Análisis de técnicas de segmentación semántica sobre imágenes aéreas con deeplabv3+

Rubén García Páez

Resumen–: En este trabajo se quiere analizar la tarea de la segmentación semántica sobre imágenes aéreas a partir del modelo de Deep Convolutional Network: Deeplabv3+. En este trabajo se quiere dar una visión general de los problemas principales de los modelos que se encargan de realizar tareas de segmentación semántica y los distintos problemas y desafíos que conlleva la clasificación de imágenes aéreas y como afrontarlas.
Palabras claves– Segmentación semántica, deep learning, imágenes aéreas, convolución en profundidad, atrous convolution, deeplabv3+.

Keywords– Semantic segmentation, deep learning, aerial images, depthwise convolution, atrous convolution, deeplabv3+

1 Introducción

Hoy en día el campo de la segmentación semántica es uno de los apartados de la visión por computación y Machine Learning (aprendizaje automático) donde más esfuerzo se está aplicando, tanto en imágenes RGB (2D), imágenes volumétricas, multiespectrales y secuencias de vídeo. En los últimos años el desarrollo de estos tipos de tecnología se ha visto impulsado en gran medida gracias al auge de los vehículos autónomos y a distintos tipos de procesos de automatización que requieren extraer información de lo que se esta viendo.

Existen distintos tipos de clasificación dentro de una imagen, tal como se puede observar en la figura 1. En este primer caso (figura 1a), lo que se esta haciendo es identificar los elementos que se encuentran dentro de la imagen y asignarle una etiqueta según las clases detectadas. El segundo ejemplo (figura 1b) consiste en l·localizarl·las clases detectadas dentro de la misma imagen, esta parte es más compleja que la anterior debido que no solo se tienen que detectar las clases sino que se tienen que localizar (de forma más o menos precisa) dentro de esta, señalándola en la imagen con cuadros u otros elementos. Por último estaría la tarea de la figura 1c, esta parte es similar a la anterior pero aún más difícil pues no basta con señalar con una cuadro el objeto detectado, sino que hay que localizar el objeto a nivel de pixel. Para mostrar estos resultados lo que se suele hacer es localizar el objeto a nivel de pixel y marcar estos para señalar que forman parte de la clase señalada.

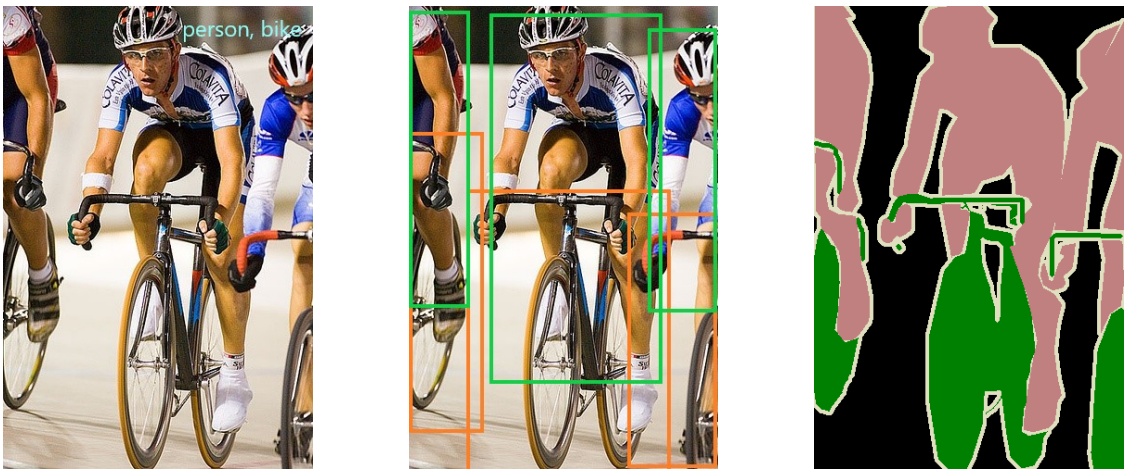
Pero en qué consiste exactamente la segmentación semántica? La segmentación semántica es una forma de clasificación donde se clasifican todos los pixeles de la imagen. A cada uno de los pixeles se le asigna una clase, el cual se codifica asignándole un color, formando de esta forma imágenes como la figura 1c.

Aparte de los tipos de imágenes mencionados anteriormente hoy en día se está popularizando también el uso de imágenes aéreas, debido a la gran cantidad de satélites que se encuentran en órbita realizando miles de fotos diarias. Y también a la accesibilidad que la gente esta teniendo a la hora de adquirir y pilotar drones.

La diferencia más relevante a la hora de diferenciar las imágenes aéreas y las imágenes tomadas desde tierra es que en éstas lo que se suele buscar es conseguir abarcar el máximo terreno posible con una sola imagen. Lo que conlleva que la información sea mucho más difícil de extraer debido al tamaño tan pequeño de los elementos de la imagen. Con tal de solucionar este problema lo que suele pasar es que se toman las imágenes con una resolución mucho más alta de lo normal o con cámaras multi-espectrales,capaces de captar información adicional como podrían ser la temperatura, profundidad, entre muchas otras. Estas son solo algunas de las posibles técnicas que se podrían emplear para suplir el problema descrito.

En este trabajo se quiere estudiar, empleando el modelo de Deep Fully Convolutiona Neural Network (FCNN) Deeplabv3+, como se ve afectado el rendimiento del modelo en función de la resolución empleada para el entrenamiento y evaluación de imágenes aéreas.

• E-mail de contacto: ruben.garciapae@e-campus.uab.cat
• Mención realizada: Enginyeria de Computació
• Trabajo tutorizado por: Robert Benavente(Ciencias de la computación)
• Curso 2018/19



(a) Reconocimiento (b) Detección de objetos (c) Segmentación semántica

Fig. 1: Distinción de diferentes tipos de técnicas para reconocimiento de elementos en imágenes, ejemplo sacado del dataset [1], del challenge del 2012

2 OBJETIVOS

El objetivo principal de este trabajo es estudiar un modelo de red neuronal que emplee deep learning y realizar con él distintas pruebas sobre imágenes aéreas para comprobar como se comportan a distintas resoluciones.

Para poder llevar a cabo este objetivo se han dictaminado distintos subobjetivos con tal de poder llevar a cabo nuestro objetivo principal:

- **Recolección de información:** Estudiar las distintas técnicas y modelos de deep FCNN que más se utilizan y mayor rendimiento tienen y escoger un modelo con el que realizar los experimentos en función al resultado que este tenga en benchmarks como [2].
- **Datasets:** Escoger los datasets sobre los cuales realizaremos nuestros experimentos y pruebas. Tal y como indica el título del proyecto algunos de estos datasets serán datasets de imágenes aéreas.
- **Comparación:** Comparar los distintos experimentos realizados en función de una métrica estandarizada sobre la cual evaluar los resultados obtenidos.

El título fue modificado en acorde una vez se encontró el modelo con el cual se decidió realizar los experimentos y como se definieron.

3 METODOLOGIA

El planteamiento del proyecto estará enfocado en la utilización progresiva de los frameworks y librerías relacionados con el deep learning.

Para poder lograr este objetivo se ha planteado una metodología de trabajo incremental que constará de la creación de un núcleo de trabajo a el cual irá aumentando a medida que el proyecto vaya progresando.

El planteamiento es simple. Se empezará buscando información tanto de las herramientas a utilizar como de como funcionan estas. En este caso el funcionamiento de como realizar entrenamientos con redes neuronales, su funcionamiento, las técnicas que emplean etc. En esta primera fase también entraría la familiarización de las herramientas utilizadas como Tensorflow. A partir de aquí se esperan realizar experimentos con datasets que ya vienen casi preparados y una vez se haya cogido familiaridad con el código y su funcionamiento, realizar los experimentos sobre los datasets aéreos.

Se ha decidido optar por este tipo de metodología porque de esta forma no nos quedaremos en un punto muerto debido a la posible complejidad que pueda surgir durante el desarrollo. De esta forma se podrá empezar a trabajar y se tendrán resultados desde el principio del proyecto, los cuales irán aumentando tanto en cantidad como en complejidad a medida que el proyecto va avanzando.

3.1 Herramientas a utilizar

Para este proyecto se utilizarán sobretodo modelos que estén implementados en Tensorflow yKeras en Python 2/3. Tensorflow es una librería de código abierto desarrollada por Google Brain y diseñada para el desarrollo de Machine Learning (ML). Keras es una librería que actúa como una capa de abstracción entre el usuario y el backend que puede estar ejecutándose sobre Tensorflow u otras librerías relacionadas con ML

El motivo principal de porque se ha escogido como lenguaje Python es por dos factores: Es más simple que C++ y soporta tanto Tensorflow como keras, aparte de soportar la gran mayoría de frameworks como se puede ver en la figura 2.

El hecho de utilizar Tensorflow también ha sido por diversos motivos. El más importante y principal siendo la gran cantidad de soporte y comunidad que tiene detrás tanto en Stackoverflow como en Github. Además también es el framework utilizado para el desarrollo de Deeplabv3+ [12], el modelo que se va a utilizar en este trabajo.

Deep Learning Framework	Release Year	Written in which language?	CUDA supported?	Does it have pretrained models?
TensorFlow	2015	C++, Python	Yes	Yes
Keras	2015	Python	Yes	Yes
PyTorch	2016	Python, C	Yes	Yes
Caffe	2013	C++	Yes	Yes
Deeplearning4j	2014	C++, Java	Yes	Yes

Fig. 2: Tabla comparativa sobre los distintos frameworks más usados. Extraído de [3]

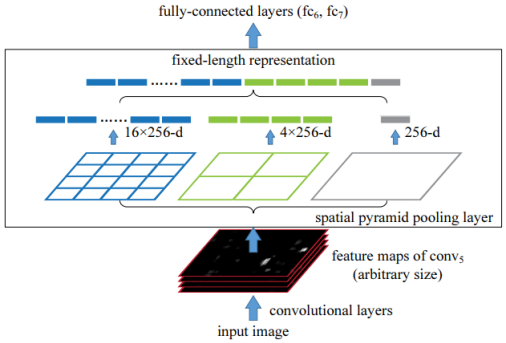


Fig. 3: Esquema del spatial pyramid pooling layer. 256 es el filtro de la última convolutional layer. Como el vector resultante de esta capa es un número fijo lo podemos pasar a las fully connected para que puedan clasificar. Extraído de [4]

4 Planificación

Para la planificación del proyecto se ha hecho un diagrama de Gantt que se puede encontrar en el apéndice en la figura 9.

La planificación se ha basado principalmente en los límites de entrega establecidos por la asignatura: dos informes de seguimiento seguido de un informe final y la presentación para la defensa de este.

La planificación es la siguiente:

1. La primera fase para la entrega del primer informe de seguimiento se hará una revisión a los distintos artículos relacionados con la segmentación semántica. Para la entrega del primer informe se busca tener ya entrenada y con resultados algún dataset con el modelo escogido.

2. Para la segunda entrega la idea es tener más resultados, incluyendo algún dataset de imágenes aéreas. Esta entrega debería incluir ciertos resultados y conclusiones.

3. Para la entrega final, se debería tener todo el trabajo listo y revisado con tal de poder centrarse en la defensa del trabajo.

5 Estado del arte

Hoy en día el mundo de la visión por computador ha sufrido una revolución causada principalmente por los avances en técnicas de machine learning. De este tipo de técnicas las que más han hecho que destaque son las redes neuronales. Hoy en día trabajos que antiguamente requerían una gran abanico de conocimiento y técnicas muy depuradas son fácilmente asequibles gracias a la utilización de las redes neuronales.

Actualmente las estructuras más utilizadas para la tarea de segmentación semántica son las *Deep Fully Convolutional Neural Network* (CNNs) que utilizan módulos de (*Spatial pyramid pooling*) (SPP) [4] o estructuras encoder-decoder [5].

5.1 Modelos y estructuras populares

Actualmente, las arquitecturas de redes neuronales más populares son las Deep FCNN (*Fully Convolutional Neural Network*). Pero el mayor problema que tienen este tipo de redes es que suelen ser muy susceptibles a los cambios de resolución de las imágenes y es bastante difícil que aprovechen información contextual de la imagen.

Los cambios a resolución vienen dados a que, durante el entrenamiento se suele entrenar la red con imágenes con resoluciones fijas o muy parecidas entre ellas. Esto, junto al hecho de que la entrada a las capas Fully connected son vectores fijos, hace que la red únicamente sea capaz de aprender a detectar los objetos cuando estos tienen la resolución y aspect-ratio que tenían durante su entrenamiento.

La parte de que no pueden aprovechar la información contextual es debido a que cada vez que la imagen pasa por una capa de convolución, las salidas van siendo cada vez más y más pequeñas, perdiendo detalle en el proceso. Con tal de aumentar el detalle habría que aumentar el tamaño de los filtros de las capas de convolución, cosa que aumentaría tanto el cómputo de la red como su tamaño en memoria, cosa contraproducente dado el limitado tamaño de las GPUs con las que se suelen ejecutar.

Para tal de solucionar los dos problemas mentados se han desarrollado distintas técnicas y estructuras que ayudan a mitigar parte de los problemas de estas. A continuación expondré cuales de estas técnicas son las más usadas y las que más relación tienen con este trabajo:

Capas Spatial pyramid pooling (SPP) [4]: Esta es una solución para aumentar la robustez de la red frente a imágenes de distintas resoluciones. Antes de seguir dejar claro que las SPP no son un tipo de arquitectura o modelo, es un tipo de capa (o layer).

La idea principal de este tipo de capa es crear un modelo que sea capaz de detectar los mismos objetos sin importar el tamaño que estos tengan.

Con tal de conseguir este tipo de robustez se emplean este tipo de capas (SPP) justo después de la última capa de convolución y justo antes de las capas Fully connected. Lo que hace este tipo de layer es codificar la información a diferentes escalas al mismo tiempo, tal como se puede ver en la figura 3. Esto crea vectores más grandes y es un extra de cómputo pero gracias a eso se gana bastante robustez en el modelo.

Encoder-Decoder [5]: Las estructuras encoder-decoder es un tipo de estructura que consiste en 2 partes:

- **Encoder:** Esta primera parte del modelo se encarga de comprimir (downsample) los mapas de características con tal de replicar la misma información que en el es-

tado original pero de forma comprimida.

- **Decoder:** Esta segunda parte utiliza la información comprimida por el encoder y se encarga de ”restaurarl-la información comprimida por en encoder (upsample).

Es un tipo de estructura bastante eficaz debido a que permite solucionar otro de los problemas que tienen las FCNN: Permiten añadir información contextual. Cada vez que se realiza downsample parte de esa información se pasará después a su contra-parte del decoder. Gracias a esto conseguimos recuperar parte de la información ”original”de la salida de las capas de convolución sin tener que usar filtros de mayor tamaño.

Este tipo de estructura se ha popularizado bastante y hay varios modelos distintos muy populares que lo utilizan como podría ser la U-net [6], un tipo de CNN para imágenes médicas, SegNet [5] y VGG-16 [7].

5.2 Datasets con los que se ha trabajado

A continuación se pasará a a explicar los datasets empleados y sus características principales:

- **PASCAL VOC 12** [1]: El dataset del reto PASCAL VOC (Pattern Analysis Statistical Modelling and Computional Learning Visual Object Classes es uno de los datasets más comunes y usados en toda la comunidad. Consta de unas 10000 imágenes y 21 clases distintas entre las cuales encontramos animales, vehículos y objetos cotidianos entre otros. Es un dataset bastante ”simple”dado que en las imágenes suelen aparecer no muchas clases distintas y que las máscaras son bastante grandes y distinguibles entre si.

- **ADE20K** [8]: Dataset utilizado por el MIT como dataset de diferentes tipos de imágenes. Desde escenarios exteriores como interiores. Este dataset cuenta con alrededor de 20000 imágenes para entrenamiento y esta formado por 151 clases distintas, además es mucho más ”denso”que algunos datasets (como el PASCAL) donde suele haber pocos elementos en la imagen. En el caso de ADE20K en cada imagen pueden llevar a aparecer un gran número de clases distintas, y siendo el dataset uno sobre imágenes de interior y exterior lo convierte en un dataset donde es bastante difícil obtener buenos resultados.

- **cityscapes** [9]: Este dataset se centra en el entrenamiento de imágenes urbanas por carretera. Las imágenes que forman el dataset rondan casi los 3000 y estas se clasifican en 30 clases distintas divididas en 8 grupos como podrían ser naturaleza, el cielo, vehículos, humanos, etc. Las imágenes forman parte de diversos vídeos de hasta 50 ciudades distintas capturadas durante horas de sol con buenas condiciones climáticas. Al ser originalmente un vídeo se han escogido frames claves para que haya bastante cantidad de clases por imagen, fondos diversos y objetos variantes.

- **ISPRS** [10]: Dataset de imágenes aéreas de la ciudad alemana de Potsdam. El dataset forma parte de una competición organizado por International Society for Photogrammetry and Remote Sensing Semantic LabelinG (ISPRS). Este dataset esta formado por imágenes RGB de muy alta definición (6000X6000) en formato tif con 6 clases diferenciadas. La organización ISPRS también tiene diversos datasets con imágenes 3D y multiespectrales. La gran difucultad de la imagen es que solo hay alrededor de unas 30-40 imágenes.

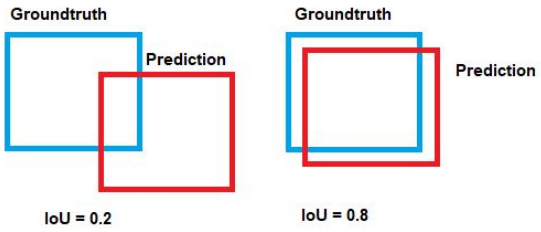


Fig. 4: Ejemplo de evaluación el IoU. Cuanto más se sobreponga correctamente una predicción con la máscara, mayor será el IoU

- **Inria**[11]: Dataset utilizado para entrenar modelos para la detección de edificios en base a imágenes aéreas de diferentes ciudades del mundo como Chicago o Vienna. Este dataset esta pensado para detectar edificios por lo que solo tiene 2 clases: edificio y background. Este dataset esta formado por alrededor de 180 imágenes de una resolución de 5000x5000.

5.3 Métricas de evaluación

A continuación se explicara que métricas más usadas y las que se usarán para evaluar los distintos experimentos que se han realizado. Hay que tener en cuenta que que es importante que se apliquen los mismos parámetros en la evaluación que durante la fase de entrenamiento ya que sino las comparaciones no serian justas.

Una de las métricas más sencillas es simplemente realizar un porcentaje de exactitud (accuracy) en base a cuantos pixeles se han clasificado correctamente.

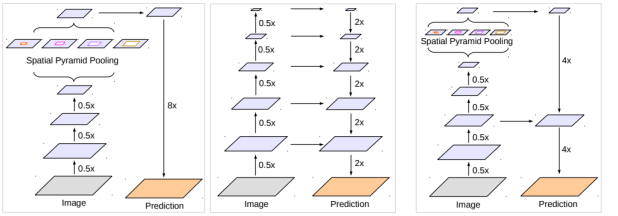
Pixel Accuracy (PA): Se trata de conseguir un porcentaje de la cantidad de pixeles que se han clasificado de forma correcta sobre el número de pixeles total.

$$PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k}\sum_{j=0}^{k} p_{ij}}$$

Donde k es el número de clases del dataset y Pij hace referencia al pixel de la posición [i][j] de la matriz de la imagen.

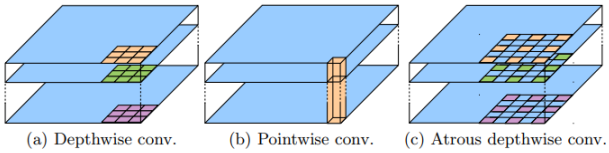
Pese a que existen distintos tipo de métricas a la hora de evaluar el resultado de nuestro modelo, la que se va a utilizar para comparar los distintos resultados es la métrica estándar en cuanto a segmentación se refiere: **Mean Intersection over Union (MIoU)**. Consiste en computar un ratio entre las intersecciones i uniones de dos sets. En este caso esto dos sets serán el groundtruth y el otro será la imagen predicha por la red, tal y como se puede ver la figura 4. La métrica consiste en: intersección de true positives entre la suma de true positives, falsos negativos y falsos positivos

$$MIoU = \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{i=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

Fig. 5: El modelo anterior deeplabv3 que utilizaba un módulo SPP (a) ha sido mejorado con la estructura encoder-decoder (b) Resultando en el esquema final (c). Gracias a esto podemos seguir obteniendo un modelo resistente a cambios de resolución y una forma de extraer características con más detalle. Extraído de [12].



(a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv.

Fig. 6: Ejemplo de una convolución 3x3. En la subfigura (a) se esta aplicando una depthwise convolution, un solo filtro por canal y en (b) se esta aplicando pointwise convolution, filtro 1x1 aplicado a todos los canales. En cambio en (c) se esta aplicando atrous depthwise convolution, una combinación del atrous convolution con la convolución en profundidad. En este caso el *rate = 2* (Atrous convolution con *rate = 1* es una convolución normal). Extraído de [12].

k es el número de clase i Pij hace referencia al valor del pixel en la posición i,j de la imagen.

6 DESARROLLO

En este apartado se pretende entrar un poco más en profundidad sobre la arquitectura utilizada, detalles sobre el entrenamiento sobre las distintos datasets, características sobre estos y en concreto sobre las imágenes aéreas entre otros temas.

6.1 Deeplabv3+

Deeplabv3+ [12] es una deep FCNN para tareas de segmentación semántica. Esta nueva versión es una mejora respecto a su antiguo predecesor Deeplabv3 [13].

La razón principal por la que se ha escogido este modelo es debido a que es uno de modelos que mejores resultados suele obtener en benchmarks [2], [14]. Y otro y el más importante, tiene su implementación en tensorflow en Github pública para cualquiera que quiera usarla.

6.1.1 Deeplabv3 y Deeplabv3+:

El modelo Deeplabv3 es un modelo que emplea un layer SPP en su última capa con tal de dotar de robustez al modelo. Pese que gracias a la SPP conseguía robustez a cambios de tamaño, esta sigue sufriendo el otro gran problema de las FCNN, es difícil conseguir información contextual para conseguir características detalladas.

Para poder solucionar ese problema se ha creado el nuevo modelo Deeplabv3+, el cual implementa una estructura encoder-decoder y utiliza el anterior Deeplabv3 como módulo encoder. Además se ha modificado la capa SPP, ahora llamada ASPP (Atrous SPP), para que emplee atrous convulation, una técnica de convolución que se explicará más adelante. De esta forma nos quedaría una estructura como la que se puede ver en la figura 7.

6.1.2 Métodos utilizados

En esta sección se describirá los distintos métodos de convoluciones y técnicas que se utiliza en este nuevo modelo.

- **Atrous Spatial Pyramid Pooling ASPP:** Atrous convolution es una técnica que permite controlar las resoluciones con las que se computan las imágenes en la red neuronal. Este tipo de convoluciones permiten calcular las propiedades de la imagen a diferentes resoluciones mediante un ratio *r* que determina el grado de dilatación (véase la figura 6(c)) con el que se quiere operar.

- **Depthwise separable convolution:** Un tipo de convolución en dos fases que permite substituir las técnicas de convolución convencionales con tal de aumentar el rendimiento computacional de estas.

  Este tipo de convolución se divide en 2 partes:

  1. **Convolución en profundidad (depthwise convolution), fase de filtrado:** Esta fase es muy similar a la convolución convencional, pero, en lugar de aplicar los kernels a todos los canales a la vez, lo que se hace es aplicarlos a cada uno de los canales por separado.
  2. **Convolución puntual (pointwise convolution):** En este caso la convolución se hace con kernels de 1x1 a lo largo de todos los canales

  Combinando estas dos técnicas (depthwise convolution y pointwise convolution) podemos convolucionar con muchas menos operaciones que de forma tradicional. De modo que no hay que aplicar tantas operaciones en todos los canales, reduciendo la complejidad y el tamaño de los modelos. Gracias a estas técnicas conseguimos suplir los costes extras que suponen realizar computar varias resoluciones al mismo tiempo.

  En la figura 6 se puede ver un ejemplo de los 3 tipos de convoluciones que se utilizan.

6.1.3 Arquitectura

La filosofía de la arquitectura principal se puede ver en la figura 7(c). Se utiliza Deeplabv3 como encoder y se la añade un nuevo apartado a modo de decoder que permite realizar upsample a las características de forma más precisa.

Deeplabv3+ tiene dos modelos que emplea de extractor de características (encoder): El ResNet-101 [15] o Aligned Xception [16], esta última variante es la que se suele usar principalmente debido a que parece rendir ligeramente mejor que la modificación de la ResNet [12]).

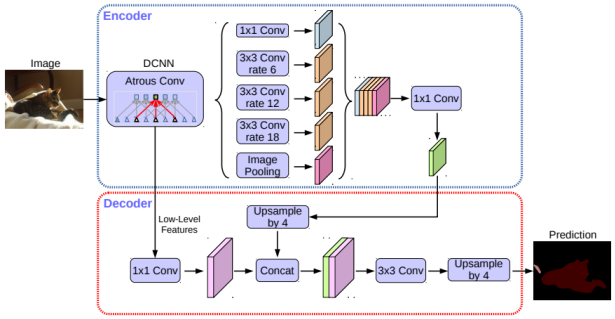


Fig. 7: Esquema de como es la estructura Deeplabv3+ con la estructura encoder-decoder

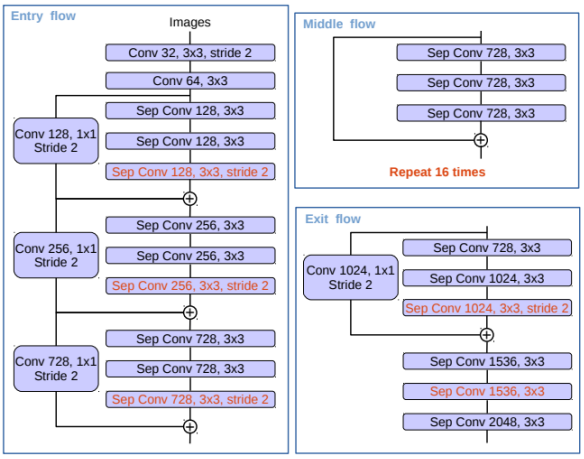


Fig. 8: Estructura Deeplabv3+. Estructura modificada tal como se ha descrito a partir de la arquitectura Xception [16]. Extraído de [12].

El modelo utiliza la estructura del Xception Aligned [16] pero con las siguientes modificaciones tal y como se puede ver en la figura 8.

- Aumentada la profundidad (más capas).
- Las operaciones de max pooling se han reemplazado por operaciones depthwise separable convolution con stride 2.
- Después de cada 3 x 3 depthwise convolution se ha añadido una capa extra de normalización y activación por ReLU

Esta parte sería la del encoder, al final del encoder tendríamos la del decoder que se encargará de realizar upsampling de la imagen con información que obtiene del encoder.

Decoder: El decoder esta basado en un output stride de 16. Se realiza un upsample por 4 de la salida del encoder. A esta imagen se le concatenarán partes de la información que se obtuvo cuando la imagen estaba con esa misma resolución.

A la información que se le concatenará a la imagen del encoder se la pasa por una convolución de 1x1 con tal de reducir la dimensión de esta.

Después de haber concatenado la información a la imagen de upsample 4 se le aplica una convolución de 3x3 y se vuelve a pasar un upsample de 4 con tal de conseguir la imagen con el mismo tamaño de la imagen de entrada.

Se puede ver el esquema en la figura 7

6.2 Entrenando con deeplab

A continuación se explicará el proceso que se ha seguido para poder llegar a realizar los experimentos usando la arquitectura deeplabv3+ [12].

Por si no estaba claro, aclarar que la segmentación semántica es un tipo de clasificación que se engloba dentro del concepto de aprendizaje supervisado. Este tipo de aprendizaje se basa en conocer como se debería clasificar la imagen a partir de imágenes que actúen como ”groundtruth”, imágenes preparadas de antemano que muestran como deberían clasificarse las imágenes.

6.2.1 Detalles iniciales

Con tal de poder realizar los experimentos necesitaremos 3 elementos básicos:

- Las imágenes que queremos clasificar.
- Las imágenes que se usaran como groundtruth y contienen las máscaras para clasificar la imagen correctamente.
- las diferentes particiones que haremos para entrenar (train, val, trainval, ect.).

Antes de poder empezar a entrenar nuestro modelo hay que especificar las características del dataset al modelo con tal de que poder realizar los cálculos correctos. Hay que especificar:

- El número de imágenes que se van a usar en cada una de las distintas particiones.
- El número de clases que el dataset tiene.
- El ignore_label. El valor de los pixeles que la red ignorará. Por ejemplo, si ponemos ignore_label=255 la red no tendrá en cuenta durante el entrenamiento los pixeles que sean blancos.

6.2.2 Preparando los datasets

El entrenamiento de los datasets que no eran de imágenes aéreas no tuvo ningún problema demasiado grande. Los 3 datasets (PASCAl_VOC, cityscapes y ade20k) venían con un número bastante considerable de imágenes y estas venían ya muy preparadas (perfiles similares, distribución de clases equivalente, etc.)

En cambio, en cuanto a los datasets de imágenes aéreas la cosa se volvió bastante más complicada. Para empezar no había demasiado datasets de imágenes aéreas que cumplieran con las características que necesitaba, es decir imágenes naturales y sus imágenes de groundtruth con las máscaras.

Para los datasets de imágenes aéreas (ISPRS y Inria) ha habido que retocar bastante las imágenes.

Para empezar, las imágenes de los datasets venían todas con una muy alta resolución (6000x6000 i 5000x5000) en

formato tif. Esto es bastante comprensible dado que es necesaria una alta resolución de las imágenes para poder conseguir buenos resultados en este tipo de imágenes. Pero dada la memoria limitada de los dispositivos con los que se entrena se ha tenido que reducir bastante la resolución de estas.

Con tal de tener imágenes con las que poder entrenar se han creado 3 tipos de datasets distintos, cada uno con una resolución inferior a la original. Los 3 "nuevos"datasets son el mismo dataset original pero con la resolución y el tipo de imagen distinto. Las 3 resoluciones han sido 1200x1200, 800x800 y 500x500. La decisión del máximo de resolución ha sido debido a la memoria del dispositivo donde se han ejecutado los entrenamientos.

Particiones: Como casi todos los datasets que se han utilizado pertenecen a concursos o distintos tipos de retos, no se ha podido acceder a la particion usada para los tests. Debido a ello se ha empleado la partición de validación para las pruebas.

El problema vuelve con las imágenes aéreas. El gran problema es que en los datasets vienen bastante pocas imágenes. En el dataset Inria tenemos unas 180 imágenes. En este caso solo tenemos la partición de trainning por lo que se ha partido en una relación más o menos 70/30, 70% trainning y el resto de test.

Para el de ISPRS el problema es aun mayor, tenemos muchas menos imágenes (23 usables) y el dataset es multiclase. En este caso se ha decidido realizar algo de data augmentation, repetir las mismas imágenes pero girando las imágenes 90 grados en los 3 sentidos restantes (rotation). Con tal de aumentar un poco el número de imágenes con las que entrenar.

6.2.3 Durante los entrenamientos

En este apartado explicaremos el procedimiento que se siguió para poder realizar con éxito los distintos entrenamientos.

Empezando con los datasets no aéreos (pascal, cityscapes y ADE20K). Decidí entrenar con estos datasets a modo de prueba y para poder familiarizarse con el funcionamiento del modelo y todo el código. Los entrenamientos no dieron demasiados problemas excepto un fallo al principio del trabajo donde hubo problemas a la hora de evaluar el modelo.

Fue cuando se empezó con el entrenamiento de los datasets aéreos que hubo que cambiar y configurar ciertos elementos.

Lo que se describe en el siguiente parágrafo ocurrió para ambos datasets de imágenes aéreas, INRIA y ISPRS.

Para empezar, se intentó empezar el entrenamiento sin utilizar ningún tipo de modelo pre-entrenado. Al intentar entrenar con este método el modelo parecía no converger en ningún momento, y aun modificando parámetros como el learning_rate y el momentum (entre otros)la función de pérdida o no bajaba u oscilaba entre valores casi aleatorios. Al investigar un poco porque la función de pérdida no parecía converger como debería descubrí el problema. Dado las pocas imágenes de las que se disponen para ambos datasets y el tiempo limitado que se le daba a cada entrenamiento (un entrenamiento de 10000-20000 steps tardaba muchísimo, parecía darse un caso de underfitting, donde todas las imágenes se clasificaban en su totalidad como "background". Es decir, la red creaba una máscara con la clase "background"que ocupaba toda la imagen. Al parecer, al clasificar toda la imagen como background se obtenía un mIOU de un 44%.

Datasets	Network backbone	Pretrained dataset
pascal_voc	Xception_65	MS-COCO VOC 12 train_aug + trainval sets
cityscapes	Xception_65	ImageNet, Cityscapes train_fine set
ADE20K	Xception_65	ImageNet, ADE20K trainning set

TABLA 1: DETALLES DE LOS DISTINTOS MODELOS CONGELADOS YA PRE-ENTRENADOS QUE SE HAN UTILIZADO PARA REALIZAR LOS EXPERIMENTOS DE LA TABLA 3

Dado ese motivo se descartó la idea de realizar los entrenamientos desde cero y se utilizaron los modelos preentrenados que había disponible en el github del mismo código.

7 RESULTADOS

Los resultados se dividirán en 2 partes: Un análisis rápido de los datasets que no son de imágenes aéreas y un análisis más exhaustivo sobre los datasets de imágenes aéreas.

7.1 Datasets convencionales (imágenes no aéreas)

Estos datasets son de los más comunes y utilizados

En la figura 3 se pueden observar los distintos resultados que se han obtenido con la Deeplabv3+ (Xception). Para estos experimentos se realizado fine-tunning gracias a modelos ya pre entrenados. Estos modelos han sido entrenados con imágenes de ImageNet [17] o MS COCO [18]. Todos estos resultados han sido ejecutados con una configuración muy similar para que los resultados sean justos y comparables. Esta configuración es la que en un princpio venía recomendada para entrenar el dataset pascal_voc, y dado la naturaleza de los hiper-parámetros se ha decidido emplearla en los otros dos datasets, véase la tabla 2. Aparte de esos parámetros los rates usados para las atrous convolutions del encoder (referencia al apartado 6.1.2) han sido [6, 12, 18] con output_stride = 16.

Aclarar que la utilización de distintos atrous rates va enfocada a la robustez del modelo a distintas resoluciones (objetos más grandes o más pequeños de lo habitual) y no tanto al resultado final de esta. Se han probado atrous rates distintos, como [12, 24, 36] con output_stride = 8 y los resultados han sido peores en todos los datasets (alrededor de un 0-5% peores).

Para el entrenamiento de estos datasets se han utilizado modelos pre-entrenados que emplean la estructura de Deeplabv3+ Xception (Aligned Xception modificado + módulo decoder)

Los resultados se puede ver en la tabla 3.

El dataset pascal suele tener buenas puntuaciones dada la naturaleza de este. Suelen ser imágenes donde el objeto a

Trainning parameters	steps	base learning rate	batch size	weight decay	momentum
ADE20K	20000	0.00015	4	0.00004	0.9
pascal_voc	20000	0.0001	4	0.00004	0.9
cityscapes	20000	0.0001	2	0.00004	0.9

TABLA 2: PARÁMETROS CON LOS QUE SE HAN EJECUTADO LOS ENTRENAMIENTOS USANDO LA VARIANTE XCEPTION. DADO EL TIEMPO QUE SE TARDABA EN REALIZAR EL ENTRENAMIENTO SE DECIDIÓ QUE SE ENTRENARÍAN TODOS LOS DATASETS CON 20000 STEPS.

mIOU	PASCAL_VOC	Cityscapes	ADE20K
Deeplabv3+	87.11%	76.49%	55.48%

TABLA 3: LOS RESULTADOS OBTENIDOS HAN SIDO ENTRENAMIENTOS EN LOS QUE SE HAN USADO CIERTOS GRAFOS PRE-ENTRENADOS SOBRE EL DATASET COCO DE MICROSOFT I/O IMAGENET

mIOU	500x500	800x800	1200x1200
INRIA	50.71%	68.40%	72.17%

TABLA 4: RESULTADOS OBTENIDOS DEL ENTRENAMIENTO DEL DATASET INRIA. SE PUEDE VER A PARTIR DE LOS RESULTADOS OBTENIDOS COMO EL HECHO DE ENTRENAR COM IMÁGENES CON MÁS RESOLUCIÓN LLEVA A MEJORES RESULTADOS.

mIOU	500x500	800x800	1200x1200
ISPRS	47.43%	58.61%	69.78%

TABLA 5: RESULTADOS DEL ENTRENAMIENTO DEL DATASET ISPRS.

clasificar suele estar bien marcado y donde suele haber muy pocas clases (incluyendo el fondo, el cual suele contarse como una imagen), cosa que lo convierte en un dataset donde es posible obtener muy buenos resultados.

Sobre cityscapes, este resultado es debido a 2 factores, el hecho de ser un dataset sobre imágenes outdoor suele ser más difícil que las imágenes de interiores debido a la propia naturaleza de estas (elementos no predecibles, máscaras irregulares, condiciones climáticas, etc.). Además cityscapes es mucho más "densa"que la pascal, con esto nos referimos a que hay más cantidad de elementos (clases) en cada una de las imágenes que de costumbre.

Pese que el resultado de ADE20K sea el más pobre de todos, tiene sentido si examinamos el tipo de dataset que es. ADE20K tiene 151 clases, un número considerable de clases si lo comparamos con las 8 de pascal o las 30 de cityscapes. Aparte del gran número de clases, este dataset esta formado tanto por imágenes de interior como de exterior. Y las imágenes suelen ser igual o más densas que el dataset cityscapes. La combinación de todos estos elementos lo convierte en un dataset donde es muy difícil obtener buenos resultados. Pese que se estaba realizando fine-tunning sobre el modelo ya entrenado, el hecho de ser un dataset tan poco concreto y extenso dificulta la tarea de clasificación dada la gran disparidad de datos que contiene.

7.2 Datasets aéreos

En cuanto a las imágenes aéreas se ha jugado con las diferentes resoluciones a la hora de entrenar i evaluar estas. Las imágenes originales venían en un formato sin compresión (tif) y tenían una resolución demasiado grande para poder trabajar con ellas (debido principalmente a la falta de memoria de las GPUs del servidor con el que se ha trabajado).

Para poder trabajar con las imágenes se ha cambiado el formato de estas y reducido la resolución para adaptarlas y poder ser capaces de entrenar la red. Pese que el caso idóneo sería poder utilizar las imágenes tal i como vienen el hecho de que sean tan grandes es una característica que no nos permite utilizarlas dados los recursos disponibles.

Antes de nada recordar que inria es un dataset de imágenes aéreas de solo 2 clases y el de Ipsrs es de 6 clases distintas.

Para el entrenamiento de estos datasets se ha hecho fine-tunning utilizando el modelo congelado para la estructura de Xception entrenado en ImageNet y COCO. Los parámetros utilizados han sido los mismos que se han utilizado para el entrenamiento de ADE20K que se puede ver en la tabla 2 con alguna que otro pequeño cambio relacionado especialmente con la resolución que se quiere entrenar y evaluar.

Empezaremos con el dataset de INRIA. Dataset de imágenes aéreas en el que hay que clasificar 2 clases distintas: edificio y no edificio.

Se pueden ver los resultados en la tabla 4. En los resultados de la tabla se puede observar como va aumentando el rendimiento del modelo en base a la resolución de la imagen.

En cuanto a los resultados del dataset ISPRS que se pueden ver en la tabla 5 se puede ver exactamente la misma tendencia que en 4

A pesar de que el modelo emplea atrous convolution y el modelo computa información de la imagen a distintas resoluciones, no se puede generar más información que la que hay contenida en la original, por ese motivo el hecho de que las imágenes estén en baja resolución perjudica el rendimiento del modelo.

Con todo esto, podemos llegar a la conclusión que, tal y como se esperaba, la resolución con la que se realizan los distintos experimentos es crucial a la hora de obtener buenos resultados. Esta claro que este fenómeno se puede extrapolar a imágenes no aéreas, pero es muy probable que este factor no sea tan agravante como con las imágenes aéreas, al fin y al cabo el entrenamiento de los datasets no aéreos ha sido en su gran mayoría con imágenes de una resolución alrededor de los 500x500p.

En el apéndice se pueden ver el resultado de alguna de las imágenes.

8 CONCLUSIONES

En este trabajo se ha hablado del modelo con el que se han realizado los experimentos. Las distintas técnicas y métodos que este emplea y la estructura general empleada.

Debido a la propia naturaleza de las imágenes aéreas, es necesario un método para poder obtener más información de la que obtendríamos con una simple imagen a color. Siendo la naturaleza de estas que, al ser imágenes tomadas desde tanta distancia hace que el número de información por metro cuadrado que se puede extraer de una imagen se vea drásticamente reducido. Reduciendo de esta forma el rendimiento de los modelos de clasificación.

En este trabajo solo se ha explorado la técnica de aumentar la resolución, pues a más resolución más pixeles por metro cuadrado disponemos, lo cual permite extraer mayor información de la imagen.

Otros métodos que no se han podido probar son aquellos relacionados con la utilización de imágenes con más de 3 canales (RGB). Imágenes extraídas por sensores más sofisticados que sean capaces de obtener información extra de la imagen.

Pese que un aumento de la resolución con la que se ha entrenado los otros datasets habría conllevado mejores resultados, estos resultados no habrían sido tan significativos como lo han sido para los datasets de imágenes aéreas. Los imágenes no aéreas han conseguido muy buenos resultados pese a la resolución de aproximada de 512x512p.

Para acabar, el mayor problema que hay que afrontar para las imágenes aéreas es el del coste computacional. La idea detrás de todo esto es que es necesaria más información por pixel, este hecho, ya sea empleando una mayor resolución o imagenes multi-espectrales conlleva un extra de coste en el modelo. Y pese que algunos de estos problemas se estan solucionando gracias a nuevas técnicas de convolución, aún falta bastante con tal de obtener mejores resultados.

Agradecimientos Este trabajo se ha realizado con la colaboración de Robert Benavente y Daniel Ponsa con soporte del proyecto BOSSS(TIN2017-89723-P).

REFERÈNCIES

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[2] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," *CoRR*, vol. abs/1704.06857, 2017.

[3] P. SHARMA, "top 5 deep learning frameworks their applications and comparisons," 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," September 2014.

[8] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (C VPR)*, 2016.

[10] "International society for photogrammetry and remote sensing semantic labeling contest," in *http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html*, 2016.

[11] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, IEEE, 2017.

[12] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018.

[13] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.

[14] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," 05 2014.

A APÉNDICE

Diagrama de Gantt con la planificación del proyecto.

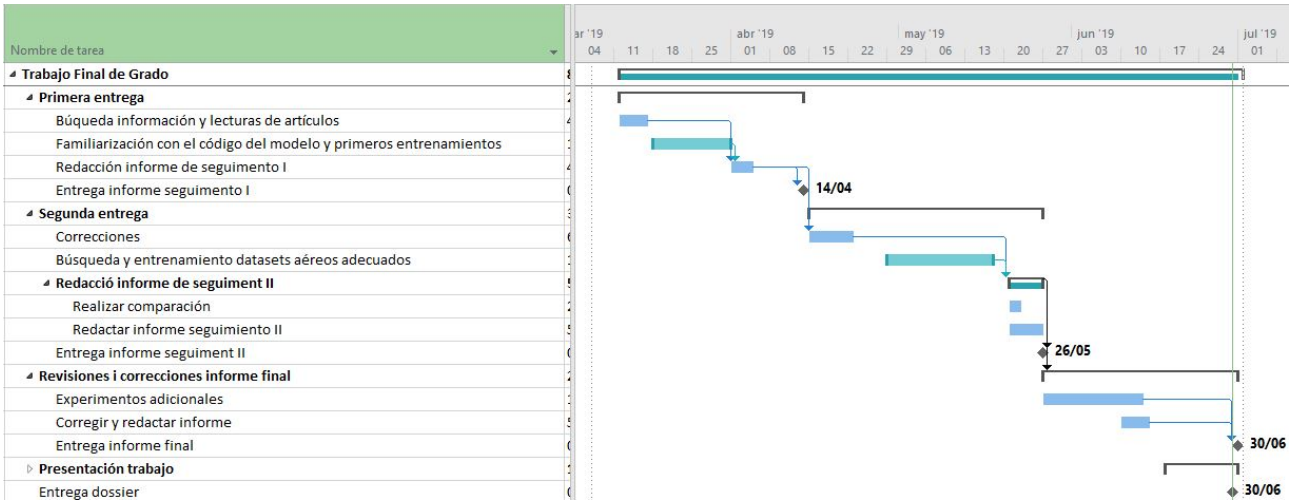
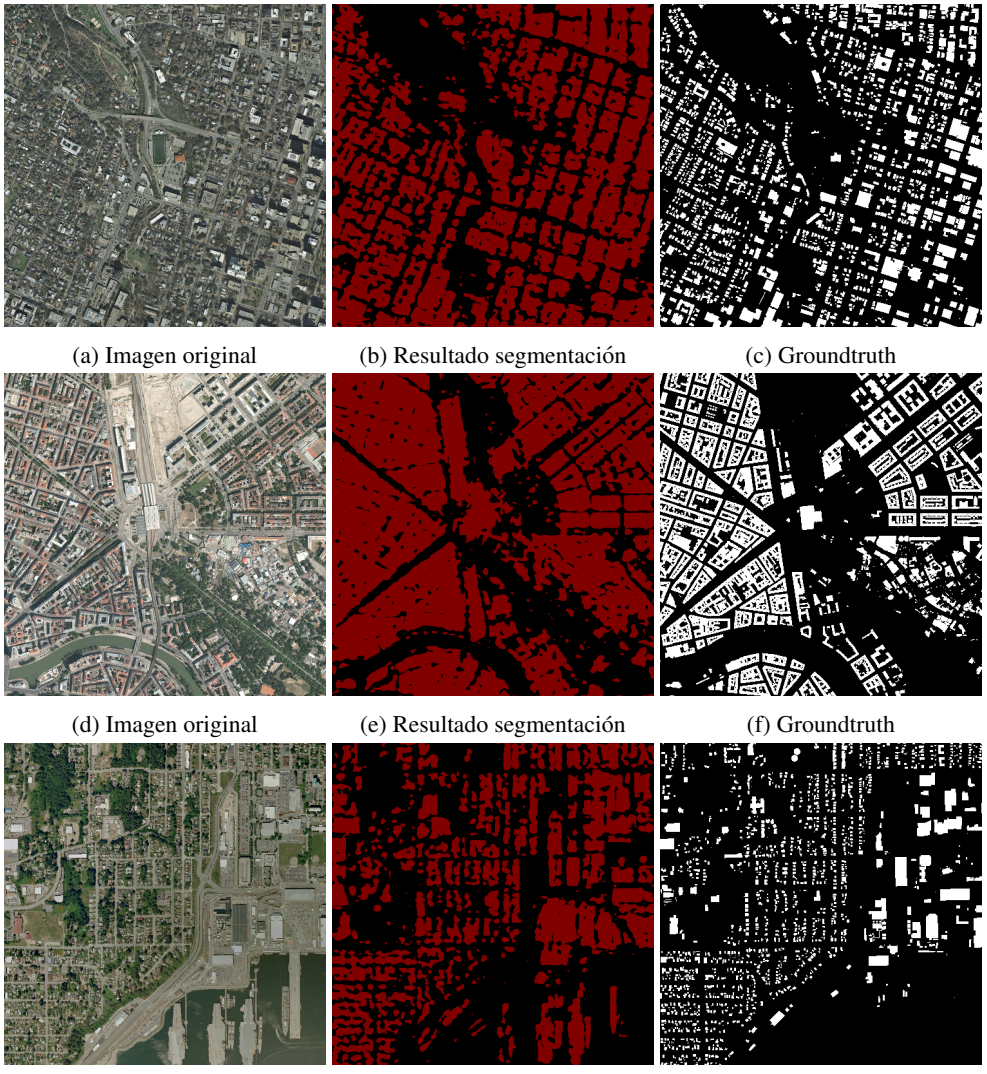


Fig. 9: Diagrama de Gantt referente al proyecto, para ver con más detalle ver el pdf "DiagramaGantt.pdf"

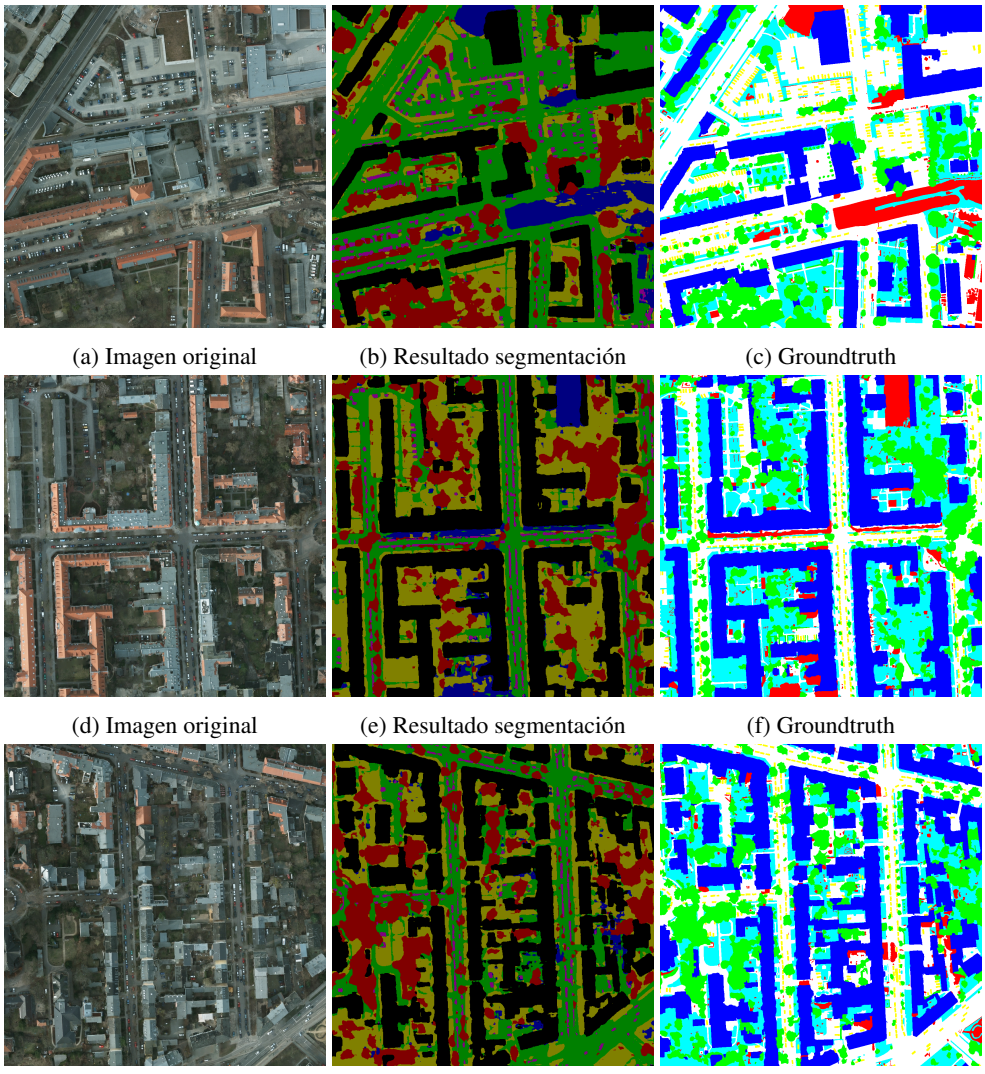


(a) Imagen original (b) Resultado segmentación (c) Groundtruth

(d) Imagen original (e) Resultado segmentación (f) Groundtruth

(g) Imagen original (h) Resultado segmentación (i) Groundtruth

Fig. 10: Algunos ejemplos del resultado del dataset INRIA resolución original 800x800. El hecho de que la máscara de la predicción sea roja en lugar de blanca es debido a la paleta de colores que se utilizó. Se puede ver como las predicciones parece dilatar las zonas donde parece que hay edificio. Las calles principales y las zonas peatonales grandes parecen ser claramente vísibles como se puede ver en (b). En cambio las carreteras más interiores y estrechas no se muestran tan bien debido a la dilatación de las máscaras de los edificios.



(a) Imagen original (b) Resultado segmentación (c) Groundtruth

(d) Imagen original (e) Resultado segmentación (f) Groundtruth

(g) Imagen original (h) Resultado de la segmentación (i) Groundtruth

Fig. 11: Algunos ejemplos del resultado del dataset ISPRS resolución original 1200x1200. Los cambios de los colores es debido a la paleta de colores empleada durante el entrenamiento. Los edificios son el elemento que mejor se ha clasificado. En cambio, las carreteras y las zonas con naturaleza también debido al ser máscaras más grandes y de colores tan similares.