eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Grado de Ingeniería Informática

▪ Trabajo Fin de Grado ▪

Computación

Detección de carreteras y caminos en imágenes por satélite

Autora

Igone Moráis Quílez

Noviembre 2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Grado de Ingeniería Informática

▪ Trabajo Fin de Grado ▪

Computación

Detección de carreteras y caminos en imágenes por satélite

Igone Morais Quilez

Noviembre 2020

Director

Manuel Graña Romay

Resumen

Hoy en día, el ámbito de la Inteligencia Artificial proporciona grandes avances y mejoras en las nuevas tecnologías, y en lo que a predicciones y búsqueda de características se presentan numerosas mejoras, generalmente, aplicadas al análisis de señales y procesado de imágenes. Para este proyecto, nos hemos basado en un Challenge de CodaLab que propone la detección automática de carreteras y redes de calles en imágenes por satélite, y ya que existen limitaciones de hardware y capacidad de computo, se ha pretendido llegar a este objetivo mediante un diseño de red neuronal convolucional y correspondientes algoritmos de Deep Learning.

Concretamente hemos aplicado una red profunda con topología en U que devuelve un mapa de clasificación de los pixeles de la imagen de entrada. Este proceso simultaneo de todos los pixeles reduce los costos computacionales de respuesta y produce mapas más consistentes que la clasificación de los pixeles individuales. Reportamos resultados obtenidos con los limitados recursos computacionales disponibles que confirman los resultados reportados en la literatura.

.

Agradecimientos

Tras horas y horas invertidas, que se traducen en estos años de carrera que ya llega a su final, quiero dedicar este breve apartado a todas las personas que me han ayudado durante este trayecto.

En primer lugar, me gustaría agradecer a una persona muy especial que me ha acompañado durante 23 años, por su apoyo y amor incondicional, que desde que era bien pequeña siempre me ha dado su primer apoyo para hacer todo aquello en lo que tenía un gran interés y entusiasmo, que fue ella quien me descubrió la música y es por ella que aprendí a tocar el piano y el clarinete, que siempre ha estado a nuestro lado cuidando de mí, y que siempre me ha animado a estudiar la carrera que me gustaba. A si que, con todo mi amor y admiración, quiero dedicar todos mis éxitos y victorias a mi amoñi Carmen. Con todo mi corazón te quiero, *Gracias.*

A mis padres, Fernando e Igone, por su apoyo incondicional y haberme dado la oportunidad de formarme en aquello en lo que me ha gustado, haberme traído café todos los días y cuidado de mí. A mi tía Carmen, quien me hizo un cuaderno para hacer ejercicios de matemáticas, quien me enseño que con creatividad e ingenio y buenas palabras se puede conseguir cualquier cosa que te propongas. A mi aitona por haberme dicho que estudiara inglés, y que hoy en día se ha convertido en mi tercer idioma y con el que algún día iré a Londres. *Os quiero.*

A todos mis profesores, y en especial a mi director Manuel Graña, excelente profesor y mejor persona, que me ha ayudado tanto y siempre ha estado ahí siempre que he tenido alguna duda o problema, y haber confiado en mí. Si no llega a ser por él, no hubiera encontrado un proyecto tan interesante. Espero seguir aprendiendo y descubriendo proyectos con él.

A todos mis amigos y compañeros, gracias por todos los buenos momentos que hemos pasado juntos.

Y finalmente, aunque la Ingenieria Informática nunca estuvo dentro de mis planes principales, ya que desde los 5 años siempre tuve muy claro que quería ser médico. Aunque, siempre me apasionó lo que más tarde conocería como ciberseguridad, y un gran experto en esta materia es José María Alonso Cebrián, más conocido como <<Chema>> Alonso, que aunque él no me conozca (ni yo a él personalmente) le tengo que agradecer el que sacara un hueco de su apretada agenda para venir a Donosti, más concretamente a la RITSI que en su momento tuvo lugar aquí para dar una conferencia, y que mientras explicaba la importancia de un antivirus y como 'pintar' un mapa de la red con la mera ayuda de un navegador, supe que quería hacer lo que él hacía.

Y cuando entré en la facultad de ingeniería informática y descubrí la rama de computación, supe que había tomado la decisión correcta.

Índice

Resumen I
Agradecimientos III
Índice V
Lista de Figuras, tablas y algoritmos VII

1. Introducción y Definiciones 8
Propósito 8
1.1. Definiciones y conceptos 9
1.1.1. Inteligencia Artificial y Machine Learning 9
1.1.2. Deep Learning 10
1.1.3. Redes Neuronales 10
1.1.4 Rede Neuronales Convolucionales 11
1.1.5 Perceptrón milticapa 13
1.1.6 Tensores 13
1.2. ¿Qué es un algoritmo de Deep Learning? 14
1.3. Problema 14
1.3.1. CodaLab Challenge 14
1.3.2 DeepGlobe 20198 15

2. Planificación y Gestión del Proyecto 21
2.1. Objetivos del proyecto 22
2.2. Tareas y Estimación de Tiempos 22
2.3. Análisis de Riesgos 23
2.4. Software: Matlab 24

3. Desarrollo del proyecto 26
3.1. Obtención de los datos 28
3.2. Casos curiosos a tener en cuenta 28

4. Diseño y Evaluación de modelos 32
4.1. Diseño del algoritmo y la red 34
4.2. Pruebas 38
4.2.1. Resultados 38
5. Conclusiones y Trabajo Futuro 45
5.1. Conclusiones 47
5.2. Otros ambitos de aplicación 47
5.3. Trabajos futuros 47

Bibliografía 49

Anexo A. Codigo 51

Lista de Figuras, Tablas y Algoritmos

FIGURAS

Figura 1. Esquema general de una red neuronal 12

Figura 2. Ejemplo de un percetrón o RNA 15

Figura 3. La obtención del IoU es una simple división del area de solapado entre los bounding boxes del area de unión 17

Figura 4. Muestra de cada tipo de imagen 30

Figura 5. Casos curiosos 31

Figura 6. Ejemplo donde la división del terreno se pude confundir con caminos 31

Figura 7. Resultados de un primer pre-proceso 32

Figura 8. Modelo de red CNN de tipo U que se ha seguido como modelo 36

Figura 9. Cada sección de la red-u mostrada de izquierda a derecha 39

Figura 10. Imágenes para el primer caso de prueba 40

Figura 11. Gráfica de resultados del primer train para el primer caso de prueba 41

Figura 12. Gráfica de resultados del segundo train para el primer caso de prueba 41

Figura 13. resultados del primer train (izda) y segundo train (dcha) 43

Figura 14. Verdad del terreno para primer caso de pruebas 43

Figura 15. Imágenes para el segundo caso de prueba 44

Figura 16. Gráfica de resultados del primer train para el segundo caso de prueba 44

Figura 17. Gráfica de resultados del segundo train para el segundo caso de prueba 45

Figura 18. Resultados del primer y segundo train para el segundo caso 46

Figura 19. Verdad del terreno para el segundo caso 47

TABLAS

Tabla 1. Resultados de la competición 15

Tabla 2. Tabla de tiempos reales y estimados de realización de tareas 23

Tabla 4. Resultados comparativos del primer y segundo train con el primer caso 41

Tabla 5. Resultados comparativos del primer y segundo train con el segundo caso 45

Tabla 6. Tiempos de ejecución para el primer y segundo caso 46

Algoritmos

Algoritmo 1.1 Segmentación semántica de imágenes multi-espectrales usando Deep Learning ..... .53

1

Introducción y definiciones

PROPOSITO

El objetivo es encontrar un algoritmo de **Deep Learning** para detectar carreteras y caminos en imágenes por satélite.

1.1. Definiciones y conceptos

A lo largo de este proyecto se van a utilizar una serie de términos comunes en el campo de la inteligencia artificial y visión por computador y cuyas definiciones nos ayudarán a contextualizar y entender mejor el entorno en el que vamos a trabajar.

1.1.1. Inteligencia Artificial y Machine Learning

En *ciencias de la computación*, las maquinas << inteligentes >> ideales son agentes flexibles que perciben el medio y ejecutan acciones que maximizan sus posibilidades de éxito en algún objetivo o tarea[1] .

Cuando se da este caso, decimos que el agente posee ***IA,*** que es la implementación de alguna forma de inteligencia por estas máquinas para pretender imitar las funciones << cognitivas >> que los humanos asocian con otras mentes humanas como las siguientes:[2]

- Percibir
- Razonar
- Aprender
- Resolver problemas

> *Inteligencia Artificial: La capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible*
>
> - Andreas Kaplan y Michael Haenlein - - Definición de inteligencia artificial [3]

En esta rama de la computación, la actualización es continua, ya que tecnologías que una vez requerían de este tipo de inteligencia pueden convertirse en tecnología común. Esto se debe a que las máquinas se vuelven más capaces con el paso del tiempo.

Un ejemplo de este caso, es el reconocimiento óptico de caracteres (ROC) (OCR según las siglas en inglés), proceso dirigido a la digitalización de textos que se reconocen automáticamente a partir de ejemplos visuales de símbolos o caracteres que pertenecen a un determinado alfabeto. Como ejemplo del avance tecnológico los OCR, hoy en día, se perciben como una tecnología común, cuando hace poco tiempo eran objeto de investigación en el área de inteligencia artificial. Sin embargo, sistemas capaces de jugar al ajedrez o al Go[I], o *sistemas de conducción autónoma* en donde se pueden aplicar algunos resultados de este proyecto, si son ejemplos de esta tecnología.

[I] El go es un juego de mesa de estrategia para dos personas, originado en China hace más de 4000 años y considerado una de las cuatro artes esenciales de la antigüedad China. Los textos más antiguos que referencian este juego son las analectas de Confucio.

1.1.2. Deep Learning

Mediante algoritmos de ***Machine Learning*** se intentan modelar abstracciones de alto nivel a partir de datos usando arquitecturas computacionales que admiten una serie de transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial.

No existe únicamente una sola definición de *Deep Learning* ya que se trata de clases de algoritmos ideados para diversos ámbitos, pero todos ellos presentan las siguientes características remarcables:

- El uso de una cascada de capas con unidades de procesamiento no lineal para extraer y transformar variables. Cada capa usa la salida de la capa anterior como entrada. Los algoritmos pueden utilizar aprendizaje supervisado o aprendizaje no supervisado, y las aplicaciones incluyen modelización de datos y reconocimiento de patrones.
- Los algoritmos se basan en el aprendizaje multinivel de características o representaciones de datos. Las características de más alto nivel se derivan de las características de niveles inferiores para formar una representación jerárquica.
- El aprendizaje en múltiples niveles de representación se corresponde con diferentes niveles de abstracción formando así una jerarquía de conceptos.

Cualquiera de las maneras de definir el *Deep Learning* tienen en común lo siguiente:

a) Múltiples capas de procesamiento no lineal
b) Aprendizaje supervisado (o no) de representaciones de características en cada capa
c) Formación jerárquica de las capas de características desde un nivel de abstracción más bajo a uno más alto.

Estos algoritmos se caracterizan por el número de transformaciones (capas) que se aplican a la señal de entrada mientras se propaga desde la capa de entrada a la capa de salida. Estas transformaciones generan o incluyen parámetros que se pueden entrenar como pesos y umbrales. Para que se considere *Deep Learning* implica que las transformaciones intermedias tienen que ser más de dos. Cuando solo hay una transformación intermedia se denomina Shallow learning. De todas maneras, las arquitecturas actuales manejan cientos de capas intermedias.

1.1.3. Redes Neuronales

Las redes neuronales son consideradas como ***sistemas conexionistas.*** Son modelos computacionales inspirados en el comportamiento del homólogo biológico[4] . Consisten en un conjunto de unidades llamadas *neuronas artificiales*, conectadas entre sí para propagar señales desde la capa de entrada a través de las capas intermedias hasta llegar a la capa de salida generando unos *valores de salida*.

Como podemos observar en el *Esquema de una red neuronal* (*fig. 1),* cada ***nodo*** o ***neurona*** está conectada con otras mediante ***conexiones*** formando un grupo interconectado de nodos similar a la vasta red de neuronas en un cerebro biológico. Estas conexiones representadas mediante flechas, van desde la *salida de una neurona* a la *entrada* de otra.

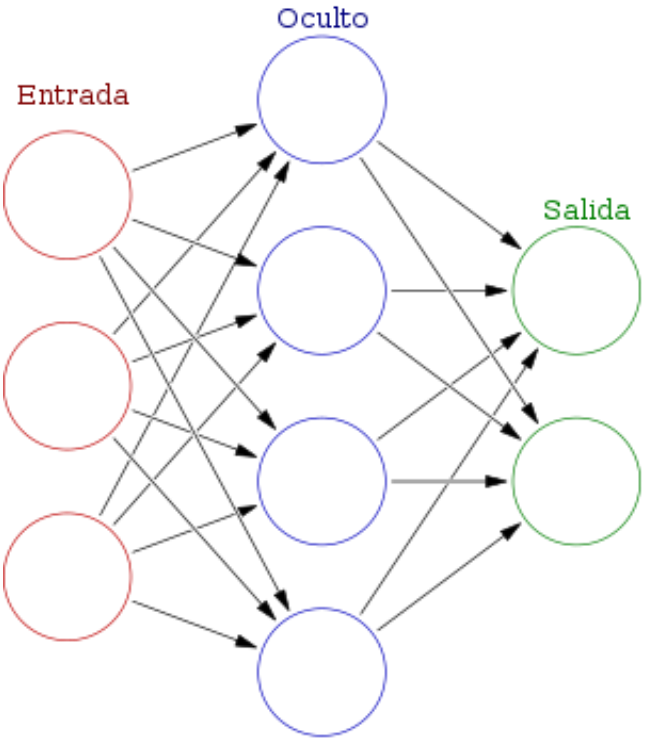

fig 1: Esquema general de una red neuronal

Estas conexiones tienen como atributos unos ***pesos***, el cual es multiplicado por la salida de una neurona anterior, que pueden *incrementar* o *inhibir* el estado de activación de las *neuronas adyacentes*. Además, el resultado a la salida de una neurona puede estar *modificado o limitado*, por una función de activación que filtra de forma no lineal la salida de la unidad, usualmente se produce una saturación fijando un valor que no se debe sobrepasar antes de propagarse a una neurona. [5] [II]

Usualmente esta función de activación suele ser de tipo **sigmoidal**, Por ejemplo, en las redes neuronales de propagación de impulsos, el modelo permanece en cero hasta que se recibe la entrada, donde la frecuencia de activación se incrementa rápidamente y gradualmente llegara a ser *asíntota* cuando la frecuencia es del 100% [III]

En la figura 1, cada nodo circular (Rojo, azul, verde) representa una neurona artificial, y a su vez, cada conjunto de nodos (separados por colores) puede representar una capa de nivel de abstracción.

II. Como las redes neuronales en ingeniería informática se pueden considerar que están inspiradas sobre la biología, esta función representa una *tasa de potencial de activación* activándose en la celda. Esta función se puede escribir como $\phi(v_i) = U(v_i)$, donde U es la función escalón, o como una función rampa, $\phi(v_i) = \mu v_i$, donde μ es la pendiente, para reflejar el incremento del *potencial de activación.* Si se usa este modelo estas redes neuronales pueden llegar a tener convergencia inestable ya que las entradas tienden a incrementarse sin límite y esta función no es normalizable.

III. La función de activación sigmoidal se puede definir como $\phi(v_i) = U(v_i)\tanh(v_i)$, donde la función de tangente hiperbólica puede ser cualquier otra función sigmoidal.

1.1.4. Redes Neuronales Convolucionales

Las **RCN** son un tipo de red neuronas artificiales donde los nodos corresponden a campos receptivos de una manera muy similar a las neuronas en la *corteza visual* primaria (V1) de un cerebro biológico, y que son variaciones de un **perceptrón multicapa**, que implementan convoluciones sobre la capa de entrada, de ahí el nombre genérico de estas redes.

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 este tipo de redes son nuestra gran apuesta para este proyecto.

Como los procesos de segmentación y clasificación se basan en extracción de características más relevantes, en esta fase, las neuronas sencillas de un perceptrón se reemplazan por objetos matriciales que realizan operaciones sobre los datos de las imágenes de dos dimensiones que pasan por ellas, en vez de un único valor.

La salida de cada neurona se puede calcular como una combinación lineal de:

$$Y_j = g\ (b_j + \sum_i K_{ij} \otimes\ Y_i)$$

Donde,

- Y_j: la salida en forma matricial
- Y_i: las salidas de las neuronas en la capa anterior
- K_{ij}: núcleo correspondiente a esa conexión
- b_j: Influencia
- $g(\bullet)$: función de activación no-lineal
- $\otimes$: Operador Convolución, que tiene el efecto de filtrar la imagen de entrada con un núcleo previamente entrenado, donde ciertas características que se buscan o se pretenden buscar son más dominantes, puesto que los pixeles que las representan tienen un valor mayor.

En este tipo de redes, existe otro tipo de neuronas que se denominan **neuronas de reducción de resolución**, que mejoran con la tolerancia a pequeñas perturbaciones en los datos de entrada. Es decir, con imágenes muy parecidas que se pueden diferenciar en algunos pixeles si se analizan de esta manera el resultado debería ser muy parecido también, por no decir el mismo. Esto se consigue, cuando reducimos la resolución, haciendo que el campo receptivo para las unidades en las capas más profundas sea mayor.

Este efecto se puede conseguir mediante un proceso de ***subsampling***, un proceso de *muestreo de color* que se aplica en la compresión, asociado al códec, y que afecta a todo el proceso desde la elección de la cámara hasta cada decisión que se toma en el *workflow*. Sin embargo, como nuestro proyecto se trata de detectar una carretera o un camino que, si se considera el porcentaje de pixeles totales de la imagen como el 100%, se puede considerar que los pixeles correspondientes a la carretera o camino constituyen entre un 10% y un 30% de los pixeles totales. Es decir, estamos trabajando con lo que se puede considerar una región de la imagen y resumimos sus características sobre esa región, tratando de etiquetar cada pixel como **[no carretera / carretera]*** . Por lo tanto, a la hora de diseñar una red neuronal, tendremos que tener en cuenta que para las *neuronas de reducción de resolución* es mucho más efectivo utilizar operaciones de ***max-pooling***, ya que son más eficaces para este tipo de proceso.

* Los caminos entran dentro de lo que se considera una carretera en este proyecto

Además, existe evidencia que este tipo de operación es similar a la forma en que la corteza visual puede resumir la información internamente.

La operación de max-pooling encuentra el valor máximo entre una ventana de muestra y pasa este valor como resumen de características sobre esa área. Como resultado, el tamaño de los datos se reduce por un factor igual al tamaño de la ventana de muestra sobre la cual se opera.

Las **neuronas de clasificación** pertenecen a la última fase de extracción de características, donde los datos ya han sido depurados hasta una serie de características únicas para la imagen de entrada. Finalmente se pueden clasificar estas características hacia una etiqueta u otra, según los objetivos de entrenamiento.

En esta fase, el comportamiento de las neuronas es idéntico al de los perceptrones multicapa, y la salida de cada neurona se puede calcular como una combinación lineal de:

$$Y_j = g\,(b_j + \sum_i W_{ij} \bullet Y_i)$$

Donde,

- Y_j: la salida de la neurona j
- Y_i: las salidas de las neuronas en la capa anterior
- W_{ij}: peso correspondiente a esa conexión
- b_j: Influencia
- $g(\bullet)$: función de activación no-lineal

En nuestro proyecto estamos trabajando con imágenes, pero las redes neuronales convolucionales también pueden ser aplicadas para la clasificación de series de tiempo o señales de audio utilizando convoluciones en 1D, así como para la clasificación de datos volumétricos usando convoluciones en 3D.

1.1.5. Perceptrón Multicapa

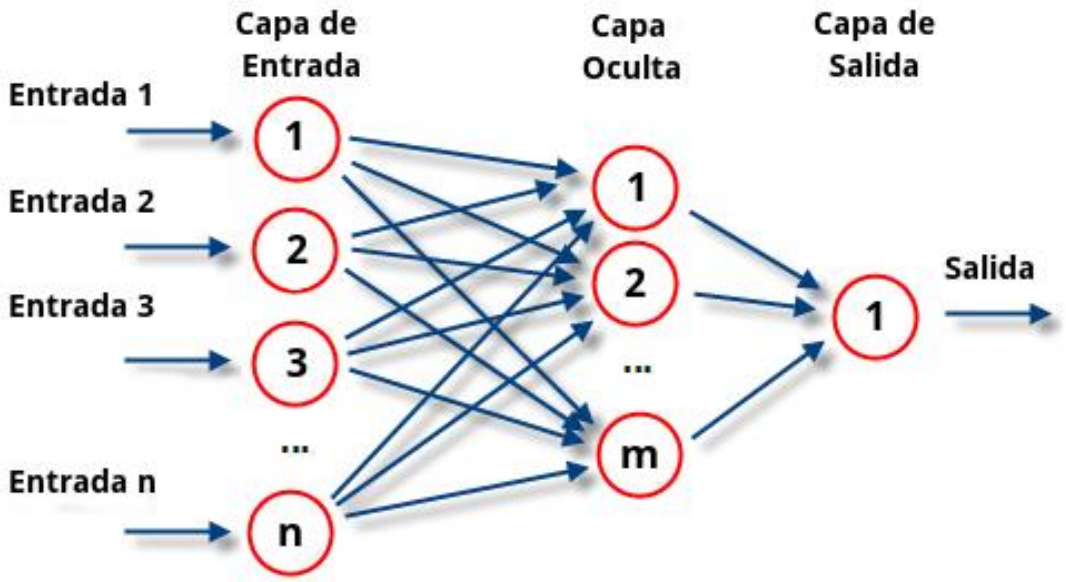

fig 2: Ejemplo de un perceptrón o RNA

El perceptrón multicapa es una RNA formada por múltiples capas, como su propio nombre indica, de tal manera que tiene capacidad de resolver problemas que no son linealmente separables. Además, puede estar conectado de dos formas distintas:

- Totalmente conectado: La salida de una neurona de la capa i es la entrada de todas las neuronal de la capa i + 1
- Localmente conectado: Cada neurona de la capa i es entrada de una serie de neuronas, denominadas región, de la capa i + 1.

1.1.6. Tensores

Un tensor es un objeto algebraico que se pueden definir mediante propiedades de transformación ante cambios de sistemas de coordenadas. Cuando esta coordenadas son cartesianas ortogonales, se les denomina tensores cartesianos.

Las como unidades de procesamiento tensorial, o TPU son circuitos integrados de aplicación específica y aceleradores de inteligencia artificial, desarrollados por Google para el aprendizaje automático con RNAs y más específicamente optimizado para usar TensorFlow, la biblioteca de código abierto para Machine Learning.

1.2. ¿Qué es un algoritmo de Deep Learning?

Los algoritmos de **Deep Learning**[9] son los que ejecutan los datos a través de varia capas de algoritmos de redes neuronales, que pasan a una representación más simple de los datos para la siguiente capa, cuando el conjunto de datos es muy grande, ya que presentaran cientos de características distintas o simplemente el tamaño en el que se representan es muy grande, llegando a tener miles de columnas.

Sin embargo, puede ocurrir que el conjunto de datos no sea estructurado pudiendo llegar a tener una cantidad tan grande de características que este proceso se puede volver completamente inviable. Por lo tanto, como estos algoritmos *aprenden progresivamente* sobre a medida que pasa por cada capa de red neuronal, son una apuesta optima para este proyecto.

Por lo general, las primeras capas aprenden a detectar características de bajo nivel, como puede ser un borde, y las siguientes capas combinan las características de las capas anteriores en una representación holística.

1.3. Problema

El objetivo principal es experimentar con arquitecturas de Deep Learning para extraer automáticamente carreteras y caminos en imágenes por satélite. Para la demostración práctica de estas técnicas nos hemos basado en los datos proporcionados por una competición que se propuso en 2018 por CodaLab.[10]

1.3.1. CodaLab Challenge

DeepGlobe Road Extraction Challenge es la competición propuesta que va acorde con lo que queremos conseguir en este poryeccto. Esta competición propone extraer automáticamente carreteras y redes de calles en imágenes por satélite, ya que, en zonas de desastre, especialmente en zonas de desarrollo, los mapas y la información de accesibilidad son cruciales para una respuesta de crisis.

Actualmente los mejores scores obtenidos con respecto a los resultados son:

POSICION	USUARIO	SCORE
1	Ali_iDST_Deep_Learning	0.6577
2	EOS_Data_Analytics	0.6560
3	Novelfor	0.6559

Tabla 1: Resultados de la competición

El problema se puede considerar como un problema binario de clasificación a nivel de pixel, cada input es una imagen tomada por satélite, y se debe predecir una máscara para ese input. Por ejemplo, una imagen binaria del mismo tamaño que la imagen de entrada.

Se proponen dos fases en el desafío:

- Fase 1: Fase de desarrollo. Se provee de un dataset de entrenamiento etiquetado y un dataset sin etiquetar de validación
- Fase 2: Fase final. El dataset de test será publicado y se tendrá que probar lo realizado con ello.

Los resultados de la clasificación se evaluan usando la métrica IoU (Intersection over Union), que se ilustra en la figura 3.

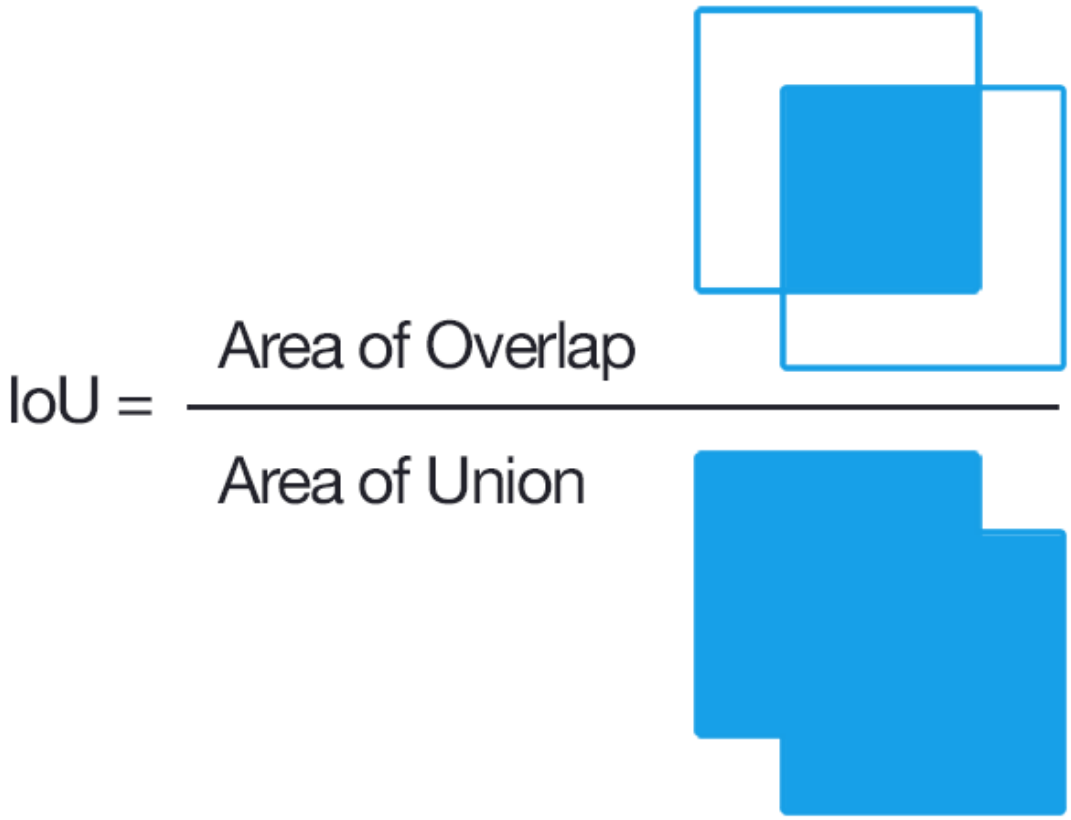

fig 3: La obtención del IoU es una simple división del área de solapado entre los bounding boxes del área de unión

1.3.2 DeepGlobe 2018

DeepGlobe 20018 Satellite Image Understanding Challenge es la competición que antecede a la que se sigue como referencia. En esta competición se realizaron tres competiciones públicas: segmentación, detección y clasificación. Cada competición corresponde a tres datasets distintos con las correspondientes metodologías de evaluación que están coherentemente empaquetadas en cada competición co-localizadas con CVPR 2018 (Computer Vision and Pattern Recognition)

Existen otras competiciones similares de Visión por Computador como pueden ser la DAVIS (densely annotated video segmentation) y COCO (Common objects in context) en las que se basan esta competición y la de CodaLab

Las imágenes por satélite son una gran fuente de información estructurada, y es la que menos está siendo investigada en comparación con las fotos del dia a dia. Esta fuente poderosa de información es más estructurada y uniforme que recientemente se utiliza para la composición de mapas, análisis poblacional, agricultura de precisión efectiva y conducción autónoma.

Unir la visión por computador moderna con el análisis de datos de teledetección puede tener un impacto crítico en la forma en que se entiende el entorno y conducir a grandes avances en la planificación urbana global o la investigación del cambio climático.

El Objetivo de DeepGlobe es atraer a diferentes investigadores de diferentes ámbitos para concienciar sobre el Remote Sensing en la visión por computador y viceversa.

Los objetivos del desafío eran los siguientes:

- Evaluar los enfoques de comprensión de las imágenes por satélite de última generación, que con suerte pueden servir como puntos de referencia para futuras investigaciones sobre el mismo tema.
- Analizar las características de cada conjunto de datos
- Definir los criterios de evaluación de las competiciones
- Proporcionar líneas de base para cada tarea.

Como bien se sabe, los métodos que dominan en el campo de vision por computador son los metodos de Machine Learning evaluados sobre los Datasets públicos que proporcionan escalabilidad y fiabilidad en la evaluación de distintos enfoques, utilizando los mismos datasets anotados y técnicas de training/validation.

Algunos ejemplos de datasets famosos:

- ImageNet -> detección de objetos
- COCO -> Image captioning
- DAVIS -> segmentación de objetos

Esta metodología de evaluación aumenta la visibilidad, disponibilidad y la factibilidad de los modelos de Machine Learning que trae algoritmos más escalables, diversos y precisos evaluados en enfoques públicos.

Alguna características específicas de DeepGlobe 2018:

1. Contiene 3 datasets estructurados para resolver 3 tareas diferentes de compresión de imagen por satélite
2. Organiza 3 competiciones públicas para proponer soluciones a esas tareas
3. Reúne a investigadores de diferentes ámbitos para resolver tareas similares en un ámbito de colaboración
4. Puntos de referencia justos y durables para futuros análisis de imagen por satélite

Como las tareas involucran problemas de visión por computación “in the wild” (clasificación de imágenes, detección y segmentación semántica) los datasets de las competiciones pueden convertirse en valiosos bancos de pruebas para el diseño de algoritmos de visión robustos

Las tres tareas que define DeepGlobe:

- Road Extractions Challenge: En zonas de desastre, especialmente en países que se están desarrollando, los mapas e información de accesibilidad son cruciales para respuestas de crisis.
  El reto: Extraer automáticamente redes de caminos y calles remotamente de las imagenes por satelite como un primer paso hacia la respuesta de crisis automática y el aumento de la cobertura del mapa para la conectividad.

- Building Detection Challenge: Como lo demuestran los recientes eventos naturales catastróficos, modelar la dinámica de la población es de gran importancia para la respuesta a desastres y recuperación. Por lo tanto, modelar la demografía urbana es una tarea vital y la detección de edificios y áreas urbanas son clave para lograrlo.
  El reto: Detectar automáticamente edificios a partir de imágenes de satélite para recopilar información urbana agregada de forma remota, así como para recopilar información detallada sobre distribución espacial de asentamientos urbanos.

- Land Cover Classification Challenge: Categorización automática y segmentación de la cobertura de suelo son clave para el desarrollo sostenible de la agricultura, silvicultura y planificación urbano.
  El reto: Clasificar los tipos de suelos a partir de las imágenes por satélite para la solución automática económica y agricultural.

Datasets combinados: +10k imagenes por satelite

Road Extraction Dataset

DigitalGlobe + Vivid Images dataset: Contiene imágenes capturadas alrededor de Tailandia, Indonesia e India.

- Ground Resolution de los pixels de la imagen: 50 cm/pixel
- 3 canales: Red, Green and Blue
- Geotiff imagenes originales: 19584 x 19584 pixels

Proceso de anotación: Embaldosar y cargar las imágenes en QGIS

- Embaldosado: Determinar las áreas útiles a muestrear para esos países
- Areas utiles: se muestrea uniformemente entre las áreas urbanas y rurales
- Después del muestreo se selecciona la correspondiente imagen DigitalGlobe tiff de esa área
- Se recortan para extraer subregiones viables y las subregiones relevantes se muestrean por expertos GIS
- Subregión viable: denota la parte de la imagen donde hay una buena proporción entre buenos y malos ejemplos. A la hora de seleccionar las subregiones se intentan muestrear las áreas interesantes uniformemente (diferentes tipos de superficie de carretera: pavimentada, sin pavimentar, tierra; áreas urbanas y rurales)

Las etiquetas generadas se realizan a nivel de pixel, donde todos los pixeles pertenecientes a la carretera están etiquetados, en vez de solo los centrales.

Dataset: 8570 imagenes, que extiende un area de 2220km^2

- Training dataset: 6226 imagenes (72%) 1632km^2
- Validation dataset: 1243 imagenes 362km^2
- Test dataset: 1101 imagenes 288km^2

Training/validation/testing: baldosas aleatorias para conseguir una distribución aproximada del 70%/15%/15%

- Training dataset: 4.5% pixeles positivos, 95.5% negativos
- Validation dataset: 3% pixeles positivos, 97% negativos
- Test dataset: 4.1% pixeles positivos, 95.9% negativos

La morfología urbana y las condiciones de iluminación, la densidad de carreteras y la estructura de las redes de calles son significativamente diversas entre las muestras

Building Detection Dataset

SpaceNet Building Detection Dataset, satellites PRRS y ISPRS estan basados en áreas pequeñas y en algunos casos, una combinación de datos ópticos y LiDAR

Inria Aerial Image Labeling cubre 810 km^2 de área con 30cm de resolución en diferentes ciudades Europeas y Americanas. Abordó la portabilidad del modelo entre áreas, ya que algunas ciudades se incluyeron solo en training data y otras solo en el testing data. SpaceNet fue el primer desafío que involucró grandes áreas, incluidas ciudades de Asia y África.

Dataset: Cuatro areas - Las Vegas, Paris, Shanghai y Khartoum

- Dataset etiquetado: 24586 200m x 200m (650 x 650 píxeles) de escenas no superpuestas que contienen un total de 302701 planos de edificios a través de diferentes áreas (urbanas, suburbanas)
- Source: sensor WorldView-3 con 31 single-band panchromatic, 1.24m multispectral sensor providing 8-band multi- spectral imagery with 11-bit radiometric resolution.
- En cada escena está identificada y proporciona un plano poligonal para cada edificio. Cualquier azotea parcialmente muestreada ha sido aproximada para representar la forma del edificio. Los edificios adyacentes han sido marcados como un único edificio.
- Training/validation/test: 60%/20%/20%
- Se prescinde del análisis de desacuerdo anotado
- Cada área es cubierta por una única imagen de satélite

Cada área está cubierta por una sola imagen por satélite, que constituye un problema de resolución más fácil comparado con datos donde las diferentes partes están cubiertas por imágenes con distintos ángulos y luz, y diferentes condiciones atmosféricas. El proceso de compensación atmosférica puede procesar imágenes para crear datos que reflejan la superficie reflectante, reduciendo así los efectos de la atmosfera, pero diferentes larguras de sombras y diferentes orientaciones de satélites pueden originar problemas para los algoritmos de detección

si esos modelos se usan para clasificar imágenes adquiridas a diferentes tiempos con condiciones de luz y ángulos de satélite distintos.

Land Cover Classification Dataset

Segmentación semántica:

- ISPRS Vaihingen: 33 imágenes de distinto tamaño (media 2000x2000) con 16 imágenes totalmente anotadas
- ISPRS Potsdam: 38 imágenes de tamaño 6000x6000 pixels con 24 imágenes anotadas
    - 6 clases
    - Área urbana de ciudades con casos limitados de objetivos urbanos (coches, arboles, edificios
- Zeebruges: dataset de 7 baldosas 10000x10000 con 8 clases (suelo y objetos) RPG de una resolucion de 5cm
- Dstl Kaggle

2

Planificación y Gestión del Proyecto

En este capítulo se explicarán los objetivos del proyecto y las tareas que se han llevado acabo para su desarrollo.

2.1. Objetivos del proyecto

Como se ha comentado anteriormente el objetivo del proyecto consiste en experimentar con técnicas de Deep learning para detectar carreteras y caminos en imágenes por satélite utilizando técnicas de Deep Learning.

Primeramente, ya que en el ámbito de la inteligencia artificial y hay distintos enfoques en los algoritmos que se usan, por tanto, habrá que buscar distintos enfoques que nos puedan servir para el proyecto.

Se seleccionará una base de datos con la que trabajar y aplicar el algoritmo.

Una vez obtenido el dataset, se someterá a un pre-procesado, donde se analizarán los datos y acondicionarlos para el problema a tratar.

Por último, se seleccionará o diseñará un algoritmo para la fase experimental, donde se evaluarán los modelos predictivos y se ejecutarán sobre la base de datos.

Finalmente, se recogerán y analizarán los resultados para extraer conclusiones.

2.2. Tareas y estimación de tiempos

Tareas	Tiempo real	Tiempo estimado
Planificación y gestión	24	24
Búsqueda y lectura de información	30	30
Reuniones	18	12
Memoria*	180	180
Captura y pre-procesado de datos	30	24
Experimentos*	240	360
Recogida de resultados*	240	360
TOTAL	**460**	**840**

Tabla 2: Tabla de tiempos reales y estimados de realización de las tareas

* Las tareas marcadas con asterisco van en paralelo

Se empezó a trabajar en este proyecto desde la segunda quincena de julio, aprovechando los meses de verano, y ha continuado hasta el fin de su defensa.

En un principio la defensa del proyecto estaba prevista para febrero, por eso se hizo la estimación de 840 horas que equivaldrían a 7 meses, desde agosto hasta el fin de la defensa en febrero.

Hemos dividido el proyecto en las siguientes tareas:

- Planificación y Gestión del Proyecto: todas las tareas relacionadas con la Planificación y Seguimiento y Control, junto con las reuniones con el tutor

- Búsqueda y lectura de información: Todas aquellas tareas dedicadas a la formación y comprensión sobre temas relacionados con el proyecto.

- Captura y pre-procesado de datos: Comprende todas aquellas tareas para la búsqueda de datos, el pre-procesado, acondicionamiento de los datos, extracción de características y features, y tratamientos de las imágenes.

- Experimentación: Recopila todas las tareas relacionadas con la construcción de los modelos y diseños de la red seleccionada y algoritmos que se aplicarán a los datos.

- Recogida de resultados: Son las tareas relacionadas con la visualización de los datos obtenidos, mediante tablas, gráficas, etc.

- Memoria: Todas las tareas relacionadas con la redacción de la memoria

2.3. Análisis de riesgos

Este proyecto trata de un trabajo completo que incluye todas las tareas, desde la obtención de los datos, hasta la construcción y evaluación de los algoritmos de clasificación. Pero, aunque existen publicaciones sobre temas relacionados, ya sea en el ámbito de la medicina o en el propio campo de la detección de objetos, se asume que no está exento de riesgos.

En general para evitar cualquier imprevisto, relacionado con el desarrollo de los algoritmos y obtención de datos, se comenzará a trabajar en los meses de verano.

La obtención de datos se hace a través de una competición de CodaLab finalizada en 2018-2019, por lo que temíamos, que se tardase un poco en obtener los datos. Para ello, se evaluará la posibilidad de buscar datos por otras vías (por ejemplo, Google, Google Maps, o la página web de la NASA) imágenes que se puedan asemejar a las de un satélite. Hay que comentar que tardé relativamente poco en obtener una respuesta (una semana) y hubo pocos problemas para obtener los datos, salvo la falta de espacio en el disco, que se arregló con discos externos.

Un riesgo importante es, la perdida de la información por problemas con la máquina con la que se va a trabajar. Para evitar esto, se usarán servicios cloud (Drive para fichero, código y memoria, GitLab y Github para el código, al igual que el propio servicio de MatLAb para edición y guardado de código conocido como Matlab Online y Mathworks; Overleaf para la memoria, al igual que copias locales en la máquina y en memorias flash o discos duros) para guardar la información.

Por otra parte, se necesita un conjunto de datos aceptable, para que la red neuronal aprenda y se obtenga unos resultados significatios, y además todos los algoritmos de Deep Learning y Machine Learning necesitan un hardware adecuado, especificamente, procesadores y GPUs potentes que permitirán paralelizar y obtener resultados de forma 'rapida' aun así estos algoritmos pueden tardar dia. Por lo que se ha optado por hacer un buen diseño de los algoritmos y de la red convolucional a usar y analizar el funcionamiento con un pequeño conjunto de datos del dataset proporcionado.

En todo momento se analizará la situación para evaluar la viabilidad de realizar mejoras que lleven a resultados positivos.

2.4. Software: Matlab

Durante el proyecto se ha trabajo con MatLab, un software que proporciona materiales y herramientas para uso académico.

Unas de las herramientas que MatLab proporciona son toolkits para el procesado de imágenes, diseño de redes y algoritmos adecuados a las herramientas disponibles

Las herramientas proporcionadas que mas se han utilizado han sido:

- Machine Learning and Deep Learning Toolbox

  Deep Network Designer: Esta herramienta nos permitirá diseñar nuestra propia red o importar ejemplos que ya están diseñados en Matlab

- Image Procesing and Computer Vision

  Image Browser: Nos permite cargar imágenes y salvar las imágenes con la extensión (*.mat*) ya que Matlab trabaja mayormente con este tipo de variables.

  Image Batch Processor: Nos permite aplicar una función programada en MatLab a un conjunto de imágenes seleccionadas, sin tener que usar bucles o cambiar el parámetro de la imagen cada vez

  Color Thresholder: Permite definir valores umbrales de color, generalmente para traducirlos a blanco y negro.

  Image Region Analyzer: Permite cargar y analizar componentes conectados en una imagen, lo que se conoce como regiones

  Image Segmenter: Permite la segmentación de una imagen y refinar las regiones

3

Desarrollo del Proyecto

En este capítulo se explica la obtención de los datos y el pre-procesado de los datos.

3.1. Obtención de los datos

Como ya se ha comentado anteriormente, los datos con los que se va a trabajar proceden de una competición, en la cual nos hemos tenido que dar de alta y esperar a que nuestra solicitud se aceptara para obtener el dataset con el que se ha trabajado

El dataset contiene 14798 imágenes con extensión JPG de tamaño 1024x1024 que se dividen entre imágenes a color RGB de las imágenes de satélite y sus máscaras de la verdad de terreno en blanco y negro. Todas estas imágenes se dividen en tres conjuntos:

- Train: 6226 imágenes a color RGB tomadas por satélite con las máscaras correspondientes
- Valid: 1243 imágenes a color RGB tomadas por satélite sin máscaras
- Test: 1101 imágenes a color RGB tomadas por satélite sin máscaras

Las imágenes que constituyen el dataset son Imágenes rurales, que pueden ser áreas de bosque y áreas de desierto (Zonas verdes y zonas de tierra), e imágenes urbanas, que pueden ser áreas de costa, autovías y autopistas y calles. *Ver fig 4*

fig 4: Muestra de cada tipo de imagen

3.2. Datos curiosos a tener en cuenta

Ya que se está trabajando con imágenes por satélite, no sabemos cómo están presentados los datos, pero si partimos de la explicación de DeepGlobe, precisamente nos encontramos con imágenes que presentan nubes, sombras e incluso impurezas en la propia imagen (zonas en las que hay puntos negros o zonas blancas que pueden haberse generado al sacar la imagen, o al exportarlo a un tipo de imagen apropiada para procesar porque la resolución es muy alta).

Otro tipo de casos curiosos que se nos presentan, son las imágenes en las que la densidad verde o la zona de tierra es tan amplia, que se puede considerar que no hay camino cuando en realidad sí que lo hay. O imágenes en las que la división de la tierra se puede confundir con caminos, o zonas de costa que no presentan caminos a priori o esta camuflado.

Imágenes en las que el camino está presente en un borde de la imagen en principio no deberían presentar un problema, pero dependiendo en qué tipo de terreno se encuentre debería tratarse con más cuidado

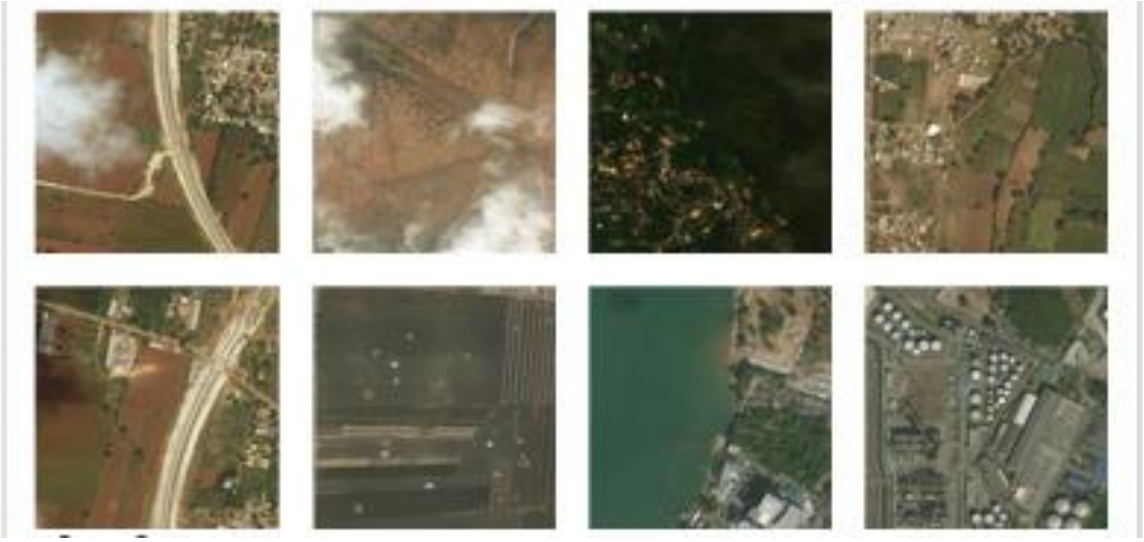

fig 5: Casos curiosos

fig 6: Ejemplo de imagen donde la división del terreno se puede confundir con caminos

Tras un primer pre-proceso con un algoritmo simple para analizar donde podemos encontrar una carretera y poder obtener una máscara, la firgura 7 son 6 de los resultados obtenidos tras aplicar dicha función mediante la herramienta Image Batch Procesing donde se han cargado todas las imágenes RGB del dataset.

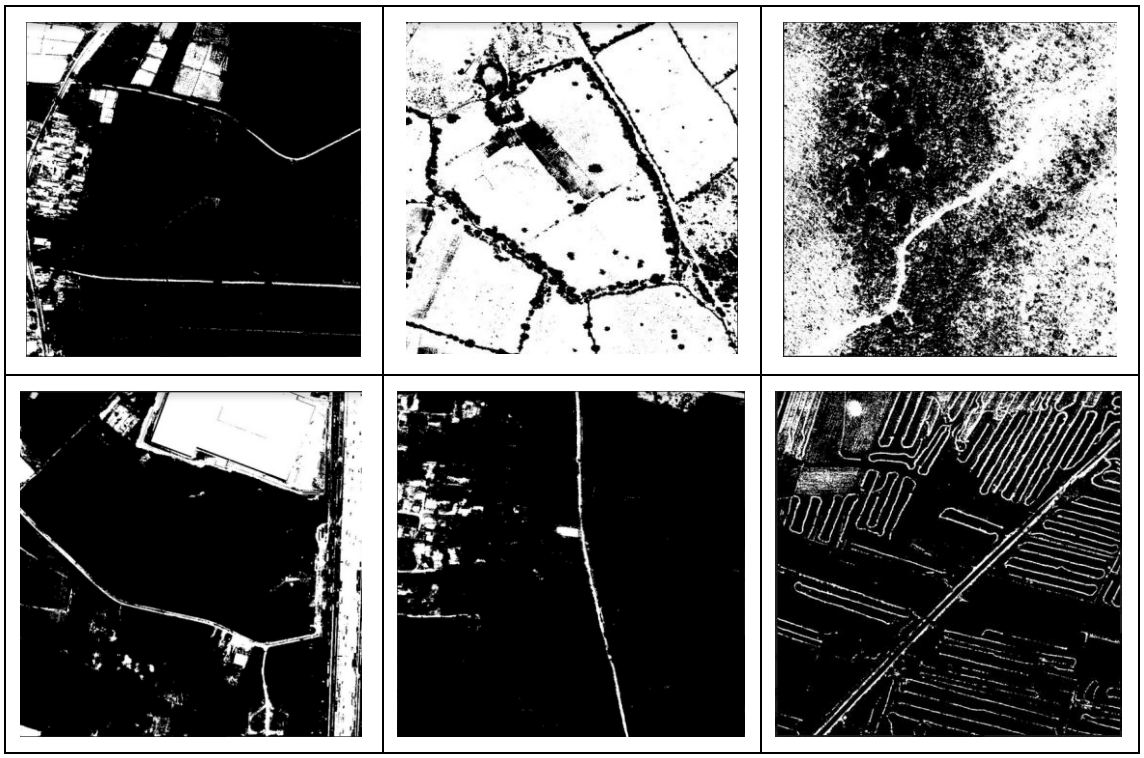

Fig 7: Resultados de un primer pre-proceso

Estas son las dos funciones principales que sigue la función que ha generado los resultados de la Tabla 2:

```
imgray = rgb2gray(im);
bw = imbinarize(imgray);
```

Mediante estas dos lineas de código lo único que estamos haciendo es pasar una imagen a color a escalar de grises y después binarizar esos valores para intentar conseguir líneas de color blanco que serán nuestras carreteras.

Como podemos observar los resultados no son buenos, ya que lo que obtenemos no coincide mucho con la verdad del terreno, ya que obtenemos más cosas además de las carreteras. Incluso en la segunda imagen de la primera fila directamente obtenemos lo que se pretende detectar en negro aparte de muchas otras cosas más. En la tercera imagen de esa misma fila, se presenta el problema que se comentaba anteriormente, donde la zona verde era muy densa y el camino se podía camuflar en esa zona.

Por lo tanto, tenemos que buscar otro enfoque que nos permita añadir técnicas de Deep Learning a nuestro algoritmo.

4

Diseño y Evaluación de modelos

En este capítulo explicaremos el procedimiento que se ha seguido para conseguir el objetivo de proyecto.

4.1. Diseño del algoritmo y la red

Como hemos concluido en el apartado anterior, necesitamos de una mejora que nos permita incorporar una red neuronal a nuestro proceso de detección.

Primeramente, se debe, diseñar la propia red, la cual va a ser clave en este proceso, y como hemos explicado anteriormente una red convolucional neuronal, va a ser nuestra mejor aliada. Tras un análisis exhaustivo, se ha llegado a la conclusión de que nos interesan las redes convolucionales del tipo U, ya que proporcionan grandes mejoras en los resultados de clasificación binaria a nivel de pixel, debido a que su aplicación es realizada en matrices bidimensionales, y una imagen es representada como una matriz bidimensional donde cada posición es el valor de un color, y son muy efectivas para tareas de segmentación de imágenes que es lo que requiere nuestra aplicación.

Para llevar a cabo estas tereas, tenemos que extraer de características más relevantes en una región, y precisamente hemos llegado a la conclusión que mediante una operación de max pooling podemos obtener dichos valores, por lo tanto, este tipo de red nos permite incorporar dicha operación, al igual que en Matlab.

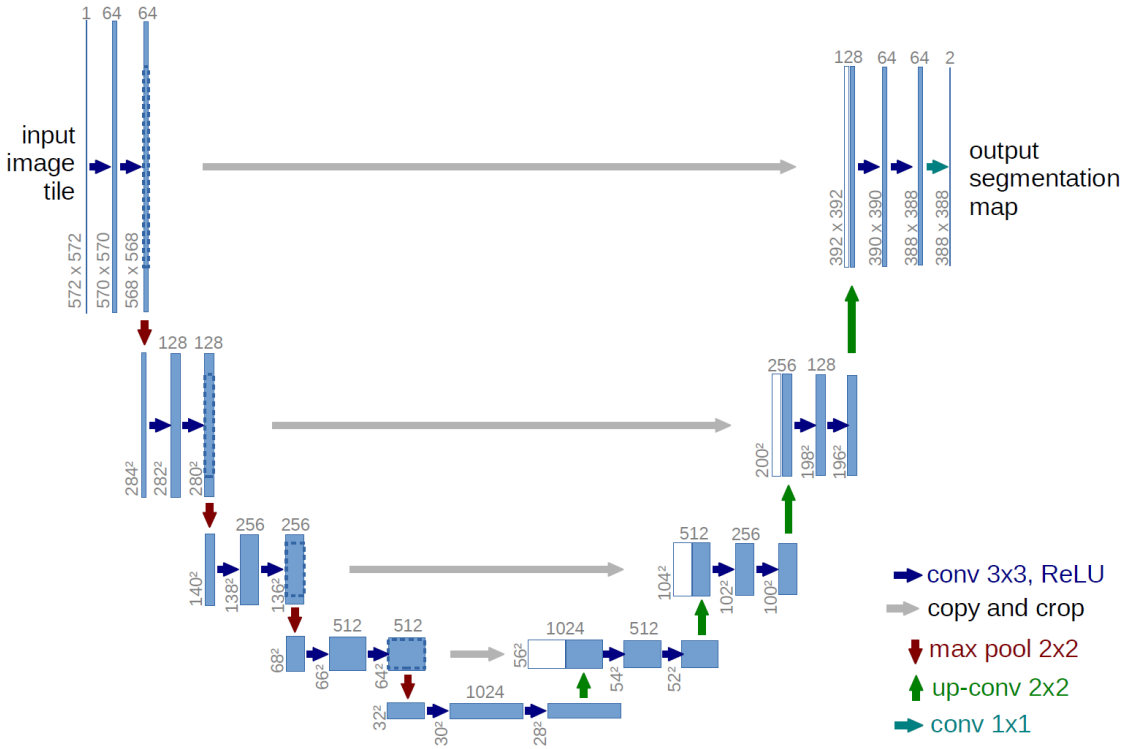

fig 8: Modelo de red CNN de tipo U que se ha seguido como modelo [10]

La arquitectura de la red U-net mostrada en la figura 8, se ha utilizado por la Universidad de Freiburg para la segmentación de Imágenes Biomédicas, y presenta muestras de 32x32 pixeles en la resolución más baja y cada caja azul corresponde a un mapa multicanal de características y la blanca representan copias, donde el número de canales es mostrado encima de cada caja. El tamaño de cada input se muestra abajo a la izquierda en cada caja. Además las flechas denotan las diferentes operaciones.

Matlab no nos proporciona un diseño propio, pero si nos permite diseñar una red similar a la de la imagen de dos maneras:

- Usando la aplicación **Deep Network Designer**
  Ver fig 8

  Cada operación en la figura 8 se va a traducir con una serie de operaciones que nos permite MatLab:

  imageInputLayer: Lo que en la imagen vemos como 'Input Image Tile.
  convolution2dLayer: Para las operaciones que se consideran conv 1x1 y conv3x3
  reluLayer: La capa que en la imagen se realiza junto con la convolución 3x3
  maxPooling2dLayer: Esta operación es la misma.
  dropoutLayer: Para hacer las operaciones de copy and crop y eliminar lo que no es necesario
  depthConcatenationLayer: Esta capa nos permite conectar salidas de capas anteriores
  transposedConv2dLayer: Lo que en la figura se identifica como up-conv 2x2
  softmaxLayer: Permite quedarnos con los valores de mayor valor y aplanar los resultados
  pixelClassificationLayer: La capa que nos clasificara cada algoritmo

  En total, se obtiene una red U-net con 58 Capas y 61 conexiones. Si usamos este método puede ocurrir que a la hora de usar el dataset, y dividirlo como train/valid, nos de un error en los tamaños del input de cada capa

- Usando la función **createUnetl(inputTileSize)**

  Esta segunda opción nos permite crear una red U-net con un parámetro de entrada que será el tamaño de la imagen y con un numero de capas definido acorde a dicho tamaño. Comparado con el método anterior se reduce significativamente el número de capas, a 34 capas al igual que las conexiones, y por tanto tendremos que usar un dataset mucho más pequeño y hacer las operaciones necesarias en el algoritmo.

  Se ha seguido el siguiente esquema de código, en el Anexo A podremos ver el código completo. **(pag 36)**

Pasos en la implementación de nuestra arquitectura

```
Cargar la imagen
      RGB
      Escala de grises

Cargar las etiquetas
      Imagen binaria (blancos y negros)
      con valores 0 o 255

Asegurarse de que las etiquetas tienen el valor adecuado (1 o 2)
y guardarla como imagen.

Extraer un trozo de la imagen
      Tamaño: 16x16 o 256x256 o 1024x1024
      Patches: 100  o  200    o   6400

Crear la red U con el tamaño del patch

Ponemos a entrenar la red con el trozo de la imagen
extraído, el grafo creado y las opciones necesarias

Net = lgraph entrenado | info = resultado

Una vez realizado el training hacemos una predicción

Segmentamos la imagen con la nueva red entrenada y el
tamaño del patch para la predicción

Se muestra la prediccion como imagen
```

fig 9: Cada sección de la red-u mostrada de izquierda a derecha

4.2. Resultados experimentales

Todas las pruebas se han realizado para verificar el correcto funcionamiento de la red y analizar cómo se comporta la estructura para el objetivo principal, utilizando los distintos casos que hemos expuesto anteriormente. Para este primer entrenamiento el algoritmo requiere de una máscara con la verdad del terreno.

La figura 10 muestra los datos para el primer caso: Imagen de zona rural verde y su máscara. Se ha aplicado la arquitectura U sobre estos datos repitiendo el entrenamiento en dos ocasiones, que se ilustran en las figuras 11 y 12.

La tabla 4 da los datos comparativos de los dos entrenamientos. La figura 12 muestra los resultados de ambos entrenamientos.

Similarmente, la figura 15 muestra los datos para el segundo caso, las figuras 16 y 17 muestran las evoluciones del entrenamiento en dos repeticiones. La tabla 5 muestra los datos comparativos de los dos entrenamientos y la figura 18 los resultados visuales del entrenamiento en este segundo caso.

fig 10: Imagenes para el primer caso de prueba: imagen satélite (izq.) y verdad del terreno (dcha)

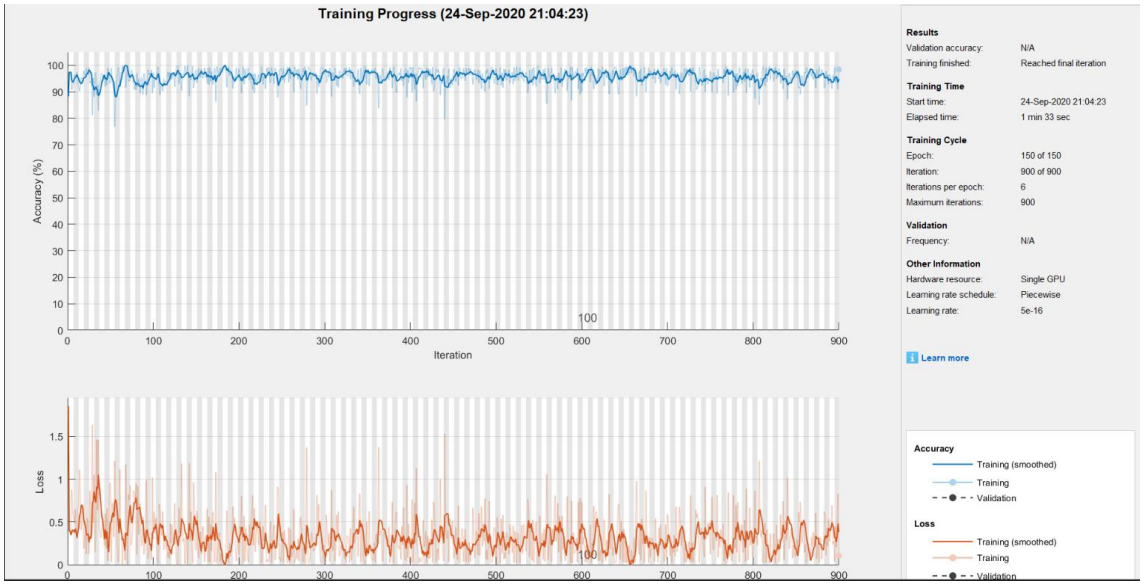

fig 11: Graficas de la evolución de los resultados del primer entrenamiento para el primer caso

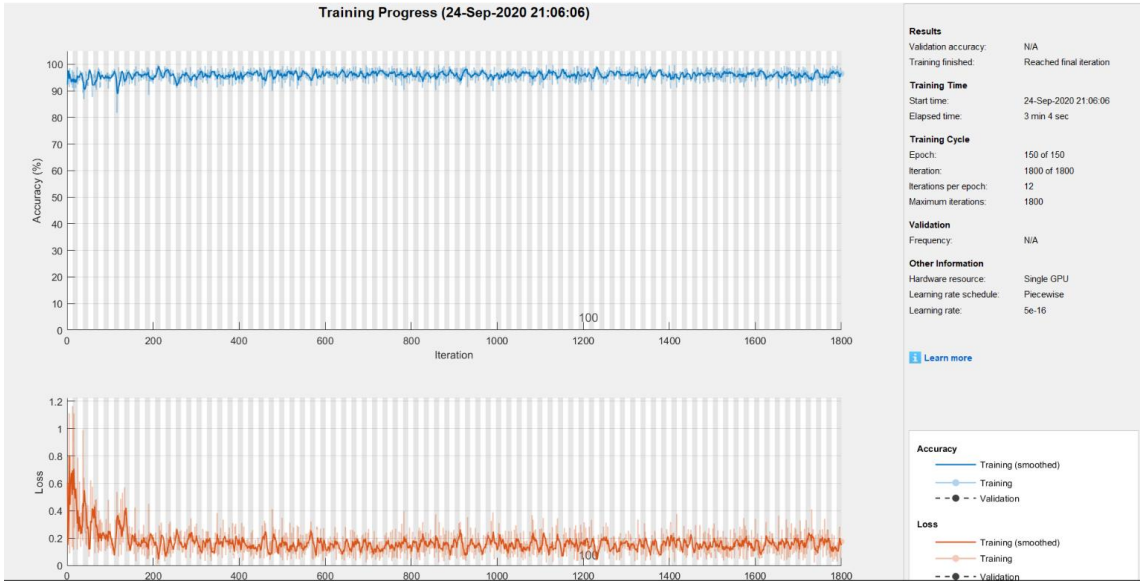

fig 12: grafica de la evolución de los resultados del segundo entrenamiento para el primer caso

INFORMACION DE LAS GRAFICAS	PRIMER TRAIN	SEGUNDO TRAIN
TAMAÑO DEL BATCH	16x16	32x32
ITERACIONES	900	1800
TRAINING LOSS	IT.1 - 1.8520 IT.450 - 0.1406 IT.900 - 0.143	IT.1 - 0.6054 IT.900 - 0.1274 IT.1800 - 0.1747

TRAINING ACCURACY	IT.1 - 88.2813 IT.450 - 96.9482 *IT.900 - 98.4375*	IT.1 - 92.3950 IT.900 - 96.4417 IT.1800 - 98.4539
BASE LEARN RATE	IT.1 - IT.120: 0.0050 IT.121 - IT.180: 5.0000E-04 IT.180 - IT.240: 5.0000E-05 IT.241 - IT.300: 5.0000E-06 IT.301 - IT.360: 5.0000E-07 IT.361 - IT.420: 5.0000E-08 IT.421 - IT.480: 5.0000E-09 IT.481 - IT.540: 5.0000E-10 IT.541 - IT.600: 5.0000E-11 IT.601 - IT.660: 5.0000E-12 IT.661 - IT.720: 5.0000E-13 IT.721 - IT.780: 5.0000E-14 IT.781 - IT.840: 5.0000E-15 IT.841 - IT.900: 5.0000E-16	IT.1 - IT.120: 0.0500 IT.121 - IT.240: 0.0050 IT.241 - IT.360: 5.0000E-04 IT.361 - IT.480: 5.0000E-05 IT.481 - IT.600: 5.0000E-06 IT.601 - IT.720: 5.0000E-07 IT.721 - IT.840: 5.0000E-08 IT.841 - IT.960: 5.0000E-09 IT.961 - IT.1080: 5.0000E-10 IT.1081 - IT.1200: 5.0000E-11 IT.1201 - IT.1320: 5.0000E-12 IT.1321 - IT.1440: 5.0000E-13 IT.1441 - IT.1560: 5.0000E-14 IT.1561 - IT.1680: 5.0000E-15 IT.1681 - IT.1800: 5.0000E-16

Tabla 4: resultados comparativos del primer y segundo train con el primer caso

fig 13: resultados del primer train (izda) y segundo train (dcha)

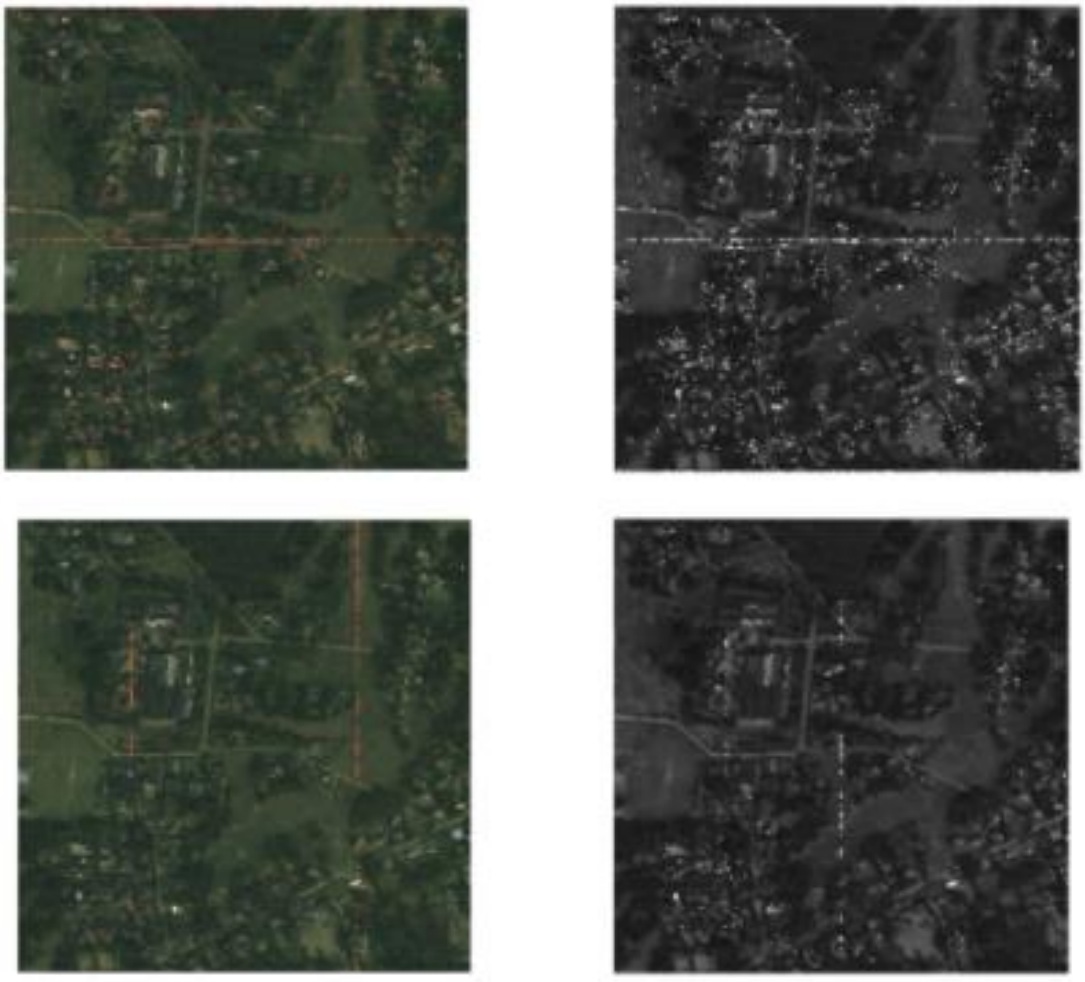

fig 14: Verdad del terreno obtenido para el primer caso en el primer entrenamiento (arriba) y segundo entrenamiento (abajo)

Segundo caso: Imagen urbana y su mascara

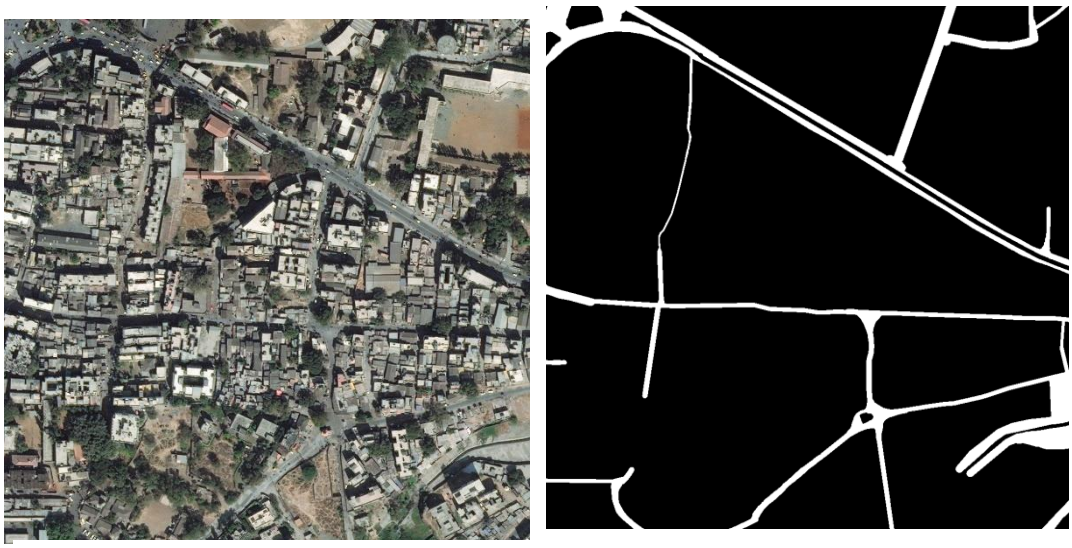

fig 15: Imagenes para el segundo caso de prueba

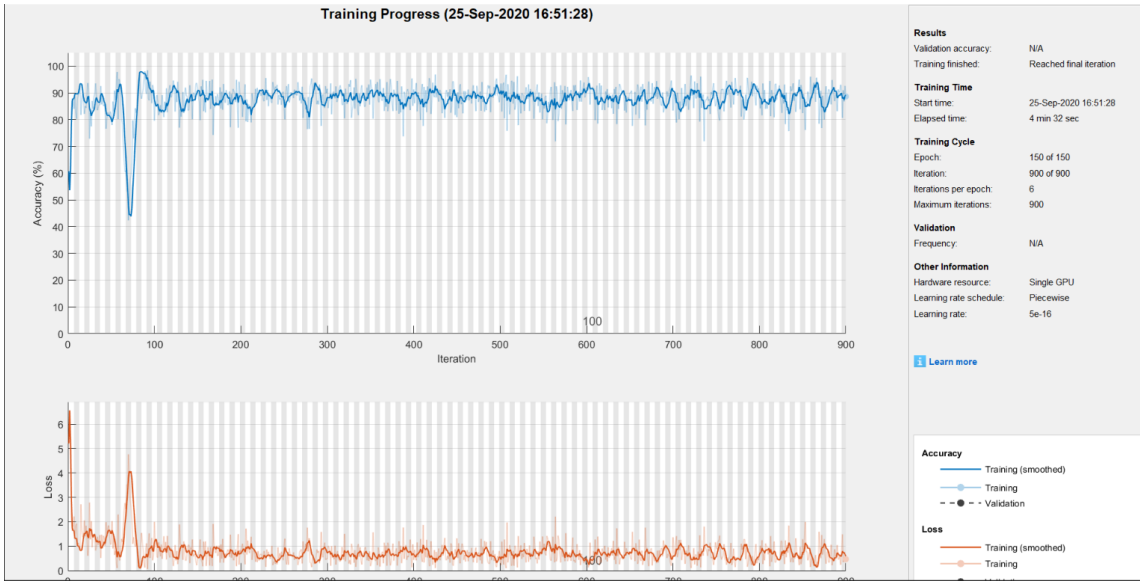

Fig16: Grafica de la evolución de los resultados del primer entrenamiento para el segundo caso de prueba

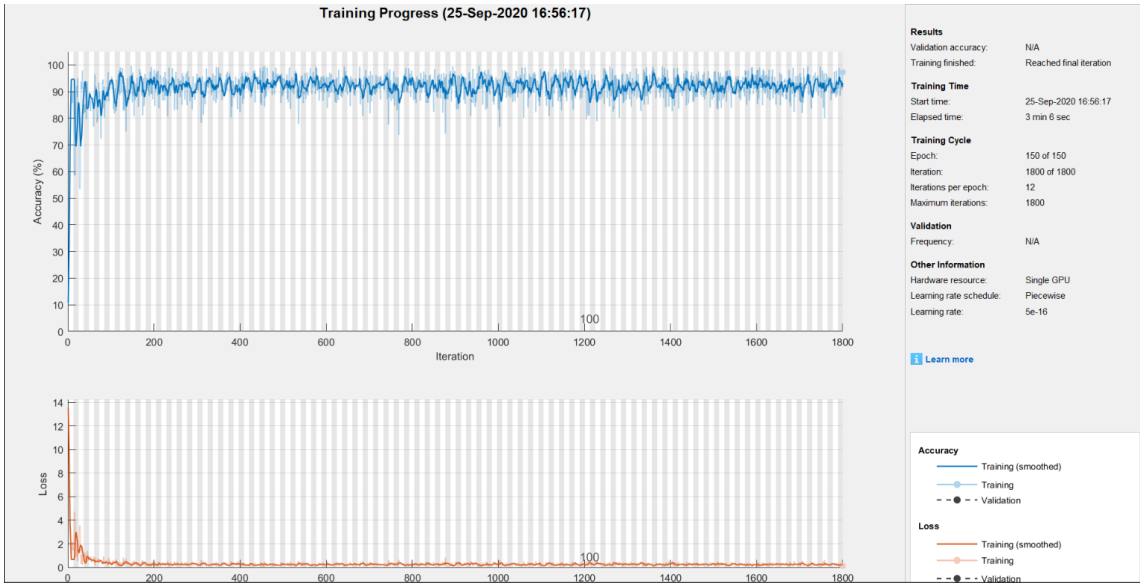

Fig17: Grafica de la evolución de los resultados del segundo entrenamiento para el segundo caso de prueba

INFORMACION DE LAS GRAFICAS	PRIMER TRAIN	SEGUNDO TRAIN
TAMAÑO DEL BATCH	16x16	32x32
ITERACIONES	900	1800
TRAINING LOSS	IT.1 - 1.8520 IT.450 - 0.1406 IT.900 - 0.143	IT.1 - 0.6054 IT.900 - 0.1274 IT.1800 - 0.1747
TRAINING ACCURACY	IT.1 - 88.2813 IT.450 - 96.9234 *IT.900 - 98.4257*	IT.1 - 92.3860 IT.900 - 96.4317 IT.1800 - 98.4539
BASE LEARN RATE	IT.1 - IT.120: 0.0050 IT.121 - IT.180: 5.0000E-04 IT.180 - IT.240: 5.0000E-05 IT.241 - IT.300: 5.0000E-06 IT.301 - IT.360: 5.0000E-07 IT.361 - IT.420: 5.0000E-08 IT.421 - IT.480: 5.0000E-09 IT.481 - IT.540: 5.0000E-10 IT.541 - IT.600: 5.0000E-11 IT.601 - IT.660: 5.0000E-12 IT.661 - IT.720: 5.0000E-13 IT.721 - IT.780: 5.0000E-14 IT.781 - IT.840: 5.0000E-15 IT.841 - IT.900: 5.0000E-16	IT.1 - IT.120: 0.0500 IT.121 - IT.240: 0.0050 IT.241 - IT.360: 5.0000E-04 IT.361 - IT.480: 5.0000E-05 IT.481 - IT.600: 5.0000E-06 IT.601 - IT.720: 5.0000E-07 IT.721 - IT.840: 5.0000E-08 IT.841 - IT.960: 5.0000E-09 IT.961 - IT.1080: 5.0000E-10 IT.1081 - IT.1200: 5.0000E-11

	IT.1201 - IT.1320: 5.0000E-12 IT.1321 - IT.1440: 5.0000E-13 IT.1441 - IT.1560: 5.0000E-14 IT.1561 - IT.1680: 5.0000E-15 IT.1681 - IT.1800: 5.0000E-16

Tabla 5: resultados comparativos del primer y segundo train con el segundo caso

fig 18: Resultado del primer entrenamiento (izq.) y segundo entrenamiento(dcha)

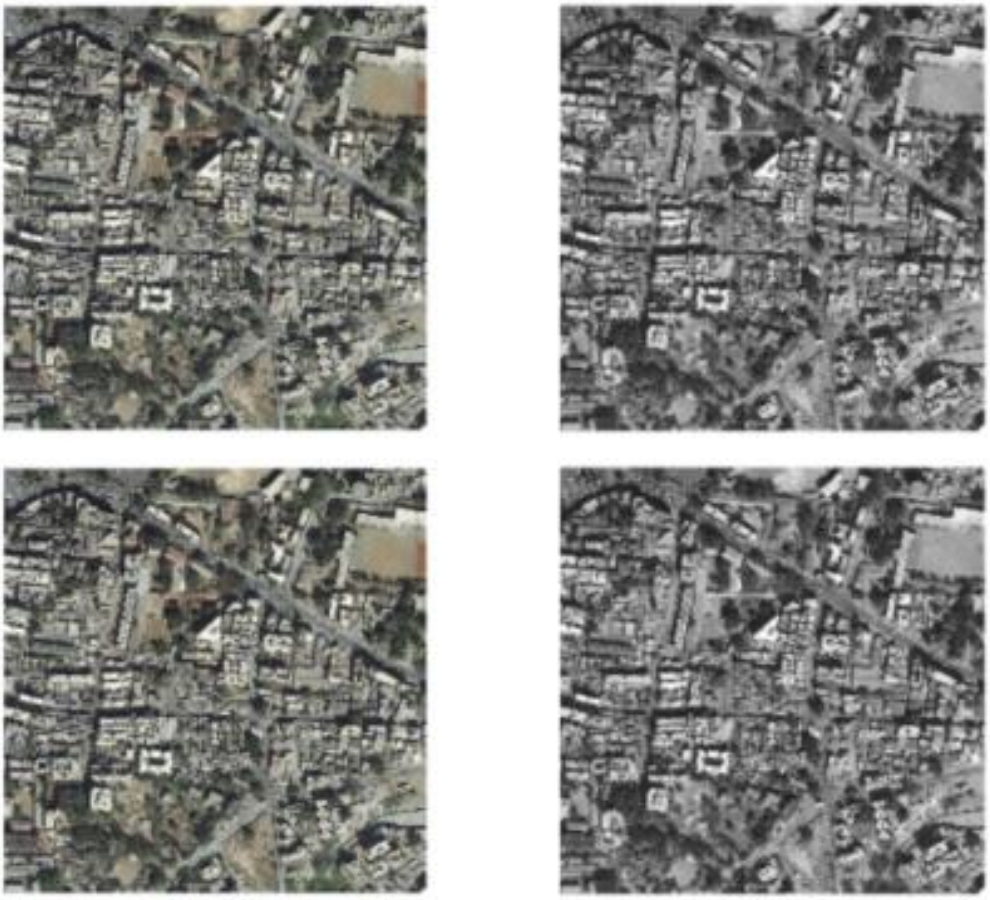

fig 19: Verdad del terreno para el primer entrenamiento(arriba) y segundo entrenamiento (abajo)

Como podemos observar los resultados no difieren de un entrenamiento a otro por limitaciones del hardware.

TIEMPOS DE EJECUCIÓN	PRIMER ENTRENAMIENTO	SEGUNDO ENTRENAMIENTO
PRIMER CASO	35 minutos	65 minutos
SEGUNDO CASO	35 minutos	65 minutos

Tabla 6: Comparativa de tiempos para ambos casos

5

Trabajos futuros y conclusiones

Breve resumen de los resultados obtenidos, experiencias y trabajos para futuros proyectos

5.1. Conclusiones

Ha sido una gran oprtunidad poder realizar este proyecto, tan interesante y agradezco una vez más a mi director el haberme guiado en este proyecto, en el que se han obtenido resultados muy satisfactorios.

5.2. Otros ámbitos de aplicación

Generalmente, las redes neuronales han generado grandes avances en la medicina, proporcionando tecnologías para la segmentación de imágenes médicas, que se pueden aplicar en resonancias para detectar pequeños tumores y distinguir las distintas partes del cuerpo. Pero, también es interesante remarcar que este tipo de redes se puede aplicar en series temporales, o cualquier otro ámbito en el que se trabaje con señales o se necesite predecir comportamientos.

5.3. Trabajos futuros

Ya que para alcanzar el objetivo principal se ha tenido que reducir al diseño de un algoritmo y una red neuronal funcional, por falta de capacidades de hardware y de espacio, sería muy interesante en continuar, como trabajo de master con este proyecto, ampliando los recursos necesarios para una rápida ejecución del algoritmo mejorando dicho algoritmo mediante enfoque distintos que mejoren los resultados obtenidos, y por su puesto ampliando el tamaño del batch de entrada, al igual que los casos de prueba

Bibliografía

[1] Poole, David. *Computational Intelligence: A Logical Approach* (en inglés). Nueva York: Oxford University Press.

[2] Russell, Stuart J.; Norvig, Peter Norvig (2009). Artificial intelligence: a modern approach (en inglés) (3.ª edición). Upper Saddle River, N.J.: Prentice Hall. ISBN 0-13-604259-7.

[3] <<Andreas Kaplan: Michael Haenlein (2019). Siri, Siri in my Hand, who's the Fairest in the Land? On the Interpretations, Illustrations and Implications of Artificial Intelligence, Business Horizons, 62(1), 15-25>>

[4] https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing << Artificial Neural Networks as Models of Neural Information Processing | Frontiers Research Topic >> (en ingles)

[5] https://es.wikipedia.org/wiki/Funci%C3%B3n_de_activaci%C3%B3n <<Funciones de activación>>

https://es.wikipedia.org/wiki/Redes_neuronales_convolucionales#Neuronas_de_Reducci%C3%B3n_de_Muestreo

[6] https://www.apd.es ¿Cuales son los tipos de algoritmos de machine learning?>>

[8] https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

[9] https://competitions.codalab.org/competitions/18467

[10] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (en inglés) <<Enlace al abstract y proyecto>>

Primer Training – Tamaño de baldosa: 16x16

```
imds = imageDatastore('2495_sat.jpg');
train_labels=imread('2495_mask.png');
train_labels(train_labels==0)=1;
train_labels(train_labels==255)=2;
train_labels=train_labels(:,:,1);
imwrite(train_labels,'train_labels.png');

classNames = ["noRoad","Road"];
pixelLabelIds = 1:2;
pxds = pixelLabelDatastore('train_labels.png',classNames,pixelLabelIds);

dsTrain = randomPatchExtractionDatastore(imds,pxds,[16,16],'PatchesPerImage',100);

inputTileSize = [16,16,3];
lgraph = createUnet1(inputTileSize);
disp(lgraph.Layers)

initialLearningRate = 0.05;
maxEpochs = 150;
minibatchSize = 16;
l2reg = 0.0001;

options = trainingOptions('sgdm',...
    'InitialLearnRate',initialLearningRate, ...
    'Momentum',0.9,...
    'L2Regularization',l2reg,...
    'MaxEpochs',maxEpochs,...
    'MiniBatchSize',minibatchSize,...
    'LearnRateSchedule','piecewise',...
    'Shuffle','every-epoch',...
    'GradientThresholdMethod','l2norm',...
    'GradientThreshold',0.05, ...
    'Plots','training-progress', ...
    'VerboseFrequency',20);

modelDateTime = datestr(now,'dd-mmm-yyyy-HH-MM-SS');
[net,info] = trainNetwork(dsTrain,lgraph,options);

save(['prueba-' modelDateTime '-Epoch-' num2str(maxEpochs) '.mat'],'net','options');

val_data=imread('2495_sat.jpg');
predictPatchSize = [256 256];
segmentedImage = segmentImage(val_data,net,predictPatchSize);
predicted_mask = segmentImage(val_data,net,predictPatchSize);
```

```
%
% predicted_mask(predicted_mask==1)=0;
% predicted_mask(predicted_mask==2)=255;
%

val_data(:,:,1)=val_data(:,:,1)+uint8((segmentedImage==2))*200;
subplot(1,2,1), imshow(val_data)
subplot(1,2,2), imshow(mask)
```

Segundo Training – Tamaño de baldosa 32x32

```
dsTrain2 = randomPatchExtractionDatastore(imds,pxds,[32,32],'PatchesPerImage',200);

inputTileSize2 = [32,32,3];
lgraph2 = createUnet1(inputTileSize2);
disp(lgraph2.Layers)

initialLearningRate = 0.05;
maxEpochs = 150;
minibatchSize = 16;
l2reg = 0.0001;

options = trainingOptions('sgdm',...
    'InitialLearnRate',initialLearningRate, ...
    'Momentum',0.9,...
    'L2Regularization',l2reg,...
    'MaxEpochs',maxEpochs,...
    'MiniBatchSize',minibatchSize,...
    'LearnRateSchedule','piecewise',...
    'Shuffle','every-epoch',...
    'GradientThresholdMethod','l2norm',...
    'GradientThreshold',0.05, ...
    'Plots','training-progress', ...
    'VerboseFrequency',20);

modelDateTime = datestr(now,'dd-mmm-yyyy-HH-MM-SS');
[net2,info2] = trainNetwork(dsTrain2,lgraph2,options);
save(['prueba2-' modelDateTime '-Epoch-' num2str(maxEpochs) '.mat'],'net','options');

val_data2=imread('2495_sat.jpg');
predictPatchSize2 = [256 256];
segmentedImage2 = segmentImage(val_data2,net2,predictPatchSize2);

%
% train_labels(train_labels==1)=0;
% train_labels(train_labels==2)=255;
%

val_data2(:,:,1)=val_data2(:,:,1)+uint8((segmentedImage2==2))*200;
imshow(val_data2)
```

Programa 1.1. Diseño del algoritmo usando redes neuronales convolucionales