backbone”. En concreto, en la publicación oficial de esta red [13] se realizan pruebas con las redes “Resnet-101” y “Xception”.

La arquitectura del módulo Decoder es mucho más simple que en otras redes del estilo Encoder-Decoder, como puede ser U-Net, ya que en este caso no se utilizan convoluciones transpuestas, sino que simplemente se utiliza un escalado bilineal para

aumentar el tamaño cuatro veces, sobre el mapa de salida del módulo Encoder. Seguidamente se realiza una convolución con filtro o kernel de tamaño 1×1 para reducir la profundidad o número de canales del mapa de características, se añaden capas de convolución filtros o kernels de tamaño 3×3 para refinar las características de la nueva imagen y finalmente se vuelve a realizar otra operación de escalado bilineal para aumentar el tamaño otras cuatro veces más, obteniendo una imagen final del mismo tamaño que la introducida. Se utiliza un valor de output stride (también conocido como Downsampling Factor) de 16 debido a que se considera que un valor de 8 tiene un gran coste computacional a pesar de que se produce un ligero aumento en la precisión.

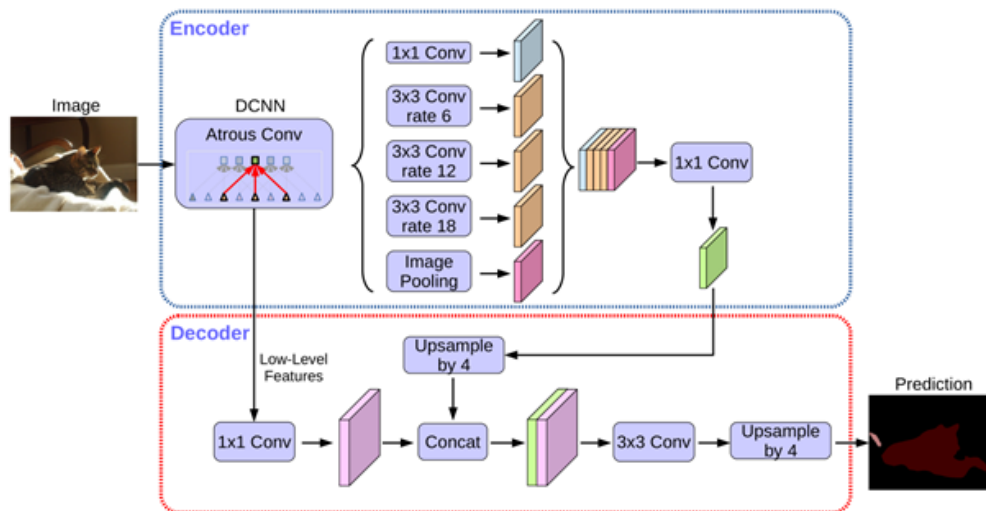


Figura 2.23.- Arquitectura de la red DeepLabV3+.^[13]

2.5.- Métricas

En esta sección se detallan las diferentes métricas encontradas en la literatura tanto como para la clasificación de cultivos, como para la segmentación semántica.

- **TruePositives (TP)**, son aquellas predicciones que se clasifican de forma correcta. En la figura 2.24, se muestra un ejemplo de los diferentes resultados de la predicción para una clasificación binaria.



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figura 2.24.- Ejemplo de los diferentes resultados de la predicción de una muestra para clasificación binaria.^[18]

- **FalsePositives (FP)**, son aquellas predicciones que se clasifican incorrectamente, es decir, se clasifican como de otra clase.
- **FalseNegatives (FN)**, son aquellas muestras que no obtienen una predicción cuando si que se deberían clasificar, es decir, que no se clasifican como de ninguna de las clases pero que tendrían que tener asociada una clase.
- **TrueNegatives(TN)**, son aquellas muestras que no obtienen una predicción y que no deben tenerla.
- **Recall**, se calcula como la cantidad de píxeles predichos correctamente entre el total de píxeles reales de esa clase. También conocido como **Producer's Accuracy (PA)** en el ámbito de clasificación de cultivos.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

- **Precision**, se calcula como la cantidad de píxeles predichos correctamente entre el total de píxeles predichos de esa clase. También conocido como **User's Accuracy (UA)** en el ámbito de clasificación de cultivos.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- **GlobalAccuracy (GA) / Pixel Accuracy (PA) / Overall Accuracy (OA)**, es el porcentaje de píxeles en la imagen que se han clasificado correctamente. Esta métrica tiene la problemática de que sí, por ejemplo, en una imagen el 95 % de los píxeles son de una clase, si se predice el 100 % de los píxeles como de esa misma clase, la precisión será del 95 %, lo que a primera vista parece un buen resultado, sin embargo, no se está realizando una segmentación en la imagen. Se calcula como la cantidad de píxeles correctamente clasificados entre el total de

píxeles del conjunto de datos.

$$GA = \frac{TruePositives}{TruePositives + FalseNegatives + TrueNegatives + FalsePositives}$$

- **Cohen's kappa coefficient of agreement (k)**, es una métrica para calcular la GlobalAccuracy teniendo en cuenta la probabilidad de que se acierte de casualidad. Para ello p_0 es la accuracy, y p_e la probabilidad de acierto por casualidad. De esta forma se obtiene una métrica más robusta.

$$k = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$$

- **Intersection-Over-Union (IOU)**, esta métrica obtiene el porcentaje del área de superposición entre el área de la unión. En este caso, para la segmentación semántica se superpone la máscara predicha sobre la máscara real (Ground Truth) y se calcula el IoU para cada una de las clases de forma similar a la vista en la figura 2.25. Finalmente, se hace la media de estos resultados obteniendo un valor final de IoU que representa la calidad de la localización de las predicciones. También conocido como **Índice de Jaccard**


$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figura 2.25.- Cálculo del IoU.^[19]

- **F-Score**, es una media armónica que combina los valores de la precisión y de recall que se utiliza para determinar en un único valor la calidad de la clasificación. Esta métrica se obtiene de la división de dos veces el área de la intersección, sobre el número total de píxeles en ambas imágenes (ver figura 2.26). Para estos cálculos se superpone la máscara predicha sobre la máscara real (Ground Truth) de forma similar al cálculo del IoU. Esta métrica se utiliza para indicar la calidad de la predicción de las fronteras de las clases, y tiende a correlacionarse mejor con la

evaluación cualitativa humana que la métrica IoU. También se le conoce como **F1-Score**, **Dice Coefficient**, **BF-Score**, o **media armónica**.

$$FScore = \frac{2 * Precision * Recall}{Precision + Recall}$$

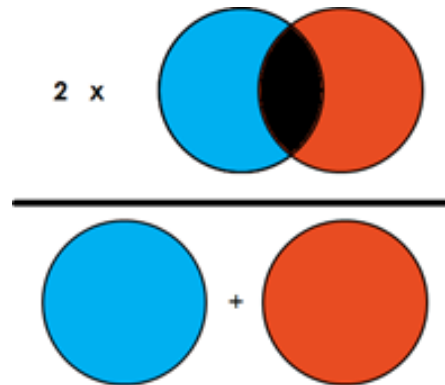


Figura 2.26.- Cálculo del F-Score.^[20]

- **Matriz de confusión**, está compuesta por la cantidad de píxeles predichos en cada clase (por filas) y la cantidad de píxeles reales de cada clase (por columnas) obteniendo así los **True Positive** en la diagonal de la matriz. Los **False Positive** se obtienen de la suma del resto de casos de cada fila, y los **False Negative** del resto de los casos de cada columna. A partir de las columnas se puede obtener el **Recall** para cada clase, dividiendo los píxeles True Positive entre la suma de los True Positive y los False Negative (TP+FN), y la Precisión de cada clase mediante la división de los píxeles True Positive entre el total de píxeles True Positive y False Positive (TP + FP). Esta explicación se puede entender de forma gráfica observando la figura 2.27. La utilidad de esta métrica es la de distinguir que clases son aquellas a las que se les conoce como “clases confusas”, que son las clases que identifican individuos incorrectamente a otra clase y viceversa.



Matriz de confusión						
	# px predichos	PSPRPA	EDZUCA	TAVI	BACKGROUND	PRECISION
# px reales		6,887,820	1,158,558	15,525,451	8,540,811	
PSPRPA	7,931,712	4,479,889	182,125	954,412	2,315,286	<i>0.56481</i>
EDZUCA	3,263,102	648,267	752,077	1,163,114	699,644	<i>0.23048</i>
TAVI	15,141,556	432,644	150,600	12,902,427	1,655,885	<i>0.85212</i>
BACKGROUND	5,776,270	1,327,020	73,756	505,498	3,869,996	<i>0.66998</i>
RECALL		<i>0.6504</i>	<i>0.6491</i>	<i>0.8311</i>	<i>0.4531</i>	0.68523

Figura 2.27.- Ejemplo de matriz de confusión

- **Representación de cuerdas.** Es una forma de representar visualmente la información de una matriz de confusión. Muestra la proporción con la que las clases se confunden entre sí. Todas aquellas cuerdas que conectan entre las diferentes clases muestran la proporción de píxeles clasificados erróneamente y su proporción puede verse en la base de cada clase en el perímetro del círculo. Aquellas cuerdas conectadas con su misma clase, creando una especie de arco relleno, son aquellos píxeles clasificados correctamente. Esta forma de visualizar los datos permite ver en un vistazo si hay dos o más clases confusas, es decir, clases que se clasifican entre sí de forma errónea.

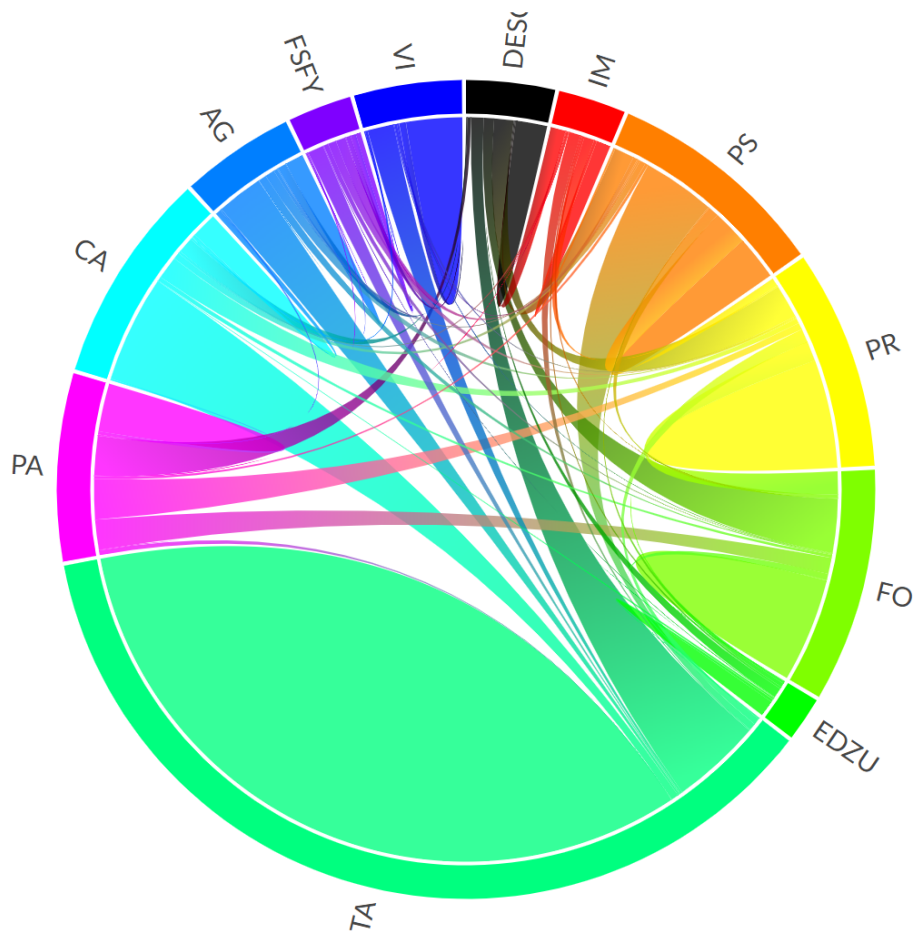


Figura 2.28.- Ejemplo de representación de cuerdas.

3. Información para el etiquetado de las imágenes

Para crear el etiquetado de las imágenes necesario para crear un conjunto de datos, es decir, el GroundTruth, se necesita una forma de clasificar cada píxel de la imagen en una clase. Se necesita que la información obtenida esté georeferenciada para poder combinar los datos del etiquetado con las imágenes de entrada a las redes, y así obtener un GroundTruth. Para esta tarea, en la literatura, generalmente se realizan expediciones “in situ” para recolectar información acerca del tipo de uso de la tierra de la zona destinada al estudio, sin embargo, esta técnica se suele apoyar en algún otro tipo de herramienta que detalla el uso de la tierra de forma genérica para el país donde se realiza el estudio. En el caso de España esta herramienta es SIGPAC (Sistema de Información Geográfica de Parcelas Agrícolas), que permite identificar de forma georeferenciada las diferentes parcelas declaradas por los agricultores y ganaderos. La información mas relevante, y que se va a utilizar, consta de: los puntos georeferenciados que conforman cada parcela y su clase o tipo de uso de la tierra. De esta forma, obtendremos los datos necesarios para generar un GroundTruth preciso, y ya que esta información es gratuita y se encuentra disponible en la red, supone un gran ahorro frente a realizar expediciones.

En la figura 3.1 se muestra la interfaz del visor oficial de SIGPAC¹, en la que se dispone del mapa del mundo con un zoom centrado en España. Para visualizar las diferentes parcelas SIGPAC se necesita utilizar el menú que se encuentra a la derecha (figura 3.2). Para poder utilizar estas opciones hay que hacer zoom hasta cierto punto. Las imágenes de este visor provienen tanto de Sentinel-2 para visualizaciones mas altas, como de ortofotos PNOA para niveles de zoom mas elevados.

¹Enlace para acceder al visor oficial de SIGPAC: <http://sigpac.mapama.gob.es/fega/visor/>.

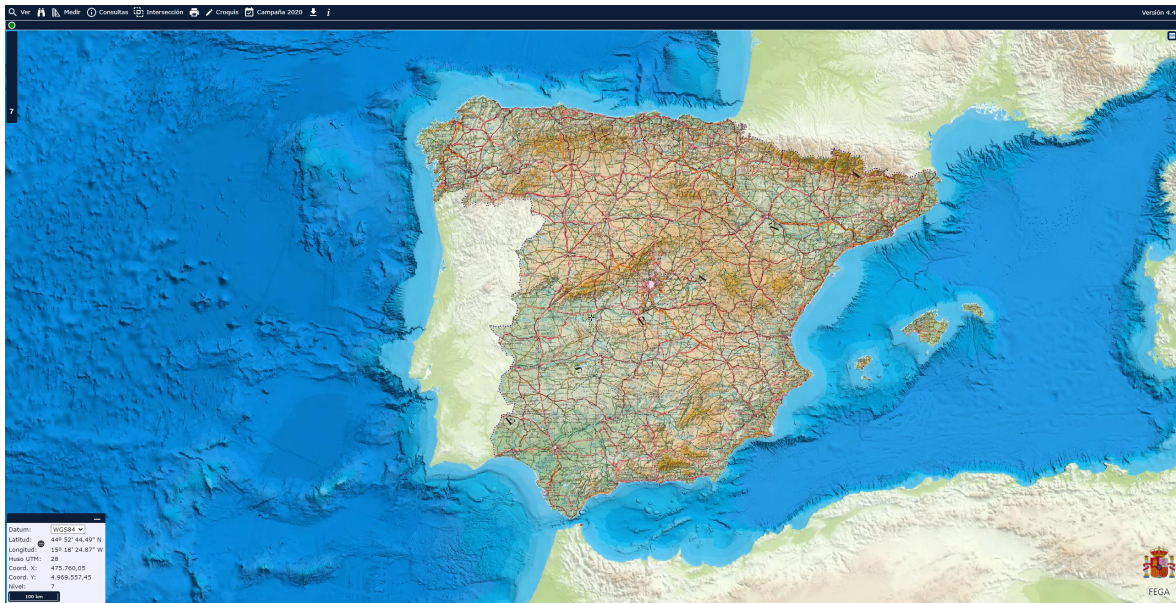


Figura 3.1.- Vista general del visor SIGPAC.

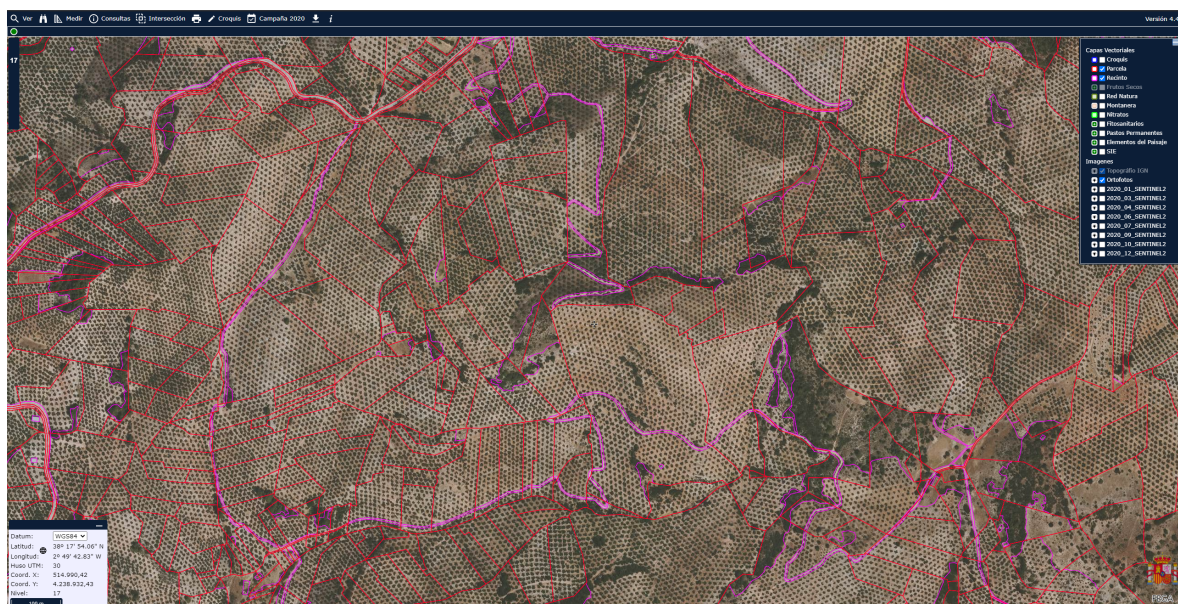


Figura 3.2.- Zoom con visualización de recintos y parcelas del visor SIGPAC.

Toda la información para la generación del GroundTruth de los conjuntos de datos de este Trabajo de Fin de Máster se apoya en la información de parcelas proporcionada por SIGPAC. Cabe destacar que SIGPAC puede contener errores en la clasificación de las parcelas, por ejemplo, en aquellos casos en los que la realidad no se ajuste al tipo de cultivo declarado en el sistema, por lo tanto, el GroundTruth obtenido no será perfecto



y contendrá siempre cierto grado de error. Dado que no se ha encontrado un sistema mejor para la obtención de estos datos, ya que etiquetar las diferentes parcelas de forma manual es una alternativa inviable para conjuntos de datos de los tamaños que se manejan en este trabajo, se decide utilizar dicha información.

Los conjuntos de datos de SIGPAC, que almacenan la información necesaria de las parcelas, se obtienen en formato CSV a través de la empresa SERESCO, que se encarga de generar y compartir los ficheros necesarios.

En SIGPAC se dispone de treinta clases diferentes (ver tabla 3.1), de las cuales para la generación de los conjuntos de datos solamente se utilizarán trece. Estas trece clases se han escogido en base a las opiniones de expertos de la empresa SERESCO dada su relevancia para el mercado y su cantidad de parcelas en la geografía española. Las trece clases escogidas son las expuestas en la tabla 3.2.

DESCRIPCIÓN USO SIGPAC	USO SIGPAC	DESCRIPCIÓN USO SIGPAC	USO SIGPAC
Asociación Cítricos-Frutales	CF	Improductivos	IM
Asociación Cítricos-Frutales de Cáscara	CS	Invernaderos y cultivos bajo plástico	IV
Asociación Cítricos-Viñedo	CV	Olivar	OV
Asociación Frutales-Frutales De Cáscara	FF	Olivar - Frutal	OF
Asociación Olivar-Cítricos	OC	Pastizal	PS
Cítricos	CI	Pasto Arbustivo	PR
Corrientes y Superficies de Agua	AG	Pasto con Arbolado	PA
Edificaciones	ED	Tierras Arables	TA
Elemento del paisaje	EP	Viales	CA
Forestal	FO	Viñedo	VI
Frutales	FY	Viñedo - Frutal	VF
Frutos Secos	FS	Viñedo - Olivar	VO
Frutos Secos y Olivar	FL	Zona Censurada	ZV
Frutos Secos y Viñedo	FV	Zona Concentrada no incluida en la Ortofoto	ZC
Huerta	TH	Zona Urbana	ZU

Tabla 3.1.- Clases disponibles en SIGPAC.



Uso
IM (IMPRODUCTIVOS)
PS (PASTIZAL)
PR (PASTO ARBUSTIVO)
FO (FORESTAL)
ED (EDIFICACIONES)
ZU (ZONA URBANA)
TA (TIERRAS ARABLES)
PA (PASTO CON ARBOLADO)
CA (VIALES)
AG (CORRIENTES Y SUPERFICIES DE AGUA)
VI (VIÑEDO)
FY (FRUTALES)
FS (FRUTOS SECOS)

Tabla 3.2.- Clases utilizadas de SIGPAC.

La metodología seguida para la selección de imágenes consiste en usar varias hojas MTN50 de las zonas de mayor interés por ser donde mas cultivos de las clases deseadas hay (en la figura 3.3 se muestra un ejemplo de la disposición de las hojas MTN50 sobre la geografía española). Seguidamente, se descargan las imágenes de PNOA o Sentinel correspondientes a esas hojas y se obtiene la información de SIGPAC de las parcelas contenidas en las coordenadas geográficas de dichas hojas. El mapa MTN50 cubre la totalidad de la geografía española.

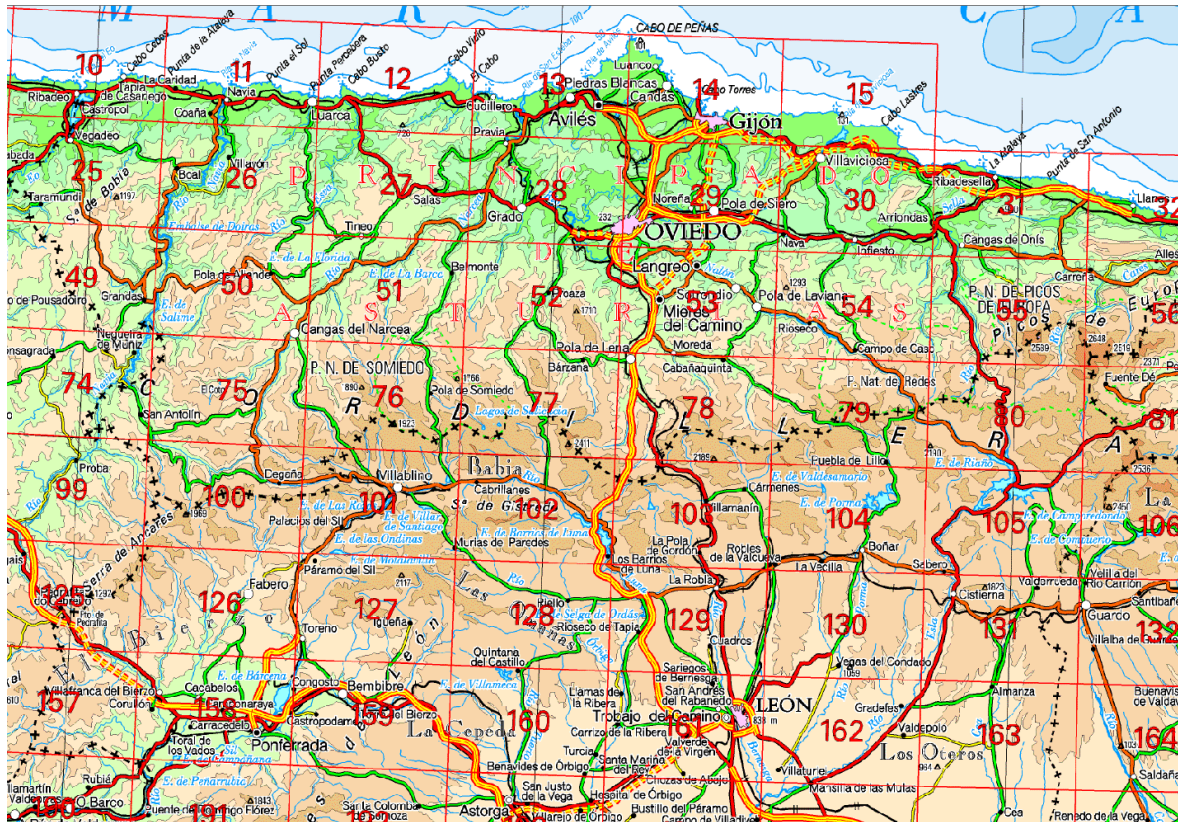


Figura 3.3.- Ejemplo de hojas MTN50 sobre parte de la geografía española.

3.1.- Información de SIGPAC para el conjunto de datos PNOA

El conjunto de datos PNOA consiste de imágenes de 256×256 píxeles en formato GEOTIFF recortadas de las imágenes de gran tamaño de PNOA de formato ECW. Estas imágenes contienen tres canales que son los correspondientes al RGB y tienen un GSD de 0.25 metros/píxel. En PNOA se dispone de una imagen completa por cada región MTN50 por lo que se obtiene la información SIGPAC de ciertas hojas MTN50 para combinar las imágenes PNOA con estos datos fácilmente.

En este documento solamente se detalla el conjunto de datos de SIGPAC de la última versión del conjunto de datos utilizado para PNOA.



3.1.1.- Conjunto de datos 20200902

De cara a la creación de un conjunto de datos para la segmentación semántica mediante redes neuronales se seleccionan las once clases representadas en la tabla 3.3. Se ha decidido unificar en una sola clase las clases “EDIFICACIONES” y “ZONA URBANA” por su parecido entre sí y su bajo número de parcelas. De la misma forma se han unificado las clases “FRUTOS SECOS” Y “FRUTALES”. Finalmente, se ha eliminado el resto de clases por tener un bajo número de parcelas. Este conjunto de datos de SIGPAC consta de información parcial de las hojas MTN50: **0162, 0199, 0233, 0314, 0343, 0345, 0351, 0368, 0402, 0405, 0482**. Los datos representados en la tabla 3.3 se obtienen calculando la media del área en metros cuadrados de cada tipo de uso de la tierra. De igual forma, como este conjunto de datos esta destinado a imágenes PNOA y dichas imágenes constan de un GSD de 0.25 metros/píxel, podemos obtener los píxeles de ancho de cada parcela si estas fueran cuadradas.

Uso	Parcelas	m ²	m ² /parcela	Px ²
IM (IMPRODUCTIVOS)	410	119,730.23	292.0249608	73
PS (PASTIZAL)	1883	3,855,894.79	2047.740199	512
PR (PASTO ARBUSTIVO)	3467	19,840,550.11	5722.68535	1431
FO (FORESTAL)	472	6,058,851.29	12836.54934	3209
EDZU (EDIFICACIONES Y ZONA URBANA)	125	1,230,587.77	9844.702135	2461
TA (TIERRAS ARABLES)	4935	71,853,689.14	14560.01806	3640
PA (PASTO CON ARBOLADO)	220	10,987,744.86	49944.29483	12486
CA (VIALES)	943	4,627,064.27	4906.748957	1227
AG (CORRIENTES Y SUPERFICIES DE AGUA)	340	1,326,466.17	3901.371085	975
VI (VIÑEDO)	1759	11,142,286.72	6334.443843	1584
FYFS (FRUTALES Y FRUTOS SECOS)	93	262,730.16	2825.055431	706

Tabla 3.3.- Resumen conjunto de datos SIGPAC 20200902 para PNOA.

Finalmente, en la figura 3.4 se muestra una comparativa de la cantidad de píxeles de cada clase. En esta figura se puede observar una gran disparidad de la clase TA (TIERRAS ARABLES) respecto al resto de clases.

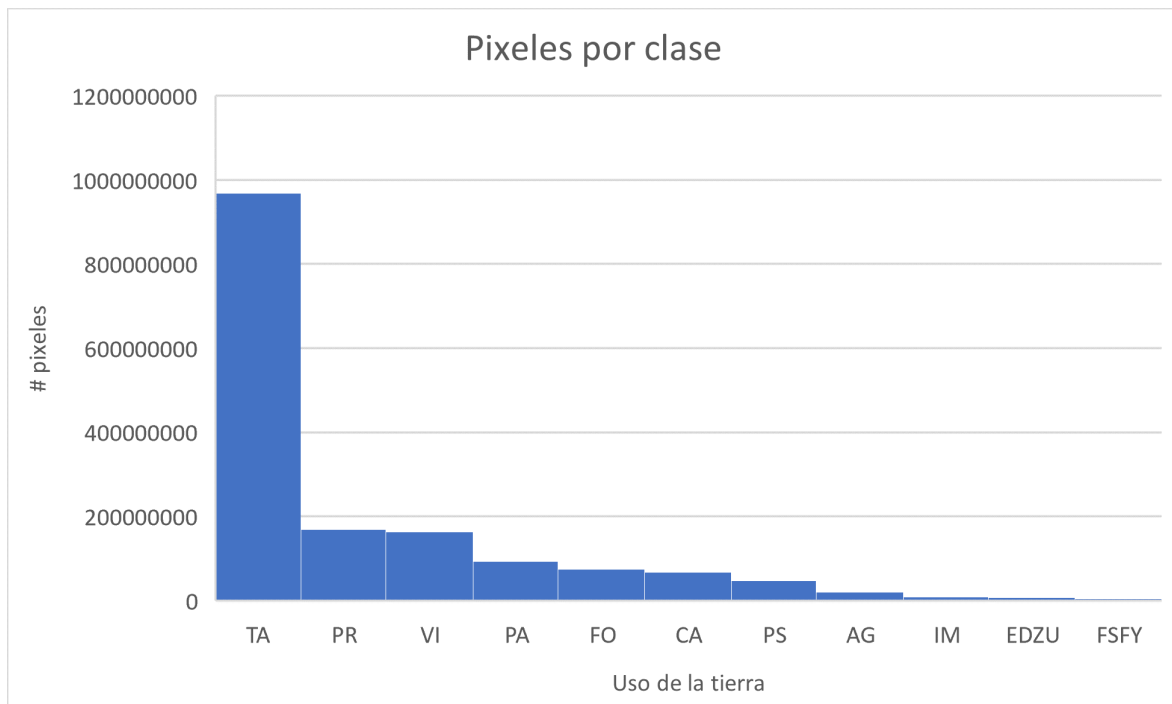


Figura 3.4.- Histograma de las clases por su número de píxeles del conjunto de datos 20200902.

3.2.- Información de SIGPAC para el conjunto de datos Sentinel-2

El conjunto de datos Sentinel-2 consiste de imágenes de 256×256 píxeles en formato GEOTIFF recortadas de imágenes de un tamaño superior del mismo formato. Estas imágenes pueden contener las trece bandas indicadas en el apartado 2.1.2, aunque para las experimentaciones presentadas en este documento se utilizan las diez bandas de la tabla 5.11. El resto de experimentaciones con diferentes bandas se dispone en el ANEXO A. Debido a que las imágenes de Sentinel-2 tienen un GSD de 10 metros/píxel se requiere de muchísima más información y área a cubrir que en el caso de PNOA. Por ello la información recogida de las hojas MTN50 es completa en lugar de obtener información parcialmente de dichas hojas. Se filtra solamente por las clases seleccionadas anteriormente en el conjunto de datos PNOA por los mismos motivos mencionados en dicho apartado y para poder comparar los resultados de ambos conjuntos de datos.



En este documento solamente se detalla el conjunto de datos de SIGPAC de la última versión del conjunto de datos utilizado para Sentinel-2.

3.2.1.- Conjunto de datos 20201209

Las hojas MTN50 utilizadas son las siguientes.

0105	0136	0158	0193	0194	0196	0235	0269	0307	0312
0345	0370	0374	0375	0376	0397	0398	0399	0407	0482
0162	0199	0233	0314	0343	0346	0351	0368	0402	0405
0161	0191	0191	0268	0306	0338	0423	0530	0555	

Este conjunto de datos de SIGPAC tiene un total de 3,169,100 parcelas, su desglose por clases se puede ver en la tabla 3.4

Clase	# Parcelas 20201209
PASTO ARBUSTIVO	885,709
TIERRAS ARABLES	850,846
PASTIZAL	662,457
EDIFICACIONES Y ZONA URBANA	216,552
FORESTAL	165,481
VIALES	103,944
VIÑEDO	98,669
PASTO CON ARBOLADO	88,133
IMPRODUCTIVOS	52,417
CORRIENTES Y SUPERFICIES DE AGUA	22,996
FRUTOS SECOS Y FRUTALES	21,896

Tabla 3.4.- Resumen conjunto de datos SIGPAC 20201209 para Sentinel-2.

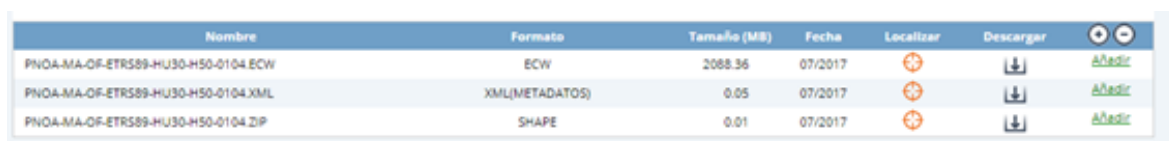
4. Experimentación con el conjunto de datos PNOA

4.1.- Obtención de las imágenes PNOA

Para realizar la descarga de las imágenes PNOA hay que acceder a la página del Centro de Descargas del CNIG¹.

Dichas imágenes son de gran tamaño cubriendo varios kilómetros, tienen una gran resolución (ej. $13,915 \times 9,680$ píxeles en formato ECW) y por tanto, cada imagen puede llegar a pesar hasta varios gigabytes (ej. 2.5 GB en formato ECW y unos 25 GB en formato GEOTIFF). Estas imágenes tienen una resolución del terreno de **0.25 metros por píxel**.

La opción de descarga recomendada “Ortofoto PNOA Máxima Actualidad” provee de las últimas imágenes obtenidas de las zonas seleccionadas, siendo estas en su mayoría entre los meses de julio y octubre, comúnmente del año 2017, y en algunos casos del 2018 o 2019. En la figura 4.1 se muestra un ejemplo de descarga de imágenes con esta opción.



Nombre	Formato	Tamaño (MB)	Fecha	Localizar	Descargar	
PNOA-MA-OF-ETRS89-HU30-H50-0104.ECW	ECW	2088.36	07/2017	📍	⬇️	⚙️
PNOA-MA-OF-ETRS89-HU30-H50-0104.XML	XML(METADATOS)	0.05	07/2017	📍	⬇️	⚙️
PNOA-MA-OF-ETRS89-HU30-H50-0104.ZIP	SHAPE	0.01	07/2017	📍	⬇️	⚙️

Figura 4.1.- Descarga de imágenes en “Ortofoto PNOA Máxima Actualidad”.

Otra posible alternativa sería la de “Ortofotos históricas del PNOA”, donde se pueden encontrar imágenes PNOA con fechas de 2005, 2007, 2011, 2014 y 2017 entre otras. También existe la posibilidad de obtener las imágenes de “Ortofotos de SIGPAC”, sin embargo, estas imágenes están fechadas entre el 2001 y 2003 por lo que están desactualizadas respecto a los datos de declaración del cultivo de la parcela del conjunto

¹Enlace a la página del Centro de Descargas del CNIG: <https://centrodedescargas.cnig.es/CentroDescargas/>



de datos de SIGPAC utilizado y a la hora de crear las máscaras GroundTruth se etiquetarán las imágenes de forma incorrecta.

4.2.- Obtención de los recortes de entrada

En este apartado se trata de obtener recortes de las imágenes PNOA completas, en donde se etiquetarán todas las parcelas a partir de un conjunto de datos de SIGPAC, con el objetivo final de obtener recortes de cada imagen PNOA y su respectiva máscara en imágenes de tamaños reducidos, de **256×256 píxeles**, para su uso en redes neuronales convolucionales destinadas a la segmentación semántica.

Como las redes neuronales no permiten como entrada imágenes de gran tamaño se recortan las imágenes PNOA en imágenes de menor tamaño, en este caso de 256×256 píxeles. Para realizar estos recortes se utiliza la función “gdal_retile” con los argumentos siguientes.

- **-csv <filename>**: Genera un fichero “.CSV” con las coordenadas geográficas UTM con el datum ETRS89.
- **-ps <x><y>**: Selecciona el tamaño de los recortes en píxeles.
- **-targetDir <dir>**: Selecciona el directorio donde almacenar los recortes.

Un ejemplo de uso de esta función se puede observar en el código 4.1.

Código 4.1.- Método gdal_retile para la generación de recortes.

```
gdal_retile PNOA.ECW -ps 256 256 -csv data.csv -targetDir C:\Recortes
```

Este método de recorte presenta el problema de que si se generan más de 128,000 recortes el programa finaliza su ejecución abruptamente sin crear todos los recortes necesarios. Para evitar este problema se ha decidido dividir las imágenes PNOA en dos “subimágenes” (figura 4.2), reduciendo así su tamaño a aproximadamente a la mitad. De esta forma la imagen pasa de tener un tamaño de 111,920×77,520 píxeles a dos subimágenes de 55,808×77,520 (figura 4.2a) y 56,112×77,520 (figura 4.2b) píxeles. La

división se ha realizado de forma que el ancho de la primera subimagen sea divisible entre 256 para evitar obtener recortes de menor tamaño.



(a) Subimagen 1



(b) Subimagen 2

Figura 4.2.- Imagen PNOA de la Hoja MTN50 0162 dividida en dos imágenes.

Para dividir la imagen PNOA completa ECW en las dos subimágenes TIF se utiliza el código 4.2, se ejecuta una vez para cada subimagen, modificando el valor de los píxeles de forma acorde a los tamaños deseados.

Código 4.2.- Método `gdal_translate` para la división en subimágenes.

```
gdal_translate -srcwin PixelDeAnchoInicial PixelDeAltoInicial PixelDeAnchoMaximo  
PixelDeAltoMaximo ImagenOriginal.ecw ImagenDestino.tif
```

Para realizar este proceso de forma automática se dispone del script “GetSubImages.py”.

Una vez obtenidas las subimágenes necesarias, se realizan los recortes de 256×256 píxeles con el comando “`gdal_retile`” como se menciona anteriormente. En la figura 4.3



se muestran tres ejemplos de estos recortes, cabe destacar que no hay solapamiento entre recortes.



(a) Recorte 1

(b) Recorte 2

(c) Recorte 3

Figura 4.3.- Ejemplo de recortes.

4.3.- Generación del GroundTruth para los recortes

En este apartado se trata de obtener máscaras para los recortes de imágenes PNOA generados en el apartado anterior, en donde se etiquetan todas las parcelas a partir de un conjunto de datos de SIGPAC.

Para la creación de las máscaras de los recortes se itera sobre el fichero “.CSV” con las parcelas Sigpac, se transforman a objetos “Geometry” de la librería de “gdal” mediante sus puntos o coordenadas, buscando para cada parcela su intersección con los diferentes recortes, convirtiendo estos recortes en otro objeto “Geometry” para poder utilizar la función “Intersecs” e “Intersection” y obtener los puntos de la intersección entre ambas. En el código 4.3 se muestra como se obtiene la intersección.

Código 4.3.- Cálculo de la intersección

```
# If there is an intersection continue to process it
if (parcelGeom.Intersects(tileGeom)):
    try:
        # Get the intersection geometry
        parcelIntersec = parcelGeom.Intersection(tileGeom)
```



Cuando se encuentra una intersección se obtienen los puntos de ésta y se traducen a píxeles en la imagen del recorte. Para traducir estas coordenadas en formato ETRS89 (EPSG:4258) a píxeles en el recorte se utiliza la función “gdalLocationInfo” que solo está disponible mediante consola y no en la librería de Python de “gdal”. Por este motivo, hay que llamar a la función de la consola de comandos desde Python y parsear su salida para obtener la traducción (ver código 4.4).

Código 4.4.- Traducción de coordenadas a píxeles

```
cmd = ['gdallocationinfo', '-  
l_srs', 'EPSG:4258', TILES+filename, str(xtemp_), str(ytemp_)]  
p = Popen(cmd, stdout=PIPE)  
p.wait()  
val = p.stdout.read()
```

Seguidamente se crea la máscara a partir de esos puntos con el uso de la función “Path” de la librería “Matplotlib” (ver código 4.5) y se guarda en disco en una carpeta diferente, pero manteniendo el mismo nombre que el recorte original.

Código 4.5.- Creación de la máscara

```
# Create Path from vertices  
path = Path(vertices)  
# Create a mesh grid for the whole image  
ym, xm = np.mgrid[:height, :width]  
# mesh grid to a list of points  
points = np.vstack((ym.ravel(), xm.ravel())).T  
grid = path.contains_points(points)  
# Mask with points inside a polygon  
mask = grid.reshape(height, width)
```

Para cada tipo de uso de la tierra se utiliza un color diferente (o escala de gris) en los píxeles de la máscara, para reconocer los diferentes tipos de cultivo. Además, pueden coexistir un número indeterminado de parcelas en la misma máscara debido a que se busca si ese recorte ya posee una máscara y de ser así pinta la parcela seleccionada sobre este mismo fichero.

Cabe destacar que para la creación de los objetos de clase “Geometry” de los recortes, hay que transformar las coordenadas del formato UTM ETR89 al Geodetic ETRS89



(EPSG:5258) mediante el código 4.6.

Código 4.6.- Creación del objeto Geometry con las coordenadas del recorte

```
ds = gdal.Open(TILES+filename)
width = ds.RasterXSize
height = ds.RasterYSize

# Get CRS from dataset
crs = osr.SpatialReference()
crs.ImportFromWkt(ds.GetProjectionRef())

# Create lat/long crs with ETR89 datum

crsGeo = osr.SpatialReference()
crsGeo.ImportFromEPSG(4258) # 4326 is the EPSG id of lat/long crs 4258
t = osr.CoordinateTransformation(crs, crsGeo)

# Transform coordinates to ETRS89 lat/long
y1, x1, z1 = t.TransformPoint(x1, y1)
y2, x2, z2 = t.TransformPoint(x2, y2)
```

Es importante tener en cuenta que no hace falta repetir todo este código para cada imagen ya que todos los recortes utilizan el mismo sistema de referencias, por lo que se puede ahorrar mucho tiempo de cómputo comprobando que no se repite este proceso innecesariamente y tener en cuenta que solamente se deben de ejecutar las líneas de transformación de las coordenadas “TransformPoint()”.

Para poder comparar las máscaras de los recortes de la imagen PNOA hay que visualizar las parcelas de la zona en el visor Sigpac. Para encontrar la parcela que se está buscando hay que rellenar los siguientes campos de la figura 4.4.

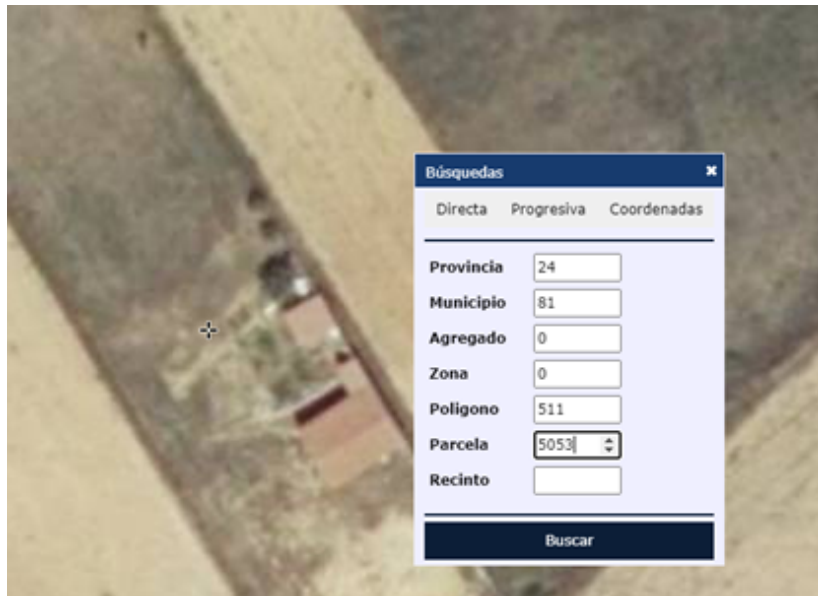


Figura 4.4.- Búsqueda de parcela en el visor de Sigpac.

Para representar las parcelas en el visor de Sigpac se debe seleccionar la capa “Recinto” en el menú de capas (figura 4.5). Una vez seleccionada la capa correspondiente se visualizarán las fronteras de las parcelas de color rosa.



Figura 4.5.- Capas del visor de Sigpac.

En la figura 4.6 se muestra el recorte de la imagen PNOA (figura 4.6a), la visualización de la imagen en el visor de SigPac (figura 4.6b), y su máscara (figura 4.6c). Esta máscara es calculada a partir de los datos del conjunto de datos de Sigpac.



Figura 4.6.- Comprobación del GroundTruth.

Esta máscara será la que se utilice en la creación del conjunto de datos para la segmentación semántica. El código de colores utilizado para la representación de las parcelas en la máscara es el correspondiente a la tabla 4.1, donde se señala el valor de los píxeles de la escala de grises. Para visualizar las imágenes que utilizan este código de colores se mapean sus valores a una escala de grises mas amplia (de 0 a 255) o a RGB. Se sigue este formato de escala de grises con valores de 0 a n-1, donde n es el número de clases, debido a que es el estándar a la hora de introducir estos datos a una red neuronal.



Clase de la parcela	Código de color
IM (IMPRODUCTIVOS)	0
PS (PASTIZAL)	1
PR (PASTO ARBUSTIVO)	2
FO (FORESTAL)	3
EDZU (EDIFICACIONES Y ZONA URBANA)	4
TA (TIERRAS ARABLES)	5
PA (PASTO CON ARBOLADO)	6
CA (VIALES)	7
AG (CORRIENTES Y SUPERFICIES DE AGUA)	8
FYFS (FRUTALES Y FRUTOS SECOS)	9
BACKGROUND	10

Tabla 4.1.- Código de colores de las máscaras GroundTruth.

4.4.- Redes neuronales convolucionales utilizables

En esta sección se decide que las experimentaciones con este conjunto de datos se llevarán a cabo con **DeepLabV3+** por ser una red mucho mas reciente que el resto de redes posibles, además esta red ha sido desarrollada por Google.

Debido a que las imágenes utilizadas en este conjunto de datos son imágenes RGB comunes, se puede aplicar cualquier red de segmentación semántica disponible. Se descartan redes como SegNet por su antigüedad y su nula aparición en la literatura de la segmentación semántica destinada a imágenes aéreas o satélites, y el resto de redes por su desconocimiento y baja representación en la literatura. Finalmente se descarta UNet debido a que en experimentaciones anteriores con las redes se observa una mejora superior al 15% y 20% en las métricas de MeanRecall y MeanPrecision respectivamente al utilizar DeepLabV3+ sobre imágenes aéreas de drones.



4.5.- Experimentación

Todas las experimentaciones realizadas con el conjunto de datos PNOA, así como sus conclusiones, se encuentran disponibles en los ANEXOS C y D, en este documento solamente se incluirán aquellos experimentos que presentan los mejores resultados para cada una de las implementaciones evaluadas.

Para la visualización de las máscaras GroundTruth y de las predicciones de la red se utiliza el código de colores de la tabla 4.2 para una visualización mas sencilla, en dicha tabla también se detallan los diminutivos utilizados para cada tipo de cultivo. La clase BACKGROUND se identifica de color negro.

Uso	R	G	B	Muestra
IM (IMPRODUCTIVOS)	255	0	0	
PS (PASTIZAL)	255	127	0	
PR (PASTO ARBUSTIVO)	255	255	0	
FO (FORESTAL)	127	255	0	
EDZU (EDIFICACIONES Y ZONA URBANA)	0	255	0	
TA (TIERRAS ARABLES)	0	255	127	
PA (PASTO CON ARBOLADO)	255	0	255	
CA (VIALES)	0	255	255	
AG (CORRIENTES Y SUPERFICIES DE AGUA)	0	127	255	
VI (VIÑEDO)	0	0	255	
FYFS (FRUTALES Y FRUTOS SECOS)	127	0	255	

Tabla 4.2.- Código de colores para visualización de las clases.

4.5.1.- Implementación DeepLabV3+ (Matlab)

Esta implementación de DeepLabV3+ es la ofrecida por Matlab. En la tabla 4.3 se muestran los parámetros de entrada del experimento para su comparación y replicación. En la tabla 4.4 se presentan los resultados globales del experimento, es decir, su GlobalAccuracy, MeanRecall, MeanPrecision y MeanIoU, mientras que en la tabla 4.5 se muestran las métricas de Recall, IoU y MeanBFScore por cada clase. En la tabla 4.6 se encuentra la matriz de confusión del experimento y en la tabla 4.7 su



versión normalizada. Finalmente, en la figura 4.7 se muestra el progreso de la función de loss y de la GlobalAccuracy durante las epochs del entrenamiento, y en la figura 4.8 se presentan algunas de las imágenes obtenidas en el test de la red. El resto de experimentos con esta implementación están disponibles en el ANEXO C.

Parámetros de entrada	
Parámetros del dataset	
Conjunto de entrenamiento	Train1Aumentado
Conjunto de test	Test1Aumentado
Parámetros de la red	
InputSize	[256 256 3]
Número de Clases	11
Network	Mobilenetv2
Downsampling Factor	8
Padding	Sí
Pesos en las clases	Median Frequency Weighting
Parámetros de entrenamiento	
Solucionador de red (solver network)	Adam
Epochs	60
BatchSize	32
LearningRate -Inicial	0.0005
LearnRateDropPeriod	-
LearnRateDropFactor	-
Gradient Clipping	-
Regularizacion L2	0.0001
Data Augmentation	Espejo en ejes X e Y
Shuffle	Si
Momentum	-
Duración del entrenamiento	17:17:08

Tabla 4.3.- Parámetros del experimento (Matlab E003-01).

GlobalAccuracy	MeanRecall	MeanPrecision	MeanIoU
0.87150	0.71623	0.73272	0.56793

Tabla 4.4.- Resultados globales del experimento PNOA DeepLabV3+ (Matlab E003-01).



Clase	Recall	IoU	MeanBFScore
IM	0.5992	0.3542	0.2849
PS	0.5544	0.3436	0.2776
PR	0.7997	0.5965	0.4833
FO	0.6725	0.6136	0.5240
EDZU	0.6071	0.5860	0.4520
TA	0.9339	0.9159	0.7634
PA	0.6563	0.6039	0.5679
CA	0.8330	0.4487	0.4690
AG	0.7072	0.3325	0.3528
FSFY	0.5690	0.5303	0.4903
VI	0.9462	0.9219	0.7437

Tabla 4.5.- Métricas de las clases (Matlab E003-01).



Matriz de confusión													
	IM	PS	PR	FO	EDZU	TA	PA	CA	AG	FSFY	VI	PRECISION	
# px reales													
IM	2153775	11475118	40717213	18251042	2020288	241382304	24567917	16901087	4950956	1009927	44315256		
PS	2779779	1290467	221956	18571	608348	241603	109239	115373	15416	12367	42813	0.46423367	
PR	13404844	6362366	2171345	271614	102813	3455746	71431	176764	154330	120983	353244	0.47463186	
FO	46429890	143280	1865963	32560973	9401	2686698	5034255	814537	292337	125349	183883	0.70129335	
EDZU	14026520	574	101840	12274328	0	181150	981647	164250	61951	21890	9766	0.87508006	
TA	1299125	38319	3708	127	1226525	11276	88	9255	4	715	500	0.94411623	
PA	230167858	94857	1813210	1085140	8665	225425272	120364	679473	357426	35100	402276	0.9793951	
CA	18255066	13918	785038	1040650	1495	88660	16124035	158224	19347	13194	192	0.88326358	
AG	28554243	310677	653638	259163	791972	45638	6505060	1741291	540793	43818	1284563	0.49303111	
FSFY	9079294	75761	380143	1004845	879529	13962	2218169	358956	517071	3500863	102573	0.38558758	
VI	648391	123	33819	2974	1745	26914	4155	785	0	574689	3187	0.88633093	
RECALL	43099273	21591	141592	92947	113217	3441	541756	187225	8389	34400	41932259	0.97292265	
	0.5992	0.5544	0.7997	0.6725	0.6071	0.9339	0.6563	0.8330	0.7072	0.5690	0.9462	0.87150187	

Tabla 4.6.- Matriz de confusión (Matlab E003-01).

Matriz de confusión normalizada													
	IM	PS	PR	FO	EDZU	TA	PA	CA	AG	FSFY	VI		
% px predichos													
IM	0.9444	0.599165187	0.009030495	0.005451159	0.001017531	0.301119444	0.001000914	0.00682637	0.00311412	0.01224544	0.0009661		
PS	0.9364	0.076241947	0.554448852	0.053327447	0.014882109	0.050890269	0.014316484	0.00290749	0.01045874	0.03117554	0.11979381	0.00797116	
PR	1.6337	0.066525055	0.162609483	0.7996857	0.148660772	0.004653297	0.011130468	0.204911176	0.04819436	0.05905373	0.12411689	0.00414943	
FO	0.7721	0.000266509	0.008874854	0.005627702	0.672527519	0	0.000750469	0.03995646	0.00971831	0.01251445	0.02167483	0.00022038	
EDZU	0.6271	0.017791552	0.000750145	9.10671E-05	6.95851E-06	0.607104037	4.67143E-05	3.5819E-06	0.0005476	8.0802E-07	0.00070797	1.1283E-05	
TA	1.3360	0.044042205	0.158012318	0.026650645	0.008003653	0.004288992	0.933893116	0.00489924	0.04020292	0.07220208	0.03475499	0.0090776	
PA	0.7674	0.006462142	0.000898727	0.019280249	0.057018662	0.000739994	0.000367301	0.65630452	0.00936176	0.0039082	0.01306431	4.3326E-06	
CA	1.4424	0.144247658	0.056961331	0.062852116	0.043393248	0.022589849	0.0269492	0.07087662	0.83297187	0.10914225	0.04338729	0.02898692	
AG	0.9391	0.035175912	0.03312759	0.024678629	0.048190618	0.006910896	0.009189443	0.01461076	0.03059395	0.70719419	0.02715246	0.00231462	
FSFY	0.5726	5.7109E-05	0.002947159	7.30404E-05	9.5611E-05	0	0.000111499	0.00016912	4.6447E-05	0	0.56904014	7.1917E-05	
VI	1.0288	0.010024724	0.012339045	0.002282745	0.006203317	0.001703223	0.00224439	0.00091404	0.01107769	0.00169463	0.03406187	0.94622626	

Tabla 4.7.- Matriz de confusión normalizada (Matlab E003-01).

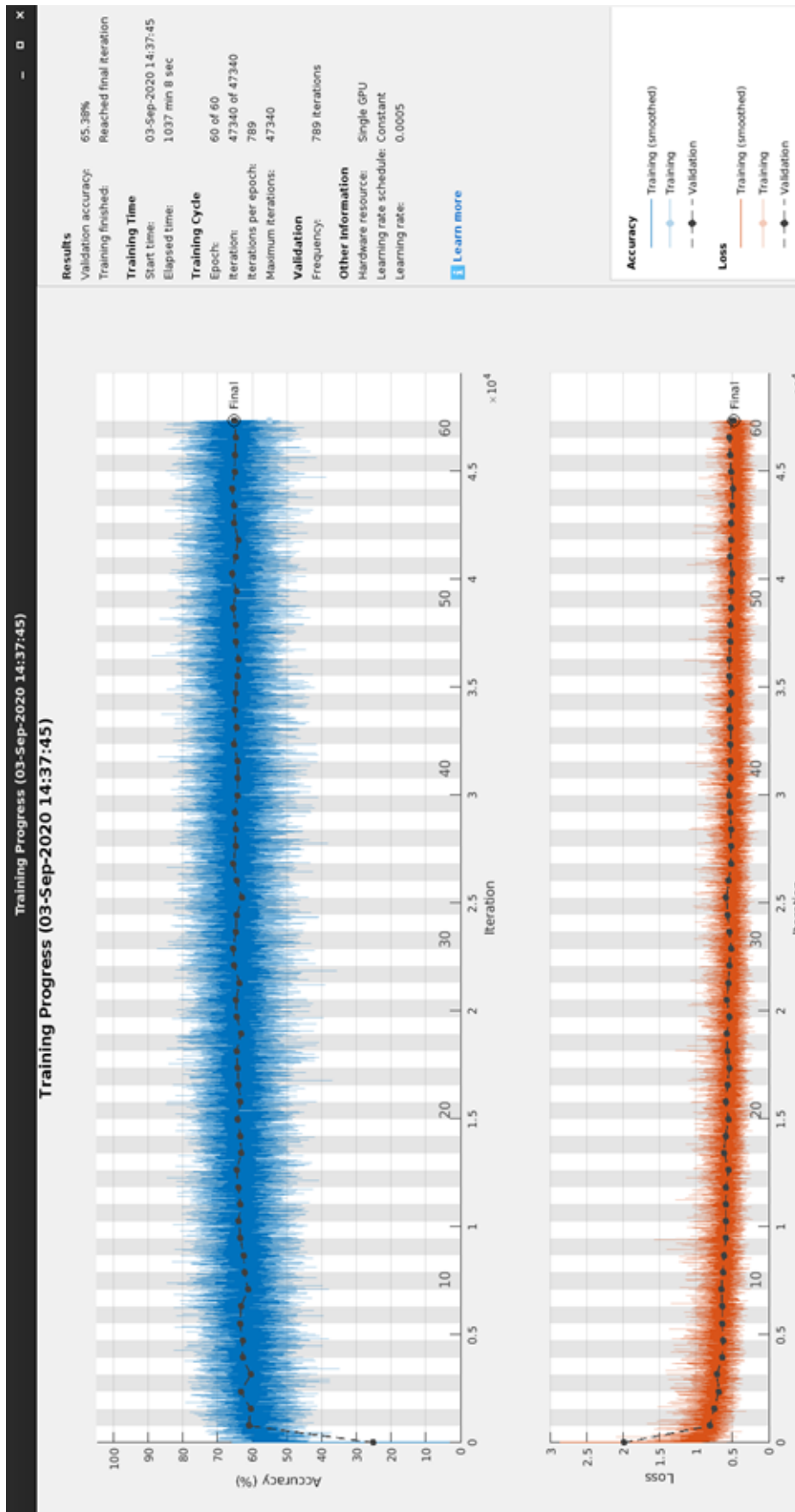


Figura 4.7.- Progreso de la GlobalAccuracy y Loss (Matlab E003-01).



Imágenes de test

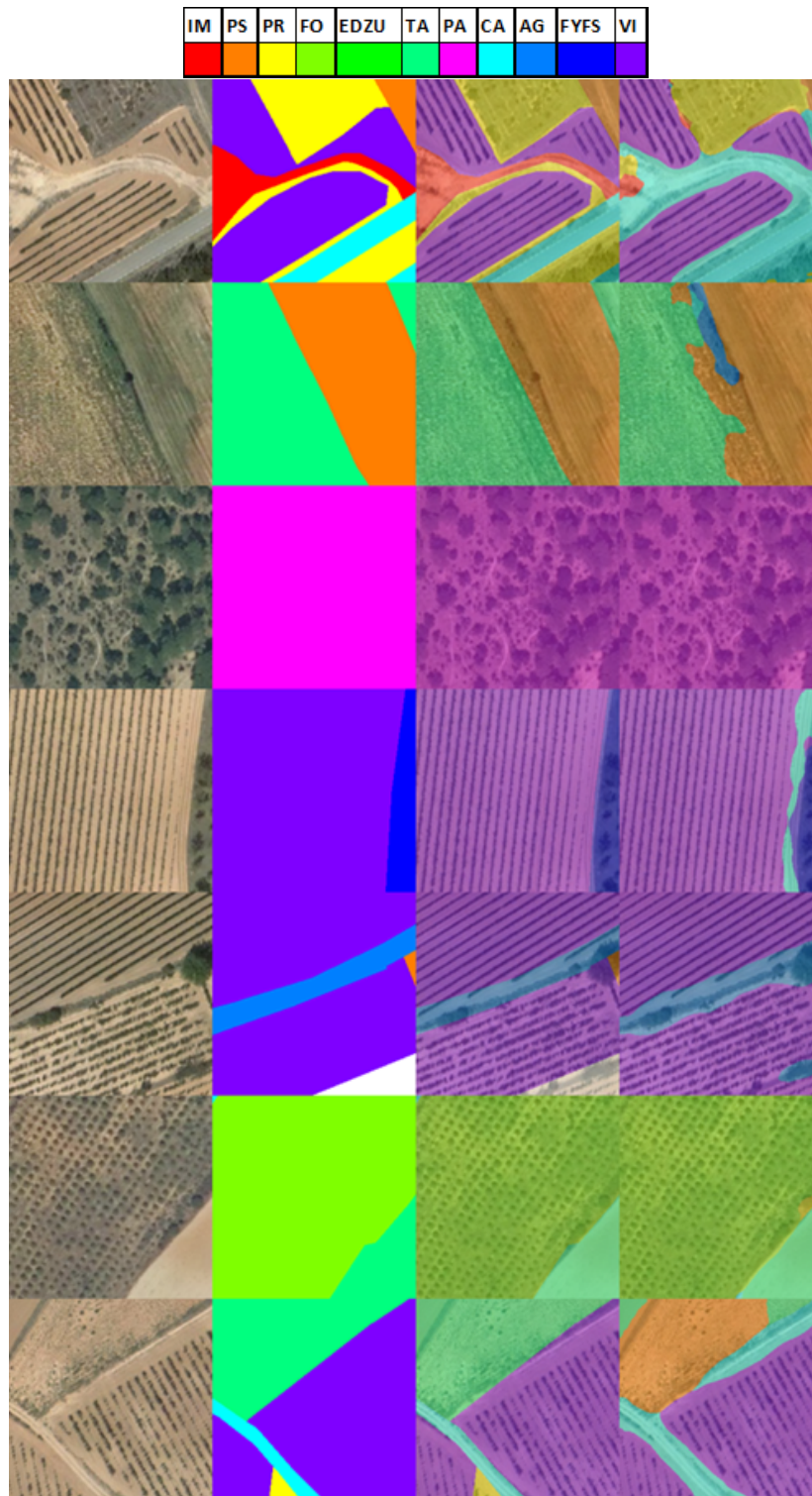


Figura 4.8.- Mejor experimento PNOA en DeepLabV3+ (Matlab E003-01). (1) Imagen Original, (2) GroundTruth, (3) ImagenOriginal+GroundTruth, (4) ImagenOriginal+Predicción.



4.5.2.- Implementación DeepLabV3+ (Tensorflow)

Esta implementación en Tensorflow se obtiene del GitHub oficial de Google. Sigue en desarrollo por lo que sus resultados pueden mejorar de cara al futuro. Debido a que DeepLab es desarrollada por Google, esta implementación ha sido creada por los autores de las publicaciones de DeepLab.

En la tabla 4.8 se muestran los parámetros de entrada del experimento para su comparación y replicación. En la tabla 4.9 se presentan los resultados globales del experimento, es decir, su GlobalAccuracy, MeanRecall, MeanPrecision y MeanIoU, mientras que en la tabla 4.10 se muestran las métricas de Recall, Precision y MeanIoU por cada clase. En la figura 4.9 se muestra el progreso de la función de loss. Finalmente, en las figuras 4.10 y 4.11 se presentan los resultados de la Recall y Precisión respectivamente para cada una de las clases, tanto para el conjunto de entrenamiento como para el de validación. En la figura 4.12 se presentan algunas de las imágenes obtenidas en el test de la red. El resto de experimentos con esta implementación están disponibles en el ANEXO D.



Parámetros de entrada	
Parámetros del dataset	
Conjunto de entrenamiento	Train1
Conjunto de test	Test1
Parámetros de la red	
InputSize	[256 256 3]
Número de Clases	11
Network	Xception41
Downsampling Factor (Output Stride)	16
Padding	Si
Pesos en las clases	Median Frequency Weighting
Parámetros de entrenamiento	
Solucionador de red (solver network)	Adam
Epochs	60
Fine Tune Batch Norm	False
BatchSize	12
Adam LearningRate	0.00005
Gradient Clipping	-
Regularizacion L2	0.0004
Data Augmentation	Si (Escala 0.25-2.0)
Shuffle	Si
Momentum	0.9
Duración del entrenamiento	11:01:30
Recursos del equipo	%CPU 27.83 %GPU 95.22 %BW VRAM 41.32 GPU VRAM 10997MiB

Tabla 4.8.- Parámetros del experimento (GitHub E010-01).

GlobalAccuracy	MeanRecall	MeanPrecision	MeanIoU
0.89837	0.78186	0.75808	0.63774

Tabla 4.9.- Resultados globales del experimento PNOA DeepLabV3+ (GitHub E010-01).



Clase	Recall	Precision	mIOU
IM	0.56430	0.66483	0.43932
PS	0.58874	0.53094	0.38730
PR	0.79528	0.78655	0.65411
FO	0.84443	0.84622	0.73209
EDZU	0.85943	0.86992	0.76156
TA	0.94752	0.97844	0.92815
PA	0.75801	0.89449	0.69579
CA	0.84077	0.60535	0.54309
AG	0.77256	0.47169	0.41418
FSFY	0.66454	0.73485	0.53602
VI	0.96488	0.95562	0.92350

Tabla 4.10.- Métricas de las clases (GitHub E010-01).

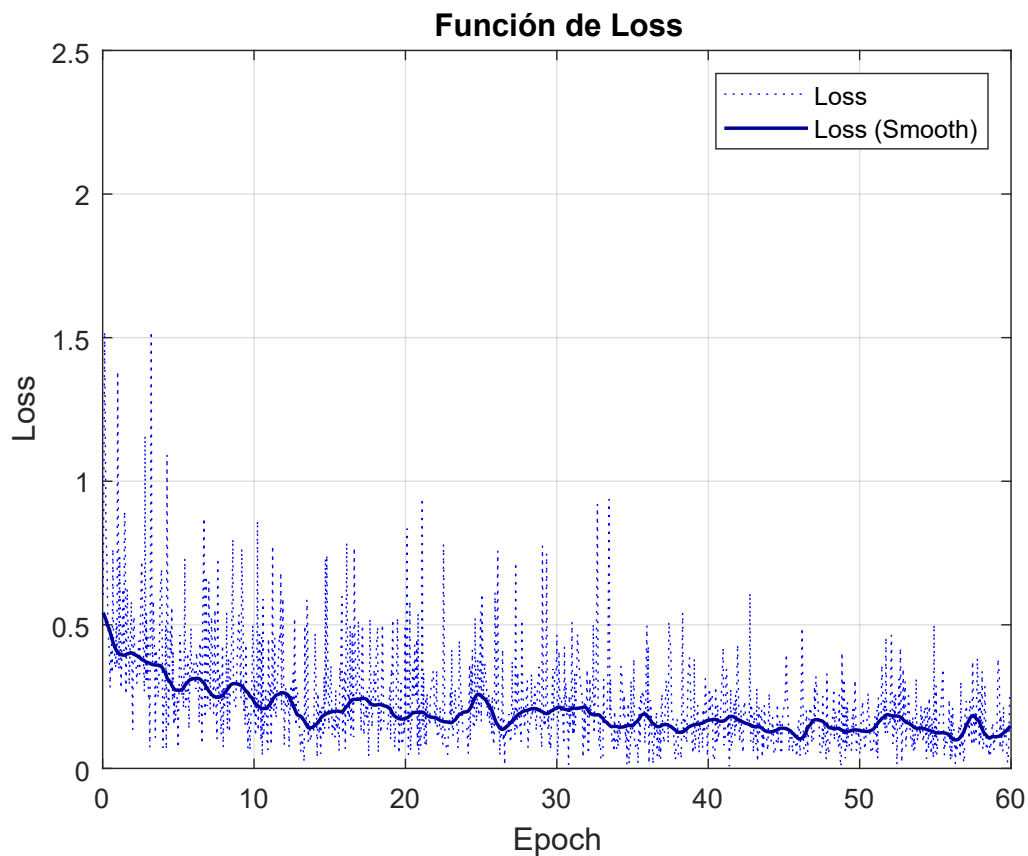


Figura 4.9.- Progreso de Loss (GitHub E010-01).

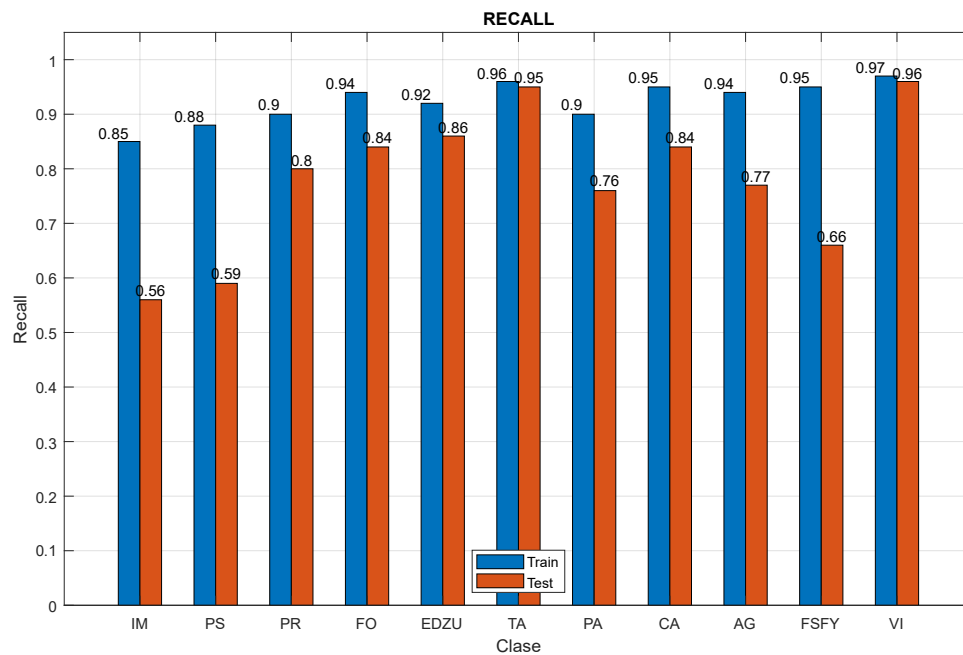


Figura 4.10.- Recall. Train vs Test (GitHub E010-01).

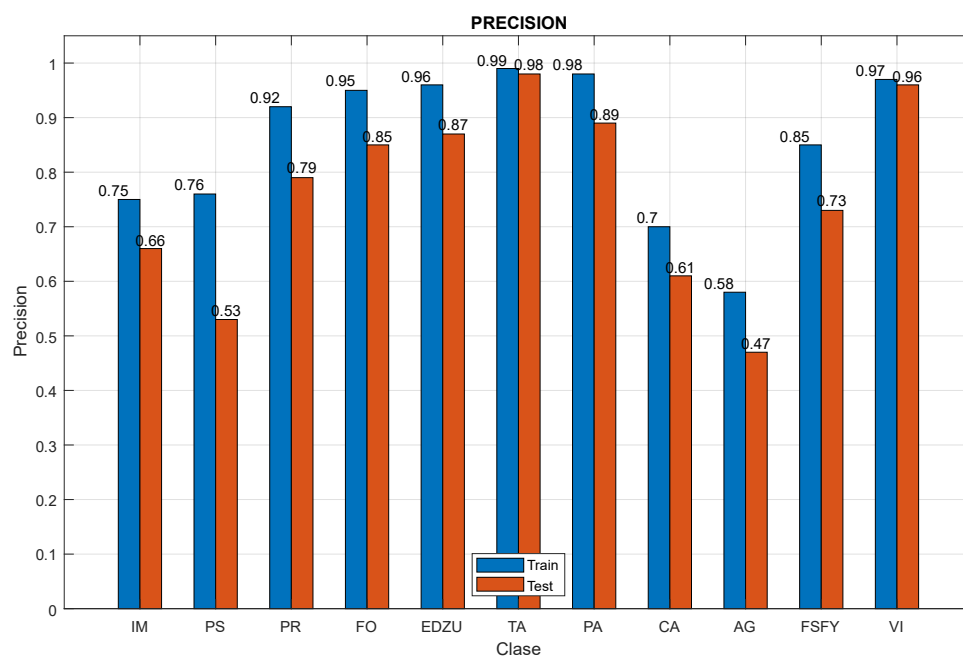


Figura 4.11.- Precision. Train vs Test (GitHub E010-01).

Imágenes de test

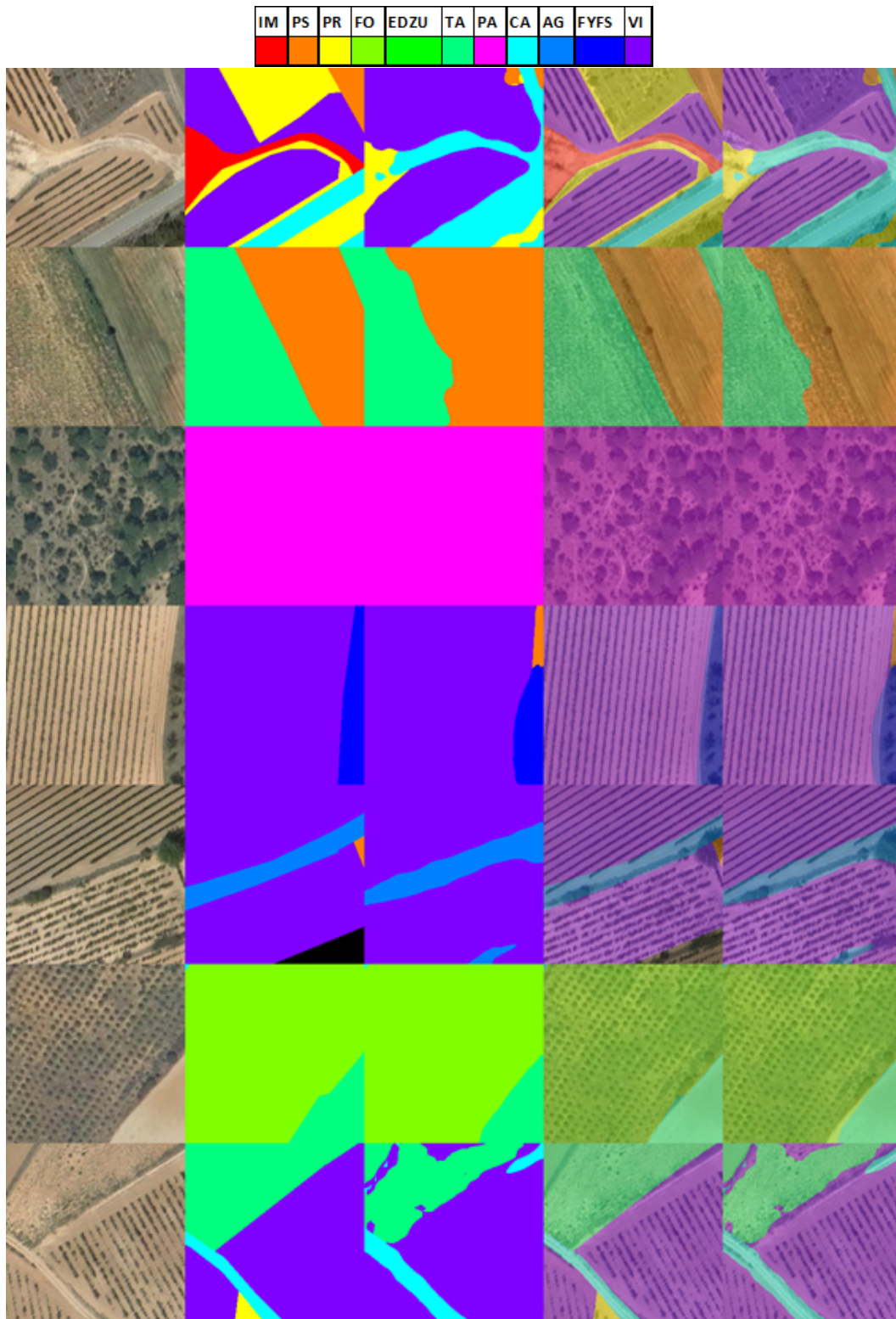


Figura 4.12.- Mejor experimento PNOA en DeepLabV3+ (GitHub E010-01).

(1) Imagen Original, (2) GroundTruth, (3) Predicción, (4)
ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.



4.5.3.- Comparaciones

Tanto la implementación de DeepLabV3+ de Matlab como la de Tensorflow ofrecen resultados muy similares, siendo ligeramente superior la implementación de Tensorflow. Ambas implementaciones tienen problemas con las clases menos numerosas como “AG”. Además, ambas implementaciones sufren el mismo problema al clasificar como “CA (VIALES)” aquellas zonas de la imagen que presentan un patrón alargado similar al de un camino, que a nivel de usuario parecen ser predicciones acertadas en la mayoría de casos aunque su etiquetado no se corresponda.

Finalmente, la implementación en Tensorflow ofrece una mejora temporal importante, reduciendo el tiempo necesario para entrenar el modelo de diecisiete horas a once horas. Una mejora de seis horas o una mejora de un treinta y cinco por ciento.

4.6.- Servicio de inferencia

Para poder apreciar el posible uso de esta tecnología se ha implementado un prototipo de servicio web dentro de un contenedor Docker al que se le envían las imágenes y devuelve un fichero comprimido con las predicciones correspondientes.

En la figura 4.13, se muestra como es la interfaz del servicio. Para desarrollar la interfaz se ha utilizado HTML5 + Bootstrap 4.

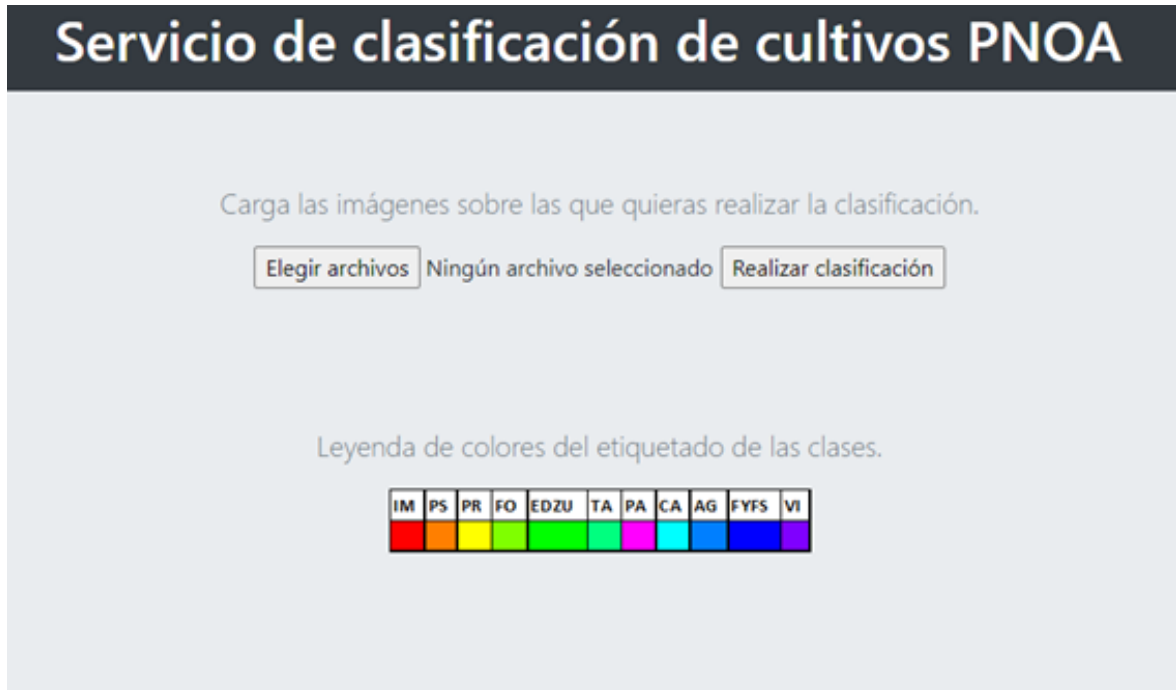


Figura 4.13.- Interfaz del servicio web desarrollado.

Después de que la página se haya cargado con éxito, hay que subir las imágenes sobre las que se desea realizar las predicciones. Para ello se debe seleccionar “Elegir archivos” (el texto del botón puede variar según el navegador) para buscar y seleccionar las imágenes que se deseen utilizar. Finalmente, hay que hacer click en “Realizar clasificación” y esperar a que se ejecute la predicción, la página queda bloqueada en espera de la respuesta, que en este caso, es un fichero comprimido llamado “detections.zip” en el que se encuentran las imágenes con las máscaras de las predicciones realizadas por la red en formato de PNG siguiendo el código de colores RGB mostrado en la página (ver figura 4.13).

Cuando se realiza la petición, se está enviando una petición REST de tipo POST a /inference en la que se adjunta el conjunto de imágenes a predecir, de forma directa y sin utilizar ficheros comprimidos. En la figura 4.14 se muestra el contenido de esta petición mediante la herramienta POSTMAN. Hay que especificar las cabeceras “Key:Content-type” y “Value:multipart/form-data”. En dicha interfaz (ver figura 4.14), si se presiona en “Save Response” se obtendrá el fichero comprimido especificado anteriormente.

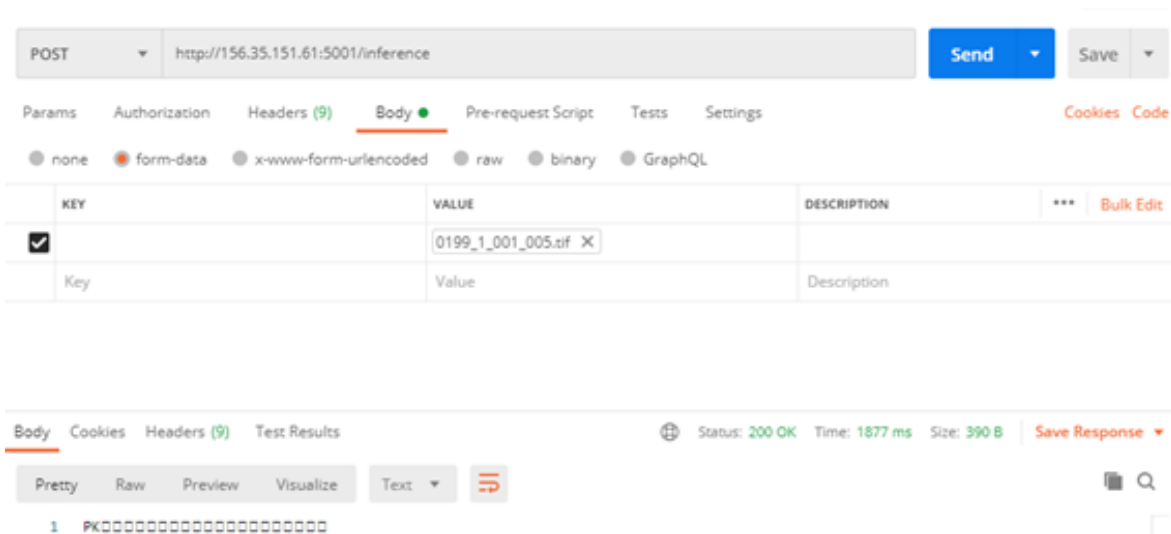


Figura 4.14.- Petición REST del prototipo web PNOA.

Este servicio utiliza Flask para gestionar las peticiones REST, que se encarga de llamar a un script de Python que utiliza Tensorflow2 para cargar el modelo entrenado anteriormente y realizar la predicción. Solamente se permiten imágenes en formato PNG, y las imágenes, aunque se recomienda que tengan un tamaño de 256×256 píxeles, se permiten de cualquier tamaño ya que antes de alimentarlas a la red se reescalan al tamaño objetivo.

Para lanzar este servicio en cualquier maquina simplemente hay que instalar Docker y llamar al método “Docker-compose up” desde la raíz directorio del prototipo que se adjunta como anexo bajo el nombre de “DeepLabGPU-TF-PNOA-docker.zip” para su versión con uso de GPU y “DeepLabCPU-TF-PNOA-docker.zip” para su versión con CPU.

La experimentación con el servicio de inferencia de esta implementación se realiza mediante un cliente y un servidor en máquinas diferentes que se comunican a través de la red. Sin embargo, los tiempos recogidos son aquellos registrados por el servidor, es decir, no se toman los tiempos de transferencia de las imágenes sobre la red, tanto para las imágenes de entrada como para el fichero comprimido de salida.

Este prototipo utiliza la implementación de Tensorflow y en las tablas 4.11 y 4.12 se presentan los tiempos de inferencia para CPU y GPU respectivamente. La **primera**



imagen a predecir de cada petición obtiene un tiempo superior al resto ya que necesita cargar en memoria todas las librerías necesarias (Tensorflow, numpy, etc), las imágenes a predecir, y el modelo entrenado antes de ejecutar la predicción, una vez cargado en memoria ya no necesita volver a hacerlo por lo que el resto de predicciones son mucho más rápidas, hasta el punto en el que su tiempo es despreciable en este contexto. Para el caso de uso con GPU este tiempo de carga es aun mayor debido a que tiene que cargar en memoria las librerías de CUDA y pasar los datos a la memoria de la GPU, por lo que aunque las predicciones se realizan diez veces más rápido, el tiempo de carga tiene un mayor peso a la hora de predecir las imágenes. Por este motivo, solo resulta interesante utilizar la GPU para realizar predicciones si se dispone a predecir un número elevado de imágenes. Esta implementación se puede optimizar para que solo se necesite cargar CUDA, el resto de librerías, y el modelo una vez, cuando se pone en marcha el servicio. En estas pruebas no se tiene en cuenta el tiempo necesario para enviar y recibir las imágenes a través de la red.

“T. de predicción. (s)” es el tiempo de inferencia total, y “T. total(s)” es el tiempo que se tarda desde que comienza la carga del modelo hasta que finalizan las predicciones. La diferencia entre el tiempo total y el tiempo de predicción corresponde a la carga de librerías, imágenes y el modelo en memoria.

# Imágenes	Tiempo de predicción (s)	Tiempo total (s)
1	~0.5	~2.5
10	~0.07, excepto la primera que tarda 0.5	~4

Tabla 4.11.- Tiempos de inferencia del prototipo PNOA con CPU.

# Imágenes	Tiempo de predicción (s)	Tiempo total (s)
1	~1.5	~5
10	~0.008, excepto la primera que tarda 1.5	~5

Tabla 4.12.- Tiempos de inferencia del prototipo PNOA con GPU.

5. Experimentación con el conjunto de datos Sentinel-2

5.1.- Obtención de las imágenes Sentinel-2

En esta sección se detalla todo el proceso seguido para la obtención de las imágenes necesarias para la creación de un conjunto de datos de Sentinel-2.

5.1.1.- Uso de la API de SentinelHub.

La descarga de imágenes multiespectrales de Sentinel-2 se realiza a través de SentinelHub API¹, que permite obtener imágenes de las regiones que se detallan con las bandas que se especifiquen de forma sencilla y para cualquier fecha. Para comenzar a trabajar con esta API hay que registrarse en la página, que dispone de un mes de uso gratuito.

Entre algunas de las opciones que se pueden configurar desde la página anterior se pueden seleccionar las bandas, tanto su calidad, formato, porcentaje de recubrimiento de nubes, si se prefiere la última imagen o la que menos nubes tenga, si se quiere obtener un conjunto de imágenes de diferentes fechas, de que satélite se quiere obtener la imagen, etc. En la figura 5.1 se muestra la interfaz web que permite realizar estas configuraciones. Cuando se acaban de realizar las configuraciones necesarias se puede copiar el ID que se indica en “Service end-points” para utilizarlo en el script a la hora de realizar las peticiones (ver código 5.1).

¹Enlace a SentinelHub API: apps.sentinel-hub.com

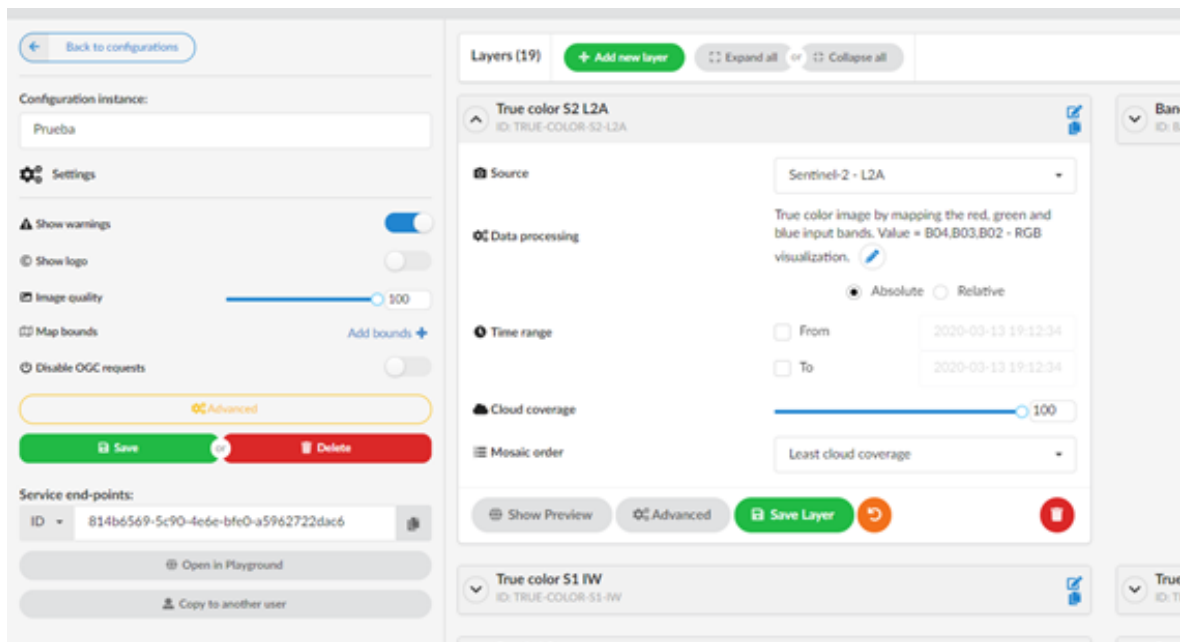


Figura 5.1.- Configuraciones de SentinelHub.

Código 5.1.- Conexión con SentinelHub API mediante el ID.

```
# Account instance ID
INSTANCE_ID = '814b6569-5c90-4e6e-bfe0-a5962722dac6'

# Load account config
if INSTANCE_ID:
    config = SHConfig()
    config.instance_id = INSTANCE_ID
else:
    config = None
```

Para seleccionar la región a descargar simplemente se tienen que especificar dos puntos que conformarían el vértice superior-izquierdo de la parcela y el inferior-derecho. Cada vértice está compuesto de dos valores numéricos que representan la longitud y latitud respectivamente (ver código 5.2).

Código 5.2.- Creación de la región a descargar.

```
# Location of the parcel
coords_wgs84 = [46.16, -16.15, 46.30, -16.00]
bbox = BBox(bbox=coords_wgs84, crs=CRS.WGS84)
```



Para la descarga de imágenes se puede utilizar el método `WcsRequest` como se muestra en el código 5.3. Este método permite una personalización de sus parámetros para obtener una imagen más concreta, por ejemplo, permite seleccionar el tamaño en píxeles de la imagen a descargar, escalando la misma si fuese necesario (si solo se especifica ancho o alto la imagen mantendrá su relación de aspecto). También se permite seleccionar que capa se va a descargar, de las configuradas anteriormente, simplemente especificando su string en el parámetro “layer”. Se debe siempre rellenar el campo de “bbox” con la región de la parcela y “config” con la configuración obtenida gracias al ID de la configuración. En “time” se puede seleccionar en qué fecha, o entre que fechas, se quiere la imagen, y si se selecciona ‘latest’ se descargará la más reciente posible. También existe la posibilidad de utilizar el campo “time_difference” para combinar varias imágenes dentro de un periodo seleccionado para evitar que la imagen quede incompleta por falta de información. Y finalmente, el parámetro “data_folder” para especificar en qué carpeta descargar la imagen. Si se desea obtener más de tres bandas se necesita especificar el tipo de fichero `TIFF_d32f` para almacenarlas todas correctamente.

Código 5.3.- Método para la petición de descarga.

```
# Request the image
wcs_request = WcsRequest(
    data_folder='C:/Users/solid/Desktop/Trabajo/imagenes',
    layer='TRUE-COLOR-S2-L1C',
    bbox=bbox,
    time='latest',
    time_difference=datetime.timedelta(hours=2),
    resx='10m',
    resy='10m',
    image_format=MimeType.TIFF_d32f,
    config=config
)
```

Para guardar la imagen una vez recibida la respuesta, simplemente hay que pedir que se almacene mediante el comando indicado en el código 5.4.

Código 5.4.- Almacenamiento local de las imágenes.



Download the image

```
wms_true_color_request.save_data()
```

Las configuraciones aplicadas en la petición sobrescriben las configuraciones de la página de SentinelHub, de forma que **si existe una contradicción entre ambas siempre se recurrirá a los parámetros de la petición enviada.**

5.1.2.- Obtención de hojas MTN50 completas.

Para simplificar la tarea de la generación de un conjunto de datos de imágenes de Sentinel-2 se ha decidido utilizar las zonas geográficas determinadas por las hojas MTN50. Esto implica que será más fácil obtener información de parcelas en la misma zona, reduciendo los píxeles que quedan sin clasificar en las imágenes. Además, de esta forma podemos descargar imágenes de gran tamaño para luego crear los recortes, reduciendo así el consumo del procesamiento gratuito que provee SentinelHub API, entre otros beneficios como una mayor facilidad para eliminar imágenes con nubes.

Las coordenadas de las regiones de las hojas MTN50 se pueden obtener en el Centro de Descargas del Gobierno de España², accediendo a la categoría de “Información geográfica de referencia y descargando la información correspondiente a “Cuadrículas cartográficas MTN25 y MTN50” (ver figura 5.2) obteniendo un fichero comprimido que contiene el fichero “MTN50_ETRS89_Peninsula_Baleares_Canarias.shp” de donde leer dichas coordenadas

²Enlace al centro de descargas: <https://centrodedescargas.cnig.es/CentroDescargas>



Cuadrículas cartográficas MTN25 y MTN50

Descripción: cuadrículas cartográficas oficiales del MTN50 y MTN25.

SGR: ETRS89 o ED50 (según edición) en la Península, Islas Baleares, Ceuta y Melilla, y REGCAN95 en las Islas Canarias. Coordenadas geográficas longitud y latitud.

Ud. descarga: toda España

Formato: shape (.shp)

[Metadatos](#)

- Cuadrícula MTN25 2015
- Cuadrícula MTN50 2013



Descargar

Figura 5.2.- Metadatos de cuadrículas cartográficas MTN25 y MTN50.

Para ello se obtienen dos imágenes de SentinelHub API por cada hoja MTN50, ya que una imagen del área de una hoja MTN50 supera el tamaño máximo permitido por SentinelHub API. En las figuras 5.3 y 5.4, se muestra un ejemplo de las imágenes obtenidas que cubren el área de la hoja MTN50 0162. Estas dos imágenes constan de 1,447x1,899 píxeles y 1,451x1,902 píxeles respectivamente.

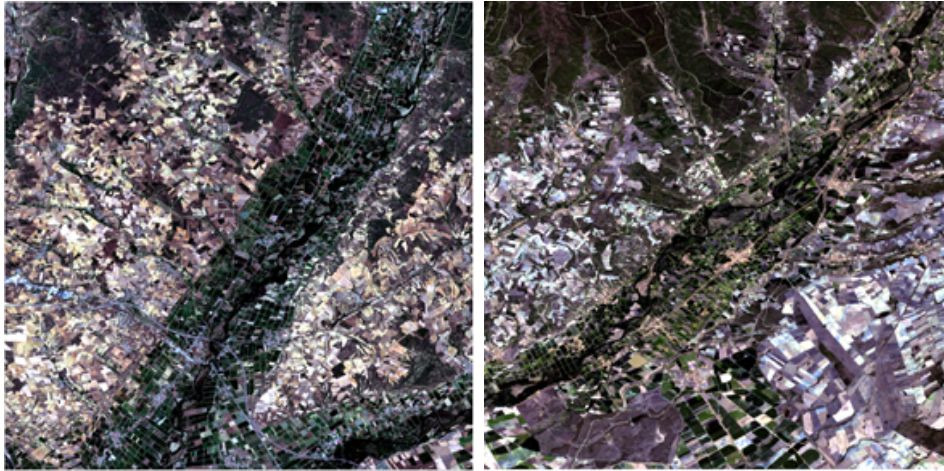


Figura 5.3.- Imagen
izquierda de la hoja
MTN50 número 0162.

Figura 5.4.- Imagen
derecha de la hoja MTN50
número 0162.

5.1.3.- Eliminación de nubes.

Las nubes y sus sombras son un gran problema en las imágenes de satélites, por ello existen herramientas para detectar o hasta corregir los defectos causados por las mismas. Sin embargo, este enfoque resulta complejo y puede llegar a alterar las imágenes originales por lo que finalmente se ha optado por seleccionar de forma manual aquellas imágenes que pertenecen a las regiones de interés descargando aquellas imágenes que no presenten nubes. Para ello, se itera por diferentes fechas hasta encontrar un día despejado. Esta tarea es posible gracias a que se obtienen imágenes de gran tamaño, por lo que el número de imágenes a descargar no es elevado. Finalmente, estas imágenes son recortadas en tamaños asumibles para una red de segmentación semántica.

5.2.- Obtención de los recortes de entrada.

El conjunto de datos de las imágenes se descarga mediante un script en Python llamado “DownloadSentinelPNOA.py”, que se encarga de realizar las peticiones de descarga a la API de SentinelHub, para almacenar las imágenes en la localización deseada. Además, se guarda la posición de las coordenadas del punto superior izquierda



en un fichero de texto llamado “topLeftPoint.txt”, en el que también se almacena la fecha de obtención de la imagen por el satélite Sentinel-2. Se obtiene una imagen en formato .tif multi-page donde se almacenan todas las bandas a la vez. Una vez descargadas todas las imágenes necesarias, se deben tratar, observando si son utilizables o no (no hay nubes, está descargada la imagen completa, etc. Y volver a descargar la misma región pero de otra fecha hasta que se considere “utilizable”), y recortar en imágenes más pequeñas (de 256×256 píxeles) las imágenes obtenidas para asegurar que todas tienen el mismo tamaño y que este, es apropiado para su uso en una red neuronal. Para realizar estos recortes se utiliza la función “gdal_retile” explicada en el apartado 4.2. En la figura 5.5 se puede observar un ejemplo de uno de estos recortes.

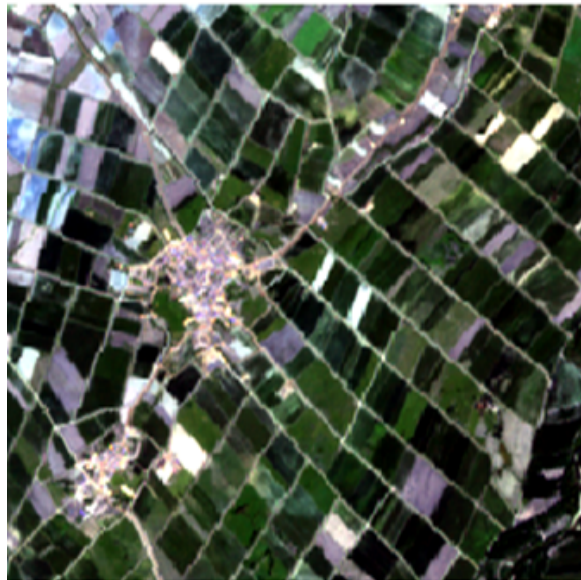


Figura 5.5.- Ejemplo de recorte de Sentinel-2 (256×256 píxeles).

5.3.- Generación del GroundTruth para los recortes

La obtención de las máscaras GroundTruth es similar a la metodología seguida en la sección 4.3.



5.4.- Redes neuronales convolucionales utilizables

En este caso, solamente se pueden utilizar aquellas redes neuronales convolucionales que no hagan uso de redes backbone preentrenadas, ya que, generalmente están diseñadas para trabajar con imágenes de tres canales, es decir, imágenes RGB. Debido a esta limitación **se decide utilizar la red UNet**, puramente convolucional y sin redes backbone, que permite utilizar tantos canales en las imágenes como se deseen, y debido a que las imágenes obtenidas del satélite Sentinel-2 tienen trece bandas, si se quiere aprovechar todo el potencial de esta información hay que utilizar una red que lo permita. UNet, como se ha mencionado anteriormente, permite un alto grado de flexibilidad en su arquitectura, es por esto que existen multitud de versiones y modificaciones de la misma, especializadas para todo tipo de tareas. Por estos motivos se decide utilizar la red UNet.

5.5.- Experimentación con once clases

Inicialmente, se realizan las experimentaciones con las once clases con las que ya se realizaron los experimentos con el conjunto de datos PNOA en el apartado 4.5.

5.5.1.- Implementación de UNet en Matlab

Esta implementación de UNet es la ofrecida por Matlab, se varían sus parámetros de entrada (ver tabla 5.1) hasta encontrar un modelo óptimo para la predicción de las once clases seleccionadas. En la tabla 5.2 se presentan los resultados globales del experimento, es decir, su GlobalAccuracy, MeanRecall, MeanPrecision y MeanIoU, mientras que en la tabla 5.3 se muestran las métricas de Recall, Precision, IoU y MeanBFScore por cada clase. En la tabla 5.4 se encuentra la matriz de confusión del experimento. Finalmente, en la figura 5.6 se muestra el progreso de la función de loss y de la GlobalAccuracy durante las epochs del entrenamiento, y en la figura 5.7 se presentan algunas de las imágenes obtenidas en el test de la red. El resto de experimentaciones con esta implementación para el conjunto de datos Sentinel-2 se encuentran en el ANEXO A.



Parámetros de entrada	
Parámetros del dataset	
Conjunto de entrenamiento	Train1Aumentado (10 bandas)
Conjunto de test	Test1Aumentado (10 bandas)
Parámetros de la red	
InputSize	[256 256 10]
Número de Clases	11
Profundidad de la red	4
Numero de filtros (Encoder)	32
Padding	Si
Pesos en las clases	Median Frequency Weighting
Parámetros de entrenamiento	
Solucionador de red (solver network)	Adam
Epochs	125
BatchSize	32
LearningRate -Inicial	0.0005
LearnRate-Final	0.0005
Gradient Clipping	1
Regularizacion L2	0.0001
Data Augmentation	Si (Eje X, Eje Y)
Shuffle	Si
Momentum	-
Duración del entrenamiento	02:41:19

Tabla 5.1.- Parámetros del experimento E010-03.

GlobalAccuracy	MeanRecall	MeanPrecision	MeanIoU
0.52786	0.52128	0.36442	0.25955

Tabla 5.2.- Resultados globales del experimento Sentinel-2 UNet (Matlab E010-03).



Clase	Recall	Precision	Iou	MeanBFScore
DESC	0.09524	0.38857	0.08283	0.07133
IM	0.51263	0.22873	0.18788	0.17357
PS	0.46359	0.21821	0.17422	0.27453
PR	0.37618	0.54389	0.28597	0.35613
FO	0.52822	0.48939	0.34054	0.31285
EDZU	0.74773	0.39670	0.34987	0.32507
TA	0.69285	0.85855	0.62186	0.60492
PA	0.58032	0.28237	0.23448	0.20275
CA	0.40201	0.13350	0.11138	0.38472
AG	0.63312	0.17658	0.16019	0.25667
FSFY	0.45366	0.08726	0.07896	0.08911
VI	0.76981	0.56926	0.48645	0.36344

Tabla 5.3.- Métricas de las clase (Matlab E010-03).



# px reales		Matriz de confusión															
		Clases reales															
DESCONOCIDO	# px predichos	DESCONOCIDO	IM	PS	PR	FO	EDZU	TA	FA	PA	CA	AG	FSY	VI	PRECISION		
DESCONOCIDO	4,679,477	390,815	1,349,052	4,097,243	2,798,867	303,735	14,805,904	1,180,123	909,026	409,546	158,407	1,030,091	0.38857187				
IM	1146977	445,683	16,348	13,585	276,851	147,237	62	177,036	52,411	2,139	367	842	0.22873355				
PS	875875	211,467	200,342	40,283	142,945	36,498	26,314	161,756	8,654	32,744	3,037	5,014	0.2182074				
PR	2866099	281,562	27,582	625,404	502,441	149,014	14,493	1,037,178	42,973	23,300	20,762	58,475	0.54388552				
FO	2833896	478,419	22,741	114,875	1,541,315	277,418	2,440	191,087	97,570	70,230	7,186	17,393	0.48939493				
EDZU	3020914	765,922	4,855	52,216	325,776	1,478,420	7,231	92,376	226,749	13,312	5,814	6,965	0.39669872				
TA	572505	143,202	30,305	55,054	29,510	7,754	227,112	1,646	22,387	4,425	6,759	2,278	0.85855393				
FA	11948229	1,282,094	15,136	98,556	90,069	22,131	907	10,258,199	4,190	105,743	28,488	12,837	0.28236567				
PA	2425394	561,691	6,751	38,852	572,345	425,642	4,861	68,354	684,848	44,491	7,477	3,881	0.1335004				
CA	2737385	224,513	36,954	91,878	326,874	60,689	9,067	1,519,280	22,049	365,442	42,983	7,218	0.17658043				
AG	1468413	120,493	17,322	49,277	136,610	115,257	2,564	655,830	24,310	80,113	259,293	4,629	0.08725601				
FSY	823588	68,617	8,600	139,644	117,210	68,180	8,183	227,590	11,978	19,192	5,619	71,863	0.56925538				
VI	1393011	95,814	3,879	29,428	35,297	10,627	501	375,145	2,795	30,025	2,467	14,054	0.7698				
RECALL		0.0952	0.5126	0.4636	0.3762	0.5282	0.7477	0.6928	0.5803	0.4020	0.6331	0.4537	0.52786339				

Tabla 5.4.- Matriz de confusión (Matlab E010-03).

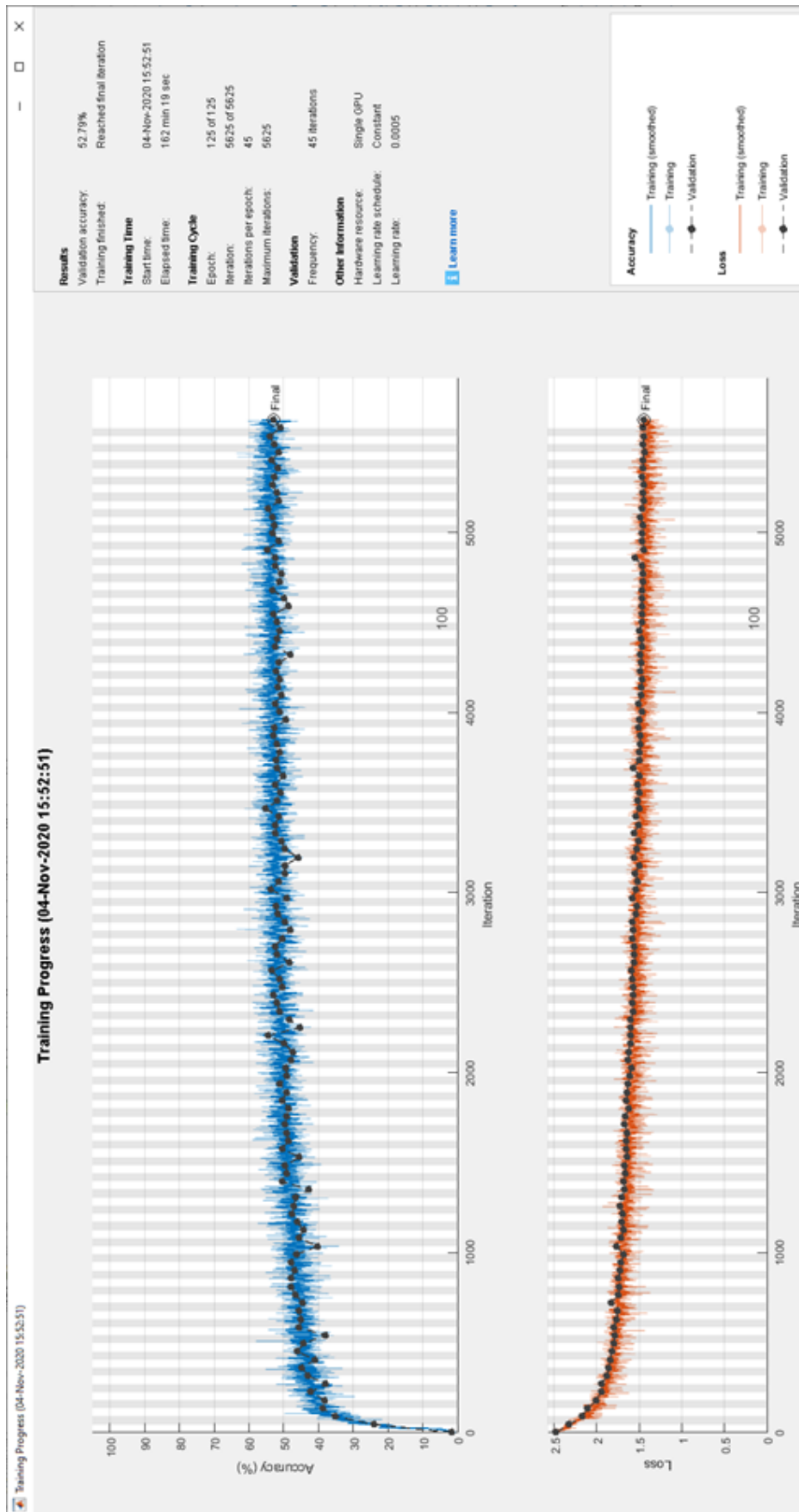


Figura 5.6.- Progreso de GlobalAccuracy y Loss (Matlab E010-03).

Imágenes de test

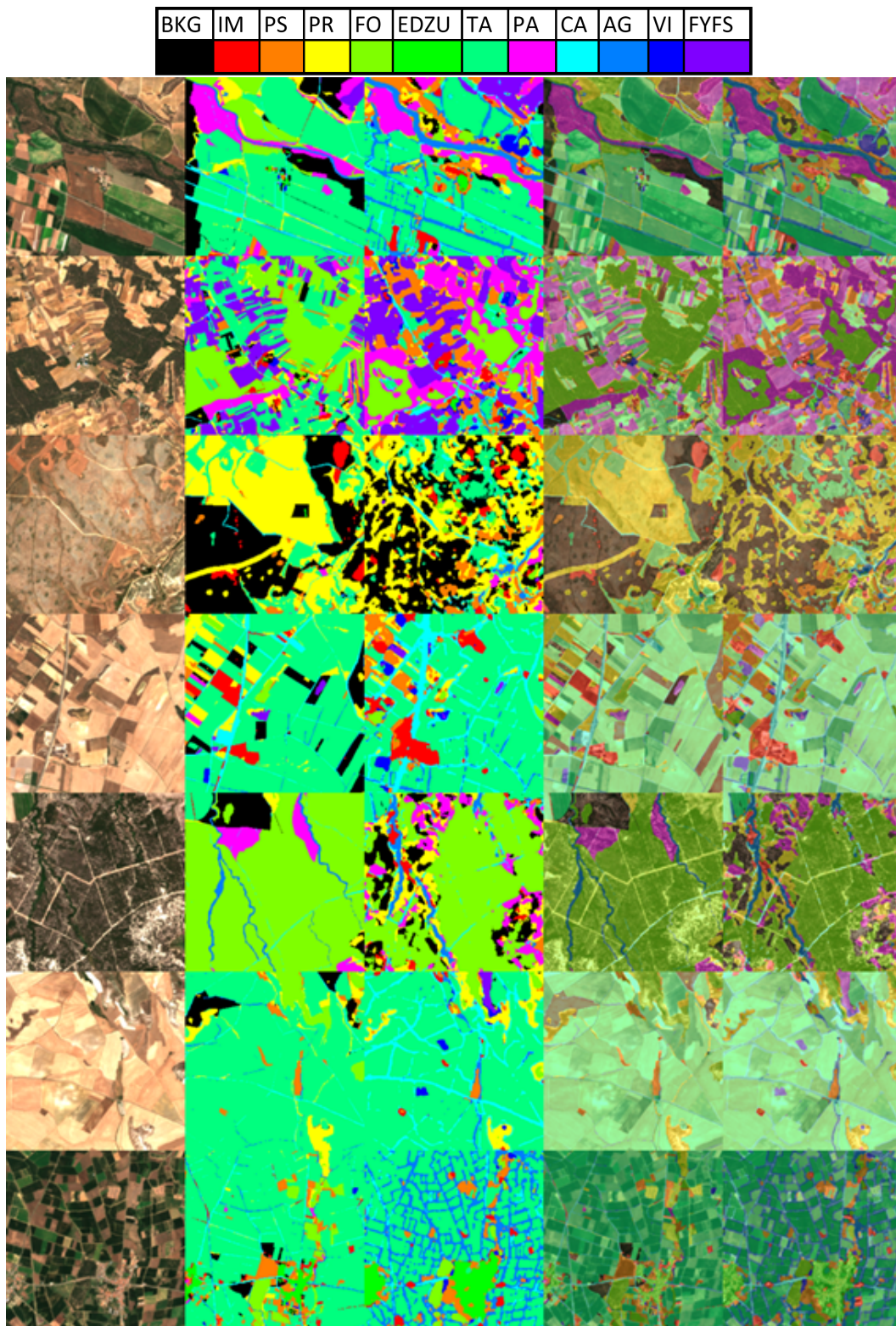


Figura 5.7.- Imágenes de test (Matlab E010-03). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.



5.6.- Experimentación con tres clases

Debido a los resultados obtenidos en la experimentación con once clases se ha realizado una segunda experimentación con solamente tres clases con el objetivo de simplificar las predicciones y mejorar sus resultados. Se considera que a esta resolución espacial de 10 m/píxel hay ciertas clases que son indistinguibles para un humano por lo que se ha decidido juntar algunas de las clases y desechar aquellas que tienen muy pocos píxeles para el entrenamiento.

Las nuevas clases son las siguientes.

- **(TAVI). TIERRAS ARABLES + VIÑEDOS:** Se unifican ambas clases en una.
- **(PSRPA). PASTOS:** Se unifican las tres clases de pastos en una sola. (PASTIZAL, PASTO ARBUSTIVO y PASTO CON ARBOLADO)
- **(EDZUCA). EDIFICACIONES, ZONA URBANA y VIALES:** Se unifican estas tres clases por estar relacionadas con la infraestructura.
- **(OTRAS). OTRAS:** El resto de las clases se unifican bajo una sola clase. No se utiliza en todos los experimentos.
- **(DESC). DESCONOCIDO:** Aquellos píxeles que no tengan información sobre a que clase pertenecen se agrupan en esta clase. No se utiliza en todos los experimentos.
- **(BKG). BACKGROUND:** Es la suma de las clases OTRAS y DESCONOCIDO, no se utiliza en todos los experimentos. Si existe OTRAS o DESCONOCIDO, no se podrá utilizar esta clase para no repetir píxeles.

5.6.1.- Implementación de UNet en Matlab

Esta implementación de UNet es la ofrecida por Matlab, se varían sus parámetros de entrada (ver tabla 5.5) hasta encontrar un modelo óptimo para la predicción de las cuatro clases seleccionadas. En la tabla 5.6 se presentan los resultados globales del experimento, es decir, su GlobalAccuracy, MeanRecall, MeanPrecision y MeanIoU. En la tabla 5.4 se encuentra la matriz de confusión del experimento. Finalmente, en la



figura 5.8 se muestra el progreso de la función de loss y de la GlobalAccuracy durante las epochs del entrenamiento, y en la figura 5.9 se presentan algunas de las imágenes obtenidas en el test de la red. El resto de experimentaciones con esta implementación para el conjunto de datos Sentinel-2 se encuentran en el ANEXO A.

Parámetros de entrada	
Parámetros del dataset	
Conjunto de entrenamiento	Train1Aumentado (10 bandas, 3 clases)
Conjunto de test	Test1Aumentado (10 bandas, 3 clases)
Parámetros de la red	
InputSize	[256 256 10]
Número de Clases	4 (3+BKG)
Profundidad de la red	4
Numero de filtros (Encoder)	32
Padding	Sí
Pesos en las clases	Median Frequency Weighting
Parámetros de entrenamiento	
Solucionador de red (solver network)	Adam
Epochs	125
BatchSize	32
LearningRate -Inicial	0.0005
LearnRate-Final	0.0005
Gradient Clipping	1
Regularizacion L2	0.0001
Data Augmentation	No
Shuffle	Si
Momentum	-
Duración del entrenamiento	02:07:41

Tabla 5.5.- Parámetros del experimento E011-10.

GlobalAccuracy	MeanRecall	MeanPrecision	MeanIoU
0.68523	0.64593	0.57935	0.43374

Tabla 5.6.- Resultados globales del experimento Sentinel-2 UNet (Matlab E011-10).



Matriz de confusión						
	# px predichos	PSPRPA	EDZUCA	TAVI	BACKGROUND	PRECISION
# px reales		6,887,820	1,158,558	15,525,451	8,540,811	
PSPRPA	7,931,712	4,479,889	182,125	954,412	2,315,286	<i>0.56481</i>
EDZUCA	3,263,102	648,267	752,077	1,163,114	699,644	<i>0.23048</i>
TAVI	15,141,556	432,644	150,600	12,902,427	1,655,885	<i>0.85212</i>
BACKGROUND	5,776,270	1,327,020	73,756	505,498	3,869,996	<i>0.66998</i>
RECALL		<i>0.6504</i>	<i>0.6491</i>	<i>0.8311</i>	<i>0.4531</i>	0.68523

Tabla 5.7.- Matriz de confusión (Matlab E011-10).

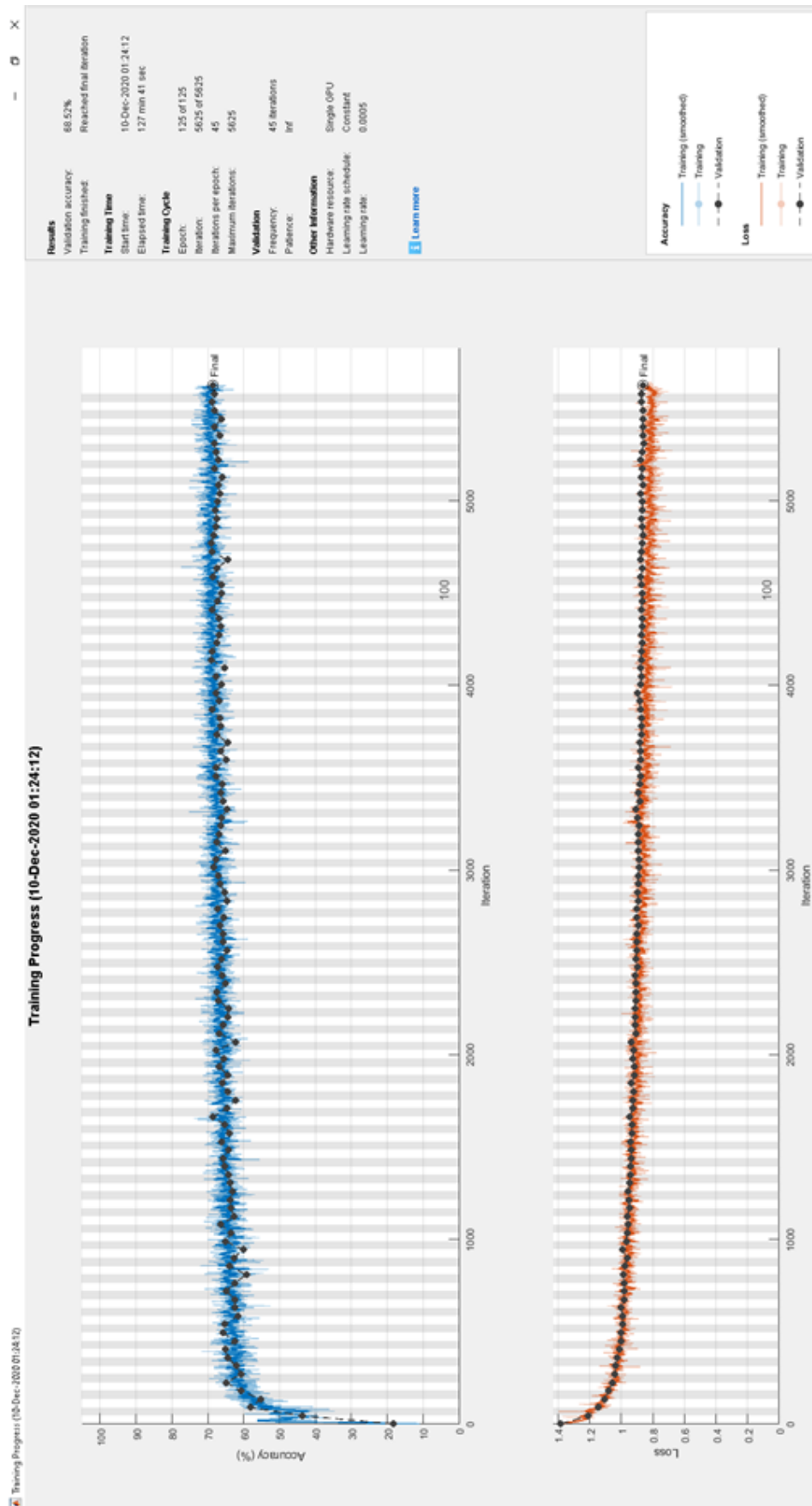


Figura 5.8.- Progreso de GlobalAccuracy y Loss (Matlab E011-10).

Imágenes de test

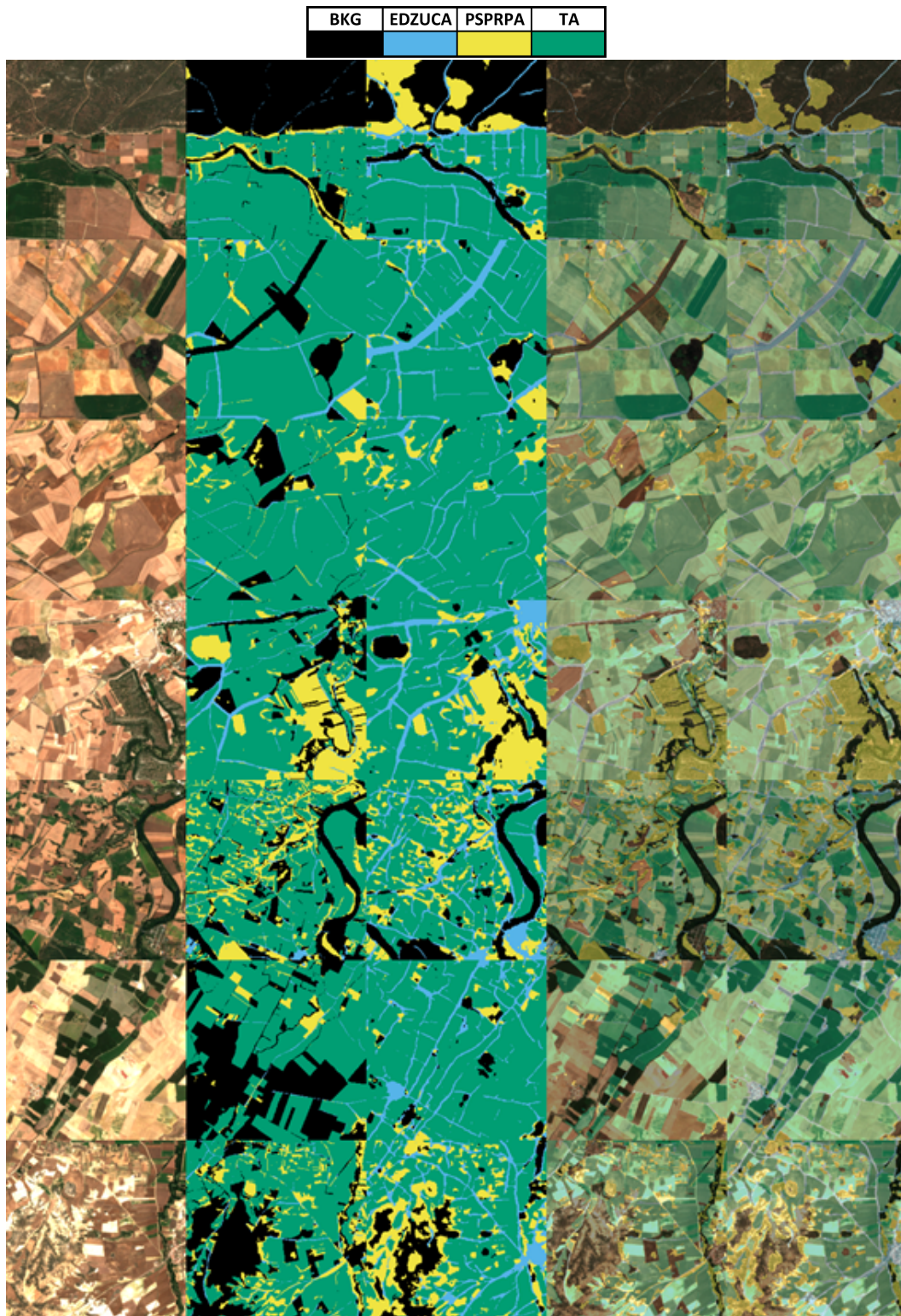


Figura 5.9.- Imágenes de test (Matlab E011-10). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.



5.6.2.- Implementación de UNet en Keras

Esta implementación se ha construido de forma manual siguiendo el modelo de Matlab para que ambas implementaciones sean exactamente iguales capa por capa. La única diferencia visible entre ambas implementaciones (sin tener en cuenta las funciones internas que utiliza para el cálculo de funcionalidades como el solucionador de Adam, etc.) es el parámetro de la regularización L2, que en la implementación de Matlab se produce una mejora en los resultados al activarla mientras que en Keras se mejora al no utilizarla.

En esta experimentación se varían los parámetros de entrada (ver tabla 5.8) hasta encontrar un modelo óptimo para la predicción de las cuatro clases seleccionadas. En la tabla 5.9 se presentan los resultados globales del experimento, es decir, su GlobalAccuracy, MeanRecall, MeanPrecision y MeanIoU. En la tabla 5.10 se encuentra la matriz de confusión del experimento. Finalmente, en la figura 5.10 se muestra el progreso de la GlobalAccuracy y en la figura 5.11 el de la función de loss durante las epochs del entrenamiento. En la figura 5.12 se presentan algunas de las imágenes obtenidas en el test de la red. En el ANEXO B se encuentran el resto de experimentaciones con esta implementación para el conjunto de datos Sentinel-2.



Parámetros de entrada	
Parámetros del dataset	
Conjunto de entrenamiento	Train1Aumentado (10 bandas, 3 clases)
Conjunto de test	Test1Aumentado (10 bandas, 3 clases)
Parámetros de la red	
InputSize	[256 256 10]
Número de Clases	4 (3+BKG)
Profundidad de la red	4
Dropout	Si
BatchNormalization	No
Upsampling	Convolución Transpuesta (2,2)
Numero de filtros (Encoder)	32
Padding	Si
Pesos en las clases	Median Frequency Weighting
Parámetros de entrenamiento	
Solucionador de red (solver network)	Adam
Epochs	125 (Validacion con parada de 30 ep)
BatchSize	32
LearningRate -Inicial	0.0005
LearnRate-Final	0.0005
Gradient Clipping	-
Regularizacion L2	-
Data Augmentation	No
Shuffle	Si
Momentum	-
Duración del entrenamiento	00:26:55

Tabla 5.8.- Parámetros del experimento (Keras K-001).

GlobalAccuracy	MeanRecall	MeanPrecision	MeanIoU
0.66330	0.63101	0.57143	-

Tabla 5.9.- Resultados globales del experimento Sentinel-2 UNet (Keras K-001).

Matriz de confusión						
	# px predichos	PSPAPR	EDZUCA	TAVI	BKG	PRECISION
# px reales ->		6,887,820	1,158,558	15,525,451	8,540,811	
PSPAPR	8,564,301	4,625,829	189,698	1,082,120	2,666,654	0.54
EDZUCA	4,234,529	855,677	759,030	1,751,000	868,822	0.18
TAVI	14,739,462	392,496	147,787	12,506,892	1,692,287	0.85
BKG	4,574,348	1,013,818	62,043	185,439	3,313,048	0.72
RECALL		0.67	0.66	0.81	0.39	0.66

Tabla 5.10.- Matriz de confusión (Keras K-001).

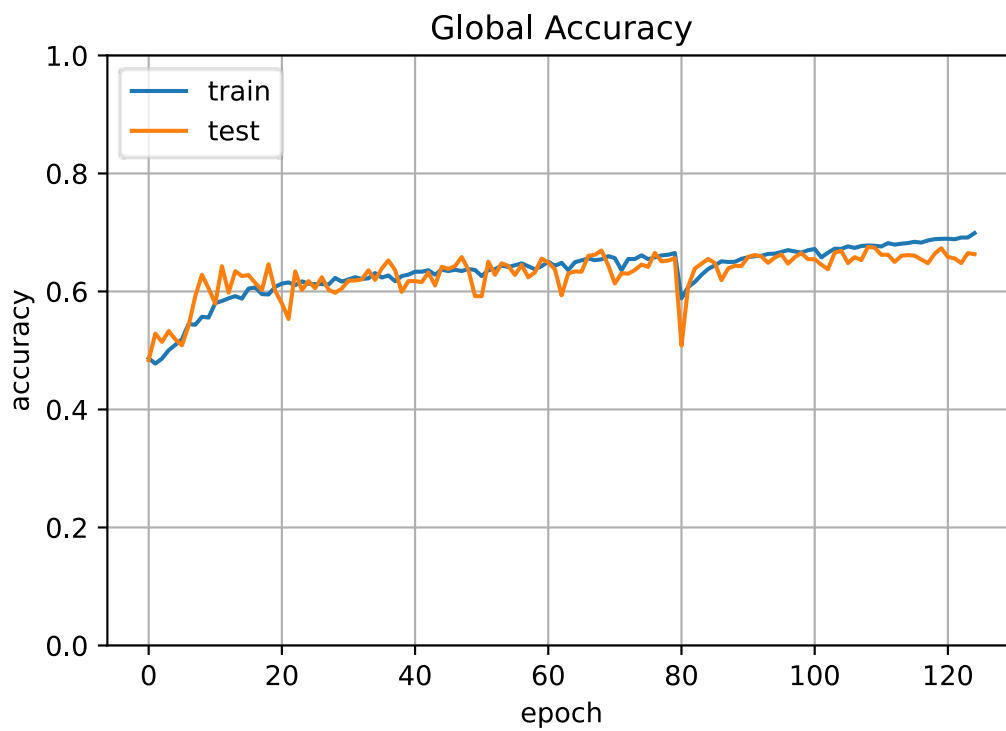


Figura 5.10.- Progreso de GlobalAccuracy (Keras K-001).

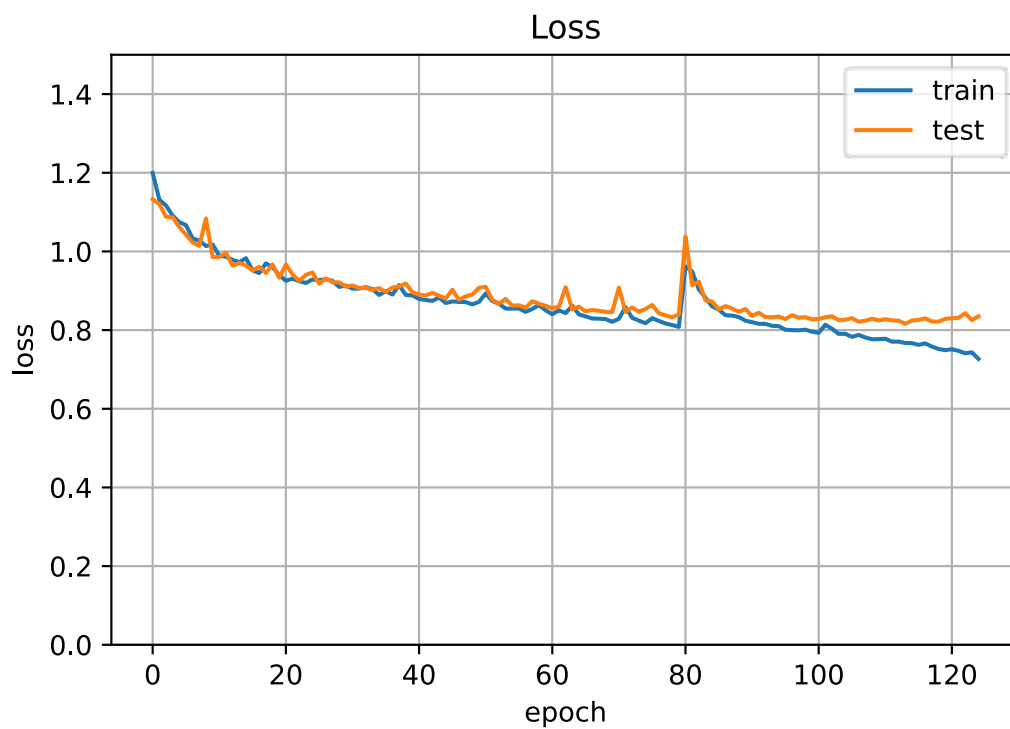


Figura 5.11.- Progreso de Loss (Keras K-001).

Imágenes de test

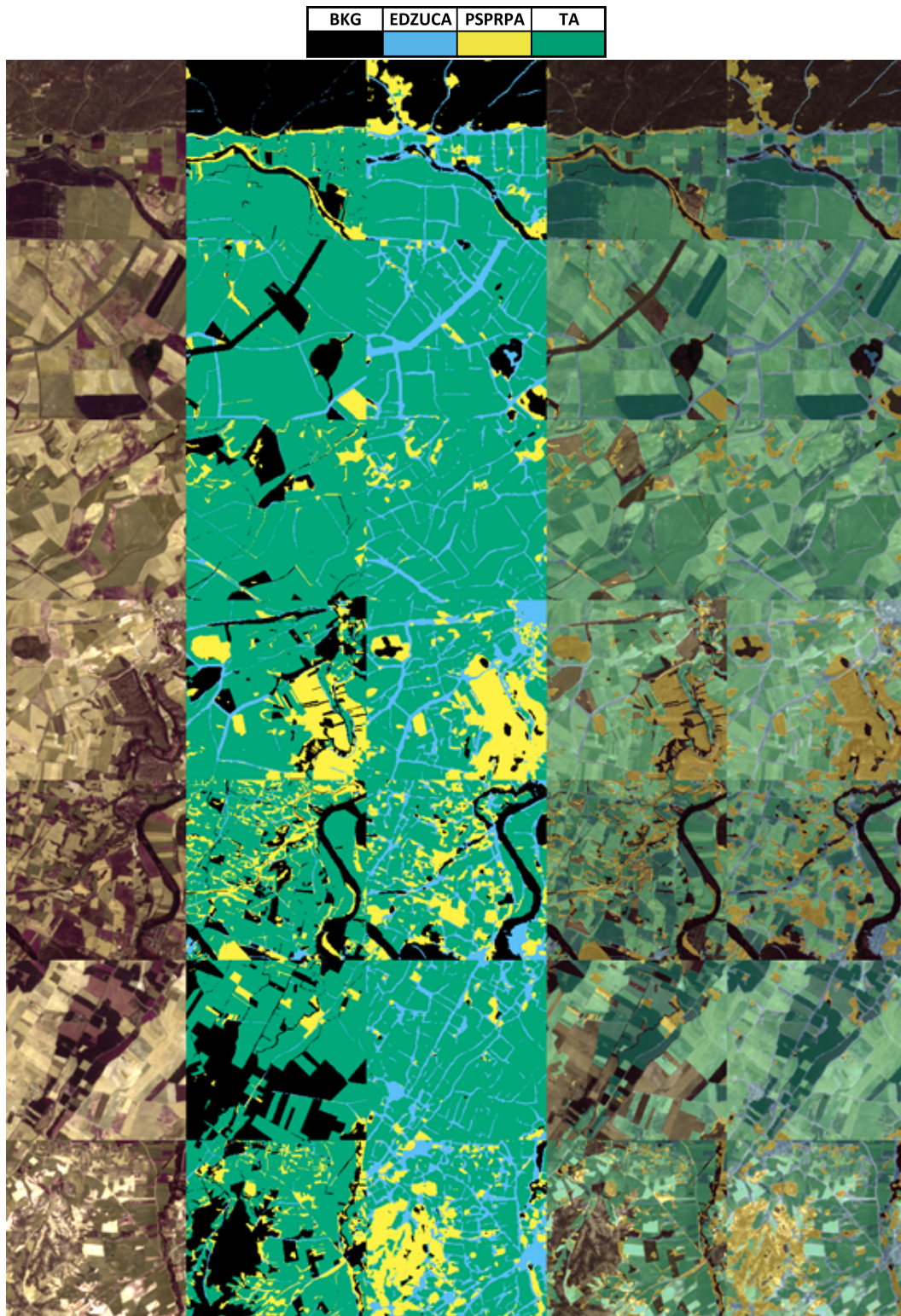


Figura 5.12.- Imágenes de test (Keras K-001). (1) Imagen Original, (2) GroundTruth, (3) Predicción, (4) ImagenOriginal+GroundTruth, (5) ImagenOriginal+Predicción.



5.6.3.- Comparaciones

Tanto la implementación de UNet en Matlab como la implementación en Keras obtienen resultados similares, sin embargo, su coste temporal es hasta cuatro veces inferior en el caso de Keras, pasando de dos horas y media a solamente media hora para un entrenamiento completo.

En cuanto a los resultados al utilizar las once clases contra utilizar solamente tres clases, ambas con BACKGROUND, parece que se obtiene una mejora sustancial al simplificar las clases a predecir ya que al unificar aquellas clases confusas se mejoran las métricas considerablemente. Solamente existe la excepción de la clase “CA (VIALES)” que tiende a clasificar las linderas de las parcelas como caminos, cuando en el GroundTruth no se etiquetan. Este problema parece ser aceptable, por lo que aunque la clase obtenga unos malos resultados, si se inspeccionan las imágenes de test podemos observar que es un comportamiento esperable, y que, en este caso, un humano no realizaría mejor esta tarea.

5.7.- Servicio de inferencia

En la sección 4.6 se especifica el prototipo de servicio web para el conjunto de datos PNOA. En esta sección se detalla una extensión con mayor funcionalidad de este prototipo web versionado para el conjunto de datos Sentinel-2. Este servicio tiene un funcionamiento similar de cara al usuario, sin embargo, en lugar de imágenes PNG, es necesario enviar imágenes en formato TIF/GEOTIF. Si se desea que los resultados devueltos estén georeferenciados la imagen debe estarlo de antemano. Además, las imágenes enviadas deben tener solamente las bandas de Sentinel-2 en el orden en el que aparecen en la tabla 5.11.



Bandas
B2 Blue
B3 Green
B4 Red
B5 VRE
B6 VRE
B7 VRE
B8 NIR
B8A Narrow NIR
B11 WIR
B12 SWIR

Tabla 5.11.- Bandas Sentinel-2 de las imágenes a enviar al prototipo web.

Este servicio permite predecir imágenes de cualquier tamaño ya que las recorta internamente de forma que de una imagen de gran tamaño se obtienen múltiples recortes de 256×256 píxeles, que, una vez realizadas sus respectivas predicciones, se unifican de vuelta a una sola imagen, devolviendo de forma transparente al usuario una predicción de la imagen completa.

Finalmente, el fichero devuelto, además de la imagen con la predicción, devuelve la misma imagen de entrada con una banda extra en la que se adjunta dicha predicción de forma que la máscara de la predicción se encuentra georeferenciada, y un fichero GEOJSON con el conjunto de polígonos detectados así como su tipo de cultivo (clase).

Para obtener los polígonos que definen las distintas regiones clasificadas se utiliza un método de poligonización, que utiliza como entrada la imagen predicha en escala de grises (donde cada valor de gris determina una clase), disponible en la librería GDAL bajo el nombre “gdal_polygonize”. A este método (ver código 5.5) se le pasa como argumentos la imagen de entrada y el directorio de salida donde almacenar el fichero shapefile (SHP) resultante con la información de los polígonos, además, hay que



especificar el número de píxeles a tener en cuenta para determinar regiones conexas. Este valor debe ser de ocho píxeles en lugar de cuatro, para evitar que se generen regiones no significativas, por ejemplo, en el caso de píxeles sueltos.

Código 5.5.- Ejemplo de gdal_polygonize.

```
gdal_polygonize Input_Image Output_Dir -8
```

Debido a que esta función devuelve una cantidad masiva de puntos que provocaría que el fichero GEOJSON fuera de gran tamaño así como un coste computacional muy alto a la hora de hacer cálculos con dichos datos, se decide simplificar las polilíneas de los polígonos. Para realizar dicha simplificación se decide utilizar el algoritmo de Ramer-Douglas-Peucker (RDP) por ser el mas utilizado y uno de los mas robustos. Finalmente, una vez se reducen dichos puntos, se transforman de píxeles a puntos georeferenciados gracias a la información de georeferencia de la imagen de entrada y se genera el fichero GEOJSON.

Después de un breve estudio de las diferentes tolerancias utilizables en el algoritmo RDP, se llega a la conclusión de que una tolerancia de 2 píxeles es el punto medio entre una buena reducción del volumen de los datos generados y que el polígono aun tenga la precisión suficiente sobre las regiones que delimita. En la figura 5.13 se muestra un ejemplo de una máscara de predicción compleja con las regiones poligonizadas ya simplificadas, donde cada región se dibuja de un color diferente de forma aleatoria. Para detallar aquellas regiones con “agujeros” se utiliza un conjunto de polígonos donde el primero especifica la región frontera y el resto sus “agujeros”.

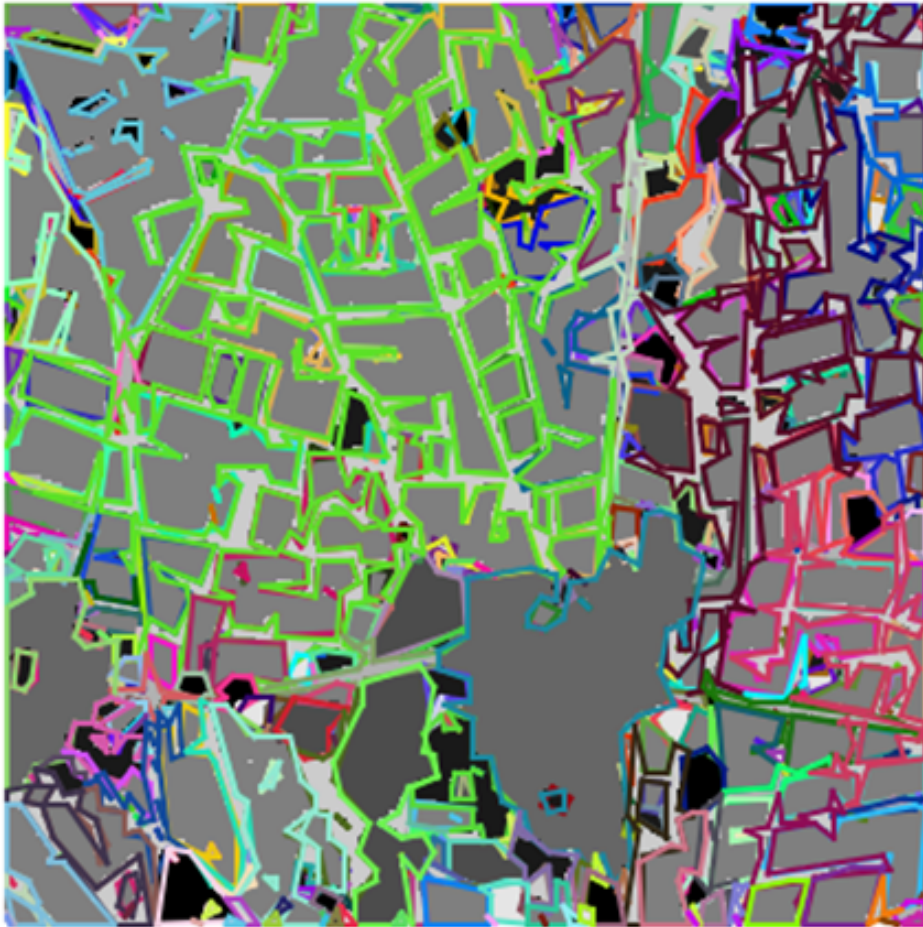


Figura 5.13.- Ejemplo de máscara predicha de Sentinel-2 poligonizada.

Finalmente, un ejemplo de la estructura del GEOJSON generado se muestra en el código 5.6, que contiene un array de entrada llamado “features” en donde se almacena cada región como tipo “Feature”.

Cada “Feature” declara la geometría de la región que contiene. Una geometría puede ser de una sola coordenada como Point, de dos como LineString y de mas de dos como Polygon o MultiPolygon dependiendo de si existen “agujeros” en la región. Se adjuntan como propiedades el tipo del cultivo y el valor del píxel (que coincide con el tipo de cultivo).

Código 5.6.- Ejemplo de GEOJSON.

```
{  
  "type": "FeatureCollection",  
  "name": "response_1_1.tif.shp",
```



```
"_comment": "CRS: WGS 84, Geometries: Multipolygon, Polygon, LineString, Point",
"features": [
  {
    "type": "Feature",
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [
            42.832965623071274, -3.496953499632623
          ],
          [
            42.8324757871663, -3.496463663727651
          ],
          [
            42.832965623071274, -3.496953499632623
          ]
        ]
      ]
    },
    "properties": {
      "classNumber": "1",
      "class": "PSPRPA"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [
            42.832965623071274, -3.496953499632623
          ],
          [
            42.8324757871663, -3.496463663727651
          ],
          [
            42.832965623071274, -3.496953499632623
          ]
        ]
      ]
    },
    "properties": {
      "classNumber": "1",
      "class": "PSPRPA"
    }
  }
]
```

La experimentación con el servicio de inferencia de esta implementación se realiza mediante un cliente y un servidor en máquinas diferentes que se comunican a través de la red. Sin embargo, los tiempos recogidos son aquellos registrados por el servidor, es decir, no se toman los tiempos de transferencia de las imágenes sobre la red, tanto para las imágenes de entrada como para el fichero comprimido de salida.

Este prototipo utiliza la implementación de Matlab para la predicción de tres clases, y obtiene unos tiempos de inferencia de unos ~5 segundos utilizando CPU para predecir una imagen, sin embargo, como se añade el procesamiento de poligonización y de generación del GEOJSON, así como el tiempo de carga del modelo, el tiempo



total aumenta a los ~ 15 segundos de media. Debido a que estos procesamientos se ejecutarán siempre en CPU, y el tiempo constante de carga del modelo, no parece existir un gran beneficio en el uso de GPU para un bajo número de imágenes, ya que para una sola imagen existe un máximo teórico de aceleración del $\sim 30\%$ y para un número lo suficientemente alto de imágenes del $\sim 80\%$, basándose en los tiempos expuestos en la tabla 5.12, por lo que no se ha implementado un prototipo con GPU. En el caso de utilizar diez imágenes el tiempo total aumenta hasta los ~ 68 segundos.

Estos tiempos están disponibles en la tabla 5.12. En estas pruebas no se tiene en cuenta el tiempo necesario para enviar y recibir las imágenes. “T. carga (s)” indica el tiempo necesario para cargar todo lo necesario en memoria (modelo, imágenes, Runtime de Matlab, etc), “T. de pred. (s)” es el tiempo de inferencia total, “T. resultados. (s)” muestra el tiempo total de los procesamientos, es decir, de la poligonización y generación del GEOJSON, y “T. total(s)” es el tiempo que se tarda desde que comienza la carga del modelo hasta que finalizan los procesamientos.

# Imágenes	T. carga (s)	T. de pred. (s)	T. resultados. (s)	T. total (s)
1	~ 9	~ 5	~ 1.5	~ 15
10	~ 9	~ 47	~ 12	~ 68

Tabla 5.12.- Tiempos de inferencia del prototipo Sentinel-2 con CPU.

6. Conclusiones

Las conclusiones de este Trabajo de Fin de Máster se agruparán en función de la resolución de las imágenes utilizadas en los conjuntos de datos, en concreto, primero se resumirán las conclusiones obtenidas al utilizar las imágenes de alta resolución del conjunto de datos PNOA, y seguidamente las imágenes de baja resolución del conjunto de datos Sentinel-2. Esta separación se debe a que ambos conjuntos de datos difieren en gran medida por lo que no son comparables directamente. La diferencia fundamental entre ambas experimentaciones, en este caso, es el GSD, que en Sentinel-2 es de 10 metros/píxel y en PNOA es de 0.25 metros/píxel (ver figura 6.1). Lo que parece indicar que un GSD de 10 metros/píxel tiene una resolución demasiado baja como para reconocer diferentes tipos de cultivos exitosamente utilizando solamente las bandas RGB. Afortunadamente, en el caso de Sentinel-2 se dispone de bandas multiespectrales que proveen de mayor nivel de información que al utilizar solamente las bandas RGB por lo que se consiguen resultados aceptables a pesar de la resolución de las imágenes.



Figura 6.1.- Comparativa entre imágenes RGB de PNOA (inferior) y de Sentinel-2 (superior) sobre la misma región.

En el conjunto de datos PNOA, con el mejor experimento realizado, obtiene una MeanRecall del 78% y una MeanPrecision del 76%. Estos resultados se obtienen en la implementación de DeeplabV3+ del GitHub oficial de Google por los autores de la red, utilizando la red backbone Xception41. Aunque, parece que cambiando tanto la implementación como la red backbone utilizada no varía enormemente los resultados obtenidos, ya que, en la implementación de Matlab con la red backbone de Mobilenetv2 se obtienen unos resultados de 72% y 73% de MeanRecall y



MeanPrecision respectivamente, de igual forma con el resto de redes backbone se obtienen resultados similares. A partir de las experimentaciones realizadas parece que el mejor solucionador de la red es Adam, con un valor de learning rate de entre 0.0005 y 0.00005, utilizando un balanceo de clases con la técnica de “Median Frequency Weighting”. Se utilizan 60 epochs por ser el punto óptimo del entrenamiento de la red, a partir del cual se comienzan a producir sobreajustes del modelo.

Si se pudiesen obtener bandas multispectrales similares a las del satélite Sentinel-2 para el conjunto de datos PNOA se podrían mejorar aun mas sus resultados, debido que este tipo de bandas añade una gran cantidad de información que, en el caso de Sentinel-2, nos permite trabajar incluso con un GSD muy elevado.

En el conjunto de datos Sentinel-2 se obtienen unos resultados de 66% y 58% de MeanRecall y MeanPrecision respectivamente al utilizar tres clases (con BACKGROUND). Si se utilizan las once clases sin unificar, se obtienen unos resultados de 56% y 42% de MeanRecall y MeanPrecision, por lo que unificar las clases confusas parece ser un método eficaz de mejorar las predicciones. El mejor solucionador de red para este caso es Adam, con un valor de learning rate de 0.005. Se utilizan 125 epochs por ser este el valor óptimo en el que el entrenamiento comienza a producir un sobreajuste del modelo. La profundidad de la red que provee de mejores resultados es de cuatro niveles, que coincide con el valor de la red original en la publicación del autor de la red. Y 32 filtros convolucionales para el primer nivel, a diferencia de los 64 de la publicación original. Aunque, estos dos últimos parámetros de profundidad de la red y de número de filtros en el primer nivel de la arquitectura de la red, no suponen una variación significativa a los resultados, afectando solamente sobre un 3% a las métricas, excepto en casos mas extremos.

Al utilizar solamente tres de las bandas de las trece bandas disponibles en Sentinel-2 se obtienen resultados completamente inútiles, inferiores al 10% de MeanRecall y MeanPrecision (para las once clases), independientemente de si estas tres bandas son las RGB o si son bandas infrarrojas. Los experimentos con solo tres bandas se pueden observar en el ANEXO A. Con seis bandas, se obtienen unos resultados que, aunque no son sorprendentes, son quizá más llamativos, llegando al 50% de MeanRecall con una

MeanPrecision que ronda el 40 %. Si se utilizan diez de las bandas o todas las bandas disponibles en Sentinel-2, se producen unos resultados similares a aquellos experimentos realizados con seis bandas, es decir, utilizar más de seis bandas parece no producir mejora alguna en los resultados de los modelos. Estas experimentaciones con diferente cantidad de bandas se encuentra en el ANEXO A.

En vista a estos resultados se concluye que al simplificar el modelo unificando varias clases similares en una sola se consigue mejorar la predicción de la red significativamente, entre un 15 % y 20 %, aunque sigue sin poder compararse con los resultados obtenidos por el conjunto de datos PNOA.

Para imágenes RGB DeepLabV3+ produce resultados mejores a UNet, sin embargo, en dicha red no se pueden utilizar imágenes con mas de tres bandas debido a que se apoya en redes backbone que solo soportan RGB. Finalmente, en las imágenes de Sentinel-2 al tener tan baja resolución se dificulta la tarea del etiquetado de las máscaras GroundTruth sobre aquellas parcelas que sean de un tamaño reducido o de una anchura reducida como es el caso de la clase CA (VIALES). Este efecto se puede observar en la figuras 6.2, 6.3 y 6.4 al observar los PASTOS de color amarillo que se encuentran en la lindera de lo que parece ser un camino.



Figura 6.2.-
Recorte original.

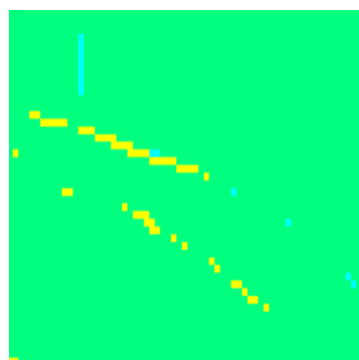


Figura 6.3.-
GroundTruth.

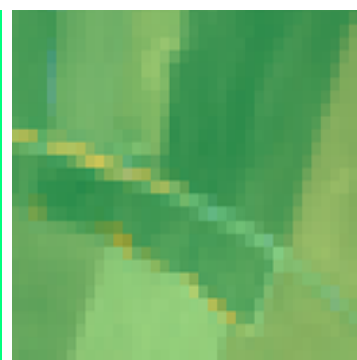


Figura 6.4.-
Superposición.

En todos los casos, las implementaciones en Matlab son significativamente mas lentas que las implementaciones realizadas en Tensorflow.



Finalmente, los servicios de inferencia implementados demuestran que no hace falta un equipo con altas prestaciones y una GPU dedicada para ofrecer predicciones rápidamente, ya que, los tiempos de inferencia son despreciables en el caso de que se utilicen pocas imágenes si se tienen en cuenta los tiempos de carga del modelo, de las transferencias por la red y de los procesamientos añadidos para la poligonización y la obtención de los GEOJSON. En el caso de que se quieran predecir muchas imágenes de una sola vez, el uso de una GPU sí que supondría un mayor beneficio. La principal ventaja de los servicios implementados es la capacidad de despliegue gracias a su contenerización, ya que se puede realizar un despliegue del servicio mediante un solo comando “docker-compose up”. La mayor desventaja de estos servicios web es la necesidad de enviar imágenes en un formato concreto muy estricto para que la red haga sus predicciones correctamente, es decir, no se pueden utilizar imágenes con bandas en diferente orden, con distinto número de bandas, GSD, o incluso resolución si no se aplicase un cropping de la imagen de entrada.

Referencias

- [1] V. Natoli, “A decade of accelerated computing augurs well for gpus,” *Nextplatform.com* <https://www.nextplatform.com/2019/07/10/a-decade-of-accelerated-computing-augurs-well-for-gpus/>, 2019.
- [2] V. Patel, H. Shah, and Y. Farooqui, “Hybrid feature based prediction of suicide related activity on twitter,” in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 590–595, 2020.
- [3] J. A. Rodrigo, “Máquinas de vector soporte (support vector machines, svms),” *Cienciadedatos.net* https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines, 2017.
- [4] N. Vimal Babu, Ramakrishan, “Modeling with multilayer perceptron for detection of fuel adulteration using python programming,” *Petroleum and Chemical Industry International*, 2018.
- [5] M. Bhide, “Using a convolutional neural network to create a program that detects face masks,” *Medium* <https://medium.com/@malharbhide/using-a-convolutional-neural-network-to-create-a-program-that-detects-face-masks-80fb04827bb>, 2020.
- [6] S. Saha, “A comprehensive guide to convolutional neural networks,” *Towardsdatascience.com* <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, 2018.
- [7] A. Garcia-Garcia, “A review on deep learning techniques applied to semantic segmentation,” *Deepai.org* <https://deepai.org/publication/a-review-on-deep-learning-techniques-applied-to-semantic-segmentation>, 2017.



- [8] R. Keim, “Understanding simple neural network training,” *Allaboutcircuits.com* <https://www.allaboutcircuits.com/technical-articles/understanding-simple-neural-network-training-and-learning/>, 2019.
- [9] S. Aleshin-Guendel, “Examining the structure of convolutional neural networks,” 2017.
- [10] Z. Yubo, Y. Zhuoran, Y. Jiuchun, Y. Yuanyuan, W. Dongyan, Z. Yucong, Y. Fengqin, Y. Lingxue, C. Liping, and Z. Shuwen, “A novel model integrating deep learning for land use/cover change reconstruction: A case study of zhenlai county, northeast china,” *Remote Sensing*, vol. 12, no. 20, p. 3314, 2020.
- [11] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.
- [12] P.-L. Pröve, “An introduction to different types of convolutions in deep learning,” *Towardsdatascience.com* <https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d>, 2017.
- [13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.
- [14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [15] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.



- [16] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [17] P. Saurabh, “Semantic segmentation: Introduction to the deep learning technique behind google pixel’s camera,” *analyticsvidhya.com* <https://www.analyticsvidhya.com/blog/2019/02/tutorial-semantic-segmentation-google-deeplab/>, 2019.
- [18] K. Ragunathan, “Convolutional neural network for link prediction based on subgraphs in social networks,” 2020.
- [19] A. Rosebrock, “Intersection over union (iou) for object detection,” *Pyimagesearch.com* <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, 2016.
- [20] E. Tiu, “Metrics to evaluate your semantic segmentation model,” *Towardsdatascience.com* <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>, 2019.
- [21] “Sentinel 1: Satellite imagery, overview and characteristics.” <https://eos.com/sentinel-1/>. Accedido: 2020-01-04.
- [22] “Todo lo que deberías saber sobre imágenes sentinel 2.” <http://www.gisandbeers.com/lo-deberias-saber-imagenes-sentinel-2/>. Accedido: 2020-01-04.
- [23] “El satélite landsat 8: Imágenes, descripción y características..” <https://eos.com/es/landsat-8/>. Accedido: 2020-01-04.
- [24] “Gaofen-1 - satellite missions.” <https://directory.eoportal.org/web/eoportal/satellite-missions/g/gaofen-1>. Accedido: 2020-01-04.
- [25] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, “Deep learning classification of land cover and crop types using remote sensing data,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, 2017.



- [26] Z. Zhou, S. Li, and Y. Shao, “Crops classification from sentinel-2a multi-spectral remote sensing images based on convolutional neural networks,” in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 5300–5303, 2018.
- [27] S. Sharma, “Epoch vs batch size vs iterations,” *Towardsdatascience.com* <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>, 2019.
- [28] H. Zulkifli, “Understanding learning rates and how it improves performance in deep learning,” *Towardsdatascience.com* <https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10>, 2018.
- [29] S. Lau, “Learning rate schedules and adaptive learning rate methods for deep learning,” *Towardsdatascience.com* <https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1>, 2017.
- [30] R. Khandelwal, “L1 l2 regularization,” *Medium* <https://medium.com/data-driven-ives-to-r/l1-l2-regularization-7f1b4fe948f2>, 2019.
- [31] V. Kakaraparthi, “Xavier and he normal (he-et-al) initialization,” *Medium* <https://medium.com/@prateekvishnu/xavier-and-he-normal-he-et-al-initialization-8e3d7a087528>, 2018.
- [32] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding,” *arXiv preprint arXiv:1511.02680*, 2015.
- [33] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” *arXiv preprint arXiv:1505.07293*, 2015.
- [34] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” *arXiv preprint arXiv:1412.7062*, 2014.



- [35] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 82–92, 2019.

Anexos



ANEXO A:

Contiene todos los experimentos relativos a la red UNet en su implementación para Matlab con el conjunto de datos de Sentinel-2.

ANEXO B:

Contiene todos los experimentos relativos a la red UNet en su implementación para Keras con el conjunto de datos de Sentinel-2.

ANEXO C:

Contiene todos los experimentos relativos a la red DeepLabV3+ en su implementación para Matlab con el conjunto de datos de PNOA.

ANEXO D:

Contiene todos los experimentos relativos a la red DeepLabV3+ en su implementación para Tensorflow con el conjunto de datos de PNOA.