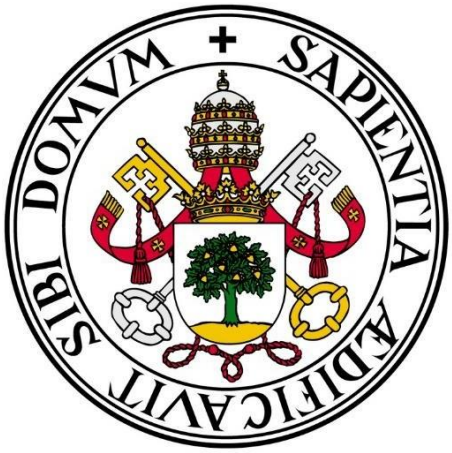


UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO DE INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

EVALUACIÓN DEL USO DE REDES NEURONALES CONVOLUCIONALES PARA CLASIFICACIÓN DE CULTIVOS MEDIANTE IMAGEN MULTIESPECTRAL

Autor:

D. Sergio González Diez

Tutores:

Dr. D. Mario Martínez Zarzuela
Dr. D. Francisco J. Díaz Pernas

Valladolid, 25 de julio de 2019

Evaluación del uso de Redes Neuronales Convolucionales para clasificación de cultivos mediante imagen multiespectral

AUTOR: **D. Sergio González Diez**
TUTOR: **Dr. D. Mario Martínez Zarzuela**
DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE: **Dr. D. Francisco J. Díaz Pernas**
SECRETARIO: **Dr. D. Mario Martínez Zarzuela**
VOCAL: **Dr. D. David González Ortega**
SUPLENTE 1: **Dra. Dª. Miriam Antón Rodríguez**
SUPLENTE 2: **Dr. D. Carlos Gómez Peña**

FECHA: **25 de julio de 2019**

CALIFICACIÓN:

Agradecimientos

Me gustaría agradecer al Dr. D. Mario Martínez Zarzuela y Dr. D. Francisco J. Díaz Pernas por su ayuda constante y consejos a lo largo de este Trabajo Fin de Grado.

A mis padres, mis hermanos, por su ayuda, ánimo y apoyo incondicional.

Agradecer al Instituto Tecnológico Agrario de Castilla y León por facilitar los datos y por permanecer en permanente contacto para resolver dudas acerca de su algoritmo de clasificación y la organización de los datos.

Por último, agradecer a mi pareja y amigos, por saber escucharme y aconsejarme en todo momento.

Resumen

El objetivo de este Trabajo Fin de Grado es la implementación de una red neuronal artificial capaz de realizar la segmentación semántica de distintos cultivos a partir de imágenes satelitales. La red tiene que extraer características y patrones para segmentar imágenes satelitales en cualquier época del año y de cualquier tipo de cultivo o vegetación.

Primero, se proporcionará una base teórica del aprendizaje automático, centrándose en aprendizaje profundo, en particular, profundizando en redes convolucionales, especialmente útiles en la clasificación de imágenes. A continuación, se explicarán los métodos tradicionales en la clasificación de cultivos, enfatizando en la técnica de aprendizaje automático actualmente utilizada por el Instituto Tecnológico de Castilla y León. Después de este análisis, se describirán las decisiones tomadas en el pre procesamiento de los datos, justificando cada una de ellas. Finalmente, se presentarán los resultados, terminando exponiendo las conclusiones y líneas futuras de trabajo, a partir de este Trabajo de Fin de Grado.

Palabras clave

Cultivos, Imágenes Satelitales, Aprendizaje Profundo, Redes neuronales.

Abstract

The main objective of this Master Thesis is the implementation of an artificial neural network, which is able to perform the semantic segmentation of different crops from satellite images. The network has to extract characteristics and patterns to segment satellite images in any type of crop or vegetation at any stage of the year.

First of all, a theoretical basis of machine learning is provided. It is focused on deep learning, specifically in deepening in convolutional networks, which are useful in classifying images. Next, the traditional methods in crop classification will be explained, emphasizing in the technique of machine learning currently used by the Technological Institute of Castilla y León. After this analysis, the decisions made in the preprocessing of the data will be described and each of them will be justified. Finally, the results will be presented, the conclusions will be exposed upon completion and ending with a proposal of future research lines.

Keywords

Crops, Satellite Images, Deep Learning, Neural Networks.

Índice

1 Introducción **11**
1.1 Motivación y objetivos 12
1.2 Fases 13
1.3 Hardware 14
1.4 Software 15
1.5 Estructura de la memoria 18
2. Aprendizaje automático **21**
2.1 Aprendizaje automático clásico 21
2.1.1 Conjunto de Datos 21
2.1.2 *Overfitting y Underfitting* *22*
2.1.3 Métricas 22
2.1.4 Regresión lineal y logística 24
2.1.5 Maquina de vectores de soporte 24
2.1.6 Árboles de decisión 25
2.1.7 Bosques aleatorios 26
2.1.8 Redes neuronales artificiales 27
2.2 Aprendizaje profundo 28
2.2.1 Parámetros e hiperparámetros 29
2.2.2 Proceso de aprendizaje de una red neuronal 29
2.2.3 Redes neuronales convolucionales 32
2.2.4 Sobreajuste en aprendizaje profundo 33
3 Revisión del estado de conocimiento **37**
3.1 Técnica de clasificación de cultivos 37
3.2 Técnica ITACyL en la clasificación de cultivos 39
4 Desarrollo de redes neuronales **41**
4.1 Diseño *dataset* 41
4.2 Autoencoder SegNet 46
4.3 Entrenamiento ventana deslizante 47
4.4 Entrenamiento basado en regjones 48
4.4.1 Entrenamiento anual 50
4.4.2 Entrenamiento mensual 58
5 Conclusiones y visiones futuras **63**
5.1 Conclusiones 63
5.2 Líneas futuras 63
Referencias 66

Índice de figuras

Capítulo 1 Introducción ***11***

Figura 1 .Logotipo de Anaconda
Figura 2. Logotipo de Python
Figura 3. Logotipo de TensorFlow
Figura 4. Logotipo de Keras
Figura 5. Logotipo de Qgis
Figura 6. Logotipo de Jupyter

Capítulo 2 Aprendizaje Automático ***21***

Figura 7. Gráficas de overfitting y underfitting.
Figura 8.Gráfica con la terminología definida y ejemplo de la construcción de dos métricas.
Figura 9. Gráfica de regresión lineal
Figura 10. Ejemplo de clasificador SVM
Figura 11.Ejemplo de árbol de decisión para tomar la decisión de jugar al golf.
Figura 12.Ejemplo de poda de un árbol de decisión, en el que se eliminan nodos intermedios.
Figura 13.Ejemplo de bosque profundo, en el cual se tiene 4 árboles de decisión con idénticas salidas y se decide por la salida más repetida.
Figura 14. Representación gráfica del Perceptrón
Figura 15.Representación de las capas a lo largo de la red neuronal.
Figura 16. Gráfica de función de activación ReLu
Figura 17. Gráfica de función de activación Sigmoide
Figura 18. Gráfica de función de activación Tangente hiperbólica
Figura 19. Se representa una variación a lo largo de la función de pérdida
Figura 20. Operación de convolución 2D.
Figura 21. Gráfico correspondiente a la primera y segunda capa de una red convolucional. Hay 32 filtros diferentes.
Figura 22. Ejemplo de Data Augmentation

Capítulo 3 Revisión del estado del conocimineto ***37***

Figura 23. Ejemplo de Convolucion 3D en cultivos
Figura 24. Red Convolucional·3D
Figura 25. Ejemplo de red Recurrente (RNN)
Figura 26. Resultados árbol de decisión. Se observan métricas como FScore o el índice Kappa

Capítulo 4 Desarrollo de redes neuronales ***41***

Figura 27. Límites terrestres del conjunto de datos
Figura 28. Imagen real recortada con su correspondiente etiqueta
Figura 29. Fechas correspondientes a las imágenes satelitales.
Figura 30. Descripción, valor de 8 bits asociado, porcentaje y el color asociado a cada clase por ITACyL.
Figura 31. Mapa de precipitaciones medias históricas en la zona de interés
Figura 32. Media y desviación tipica de cada canal en los datos de entrenamiento
Figura 33. A partir de las imágenes satelitales y los pixeles etiquetados obtenemos las imágenes de entrenamiento.
Figura 34. Esquema de inferencia. Ground truth se utiliza para el cálculo de las métricas
Figura 35. Modelo VGG16

Figura 36. Autoencoder SegNet
Figura 37. Ejemplo inferencia
Figura 38. 19 clases elegidas
Figura 39. Número de puntos y contornos cerrados para los datos de ITACyL
Figura 40. Distancia en píxeles de la esquina superior izquierda al centro de masas de la clase Urbano-viales.
Figura 41. Representación de la expresión de soft dice loss.
Figura 42. Recall (sensitivity) y loss a lo largo de 150 épocas
Figura 43. Sensitivity dice y métrica X
Figura 44. Matriz de confusión
Figura 45. Imagen de test, predicción y etiqueta correspondiente
Figura 46. Matriz de confusión para modelo con poco peso para cebada.
Figura 47. Matriz de confusión para modelo con poco peso para cebada y trigo.
Figura 48. Métricas para el modelo con poco peso para la cebada y trigo
Figura 49. Curvas de recall y loss en 300 épocas
Figura 50. Matriz de confusión.
Figura 51. Curvas de accuracy y loss en 300 épocas
Figura 52. Matriz de confusión.

Capítulo 5 Conclusiones y visiones futuras ***63***

Anexo ***69***

Figura 53. Inferencia para una imagen de CityScapes
Figura 54. Gráfica de la evolución de Recall y la loss a lo largo de las 150 épocas.
Figura 55. Gráfica de la evolución de la accuracy y la loss a lo largo de las 150 épocas.
Figura 56. Ejemplo de predicción de una imagen de CityScapes.

Capítulo 1. Introducción

En este capítulo se describen los objetivos de este Trabajo Fin de Grado, enmarcándolo en su escenario de trabajo.

El Grupo de Telemática e Imagen (GTI) es un grupo de investigación de la Universidad de Valladolid (UVA) en el que se desarrollan tareas de procesado de imagen. En los últimos años, en este grupo se han hecho labores de visión artificial y algoritmos de aprendizaje por computador. Una de estas tareas es la clasificación de imágenes mediante algoritmos de Aprendizaje Automático (*Machine Learning*), con la finalidad de detectar movimientos en aplicaciones de Realidad Virtual, tales como Juegos Serios para rehabilitación física y cognitiva. En otros trabajos, desarrollan algoritmos para la detección automática de enfermedades en imagen médica a través de algoritmos de Aprendizaje Automático.

En el ámbito de la agricultura, un tema crucial es la clasificación de cultivos. La escasez de personal cualificado y el tiempo y dinero empleados en la clasificación de cultivos, al realizarse manualmente resulta una tarea muy costosa. La teledetección es una técnica usada para obtener información sin estar presente de manera física. En los últimos años se ha producido un aumento de la cantidad y calidad de las imágenes satelitales, facilitando la tarea de la obtención de datos de buena calidad.

El Aprendizaje Profundo (*Deep Learning*) es una técnica dentro del aprendizaje automático. El aprendizaje automático a su vez se considera una las ramas de la Inteligencia Artificial. Una de las utilidades del aprendizaje profundo es la segmentación de imágenes, clasificando la totalidad de sus píxeles de manera individual.

Este Trabajo de Fin de Grado, está enfocado en el entrenamiento de una red neuronal profunda para que consiga detectar y segmentar los cultivos mediante fotografías satelitales multiespectrales. Con esto es posible clasificar píxel a píxel los distintos cultivos y otras estructuras artificiales en el terreno. De esta manera se puede lograr inferir qué cultivo se encuentra en una parcela de terreno.

Existen diferentes técnicas históricamente utilizadas para la clasificación de cultivos, las más empleadas son RF *(random forest)*, SVM *(Support Vector Machines)* o XGBoost. Sin embargo, en los últimos tiempos, se han utilizados métodos de aprendizaje profundo para la clasificación de cultivos a partir de las imágenes capturadas. Se emplean distintos tipos de redes neuronales para la clasificación: CNNs, RNNs, ANNs.

1.1 Motivación y objetivos

Este trabajo tiene por objetivo desarrollar una herramienta capaz de segmentar de manera automática las imágenes obtenidas por los satélites. Se busca la comparación de resultados entre las técnicas tradicionales y la técnica propuesta.

Hoy en día, cualquier usuario puede obtener imágenes satelitales sin ningún problema mediante *Google Earth*. Estas imágenes contienen el espectro visible (RGB), siendo estas tres bandas una limitación para los sistemas de clasificación existentes. Es posible aumentar la confiabilidad en los algoritmos de clasificación usando todas las bandas que sean posibles obtener a través de los satélites.

En el ámbito de la agricultura, las aplicaciones de teledetección se encuentran en torno a la estimación de cosechas y clasificación de cultivos, entre muchas otras. Para lograr todo esto, se extraen características de las imágenes para localizar patrones e ir aprendiendo a partir de las propias imágenes. Apoyándose en el procesamiento de imágenes y la computación inteligente se construyen algoritmos automáticos para reconocer y segmentar cultivos.

Para este Trabajo de Fin de Grado se han usado datos del Instituto Tecnológico Agrario de Castilla y León (ITACyL). Tradicionalmente, ITACyL obtenía información aérea de distintas formas [1].

- **La ortofoto** es una capa de fotografía aérea. Este material esta georreferenciado. Se genera a partir de cientos de fotos individuales a través de cámaras digitales situadas en los aviones. El Plan Nacional de Ortofotografía Aérea establece la renovación de las fotos cada 3 años (2017, 2014,2011). Resolución de 25 o 50 cm de terreno en cada pixel.

- **Los fotogramas de vuelo**, son las fotografías originales de las cámaras aéreas.

Actualmente existen una gran cantidad de satélites con lo cual se tiene un montón de datos en forma de imágenes provenientes del espacio [2].

ITACyL realiza mapas de cultivos de la Comunidad de Castilla y León. Los satélites que captan las imágenes son Landsat-8 (NASA), Sentinel 2-A/B (ESA) y Deimos-1 (Deimos Imaging). En los últimos tiempos se puede tener fotografías de la misma zona en intervalos cada vez más cortos de tiempo, siendo esto crucial para obtener información de carácter temporal en los cultivos.

Recientemente, las redes neuronales de aprendizaje profundo han alcanzado fama en los modelos de aprendizaje automático debido a la extracción de características y reconocimiento de patrones en las imágenes. Gracias a la estructura jerárquica de las distintas capas de la red neuronal, es posible extraer tanto características generales de las imágenes en las primeras capas (bordes) como características específicas de las imágenes en las últimas capas de la red neuronal.

Tras una gran búsqueda de información y su estudio, un conjunto de datos proporcionado por el Instituto Tecnológico Agrario de Castilla y León, se decidió la utilización del aprendizaje profundo para lograr superar otros mecanismos de clasificación de aprendizaje automático como puede ser los árboles de decisión, actualmente usado por el Instituto Tecnológico Agrario de Castilla y León.

Con lo cual, el objetivo principal de este Trabajo Fin de Grado es el diseño de una red neuronal capaz de segmentar distintos tipos cultivos en imágenes satelitales. La aplicación de los modelos en el conjunto de datos de imágenes satelitales a lo largo de 9 meses aprovecha la información espacial, espectral y temporal de las imágenes satelitales en las distintas zonas geográficas.

1.2 Fases

Las fases seguidas en este Trabajo Fin de Grado se mencionan a continuación:

- **Se realizó un estudio sobre métodos de aprendizaje automático para la clasificación de imágenes satelitales multiespectrales**: Para este trabajo ha sido necesario el estudio de técnicas usadas en la clasificación de cultivos, obteniendo las respectivas ventajas e inconvenientes de cada técnica. Se basó mayormente en el estudio de árboles de decisión y mecanismos de aprendizaje profundo. Se estudiaron ventajas en algunos modelos de redes neuronales profundas sobre el algoritmo del árbol de decisión. Se consultaron artículos científicos a este respecto [3] – [10].
- **Estudio de Aprendizaje Profundo**: una vez decidido el algoritmo de clasificación a usar, fue necesario un estudio detallado de la técnica para aplicarlo al caso de estudio.
  Para usar la técnica de aprendizaje profundo es fundamental tener claro los componentes teóricos sobre aprendizaje automático y aprender sobre los distintos modelos neuronales existentes Se consultaron libros [11,12] y realizaron cursos NVIDIA y Google para afianzar el conocimiento de redes neuronales. [13,14]
- **Búsqueda de Hardware**: Para lograr entrenar las redes neuronales de manera eficiente ha sido necesario valorar distintas opciones para poder tener suficiente memoria de almacenamiento. Los datos de las imágenes satelitales a lo largo de 9 meses ocupan grandes cantidades de memoria. De manera paralela es necesario tener suficientes GPUs con el fin de poder sostener el entrenamiento de la red. En algunos casos se produce un compromiso entre estos dos factores
- **Estudio de Python.** En concreto se aprendió sobre módulos y herramientas para procesar imágenes de grandes dimensiones y múltiples bandas espectrales y módulos para trabajar con redes neuronales de manera eficiente (TensorFlow y Keras) [15] – [21].
- **Estudio de modelos**: Se obtuvo las ventajas y desventajas de los distintos modelos de aprendizaje profundo para el caso de estudio [22,23].

- **Pre procesamiento de datos**: Tras la obtención de los datos por parte de ITACyL fue necesario observar la distribución del mismo y adaptar estas imágenes para la red neuronal. Esto es fundamental para lograr una buena partición de los datos consiguiendo que la red neuronal "aprenda" de manera efectiva.
- **Conseguir una red neuronal configurable.** Implementar en Keras una red que permita cambiar todos los parámetros e hiperpárametros de una red neuronal de manera eficiente.
- **Realización de pruebas**. Variando la configuración de la red neuronal elegida y observando los resultados obtenidos. Para cada diseño elegido, se realizan pruebas y evaluación de los resultados con el fin de obtener una red neuronal óptima.

1.3 Hardware

Para las primeras pruebas se hizo uso de un ordenador portátil con las siguientes características:

- Un procesador Intel® Core™ i7-5500U CPU @ 2.40Ghz (4 CPUs)
- Memoria RAM: 8192 MB
- Una tarjeta gráfica: NVIDIA GeForce 840M.
    - Memoria: 2048 MiB
- Almacenamiento en disco: 448 GB

Para la finalización de este Trabajo Fin de Grado, se ha hecho uso del siguiente equipamiento hardware disponible en el Laboratorio de GTI (Dilbert):

- Dos procesadores Intel® Xeon® X5650 con las siguientes características cada uno (Intel Corporation, n.d.):
    - Velocidad de reloj: 2.67 GHz
    - Número de núcleos: 6
    - Cantidad de subprocesos: 12
- Memoria RAM: 24 GB
- Dos tarjetas gráficas:
    - GeForce GTX 1080 Ti (NVIDIA Corporation, 2017)
        - CUDA *Cores*: 3584
        - Frecuencia de reloj normal: 1480 MHz
        - Frecuencia de reloj acelerada: 1582 MHz
        - Velocidad de la memoria: 11 Gb/s
        - Configuración de memoria estándar: 11GB GDDR5X
        - Ancho de banda de memoria: 484 GB/s
    - GeForce GTX 970 (NVIDIA Corporation, 2014)
        - CUDA *Cores*: 1664
        - Frecuencia de reloj normal: 1050 MHz
        - Frecuencia de reloj acelerada: 1178 MHz
        - Velocidad de la memoria: 7 Gb/s

    - Configuración de memoria estándar: 4 GB GDDR5
    - Ancho de banda de memoria: 224 GB/s
- Almacenamiento: 8.1 T

Para el trabajo a realizar, este servidor ha estado compartido en todo momento, pudiéndose aprovechar solamente una GPU para el entrenamiento.

1.4 Software

Anaconda

Es una distribución de Python de código abierto, ampliamente utilizado debido a su facilidad para la instalación y control de los paquetes de software. Es usado, por ejemplo, en técnicas de Aprendizaje Automático (*Machine Learning*), haciendo sencillo la instalación y manejo de software (Ej: TensorFlow).

Funciona correctamente tanto para los SO de Windows, MacOS y Linux.

Se ha usado esta distribución para instalar TensorFlow en el ordenador portátil para el estudio y las primeras pruebas de aprendizaje profundo.

Figura 1. Logotipo de Anaconda.

Python

Es un lenguaje de programación orientado a objetos e interpretado, es decir, el código se ejecuta línea a línea por un intérprete. Además es un lenguaje indexado que hace que sea muy legible por el usuario.

Este lenguaje fue creado por Guido van Rossum a finales de los años ochenta.

Para la realización de este Trabajo Fin de Grado se ha usado la versión 3.6 de este lenguaje de programación. En concreto ha sido muy importante el módulo **rasterio**, capaz de trabajar con imágenes satelitales de grandes dimensiones y de múltiples bandas espectrales.

Figura 2. Logotipo de Python

TensorFlow

Es una plataforma de código abierto dedicado al aprendizaje automático. Permite a la comunidad herramientas, bibliotecas y recursos que permiten desarrollar aplicaciones basadas en el aprendizaje automático. TensorFlow está basado en el uso de Tensores, eficientes computacionalmente

Originalmente fue desarrollado por Google, utilizado grafos de flujo de datos haciendo muy eficiente la computación.

Para la realización de este Trabajo Fin De Grado el uso de TensorFlow se ha limitado a utilizarse como el *Backend* de Keras.

Figura 3. Logotipo de TensorFlow

Keras

Es una API de alto nivel destinado a redes neuronales, capaz de ejecutarse sobre TensorFlow o Theano. Keras contiene múltiples implementaciones de los bloques de las redes neuronales.

Este módulo Python de redes neuronales profundas destaca por su facilidad de uso, la rápida creación de modelos y su funcionamiento con CPUs Y GPUs.

Este módulo se ha utilizado para la creación de los distintos modelos de redes neuronales usados en el Trabajo de Fin de Grado.

Figura 4. Logotipo de Keras

QGIS

Es un Sistema de Información Geográfica de código abierto disponible en Windows, Unix MacOS y Android. Es capaz de trabajar con numerosos formatos de datos vectoriales o rasterizados y con bases de datos.

Figura 5. Logotipo de Qgis

Jupyter

Jupyter Notebook es una aplicación web lanzada en 2015 desarrollada por el Proyecto Jupyter. Crea y comparte documentos web (JSON) que contienen celdas de entrada y de salida ordenadas. Estas celdas contienen código (Python), texto (Markdown), ecuaciones matemáticas y contenido multimedia. La gran ventaja es la legibilidad ya que es posible analizar los resultados de las celdas ejecutadas y compararlos con posibles tablas y figuras.

Figura 6. Logotipo de Jupyter

1.5 Estructura de la memoria

Esta memoria está estructurada en capítulos, en ellos se describen las diferentes acciones realizadas para lograr los objetivos de este Trabajo Fin de Grado:

En el primer capítulo se presenta el caso de estudio, la clasificación de cultivos, definiendo las motivaciones de este.

En el segundo capítulo, se estudiará diferentes técnicas de aprendizaje automático, profundizando en las de especial interés para este Trabajo fin de Grado.

El tercer capítulo se describirá la información recopilada a lo largo de la preparación del Trabajo Fin de Grado. Se discutirá la importancia sobre la clasificación de cultivos, modelos de aprendizaje automático capaces de lograrlo, información de los datos adquiridos y algoritmo de clasificación usado por el Instituto Tecnológico Agrario de Castilla y León.

El cuarto capítulo se mostrará el distinto pre procesamiento de datos, distintas variaciones de los modelos de aprendizaje profundo con sus respectivos parámetros.

En el quinto capítulo, se mostrará los resultados obtenidos para los distintos casos estudiados anteriormente.

Para terminar, en el sexto capítulo se discutirá sobre las conclusiones obtenidas y las posibles líneas futuras de trabajo.

Adicionalmente, se incluye un anexo. En este anexo se muestran los resultados de una red de segmentación semántica a lo largo de 12 clases seleccionadas del *dataset* público *CityScapes*. Se muestra como problema previo a nuestro caso de estudio, con el fin de lograr conocimientos de redes neuronales para la segmentación semántica de imágenes.

Capítulo 2. Aprendizaje Automático

Aprendizaje automático está definido como el estudio y diseño de herramientas informáticas las cuales son capaces de aprender de los datos y tomar decisiones futuras.

En el aprendizaje automático se encuentran diferentes tipos de problemas: aprendizaje no supervisado, aprendizaje por refuerzo o aprendizaje supervisado.

El aprendizaje no supervisado consiste en realizar el entrenamiento de los datos sin etiqueta obligando al algoritmo a encontrar patrones en el conjunto de datos.

El aprendizaje por refuerzo está enfocado al aprendizaje en función de la retroalimentación que obtenga de sus acciones, es decir, un aprendizaje basado en el ensayo-error.

El aprendizaje supervisado trata de aprender a partir de los datos etiquetados, es decir, contienen la respuesta correcta. El sistema al final del entrenamiento tendrá que ser capaz de predecir datos no observados anteriormente.

Un concepto importante en todos los algoritmos de aprendizaje automático es la generalización, para conseguir aplicar el modelo para cualquier ejemplo y lograr tener la predicción deseada. Se trata de ser capaz de decidir correctamente sobre datos desconocidos a partir de ejemplos perfectamente conocidos. En sus diferentes técnicas, el aprendizaje automático combina las matemáticas, ciencias de la computación, estadística y otras áreas de conocimiento.

2.1 Aprendizaje automático clásico

2.1.1 Conjunto de Datos

Generalmente los datos utilizados en un algoritmo se dividen en tres grandes bloques:

- **Datos de entrenamiento**: estos datos son utilizados para adquirir los parámetros del modelo.
- **Datos de validación:** sirven para comprobar como de bueno es el algoritmo e ir modificando los **hiperparámetros** para obtener mejores resultados. No modifican ningún parámetro (pesos y sesgos).
- **Datos de test:** están reservados para la evaluación final del algoritmo. Estos datos por lo general son secretos y sirven para comparar distintos algoritmos y ver cual se comporta mejor ante datos nunca observados.

2.1.2 *Overfitting y Underfitting*

En aprendizaje automático, una de las principales causas de no obtener los resultados deseados son el *overfitting* y el *underfitting*.

- ***Underfitting:*** se produce cuando no se dispone de muestras suficientes para que el algoritmo generalice el aprendizaje. Supóngase que se tiene muy pocas muestras de perros, es imposible que el modelo extraiga características y logre clasificar cualquier perro de la manera correcta.
- ***Overfitting*** ocurre cuando se dispone de datos suficientes de una misma naturaleza y el modelo se aprendió muy bien las imágenes de entrenamiento, sin embargo, en una pequeña variación de los datos nuestro modelo fallará. Si se desea un algoritmo que detecte la presencia o ausencia de un gato en una imagen, pero todos nuestros datos de entrenamiento contienen gatos de color blanco. En este caso, el algoritmo será capaz únicamente de clasificar bien los gatos de color blanco.

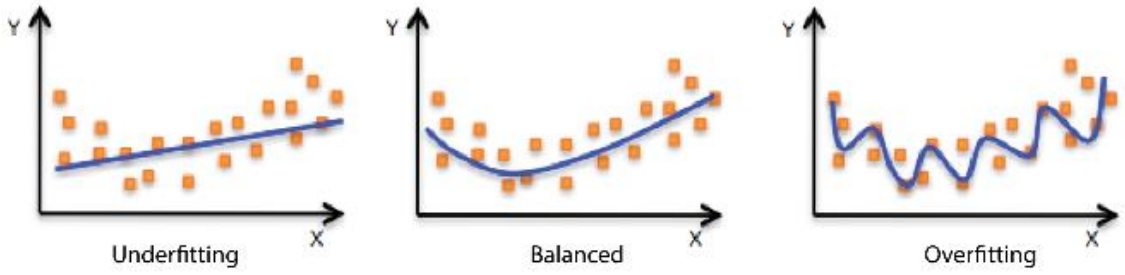


Figura 7. Gráficas de overfitting y underfitting [14].

El objetivo principal al crear un algoritmo de aprendizaje automático es lograr la generalización de los datos, se tiene que lograr buenos resultados con datos que nunca se ha observado. Para lograr esto, los datos tienen que ser representativos, es decir, abarcar todo el conjunto de datos. Debemos encontrar el equilibrio entre el *underfitting* y el *overfitting*, como se muestra en la Figura 7. Una buena práctica es la partición del conjunto de datos, se separa los datos en datos de entrenamiento y validación de manera que existan muestras diversas en ambos conjuntos. Existe un compromiso entre el número de datos de entrenamiento y validación, ya que con un número pequeño de datos de entrenamiento se puede caer en *underfitting*, en cambio sí se dispones de pocos datos de validación no podremos observar si nuestro algoritmo generaliza correctamente.

2.1.3 Metricas

En los resultados, se usa una cierta terminología para evaluar el rendimiento del algoritmo. [24,25]

Siendo X una clase genérica.

True positive (TP): todas las instancias de la clase X clasificadas como X.

True negative (TN): todas las instancias de la clase X, no clasificadas como X.

False positive (FP): todas las instancias no pertenecientes a la clase X, clasificadas como X.

False negative (FN): todas las instancias pertenecientes a la clase X, no clasificadas como X.

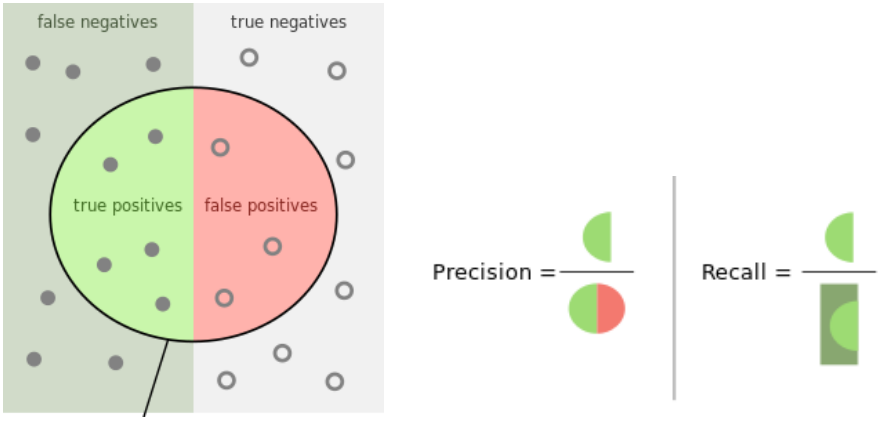


Figura 8.Gráfica con la terminología definida y ejemplo de la construcción de dos métricas. Fuente: https://en.wikipedia.org/wiki/Precision_and_recall Consultado: [11-07-2019]

En función de la terminología anterior podemos definir métricas para nuestro modelo. Algunas de estas métricas son *Precision* (1), *Recall* o *Sensitivity* (2), *Specifity* (3), *F1 score* o *dice* (4).

$$\frac{TP}{(TP + FP)} \quad (1)$$

$$\frac{TP}{(TP + FN)} \quad (2)$$

$$\frac{TN}{(TN + FP)} \quad (3)$$

$$\frac{2 * Recall * Precision}{(Recall + Precision)} \quad (4)$$

En aprendizaje automático, una **matriz de confusión** es un diseño de una tabla que nos permite visualizar los resultados de nuestro algoritmo. Las filas de nuestra matriz serán las clases reales, mientras que las columnas serán las clases predichas. En este tipo de tablas es muy fácil observar cuando una clase se está confundiendo con otra.

En la Figura 8 se puede ver un ejemplo de la construcción de dos métricas usadas habitualmente.

2.1.4 Regresión lineal y logística

El método de regresión lineal, conocido como el método de los mínimos cuadrados, consiste en sumar todas las distancias entre los puntos definidos por la recta y los puntos reales, siendo la mejor recta posible la que minimice esta suma de distancias.

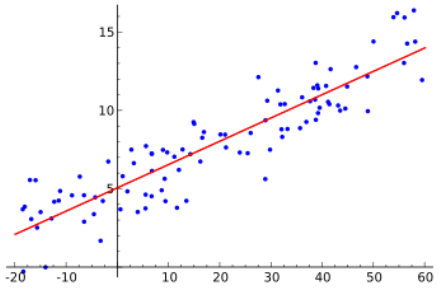


Figura 9. Gráfica de regresión lineal. Fuente: https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal Consultado: [11-07-2019]

Por su parte, en los modelos de regresión logística se tiene salidas discretas, con lo que se tendrá que usar cocientes de verosimilitud para comparar modelos. Si este coeficiente no se puede demostrar que un modelo resulta mejor que el otro, el más sencillo será el más adecuado.

2.1.5 Máquinas de vectores de soporte

Los SVM (máquinas de vectores de soporte) son un conjunto de algoritmos de aprendizaje supervisado. Cuando se entrena una máquina de vectores de soporte se busca el hiperplano que maximiza la distancia con los puntos más cercanos al propio hiperplano. Con lo cual SVM es un tipo de clasificador lineal.

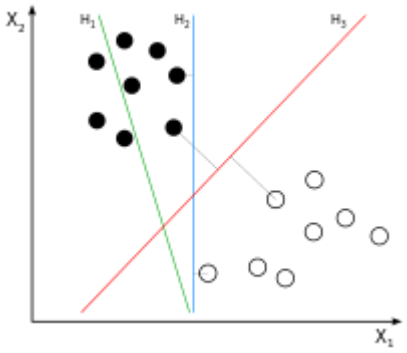


Figura 10. Ejemplo de clasificador SVM. Fuente: https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte Consultado: [11-07-2019]

2.1.6 Árbol de decisión

Los árboles de decisión (*Decision Trees*) son una técnica de aprendizaje automático usado para la clasificación o regresión. Consta de un nodo raíz, varios nodos rama y varios nodos hoja o terminales.

Estos árboles responden a condiciones concretas produciendo una única salida. Se pueden tener múltiples salidas en un árbol de decisión, pero estas son mutuamente excluyentes.

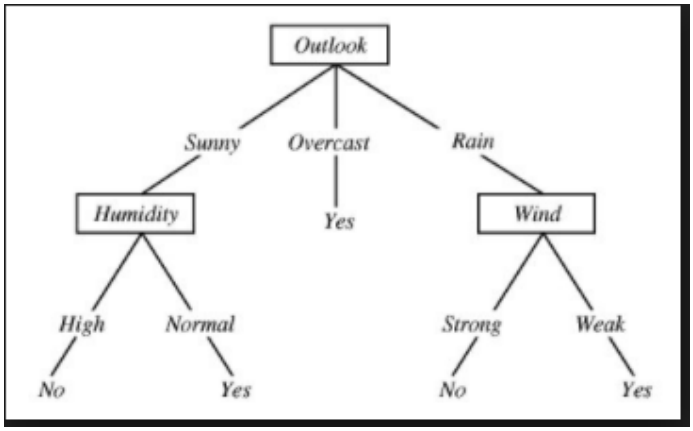


Figura 11. Ejemplo de árbol de decisión para tomar la decisión de jugar al golf. Fuente: http://science.slc.edu/~jmarshall/courses/2005/fall/cs151/lectures/decision-trees/ Consultado: [11-07-2019]

En la Figura 11 se ve un árbol de decisión que clasifica si es posible jugar al golf o no. En los rombos se decide uno u otro camino pero nunca ambos.

Los árboles de decisión son fáciles de entender y funcionan bien con grandes cantidades de datos, sin embargo eligen la mejor opción en cada nodo y esto no garantiza la mejor decisión en los últimos nodos, llamados nodos hoja. Como se verá más adelante los árboles de decisión son propensos al sobreajuste de sus muestras.

Un problema muy común en los árboles de decisión es el sobreajuste, un árbol de decisión muy elaborado suele dar lugar a sobreajuste. Teniendo múltiples nodos con el objetivo de distinguir entre unos pocos datos de entrenamiento muy parecidos. Recuérdese que el objetivo es la generalización. La **poda** de los árboles de decisión consiste en eliminar nodos intermedios con el fin de generalizar los resultados en los nodos finales.

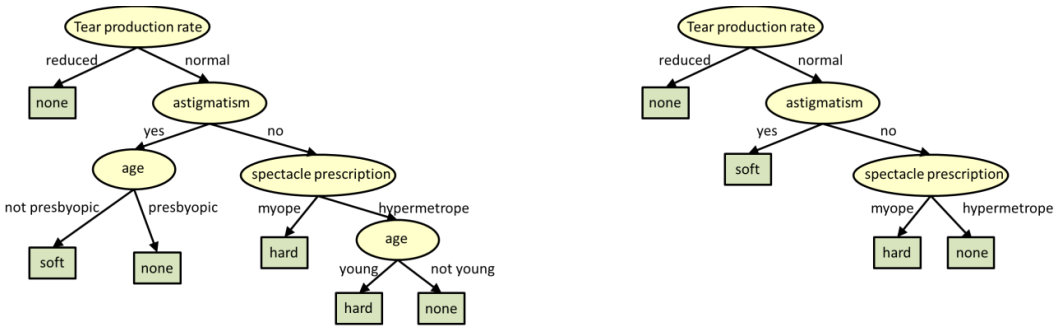


Figura 12. Ejemplo de poda de un árbol de decisión, en el que se eliminan nodos intermedios. Fuente: https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/ Consultado: [11-07-2019]

2.1.7 Bosques aleatorios

Los bosques aleatorios (*Random Forest*) son agrupaciones de árboles de decisión con idénticas salidas pero con entradas diferentes (diferentes características en diferentes árboles de decisión), esto evita árboles muy complejos (tienden a memorizar datos de entrenamiento). El resultado final se elige en base a diferentes métodos, el método más trivial es elegir la salida mayoritaria a lo largo de los árboles de decisión. Sin embargo es posible obtener la media de las salidas para un problema de regresión lineal (valor de una casa en Nueva York).

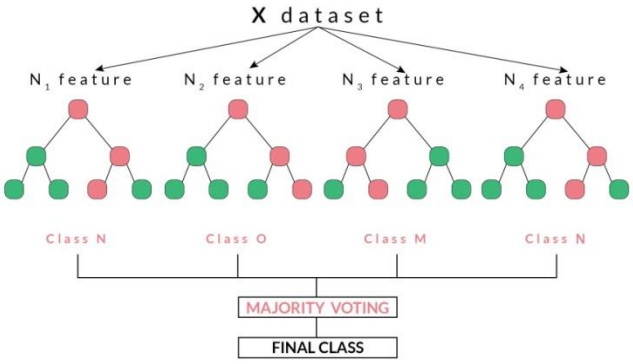


Figura 13. Ejemplo de bosque profundo, en el cual se tiene 4 árboles de decisión con idénticas salidas y se decide por la salida más repetida. Fuente: https://blog.quantinsti.com/random-forest-algorithm-in-python/ Consultado: [11-07-2019]

2.1.8 Redes neuronales artificiales

Las redes neuronales artificiales son una técnica de aprendizaje automático, imita la red neuronal biológica y permite que a través de unas determinadas entradas tengamos una salida correspondiente.

El **perceptrón** es la unidad básica de una red neuronal. Varios perceptrones unidos forman una red neuronal. Cada perceptrón dispone de varias entradas y una salida única. La salida corresponde con un sumatorio de las entradas ponderadas más el sesgo de la neurona (multiplicadas por un respectivo peso) aplicándolo a una **función de activación**. El sesgo de la neurona se asemeja a la facilidad de disparo de una neurona biológica. La variación de estos pesos para obtener la salida deseada constituye el **proceso de aprendizaje**.

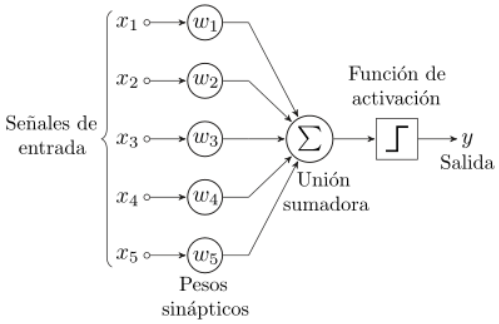


Figura 14. Representación gráfica del Perceptrón. Fuente: https://es.wikipedia.org/wiki/Perceptr%C3%B3n Consultado: [11-07-2019]

Otra característica importante de las redes neuronales es su arquitectura. Cuanto más compleja es la arquitectura de una red requiere un mayor coste computacional y

conlleva un nivel más alto de abstracción. No existe una arquitectura mejor que otra como norma general, depende del problema que se quiera resolver.

La arquitectura está definida por el número de capas, número de neuronas por capa sus conexiones. En una red totalmente conectada todas las neuronas de la capa anterior tiene una conexión para cada neurona de la siguiente capa.

- **Capa de entrada:** es una capa única en la red y existe una neurona para cada característica de entrada.
- **Capas ocultas**: no tienen un número de capas y neuronas definido, se utiliza para procesar los datos y dar una salida deseada. En aprendizaje profundo existen una gran cantidad de redes ocultas, se hablará de aprendizaje profundo en el próximo capítulo.
- **Capa de salida:** se utiliza para representar la salida, se tienen tantas neuronas como clases se tengan a la salida.

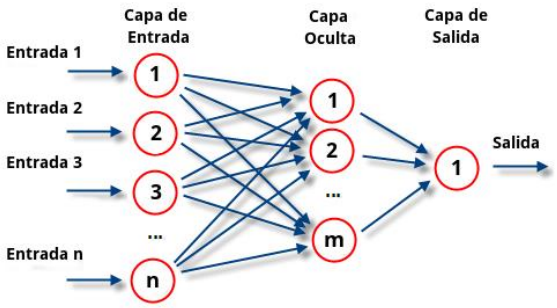


Figura 15. Representación de las capas a lo largo de la red neuronal. Fuente: https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa Consultado: [11-07-2019]

2.2 Aprendizaje profundo

Hasta aproximadamente 2012, las mejoras en la computación eran debidas a las CPUs. La capacidad de computación necesaria para aprendizaje profundo ha sido gracias a sistemas de procesamiento en paralelo (aceleradores **GPU**). Estos resultan ser más eficientes que las CPU para determinados algoritmos.

Originalmente, algunas empresas dedicadas a supercomputación como NVIDIA desarrollaron chip paralelos dedicados a las tarjetas gráficas para incorporarlos en los videojuegos recientes, ya que estos necesitaban gran cálculo matemático sobre matrices.

La investigación ha avanzado mucho gracias al aumento exponencial de la supercomputación, los investigadores han recuperado viejas ideas y las han puesto a prueba.

Otro factor a tener en cuenta es el fenómeno del ***Big Data***, en la era en la que se encuentra todo registrado, está al alcance de la mano conseguir gran cantidad de datos

para entrenar una red neuronal. A su vez podemos encontrar gran cantidad de modelos pre entrenados y completamente accesibles.

2.2.1 Parámetros e hiperparámetros.

Los parámetros se configuración de manera automática a través del entrenamiento de la red mientras que los hiperparámetros son totalmente configurables por el diseñador de la red. En aprendizaje profundo es extremadamente compleja la elección de estos hiperparámetros. Existen hiperparametros tanto a nivel de estructura de la red como a nivel de algoritmo de aprendizaje. Algunos de los hiperparámetros más importantes son:

- **Épocas (*epochs*)**: Indica el número de veces que se pasa los datos de entrenamiento por la red neuronal para ajustar nuestros parámetros.
- ***Batch size*:** los datos de entrenamiento se pueden dividir en mini lotes, los mini lotes definen cada cuántos datos se actualiza el gradiente para disminuir la pérdida.
- ***Learning rate***: este hiperparámetro cuantifica el paso que damos a través de la función de pérdida en cada mini lote. Este valor mide cuánto varían los parámetros en cada actualización. Un valor elevado indica saltos grandes, es decir, el aprendizaje será rápido, sin embargo, se puede saltar el mínimo de la función de pérdida o incluso la red podría llegar a divergir. Si el *learning rate* es bajo se tendrá más posibilidades de encontrar el mínimo pero el aprendizaje será lento. Existe la posibilidad de modificar este hiperpárametro en función de los resultados obtenidos durante el entrenamiento.

2.2.2 Proceso de aprendizaje en una red neuronal.

El aprendizaje en una red es un proceso iterativo en el cual se puede distinguir tres fases importantes en cada una de las repeticiones del proceso:

- ***Forward propagation***: es la primera fase del entrenamiento y consiste en introducir un ejemplo de entrenamiento a la red, tras cruzar por todos los componentes de la red se obtiene una predicción.

- **Función de *loss***: se realiza una estimación de cómo fue el resultado a partir del valor que se ha obtenido y la etiqueta del ejemplo de entrenamiento que denota el valor esperado. Existen diferentes funciones de pérdida.

- ***Backpropagation***: Se propaga la información de la función de pérdida a través de las neuronas de la capa oculta y se actualizan los parámetros utilizando algún método de optimización, como el conocido **descenso del gradiente (SGD – Stochastic Gradient Descent)**. Se calcula la derivada de la función de perdida y se trata de avanzar hacia el mínimo global. Cabe añadir que esto se actualiza cuando hayamos pasado todo el mini-lote por la red con el fin de ahorrar coste computacional.

Otros términos importantes en una red neuronal y su aprendizaje son los siguientes:

- **Función de activación**. Es la función que se aplica al sesgo más el sumatorio de las entradas multiplicadas por los pesos. Esta función simula el umbral de activación en una neurona biológica. Algunas de estas funciones más utilizadas son:

i. **ReLu**: a partir de 0 tiene un comportamiento lineal

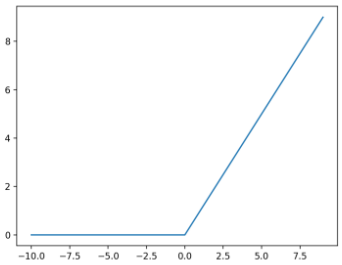


Figura 16. Gráfica de función de activación ReLu. Fuente: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ Consultado: [11-07-2019]

ii. **Función sigmoide** : se utiliza para reducir valores atípicos o extremos sin eliminarlos siendo su rango de entrada infinito y su salida entre 0 y 1

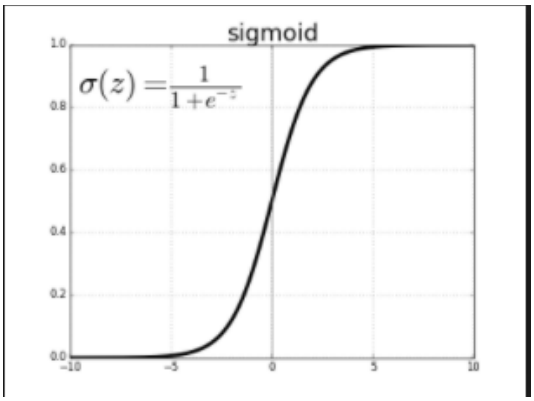


Figura 17. Gráfica de función de activación Sigmoide. Fuente: https://ml4a.github.io/ml4a/es/neural_networks/ Consultado: [11-07-2019]

iii. **Tangente hiperbólica**: similar a la función sigmoidea pero en este caso el rango de salida se encuentra entre -1 y 1

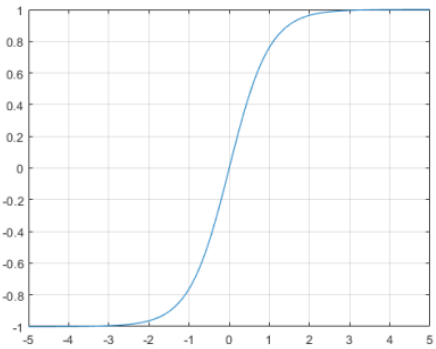


Figura 18. Gráfica de función Tangente hiperbólica. Fuente: http://www.sc.ehu.es/sbweb/fisica3/especial/trigonomerica/trigonometrica.html Consultado: [11-07-2019]

- **Optimizadores**: definen la forma en la que se provoca la variación de los parámetros.

- **Descenso de gradiente**: es un algoritmo de optimización encargado de realizar la derivada de la función de pérdida. Esto se hace capa a capa, con lo cual se obtiene el valor de las derivadas de la función de perdida en cada neurona y se actualiza el valor de los parámetros en sentido negativo de la derivada, ya que apunta hacia el mínimo la perdida. La cantidad que nos desplazamos está marcada por el *learning rate*.

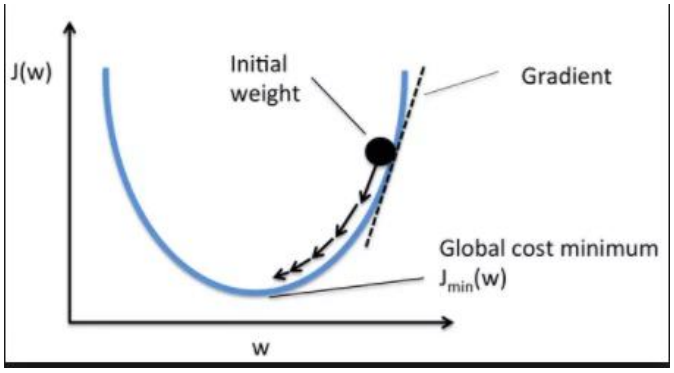


Figura 19. Se representa una variación a lo largo de la función de pérdida. Fuente: https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c Consultado: [11-07-2019]

El **descenso estocástico de gradiente (SGD)** marca la frecuencia con la que se disminuye la pérdida. En este caso se hace cada cierto número de *batch*, que contiene un conjunto de ejemplos de entrenamiento. Esto no es óptimo en cuanto a la disminución de la perdida pero ahorra tiempo de computación debido a que la GPU obtiene resultados muy rápidos con operaciones matriciales.

2.2.3 Redes neuronales convolucionales.

Estas redes son ampliamente utilizadas para tratar con imágenes, están formadas por una o más capas convolucionales. Estas capas tienen la ventaja de extraer características de las imágenes con lo cual se puede aprender características dentro de unas pequeñas ventanas de la imagen y no características globales de la imagen como lo que haría una red totalmente conectada. Las primeras capas de convolución extraen características generales de la imagen, las capas más avanzadas se centran en los detalles específicos de la imagen. Las imágenes están definidas por su altura, anchura y su profundidad (*depth*), la cual define los canales de la imagen siendo 1 para imágenes en blanco y negro, 3 para imágenes RGB y >3 para imágenes **multiespectrales**.

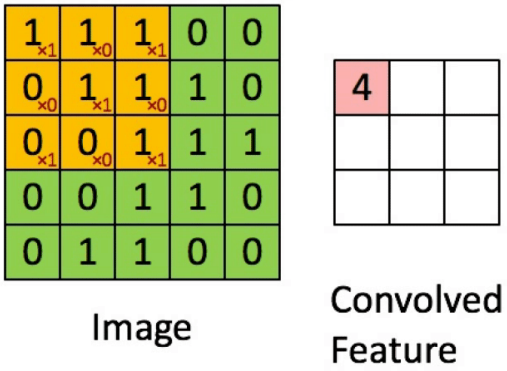


Figura 20. Operación de convolución 2D. Fuente: https://cleverpy.com/red-convolucional-pytorch/ Consultado: [11-07-2019]

Los componentes en la operación de convolución son los siguientes:

- ***Kernel size***: define el tamaño de la ventana para la operación de convolución, con lo cual el número de neuronas de la capa oculta de nuestra red. Si se tiene una imagen de 30x30 y usamos un *kernel* de 6x6 obtendremos una imagen de 25x25 a la salida de la operación de convolución y necesitaríamos un total de 25x25 neuronas (suponiendo ***stride***=1 y sin ***padding***) en esta segunda capa. Se reduce muchísimo el número de neuronas en comparación con una red totalmente conectada.

- **Número de filtros**: una cierta operación de convolución permite extraer una característica de la imagen, este parámetro define el número de características que se quieren extraer. Si se quiere obtener 24 características diferentes necesitaremos 24 filtros y la segunda capa de la red ascendería a 25x25x24 neuronas.

- ***Stride***: indica el desplazamiento en pixeles del *kernel* a través de la imagen.

- ***Padding***: esta técnica consiste en añadir 0 en los extremos de la imagen para conseguir que los filtros tengan el centro en todos los pixeles de la imagen. En este caso la longitud de la imagen de salida será igual a la entrada. No se modifica el tamaño de la imagen

  La entrada a la red neuronal convolucional es por definición los pixeles de la imagen (alto x ancho x *depth* sería el número de neuronas de la capa de entrada). Para cada filtro, la primera capa oculta tendrá una neurona por cada ventana que hayamos elegido para nuestra operación de convolución.

  Típicamente después de una capa de convolución se añade una capa de *pooling*, esta capa tiene por objetivo concentrar la información. Esta operación consiste en aplicar una ventana a la imagen y adquirir el valor máximo de los pixeles (el valor medio en algunos casos), se logra disminuir el tamaño de la imagen. Es necesario realizar esta operación para todos los filtros de convolución. En el ejemplo de la Figura 21, en una imagen de 24x24 realizando una operación de *pooling* con una ventana de 2x2 se obtendrá una imagen de 12x12.

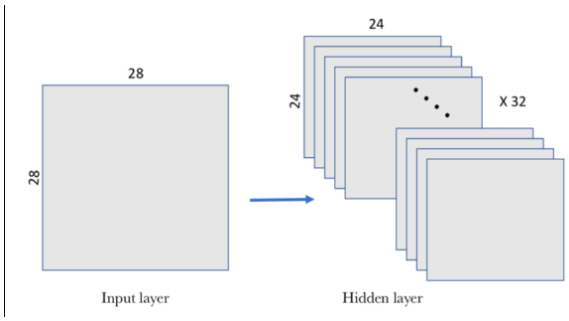


Figura 21. Gráfico correspondiente a la primera y segunda capa de una red convolucional. Hay 32 filtros diferentes. Fuente: https://torres.ai/deep-learning-inteligencia-artificial-keras/ Consultado: [11-07-2019]

2.2.4 Sobreajuste en Deep Learning

Algunas técnicas para disminuir el efecto del sobreajuste consisten en añadir capas adicionales a la red neuronal. [26]

- **Capas de *Dropout*:** tratan de prevenir complejas adaptaciones a los datos de entrenamiento mediante el "abandono" de ciertas neuronas (sus parámetros).

- **Capas de regularización L1 y L2:** consisten en añadir penalizaciones a la función de pérdida, es decir, se añade un término a la función de pérdida. Las capas de

regularización permiten disminuir la función de pérdida evitando la complejidad del modelo. Modelos complejos tienden al sobreajuste.

$$J = MSE + \alpha * C \qquad (5)$$

En la regularización L1 (*Lasso*), el parámetro C se define como el valor medio de los pesos del modelo. Esta generalización sirve para descartar características en la entrada de la red y de esa manera evitar el sobreajuste.

$$C = \frac{1}{N}\sum_{j=1}^{N} abs(Wi) \qquad (6)$$

La regularización L2 (*Ridge*), el parámetro C es la media al cuadrado de los pesos del modelo. Está regularización es efectiva cuando las características de entrada de nuestro modelo tenga correlación, con este método logramos disminuir el efecto de esta correlación.

$$C = \frac{1}{2N}\sum_{j=1}^{N} Wi^2 \qquad (7)$$

Es posible combinar estas dos técnicas de regularización en una sola, siendo r la importancia asignada a cada uno de los dos métodos.

$$C = r * Lasso + (1 - r) * Ridge \qquad (8)$$

Otra manera de disminuir el *overfitting* o *underfitting* es el aumento de datos (*data augmentation*). Este proceso consiste en coger los datos existentes y modificarlos con el fin de aumentar la variedad (evitando el sesgo) durante el entrenamiento. En

clasificación de imágenes existen varias técnicas como puede ser la rotación o trasposición de las imágenes, añadir ruido aleatorio o variaciones en el color. Estas técnicas mejoras la capacitación de la red neuronal.

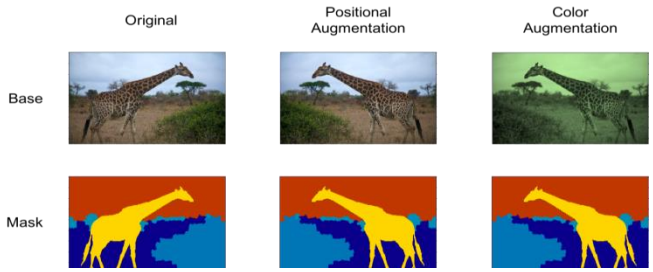


Figura 22. Ejemplo de Data Augmentation. Fuente: https://mxnet.incubator.apache.org/versions/master/tutorials/python/data_augmentation_with_masks.html Consultado: [11-07-2019]

El aumento de datos consigue que todas las clases tengan las mismas muestras, es decir que los datos estén **balanceados.** Las clases sesgadas es uno de los principales motivos para no lograr la generalización de los datos. De la misma manera, los datos de validación tienen que estar balanceados para lograr obtener resultados "reales".

Capítulo 3. Revisión del estado de conocimiento

3.1 Técnicas de clasificación de cultivos.

Las redes neuronales convolucionales son novedosas para clasificar imágenes remotas de cultivos. En las convoluciones 3D, el *kernel* 3D se utiliza para estructuras espaciales y multi temporales. Con esta técnica se puede entrenar y afinar los parámetros con los ciclos completos de los cultivos.

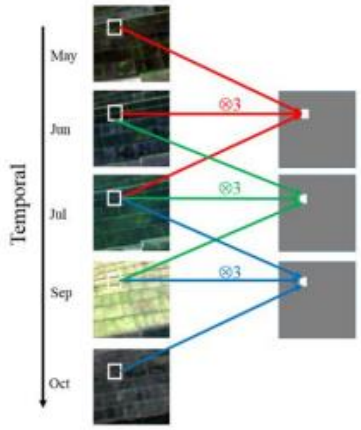


Figura 23. Ejemplo de Convolucion 3D en cultivos. [9]

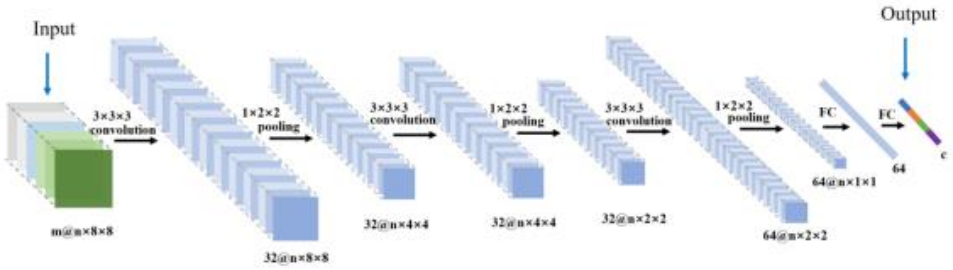


Figura 24. Red Convolucional·3D. [9]

Debido a la gran cantidad de imágenes satelitales capturadas durante ciclos de tiempo cada vez más pequeños, clasificar las imágenes de manera automática se ha convertido en una tecnología fundamental para la estimación del rendimiento, la evaluación del terreno y los correspondientes intereses económicos. Algunos métodos convencionales para la clasificación de cultivos son las máquinas de vectores de soporte (SVM), los

vecinos cercanos (KNN) y el estimador de máxima verosimilitud. Estas técnicas son válidas, pero podrían ser insuficientes bajo las circunstancias de la vegetación.

Actualmente, las organizaciones de sectores de la agricultura están haciendo costosos etiquetados manuales. Los métodos de la teledetección no necesitan de presencia física en el cultivo para su clasificación y se dedican a extraer características espaciales, espectrales y/o temporales etiquetando los distintos tipos de cultivo. Típicamente, las características espaciales y espectrales se extraen a través de índices de vegetación, que representan las características de la vegetación, el más utilizado es el índice de vegetación normalizado (NVDI). Estos índices son usados para la clasificación de cultivos. El índice de vegetación normalizado usa pocas bandas y puede tener dificultades para ofrecer un alto rendimiento en situaciones complicadas.

En aprendizaje profundo, las convoluciones 1D logran extraer características espectrales, las convoluciones 2D sobre imágenes logran extraer información espectral y espacial y las convoluciones 3D logran extraer información espectral, espacial y temporal. Cabe añadir que en los cultivos la información temporal es de vital importancia ya que ciertos cultivos son iguales en algunos ciclos temporales.

$$\mathbf{Y} = \boldsymbol{\sigma}\left(\sum_{\mathbf{n=0}}^{\mathbf{N}} \sum_{\mathbf{i=0}}^{\mathbf{M}} \sum_{\mathbf{j=0}}^{\mathbf{M}} \mathbf{Wij, n}(\mathbf{c}+\mathbf{i})(\mathbf{d}+\mathbf{j}), \mathbf{n} + \mathbf{b}\right) \quad (9)$$

Si en el caso de estudio se tiene que conseguir clasificar zonas invariantes en el tiempo (edificios, carreteras, caminos), es de vital importancia la obtención de estas características temporales a través de convoluciones 3D.

En algunas situaciones es difícil recolectar muestras del tipo de cultivo y es necesario un gran conocimiento específico para etiquetar imágenes satelitales, especialmente algunos cultivos como arroz, trigo, cebada o maíz. Es muy importante la calidad de las muestras y el preprocesamiento utilizado en las imágenes de entrenamiento.

Algunos algoritmos de clasificación como los árboles de decisión o los bosques aleatorios utilizan una clasificación pixel a pixel. Esto es una gran desventaja ya que se tiene en cuenta las variaciones espaciales. El objetivo es conseguir un extractor de características que se adapte automáticamente sin mucho coste computacional.

El aprendizaje profundo soluciona el problema de la extracción de características en distintas fases temporales. Es posible la utilización de redes recurrentes (RNN) para la obtención de características en distintas series temporales.

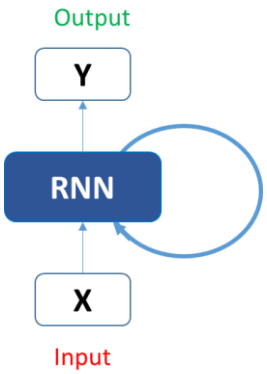


Figura 25. Ejemplo de red Recurrente (RNN). Fuente: Fundamentals of Deep Learning – Introduction to Recurrent Neural Network.

Con esto se pretende detectar patrones en los crecimientos de los cultivos.

3.2 Técnicas ITACyL en la clasificación de cultivos.

Castilla y León tiene 94.223 km^2 en términos de superficie. La dificultad de discriminar cultivos y la actualización plurianual de ellos hace inviable una clasificación detallada de cultivos. Para evitar errores de comisión hay que incluir todas las tipologías distintas en el entrenamiento y no limitarse a cultivos.

El Instituto Tecnológico Agrario de Castilla y León para realizar su clasificación usa imágenes satelitales Deimos-1, Landsat-8 y Sentinel-2. El objetivo principal es producir un mapa de cobertura de suelo que represente los cultivos, las áreas de vegetación natural y los cultivos permanentes. ITACyL distingue entre 122 clases de cobertura terrestre mediante un sistema de clasificación supervisado basado en aprendizaje automático con grandes cantidades de imágenes satelitales [27,28]

El aprendizaje automático consigue predecir resultados sin una programación explícita, lo consigue a partir de reconocer patrones en los datos de entrenamiento. En particular, el algoritmo de árbol de decisión es adecuado para clasificar la ocupación del suelo. Estos resultados pueden utilizarse como herramienta útil para controles administrativos y monitorización de recursos ambientales.

Además de las imágenes satelitales, se incluyen datos auxiliares para maximizar los resultados de la clasificación. Estos conjuntos de datos sirven de complemento para la clasificación:

- Mapa promedio de precipitación.
- Modelo de elevación digital (DEM)
- Índice de vegetación normalizado (NVDI)

Estos datos auxiliares son repixelados para coincidir con la resolución de las imágenes satelitales con la finalidad de introducir al árbol de decisión la información de manera conjunta.

Para una buena clasificación es necesario tener los distintos casos de entrenamiento repartidos en el terreno

Land cover (treshold)	Class description	nº	Map area (%)	Ref. area (%)	CI UA	CI PA	FS	Est. K
Arable Crops (>0.25% map area & >0.34% reference data)	Wheat	1	11.95	29.20	84.46±0.14	89.94±0.12	87.12	0.78
	Maize	4	1.29	3.20	99.01±0.12	94.7±0.26	96.81	0.99
	Barley	5	9.82	27.67	87.21±0.13	91.22±0.11	89.17	0.82
	Rye	6	0.71	2.94	83.31±0.67	39.44±0.6	53.54	0.83
	Oats	8	0.72	3.80	78.12±0.67	34.53±0.51	47.89	0.77
	Triticale	13	0.48	0.81	51.98±1.54	30.21±1.08	38.22	0.52
	Fallow	21	1.92	1.55	77.53±0.66	87.4±0.56	82.17	0.77
	Sunflower	33	3.87	8.96	92.42±0.19	92.22±0.19	92.32	0.92
	Rape	35	0.47	1.47	90.5±0.54	81.46±0.67	85.74	0.90
	Green Peas	40	0.38	1.48	86.34±0.64	75.34±0.74	80.46	0.86
	Vicia sativa	52	1.11	4.57	87.33±0.36	70.45±0.45	77.99	0.87
	Alfalfa	60	0.74	3.34	92.34±0.34	74.24±0.5	82.3	0.92
	Forage	61	0.63	0.34	25.98±0.94	73.25±1.59	38.36	0.26
	Raygrass	69	1.15	0.48	42.68±1.14	74.08±1.33	54.16	0.42
	Sugar beet	82	0.27	0.38	92.69±0.87	97.22±0.57	94.9	0.93
	Potatoes	94	0.26	0.37	80.27±1.3	90.27±1.03	84.98	0.80
Irrigated Arable Crops (>0.15% map & >0.10% ref)	Irrigated wheat	70	0.38	1.48	82.42±0.84	50.61±0.87	62.71	0.82
	Irrigated barley	71	0.16	0.55	61.14±1.59	46.39±1.42	52.75	0.61
	Irrigated alfalfa	72	0.44	0.72	70.27±1.02	86.44±0.85	77.52	0.70
	Irrigated sunflower	73	0.23	0.47	70.3±1.38	73.37±1.36	71.8	0.70
	Irrigated oats	76	0.17	0.12	37±1.94	81.34±2.32	50.86	0.37
Permanent crops (>0.14 & >0.08%)	Vineyard	100	0.96	1.14	98.28±0.26	96.86±0.34	97.56	0.98
	Olive groves	101	0.14	0.09	99.1±0.67	99.26±0.62	99.18	0.99
Forest Areas with different canopy cover fraction expressed in % (>0.1% map area & >0.01% reference data)	Pinus sylvestris (>70%)	120	2.50	0.09	91.36±1.99	91.36±1.99	91.36	0.91
	Pinus nigra (>70%)	124	0.83	0.01	63.44±7.63	94.99±4.23	76.07	0.63
	Pinus pinaster (>70%)	126	2.29	0.10	83.31±2.53	83.83±2.51	83.57	0.83
	Pinus pinaster (40-70%)	127	0.56	0.03	71.68±6.73	50.75±6.28	59.42	0.72
	Pinus pinaster (10-40%)	128	0.62	0.01	51.76±10.55	46.15±9.94	48.8	0.52
	Pinus radiata (>70%)	134	0.15	0.03	96.82±2.03	99.38±0.92	98.09	0.97
	Quercus ilex (>70%)	143	1.39	0.02	52.48±6.38	69.62±6.76	59.85	0.52
	Quercus ilex (40-70%)	144	1.87	0.04	38.99±4.25	61.08±5.32	47.6	0.39
	Quercus ilex (10-40%)	145	2.37	0.04	42.64±4.45	59.67±5.23	49.74	0.43
	Quercus pyren. (40-70%)	186	0.92	0.02	52.85±9.27	43.7±8.37	47.84	0.53
	Quercus pyren. (>70%)	187	3.36	0.06	72.99±3.57	86.7±2.98	79.25	0.73
	Quercus pyren. (10-40%)	188	2.10	0.02	42.16±7.08	54.01±8.09	47.36	0.42
	Populus plantat. (>70%)	198	0.62	0.27	93.63±0.97	98.6±0.48	96.05	0.94
	Quercus rubber (>70%)	241	0.50	0.23	93.89±1.02	97.9±0.62	95.85	0.94
	Quercus rubber (40-70%)	242	0.10	0.03	85.59±6.89	39.33±6.5	53.89	0.86
	Castanea sativa (>70%)	243	0.45	0.17	93.83±1.2	97.64±0.77	95.7	0.94
	Castanea sativa (40-70%)	244	0.12	0.02	84.2±7.27	46.31±7.37	59.76	0.84
Other (>0.5% & >0.05%)	Rocky areas	9	0.79	0.06	98.35±1.11	98.11±1.18	98.23	0.98
	Artificial and urban areas	3	1.31	0.09	95.97±1.33	99.99±0.08	97.94	0.96
	Water cover	255	0.67	1.50	99.86±0.07	98.74±0.19	99.3	1.00
Overall accuracy								83.94
Kappa index								0.80

Figura 26. Resultados árbol de decisión. Se observan métricas como FScore o el índice Kappa.[28]

Capítulo 4. Desarrollo de redes neuronales

Durante este capítulo se describe el proceso seguido desde la obtención de los datos hasta que la red es entrenada a través de distintos enfoques en la división de los datos y en la arquitectura de la red neuronal.

4.1 Diseño Dataset

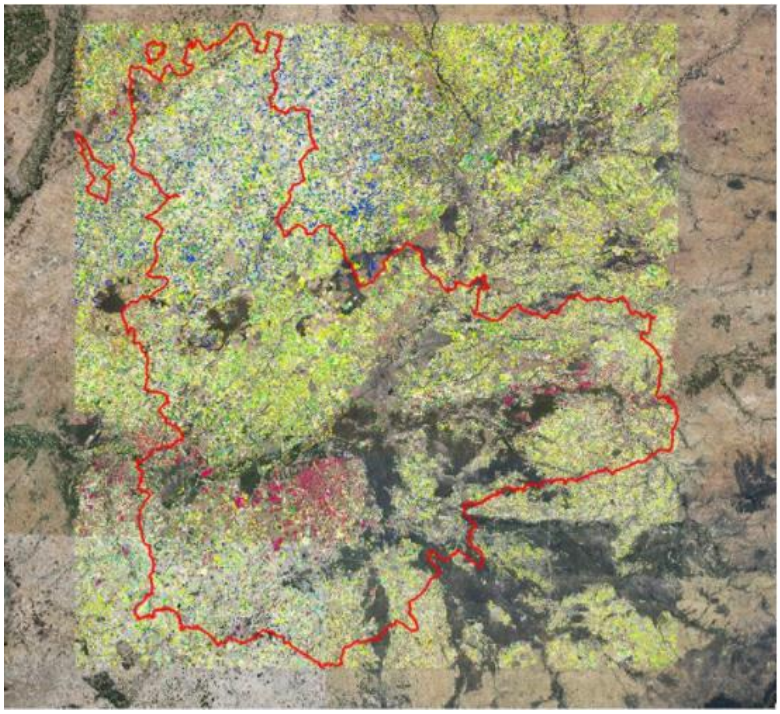


Figura 27. Límites terrestres del conjunto de datos. Fuente: ITACyL

El conjunto de datos proporcionado por el Instituto Tecnológico de Castilla y León consta de imágenes satelitales de Sentinel-2. Estas imágenes satelitales están recortadas a la zona de interés (Provincia de Valladolid) como observamos en la Figura 27. El *dataset* está formado por 24 imágenes multiespectrales (Figura 33) a lo largo de 10 meses (Dic-2017–Oct-2018). Todas las imágenes tienen tamaño 13580 x 12680 píxeles, siendo cada pixel 10 metros sobre el terreno. Están compuestas por 6 bandas de las 14 de Sentinel-2 (2, 4, 5, 8, 11,13). Las bandas que no tienen la resolución más alta se realiza re pixelado para el entrenamiento.
Un ejemplo de los datos sería:

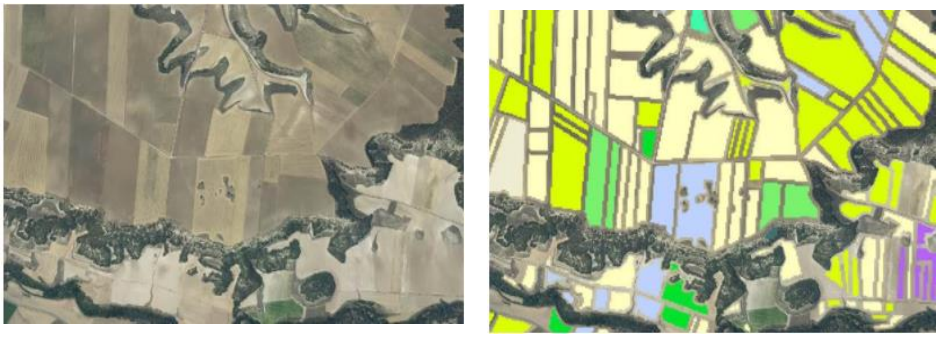


Figura 28. Imagen real recortada con su correspondiente etiqueta. Fuente: ITACyL

FECHA	nº
18/12/2017	1
17/01/2018	2
27/01/2018	3
21/02/2018	4
26/02/2018	5
17/04/2018	6
27/04/2018	7
17/05/2018	8
22/05/2018	9
16/06/2018	10
26/06/2018	11
06/07/2018	12
26/07/2018	13
31/07/2018	14
10/08/2018	15
15/08/2018	16
20/08/2018	17
25/08/2018	18
30/08/2018	19
09/09/2018	20
14/09/2018	21
19/09/2018	22
24/09/2018	23
29/09/2018	24

Figura 29. Fechas correspondientes a las imágenes satelitales. Fuente: ITACyL

La etiqueta es un ráster con los casos de entrenamiento y está formada por una imagen en escala de grises de tamaño 13580 x 12680 pixeles, en cada uno de los cuales hay un valor de 0 a 255 que representa una de las 29 clases. En la Figura 30 se puede observar la leyenda de colores de las diferentes clases junto con el porcentaje de cada clase y su descripción. Estos pixeles de entrenamiento se basa en declaraciones del PAC, donde los agricultores declaran que cultivos van a plantar sobre el terreno.

	Etiqueta	Clase	Porcentaje(%)
0	0	FONDO	0.674883
1	1	TRIGO	0.077656
2	3	URBANO-VIALES	0.000191
3	4	MAIZ	0.001845
4	5	CEBADA	0.109041
5	6	CENTENO	0.004318
6	8	AVENA	0.011618
7	13	TRIRICALE	0.000875
8	21	BARBECHO	0.019462
9	33	GIRASOL	0.030990
10	35	COLZA	0.001350
11	39	OTRAS LEGUMINOSAS GRANO	0.005798
12	40	GUISANTES	0.008049
13	52	VEZA	0.018634
14	60	ALFALFA	0.020676
15	61	FORRAJERAS	0.001672
16	82	REMOLACHA	0.001427
17	85	PASTIZAL	0.000089
18	94	PATATAS	0.001159
19	100	VIÑEDO	0.007365
20	101	OLIVAR	0.000378
21	102	ADORMIDERA	0.000058
22	104	FRUTALES DE CASCARA	0.000513
23	105	FRUTALES	0.000038
24	193	CONÍFERAS	0.001166
25	201	FRONDOSAS SIEMPREVERDE	0.000056
26	202	FRONDOSAS CADUCIFOLIAS	0.000401
27	203	HORTÍCOLAS	0.000104
28	204	MATORRAL	0.000057
29	255	LAMINA AGUA	0.000130

Figura 30. Descripción, valor de 8 bits asociado, porcentaje y el color asociado a cada clase. Fuente: ITACyL

ITACyL también proporcionó datos auxiliares que usan en su algoritmo de árbol de decisión para clasificar la cobertura de suelo. Estos datos son básicamente imágenes derivadas del Modelo Digital del terreno (MDT, *slope* y *aspect*), del LIDAR (importante para clasificar la parte forestal) y un mapa con precipitaciones medias históricas (Figura 31).

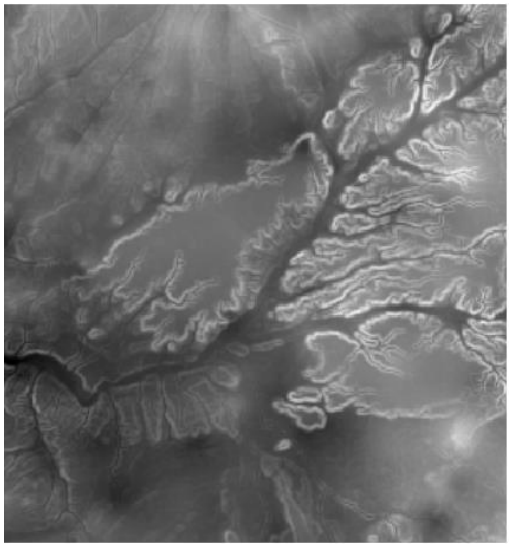


Figura 31. Mapa de precipitaciones medias históricas en la zona de interés. Fuente: ITACyL

Las redes neuronales para un buen aprendizaje es necesario que tengan una buena normalización [29] La normalización se realiza a menudo en la entrada. Para normalizar se tiene que restar cada pixel por la media (10) del canal y dividir por la desviación típica (11) del canal. Estos dos valores (media muestral y desviación típica muestral) serán a su vez la media de las 24 imágenes temporales que tenemos.

$$\overline{\mathbf{X}} = \frac{\mathbf{1}}{\mathbf{n}} * \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{X_i} \qquad (10)$$

$$\mathbf{S^2} = \frac{\mathbf{1}}{\mathbf{n}} * \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{X^2}_\mathbf{i} - \overline{\mathbf{X^2}} \qquad (11)$$

Canal	Media	Desviación Típica
1	1095.4885	863.49940
2	1760.0636	1026.15710
3	3067.5896	983.56800
4	1431.1594	1059.32420
5	2118.9287	1068.74500
6	1591.8949	899.83405

Figura 32. Media y desviación tipica de cada canal en los datos de entrenamiento

El siguiente paso es eliminar el 67% del terreno que se encuentra sin etiquetar, se incluye una clase al modelo que etiquete todos los pixeles no etiquetados con clase 0.

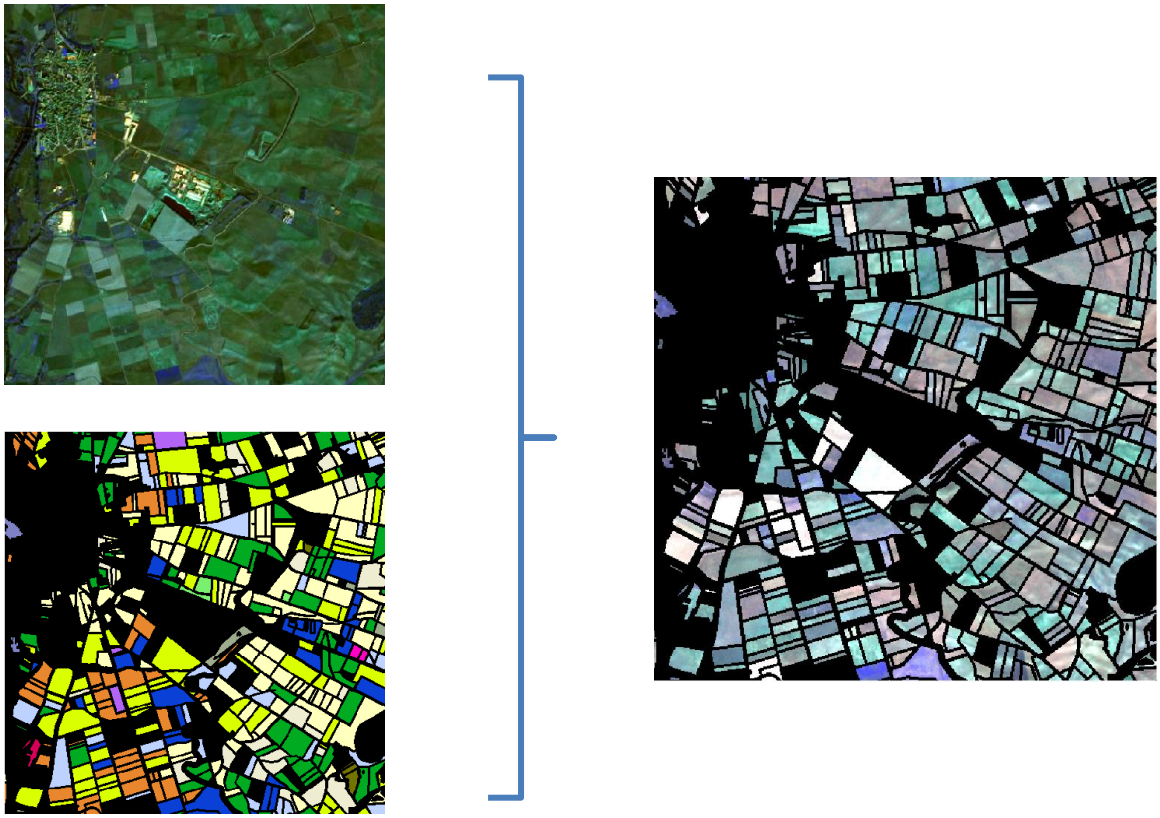


Figura 33. A partir de las imágenes satelitales y los pixeles etiquetados obtenemos las imágenes de entrenamiento.

Los datos mostrados a la red neuronal son los píxeles de las imágenes tras su procesamiento y después de realizar el reparto de datos de los apartados 4.3 y 4.4. Las imágenes con las cuales se calcula la pérdida para posteriormente hacer el *backpropagation* son las etiquetas recortadas a la zona de interés.

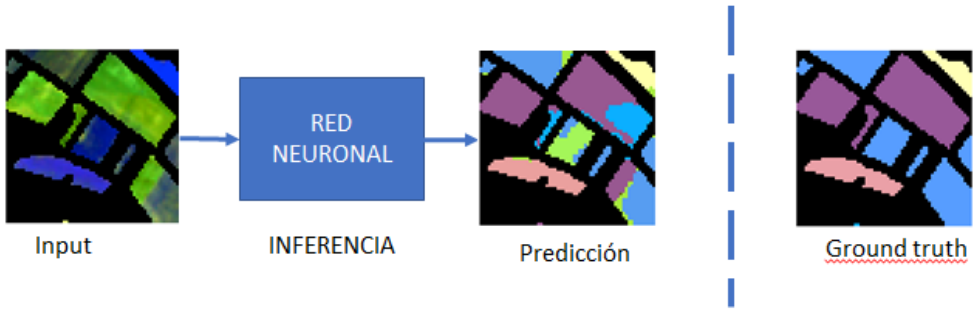


Figura 34. Esquema de inferencia. Ground truth se utiliza para el cálculo de las métricas

4.2 Autoencoder SegNet.

Los Autoencoder son redes neuronales que se utilizan para la segmentación de imágenes entrenando una arquitectura codificador-decodificador extremo a extremo.
Esta arquitectura genera una distribución de probabilidad para cada píxel siendo posible calcular la pérdida en cada uno de los píxeles. Si se obtiene la pérdida de todos los píxeles, se calcula la pérdida de la imagen. El objetivo es minimizar esta pérdida durante el entrenamiento.

La red en el codificador es idéntica a las capas convolucionales en VGG16 (Figura 35) eliminando la capas totalmente conectadas para conseguir un entrenamiento más cómodo. El decodificador consta de capas jerárquicas correspondientes a las capas de codificador.

VGG16 es una red neuronal convolucional. Esta red usa capas convolucional (kernel 3x3) apiladas. El número 16 denota las 16 capas de la red. Se realizan agrupaciones a través de una ventana 2x2 (Pooling). Las capas convolucionales van seguidas de tres capas totalmente conectadas. La salida original de la red es una capa de probabilidad a lo largo de las 1000 clases para las que fue entrenada originalmente.

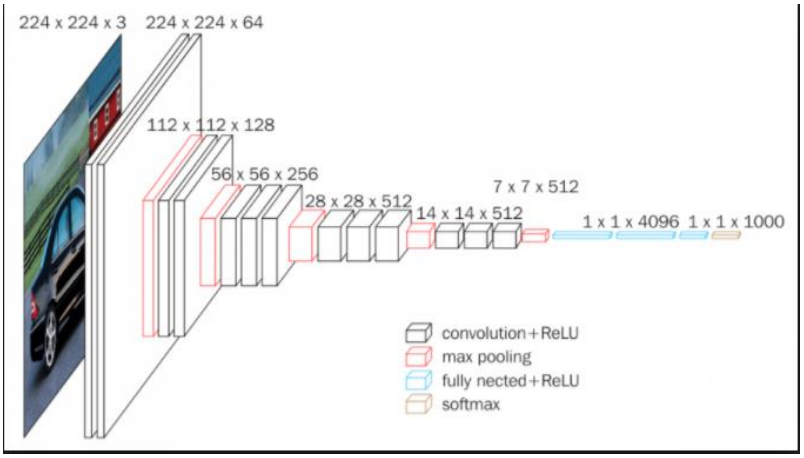


Figura 35. Modelo VGG16. [23]

Para el modelo de *deep learning* hemos usado siempre un Autoencoder SegNet (Figura 36) con segmentación extremo a extremo en el cual se ha añadido capas para evitar el sobreajuste (*Dropout* y Regularización L1).

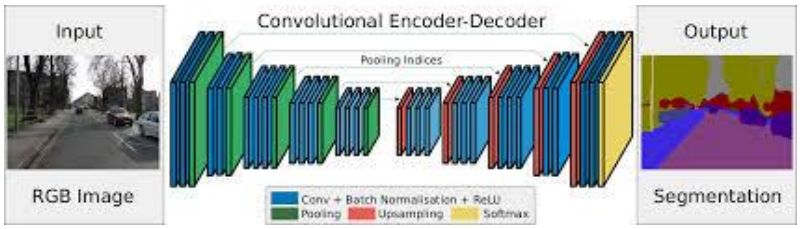


Figura 36. Autoencoder SegNet. [22]

4.3 Entrenamiento ventana deslizante

La primera estrategia usada para obtener el subconjunto de datos para la red neuronal es obtener imágenes con una ventana recorriendo la imagen inicial. La ventana tiene tamaño 512 x 512 píxeles, con lo que se obtiene 24 x 26 imágenes distintas (se descartan los extremos de la imagen original cuando la sub imagen es menor a 512), ya que dividiendo las dimensiones original entre la ventana se tienes píxeles sobrantes.
Estas imágenes se dividen en datos de entrenamiento (70%), validación (20%) y test (10%) de manera completamente aleatoria a lo largo de todo el terreno.

Observando una predicción con el 82% de píxeles acertados vemos que tiene muy buenos resultados con algunas clases pero con otras clases pequeñas y minoritarias, los resultados son malos.

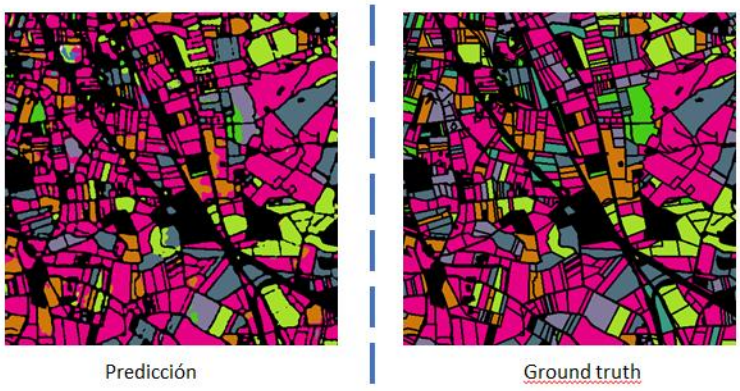


Figura 37. Ejemplo inferencia

4.4 Entrenamiento basado en regiones

Con la técnica anterior no se obtuvieron resultados muy satisfactorios, ya que las clases no estaban balanceadas y existen mucho más píxeles de una clase (no etiquetado) que de las otras.

Para evitar el problema de las clases sesgadas se redujo el número de clases a 19 (Figura 38) y se va con la ventana a buscar regiones de cada tipo de clase.

	Crop
0	FONDO
1	TRIGO
2	URBANO-VIALES
3	MAIZ
4	CEBADA
5	BARBECHO
6	GIRASOL
7	ALFALFA
8	REMOLACHA
9	PASTIZAL
10	VIÑEDO
11	OLIVAR
12	ADORMIDERA
13	FRUTALES
14	CONÍFERAS
15	FRONDOSAS SIEMPREVERDE
16	FRONDOSAS CADUCIFOLIAS
17	MATORRAL
18	LAMINA AGUA

Figura 38. 19 clases elegidas

	Value	Clase	Número de Puntos	Número de Contornos Cerrados
0	1	TRIGO	1636530	81653
1	3	URBANO-VIALES	192372	12
2	4	MAIZ	1880856	2064
3	5	CEBADA	1602814	111574
4	6	CENTENO	2447810	4854
5	8	AVENA	1887376	11777
6	13	TRITICALE	1808265	1198
7	21	BARBECHO	1594211	22614
8	33	GIRASOL	1516105	27001
9	35	COLZA	1870515	1121
10	39	OTRAS LEGUMINOSAS GRANO	2161766	5223
11	40	GUISANTES	1920972	7640
12	52	VEZA	1593828	17022
13	60	ALFALFA	1750452	17800
14	61	FORRAJERAS	2400802	2297
15	82	REMOLACHA	2255435	2032
16	85	PASTIZAL	253972	56
17	94	PATATAS	2137476	1807
18	100	VIÃ'EDO	2076633	8904
19	101	OLIVAR	505257	168
20	102	ADORMIDERA	134869	37
21	104	FRUTALES DE CASCARA	1151544	613
22	105	FRUTALES	103816	45
23	202	CONÍFERAS	401420	1016
24	204	FRONDOSAS SIEMPREVERDE	180600	153
25	203	FRONDOSAS CADUCIFOLIAS	330708	163
26	193	HORTÍCOLAS	1638387	2119
27	201	MATORRAL	310280	62
28	255	LAMINA AGUA	79540	21

,

Figura 39. Número de puntos y contornos cerrados para los datos de ITACyL

Para evitar tener sesgo en los datos de entrenamiento se busca regiones de las clases localizando su centro de masas. Después se aplica la ventana a las coordenadas de centro de masas y se obtienen 100 ventanas por cada clase (siendo su centro siempre perteneciente a la clase de interés). Estas ventanas tendrán tamaño 64 x 64 píxeles reduciendo el tamaño anterior para lograr maximizar la región central en la ventana y eliminar el porcentaje de clase no etiquetado.

Se repite esto para las distintas imágenes a lo largo de los 9 meses siendo siempre las ventanas coincidentes en la fase temporal. Estas 100 regiones se dividen en datos de entrenamiento (100 regiones) y datos de validación (10 regiones).

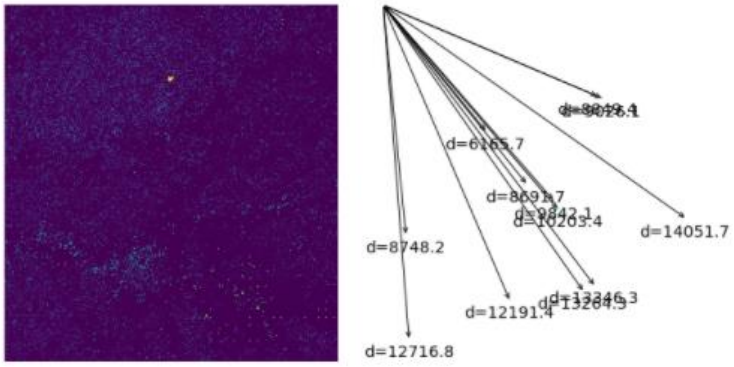


Figura 40. Distancia en píxeles de la esquina superior izquierda al centro de masas de la clase Urbano-viales.

Para una primera prueba se ha utilizado un modelo SegNet, con una función de pérdida *dice loss weigthed*, tamaño del *batch* 20 y *dropout* del 50% durante 50 épocas de entrenamiento.
La función de pérdida *dice loss* tiene la forma representada en la Figura 41.

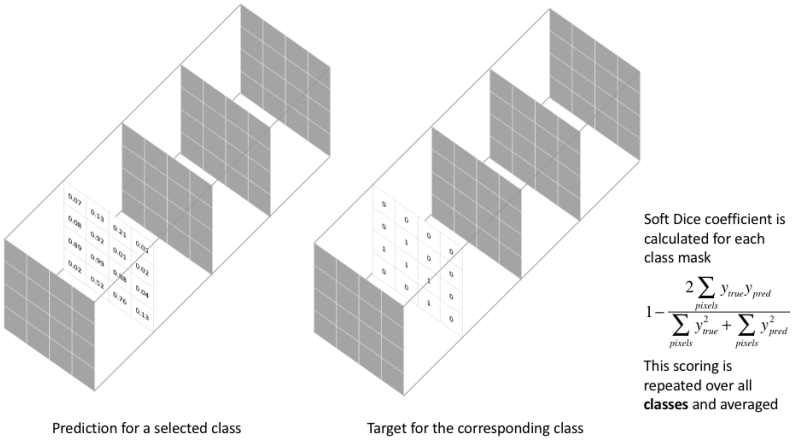


Figura 41. Representación de la expresión de soft dice loss. Fuente: https://www.jeremyjordan.me/semantic-segmentation/ [Consultado 11-07-2019]

El minuendo de la expresión de la Figura 41 corresponde con la métrica *dice* o *F1 score.*
A esta función se aplica un vector de pesos en el que se decide que peso asignar a cada clase. Para esta primera prueba, todos los pesos son iguales.

El enfoque de entrenamiento se ha realizado de dos maneras diferentes:

4.4.1 Entrenamiento anual.

Se entrena con imágenes de todos los meses y se intenta predecir regiones diferentes en esos mismos meses. Para la prueba se ha utilizado un modelo SegNet, con una función

de pérdida *dice loss weigthed*, tamaño del *batch* 20 y *dropout* del 50% durante 150 épocas de entrenamiento.

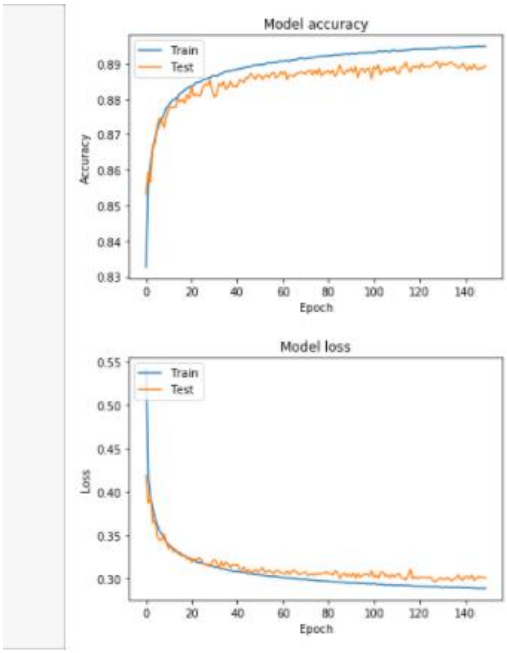


Figura 42. Recall (sensitivity) (2) y loss a lo largo de 150 épocas

Se ve que a partir de la época número 30 se produce sobreajuste (la curva de pérdida de entrenamiento se encuentra por debajo de la de validación) en el modelo, pero sigue mejorando a su vez en validación.

Los resultados obtenidos son:

	Crop	sensitivity	dice	X
0	FONDO	0.999795	0.999717	1.000000
1	TRIGO	0.446305	0.489787	0.475297
2	URBANO-VIALES	0.996730	0.976442	1.000000
3	MAIZ	0.425435	0.498846	0.445423
4	CEBADA	0.724977	0.605241	0.742722
5	BARBECHO	0.627133	0.546624	0.602381
6	GIRASOL	0.344868	0.413294	0.310802
7	ALFALFA	0.297293	0.398360	0.255376
8	REMOLACHA	0.319581	0.401878	0.380952
9	PASTIZAL	0.530675	0.537402	0.756410
10	VIÑEDO	0.637907	0.578334	0.621053
11	OLIVAR	0.520268	0.482913	0.315476
12	ADORMIDERA	0.208420	0.307368	0.192460
13	FRUTALES	0.692231	0.782538	0.596354
14	CONÍFERAS	0.564544	0.668540	0.831018
15	FRONDOSAS SIEMPREVERDE	0.797361	0.713323	0.819444
16	FRONDOSAS CADUCIFOLIAS	0.644838	0.710470	0.696078
17	MATORRAL	0.573483	0.600106	0.500000
18	LAMINA AGUA	0.774306	0.739889	0.756410

Figura 43. Sensitivity (2) dice (4) y métrica X

Se puede ver las metricas mas importantes para nuestro caso de estudio de las 19 clases. La metrica X define si en una región concreta, la clase dominante predicha (clase con mas píxeles sobre la región) se corresponde con la región real en la etiqueta. Está metrica solo mide regiones mayores de 10 pixeles.

La matriz de confusión del modelo:

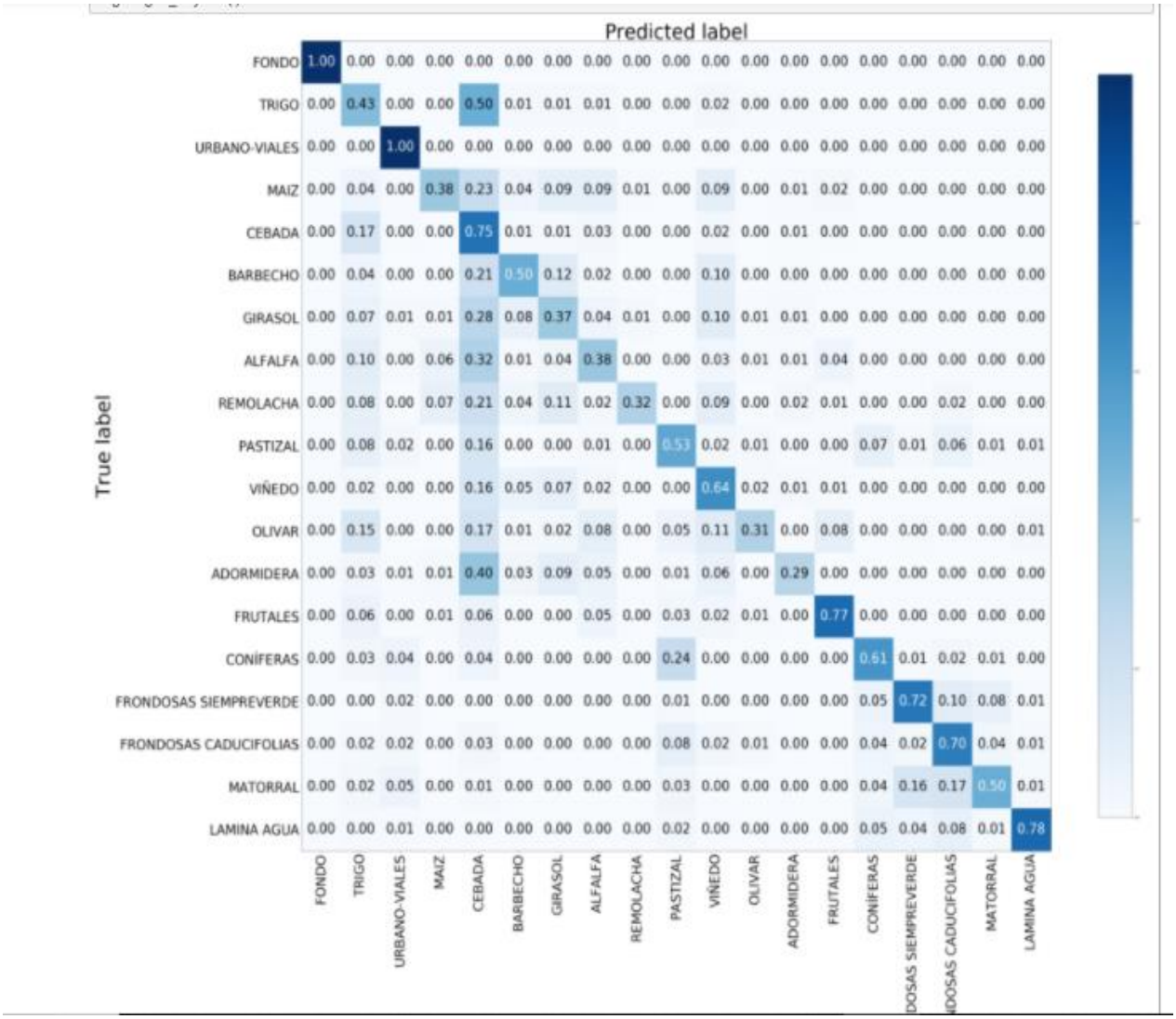


Figura 44. Matriz de confusión

Un ejemplo de predicción (Figura 45) con *sensitivity* 92% y con 10/12 regiones acertadas:

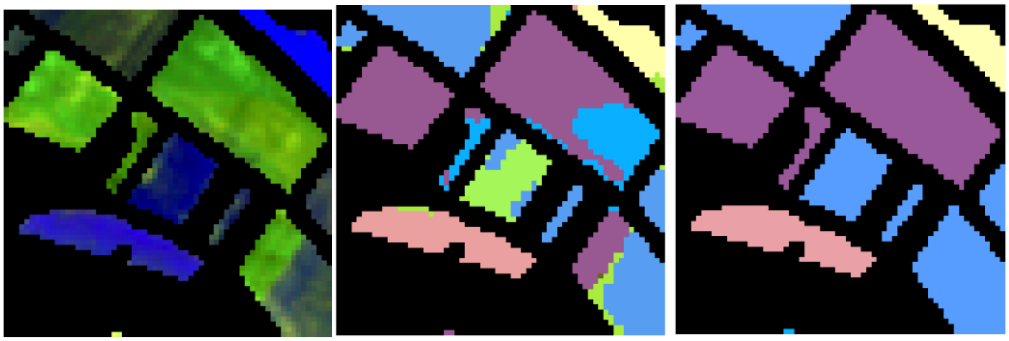


Figura 45. Imagen de test, predicción y etiqueta correspondiente

Se observa en una primera prueba que los resultados son buenos para las clases invariantes con el tiempo debido a su repetición a lo largo de las imágenes temporales. Existe una gran confusión entre muchas clases y las dos clases mayoritarias (trigo y cebada). Esto es debido a que aunque se esté seleccionando las clases de manera manual, en la ventana siguen cayendo regiones de otras clases y en concordancia con los porcentajes de regiones y píxeles es más probable que sean tanto trigo como cebada.

En las siguientes pruebas se ha variado los pesos de las clases trigo y cebada obteniendo las siguientes matrices de confusión. Los hiperpárametros son similares al modelo anterior:

- **Poco peso a la cebada:** Se observa como toda la cebada se clasifica como trigo

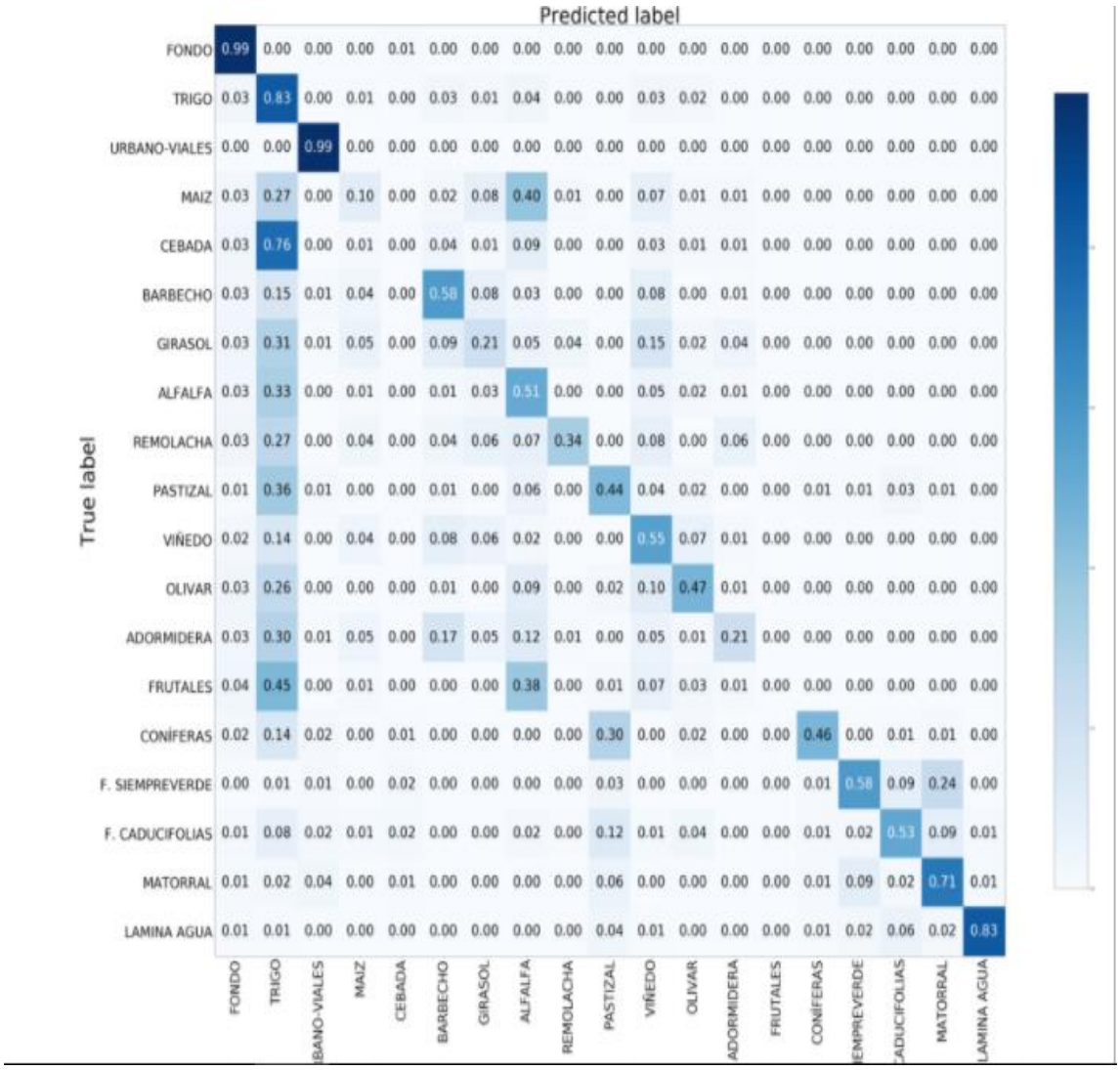


Figura 46. Matriz de confusión para modelo con poco peso para cebada.

- **Poco peso a la cebada y trigo:** Se observa se clasifica todo como cebada

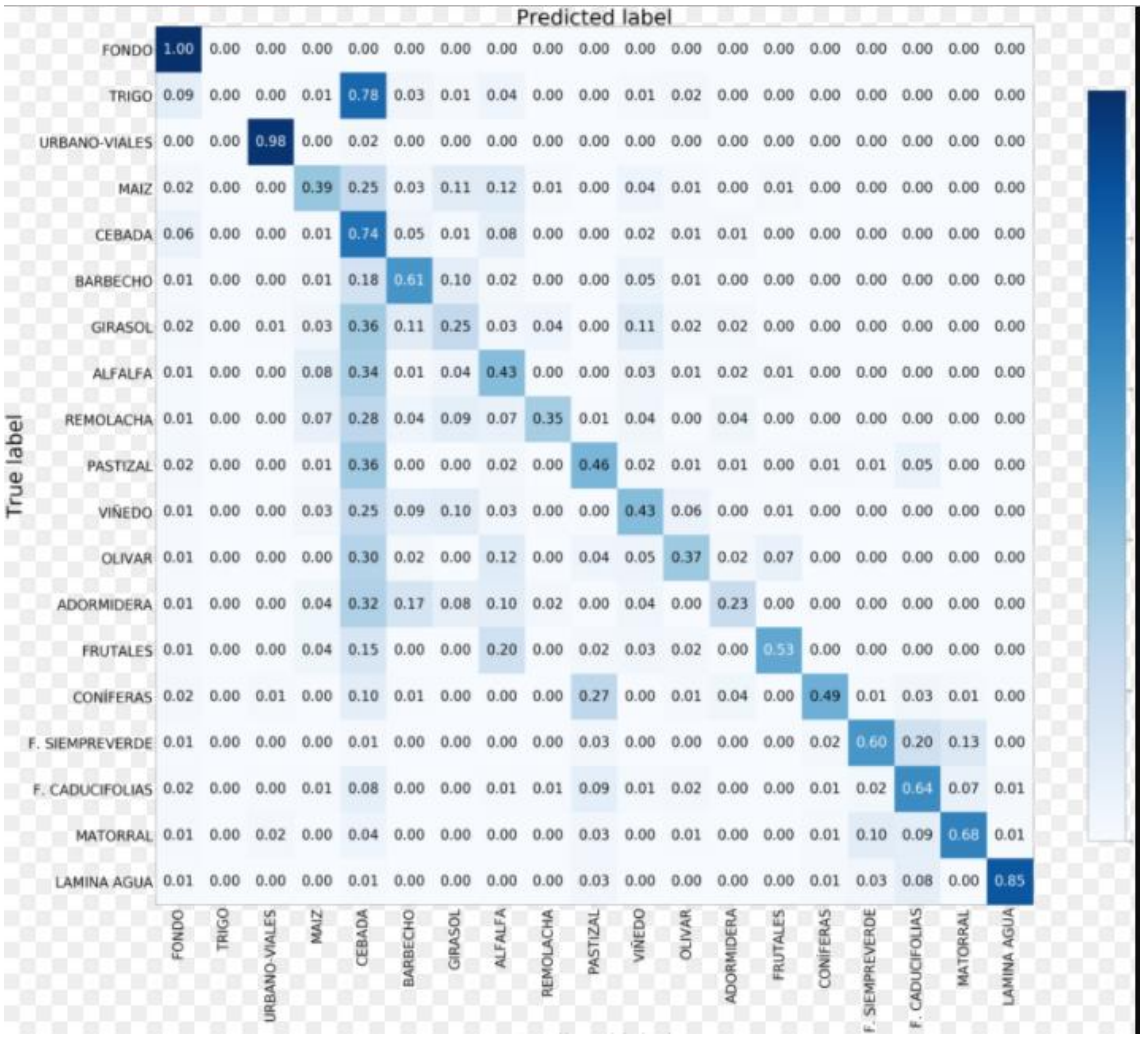


Figura 47. Matriz de confusión para modelo con poco peso para cebada y trigo.

	Crop	sensitivity	dice/f1	x
0	FONDO	0.996431	0.990986	1.000000
1	TRIGO	0.000000	0.000000	0.000000
2	URBANO-VIALES	0.976351	0.974949	0.993055
3	MAIZ	0.393013	0.441239	0.451291
4	CEBADA	0.743126	0.529493	0.800855
5	BARBECHO	0.605447	0.517744	0.573413
6	GIRASOL	0.251402	0.309994	0.249691
7	ALFALFA	0.430678	0.395086	0.394489
8	REMOLACHA	0.346338	0.351653	0.404762
9	PASTIZAL	0.462032	0.453122	0.810897
10	VIÑEDO	0.429069	0.460137	0.420614
11	OLIVAR	0.370268	0.344201	0.359127
12	ADORMIDERA	0.226147	0.279815	0.214286
13	FRUTALES	0.526827	0.646362	0.357639
14	CONÍFERAS	0.490506	0.638179	0.807870
15	F. SIEMPREVERDE	0.603264	0.666155	0.583333
16	F. CADUCIFOLIAS	0.644716	0.675589	0.776961
17	MATORRAL	0.683318	0.635062	0.666666
18	LAMINA AGUA	0.846983	0.879413	0.846154

Figura 48. Métricas para el modelo con poco peso para la cebada y trigo

4.4.2 Entrenamiento mensual

Cambiando el enfoque se escoge los datos de entrenamiento como todo lo visto hasta un mes y se a trata de predecir en los meses posteriores. Se distinguen dos casos de interés:

- **Entrenamiento hasta abril y predicción en mayo:**

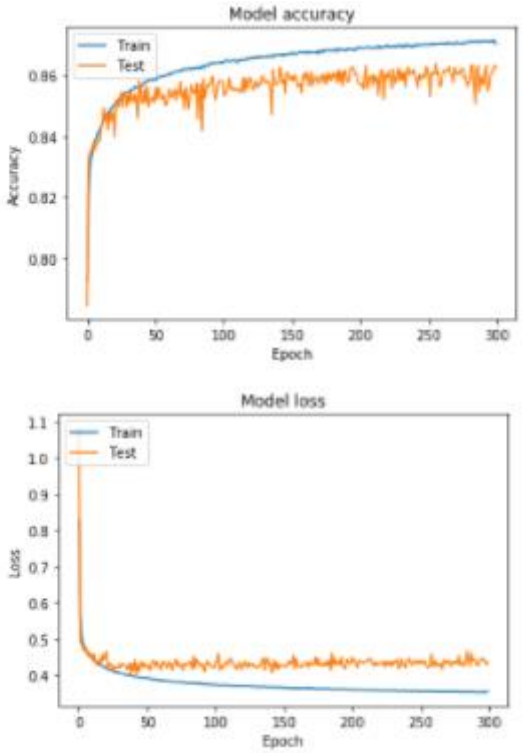


Figura 49. Curvas de recall (2) y loss en 300 épocas

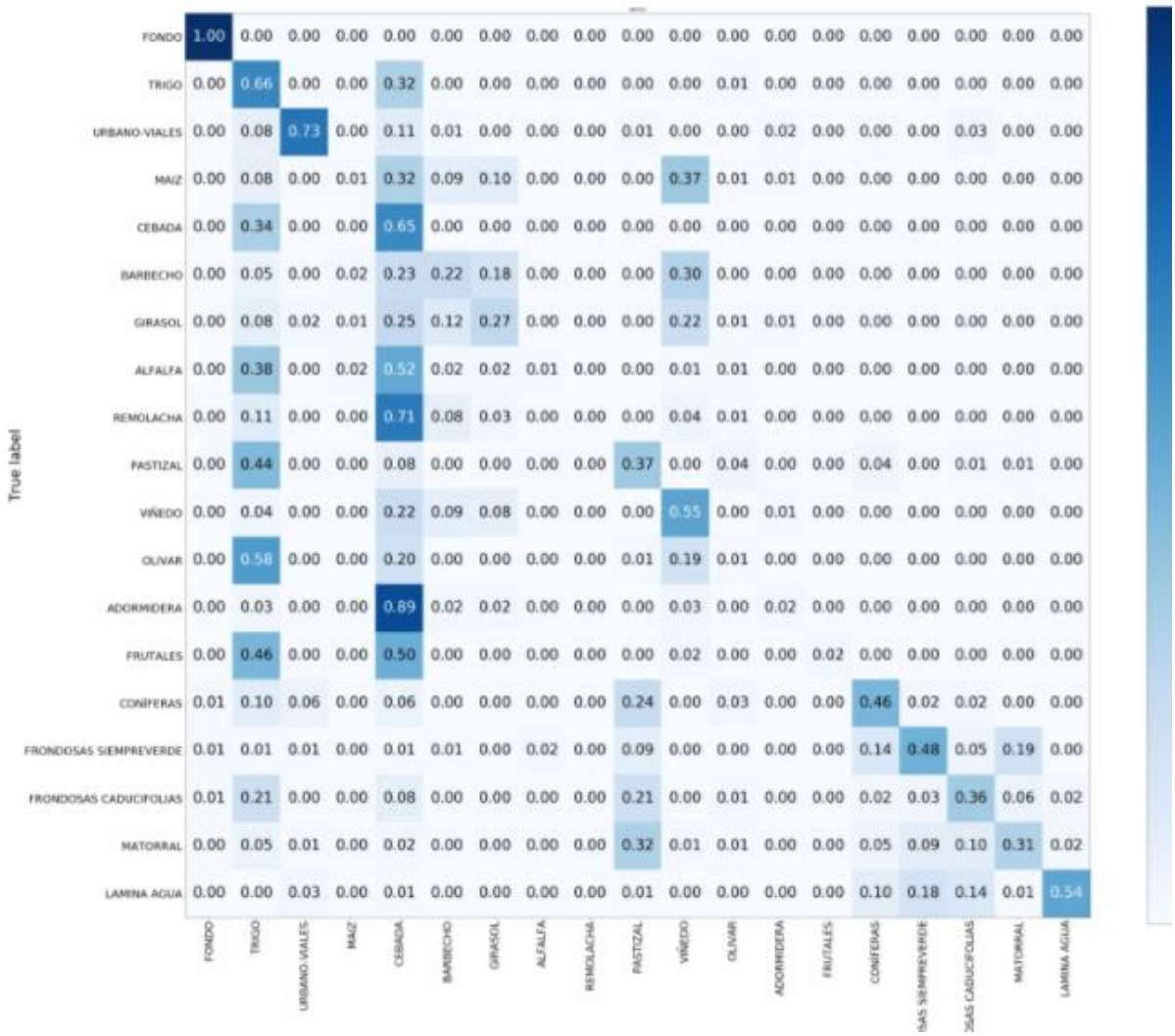


Figura 50. Matriz de confusión.

- **Entrenamiento hasta julio y predicción en agosto:**

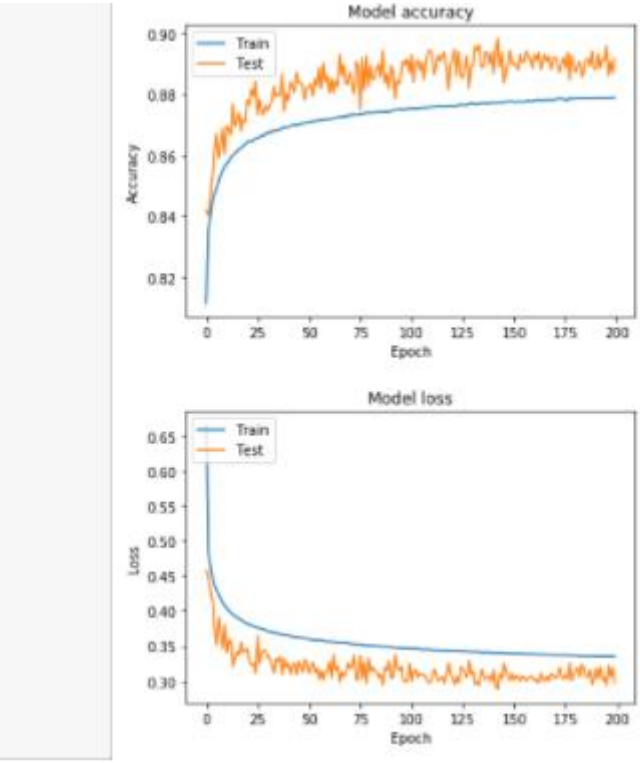


Figura 51. Curvas de recall (2) y loss en 300 épocas

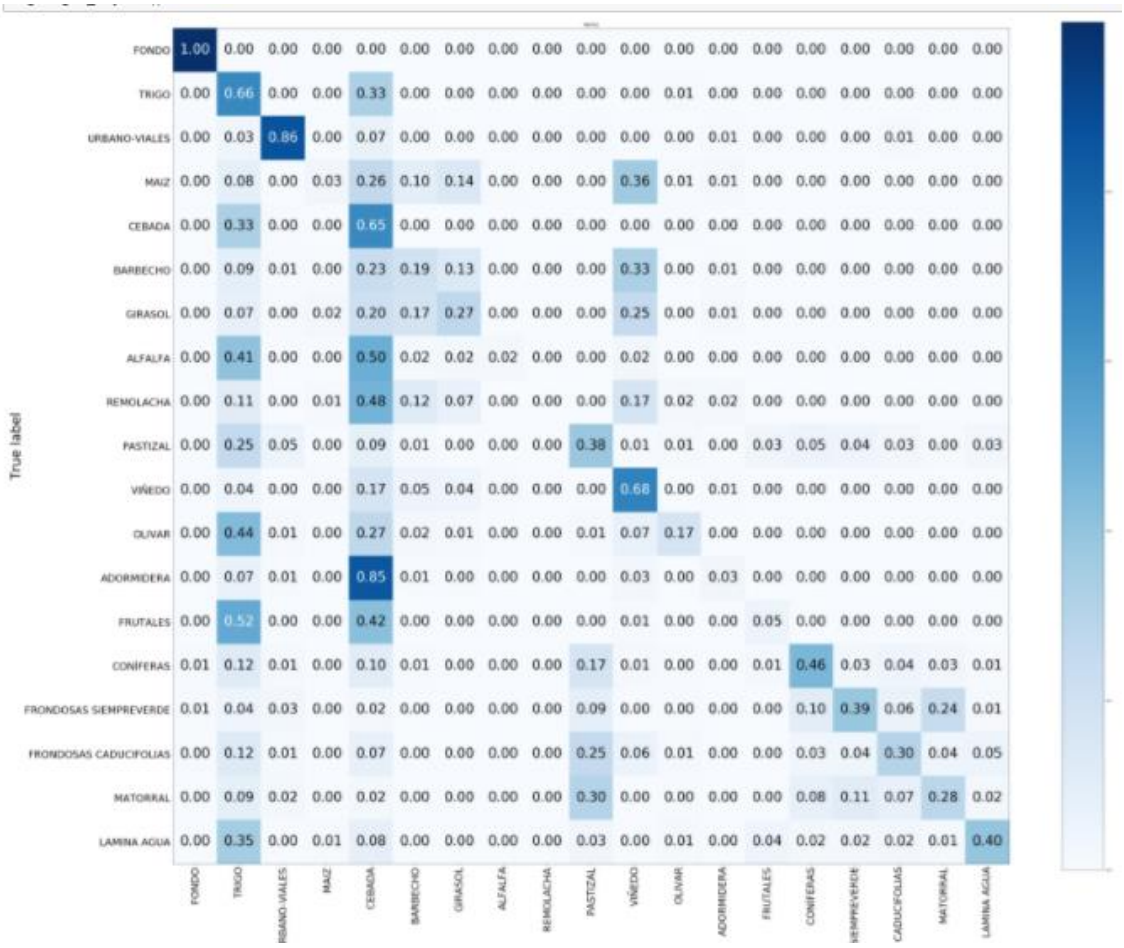


Figura 52. Matriz de confusión.

Capítulo 5. Conclusiones y líneas futuras

Desarrollar una red neuronal para automatizar la clasificación de cultivos es una tarea interesante, ya que el aprendizaje profundo ha crecido de manera exponencial en los últimos años adquiriendo mucha fama en cualquier problema de clasificación.

5.1 Conclusiones

El objetivo principal de este Trabajo Fin de Grado era conocer el funcionamiento de las redes neuronales para lograr la segmentación semántica de los distintos tipos de cultivos mediante imágenes satelitales multiespectrales.

Este TFG ha demostrado el potencial de las redes neuronales profundas para trabajar con imágenes multiespectrales a través de CNNs. Este modelo de capas convolucionales ha conseguido extraer características de los cultivos suficientemente generales para lograr predecir un número significativo de cultivos con un alto porcentaje de acierto.

Se ha comprobado que los distintos modelos de aprendizaje profundo son capaces de extraer características relevantes de las imágenes. Por el contrario, otros métodos tradicionales de clasificación de cultivos basados en técnicas de aprendizaje automático clásicas no tienen esta capacidad de extraer información espacial y temporal de las imágenes convolucionales.

Sin embargo, en cualquier problema de aprendizaje profundo es necesario tener los datos lo más representativos posibles, evitando el sesgo en las distintas clases a clasificar y evitando el porcentaje de la clase no etiquetada.

5.1 Líneas futuras

Tras las conclusiones obtenidas en el Trabajo Fin de Grado es posible afrontar algunas líneas futuras, con el fin de mejorar los resultados obtenidos.

Una opción es modificar el etiquetado de los datos, eliminando las zonas no etiquetadas en todo lo posible y seleccionando los cultivos con un mayor número de píxeles, con el objetivo de extraer más características durante el entrenamiento. Esto último supone a su vez disminuir el entrenamiento de las clases mayoritarias y la clase no etiquetada.

Otra posibilidad es utilizar otros modelos de redes neuronales para explotar la información temporal de los cultivos a lo largo de los meses. Las redes neuronales recurrentes son capaces de extraer información secuencial. En las redes convolucionales es posible aplicar convoluciones 3D, para explotar la información espacial y temporal

en los cultivos.

REFERENCIAS

[1] itacyl.es, "Fotografía Aérea2, 2018. [Online]. Available: http://www.itacyl.es/agro-y-geo-tecnologia/descarga-datos-geograficos/fotografia-aerea [Accessed: 10- Jul- 2019].

[2] itacyl.es, "Teledetección Espacial", 2018. [Online]. Available: http://www.itacyl.es/agro-y-geo-tecnologia/teledeteccion-y-sig/teledeteccion-espacial [Accessed: 10- Jul- 2019].

[3] R. M. Rustowicz, "Crop Classification with Multi-Temporal Satellite Imagery", Final Reports. Univ. Stanford, 2017.

[4] L. Zhong, L. Hu y H. Zhou, "Deep learning based multi-temporal crop classification", *Remote Sensing of Environment*, Vol. 221. pp 430-443 .Feb. 2019.

[5] A. Alcolea, J. Resano, "On-board Efficient Hyperspectral Image Classification Using Convolutional Neural Networks on Reconfigurable Hardware", Univ. Zaragoza, 2017.

[6] R. Setiono, W.K. Leow, "On mapping decision trees and neural networks" *Knowledge-Based Systems*, vol. 12, pp 95-99, Jun 1999.

[7] G, Qishuo, L. Samsung y J, Xiuping, "Hyperspectral Image Classification Using Convolutional Neural Networks and Multiple Feature Learning", *Remote Sensing*, vol. 10, Feb 2018.

[8] J.C. Goddard, J.M. Cornejo, F.M. Martínez, A.E. Martínez. H.L. Rufiner, y R.C Acevedo, "Redes Neuronales y Árboles de Decisión: Un Enfoque Híbrido", GMMCRA. Univ. Nac. Litoral, Nov 1995.

[9] S. Ji, C. Zhang, A. Xu , Y. Shi y Y. Duan "3D Convolutional Neural Networks for Crop Classification with Multi-Temporal Remote Sensing Images" *Remote Sensing*, vol 10, Ene 2018.

[10] N. Liberman, "Decision Trees and Random Forests" *Towards Data Science* ,Available*: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991* [Accessed: 20-06-2019].

[11] M Nielsen, *Neural Networks and Deep Learning*, 2013. [Online] .Available: http://neuralnetworksanddeeplearning.com/ [Accessed 20-06-2019].

[12] Jordi Torres, *DEEP LEARNING Introducción práctica con Keras* 2018.[Online]. Available: https://torres.ai/deep-learning-inteligencia-artificial-keras/ / [Accessed: 20-06-2019].

[13] Nvidia, "Cursos Nvidia". [Online]. Available: https://www.nvidia.es/object/deep-learning-courses-es.html [Accessed 20-06-2019].

[14] Amazon , "Amazon Machine Learning, Developer Guide*"* .[Online]. Available: https://docs.aws.amazon.com [Accessed 10-07-2019].

[15] Anaconda, (2019), Anaconda. [Online] . Available: https://www.anaconda.com/ [Accessed 10-07-2019].

[16] Python, (2019), Python. [Online] .Available: https://www.python.org/ [Accessed 20-06-2019].

[17] Tensorflow, (2019), Tensorflow. [Online]. Available: https://www.tensorflow.org/ [Accessed 20-6-2019].

[18] Keras, (2019), Keras. [Online]. Available: https://keras.io/ [Accessed 20-6-2019].

[19] Qgis, (2019), QGis. [Online]. Available: https://www.qgis.org/es/site/ [Accessed 10-07-2019].

[20] Jupyter, (2019), Jupyter. [Online] Available: http://jupyter.org/ [Accessed 20-06-2019].

[21] T. Hope, Y. S. Resheff y I. Lieder, *Learning Tensorflow*, O'Reilly, 2017.

[22] V. Badrinarayanan, A. Kendall, R. Cipolla, S. Member : "A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", vol. 3, arXiv:1511.0056, Oct 2016.

[23] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks For Large-scale Image Recognition*",* Vol. 6, arXiv:1409.1556v6, Apr 2015.

[24] W. Koehrsen, *"*Beyond Accuracy: Precision and Recall*", Toward Data Science* .[Online]. Available: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c [Accessed 20-06-2019].

[25] K. P. Shung ,*"*Accuracy, Precision, Recall or F1*?,"Toward Data Science* .[Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 [Accessed 20-06-2019].

[26] iartificial.net, "Regularización Lasso L1, Ridge L2 y ElasticNet". [Online]. Available: https://iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet [Accessed: 20-06-2019].

[27] V. del Blanco Medina y D. A. Nafría García ,“Mapa de cultivos y superficies naturales de Castilla y León”. [Online]. Available: http://ftp.itacyl.es/Atlas_Agroclimatico/03_ActividadAgraria/Agricultura/CultivosYSuperfNaturales/2015-07-06-Articulo_AET2015_Mapa_de_cultivos_y_superficies_naturales_%20de_%20Castilla_y_Le%C3%B3n.pdf [Accessed: 10-07-2019].

[28] V. Paredes Gómez, V. Del Blanco Medina, J. L. Bengoa y D.A. Nafría García, “Accuracy Assessment of a 122 Classes Land Cover Map Based On Sentinel-2, Sandsat 8 and Deimos-1 Images and Ancillary Data”, 5453-5456. 10.1109/IGARSS.2018.8519262.

[29] A. Douillard , 2018, *Normalization in Deep Learning.* [Online]. Available: https://arthurdouillard.com/post/normalization [Accessed 20-06-2019].

[30] CityScapes.com, (2019), “Dataset”. [Online]. Available: https://www.cityscapes-dataset.com/ [Accessed: 10-07-2019].

ANEXO:

Con el fin de aprender cómo funcionan las redes neuronales de segmentación semántica de imágenes se entrenó una red para la segmentación de 12 clases en imágenes de una ciudad. Este conjunto de datos es ampliamente conocido como CityScapes [30]. Este dataset contiene originalmente 30 clases, sin embargo, esta prueba se realizó únicamente con 12 clases. Las imágenes RGB tienen un tamaño 360 x 480 píxeles.

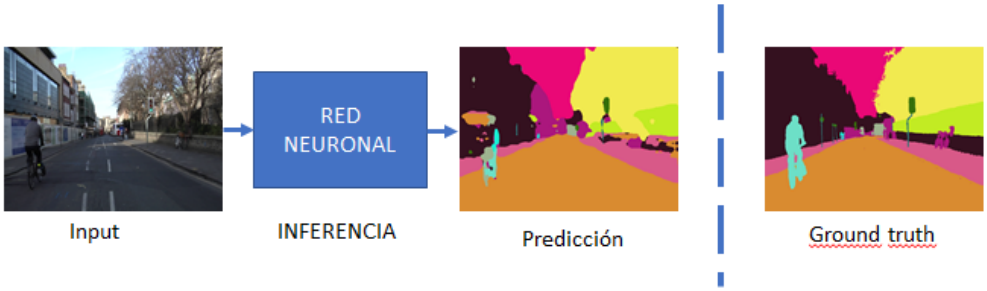


Figura 53. Inferencia para una imagen de CityScapes.

Tras entrenar 150 épocas, siendo 4 el *batch size*, un *dropout* del 50% y con la SegNet idéntica al apartado anterior.

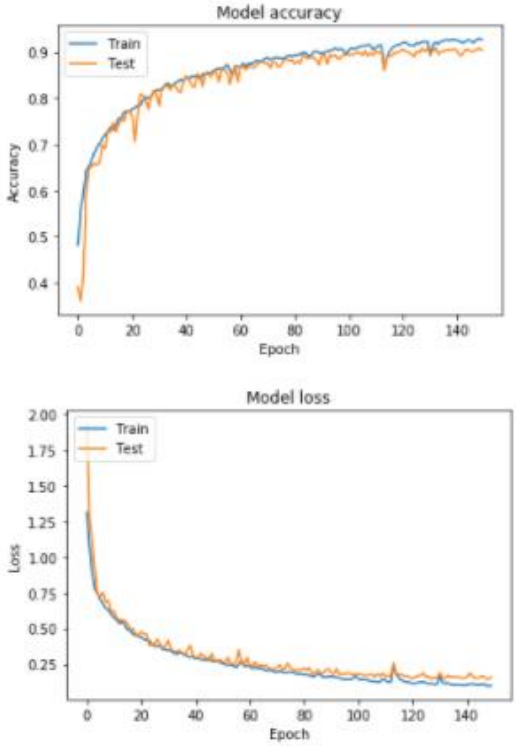
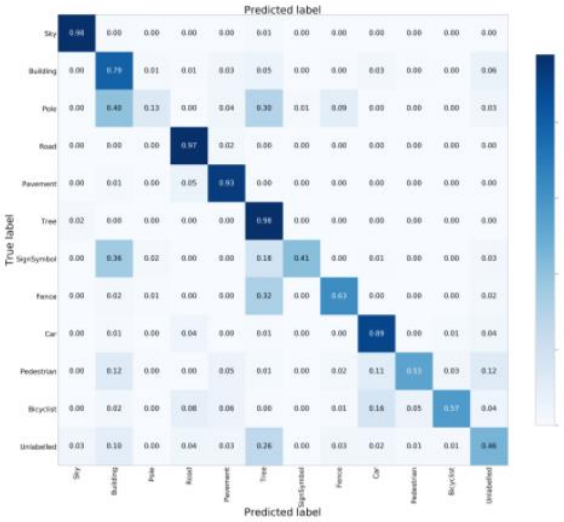


Figura 54. Gráfica de la evolución de Recall (2) y la loss a lo largo de las 150 épocas.



	Class	sensitivity	specificity	accuracy	dice
0	Sky	0.978552	0.978552	0.993355	0.964322
1	Building	0.791279	0.791279	0.935131	0.863647
2	Pole	0.127940	0.127940	0.991510	0.145376
3	Road	0.974054	0.974054	0.983415	0.971436
4	Pavement	0.930269	0.930269	0.976852	0.875129
5	Tree	0.977421	0.977421	0.963782	0.898141
6	SignSymbol	0.413411	0.413411	0.994022	0.552392
7	Fence	0.629376	0.629376	0.985784	0.731571
8	Car	0.890784	0.890784	0.985356	0.679905
9	Pedestrian	0.532973	0.532973	0.994087	0.539751
10	Bicyclist	0.571373	0.571373	0.989149	0.700172
11	Unlabelled	0.460738	0.460738	0.970283	0.345618

Figura 55. Matriz de confusión y distintas métricas (1,2,3,4)

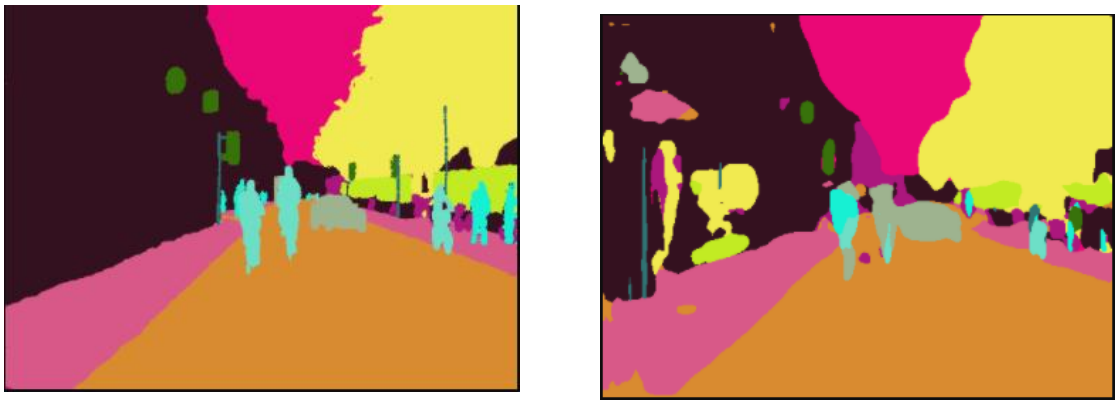


Figura 56. Ejemplo de predicción de una imagen de CityScapes.