Multispectral Earth Simulator

Guillermo Cid Munuera

Escola de Enginyeria, Universitat Autònoma de Barcelona

1 Introducción

Actualmente disponemos de una serie de servidores web que nos ofrecen un servicio de mapas en los que podemos observar distintos tipos de imágenes aéreas ya sea de satélites, aviones o incluso drones.

En el caso de las imágenes de mayor calidad como las de aviones, estas se suelen realizar cada uno o dos años por institutos especializados en cartografía, como por ejemplo en España el IGN [4]. En cambio, en el caso de las imágenes de satélite estas suelen estar al alcance de cualquiera y se pueden obtener datos prácticamente a diario, el inconveniente es la menor calidad respecto a las imágenes de avión o dron.

La Agencia Espacial Europea [1] desarrollo a finales de los años noventa un programa espacial el cual pretende recoger datos aéreos para la comunidad científica. Uno de sus proyectos es el satélite Sentinel2, este consta de dos satélites idénticos Sentinel-2A y Sentinel-2B. Estos cubren toda superfície terrestre entre las latitudes 56 grados sur y 84 grados norte, recolectando imágenes mustiespectrales de la misma zona cada cinco o diez dias dependiendo de la latitud, las imágenes constan de 13 bandas cuyas resoluciones varian entre los 10 metros, 20 metros o los 60 metros por píxel.

Durante los últimos años ha habido un gran auge en el campo del aprendizaje computacional, en concreto el aprendizaje profundo. Esto es debido a los grandes resultados obtenidos por ejemplo por modelos de predicción como las redes convolucionales. Esta es la tecnología que se utilizara para conseguir los objetivos del proyecto.

En este proyecto queremos implementar un módulo que a través de redes convolucionales tengamos una imagen de Sentinel2 como entrada i la salida sea una imagen con una resolución igual a la de un dron y una estimación de las alturas (mapa DTM) para una posterior recreación 3D, esta recreación no entraria dentro del proyecto ya que este esta centrado exclusivamente en el aprendizaje profundo.

Esto permite un gran número de futuras implementaciones como poder recrear el 3D de cualquier zona del mundo, así poder observar esa zona durante un periodo de tiempo concreto y con distintas bandas que no ofrecen las cámaras de los drones.

2 Objetivos

El proyecto se realizará usando el lenguaje de programación Python [3], con ayuda del framework Pytorch [2] para el apartado del deep learning.

Este proyecto esta dividido en tres módulos:

1. Recrear imagén de Sentinel2 de una resolución de 10 m/píxel a avión 25 cm/píxel
2. Superresolución de imágen de avión de una resolución de 25 cm/píxel a dron 5 cm/píxel
3. Estimación del mapa de alturas a partir de una imágen de avión de 25 cm/píxel

En cada uno de los módulos se realizará un previo estudio del estado del arte antes de adentrarse en la implementación. Cada módulo particular también necesita la obtención previa de imágenes ya sea a través de servicios web o APIs, esto requerirá la implementación de funcionalidades automatizadas para la obtención de estos datos.

En el Apartado de planificación se muestra cada uno de los módulos subdivididos en distintos apartados. Se puede observar de forma más específica el trabajo realizado en cada módulo y su duración.

3 Planificación

Al haber tres grandes objetivos dentro del TFG, para una correcta planificación se muestra a continuación el diagrama de gantt de los aspectos técnicos que conlleva la realización de los módulos. En estos se muestra la durada aproximada de cada tarea.

Como ya he comentado, este diagrama solo incluye tareas técnicas. No se incluyen las reuniones con el tutor del TFG, y tampoco se incluye la realización de la documentación final.

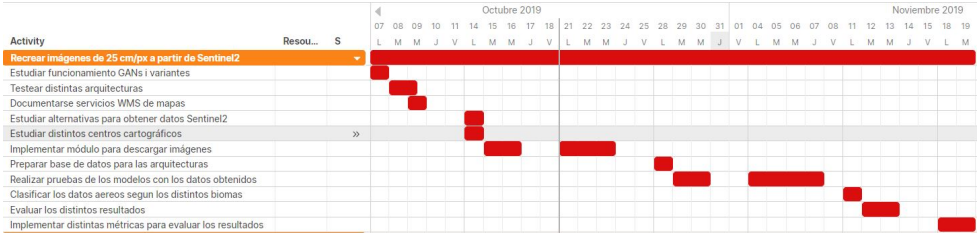

Fig. 1: Diagrama de Gant del primer módulo descrito en objectivos del informe

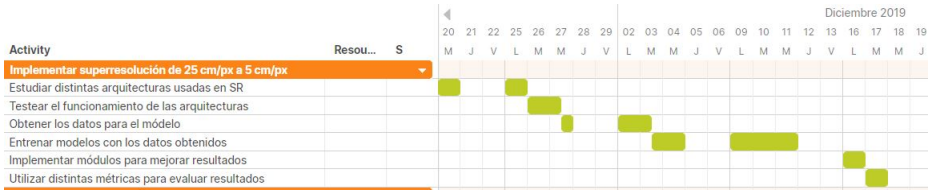

Fig. 2: Diagrama de Gant del segundo módulo descrito en objectivos del informe

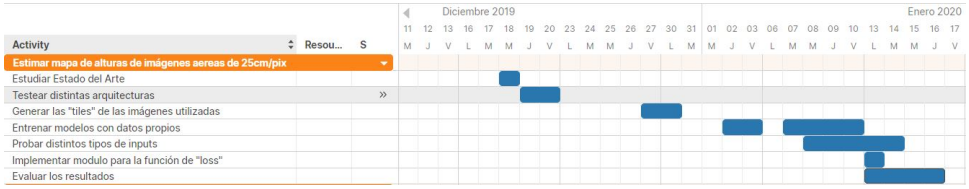

Fig. 3: Diagrama de Gant del tercer módulo descrito en objectivos del informe

4 Metodología

Todo el conjunto de objetivos del TFG se realizarán usando la metodología ágil de Kanban, pero confeccionada para el trabajo en este caso individual, se realizarán reuniones diarias de corta duración con el tutor del TFG donde habrá un control del proyecto, para poder definir las direcciones a seguir.

5 Desarrollo

5.1 Recolección de datos

Para realizar los distintos experimentos se requiere un mecanismo que nos permita obtener una gran cantidad de datos de forma sistematizada. Para la obtención de datos del satélite Sentinel-2, el input de la red, existen distintos servicios/APIs de pago o gratuitos, en los de pago los datos se obtienen ya procesados, en cambio en los servicios gratuitos se obtienen sin procesar. Para la obtención de las imágenes de Avión, el ground truth de la red, se ha utilizado un WMS ("web map service").

5.1.1 SentinalSat Python API

SentinelSat es una API programada en Python que accede al servicio Copernicus Open Access Hub de la Estación espacial Europea (ESA), este servicio proporciona datos gratuitos de los satélites Sentinel-1, Sentinel-2, Sentinel-3 i Sentinel-5P.

En las peticiones a la API se incluye un conjunto de parámetros determinados, la respuesta se recibe en formato "geodataframe" con la información solicitada, esto son los parámetros usados al hacer la petición:

- Región: zonas con de un tamaño de 100x100km2.
- Fecha: Rango de fechas deseado.
- Plataforma: Satélite de donde se obtienen los datos, en nuestro caso el Sentinel-2.
- Porcentaje de nubes: porcentaje de nubes máximo aceptado en la región obtenida, el usado ha sido de 10

Una vez hecha la solicitud, se descarga la opción con un menor porcentaje de nubes y que esta no contenga zonas no visibles.

5.1.2 Generación del Dataset

Sentinel sat nos proporciona un simple sistema de ficheros dónde se encuentra información geográfica y las correspondientes 13 bandas del Sentinel-2. Estas imágenes están codificadas en un formato J2P de 12 bits y tienen un procesado atmosférico pobre (L1C) que requiere de un software llamado Sen2Cor para conseguir una mejor corrección atmosférica (L2A).

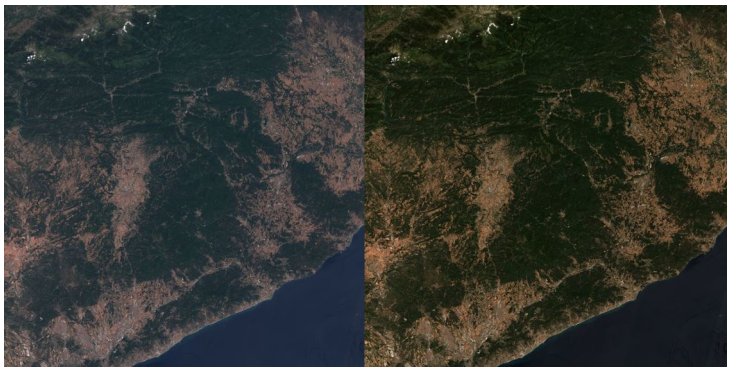

Fig. 4: Ejemplo de corrección atmosferica del software Sen2Cor

Con la librería Open Source Gdal conseguimos crear un archivo virtual juntando las capas 4, 3 y 2 en este preciso orden para conseguir la "True color image" (TCI). Y posteriormente aplicamos una conversión de 12 bits a 8 para construir la imagen en RGB. La ESA también nos proporciona un clasificador de la región (SCL) por tipo de zona, por ejemplo como bosque, urbano, mar o cultivo.

Una vez los datos están procesados se han generado pequeños recortes de 32x32 pixels para así disponer de un dataset de un tamaño total de más de 100 mil imágenes de satélite, cada una de ellas con información sobre sus coordenadas.

Hasta ahora se han conseguido las imágenes de Satélite (10 m/píxel) que suponen la entrada a la red, ahora se necesitan las imágenes de Avión (25

cm/píxel) para que así la red aprenda a generarlas. El instituto cartográfico y geológico de Cataluña (ICGC) dispone de un servicio web de mapas (WMS) en el que se pueden solicitar datos de Ortofotos de Avión de distintos años. A través de la librería de Python OWSLib se realiza la petición correspondiente pudiendo obtener la imágen deseada a partir de unas coordenadas obtenidas de las imágenes de satélite. Algunas de las imágenes del conjunto de datos del ICGC se ha visualizado que tienen un cierto grado de distorsión, para descartar que este tipo de datos entren en el conjunto de entrenamiento de la red se ha aplicado una tècnica que asume que el ruido sigue una distribución Gaussiana y estima el valor de la desviación estándar de esta gaussiana. Las imágenes de mejor calidad con un mayor conjunto de contornos serán las que tienen una valor estimado más elevado.

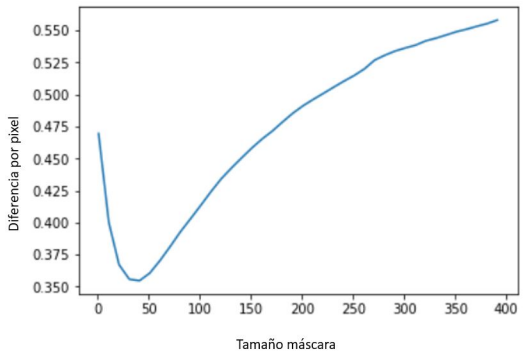

Fig. 5: Diferencia por píxel respecto imagen emborronada e imagen satelite con diferentes tamaños de mascara Gasussiana

Con estas dos implementaciones con la SentinelAPI i el WMS del ICGC se ha generado el dataset para el aprendizaje de la red.

Fig. 6: Diferencia entre imagenes del ICGC con una alto ruido (izquierda) respecto imagen nitida (derecha)

5.2 Arquitectura Utilizada

5.2.1 Super Resolución

Para resolver el problema de super resolución usamos la arquitectura diseñada en [referencia carn]. CARN es un modelo basado en ResNet, pero los modulos residuales pasan a ser tener interconexiones de tipo local y global. Es decir, se utilizan modulos de cascada ("Cascading modules"). Los outputs de capas intermedias se usan como inputs de capas más avanzadas. Se puede ver en la Fig. 7. como es la estructura de cada bloque y de la red. Esta arquitectura esta compuesta relativamente de un número bajo de bloques cascada y a diferencia de otras redes donde el aumento de tamaño de imagen se hace al inicio, CARN tiene un bloque final para realizar el 'upsampling'. Estos factores comportan que la red sea ligera, por lo tanto, tiene menos parametros y realiza menos operaciones que el resto de redes de su ambito, sin que su rendimiento se vea afectado significativamente. Para el entrenamiento de la red usamos recortes de imágenes de

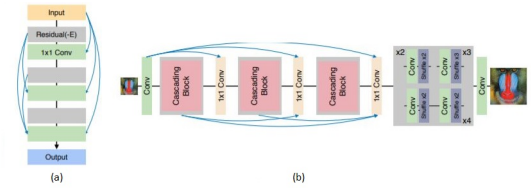

Fig. 7: (a) Bloque cascada (b) Arquitectura de CARN

64x64, y aplicamos técnicas de para aumentar el número de muestras de forma "online". A cada imágen cargada para que la red la trabaje le realizamos un flip de izquierda a derecha y de arriba a abajo aleatorio, y entre 0 y 4 rotaciones de 90 grados. Con esto conseguimos multiplicar la posibilidad de imáges trabajadas por el modelo en 8 como se ve en la Fig. 8.

5.2.2 GANs

Para conseguir la alucinación nos hemos fijado en las arquitecturas que ya han conseguido esta funcionalidad en otros ámbitos, estas arquitecturas son las Redes generativas antagónicas o en inglés "Generative Adversarial Networks", más conocidas como GANs.

Estas están compuestas por dos subredes que compiten entre ellas. Por un lado tenemos la red generativa, la encargada de generar la imagen de salida, por lo tanto, esta tiene una arquitectura con capas totalmente convolucionales. Por otro lado, tenemos la red discriminadora, esta analiza el producto de la red generativa y determina si se ajusta a lo que está buscando. Estos conjuntos de redes permiten que la red generadora, al querer engañar a la otra red, produzca

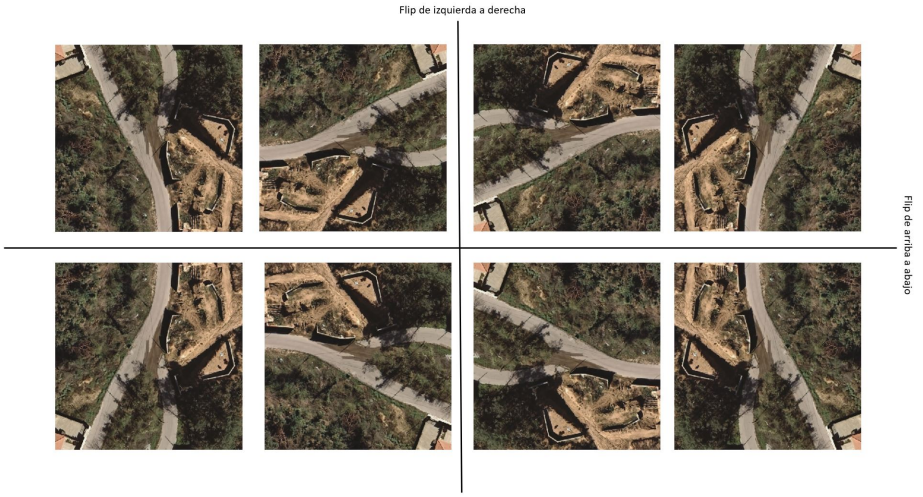

Fig. 8: Ejemplo de aumento de los inputs en la fase de entrenamiento realizando Flips y rotaciones

imágenes de salida que no existen, solo están basadas en la realidad para que la discriminadora de como real la imagen de salida.

El forzar a que genere contenido irreal da pie a que las GANs tengan una estabilidad inconstante, es difícil crear un modelo que dé buenos resultados ya que es fácil que el propio modelo colapse, los datos naturales suelen tener concentraciones de muestras de datos similares, pero distintos en cada tipo de imagen. Como el generador solo quiere engañar al discriminador puede llegar a producir muestras de un solo tipo de concentraciones de datos o patrón de datos.

Otra de las complicaciones de las GANs es la de saber cuándo se debe acabar la fase de entrenamiento. No podemos determinar cuando el modelo converge a partir de la función de "Loss" del generador y del discriminador. Por lo tanto, es difícil determinar cuando el generador está realizando imágenes de calidad y en muchos casos se debe recurrir a la visualización.

5.3 Metricas utilizadas

Para ser consciente de la calidad de los resultados hemos escogido varios tipos de metricas para nuestro caso. En problemas similares al nuestro es común usar el PSNR y el SSMI. 'Peak signal to Noise Ratio' es una metrica que consiste en la inversa proporcional del logaritmo del error medio cuadratico (MSE) ente la imagen de salida y el groundtruth. Cuanto mayor es el PSNR mejor.

$$\mathrm{PSNR} = 10 \log_{10} \frac{L^2}{MSE} \tag{1}$$

Esta es la formula del PSNR donde L es el valor máximo del pixel de la imagen en concreto, es decir, si la imagen es rgb de 8 bits el valor es 255. Esta métrica es

realmente simple ya que solo cálcula el error por pixel, lo cual, en algunos casos no representa una medida adecuada a la perceptual por el ojo humano.

El SSIM es una mética para medir la similaridad estructural entre imágenes, donde se compara entre la imagen de salida y el groundtruth la luminancia, el contraste y la estructura. Cuanto mayor es el SSMI mejor.

$$\text{SSMI(i,j)} = \frac{(2\mu_i\mu_j + C_1)(\sigma_{ij} + C_2)}{(\mu_i^2 + \mu_j^2 + C_1)(\sigma_i^2 + \sigma_j^2 + C_2)} \tag{2}$$

Podemos ver en la formula donde la μ_i representa la media de la imagen, la σ_i es la desviación estandard y C1, C2 son constantes para evitar la inestabilidad.

Estas dos últimas métricas explicadas que evaluan el error por pixel y la estructura como bien hemos comentado, son un problema para evaluar de forma númerica las imágenes generadas por las GANs. Estas son imágenes generadas son totalmente nuevas, por lo tanto, contienen algo de distorsión y ruido, lo más probable es obtener un mal resultado de PSNR y SSMI aunque perceptualmente, al ojo humano, el resultado sea acceptable. El FID es una métrica capaz de evaluar imágenes con resultados significativos y comparables para imágenes generadas con GANs, en el PAPEEER se demuestra como la métrica es robusta a ruido como el gausiano, emborronamiento y 'salt and pepper'.

$$\text{FID(x,g)} = ||\mu_x - \mu_g||_2^2 + Tr(\sum x + \sum g - 2(\sum x \sum g)^{\frac{1}{2}}) \tag{3}$$

5.4 Fase Experimental

5.4.1 Fase inicial

Una vez que disponemos de todo el conjunto de datos explicado en el apartado de recolección de datos, el primer paso a realizar ha sido el de testear distintas arquitecturas como la Styled GAN o la pix2pix / pix2pixHD, está última ha sido la usada en la mayoría de experimentos.

Se han cogido un conjunto reducido, aproximadamente 4000 imágenes, del total del dataset de imágenes de avión. A estas imágenes se ha disminuido su resolución con la tècnica de interpolación por aproximación. En este primer entrenamiento, hemos podido sacar varias conclusiones a partir de los resultados. Estos han sido imágenes en que la red ha recreado algunos patrones propios de las imágenes reales como por ejemplo árboles, cultivos, caminos... En cambio otros patrones de zonas urbanas no los ha podido recrear. Esto es debido a que estos no son mayoritarios y que precisamente estos son más difícil de producir. Al querer cumplir el conjunto de objetivos propuestos al inicio del proyecto, se realizarán los próximos experimentos con un conjunto de datos más concretos. Este conjunto excluye las zonas urbanas y se centra en zonas de bosque. Estos datos deberán ser muy concretos y similares entre sí para evitar las complicaciones que pueden surgir al entrenar la red, explicado en el apartado de la Arquitectura utilizada.

Se ha realizado el mismo experimento anterior pero con un grupo más selecto dentro del conjunto de entrenamiento y los resultados son más favorables. Esto

hace reflexionar y podemos pasar a la siguiente paso ya enfocado la entrada de las imágenes a un conjunto más similar a las imágenes de Satélite. Para conseguir este objetivo se han realizado dos tipos de trato con la imagen “Offline”, es decir, fuera de la fase de entrenamiento de la propia red, y un trato “Online” en este caso el procesado se produce durante la fase de entrenamiento.

Fig. 9: Resultados de la red pix2pix input, groundtruth y output (de arriba a abajo)

5.4.2 Tratamiento Offline de imágenes

En este caso el conjunto de imágenes de Satélite se ha aumentado su resolución a 1024 x 1024 y de las imágenes de Avión se ha aplicado un emborronamiento de tipo Gaussiano, el objetivo es conseguir una imágen lo más similar a la de Satélite. Aplicando varios tipos de máscara para emborronar en un conjunto reducido del dataset se ha obtenido una máscara óptima que se ha aplicado a la totalidad de imágenes de Avión. Para medir la similaridad de las imágenes de Satélite con las emborronadas, se normaliza por banda las imágenes y se calcula el error por pixel aplicando la suma de todas las diferencias absolutas entre los propios pixels de las imágenes.Las imágenes emborronadas con mayor similitud a las de Satélite son las que se utilizan como conjunto de entrenamiento.

5.4.3 Tratamiento Online de imágenes

En este proceso se ha modificado la implementación de la red neuronal, se calcula la media y desviación estándar de cada mini-batch (en el caso de nuestra arquitectura es de 1), se normaliza este mini-batch a partir de este cálculo. Por lo tanto, cada imágen de entrada tiene una normalización determinada. Esto permite entrenar con las imágenes emborronadas producidas en el tratamiento Offline y posteriormente realizar un fine tuning con las imágenes de Satélite que

normalizando estos datos tienen una gran similitud. Por lo tanto, se espera un resultado similar.

5.4.4 Magnitud de las GANs

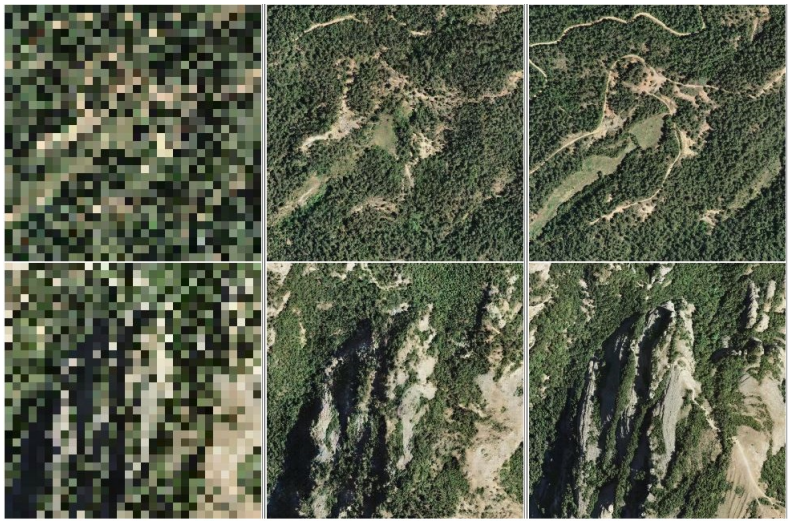

Fig. 10: Fase Recreación primer experimento de 16 m/px a 1 m/px (input,output,groundtruth)

Hemos decidido realizar una série de experimentos para probar la magnitud de las GANs, hasta que punto la red genera imágenes factibles para el ojo humano. El objetivo de este experimento es pasar de una imágen de 32x32 y 16 m/pixel a una de 2048x2048 y 25 cm/pixel, es decir, un aumento de x128. Realizaremos un total de 3 experimentos para poder sacar una serie de conclusiones, estos experimentos tendrán una primera fase de recreación o alucinaje donde se usará la arquitectura comentada en el punto 5.2.2 y una segundo fase de super resolución, punto 5.2.1, si es necesaria.

1. Recrear imágen de 16 metros/pixel a 1 m/px (x32) y super resolución de 1 m/px a 25 cm/px (x4).
2. Recrear imágen de 16 m/px a 50 xm/px (x64) y super resolución de 50 cm/píxel a 25 cm/px (x2).
3. Recrear imágen de 16 m/px a 25 cm/px sin fase de super resolución (x128).

Con este experimento podremos llegar a observar el alcance que tienen estas arquitecturas jugando con el nivel de 'upsampling' y finalmente sacando métricas descritas en el punto 5.3.

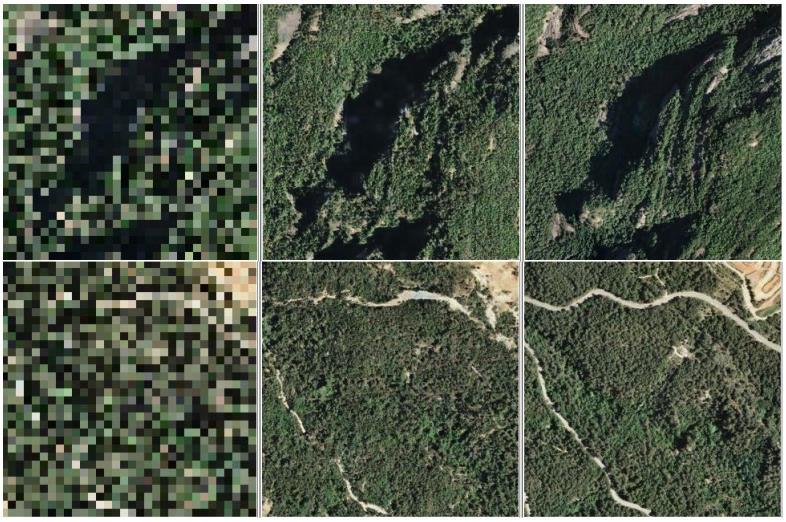

Fig. 11: Fase Recreación primer experimento de 16 m/px a 50 cm/px (input,output,groundtruth)

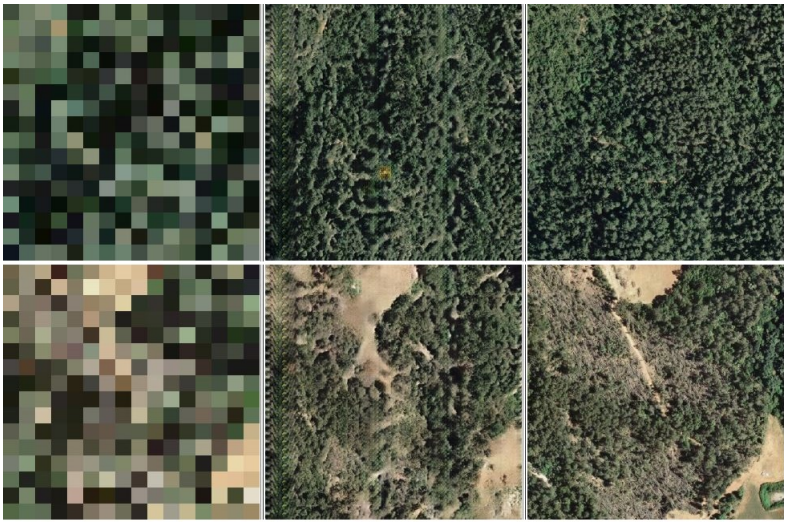

Fig. 12: Fase Recreación primer experimento de 16 m/px a 25 cm/px (input,output,groundtruth)

Mostramos los resultados, de la fase de recreación únicamente, más significativos de cada experimento. En la Fig. 10, Fig. 11 y Fig. 12 se ven de visualmente y en la Tabla. 1 numericamente en este caso el FID.

Fase Recreación	**x128**	**x64**	**x32**
FID	157.40	148.45	135.86

Table 1: Métrica FID per cada uns dels experiments de la fase de recreació

6 Conclusiones

En las primeras conclusiones nos centraremos en estos últimos resultados de la fase experimental, en el tercer experimento (x128) se han obtenido resultados que visualmente recrean parte de la imágen pero contienen demasiado ruido y ciertos patrones que distorsionan la imagen que se repiten en cada una de las muestras de test. En el segundo experimento (x64) se obtiene una imágen más creible y más parecida a la real, però por lo general y como se puede ver en la FFF se obserba como hay patrones de la imágen que le cuesta reacrear. En caso del primer experimento (x32) conseguimos mejorar bastante el problema del segundo y se consigue recrear con bastante detalle el patrón de la imagen real. A estas observaciones visuales le acompaña los resultados numéricos de la métrica utilizada.

Nos encontramos ante una situación que a la red le cuesta cuando tiene una gran diferencia de tamaño entre la imágen de entrada y la generada, como es lógico. A falta de la fase de superresolución en los experimentos 1 y 2, se puede enfocar el experimento 3 en usar una arquitectura que permita generar una imágen falsa sin enfocar el resultado en conseguir un output que siga la estructura de la imágen real sino en que la imagen generada sea simplemente creible para el ojo humano basandonos en [referencia a progressive growing GAN].

References

1. ESA Sentine Homepage, https://sentinel.esa.int. Last accessed 6 Oct 2019

2. Pytorch Homepage, https://pytorch.org/. Last accessed 6 Oct 2019

3. Python Homepage, https://www.python.org/. Last accessed 6 Oct 2019

4. Instituto Geográfico Nacional Homepage, https://www.ign.es. Last accessed 6 Oct 2019

5. SentinelSat API Homepage, https://sentinelsat.readthedocs.io/en/stable/api.html. Last accessed 9 Nov 2019

6. Institut Cartogràfic i Geològic de Catalunya Homepage, https://www.icgc.cat/. Last accessed 9 Nov 2019

7. Copernicus Open Access Hub Homepage, https://scihub.copernicus.eu.Last accessed 9 Nov 2019

8. OWSLib 0.18.0 documentation, https://geopython.github.io/OWSLib/. Last accessed 9 Nov 2019

9. Sen2cor Homepage https://step.esa.int/main/third-party-plugins-2/sen2cor/.Last accessed 9 Nov 2019

10. OSGEO GDAL Homepage https://www.osgeo.org/projects/gdal/. Last accessed 9 Nov 2019

11. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In NIPS'2014.

12. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-Image Translation with Conditional Adversarial Networks. arXiv.org (2016)

13. T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation.