eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

SISTEMAS ELECTRÓNICOS AVANZADOS

TRABAJO FIN DE MÁSTER

SISTEMA EMBEBIDO DE DETECCIÓN DE NUBES PARA CÁMARAS MULTIESPECTRALES DE ALTA RESOLUCIÓN EN MICROSATÉLITES DE OBSERVACIÓN TERRESTRE

Estudiante *Díaz Rodríguez, Mikel*
Director/Directora *Basterretxea Oyarzabal, Koldo*
Departamento Tecnología Electrónica
Curso académico *2021-2022*

Bilbao, 18, septiembre, 2022

Resumen

La próxima generación de cámaras binoculares iSIM que la empresa *Satlantis* desarrolla y comercializa para su instalación en microsatélites artificiales dispondrá de sensores multiespectrales de hasta 5 bandas (RGB + NIR + Pancromático) con una alta resolución de imagen. La gran resolución espacial que ofrecen las cámaras iSIM, combinadas con su compacto diseño y bajo peso, suponen un logro tecnológico que permite reducir notablemente los costes de las misiones destinadas a la observación terrestre. Sin embargo, son precisamente estas características las que hacen más necesario maximizar y optimizar el procesamiento de las imágenes en la propia cámara (procesamiento onboard). Siguiendo con este propósito, en este trabajo se propone la detección en tiempo real de la presencia de formaciones nubosas en las imágenes tomadas por las cámaras iSIM. Este sistema proporcionará la posibilidad de descartar automáticamente estas imágenes antes de ser procesadas, almacenadas o transmitidas, y ello redundará en la optimización de la utilización de los limitados recursos disponibles en los satélites. Para la detección de nubes, tan solo se dispone de las bandas del visible y del infrarrojo cercano (VNIR), siendo éste un reto en el desarrollo, ya que generalmente se utilizan las bandas del infrarrojo de onda corta (SWIR) para este tipo de segmentación debido a la similitud de las características espectrales del VNIR en terrenos de alto albedo (nieve, luminosidad de las ciudades, etc.). Recientemente, el aprendizaje automático ha ofrecido soluciones prometedoras al problema del enmascaramiento de nubes, permitiendo una mayor flexibilidad que las técnicas tradicionales de umbralización, que se limitan únicamente a utilizar las bandas espectrales. Sin embargo, son pocos los estudios que se centran en ofrecer un rendimiento robusto en los entornos de alto albedo y que además ofrezcan pruebas experimentales en dispositivos que se puedan utilizar a bordo del satélite para su procesamiento en tiempo real. El algoritmo de aprendizaje profundo propuesto denominado *SatCloud* tiene en cuenta estas limitaciones, pudiendo capturar y agregar características a varias escalas (multiescala), garantizando que las características semánticas de alto nivel extraídas de los terrenos de alto albedo, como la nieve y las nubes, sean más distintivas. Gracias a una serie de opciones de diseño y técnicas de entrenamiento que mejoran el rendimiento, muestra un rendimiento de vanguardia, comparable al de otros métodos, una gran velocidad y una gran solidez ante todo tipo de situaciones. La experimentación se lleva a cabo en un MPSoC (Multiprocessor System on Chip), demostrando su viabilidad para la implementación en tiempo real del algoritmo propuesto en órbita.

Palabras Clave: microsatélites, imagen multiespectral, visión artificial, aprendizaje profundo, detección de nubes, fusión de características multiescala, sistema embarcado, MPSoC.

Laburpena

Satlantis enpresak mikrosatelite artifizialetan instalatzeko garatzen eta merkaturatzen dituen iSIM kamera binokularren hurrengo belaunaldiak irudi-bereizmen handiko 5 banda (RGB + NIR + Pankromatikoa) arteko espektro anitzeko sentsoreak izango ditu. ISIM kamerek eskaintzen duten espazio-bereizmen handia, beren diseinu trinkoarekin eta pisu txikiarekin konbinatuta, lorpen teknologiko bat dira, lurreko behaketara bideratutako misioen kostuak nabarmen murriztea ahalbidetzen duena. Hala ere, ezaugarri horiek dira, hain zuzen ere, kameran bertan irudien prozesamendua maximizatzea eta optimizatzea beharrezkoena egiten dutenak (prozesamendu onboard). Helburu horrekin jarraituz, lan honetan proposatzen da iSIM kamerek hartutako irudietan hodei-formazioak denbora errealean detektatzea. Sistema horrek aukera emango du irudi horiek prozesatu, biltegiratu edo transmititu aurretik automatikoki baztertzeko, eta horrek sateliteetan eskuragarri dauden baliabide mugatuen erabilera optimizatzea ekarriko du. Hodeiak detektatzeko, ikuslearen eta infragorri hurbilaren (VNIR) bandak baino ez daude, eta hori garapenerako erronka bat da; izan ere, uhin laburreko infragorriaren (SWIR) bandak erabiltzen dira segmentazio mota horretarako, albedo altuko lurretan (elurra, hirietako argitasuna, etab.) VNIRen ezaugarri espektralen antzekotasuna dela eta. Berriki, ikaskuntza automatikoak irtenbide oparoak eskaini dizkio hodeiak ezkutatzearen arazoari, eta, horri esker, ohiko atalkatze-teknikak baino malgutasun handiagoa lortu da, horiek espektro-bandak erabiltzera mugatzen baitira. Hala ere, gutxi dira albedo altuko inguruneetan errendimendu sendoa eskaintzen duten eta satelite barruan denbora errealean prozesatzeko erabil daitezkeen gailuetan proba esperimentalak eskaintzen dituzten ikerketak. Proposatutako ikasketa sakoneko algoritmoak, *SatCloud* izenekoak, muga hauek hartzen ditu kontuan, eta hainbat eskalatan (multieskalan) ezaugarriak harrapatu eta gehitu ditzake, goi albedoko lurretatik (elurra eta hodeiak, esaterako) ateratako goi mailako ezaugarri semantikoak bereizgarriagoak izan daitezen bermatuz. Errendimendua hobetzen duten diseinu-aukerei eta entrenamendu-teknikei esker, abangoardiako errendimendua erakusten du, beste metodo batzuekin aldera daitekeena, abiadura handia eta sendotasun handia egoera guztien aurrean. Esperimentazioa MPSoC (Multiprocessor System on Chip) batean egiten da, orbitan proposatutako algoritmoa denbora errealean ezartzeko bideragarria dela frogatuz.

Gako-hitzak: mikrosateliteak, espektro anitzeko irudia, ikusmen adimenduna, hodeien detekzioa, eskala anitzeko ezaugarrien fusioa, ontziratze-sistema, MPSoC.

Abstract

The next generation of iSIM binocular cameras developed and marketed by *Satlantis* for installation on artificial microsatellites will feature multispectral sensors with up to 5 bands (RGB + NIR + Panchromatic) and high image resolution. The high spatial resolution offered by iSIM cameras, combined with their compact design and low weight, is a technological achievement that can significantly reduce the cost of Earth observation missions. However, it is precisely these characteristics that make it all the more necessary to maximise and optimise image processing in the camera itself (onboard processing). In line with this purpose, this work proposes the real-time detection of the presence of cloud formations in the images taken by iSIM cameras. This system will provide the possibility to automatically discard these images before they are processed, stored or transmitted, thus optimising the use of the limited resources available on the satellites. For cloud detection, only the visible and near-infrared (VNIR) bands are available and this is a development challenge, as the shortwave infrared (SWIR) bands are generally used for this type of segmentation due to the similarity of the spectral characteristics of the VNIR in high-albedo terrain (snow, city lights, etc.). Recently, machine learning has offered promising solutions to the problem of cloud masking, allowing greater flexibility than traditional thresholding techniques, which are limited to using only spectral bands. However, few studies focus on robust performance in high-albedo environments and offer experimental tests on devices that can be used onboard the satellite for real-time processing. The proposed deep learning algorithm called *SatCloud* takes these constraints into account, being able to capture and aggregate features at various scales (multi-scale), ensuring that high-level semantic features extracted from high-albedo terrains, such as snow and clouds, are more distinctive. Owing to a variety of performance-enhancing design choices and training techniques, it shows state-of-the-art performance where comparable to other methods, high speed, and robustness to all kinds of situations. Experimentation is carried out on an MPSoC (Multiprocessor System on Chip), demonstrating its feasibility for real-time implementation of the proposed algorithm in orbit.

Keywords: microsatellites, multispectral imaging, computer vision, deep learning, cloud detection, multi-scale feature fusion, on-board system, MPSoC.

Índice general

Índice de figuras

Índice de tablas

Índice de acrónimos

AI Artificial intelligence.

ANN Artificial Neural Network.

ASIC Application-specific integrated circuit.

BOA Bottom-Of-Atmosphere.

CBR Convolution layer + Batch normalization + Relu.

CNN Convolutional Neural Network.

CR Convolution layer + Relu.

DEM Digital Elevation Map.

DL Deep Learning.

DNN Deep Neural Network.

DPU Deep Learning Processor Unit.

EO Earth Observation.

ESA European Space Agency.

FCN Fully Convolutional Networks.

FN False Negatives.

FP False Positives.

FPGA Field-Programmable Gate Array.

GAN Generative Adversarial Networks.

GPU Graphics Processing Unit.

GSWO Global Surface Water Occurrence.

HOT Haze-Optimized Transformation.

IR Infrared.

MCM Multi-temporal Cloud Masking.

MFF Multi-scale Feature Fusion.

ML Machine Learning.

MPSoC Multiprocessor System on Chip.

NIR Near-Infrared.

PCA Principal Component Analysis.

PL Programmable Logic.

PS Processing System.

RF Random Forest.

SHAVE Streaming Hybrid Architecture Vector Engine.

SIFT Scale-Invariant Feature Transform.

SVM Support Vector Machine.

SWIR Short-wavelength infrared.

TN True Negatives.

TOA Top-Of-Atmosphere.

TP True Positives.

TPU Tensor Processing Unit.

VNIR Visible + Near-Infrared.

VPU Vision Processing Unit.

Se han definido todos los acrónimos en inglés para mantener la coherencia con los utilizados en la literatura científica.

Capítulo 1

Introducción y objetivos

La próxima generación de cámaras binoculares iSIM que la empresa Satlantis desarrolla y comercializa para su instalación en microsatélites artificiales dispondrá de sensores multiespectrales de hasta 5 bandas (RGB + NIR + Pancromático) con una alta resolución de imagen. Esta tecnología permitirá ampliar de forma espectacular la aplicabilidad de las cámaras iSIM en el ámbito de la observación terrestre por imagen satelital. La gran resolución espacial que ofrecen las cámaras iSIM, combinadas con su compacto diseño y bajo peso, suponen un logro tecnológico que permite reducir notablemente los costes de las misiones destinadas a la observación terrestre. Sin embargo, son precisamente estas características las que hacen más necesario maximizar y optimizar el procesamiento de las imágenes en la propia cámara (procesamiento onboard). Por una parte, las limitaciones temporales y de ancho de banda de las comunicaciones entre los satélites y las estaciones terrestres aconsejan transmitir únicamente la información relevante obtenida por sus sensores. Por otra parte, el consumo de potencia en microsatélites es un factor determinante para la operatividad de sus instrumentos y el éxito de sus misiones. En las aplicaciones de imagen multiespectral en particular, estos requerimientos cobran mayor importancia debido a la ingente cantidad de datos que se genera durante la observación. Así, en lugar de almacenar y transmitir las imágenes en crudo (cubos multiespectrales), es necesario procesar la información generada para discriminar la información relevante de la que no lo es antes de ser almacenada y transmitida. Es decir, la cámara debe incorporar cierta "inteligencia" para "entender" lo que está viendo y discriminar la información, descargando así al computador del satélite de la realización de estas tareas y proporcionando un gran valor añadido a los productos de Satlantis.

Siguiendo con este propósito, en este trabajo se propone el diseño y la implementación en chip de un sistema embebido de detección en tiempo real de la presencia de formaciones nubosas en las imágenes tomadas por las cámaras iSIM. Este sistema proporcionará la posibilidad de descartar automáticamente estas imágenes antes de ser procesadas, almacenadas o transmitidas, y ello redundará en la optimización de la utilización de los limitados recursos disponibles en los satélites. Para la detección de nubes, tan solo se dispone de tres bandas en el espectro visible y una banda en el infrarrojo cercano (VNIR), siendo éste un reto en el desarrollo, ya que generalmente se utilizan las bandas del infrarrojo de onda corta (SWIR) para este tipo de segmentación debido a la similitud de las características espectrales del VNIR en las formaciones nubosas y en otras superficies de alto albedo como nieve, luminosidad de las ciudades, etc. Para ello, se analizarán diferentes arquitecturas, desde las más clásicas como son los árboles de decisiones, SVM [1], hasta las más actuales basadas en aprendizaje profundo como son las U-Net [2]. Estas últimas tradicionalmente han aportado unos resultados muy superiores, pero debido al alto coste computacional, no se han utiliza-

do en misiones reales a bordo. Esta tendencia está cambiando en los últimos años, puesto que optimizando el diseño de las redes para obtener modelos con menos parámetros y aplicando diferentes técnicas a la generación de los modelos de segmentación, han conseguido ejecutarlas con tiempos de inferencia bajos y en el propio microsatélite [3].

El objetivo principal de este trabajo es el desarrollo de un sistema de detección de nubes en tiempo real adaptado a las características específicas de las cámaras iSIM de la empresa Satlantis y a su procesador embebido. En primer lugar, se debe crear una base de datos sólida de imágenes satelitales multiespectrales cuyas características espectrales sean similares a las utilizadas por la iSIM, ya que en el momento de la realización de este trabajo no se disponen de imágenes tomadas por estas cámaras. Del mismo modo, se deben generar máscaras de nubes para el entrenamiento supervisado de los algoritmos de detección. Una vez se tiene un conjunto de imágenes para la experimentación, se debe realizar un análisis y selección de los algoritmos de detección de nubes que se utilizan en la actualidad teniendo en cuenta que debe ser ejecutado en el propio satélite y en tiempo real. Para ello, se diferencian dos tipos de métodos: los clásicos, también denominados métodos físicos por estar basados en la extracción de información únicamente de las características espectrales de la imagen y los métodos basados en técnicas de aprendizaje automático. Se deberán explorar ambos caminos para determinar la viabilidad de cada una de estas soluciones teniendo en cuenta las limitaciones expuestas anteriormente.

La evaluación de los algoritmos se realizará en dos fases: En primer lugar, se analizarán los métodos clásicos por su simplicidad computacional para verificar hasta qué punto el uso de solo las bandas VNIR, sin SWIR, resulta una limitación en este desarrollo. Posteriormente, se explorarán los algoritmos de aprendizaje profundo (DL) para determinar en que medida se puede mejorar el rendimiento de los sistemas de segmentación de nubes introduciendo las técnicas de convolución de las redes neuronales profundas (DNNs). De esta manera, se podrá tomar una decisión de compromiso (trade-off) entre el rendimiento de la segmentación y el coste computacional. Los algoritmos deberán ser primero implementados por software para poder ser evaluados y en función de los resultados obtenidos se procederá, por una parte, al refinamiento de los algoritmos más simples con el objetivo de mejorar su exactitud sin penalizar en exceso su complejidad computacional y, por otra parte, a la simplificación de los algoritmos más sofisticados sin comprometer en exceso su exactitud y robustez. De este estudio deberán seleccionarse las opciones consideradas más adecuadas para su implementación final. Por último, se deberá desarrollar un prototipo en un MPSoC (Multiprocessor System on Chip) caracterizando el rendimiento del sistema con el objetivo de determinar si el algoritmo seleccionado cumple con los requisitos mínimos necesarios tanto de utilización de recursos como de 'throughput' para su uso como sistema embarcado de tiempo real.

Capítulo 2

Antecedentes y estado del arte

2.1. Tareas de visión artificial

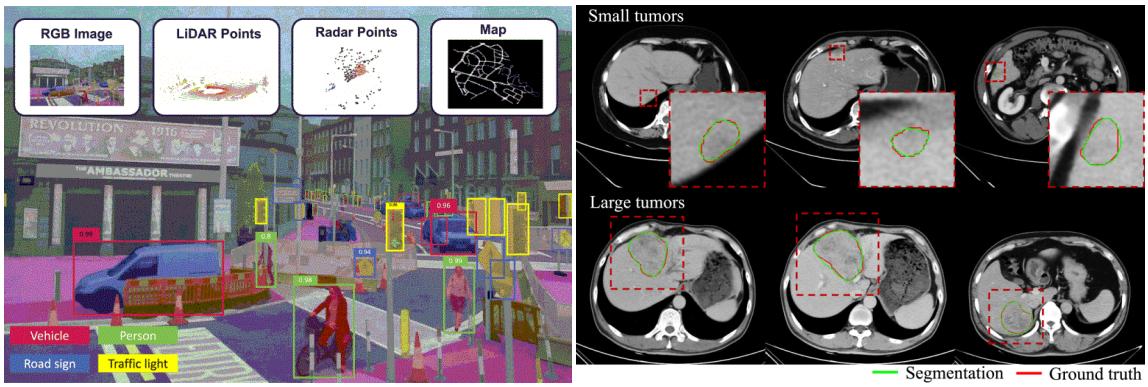
La visión artificial se engloba dentro de un conjunto de tecnologías que permiten adquirir, gestionar y analizar la información visual. En concreto, la visión artificial es una tecnología que permite a las computadoras "ver" y procesar esa información para tomar decisiones. Aplicado al campo de la Inteligencia Artificial (AI), este subcampo tiene como objetivo dar a las computadoras una comprensión visual del entorno para así poder automatizar ciertas tareas que antes solo podían realizar los humanos [4].

2.2. Clasificación de imágenes multiespectrales

La clasificación de imágenes en el campo de la visión artificial consiste en identificar cada imagen a que clase pertenece, mientra que en la segmentación se identifican regiones o segmentos de una imagen pertenecientes a sus respectivas clases. De esta forma, la segmentación de imágenes es una extensión de la clasificación de imágenes en la que, además de la clasificación, se señala dónde está presente el objeto correspondiente, delimitando su frontera. Respecto a las imágenes multiespectrales, estás son un conjunto de imágenes individuales capturadas en longitudes de onda específicas, frecuentemente tomadas por filtros, a través del espectro electromagnético. Las imágenes multiespectrales pueden considerarse como un caso especial de las imágenes hiperespectrales en las que el rango de longitudes de onda recogidas no puede considerarse como continuo. Es decir, en lugar de una medición continua entre un determinado rango de longitudes de onda, las imágenes multiespectrales contienen información en longitudes de onda discretas y específicas. En cambio, las imágenes hiperespectrales pueden tener cientos o miles de bandas, normalmente con una separación entre bandas muy estrecha.

La literatura sobre clasificación/segmentación de imágenes multiespectrales e hiperespectrales es muy abundante. Muchas aplicaciones requieren de su uso y está muy extendido en diferentes sectores. Por ejemplo, la segmentación de imágenes se puede utilizar como paso previo de los modelos de conducción autónoma para conocer en detalle el entorno, detectando personas, automóviles, señales de tráfico, etc [5]. También es muy común su uso en medicina para detectar tumores u otro tipo de enfermedades [6]. Por todo ello, este es uno de los campos de mayor investigación en la actualidad.

En lo referente a satélites espaciales para la observación de la tierra, existen multitud de



(a) Segmentación de un escenario urbano complejo para la conducción autónoma [5].

(b) Segmentación visual de tumores pequeños y grandes [6].

Figura 2.1: Aplicaciones que hacen uso de la segmentación de imágenes.

aplicaciones que realizan la clasificación de imágenes para explotar la capacidad de adquisición de imágenes desde el espacio. Además, muchas de ellas se realizan en tiempo real en el propio satélite. Las aplicaciones más comunes para este uso son, por ejemplo, la agricultura, la detección de distintos tipos de vegetación, el control de fronteras, el control marítimo en alta mar, etc [7, 8]. En este trabajo centraremos nuestra revisión en el caso particular de los algoritmos de detección de nubes.

2.3. Algoritmos de detección de nubes en imágenes satelitales

La investigación en sistemas de detección automática de nubes en imágenes satelitales es un área bastante madura. En las últimas décadas, se han propuesto una gran variedad de métodos para la detección de nubes [9–11] que pueden dividirse en dos clases principales, los métodos más clásicos basados en umbrales derivados de observaciones físicas o empíricas y los basados en técnicas de ML.

Los métodos más clásicos utilizan ciertos índices o características físicas de la reflectividad multiespectral de los píxeles en una imagen para tratar de determinar si corresponden a formaciones nubosas utilizando ciertos valores umbrales, fijos o dinámicos, para realizar la discriminación [12–21]. La principal ventaja de estos métodos es su velocidad de cómputo, ya que los algoritmos en los que se fundamenta su programación son muy sencillos. El principal problema reside en el ajuste de los umbrales y la escasa universalidad de este, si bien pueden funcionar correctamente si se ajustan específicamente para un sensor, sobre todo si se utilizan técnicas de umbrales dinámicos. Como mejora a estos métodos, se puede añadir las características texturales y espaciales de las nubes. La textura de una imagen refleja la variación espacial del brillo espectral de la misma. Las nubes presentan diferentes texturas, pero su textura es estadísticamente distinguible de la textura de los objetos de la superficie terrestre que cubren. Añadir estos métodos de detección de características, agrega un coste computacional muy intensivo, y pueden resultar difíciles de procesar con una latencia reducida [22–24].

Por otro lado, con el avance de la tecnologías de computación, en los últimos años se han aplicado técnicas de aprendizaje en máquinas (ML) en el campo de la detección de nubes [25, 26]. Los métodos de ML toman las bandas espectrales (o alguna función de ellas) como entradas, y a través de algún proceso de optimización derivan un modelo que mapea el espacio de entrada en las clases deseadas, con poca o ninguna dependencia de la física

subyacente. Pueden estar basados en técnicas clásicas de extracción de características para alimentar los predictores o clasificadores (redes neuronales artificiales (ANN) [27], Support Vector Machine(SVM) [1, 28, 29], Random Forest (RF) [26], etc.) o, más recientemente, pueden aplicar técnicas de aprendizaje profundo (DL) basadas en filtros de convolución para tratar de mejorar la universalidad y robustez de los sistemas [30, 31]. Sin embargo, debe tenerse en cuenta que los sistemas de ML necesitan una gran cantidad de datos de entrenamiento y test para el correcto ajuste y validación de los mismos, y que de ello depende en gran medida su capacidad de generalización. Algunas propuestas recientes utilizan métodos de aprendizaje no supervisado como las redes generativas antagónicas (GAN) combinados con autoencoders variacionales para superar esta limitación [32]. En todo caso, dependiendo de los modelos utilizados, la ejecución de la inferencia (predicción) también puede resultar computacionalmente exigente.

2.3.1. Algoritmos basados en umbrales

A principios de la década de 1980 se reconoció la necesidad de enmascarar automáticamente las nubes, debido a las cantidad cada vez mayor de datos para la Observación de la Tierra (EO) [33]. En [34] se descubrió que una técnica por umbrales basada en el análisis de transferencia radiativa era significativamente mejor cuando se comparaba con otras técnicas por umbrales basadas en las radiancias del cielo claro. El método de transferencia radiativa tenía éxito porque calculaba dos operaciones de umbrales sucesivas, tanto en la banda visible como en la infrarroja (IR), lo que lo hacía más robusto que las técnicas que se basaban en un solo umbral. Desde entonces, las técnicas por umbrales han adoptado a menudo la forma de árboles de decisión en los que los nodos se calculan utilizando operadores relacionales, sobre valores de banda o parámetros físicos derivados como la temperatura [12, 35–37].

Algunos algoritmos utilizan luego técnicas de post-procesamiento para refinar los resultados. Por ejemplo, en [38] se propone un método llamado Fmask que utiliza las condiciones de iluminación conocidas de la escena para refinar sus estimaciones mediante medidas geométricas de similitud entre las nubes y las sombras detectadas. En [39] se propuso una versión mejorada de Fmask. Este método se benefició de una de las bandas SWIR de Landsat-8 para aumentar la precisión de las nubes detectadas y actualmente se utiliza para producir máscaras de nubes en las imágenes de Landsat de nivel uno [40]. Además, lo adaptaron para las imágenes de Sentinel-2. En [41] integraron la información del mapa de elevación digital (DEM) en Fmask para reducir la influencia de las variaciones en las bandas cirros y mejoraron su rendimiento en zonas montañosas. Recientemente, se ha desarrollado Fmask 4.0, logrando una precisión del 94,3 % en la detección de nubes en las imágenes de Sentinel-2 [42]. Utiliza la presencia global de aguas superficiales (GSWO) para separar mejor la tierra y el agua, y el DEM.

La Transformación Optimizada de la Neblina (HOT) es un algoritmo muy utilizado para la identificación de nubes de forma manual. Otra de las técnicas para mejorar la detección de nubes es el enmascaramiento multitemporal (MCM) [43]. Se basa en utilizar un conjunto de imágenes de una misma escena, tomadas de forma continua durante un periodo de tiempo finito, para llevar a cabo la detección de nubes. Los enfoques multitemporales suelen ser más potentes que los de una sola escena, ya que la presencia de nubes varía mucho de una adquisición a otra, mientras que la superficie puede suponerse estacionaria en un sentido amplio. Sin embargo, hay dos limitaciones prácticas que suelen dificultar su uso operativo: el acceso al archivo completo de imágenes de satélite y la potencia de cálculo necesaria. En [44], utilizaron la correlación entre dos bandas espectrales de imágenes Landsat para distinguir las nubes finas de las regiones claras. Más recientemente, en [45] añadieron una

prueba de HOT y una banda térmica al algoritmo de MCM para mejorar la detección de neblina, cirros finos y nubes gruesas. También mejoraron el anterior MCM en la detección de las sombras de las nubes añadiendo una banda azul. Aunque su método es factible y eficaz, el rendimiento general de la segmentación de nubes es limitado. También existen más técnicas que se centran en la clasificación de subcategorías particulares de nubes, como los cirros [46, 47], o los cumulonimbus [48].

Las correlaciones espaciales entre píxeles vecinos también pueden utilizarse para mejorar la precisión de la clasificación y suavizar los límites de las nubes detectadas, como se ejemplifica en [49]. En [50], se clasificaron los píxeles nubosos y claros adyacentes a la nube, motivado por la importancia del rendimiento en los límites de la nube.

Como se señala en [51], la base física que subyace a los métodos basado en umbrales permite que puedan ser transferidos directamente entre diferentes instrumentos y conjuntos de datos, asumiendo una resolución y un rango espectral similares. Otra ventaja de las técnicas por umbrales es que no se requiere un conjunto de datos de entrenamiento, cuya elaboración requiere mucho tiempo. Sin embargo, su simplicidad es una limitación significativa en su rendimiento, ya que son incapaces de utilizar relaciones más complejas entre las entradas, y están limitadas a los propios instrumentos que proporcionan las bandas espectrales.

2.3.2. Técnicas de aprendizaje automático

Los algoritmos de ML se diferencian fundamentalmente de los algoritmos 'físicos' de enmascaramiento de nubes en que para generar la función que asigna los datos de entrada a la predicción, lo realizan mediante el análisis estadístico de un conjunto de datos de entrenamiento. El número y la disposición de los parámetros entrenables dentro del modelo empleado dictan el espacio de posibles soluciones, mientras que el conjunto de entrenamiento utilizado dicta los valores de esos pesos y, por tanto, la solución específica encontrada dentro de ese espacio [52]. Por lo tanto, las variaciones observadas en el rendimiento entre los métodos de enmascaramiento de nubes se ven afectadas tanto por la arquitectura del modelo como por los datos de entrenamiento.

Antes de la reciente aparición del DL, la mayoría de los enfoques basados en ML se centraban en píxeles individuales. Los métodos que utilizan valores de píxeles individuales como entradas para los algoritmos de clasificación están ya muy bien estudiados, con clasificadores como los SVM [1, 28, 29, 53], análisis de componentes principales (PCA) [54] y los métodos Bayesianos clásicos [47]. En [55], primero calcularon los índices espectrales de los diferentes tipos de cobertura terrestre, los introdujeron en un RF para producir máscaras de nubes preliminares y, posteriormente, las introdujeron en el algoritmo de segmentación de superpíxeles extraídos mediante muestreo impulsado por energía (SEEDS) para obtener las máscaras finales. Las ANN de un solo píxel también son comunes entre los algoritmos de enmascaramiento de nubes [27, 49]. En [56], compararon seis métodos clásicos: SVM, RFs, ANN, Sen2Cor [57], Fmask y MAJA [58] para la detección de nubes en las imágenes de Sentinel-2 y concluyeron que el SVM era la mejor opción entre estos métodos basados en el aprendizaje. Estos métodos se benefician de su velocidad de cálculo, ya que no se calculan relaciones más complejas entre los valores de los píxeles vecinos. Sin embargo, las correlaciones espaciales que estas técnicas de un solo píxel ignoran, son importantes en la mayoría de imágenes que se toman de la tierra, incluidos los datos para la detección de nubes. En esencia, estas técnicas a nivel de un solo píxel exploran un espacio de entrada de dimensionalidad limitada (principalmente, el número de bandas espectrales del instrumento), y no pueden expresar funciones que contengan un nivel de entendimiento de la imagen superior

al que se obtiene de las técnicas por umbrales derivadas físicamente.

La ampliación de la dimensionalidad del espacio de entrada mediante la inclusión de múltiples píxeles, introduce la posibilidad de establecer relaciones más complejas que las que pueden dar a partir de un solo píxel, y cabe esperar que aumente el rendimiento potencial. Sin embargo, el modelo implementado debe ser capaz de extraer información significativa de este espacio de entrada más amplio. Las características espaciales de una sola banda se han utilizado como entradas para los clasificadores [59], o junto con las características espectrales a nivel de píxel [60]. Las CNN han tenido un gran éxito en las aplicaciones de procesamiento de imágenes debido a su capacidad para aprender eficazmente las características de una escena extensa [61] y, normalmente, se utilizan para tareas de clasificación debido a su gran rendimiento.

Si no se desea la segmentación por píxeles y se acepta la clasificación por áreas, se pueden emplear redes convolucionales estándar (CNN) para clasificar regiones cuadradas de una imagen, como en [62]. Otro método para aproximar la segmentación de nubes como una tarea de clasificación es a través de la construcción de superpíxeles. Los superpíxeles pueden utilizarse para dividir las escenas en un conjunto de regiones que contienen valores de píxeles similares [63]. Esto permite que las regiones sean tratadas como objetos discretos, para ser clasificadas como nube o no nube por una CNN [64–66] o arquitecturas a medida como PCANet [67]. Sin embargo, esto puede causar problemas, ya que las nubes suelen ser amorfas y se fusionan entre sí, lo que significa que el rendimiento es muy sensible a la construcción inicial de los superpíxeles, que no puede actualizarse durante el entrenamiento.

En los últimos años, se han propuesto muchos métodos basados en el DL para detectar nubes, y la fusión de características multiescala se utiliza a menudo en estos métodos. Sin embargo, la mayoría de los métodos existentes son simples, ya que fusionan características a través de la concatenación y la suma de elementos, aunque pueden mejorar la obtención de la información espacial. Además, algunos resultados de detección de nubes no son lo suficientemente precisos cerca de los límites de las nubes. Teniendo en cuenta estos problemas, en [68] propusieron una red de detección de nubes llamada ABNet, que incluye módulos de fusión de características a toda escala y un módulo de predicción de puntos límite. El módulo de fusión de características a toda escala puede optimizar las características y recuperar la información espacial integrando las características de todas las escalas. El módulo de predicción de puntos límite mejora la clasificación de los límites de las nubes clasificando los límites de las nubes por separado.

Recientemente, las Fully Convolutional Networks (FCNs) [69] han demostrado ser prometedoras en los problemas de segmentación de imágenes en una variedad de campos (por ejemplo, U-Net en imágenes biomédicas [70] y SegNet para la segmentación de imágenes naturales [71]). Para ello, aplican tanto las convoluciones como sus inversas (convoluciones de transposición), dando lugar a una transformación directa entre la imagen y la máscara resultante. De esta manera, el modelo es capaz de fusionar las características espectrales a nivel de píxel y las características espaciales. En [72], utilizaron una FCN para el enmascaramiento de nubes. Su algoritmo incluye un paso basado en umbrales como preprocesamiento para la discriminación de nieve y hielo, y utiliza una máscara de nubes existente de Landsat-8 como un previo para mejorar el rendimiento. En una línea similar, [73] utilizan el reescalado para combinar características de diferentes profundidades en una U-Net.

Aunque son prometedores, los enfoques de aprendizaje profundo aún no han demostrado ser compatibles en diferentes tipos de sensores de satélite sin que sea necesario realizar un reentrenamiento [74]. Esta es una ventaja significativa de las técnicas por umbrales, ya que utilizan el conocimiento físico para comprender cómo interfieren las diferentes bandas,

mientras que las arquitecturas actuales de ML no logran integrar este conocimiento físico en su toma de decisiones (aunque esta es un área de trabajo teórico activo en ML [75]). Por lo tanto, a pesar del prometedor rendimiento de los modelos de ML, esto hace que la generalización a los instrumentos con otras bandas espectrales sea un reto. Además, independientemente de la arquitectura, los enfoques de ML supervisado requieren suficientes datos de entrenamiento para que puedan alcanzar el máximo rendimiento. Aunque el aprendizaje por transferencia puede permitir que un modelo entrenado en un conjunto de datos se utilice más fácilmente en un nuevo instrumento, es inevitable que el modelo necesite algunos datos de entrenamiento extra para cada nueva aplicación.

2.3.3. Algoritmos basados exclusivamente en las bandas del VNIR

La segmentación de nubes y sombras es más difícil cuando solo se dispone de un número limitado de bandas espectrales. A pesar de ello, es un área de interés en la investigación, ya que muchos satélites, especialmente los nanosatélites de bajo coste, no tienen las capacidades multiespectrales de, por ejemplo, el satélite Landsat-8 o Sentinel-2, como en el caso de las cámaras iSIM de Satlantis que únicamente disponen de sensores con cuatro bandas en el espectro VNIR. Para resolver este problema, en [30] propusieron un algoritmo basado en DL que utiliza las CNNs para la detección de nubes y sombras usando las bandas del VNIR y realizando el entrenamiento con las imágenes provenientes de los satélites World-View-2 y Sentinel-2. Del mismo modo, en [76] compararon diferentes algoritmos, desde una arquitectura basada en ANN hasta una basada en CNN, para las imágenes con bandas VNIR provenientes del satélite SPOT 6-7. En [77], propusieron un algoritmo que consiste en una FCN, que se entrena a partir de múltiples parches de imágenes Landsat-8. Esta red, conocida como Cloud-Net, es capaz de capturar las características globales y locales de las nubes en las imágenes mediante bloques convolucionales. Para el entrenamiento, utiliza las bandas 2, 3, 4 y 5 provenientes de las imágenes espectrales de Landsat-8, siendo estas bandas las del VNIR.

La detección de nubes y nieve tiene importantes aplicaciones en la teledetección, ya que las nubes y la nieve comparten características de bajo nivel similares debido a sus distribuciones de color consistentes y a los patrones de textura locales. Además, al no disponer de las bandas SWIR, la separabilidad espectral entre nubes y nieve se ve disminuida. Por lo tanto, distinguir con precisión las nubes de la nieve a nivel de píxel en las imágenes de satélite es siempre una tarea difícil con los enfoques tradicionales.

La contribución a la mejora de la precisión de la detección se centra principalmente en la incorporación de componentes complejos o en la modificación de la estructura de la red. Por un lado, algunos métodos introducen componentes para optimizar la red. En [78], los autores añadieron el módulo de fusión de información semántica multinivel para mejorar la precisión de la detección de nubes y nieve utilizando imágenes satelitales para las tres bandas del visible. En [79] crearon un red llamada MFFSNet que toma ResNet101 como estructura principal e introducen un módulo mejorado de fusión de pirámides (PPM) [80]. Esta red podía obtener la relación de las nubes y las sombras y combinar los distintos niveles de características abstractas e información espacial. MF-CNN [81] incorporó un módulo multiescala que podía agregar características abstractas de varias escalas. Al mismo tiempo, la red fusionaba características espaciales de bajo nivel y semánticas de alto nivel para separar con precisión las nubes gruesas de las finas. Por otro lado, algunos enfoques se centran en modificar los extractores de características en el codificador y el decodificador. En [82] tomando como base la red Resnet50 mejorada [83], propusieron una red llamada MSCFF. Antes de obtener un resultado unificado, fusionaba varios niveles de mapas de característi-

cas en la parte de la decodificación. Este sistema promovía la fusión de información semántica avanzada de varios niveles. A partir de ahí, MSCFF mostró una excelente precisión en una variedad de conjuntos de datos satelitales.

2.4. Integración de los algoritmos en un sistema embebido

En los últimos años, el interés por las aplicaciones de aprendizaje automático (ML) ha crecido muy rápidamente. Normalmente, estas aplicaciones se ejecutan en la nube, impulsadas por granjas de unidades de procesamiento gráfico (GPU) que funcionan como un acelerador de hardware global, o en aceleradores de hardware dedicados. La nube proporciona la mayor flexibilidad en todos los casos en los que no hay limitaciones de ancho de banda ni problemas de privacidad. Por el contrario, en aplicaciones de automoción, espaciales o en tiempo real, el paradigma de la nube podría no ser la opción adecuada [84]. Por ello, varias empresas han desarrollado sus propios aceleradores de hardware de ML. Los aceleradores se pueden clasificar en función del tipo de procesador en: unidad de procesamiento de visión VPU, unidad de procesamiento tensorial (TPU), GPU, matriz de puertas lógicas programable en campo (FPGA) y unidad procesadora de aprendizaje profundo (DPU). Los dos primeros son los que mejor rendimiento tienen en términos de potencia por inferencia, ya que han sido concebidos para acelerar las inferencias. En cambio, las GPUs y las FPGAs tienen fines más generales y son las más potentes en términos de capacidad de cálculo. Las DPUs permiten la implementación eficiente combinando la lógica programable (PL) y el sistema de procesamiento (PS) de los MPSoCs.

- **TPU:** La TPU es un innovador acelerador de hardware dedicado a una estructura de datos particular llamada tensores [85]. Los tensores son la base del entorno de TensorFlow desarrollado por Google [86]. Las estructuras estándar y las librerías dedicadas para la GPU y la VPU hacen que los tensores y, por consiguiente, TensorFlow sean herramientas muy potentes en el mundo del ML. La TPU Coral Edge es un ejemplo de acelerador hardware cuyas prestaciones son muy prometedoras, especialmente en la aceleración del procesamiento de imágenes como, por ejemplo, en las arquitecturas basadas en CNN, FCN, etc. Las mejores prestaciones de esta plataforma hardware se alcanzan con la herramienta TensorFlow Lite y la cuantificación de enteros de 8 bits, aunque esta última pueda tener un gran impacto en las métricas del modelo.
- **GPU:** Las GPUs [87] son las más utilizadas para llevar a cabo tanto la inferencia como el proceso de entrenamiento de los modelos típicos de ML. Su potencia de cálculo se debe a la estructura paralela del hardware que computa operaciones entre matrices a una velocidad muy alta. Nvidia y AMD lideran el mercado de las GPUs para el entrenamiento de ML, utilizando respectivamente CUDA Core (Nvidia) y Stream processor (AMD), como se muestra en [88,89]. Además, varios entornos permiten utilizar la potencialidad que ofrecen las GPUs, incluyendo TensorFlow, TensorFlow Lite y PyTorch. Este hardware puede cuantificar el modelo y ejecutar inferencias soportando un amplio rango de precisiones computacionales, por ejemplo, 32 y 16 bits de coma flotante, 16, 8, 4 y 2 bits de enteros. La parte negativa de estas soluciones es que consumen una gran cantidad de energía, por lo que no pueden utilizarse para aplicaciones de bajo consumo.
- **FPGA:** Las FPGAs son soluciones de hardware extremadamente flexibles, que pueden ser completamente personalizadas. Sin embargo, esta personalización representa el cuello de botella para un desarrollo rápido [90]. De hecho, el uso de una FPGA requiere muchos pasos de diseño adicionales en comparación con los circuitos integrados

de aplicación específica (ASICs), incluyendo el diseño de la arquitectura del acelerador de hardware y la cuantificación del modelo, para los enfoques que explotan la representación de punto fijo. Las FPGAs son producidas por numerosas empresas como Xilinx, MicroSemi, Intel. Algunas FPGAs, son también resistentes a la radiación, lo que significa que estas placas pueden tolerar las radiaciones sufridas durante la vida de la misión, como se explica en [91]. La alternativa a implementar los algoritmos directamente en HDL es usando DPUs.

- **DPU:** La DPU es un motor programable optimizado para redes neuronales convolucionales presentado por Xilinx. Se compone de un módulo programador de alto rendimiento, un módulo de matriz de computación híbrida, un módulo de unidad de obtención de instrucciones y un módulo de reserva de memoria global. La DPU utiliza un conjunto de instrucciones especializado que permite la implementación eficiente de muchas redes neuronales convolucionales. El IP de la DPU puede implementarse en la lógica programable (PL) del dispositivo Zynq-7000 SoC o Zynq UltraScale+ MPSoC seleccionado con conexiones directas al sistema de procesamiento (PS). La DPU requiere instrucciones para implementar una red neuronal y ubicaciones de memoria accesibles para las imágenes de entrada, así como datos temporales y de salida. También se necesita un programa que se ejecute en la unidad de procesamiento de aplicaciones (APU) para atender las interrupciones y coordinar las transferencias de datos.

- **VPU:** Las VPUs representan una nueva clase de procesadores capaces de aumentar la velocidad de procesamiento visual como redes neuronales convolucionales (CNN), Scale-Invariant Feature Transform (SIFT) [92] y similares. Los aceleradores más prometedores de esta categoría son las VPU Intel Movidius Myriad. Actualmente, existen dos versiones de este acelerador, el Myriad 2 y el Myriad-X. El núcleo de ambos procesadores es un motor de cálculo que utiliza grupos de vectores especializados de procesadores de texto de instrucción muy largas (VLIW) llamados Streaming Hybrid Architecture Vector Engine (SHAVE) capaces de consumir sólo unos pocos vatios (W) [93]. Myriad 2 tiene 12 SHAVEs, mientras que Myriad-X tiene 18 SHAVEs. El Myriad 2 y Myriad-X muestran un mejor rendimiento cuando aceleran el modelo de CNNs u otras capas soportadas que los procesadores de bajo consumo de uso general. Para reducir el esfuerzo computacional, todos los registros y operaciones del procesador utilizan aritmética de punto flotante de 16 bits. Además, el procesador Myriad 2 ya ha superado las pruebas de radiación en el CERN [94].

El proyecto CloudScout, financiado por la Agencia Espacial Europea (ESA) para la misión Φ-Sat-1 , representa la primera demostración en órbita que se conozca de una CNN aplicada a imágenes hiperespectrales para la detección de nubes [95]. La primera instancia se realizó con una VPU Myriad 2 de Intel a bordo de un CubeSat Hyperscout-2 optimizado para tener un bajo coste, tamaño y eficiencia energética, obteniendo un tiempo de inferencia de 325 ms. El rendimiento del algoritmo muestra una precisión del 92 % y un 1,03 % de falsos positivos considerando que la imagen es nublada a partir de un umbral del 70 %, ya que el modelo, al ser una CNN, solo puede distinguir imágenes completas, es decir, no puede predecir pixel por pixel. Recientemente, presentaron una nueva versión del modelo basada en una U-Net la cual reduce el tiempo de inferencia a 102 ms [3]. Además, el rendimiento del modelo mejora, obteniendo una precisión del 95,1 % y un 0,8 % de falsos positivos fijando también un umbral del 70 %. En este caso, como es una segmentación 'per pixel', se puede analizar el rendimiento pixel a pixel, alcanzando una precisión del 88,4 % y un 5,6 % de falsos positivos. En [96], proponen la misma implementación pero utilizando FPGAs. Los resultados obtenidos en su trabajo muestran que la solución basada en FPGAs se caracteri-

za por un menor tiempo de inferencia, y una mayor posibilidad de personalización, pero a costa de un mayor consumo de energía y un mayor tiempo de desarrollo. Con este método, el tiempo medio de inferencia que consiguen en una FPGA Ultrascale+ ZCU106 es de 141,68 ms.

En [97] realizan una integración completa de un sistema de detección de nubes basado en FPGAs. Teniendo en cuenta el hardware, en [98] muestran cómo implementar un autoencoder teniendo en cuenta la limitación del hardware en un satélite. En [99] implementan una U-Net ligera en una plataforma embebida basada en máquinas RISC haciendo uso de un sistema de compresión de imágenes. En [100] calcularon el tiempo de inferencia de los algoritmos FCN+Cls [78], CloudFCN [101], RS-Net [31] y su propio algoritmo para analizar el tiempo que tarda el sistema en detectar las nubes en una imagen de 512x512 una vez entrenado previamente el modelo. Los resultados muestran que los algoritmos tardan entre 0,054 s - 0,410 s utilizando una tarjeta gráfica de altas prestaciones Nvidia GeForce RTX 2080 Ti, la cual puede llegar a consumir más de 300W.

Como se ha visto, la literatura sobre sistemas de detección automática de nubes en imágenes satelitales es un área bastante madura. Las técnicas clásicas destacan por su simplicidad computacional, aunque presentan dificultades a la hora de detectar nubes en áreas de gran albedo (hielo, nieve, etc.). A pesar de ello, se deberá comprobar, con resultados experimentales, su rendimiento. Los algoritmos de aprendizaje profundo (DL) son los que en principio presentan mejores resultados, pero se deberá estudiar si pueden ser optimizados para que puedan ser implementados en un sistema embebido/embarcado y en tiempo real, es decir, de baja latencia. Las redes basadas en la U-Net son las que mejor rendimiento muestran en los diferentes artículos que se han visto, aunque en la mayoría destacan la necesidad de añadir modificaciones para aumentar el rendimiento y la robustez, o para simplificar el modelo. A pesar de ello, se deberá comprobar si realmente es así, comenzando por la U-Net clásica hasta llegar a un modelo personalizado, comprobando que cada especificación añadida cumple el objetivo y mejora los resultados. Respecto a la integración del modelo final a bordo de las cámaras iSIM, la implementación se debe llevar a cabo en un MPSoC, ya que es el dispositivo que incorpora las cámaras de Satlantis en órbita. Se deberá estudiar si es posible su implementación mediante el uso de DPUs, examinando los tiempos de ejecución de inferencia para determinar si es posible su ejecución en tiempo real para la aplicación requerida. De esta manera, se podría evitar el diseño de IPs específicos en HDL, reduciendo los tiempos de desarrollo y facilitando su integración en la arquitectura de Satlantis.

Capítulo 3

Creación de una base de datos para la experimentación de algoritmos ML con imagen satelital

En este capítulo se describe la selección de la base de datos experimental (conjunto de imágenes satelitales) adecuada que cumpla las características de las cámaras iSIM de Satlantis con el objetivo de utilizarlas para la experimentación de los algoritmos de detección de nubes que se desarrollen. Además, se estudiarán las características espectrales de nubes y otras superficies terrestres de bajo y alto albedo (nieve y hielo) con el objetivo de estudiar la separabilidad espectral.

3.1. Características de las camáras iSIM

Antes de seleccionar una base de datos y ajustar los modelos de entrenamiento, se deben estudiar las características de las cámaras de Satlantis para poder seleccionar la base de datos más adecuada. Además, los modelos de detección de nubes también se deberán optimizar a dichas características, como pueden ser, el número de bandas disponibles, la resolución espacial y el tamaño de la imagen.

Las cámaras iSIM abarcan el espectro visible y el infrarrojo cercano. En la Tabla 3.1 se muestran las características espectrales de los sensores que incorporan actualmente estas cámaras. Respecto a las características espaciales, las imágenes que se reciben de las cámaras tendrán un nivel de preprocesamiento mínimo. La resolución radiométrica, que indica la sensibilidad del sensor, es de enteros de 12 bits, es decir, de 0 a 4096. Las imágenes se enviarán a razón de 2 o 3 segundos, siendo el tiempo total que se dispone para decidir si la imagen contiene una cantidad de nubes suficientemente alta como para que sea descartada.

Tabla 3.1: Características espectrales del sensor iSIM.

Filtro (nm)	**λc**	**λmin**	**λmax**	**anchura**
Azul	492	459	525	66
Verde	559	541	577	36
Rojo	665	650	680	30
NIR	833	780	886	106

El satélite en el que viajarán las cámaras tiene una órbita 'sun-synchronous' o heliosíncronas, lo que asegura un ángulo constante con respecto a la radiación solar (con pequeñas variaciones estacionales).

3.2. Bases de datos de imágenes multiespectrales compatibles

Como se ha mencionado en capítulos anteriores, todavía no se disponen de imágenes reales tomadas por las cámaras iSIM de Satlantis. Es por ello que es necesario buscar una base de datos que se aproxime lo máximo posible a las características de las cámaras iSIM. Los aspectos más importantes a tener en cuenta son el nivel de preprocesamiento y que las bandas espectrales sean lo más parecidas a las bandas de la iSIM. La base de datos debe cubrir una gran cantidad de localizaciones a lo largo de las distintas estaciones del año. Se han buscado bases de datos tanto públicas como de pago, donde primero se han analizado las bases de datos más utilizas en la actualidad hasta seleccionar las potencialmente más adecuadas para las características requeridas.

Las principales bases de datos públicas en la actualidad son las proporcionadas por las misiones Sentinel-2 y Lansat-8, desarrolladas por la ESA y la NASA respectivamente. Ambas disponen de las bandas VNIR, aunque, a diferencia de las de Sentinel-2 que son prácticamente idénticas, las de Lansat-8 no coinciden exactamente con las de las cámaras iSIM. El principal inconveniente de Lansat-8 es su resolución, ya que es de 30m. También hay disponible bases de datos que son accesibles para los investigadores de forma abierta, aunque requieren que los proyectos para los que se hace uso sean verificados por parte de la ESA, como son el caso de los satélites WorldView-3 y WorldView-4. Los dos son satélites comerciales de observación de la Tierra propiedad de DigitalGlobe, aunque el WorldView-4 se encuentra fuera de servicio. Al igual que el Lansat-8, sus bandas del VNIR son aproximadas a las iSIM, aunque en este caso la resolución espacial, que es de 2m, sí es adecuada. Respecto a las bases privadas, las más destacables son Spot 6/7 y PlanetScope. El principal punto a favor de estas bases de datos es que proporcionan imágenes con procesamientos de nivel bajo, similares al nivel de procesamiento que sufrirían las imágenes de iSIM antes de pasar al algoritmo de detección de nubes. Además, las características espectrales son muy similares a las iSIM. En la Tabla 3.2 pueden verse las distintas bases de datos analizadas.

A continuación, del análisis preliminar anterior, se estudian en mayor profundidad las bases de datos más adecuadas para su uso en este proyecto.

3.2.1. Sentinel-2

Sentinel-2 es una misión desarrollada por la ESA dentro del programa Copérnico que tiene como objetivo observar el planeta Tierra para el seguimiento de acontecimientos de interés, como pueden ser, la evolución de los bosques o la gestión de desastres naturales. La misión está compuesta por dos satélites idénticos llamados Sentinel-2A y Sentinel-2B. Los satélites ofrecen imágenes de 13 bandas espectrales que incluyen el espectro visible, el infrarrojo cercano e infrarrojos de onda corta. Son especialmente interesantes las bandas 2, 3, 4 y 8, ya que coinciden exactamente con las cuatro bandas del sensor iSIM. En estas bandas, la resolución espacial es de 10m (Figura 3.1). La resolución radiométrica es de 12 bits. El tipo de órbita es 'sun-synchronous'. Las imágenes tomadas por los satélites son públicas, pero solo ofrecen los productos con nivel de procesamiento 1C y 2A. Los productos de nivel 0, 1A y 1B no están disponibles. Las características de cada uno de los niveles son las siguientes:

Tabla 3.2: Bases de datos analizadas.

	Filtro (nm)	l_c (nm)	l_{min} (nm)	l_{max} (nm)	anchura (nm)	Resolución (m)	Máscara de nubes	Institución	Sistema	Sun-synchronous	Nivel de procesado
iSim	Azul	492	459	525	66	3,2	-	Satlantis	-	Sí	Nivel 1A
	Verde	559	541	577	36						
	Rojo	665	650	680	30						
	NIR	833	780	886	106						
Sentinel-2	Azul	492,7	460	526	66	10	Sí	ESA	Público	Sí	Nivel 1C
	Verde	559,8	542	578	36						
	Rojo	664,6	649	680	31						
	NIR	832,8	780	886	106						
PlanetScope	Azul	490,5	464	517	53	3	Sí	Planet	De pago	Sí	Nivel 1A
	Verde	566	547	585	38						
	Rojo	666	650	682	32						
	NIR	867	846	888	42						
SkySats (16-21)	Azul	482,5	450	515	65	0,72	No	Planet	De pago	No	Nivel 1A
	Verde	555	515	595	80						
	Rojo	650	605	695	90						
	NIR	820	740	900	160						
Lansat-8	Azul	482	452	512	60	30	Sí	NASA	Público	Sí	Nivel 1
	Verde	561,5	533	590	57						
	Rojo	654,5	636	673	37						
	NIR	865	851	879	28						
Spot 6 / 7	Azul	485	450	520	70	8	Sí	Airbus	De pago	Sí	Primario
	Verde	560	530	590	60						
	Rojo	660	625	695	70						
	NIR	825	760	890	130						
Worldview-3	Azul	480	450	510	60	2	No	DigitalGlobe (Maxar)	Validación por parte de la ESA	Sí	Nivel 2
	Verde	545	510	580	70						
	Rojo	660	630	690	60						
	NIR	832,5	770	895	125						
Worldview-4 (fuera de uso)	Azul	480	450	510	60	2	No	DigitalGlobe (Maxar)	Validación por parte de la ESA	Sí	Nivel 2
	Verde	545	510	580	70						
	Rojo	672,5	655	690	35						
	NIR	850	780	920	140						

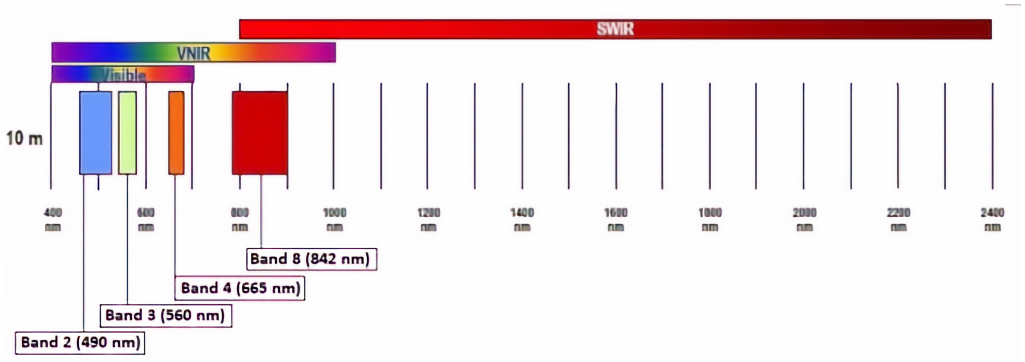


Figura 3.1: Resolución espacial de las bandas 2, 3, 4 y 8 de Sentinel-2 [102].

Nivel 0

Las operaciones de procesamiento de nivel 0 transcurren en tiempo real durante la recepción de los datos para empaquetar los datos brutos adquiridos por las cámaras junto con los metadatos. De esta manera, facilita el procesamiento de niveles superiores.

Nivel 1

El procesamiento de nivel 1 utiliza como entrada los datos producidos por el nivel 0. El procesamiento del Nivel-1A se centra en la descompresión de los paquetes relevantes para la misión. El procesamiento de Nivel-1B utiliza el producto de Nivel-1A y aplica las correcciones radiométricas necesarias. El procesamiento de Nivel-1C utiliza el producto de Nivel-1B y aplica correcciones radiométricas y geométricas (incluyendo ortorectificación y el registro espacial). Las imágenes de Nivel 1C están en unidades enteras de reflectancia de la parte superior de la atmósfera (TOA). Dividiendo cualquier banda por 10.000, se transforman en unidades reales.

Nivel 2

El procesamiento del Nivel-2A incluye una clasificación de la escena y una corrección atmosférica aplicada a los productos de Nivel-1C de la TOA. El resultado principal del Nivel-2A es un producto de reflectancia corregido por el fondo de la atmósfera (BOA).

Como se ha visto, esta base de datos cumple con la mayoría de requisitos salvo el nivel de procesamiento, ya que las imágenes que se van a evaluar tendrán un nivel de procesamiento inferior antes de introducirlas en el algoritmo de detección de nubes. Es por ello que, en caso de optar por esta base de datos, habría que aplicar a las imágenes un preprocesamiento para que se asemejen a las que se reciban por parte de las cámaras iSIM de Satlantis.

3.2.2. PlanetScope

La constelación de satélites PlanetScope de la compañía Planet está diseñada para la observación de la Tierra. Dispone de 4 bandas espectrales que cubren el espectro visible y el infrarrojo cercano (VNIR). En su documentación, indican que las bandas 2, 3 ,4 y 8a de la base de datos de Sentinel-2 son equivalentes a las del PlanetScope, aunque analizando los datos en detalle se observa que no son exactamente iguales, pero las diferencias son muy pequeñas. Así, desde el punto de vista espectral, serían aptas para este proyecto. La mayor diferencia respecto a la iSIM se encuentra en la banda del infrarrojo cercano (NIR). La resolución espacial es entre 3.7-4.1m, aunque se remuestrean a 3m. La resolución radiométrica nativa es de 12 bits, aunque realizan un escalado a 16 bits. PlanetScope también tiene una órbita 'sun-synchronous'. Dispone de productos para los clientes con nivel de procesamiento 1B, siendo este nivel de procesamiento similar al que entrarán las imágenes provenientes de la iSIM al algoritmo de detección de nubes. Otra de las mayores ventajas de esta base de datos es que dispone de máscaras de nubes ya generadas por parte de la compañía. Como punto negativo a destacar es que la base de datos es de pago, aunque la ESA ofrece un formulario de solicitud para proyectos de investigación.

3.3. Selección y organización de la base de datos escogida para la experimentación

Se ha optado por utilizar la base de datos de Sentinel-2 dada su accesibilidad y su total compatibilidad espectral y espacial con las futuras imágenes de iSIM. Sin embargo, tampoco es descartable la de PlanetScope para futuras versiones, ya que ofrece un nivel de procesamiento similar al que estarán las imágenes provenientes de las cámaras iSIM. La base de datos oficial de Sentinel-2 no dispone de máscaras de nubes pero muchos investigadores publican sus propias bases de datos basadas en Sentinel-2 con máscaras de nubes creadas por ellos mismos, bien de forma manual o generadas de forma semiautomática. En el artículo [2], utilizan una base de datos de código abierto denominada WHUS2-CD+ para probar el rendimiento del modelo de detección de nubes para las imágenes de Sentinel-2A. La base de datos consta de 36 imágenes, de las cuales 12 se utilizan para los test. Tiene los principales tipos de cobertura terrestre y las imágenes fueron tomadas entre abril de 2018 y mayo de 2020, cubriendo las cuatro estaciones del año. Las máscaras de nubes de referencia de la base de datos están etiquetadas manualmente a una resolución espacial de 10m. La base de datos solo diferencia las nubes de las no nubes, pero para esta experimentación se ha considerado necesario disponer de etiquetas adicionales para realizar un análisis de separabilidad espectral. Las clases a considerar son: tierra/mar, hielo/nieve y nube. Para ello, se ha realizado manualmente un etiquetado adicional de pixeles de nieve/hielo utilizando una librería de Python llamada labelme [103]. Para la experimentación de los algoritmos y poder comparar su rendimiento, esta base de datos cumple con todos los requisitos descritos más arriba.

Para prevenir el sobreajuste en el entrenamiento supervisado de los algoritmos de ML, es conveniente utilizar imágenes de validación. Es por ello que, a partir del conjunto de imágenes, se han separado 5 para la validación, en concreto, las imágenes 7, 12, 14, 18 y 22 de entrenamiento. Más adelante habrá que comprobar si con estas 5 imágenes para la validación es suficiente. Así, tras realizar esta partición, la base de datos utilizada para la experimentación se compone de 19 imágenes de entrenamiento, 5 imágenes para la validación y 12 imágenes para test.

La base de datos contiene multitud de ficheros (metadatos) para cada imagen. Para este trabajo, solo es necesaria la reflectancia de cada pixel en las bandas de interés. Es por ello, que se ha estructurado la base de datos de otra forma más accesible agrupando las bandas VNIR por un lado y las SWIR por otro. Las bandas SWIR en el entrenamiento final no se pueden utilizar, ya que las cámaras iSIM no disponen de dicha banda espectral pero sí que es interesante estudiarlas para poder observar el efecto que tienen en la separabilidad espectral de las nubes. De esta manera, por cada imagen se ha creado una matriz que contiene las bandas VNIR, otra que contiene las bandas SWIR y otra más con las máscaras de nubes. Las matrices de VNIR y SWIR son de tres dimensiones (filas x columnas x bandas) y la máscara de nubes es una matriz de 2 dimensiones (filas x columnas) en la que '0' representa un pixel no etiquetado, '128' un pixel no-nube y '255' un pixel de nube.

La distribución en carpetas con las bandas VNIR puede verse en la Figura 3.2, siendo equivalente a SWIR, pero sin máscaras de nubes, ya que comparten las mismas. Para validación y test se sigue la misma anotación. Cuando se requieran hacer los análisis con las bandas SWIR, se tendrán que combinar las matrices de las bandas VNIR con las de SWIR.

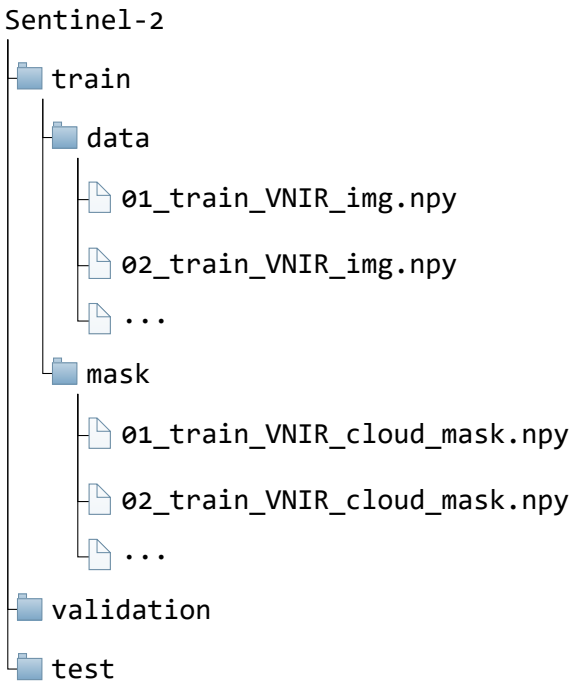


Figura 3.2: Distribución de la base de datos.

3.4. Análisis de características espectrales para la detección de nubes

En los sistemas de clasificación/segmentación de nubes para imagen satelital se suelen distinguir distintas clases de nubes que se clasifican en nubes densas/opacas, cirros (nubes finas, transparentes o semitransparentes, que se forman a gran altura, aproximadamente a 6-7 km por encima de la superficie de la Tierra) y nubes de hielo. Las nubes densas, también llamadas opacas, se caracterizan por una alta reflectancia en la región espectral azul (banda B2). Para evitar la detección de nubes falsas en terrenos de alto albedo, principalmente debido a la confusión entre nieve y nubes, también se utiliza la reflectancia SWIR, en concreto, las bandas B11 y B12. Tanto la nieve como las nubes tienen una alta reflectancia en el azul. La reflectancia de las nubes es alta en el SWIR, mientras que la nieve presenta una reflectancia baja. Para evitar la confusión de nubes de hielo y nieve a gran altura se utiliza la banda de reflectancia SWIR B10, ya que ambas tienen una reflectancia baja en las bandas SWIR B11 y B12.

En el caso de las cámaras de Satlantis, la iSIM no dispone de bandas SWIR. Por ello, los algoritmos deberán tener en cuenta este hecho y deberán ser validados en terrenos de alto albedo. Además de la nieve, hay otras zonas en las que pueden darse estas circunstancias, como son las zonas urbanas luminosas. Los cirros, al ser semitransparentes, solo se pueden detectar con la ayuda de las bandas SWIR, siendo indetectables para iSIM. A efectos de este proyecto, esta limitación no se ha considerado relevante, ya que el objetivo del sistema final es descartar las imágenes con formaciones nubosas opacas que imposibilitan la visión de la superficie terrestre.

Otro aspecto en el que los algoritmos tienden a fallar es en el borde de las nubes, especialmente cuando hay píxeles claros adyacentes a las nubes. En este caso, no es un problema ya que el objetivo del trabajo no es hacer un enmascaramiento de nubes extremadamente preciso, sino estimar de forma robusta el porcentaje de cobertura nubosa de las imágenes adquiridas.

Tabla 3.3: Número de muestras de nieve, nubes y 'resto' utilizadas para en análisis de separabilidad espectral.

Clases	**Número de muestras**	**Porcentaje de muestras (%)**
Nubes	576.936.248	57,19
Nieve	7.446.521	0,74
Resto	424.361.263	42,07

Como se ha mencionado más arriba, para realizar un análisis de separabilidad espectral exhaustivo, se ha añadido la clase de nieve, ya que es uno de los elementos más problemáticos debido al solapamiento espectral que tiene en las bandas VNIR con la clase de nubes. Las clases que se han utilizado son:

- **Nubes:** Todas las muestras de nubes de la base de datos, es decir, todas las nubes que aparecen en las 36 imágenes de la base de datos.
- **Nieve:** Las muestras de nieve clasificadas manualmente. Dichas muestras se han extraído de las imágenes 3 de entrenamiento/validación y 6, 8, 9, 10,11 de test. Se ha evitado coger los bordes con la intención de asegurar que todas las muestras clasificadas como nieve sean realmente de nieve.
- **Resto:** Todos los píxeles que no son ni nubes ni nieve. Se han seleccionado una gran variedad de muestras que contienen todo tipo de terrenos posibles: arbustos, yermo, hierba, tierra de cultivo, agua, bosque, urbano. En concreto, son las imágenes 8, 12, 18, 23 de entrenamiento/validación y 1 de test de las que se han eliminado los píxeles de nubes. No se han añadido más muestras debido a que con estas se abarcan todos los terrenos posibles de la base de datos que se está utilizando, sin que sature la memoria RAM de la computadora.

En la Tabla 3.3 se muestra el número de píxeles que se han escogido por cada clase de nubes, nieve y 'resto'. De la clase nieve, solo hay disponibles 0,74 % del total de píxeles, pero no supone ningún inconveniente debido a que la nieve no tiene tanta variación espectral comparada con las nubes o con el resto de terrenos.

En primer lugar, se han calculado los valores medios de las reflectancias TOA para obtener una representación de las firmas espectrales de las bandas VNIR y SWIR, con el objetivo de extraer alguna característica o tendencia que puedan tener los espectros. En la Figura 3.3 se muestran las firmas espectrales de las clases de nieve, nubes y 'resto'. Posteriormente, se hará un histograma de densidad de probabilidad para ver la variabilidad de cada clase.

Al pasar de la región espectral roja (B4) a la NIR (B8), se aprecia que las muestras de nubes y 'resto' tienen una tendencia creciente. En cambio, la nieve tiene una tendencia decreciente. Esta diferencia no es muy acusada, pero podría servir para tratar de plantear algún método de extracción de características que permitiera aumentar la separabilidad de la nieve del resto. Lo mismo sucede al pasar de la región espectral (B3) a la roja (B4), siendo las nubes y la nieve las que tienen una tendencia creciente y el 'resto' la que tiene una tendencia decreciente.

Como se ha mencionado anteriormente, se comprueba que la reflectancia de las nubes y la nieve en las bandas VNIR es alta comparada con la de 'resto'. En cambio, en las bandas SWIR son las nubes las que solo tienen reflectancia alta en comparación con la nieve y 'resto'.

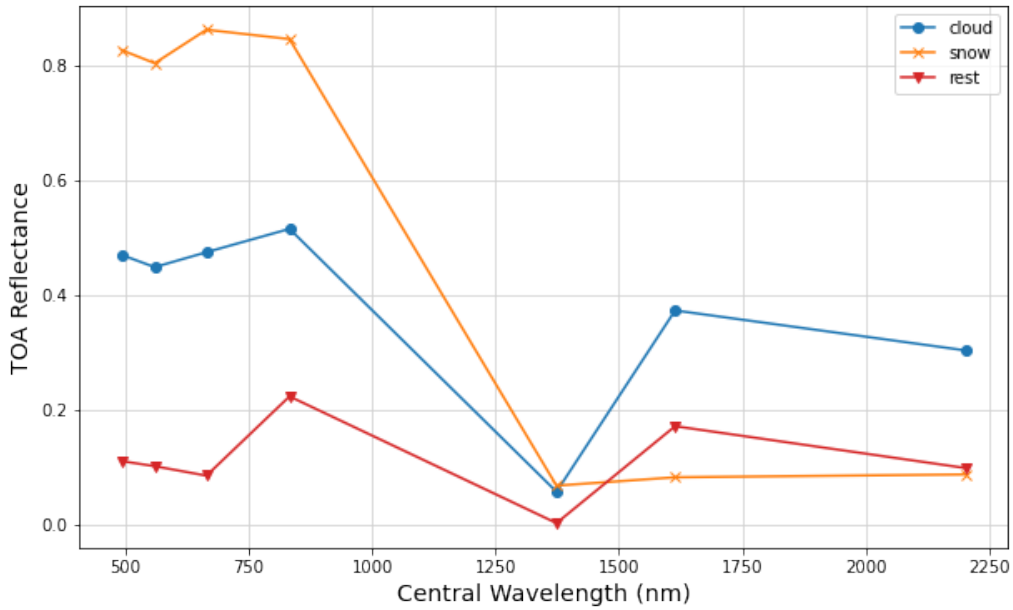


Figura 3.3: Firma espectral de las bandas SWIR + VNIR.

La banda B10 se utiliza para diferencia las nubes cirros, haciendo que en este análisis no se pueda extraer ninguna característica.

Las Figuras 3.4, 3.5, 3.6 y 3.7 muestran los histogramas de reflectancia de las bandas VNIR y SWIR para las clases de nieve, nubes y 'resto' utilizando el mismo número muestras que en el análisis anterior. También se ha dibujado para solo las clases de nieve y nubes con la intención de poder diferenciar claramente ambas clases. Cada intervalo del histograma se ha convertido a una densidad de probabilidad normalizada para poder comparar las clases entre sí. Cada contenedor (bin) mostrará el recuento bruto (counts) dividido por el número total de recuentos y la anchura del contenedor, de modo que el área bajo el histograma se integre en 1.

$$Density = -\frac{counts}{\sum counts \times (bins_n - bins_{n-1})} \tag{3.1}$$

$$\sum (Density \times (bins_n - bins_{n-1})) == 1 \tag{3.2}$$

siendo *Density* el array que conforma el histograma de una clase.

En la Figura 3.4 se puede ver el histograma de las 3 clases para las bandas VNIR. De esta figura, se puede extraer que la separabilidad de la clase 'resto' respecto a nieve y nubes es destacable, sobre todo para las tres bandas del visible, ya que las muestras de 'resto' se concentran en valores de reflectancia bajos. A pesar de ello, se aprecia un solapamiento con las nubes debido a que la mayoría de estas muestras pertenecen a nubes que están cerca de los bordes y a nubes con poca densidad. En la Figura 3.5 se puede apreciar con mayor claridad los histogramas de nubes y de nieve. Como se puede ver, un gran número de muestras están solapadas entre las dos clases. Este hecho puede deberse a que los valores de reflectancia de las nubes varían mucho en un amplio rango junto con las de nieve, aunque estas están un poco más concentradas en valores de reflectancia elevados. El motivo de que las nubes tengan una alta dispersión de reflectancia viene dado por los diferentes niveles de nubosidad que se pueden dar, haciendo que la reflectancia varíe para cada situación. Estos

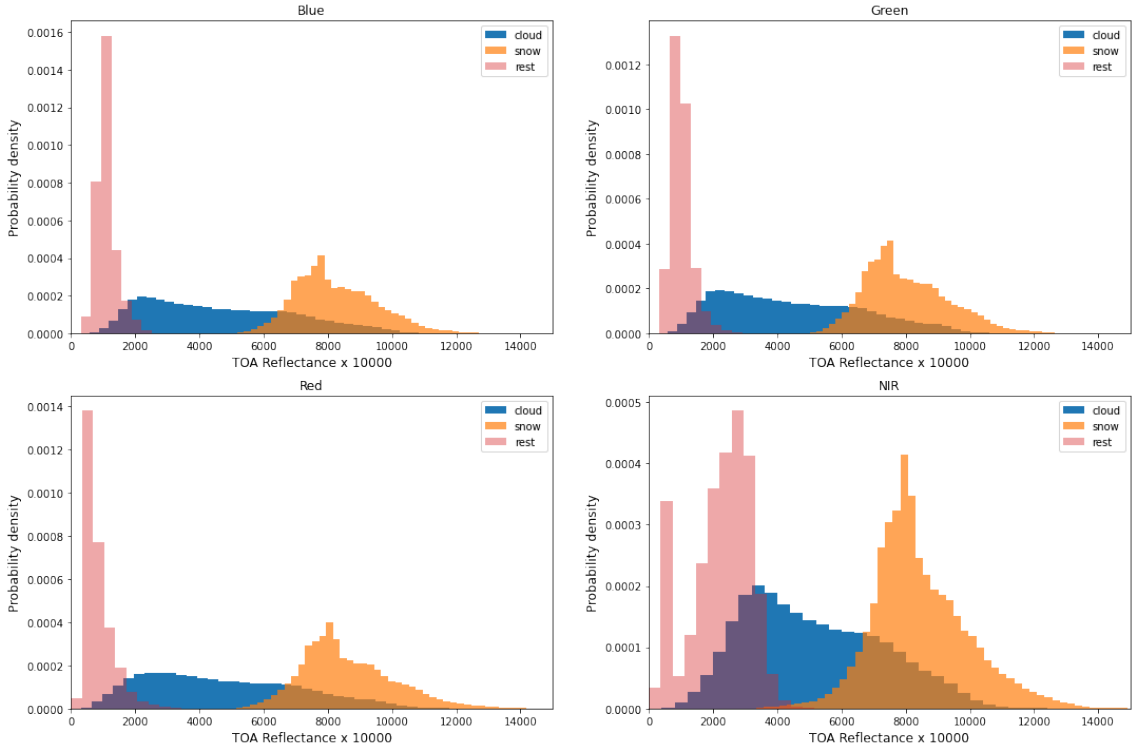


Figura 3.4: Histograma de las reflectancias de las bandas VNIR para las clases de nubes, nieve y resto.

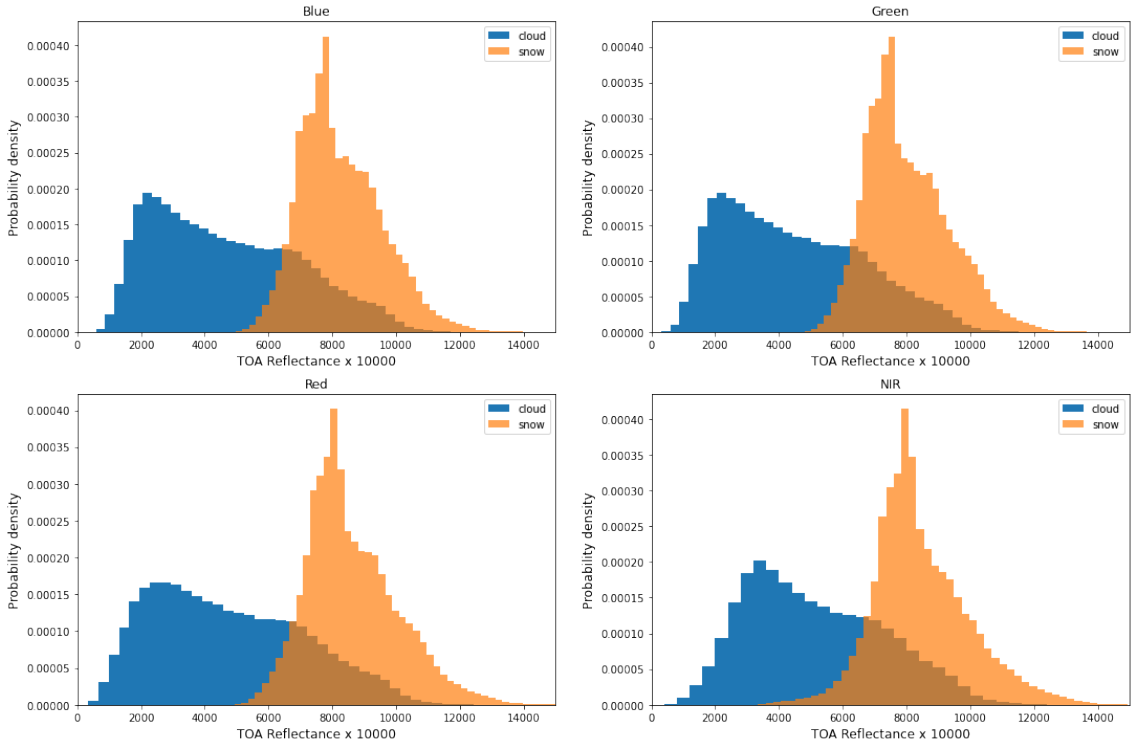


Figura 3.5: Histograma de las reflectancias de las bandas VNIR para las clases de nubes y nieve.

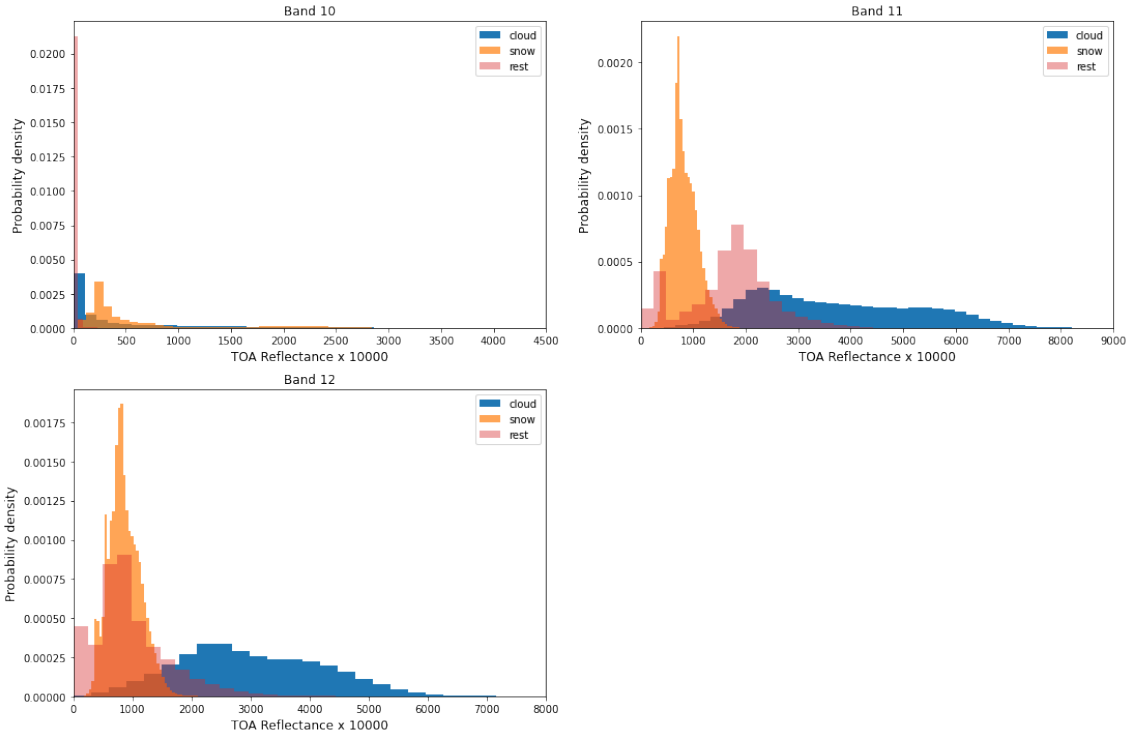


Figura 3.6: Histograma de las reflectancias de las bandas SWIR para las clases de nubes, nieve y resto.

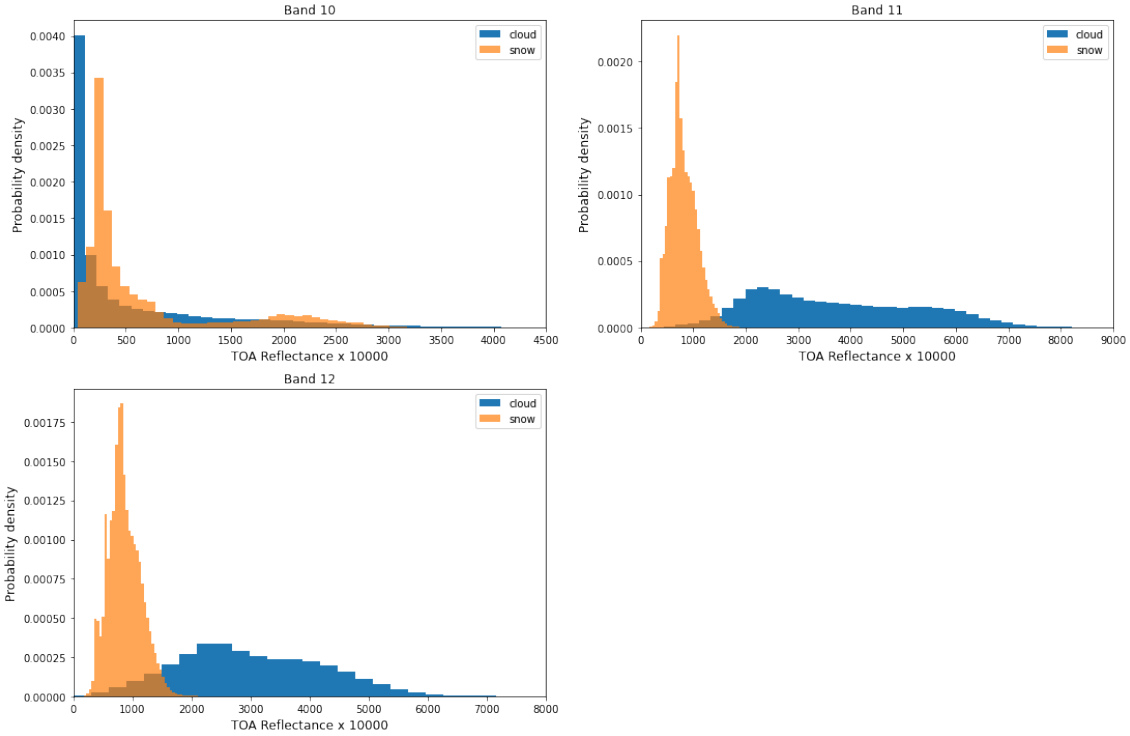


Figura 3.7: Histograma de las reflectancias de las bandas SWIR para las clases de nubes y nieve.

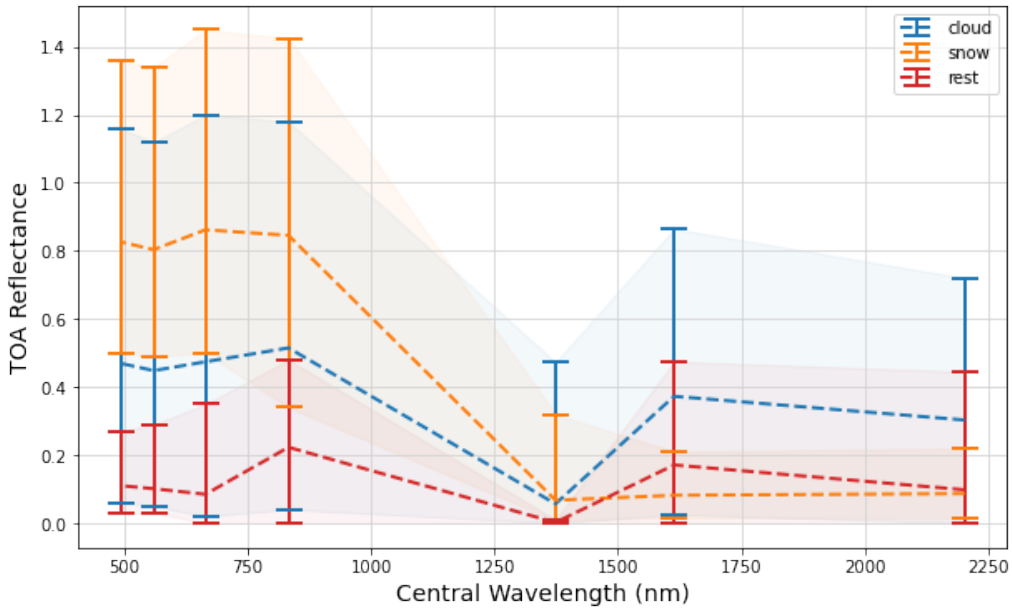


Figura 3.8: Barra de error de las bandas SWIR + VNIR.

solapamientos se estudiarán cuantitativamente en los análisis de separabilidad espectral que se harán en el siguiente apartado.

En el caso de los histogramas para las bandas SWIR (Figuras 3.6 y 3.7), se puede ver que no son útiles para diferenciar la clase 'resto' de nubes y de nieve. En cambio, para diferenciar las clases nubes y nieve son ideales utilizando las bandas B11 y B12 ya que no hay apenas solapamiento debido a que las muestras de nieve se concentran en valores de reflectancia bajos. Como se ha mencionado anteriormente, la banda B10 se utiliza para diferenciar las nubes cirrus, haciendo que carezca de sentido para este análisis.

A partir de los histogramas de densidad de probabilidad, se grafica la barra de errores en la que se representa la dispersión espectral que tienen cada una de las clases. Se han considerado las muestras con una densidad de probabilidad mayor que 10^{-6}. En la Figura 3.8 se representa la media de las firmas espectrales de cada clase con una línea discontinua y el máximo y mínimo que alcanzan con una densidad de probabilidad mayor que 10^{-6}. Como se puede ver, existe un gran solapamiento entre las firmas espectrales tal y como se veía en los histogramas. Las nubes son las que más dispersión tienen, ya que abarcan la mayor parte de la reflectancia. La tendencia que siguen los máximos es similar a la estudiada en las medias a excepción del paso de la región espectral roja (B4) a la NIR (B8) por parte de las nubes, ya que en la media tiene una tendencia creciente y en el máximo tiene una tendencia un poco decreciente.

3.5. Separabilidad espectral

Los índices de separabilidad espectral proporcionan información sobre lo bien que un sistema de clasificación podría diferenciar potencialmente las bandas espectrales de las diferentes clases que se tengan, en este caso, entre nubes y no nubes. También se ha añadido la clase de nieve para observar con detalle este aspecto ya que, como se ha mencionado anteriormente, es el elemento más difícil de diferenciar. Es por ello que en la clase de no nubes ('resto') no se ha añadido nieve. Se han seleccionado todas las imágenes del conjunto de

Tabla 3.4: Separabilidad espectral entre nieve, nubes y 'resto' utilizando la métrica JM para las bandas VNIR.

VNIR	**Resto (%)**	**Nieve (%)**	**Nubes (%)**
Resto	-	1,9970	1,5815
Nieve	1,9970	-	1,2307
Nubes	1,5815	1,2307	-

Tabla 3.5: Separabilidad espectral entre nieve, nubes y 'resto' utilizando la métrica JM para las bandas VNIR y SWIR.

VNIR	**Resto (%)**	**Nieve (%)**	**Nubes (%)**
Resto	-	2,0000	1,9332
Nieve	2,0000	-	1,9938
Nubes	1,9332	1,9938	-

entrenamiento para el análisis, es decir, tanto las de entrenamiento como las de validación.

En la literatura se pueden encontrar varios criterios para evaluar la separabilidad entre clases. En particular, para las aplicaciones de teledetección, la Divergencia Transformada y la distancia Jeffreys-Matusita son las métricas más utilizadas [104]. Dado que el índice JM estima la probabilidad de clasificación correcta, se ha decidido utilizar esta métrica. Dadas 2 clases *i,j* la distancia JM se define mediante la siguiente ecuación:

$$JM_{i,j} = [2(1 - e^{-B_{i,j}}]^{1/2} \tag{2}$$

donde B_{ij} es la distancia de Bhattacharyya, definida por:

$$B_{i,j} = \frac{1}{8}(\mu_i - \mu_j)^T \frac{\Sigma^{-1}}{2}(\mu_i - \mu_j) + \frac{1}{2} ln \frac{|\Sigma|/2}{\sqrt{|\Sigma_i||\Sigma_j|}} \tag{3}$$

siendo μ_i y μ_j, y Σ_i y Σ_j los vectores medios y las matrices de covarianza de las clases *i*, *j* respectivamente, y $\Sigma = \Sigma_i + \Sigma_j$.

El índice JM está delimitado entre 0, solapamiento total de las clases, y 2, separabilidad total. Más concretamente, un valor entre 0 y 1 indica una separabilidad muy pobre; un valor entre 1,0 y 1,9 significa una separabilidad moderada (es decir, las dos firmas son separables, hasta cierto punto) y un valor entre 1,9 y 2,0 implica una buena separabilidad.

Las Tablas 3.4 y 3.5 muestran la distancia JM para cada par de clases. Como se observa, utilizando las bandas SWIR se logra una buena separabilidad para todas las clases, pero con la VNIR tan solo se logra una separabilidad moderada a excepción de las clases 'resto' y nieve que logran una muy buena separabilidad entre sí. En el caso de la VNIR, las clases nieve y nubes son las que más se solapan entre sí. La parte positiva es que ninguna de las clases muestra valores por debajo de 1, haciendo que la clasificación sea posible, aunque previsiblemente con problemas de robustez.

3.6. Evaluación de los algoritmos de clasificación y métricas

Dependiendo de la aplicación y de la interpretación del resultado, la salida del modelo de detección de nubes puede darse en dos formatos diferentes:

- **Máscara de clase:** Clasifica cada píxel como claro y nublado, es decir, a nivel binario como 0 o 1 utilizando un umbral. La principal ventaja de este formato es que es fácilmente extensible a problemas multiclase. Además, pueden generarse máscaras de probabilidad, ya que el resultado del modelo puede tomarse como una probabilidad entre 0 y 1 de que cada píxel sea de nubes.
- **Estimación de la cobertura de nubes:** En vez de tener que fijar un umbral, se realiza el promedio de toda la imagen para estimar la cubierta de nubes total. Este formato permite que el modelo realice estimaciones para las nubes finas y densas, lo que da lugar a porcentajes más precisos a través del promedio.

Para este proyecto se busca el enmascaramiento de las nubes, lo cual el primer formato puede es el más interesante ya que se puede elegir a partir de qué valor de nubosidad se considera una nube mediante la elección del umbral. De este modo, se podrían eliminar los falsos positivos, ya que, como veremos, los algoritmos de clasificación tienden a confundirse más en el límite entre lo que se considera claro y lo que se considera nublado. El segundo método es muy interesante, pero para este proyecto las imágenes muy poco densas no se quieren descartar, es decir, las nubes muy finas se quieren considerar como no nubes, por lo que este método no es adecuado, ya que las tendría en cuenta para estimar la nubosidad total de la imagen.

Para evaluar cuantitativamente el rendimiento de los algoritmos, se han utilizado las métricas propuestas en los artículos relacionados con la detección de nubes [2,105]. El objetivo principal es la comparación de los resultados de los diferentes algoritmos que se diseñen con los propuestos en los artículos y con los propios que se definan en este trabajo. La selección se ha llevado a cabo teniendo en cuenta los artículos de referencia que se han tomado como base de este proyecto, destacando principalmente aquellos que penalizan en mayor medida la detección de falsos positivos (FP), es decir, las imágenes que realmente no son nubladas pero que son detectadas como nubladas. La precisión es un buen indicativo para conocer la calidad global de un modelo, pero el principal parámetro que fija el rendimiento y representa el índice real de calidad es la tasa de FP. La razón principal es que al utilizar esta inferencia en una misión operativa para decidir cuáles de las imágenes merecen ser descargadas a tierra y cuáles pueden ser descartadas en el satélite, los falsos positivos representan la pérdida neta de buenos datos. Por ello, es importante no descartar imágenes que realmente tengan información útil. En [3], proponen como requisito más estricto que el porcentaje de falsos positivos esté por debajo del 1% en las imágenes de todo el conjunto de pruebas. Las métricas vienen definidas por las siguientes fórmulas:

$$OA(Exactitud\ global) = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

$$OP(Precisión\ global) = \frac{TP}{TP + FP} \tag{3.4}$$

$$AP(Precisión\ media) = \frac{1}{2}\left(\frac{TP}{TP + FP} + \frac{TN}{TN + FN}\right) \tag{3.5}$$

$$Recuerdo(recall) = \frac{TP}{TP + FN} \tag{3.6}$$

$$F1 - score = \frac{2 \cdot OP \cdot Recuerdo}{OP + Recuerdo} \tag{3.7}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.8}$$

$$MIoU = \frac{1}{2}\left(\frac{TP}{TP + FP + FN} + \frac{TN}{TN + FP + FN}\right) \tag{3.9}$$

$$kappa = \frac{2 \cdot (TP \cdot TN - FN \cdot FP)}{(TP + FP) \cdot (FP + TN) + (TP + FN) \cdot (FN + TN)} \tag{3.10}$$

donde TP, TN, FP y FN son los números de píxeles de nubes correctamente predichos (True Positives), píxeles sin nubes correctamente predichos(True Negatives), píxeles de nubes erróneamente predichos (False Positives), y píxeles sin nubes erróneamente predichos (False Negatives), respectivamente. OA es un indicador utilizado para evaluar el porcentaje de píxeles correctamente predichos con respecto al total de píxeles, la precisión y el recuerdo (recall) están relacionados con los errores de comisión y omisión, la puntuación F1 es un índice global que combina la precisión y el recuerdo, y IoU es la relación entre la intersección y la unión de las regiones de nubes verdaderas y predichas. OA, F1-score e IoU son indicadores exhaustivos que pueden evaluar el rendimiento de la clasificación con mayor precisión que la precisión y el recuerdo por sí solos. El Coeficiente kappa de Cohen es una medida muy buena para problemas de múltiples clases y para clases desequilibradas. Básicamente, indica cuánto mejora el rendimiento de un clasificador con respecto al rendimiento de un clasificador que simplemente adivina al azar según la frecuencia de cada clase.

El desequilibrio entre clases es un tema ampliamente estudiado, ya que es un punto crítico a la hora de abordar problemas de clasificación. Por lo general, suele afectar a los algoritmos de clasificación en su proceso de generalización de la información, perjudicando a la clase con menor presencia de píxeles. Existen diferentes estrategias para tratar de solventarlo o de reducir su impacto en los resultados finales. Una de ellas es evaluar los algoritmos mediante métricas que no penalicen en exceso el desequilibrio entre clases, como es el caso del Coeficiente kappa de Cohen. Además se pueden utilizar para la etapa de validación de los algoritmos en el entrenamiento, en lugar de utilizar la precisión o la función de pérdidas directamente. En los siguientes capítulos se verán otras estrategias para solventar el problema del desequilibrio entre clases modificando la base de datos de entrenamiento.

Capítulo 4

Algoritmos de detección de nubes clásicos

En este capítulo, se van a verificar experimentalmente las limitaciones de utilizar algoritmos clásicos (umbrales de reflectancia) cuando no se dispone de bandas SWIR. De esta manera, se pretende verificar los resultados del análisis previo de separabilidad espectral realizados en el capítulo anterior para poder extraer conclusiones sobre este tipo de algoritmos y justificar con datos experimentales su funcionamiento.

4.1. Algoritmos de detección de nubes basados en umbrales

Para el enmascaramiento de nubes, Sentinel-2 utiliza un algoritmo basado en umbrales píxel a píxel, es decir, sin tener en cuenta las características espaciales, y se basa en la reflectancia de la banda azul [102]. Para evitar la falsa detección, debida principalmente a la confusión nieve/nube, utiliza también la reflectancia SWIR. En concreto, utiliza las bandas espectrales B1 o B2, B10, y B11 o B12. El procesamiento se realiza con datos muestreados a una resolución de 60 m para todas las bandas espectrales y las clases que diferencia son las nubes opacas, nubes densas y nubes cirro.

Para minimizar los errores que se producen como resultado de un valor atípico (como un error geométrico o la presencia de un objeto altamente reflectante) y para garantizar que cualquier píxel marcado como libre de nubes está realmente libre de nubes, proponen llevar a cabo un paso de filtrado adicional basado en la morfología. Con este filtrado, eliminan la presencia de píxeles espacialmente aislados (erosión) y rellenan los huecos entre las nubes (dilatación).

La operación de morfología la aplican por separado a la máscara de nube opaca y a la máscara de nube de cirros, y el tamaño preciso del área de píxeles vecinos involucrados en la dilatación y en la erosión lo toman del grupo de píxeles cercano asociado. Si después de las operaciones de morfología, un píxel es denso y cirro, prevalece la máscara de nube densa.

Después de todos los pasos de filtrado, la máscara de nube está disponible a una resolución espacial de 60 m. Luego vuelven a muestrear a resoluciones espaciales de 10 m y 20 m para cada banda espectral correspondiente. El remuestreo que hacen no es una transformación geométrica sino una interpolación radiométrica.

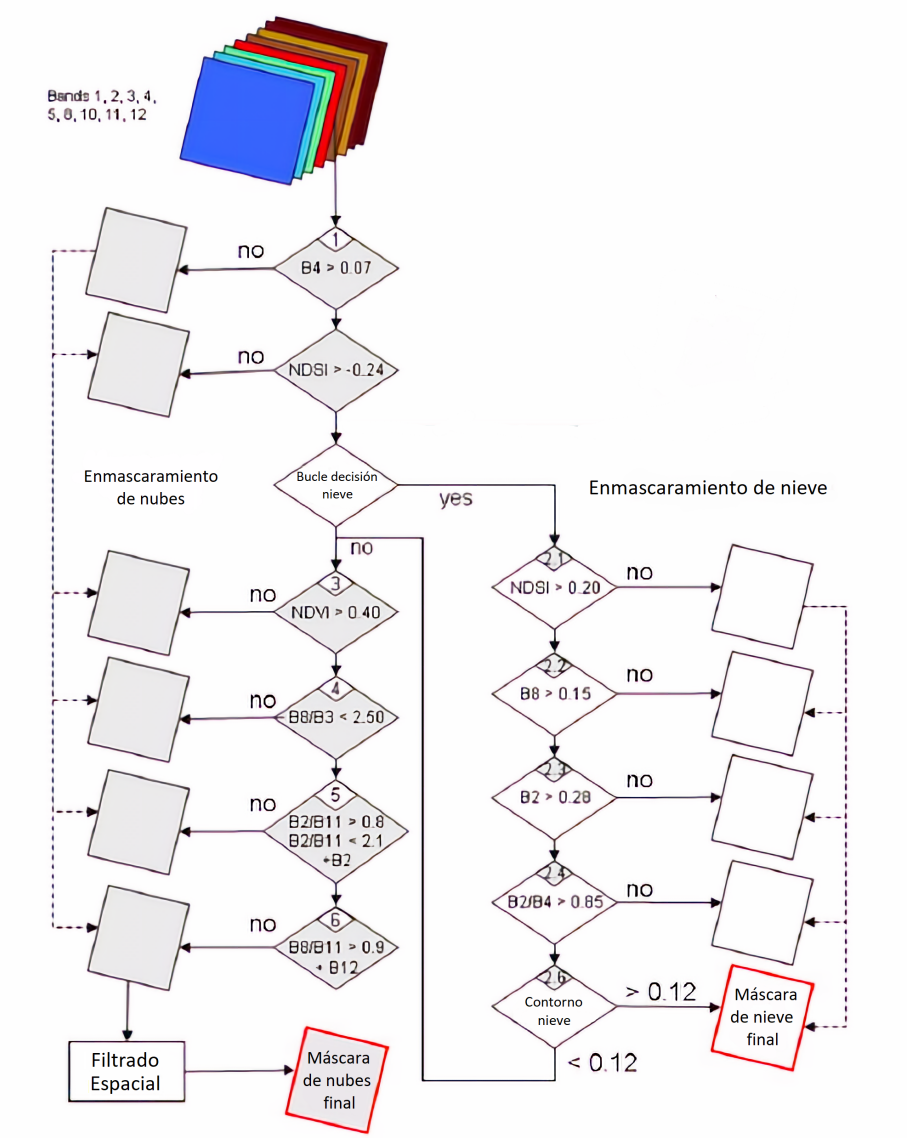


Figura 4.1: Algoritmo de detección de nubes/nieve de Sentinel-2 [102].

4.1.1. Procesamiento Sentinel-2

Como se ha visto, en el procesamiento de nivel 2A realizan la detección de nubes, nieve, cirrus, sombras de nubes y la generación del mapa de máscaras. A pesar de que utilizan las bandas SWIR, se ha analizado el proceso que siguen, ya que la metodología utilizada puede ser aplicable a este proyecto y, además, alguna de las características puede que sea interesante de incorporar. Los pasos de la secuencia del algoritmo de detección de nubes/nieve se presentan en la Figura 4.1.

El paso 1a, descarta que sean nube los píxeles que están por debajo del umbral de 0,07. Los pixeles con un umbral por encima de 0,25 los considera que tienen una probabilidad igual a 1,0 de que sean de nubes y los que están entre 0,07 y 0,25 considera que son potencialmente nubes con una probabilidad de entre 0,0 a 1,0. Estos dos últimos pasan al cálculo del índice de la diferencia de nieve normalizada utilizando la banda visible B3 y la banda SWIR B11.

En el siguiente paso, detectan los píxeles de nieve y crean una máscara de confianza de

nieve utilizando las bandas B2, B3, B8 y B11. Los pasos los dividen en 4 partes y van descartando los píxeles que no son de nieve. En cada paso, se basan en el umbral de una banda. El problema ya surge en el primer paso donde utilizan la banda SWIR B11, haciendo que no sea aplicable para la iSIM. Habría que buscar alguna alternativa de, saltando el primer paso, poder crear la máscara de nieve. Posteriormente, como se ha mencionado, utilizan los umbrales de VNIR. Por ejemplo, en el siguiente paso, se basan en el umbral de la B8. Por debajo de 0,15 se considera que no es un pixel de nieve, por encima de 0,35 consideran que es un pixel de nieve (probabilidad de 1,0) y entre 0,15 y 0,35 consideran que puede ser potencialmente de nieve (probabilidad entre 0,0 a 1,0). Con este procedimiento y adaptando los umbrales para las bandas B2 y B4 crean la máscara de nieve. Para completar el paso 2, realizan un proceso de eliminación de nubes falsas en los límites de una región nevada en la que un píxel mixto (nieve y suelo) podría detectarse como 'nube' en el algoritmo de detección de nubes. Para ello, se basan en la reflectancia de la banda SWIR B12. El paso 2 termina considerando la probabilidad de nieve del píxel. Si está por debajo de un valor umbral, los píxeles pasan al siguiente paso de detección de nubes, mientras que los píxeles que tienen una probabilidad de nieve superior a este umbral, se clasifican como píxeles de 'nieve'.

En el tercer paso calculan el índice de vegetación de diferencia normalizada basándose en las bandas B8 y B4 para descartar píxeles que no son de nubes y para clasificar los píxeles de vegetación. En el cuarto paso eliminan la vegetación senescente altamente reflectante mediante pruebas de filtrado de la relación de reflectancia de las bandas B8 y B3. Esta prueba se basa en el hecho de que la vegetación senescente es altamente reflectante en el NIR y más altamente reflectante en la vegetación verde debido a la pérdida de clorofila. La relación de reflectancia de las bandas B8 y B3 es mayor para la vegetación que para las nubes u otras características de la escena. De esta manera, aunque también se use para la clasificación de la vegetación, también se consigue descartar píxeles que no son de nubes.

Los pasos 5 y 6 están enfocados en la clasificación de tipos de suelo, agua, rocas y arena del desierto haciendo uso de las bandas B2, B8 y B11. Por último, hacen un paso opcional relacionado con el filtrado espacial. Para esta etapa debe estar disponible las máscaras de confianza de nubes y nieve. Este filtrado opcional tiene en cuenta el ligero desregistro de las bandas espectrales a la altura de las nubes. De hecho, las bandas espectrales se co-registran a nivel del suelo con el uso de un Modelo Digital de Terreno (DEM). El filtrado espacial también ayuda a reducir la detección de nubes falsas que se producen en los bordes de regiones muy contrastadas, como son los contornos de los ríos o las costas. Una vez completado el proceso, hacen dos pasos más. Primero realizan la detección de nubes cirrus a partir de la banda B10 y posteriormente detectan las sombras de las nubes basándose en los mapas de Kohonen.

Por lo tanto, de este estudio se puede extraer que, en caso de realizar la clasificación basándose únicamente en umbrales, se deben plantear distintas fases de clasificación, siendo el primer paso la detección de nieve. Además, en la mayoría de los pasos hacen uso de bandas SWIR, haciendo que no sea viable aplicarlo directamente en la iSIM, aunque se podría incluir alguno de los pasos intermedios que se hacen usando solamente las bandas VNIR como prepocesamiento del algoritmo. También se puede ver que hacen uso de algún tipo de filtrado espacial simple, aunque sin utilizar las bandas SWIR, debería ser más complejo. En todo caso, utilizando las bandas SWIR o no, sí que se ve que es necesario aplicar algún tipo de procesado para extraer las características espaciales de las imágenes multiespectrales.

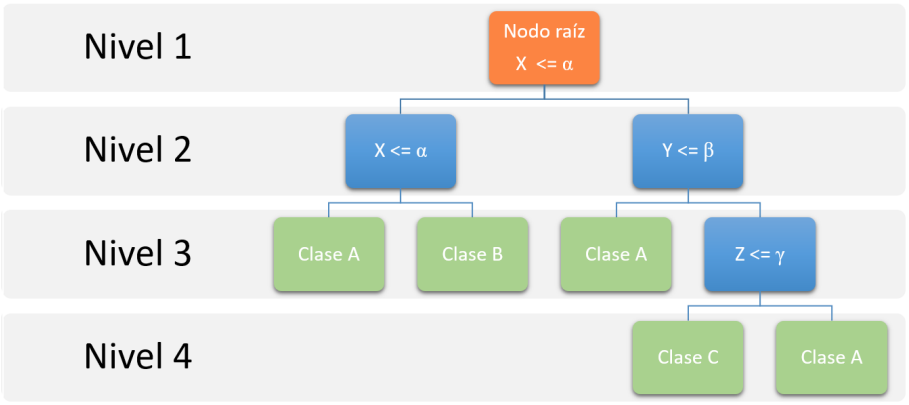


Figura 4.2: Diagrama del árbol de decisiones.

4.2. Algoritmos de detección de nubes basados en técnicas de aprendizaje automático

Las técnicas de clasificación, a las que pertenecen los algoritmos de detección de nubes, son algoritmos de aprendizaje automático (ML) que son capaces de aprender sin necesidad de tener que suministrar una ecuación predeterminada como modelo, es decir, aprenden directamente de los datos. A medida que se va aumentando el número de muestras disponibles para el aprendizaje, se mejora el rendimiento de los algoritmos, ya que se estarán creando modelos más ajustados y precisos.

Las técnicas de ML supervisado predicen las salidas a partir de un modelo obtenido con muestras de entrada y las etiquetas (clases) a las que pertenecen dichas muestras. En ocasiones, no se conocen las posibles clases de salida a las que pueden pertenecer las muestras de entrada. Para ello, se recurre a las técnicas de ML no supervisado, ya que son capaces de descubrir patrones en los datos de entrada.

En esta sección, se van a analizar los algoritmos de ML supervisado más sencillos y con menor coste computacional para la detección automática de nubes. Por un lado, se han escogido los árboles de decisión para seguir analizando los algoritmos basados en umbrales, pero que en este caso, el ajuste de los umbrales y la estructura de decisión en sí no se establece a priori basándose en el conocimiento físico del problema (datos), sino que se aplican técnicas de aprendizaje automático para su definición. También, se analizarán las redes neuronales artificiales (ANN) de pocas capas debido a que son uno los algoritmos de ML más utilizados, sin llegar a la complejidad de los algoritmos de aprendizaje profundo (DL) compuestos de múltiples capas.

4.2.1. Árbol de decisiones

Los árboles de decisión son una técnica de aprendizaje automático basada en umbrales. El algoritmo aprende a predecir la respuesta a partir de un conjunto de datos de entrenamiento. Las imágenes se introducen píxel a píxel. Se componen de dos tipos de elementos: nodos y ramas. En cada nodo, se evalúa una de las características de los datos para dividir las observaciones en dos rama. En el proceso de entrenamiento, los árboles de decisión se construyen evaluando recursivamente diferentes características y utilizando en cada nodo la característica que mejor divide los datos. La estructura del árbol de decisión es similar a los diagramas de flujo con condiciones. El nodo superior es es nodo raíz y es el que inicia el

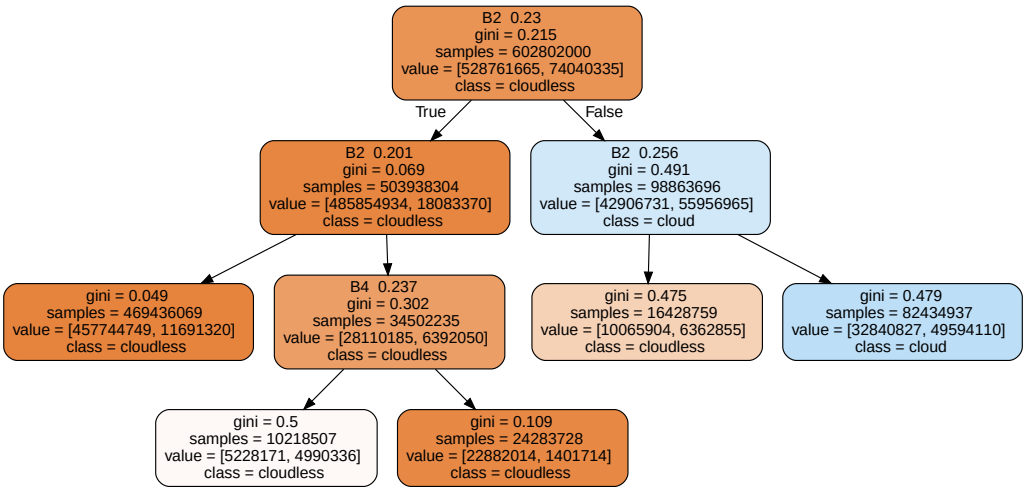


Figura 4.3: Árbol de decisiones con bandas VNIR.

gráfico. Teóricamente, en este nodo se debe evaluar la variable que mejor divide los datos. En los nodos intermedios es donde se evalúan las variables y en los nodos finales se realizan las predicciones de una clase. En la Figura 4.2 puede verse de manera gráfica el esquema general de un árbol de decisión.

Para el entrenamiento y test se ha utilizado la base de datos WHUS2-CD+ con la separación que se ha hecho para tener un conjunto de entrenamiento, validación y test. Debido a las limitaciones del hardware utilizado, para entrenar el árbol de decisiones se han utilizado las 5 imágenes de validación para no tener tantas muestras de entrenamiento. De esta manera, se ha entrenado con 602.802.000 píxeles. Para realizar los test, se han hecho uso de las 12 imágenes disponibles. En la primera prueba, se realiza la clasificación entre nubes y no nubes sin separar previamente los píxeles de nieve para las bandas VNIR.

En la Figura 4.3 se observa gráficamente la selección de umbrales que ha tomado el árbol de decisiones. Ha tomado 4 niveles de ramificación y se ha limitado a utilizar las bandas B2 y B4 para realizar la clasificación, pero en realidad solamente tiene en cuenta que la banda B2 sea mayor que 0,26 para considerar que el píxel sea de nube. Si se compara con el algoritmo de Sentinel-2, se ve que en el paso 1a, asegura que para muestras mayores que 0,25 en la banda B4 es muy probable que el píxel sea de nube, pero todavía no lo diferencia de la nieve. Por lo tanto, el árbol de decisiones toma mayormente como nubes los píxeles de nieve. Este hecho se ve reflejado en los resultados, en donde el IoU da un valor muy pequeño (Tabla 4.1).

Para forzar al árbol de decisiones a distinguir las nubes de la nieve, se ha decidido diseñar un árbol de decisión concatenado en dos fases: primero, se ha entrenado un árbol para clasificar la nieve y las nubes del resto. Seguidamente, se ha entrenado un segundo árbol para clasificar las nubes de la nieve. De esta manera, se han introducido 482.241.600 píxeles en las que los píxeles de nieve y nubes pertenecen a la misma clase. Una vez clasificados, los pixeles de la clase nieve y nubes se han introducido en otro árbol de decisiones para poder diferenciarlos entre sí. Como de la clase nieve solo se disponen 7.446.521 píxeles, se ha introducido el mismo número de píxeles de nube para evitar que una clase tenga más peso que la otra. Otra opción era dar menos peso a la clase con mayor número de píxeles, pero se ha comprobado que realiza la misma elección de umbrales.

En este caso, para diferenciar las nubes/nieve de “resto” se observa que el árbol utiliza las bandas B2 y B4 (Figura 4.4) y para separar la nieve de las nubes utiliza las bandas B2, B3 y B8 (Figura 4.5). Con esta combinación se consigue mejorar los resultados, pero no

Tabla 4.1: Resultados obtenidos al introducir las bandas VNIR.

img	OA (%)	OP (%)	AP (%)	Recall (%)	FI-score	IoU	MIoU	kappa
1	99,42	77,13	88,32	42,33	0,5466	0,3761	0,6851	0,5439
2	98,59	82,38	90,58	42,13	0,5575	0,3864	0,6861	0,5511
3	99,24	89,88	94,59	43,91	0,5900	0,4184	0,7054	0,5866
4	87,94	99,92	91,02	73,11	0,8443	0,7306	0,7757	0,7497
5	98,08	93,23	95,73	60,60	0,7345	0,5804	0,7804	0,7250
6	92,40	24,52	60,95	40,72	0,3060	0,1807	0,5517	0,2684
7	85,11	22,50	60,14	67,32	0,3372	0,2028	0,5240	0,2762
8	87,06	58,94	76,64	72,90	0,6518	0,4835	0,6681	0,5734
9	51,38	3,24	51,36	86,35	0,0624	0,0322	0,2690	0,0273
10	55,74	-	-	-	-	-	0,5574	-
11	67,54	30,35	60,38	66,06	0,4159	0,2626	0,4478	0,2318
12	20,66	-	-	-	-	-	0,2066	-
media	78,60	58,21	76,97	59,54	0,5046	0,3654	0,5714	0,4533

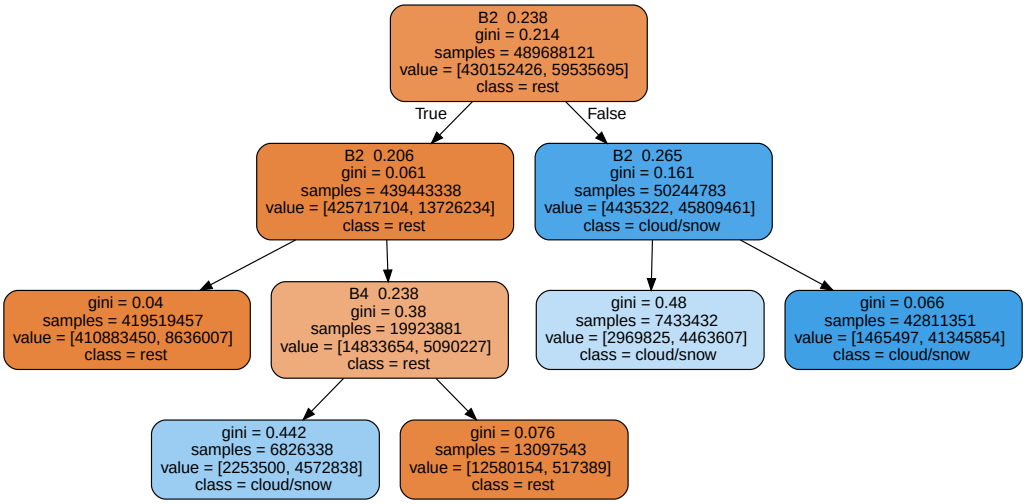


Figura 4.4: Árbol de decisión con bandas VNIR para las clases resto y nieve/nubes.

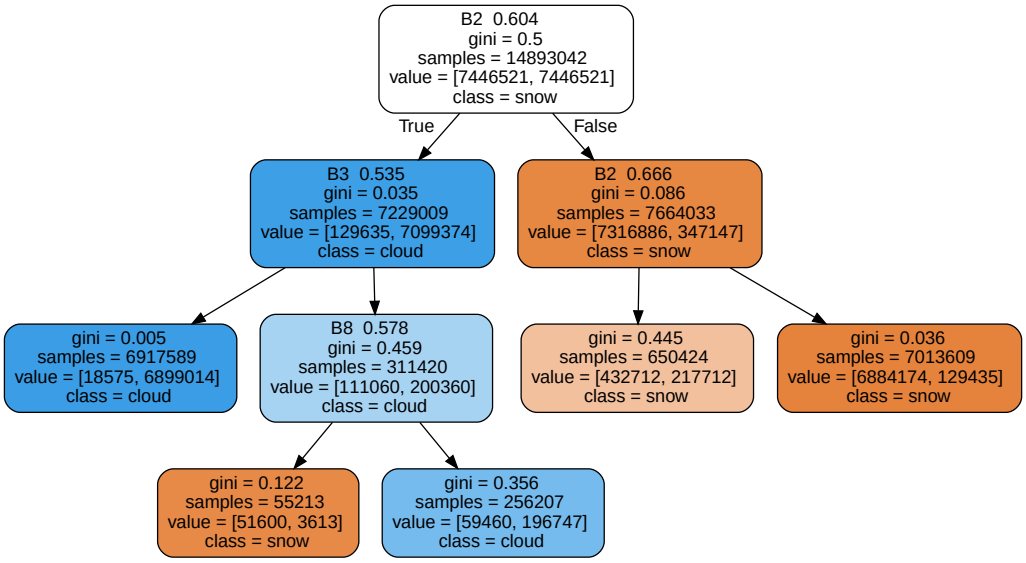


Figura 4.5: Árbol de decisión con bandas VNIR para las clases de nieve y nubes.

Tabla 4.2: Resultados obtenidos al introducir las bandas VNIR para clasificar la nieve y las nubes tras diferenciarlas del resto.

	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
media	82,95	60,64	77,14	65.72	0,5129	0,3995	0,5957	0,4721

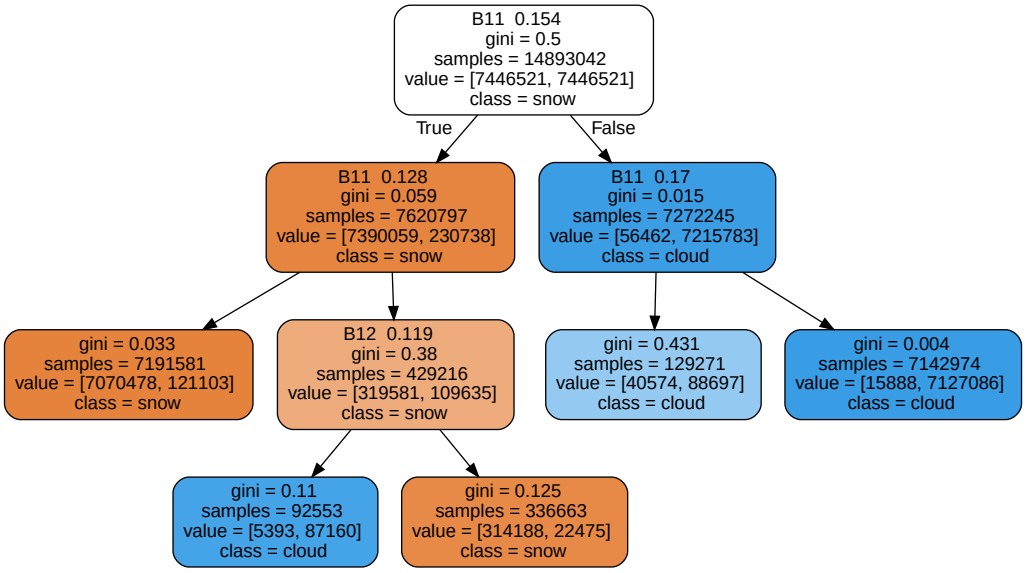


Figura 4.6: Árbol de decisión con bandas SWIR para las clases de nieve y nubes.

de una manera notable. Para el cálculo de las métricas, se ha utilizado una selección de imágenes de test, donde se han diferenciado los píxeles de nieve del resto. Es por ello, que el valor de las métricas se muestra directamente con la media (Tabla 4.3). De estos resultados se puede concluir que utilizando directamente las firmas espectrales de las bandas VNIR, no es posible diferenciar de forma consistente/robusta la nieve de las nubes utilizando un método basado en umbrales sin haber extraído previamente alguna característica espectral o espacial.

Para comprobar el potencial de los árboles de decisión que incorporan información en las bandas SWIR, se han introducido al árbol de decisión las bandas VNIR junto con las del SWIR. Incorporando directamente las reflectancias de las bandas SWIR en el algoritmo, no se han obtenido buenos resultados. Esto se debe a que por sí solo no prioriza las bandas SWIR para diferenciar las nubes de la nieve. Primero utiliza las bandas VNIR y, después de descartar un número de pixeles, utiliza las bandas SWIR haciendo que los resultados que se obtienen sean incluso peores al de utilizar solo las bandas VNIR. Para tratar de solucionar este problemas, ya que las bandas B12 y B11 son ideales para diferenciar la nieve de las nubes, se ha entrenado otro árbol múltiple en dos fases concatenadas: el primer árbol debe clasificar la nieve/nubes de "resto" utilizando las 4 bandas VNIR y, a continuación, se diferencian las nubes de la nieve utilizando las bandas SWIR B12 y B11, es decir, igual que en la prueba anterior, pero en el segundo árbol de decisiones se utilizan las dos bandas SWIR. El número de píxeles que se ha introducido para el entrenamiento han sido los mismos de nieve que de nubes, 14.893.042 en total. Realizándolo de esta manera, se consiguen unos resultados muy superiores al de utilizar solo las bandas VNIR, como se puede ver en la Tabla 4.3.

Tabla 4.3: Resultados obtenidos al introducir las bandas SWIR B12 y B12 para clasificar la nieve y las nubes tras diferenciarlas del resto con las bandas VNIR.

	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
media	92,85	85,27	89,84	76,61	0,6797	0,5212	0,7961	0,6554

4.2.2. Redes neuronales artificiales (ANN)

Las redes neuronales son un modelo computacional basado en la estructura interconectada de las neuronas en el cerebro. Su estructura se divide en capas conectadas entre sí. La información de entrada atraviesa la red neuronal hasta producir los valores de salida. Se puede entrenar una red neuronal para que reconozca patrones, clasifique datos o pronostique sucesos futuros. Las redes neuronales artificiales (ANN) surgieron como un intento de descubrir la arquitectura del cerebro humano para realizar tareas en las que los algoritmos convencionales habían tenido poco éxito. Normalmente los modelos están asociados con un algoritmo de entrenamiento o una regla de aprendizaje. Las ANN están formadas por neuronas (nodos) conectadas entre sí formando capas. Las redes constan de una capa de entrada, una o varias capas ocultas y una capa de salida. Las redes neuronales que constan de muchas capas se denominan redes profundas o deep neural networks. Existen diferentes topologías de redes neuronales que determinan cómo están interconectadas las neuronas en las capas ocultas. Según el tipo de conexiones, las ANN se pueden clasificar en:

- **Redes de propagación hacia delante,** tienen como objetivo aproximar una función determinada. La característica principal es que las conexiones entre neuronas son siempre hacia adelante, es decir, en un solo sentido desde la capa de entrada hacia la capa de salida.
- **Redes recurrentes,** permite conexiones hacia atrás, es decir, las conexiones pueden realimentarse.

El primer modelo matemático de una red neuronal artificial fue descrito por Walter Pitts, junto con Warren McCulloch, en un artículo titulado 'Cálculo lógico de ideas inherentes en la actividad nerviosa' [106] en 1943. En él explican la arquitectura de una neurona sencilla y es todavía el estándar de referencia en el campo de las redes neuronales.

La neurona de McCulloch-Pitts (Figura 4.7) es la unidad esencial con la que se construye una ANN. El modelo se calcula realizando una suma ponderada de las entradas, seguida de la aplicación de una función no lineal llamada, función de activación *f*. Esta función se elige de acuerdo a la tarea realizada por la neurona. Entre las más comunes dentro del campo de las ANN destacan la función identidad, escalón, gaussiana y sinusoidal. El primer modelo que propusieron McCulloch-Pitts fue utilizando la función escalón. Con ello, conseguían una salida binaria, 0 ó 1.

La salida de una neurona está definida como,

$$y = f(\sum_{i=1}^{N} W_i x_i - \theta) \tag{4.1}$$

donde W_i representa la intensidad de interacción entre la neurona y las entradas, N es el número de características (atributos) que tienen las muestras de entrada y f es la función de

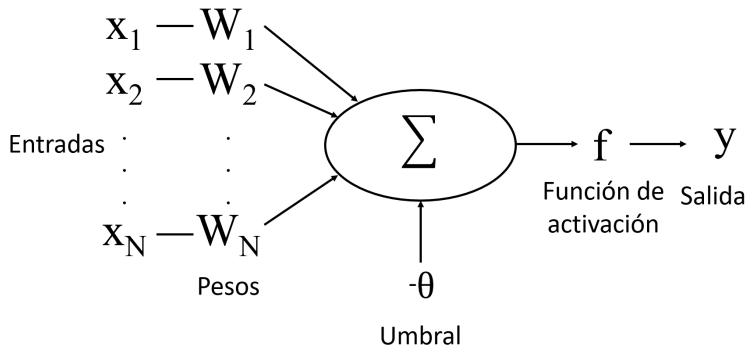


Figura 4.7: Modelo computacional de una neurona.

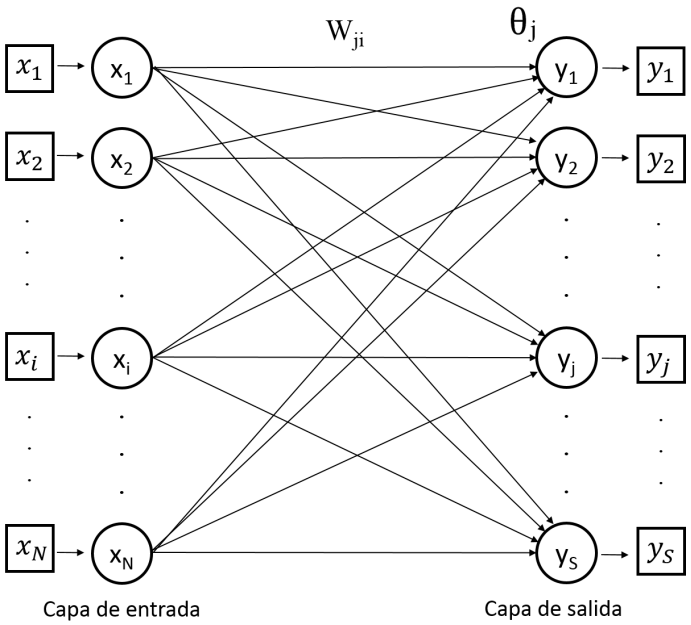


Figura 4.8: Arquitectura de un Perceptrón simple.

activación. El parámetro W_i se conoce como peso y se va modificando en el llamado proceso de aprendizaje. θ es el valor umbral o sesgo que, en este caso, se puede considerar como un umbral de actuación de la neurona.

Inspirado en el modelo de neurona anterior, Rosenblatt creó el Perceptrón simple (Figura 4.8) en 1962 [107]. Este modelo unidireccional está compuesto por dos capas de neuronas, una de entrada y otra de salida, y se puede utilizar como clasificador, ya que el algoritmo de entrenamiento puede determinar automáticamente los pesos que clasifican un conjunto de patrones a partir de un conjunto de ejemplos etiquetados.

Si la red neuronal está constituida por un número *j* de neuronas de salida, la salida neuronal de la *j*-ésima neurona y_j se define como,

$$y = f(\sum_{i=1}^{N} W_{ji}x_i - \theta_j) \tag{4.2}$$

El problema que tenía este modelo era que solo podía diferenciar dos regiones separadas

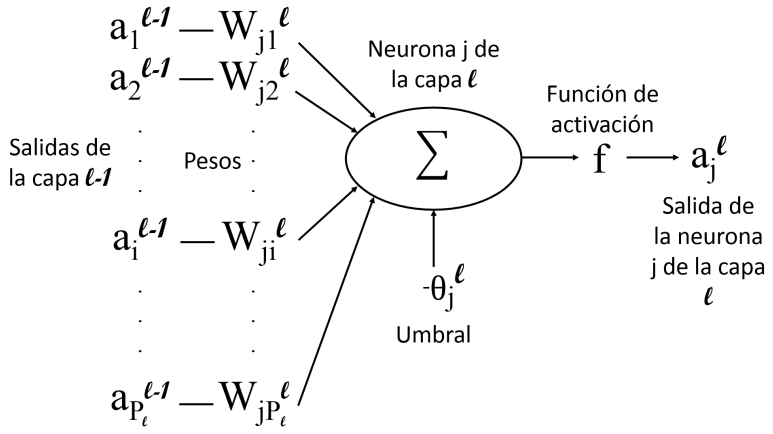


Figura 4.9: Modelo computacional de una neurona j de la capa l.

por un hiperplano (una recta en el caso de dos neuronas de entrada). Para solventar estas limitaciones se propuso el Perceptrón multicapa, que añadía capas ocultas.

Sea un Perceptrón multicapa con *L* capas ($L-2$ capas ocultas) y P_l neuronas en la capa *l*, para $l = 1, 2, ..., L$. Sea ${W_{ji}}^l$ la matriz de pesos que representa el peso de las conexiones de las neuronas de la capa $l-1$ con la neurona j de la capa l para $l = 2, ..., L$. Se denota a_j^l a la activación (salida) de la neurona *j* de la capa *l*. Estas activaciones se calculan del siguiente modo:

Activación de las neuronas de la capa de entrada

$$a_j^1 = x_i \tag{4.3}$$

siendo x_i un atributo $i = 1, 2, ..., N$.

Activación de las neuronas de la capa oculta

$$a_j^l = f(\sum_{i=1}^{P_l-1} W_{ji}^l a_i^{(l-1)} - \theta_j^l) \tag{4.4}$$

donde a_i^{l-1} son las activaciones de las neuronas de la capa $l-1$.

Activación de las neuronas de la capa de salida

$$a_j^L = y_j = f(\sum_{i=1}^{P_l-1} W_{ji}^L a_i^{(L-1)} - \theta_j^L) \tag{4.5}$$

siendo y_j la salida de una neurona $j = 1, 2, ..., S$.

Las funciones de activación *f*, para el Perceptrón multicapa, más utilizadas son la función sigmoidal y la función tangente hiperbólica.

$$f_{sigm}(x) = \frac{1}{1+e^{-x}} \tag{4.6}$$

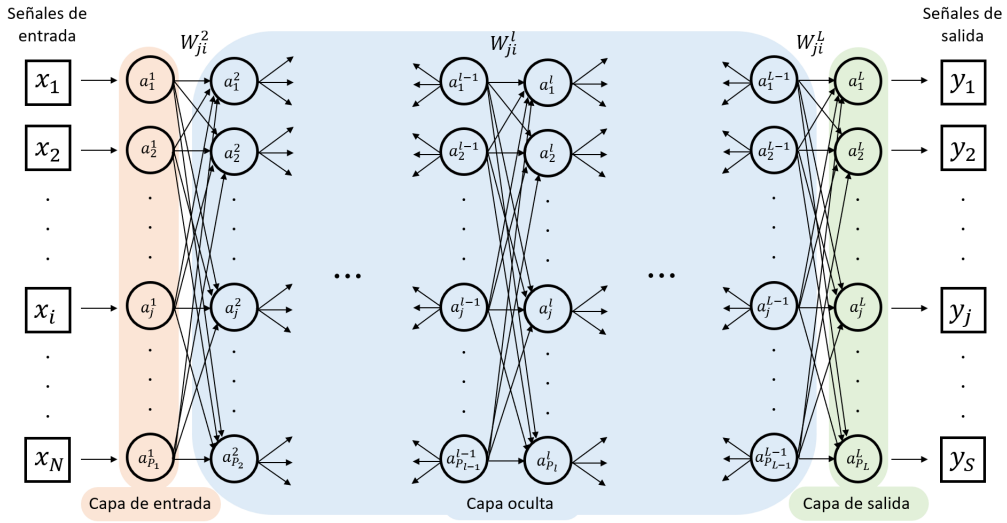


Figura 4.10: Modelo computacional de una neurona j de la capa l.

$$f_{thip}(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{4.7}$$

Ambas funciones son crecientes con dos niveles de saturación y se relacionan mediante la expresión:

$$f_{thip}(x) = 2f_{sigm}(x) - 1 \tag{4.8}$$

Como se ve en la Figura 4.10, la arquitectura general de una ANN puede estar compuesta por más de una capa oculta. Hay que destacar que en el diagrama no se ha considerado las conexiones hacia atrás. Para el caso del algoritmo de detención de nubes con ANN, se ha optado por una arquitectura con dos capas ocultas compuesta por 100 neuronas utilizando la función de activación tangente hiperbólica. Se han entrenado tomando 24 millones de pixeles de las imágenes de entrenamiento de forma equilibrada (12 millones de píxeles de resto y otros 12 millones de píxeles de nubes). De esta manera, la red tiene cuatro neuronas de entrada correspondientes a las 4 bandas VNIR, 100 neuronas en la capa oculta y dos neuronas en la capa de salida para predecir si el píxel introducido es de nubes o no (4-100-100-2). En total, la red está compuesta por 10.800 parámetros.

En la Tabla 4.4 se observan las métricas del modelo por cada imagen de test. Comparando los resultados con el árbol de decisiones de solo bandas VNIR, puede verse que son peores, sobre todo en las métricas que penalizan en mayor mediada los falsos positivos (F1-score, IoU, MIoU y kappa). Con el objetivo de aumentar la precisión y reducir los falsos positivos, se ha impuesto un umbral en la salida de 0,9, es decir, exigiendo que los positivos (nubes) tengan una probabilidad superior al 90 %; si no, se clasifican como resto. En la Tabla 4.5 puede verse que los resultados mejoran si se comparan con la red sin fijar un umbral y son muy parecidos a los obtenidos con el árbol de decisión. A pesar de haber utilizado un algoritmo mucho más avanzado, se han obtenido prácticamente los mismos resultados al utilizar solo las bandas VNIR.

Para poder extraer alguna característica o relación entre los espectros, se han añadido tres variables más a la entrada. Estas tres variables son las diferencias entre los valores consecutivos de las cuatro reflectancias, similar a una derivada. De esta manera, la red tiene 7 neuronas de entrada, estando ahora compuesta por 11.100 parámetros (7-100-100-2).

Tabla 4.4: Resultados del ANN.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	97,21	20,86	60,33	77,51	0,3288	0,1967	0,5084	0,3190
2	86,83	14,06	56,91	91,32	0,2437	0,1388	0,5021	0,2120
3	97,07	26,53	63,13	77,97	0,3959	0,2468	0,6086	0,3850
4	89,73	85,62	89,58	92,45	0,8891	0,8003	0,8129	0,7940
5	95,83	53,39	76,18	79,37	0,6384	0,4688	0,7128	0,6170
6	87,07	20,83	59,87	77,32	0,3282	0,1963	0,5314	0,2820
7	82,90	24,12	61,34	81,56	0,3723	0,2288	0,5243	0,3060
8	81,17	46,10	71,05	83,01	0,5928	0,4212	0,6015	0,4830
9	40,95	28,25	51,29	94,72	0,5486	0,2820	0,2137	0,2000
10	47,94	-	-	-	-	-	0,2397	-
11	67,01	31,58	61,17	69,59	0,4345	0,2708	0,4499	0,2450
12	40,00	-	-	-	-	-	0,2000	-
media	76,14	35,14	65,08	82,48	0,4772	0,3250	0,4921	0,3843

Tabla 4.5: Resultados del ANN fijando un umbral de 0,9.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	99,29	64,39	81,94	42,02	0,5086	0,3410	0,6669	0,5050
2	98,65	74,44	86,79	63,69	0,6865	0,5226	0,7544	0,6800
3	99,20	72,80	86,13	56,43	0,6358	0,4660	0,7290	0,6320
4	87,02	97,58	89,68	72,64	0,8328	0,7135	0,7608	0,7300
5	97,61	91,51	94,64	53,33	0,6739	0,5082	0,7418	0,6620
6	94,56	38,02	67,98	52,37	0,4405	0,2825	0,6135	0,4130
7	92,43	42,27	69,75	59,49	0,4942	0,3282	0,6248	0,4550
8	88,93	74,71	82,70	49,81	0,5977	0,4262	0,6528	0,5360
9	62,60	34,95	51,37	73,94	0,6674	0,3452	0,3277	0,3300
10	73,31	-	-	-	-	-	0,3666	-
11	78,66	40,60	63,44	37,17	0,3881	0,2408	0,5059	0,2590
12	15,33	-	-	-	-	-	0,7665	-
media	82,30	63,13	77,44	56,09	0,5925	0,4174	0,6259	0,5202

Tabla 4.6: Resultados del ANN añadiendo las tres variables.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	97,24	21,11	60,45	78,13	0,3324	0,1993	0,5857	0,3230
2	84,39	12,23	56,01	92,59	0,2161	0,1211	0,4808	0,1820
3	97,00	26,11	62,92	78,42	0,3918	0,2436	0,6067	0,3800
4	88,70	83,87	88,61	92,39	0,8792	0,7845	0,7963	0,7740
5	94,27	43,64	71,33	80,64	0,5664	0,3951	0,6678	0,5390
6	86,79	20,56	59,75	77,94	0,3254	0,1943	0,5289	0,2790
7	81,58	22,92	60,78	83,02	0,3593	0,2190	0,5124	0,2900
8	81,93	47,34	71,80	84,02	0,6056	0,4343	0,6123	0,5000
9	40,27	28,02	51,29	95,04	0,5443	0,2798	0,2101	0,2000
10	46,57	-	-	-	-	-	0,2329	-
11	66,57	31,54	61,34	71,42	0,4376	0,2801	0,4479	0,2480
12	75,20	-	-	-	-	-	0,0376	-
media	78,38	33,73	64,43	83,36	0,4658	0,3151	0,4766	0,3715

Tabla 4.7: Resultados del ANN añadiendo las tres variables y fijando un umbral de 0,9.

img	**OA (%)**	**OP (%)**	**AP (%)**	**Recall (%)**	**F1-score**	**IoU**	**MIoU**	**kappa**
1	99,30	65,71	82,61	43,72	0,5251	0,3560	0,6745	0,5220
2	98,55	70,50	84,84	64,94	0,6761	0,5107	0,7480	0,6690
3	99,22	73,20	86,34	57,46	0,6439	0,4748	0,7334	0,6400
4	86,87	96,59	89,26	73,09	0,8321	0,7125	0,7590	0,7280
5	97,45	88,41	93,05	51,74	0,6527	0,4845	0,7292	0,6400
6	94,59	38,25	68,10	52,82	0,4437	0,2851	0,6149	0,4160
7	93,47	48,02	72,70	61,08	0,5377	0,3677	0,6499	0,5030
8	89,58	77,36	84,24	52,13	0,6229	0,4523	0,6691	0,5650
9	62,01	34,00	51,30	72,98	0,6498	0,3358	0,3243	0,3100
10	70,96	-	-	-	-	-	0,3548	-
11	78,38	40,11	63,24	38,07	0,3906	0,2427	0,5052	0,2590
12	16,35	-	-	-	-	-	0,8174	-
media	82,23	63,22	77,57	56,80	0,5974	0,4222	0,6316	0,5252

Como puede verse en las Tablas 4.6 y 4.7, los resultados prácticamente no mejoran a los que utilizan directamente las bandas VNIR, sin añadir variables extra, demostrando que la relación entre las firmas espectrales no tiene prácticamente impacto en los resultados. Por lo tanto, de este análisis se puede extraer que utilizando algoritmos que solo hacen uso de las características espectrales, los resultados están limitados a unos porcentajes de precisión bajos y a una cantidad elevada de falsos positivos, sobre todo en terrenos como nieve/hielo debido a su similitud espectral con las nubes.

Capítulo 5

Algoritmos basados en el aprendizaje profundo

En este capítulo se van a abordar los algoritmos más sofisticados de la literatura científica en lo que respecta a la segmentación de nubes en imagen satelital. En primer lugar, se evaluaran aquellos algoritmos que reporten un mayor rendimiento desde el punto de vista de la precisión y la robustez sin considerar la complejidad computacional de los mismos. Posteriormente, se buscarán versiones reducidas de estos algoritmos que permitan el procesamiento en tiempo real en un sistema embebido sin perder los requisitos mínimos de precisión que se exigen. Dichos requisitos son que el algoritmo sea capaz de evaluar el nivel de nubosidad de la imagen en cualquier terreno y condición atmosférica, minimizando el número de falsos positivos.

Los algoritmos más sofisticados están basados en modificaciones del modelo U-Net. Es por ello, que en primer lugar se estudia esta arquitectura y se evalúan sus ventajas y desventajas con respecto al resto de modelos. Una vez estudiadas, se aplican diferentes modificaciones tanto para aumentar la precisión y robustez del algoritmo, como para simplificarlo. Finalmente, se realiza una optimización de los hiperparámetros del modelo propuesto con el objetivo de encontrar un compromiso adecuado entre la precisión de la segmentación y la complejidad computacional de la ejecución del modelo en la fase de inferencia.

5.1. U-Net

La U-Net es una arquitectura basada en redes convolucionales para la segmentación precisa de imágenes (Figura 5.1). Originalmente fue propuesta para la segmentación de imágenes biomédicas aunque su uso se a extendido a multitud de áreas [70]. Gracias a la ausencia de capas densas o capas totalmente conectadas, el modelo es más sencillo de entrenar y utiliza menos recursos computacionales que, por ejemplo, la redes neuronales artificiales (ANN). Consta de tres partes fundamentales: codificador (encoder), decodificador (decoder) y conexiones omitidas (skip connections).

La primera etapa se encarga de extraer las características espaciales de la escena, reduciendo las dimensiones espaciales y aumentando el número de canales. Por su parte, el decodificador toma estas características y las reproyecta para crear la máscara de salida. Las conexiones residuales enlazan los puntos intermedios del codificador y el decodificador para permitir la fusión entre las características de bajo y alto nivel.

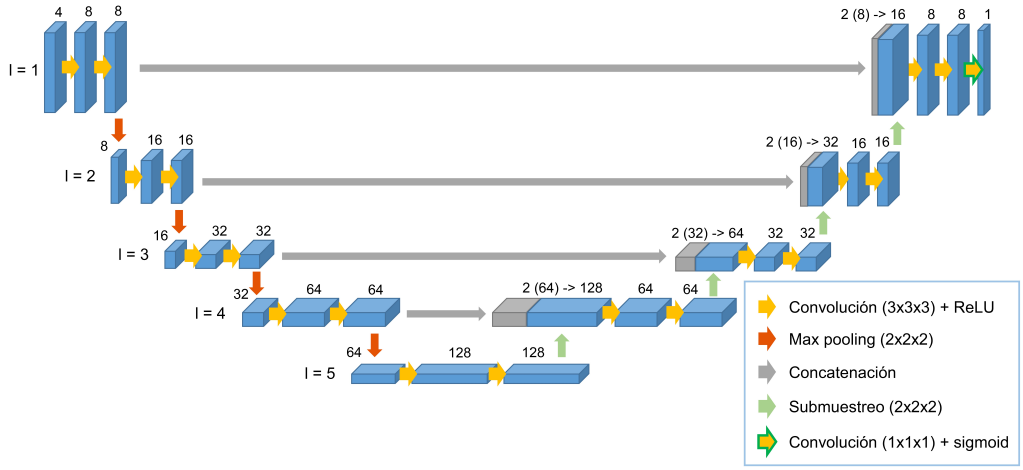


Figura 5.1: Arquitectura de la U-Net.

5.1.1. Arquitectura de la U-Net

La primera etapa está compuesta por la combinación de manera sucesiva de dos operaciones de convolución seguidas de una operación de *max-pooling*. La función de activación de ambas convoluciones es de tipo ReLU. Cada vez que se repite esta sucesión, se duplica el número de filtros convolucionales. Las capas convolucionales del modelo están diseñadas para extraer características invariables por traslación de toda la imagen. Cada filtro de una capa se le aplica la convolución con la entrada, produciendo un mapa de respuesta de los lugares de la imagen en los que el patrón del filtro es similar. La ventaja de las capas convolucionales es que comparten los pesos, ya que al utilizar los mismos filtros en toda la imagen, se reduce el número de pesos y se refuerza la invariabilidad de las características aprendidas. Además, el uso de capas exclusivamente convolucionales es lo que permite crear modelos que funcionan con tamaños de imagen arbitrarios, ya que ninguna capa requiere un tamaño de entrada específico, a diferencia de las capas totalmente conectadas que suelen verse en las redes de clasificación. La operación de *max-pooling* reduce los píxeles de entrada a lo largo de sus dimensiones espaciales (altura y anchura) tomando el valor máximo sobre una ventana (filtro) de entrada por cada canal. En la U-Net, el tamaño de la ventana es de dimensión 2x2 con un desplazamiento de ventana (stride) de 2, de modo que el área de la imagen se divide por 4. Al final de la etapa de codificación se añaden una par de capas de *dropout* para evitar el sobreentrenamiento del modelo.

El modelo de U-Net utiliza filtros convolucionales 3x3, ya que es el tamaño que se estableció después de un proceso de evolución en los primeros años de desarrollo de los modelos convolucionales. En 2012, AlexNet tenía una primera convolución de tamaño 11x11 [61]. En 2013, ZFNet sustituyó esta capa convolucional por una de 7x7 [108]. En 2014, el núcleo de convolución más grande de GoogleNet era de 5x5 aunque mantuvo una primera capa convolucional de 7x7 [109]. Ese mismo año, apareció el modelo VGG que solo utilizaba núcleos de convolución de 3x3 [110]. En las versiones posteriores, la capa convolucional 5x5 de la primera versión de GoogleNet fue sustituida por 2 capas convolucionales 3x3 apiladas, copiando al modelo VGG16. Este proceso de reducción principalmente se debió a la necesidad de reducir el número de parámetros del modelo, ya que crece cuadráticamente con el tamaño del filtro. Esto hace que en la mayoría de casos no sea conveniente utilizar filtros de convolución grandes. Por ello, la opción más común es utilizar filtros de convolución de 3x3 o de 5x5, aunque en la primera capa convolucional se pueden aplicar tamaños más grandes puesto que solo tiene los canales de entrada de la imagen. El motivo por el cual se apilan dos

convoluciones seguidas de 3x3 es porque se comporta como si una ventana convolucional 5x5 estuviera escaneando la entrada, pero con un número de parámetros inferior respecto a utilizar una sola capa convolucional de 5x5 en una relación de 7/5. Además de reducir el número de parámetros, implementar dos convoluciones sucesivas mejora los resultados y reduce el sobreajuste. En todo momento se utilizan filtros de tamaño impar porque dividen simétricamente los píxeles de la capa anterior alrededor del píxel de salida, evitando distorsiones a través de las capas.

Por otro lado, la etapa de decodificación comienza por una operación opuesta al *max-pooling* llamada *upsampling* que consiste en un sobremuestreo del mapa de características. Dicha capa funciona repitiendo las filas y columnas de la entrada con una interpolación, típicamente del vecino más cercano o de interpolación bilineal, pero no realiza ningún aprendizaje. Por ello, es necesario introducir una capa de convolución para que aprenda a interpretar la entrada duplicada y, mediante el entrenamiento, sea capaz de traducirla en detalles significativos. Para evitar o reducir el efecto de tablero causado por la superposición desigual en algunas partes de la imagen, se utiliza un tamaño de filtro divisible por el *stride*. En el caso de la U-Net original se utiliza un tamaño de filtro de 2x2 que es divisible por un *stride* de 2.

Otra opción para realizar el sobremuestreo sin utilizar el *upsampling*, es la convolución transpuesta. Esta técnica consiste en realizar un muestreo ascendente de una imagen con parámetros aprendibles. El proceso que sigue es el inverso de una convolución normal, es decir, la entrada es una imagen de baja resolución y la salida es una imagen de alta resolución. Esto se consigue debido a que una convolución normal puede expresarse como una multiplicación matricial de la imagen de entrada y del filtro para producir la imagen de salida. Con sólo tomar la transposición de la matriz del filtro, se puede invertir el proceso de convolución, de ahí el nombre de convolución transpuesta. Este método es el más común pero suele producir más artefactos en la imagen en comparación a utilizar una operación de *upsampling* seguida de una convolución normal.

Después de aplicar el sobremuestreo, el resultado se concatena con la salida que se encuentra en el mismo nivel de profundidad de la etapa de codificación, en concreto, se concatena con la salida de la doble convolución, antes de aplicar el *max-pooling*. Esta conexión residual permite la recuperación de detalles aprendidos en la parte de la codificación para reconstruir la imagen en la parte del decodificador. De manera consecutiva se vuelve a aplicar dos convoluciones 3x3 seguidas manteniendo la simetría con la etapa de codificación. Finalmente, en la última capa se emplea una convolución de 1x1 para asignar el vector de características al número de clases.

5.1.2. Función de pérdidas

La función de pérdidas, o también llamada función de coste, permite comparar lo que ha predicho el modelo con el esperado. Para ello, se impone un coste a cada predicción dependiendo de lo que se ha desviado respecto al real. Es por ello que en el entrenamiento se busca minimizar la función de perdidas, ya que cuanto menor sea la pérdida, más se parecerá el conjunto predicho al real. Una de las estrategias que se utiliza en el entrenamiento es calcular la función de pérdidas del conjunto de validación para escoger la relación de pesos que minimiza dicha función. Con este fin, en cada época se calcula la función de pérdidas del conjunto de validación y se compara con el de la época anterior. Si el nuevo valor calculado es menor, significará que el modelo ha aprendido mejor y la relación de pesos calculada se utilizará para la siguiente época. En caso contrarío, se descarta lo calculado en esta época y se utiliza el de una época anterior.

Hay una gran variedad de funciones de pérdidas que se utilizan dependiendo del tipo de entrenamiento (clasificación o regresión) y de las características de lo que se desea entrenar. Para este proyecto la función de coste debe ser binaria debido a que el objetivo del entrenamiento es separar las nubes del resto de píxeles, es decir, una clasificación entre dos clases. La entropía cruzada binaría es la función de pérdidas más utilizada en la actualidad y se define de la siguiente forma,

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{5.1}$$

siendo *y* la etiqueta (*1* para nubes y *0* para el resto) y *p(y)* la probabilidad predicha de que el píxel sea de nube para los *N* píxeles. De la fórmula se puede extraer que para cada píxel de nubes, se agrega *log(p(y))* a la pérdida, es decir, la probabilidad logarítmica de que el píxel sea de nubes. En cambio, se agrega *log(1-p(y))*, es decir, la probabilidad logarítmica de que no sea de nubes. De esta manera, a medida que la probabilidad predicha de la clase de nubes se acerque a cero, la pérdida aumenta exponencialmente, dando mayor coste cuanto más lejos esté de predecir el píxel correctamente.

5.1.3. Tasa de aprendizaje

La tasa de aprendizaje, *learning rate*, es un hiperparámetro configurable que se utiliza en el entrenamiento de las redes neuronales cuyo propósito es determinar el tamaño del paso en cada iteración mientras se avanza hacia un mínimo de la función de pérdidas. La tasa de aprendizaje controla la rapidez con la que el modelo se adapta al problema. Las tasas de aprendizaje más pequeñas requieren más épocas de entrenamiento dado que los cambios en los pesos son menores en cada actualización, mientras que las tasas de aprendizaje más grandes dan lugar a cambios rápidos y requieren menos épocas de entrenamiento.

Una tasa de aprendizaje demasiado grande puede hacer que el modelo converja demasiado rápido a una solución que no sea óptima, mientras que una tasa de aprendizaje demasiado pequeña puede hacer que el proceso se atasque. El reto de entrenar redes neuronales de aprendizaje profundo implica seleccionar cuidadosamente la tasa de aprendizaje. Los optimizadores más avanzados actualizan la tasa de aprendizaje durante el entrenamiento, mejorando el rendimiento del modelo.

5.1.4. Optimizador

En los modelos de aprendizaje profundo, es necesario modificar los pesos cada época y minimizar la función de pérdida. Un optimizador es una función o un algoritmo que modifica los atributos de la red neuronal, como los pesos y la tasa de aprendizaje. Así, ayuda a reducir la pérdida global y a mejorar la precisión. El problema de elegir los pesos adecuados para el modelo es una tarea de enormes proporciones, ya que un modelo de aprendizaje profundo suele estar formado por millones de parámetros. Esto plantea la necesidad de elegir un algoritmo de optimización adecuado para su aplicación. Se pueden utilizar diferentes optimizadores para realizar cambios en los pesos y la tasa de aprendizaje. Sin embargo, la elección del mejor optimizador depende de la aplicación.

El Descenso del Gradiente es considerado el optimizador más común dentro de la clase de optimizadores. Este algoritmo de optimización utiliza el cálculo para modificar los valores de forma consistente y alcanzar el mínimo local.

$$x_{n+1} = x_n - \alpha \nabla f(x_n) \tag{5.2}$$

La ecuación 5.2 muestra el cálculo del gradiente, siendo α el tamaño de paso que representa hasta donde se mueve el gradiente en cada iteración. El cálculo del descenso del gradiente comienza con algunos coeficientes, calcula su coste y busca un valor de coste menor que el actual. De esta manera, escoge el peso más bajo y actualiza el valor de los coeficientes. El descenso del gradiente funciona bien para la mayoría de aplicaciones pero tiene algunas desventajas. El principal problema es que es costoso calcular los gradientes si el tamaño de los datos es enorme. Además, el descenso de gradiente funciona bien para las funciones convexas, pero no sabe qué distancia recorrer a lo largo del gradiente para las funciones no convexas.

Para solventar estos inconvenientes, surgieron modificaciones del optimizador del descenso del gradiente. Una de las más destacables es el descenso del gradiente adaptativo (Adagrad). Este optimizador utiliza diferentes tasas de aprendizaje para cada iteración. El cambio en la tasa de aprendizaje depende de la diferencia en los parámetros durante el entrenamiento. Cuanto más cambien los parámetros, menor será el cambio de la tasa de aprendizaje. Esta modificación es muy beneficiosa porque los conjuntos de datos del mundo real contienen características tanto dispersas como densas. Por tanto, es inadecuado tener el mismo valor de tasa de aprendizaje para todas las características. El algoritmo Adagrad utiliza la siguiente fórmula para actualizar los pesos.

$$W_t = W_{t-1} - \eta'_t \frac{\partial L}{\partial w(t-1)} \tag{5.3}$$

$$\eta'_t = \frac{\eta}{\sqrt{\alpha_t + \epsilon}} \tag{5.4}$$

siendo $\alpha(t)$ las diferentes tasas de aprendizaje en cada iteración, *n* una constante y ϵ un pequeño positivo para evitar la división entre 0.

La ventaja de utilizar Adagrad es que elimina la necesidad de modificar manualmente la tasa de aprendizaje. Es más fiable que los algoritmos anteriores, y alcanza la convergencia a mayor velocidad. El gran inconveniente del optimizador AdaGrad es que disminuye la tasa de aprendizaje de forma agresiva y monótona. Puede haber un punto en el que la tasa de aprendizaje sea extremadamente pequeña. Esto se debe a que los gradientes al cuadrado en el denominador siguen acumulándose, y por lo tanto la parte del denominador sigue aumentando. Debido a las pequeñas tasas de aprendizaje, el modelo acaba siendo incapaz de adquirir más conocimientos y, por tanto, la precisión del modelo se ve comprometida.

AdaDelta puede considerarse una versión más robusta del optimizador AdaGrad. Se basa en el aprendizaje adaptativo y está diseñado para hacer frente a los importantes inconvenientes del optimizador AdaGrad y similares, como puede ser el optimizador RMS prop (optimizador de media cuadrática). El principal problema de estos optimizadores es que la tasa de aprendizaje inicial debe definirse manualmente. Además, otro problema de estos algoritmos es la tasa de aprendizaje decreciente, que se vuelve infinitesimalmente pequeña en algún momento. Debido a ello, pasadas un cierto número de iteraciones, el modelo ya no puede aprender nuevos conocimientos. El proceso se repite hasta alcanzar el mínimo local, es decir, hasta alcanzar un punto más allá del cual no se puede avanzar.

Para hacer frente a estos problemas, en lugar de acumular todos los gradientes pasados al cuadrado, AdaDelta restringe la ventana de gradientes pasados acumulados a un tamaño

fijo. Para no almacenar ineficientemente los gradientes cuadrados, se define recursivamente la media decreciente de todos los gradientes $E\left[g^2\right]_t$, haciendo que la media actual solo dependa de la media anterior y del gradiente actual:

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma)\, g_t^2 \tag{5.5}$$

siendo normalmente γ entorno a 0,9. Reescribiendo las actualizaciones del SGD (Descenso del Gradiente Estocástico) en términos del vector de actualización de los parámetros:

$$\Delta\theta_t = -\eta \cdot g_{t,i} \tag{5.6}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{5.7}$$

AdaDelta se define de la siguiente forma:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t \tag{5.8}$$

donde g_t es el gradiente reescalado.

Para todos los entrenamientos, se ha utilizado el optimizador AdaDelta debido a las ventajas que ofrece respecto a los demás optimizadores. Además, es uno de los optimizadores más utilizados para la detección de nubes [101].

5.1.5. Función de activación de la última capa

La función de activación de la última capa tiene como objetivo transformar la salida del modelo en probabilidades de pertenencia a una de las clases de salida. Para la clasificación binaria, una de las más utilizadas es la función de activación sigmoidea. Independientemente de la entrada, la función siempre produce un valor entre 0 y 1. La activación sigmoidea es una función de activación ideal para un problema de clasificación binaria en el que la salida se interpreta como una distribución de probabilidad binomial. La función de activación sigmoidea también puede utilizarse como función de activación para problemas de clasificación multiclase en los que las clases no son mutuamente excluyentes. A menudo se denomina clasificación multietiqueta en lugar de clasificación multiclase. La función de activación sigmoidea no es apropiada para los problemas de clasificación multiclase con clases mutuamente excluyentes en los que se requiere una distribución de probabilidad multinomial. En su lugar, se requiere una activación alternativa denominada función softmax.

Para obtener la máscara binaria, se debe establecer un umbral para el cual se considere que el pixel es de nubes o no, siendo 0.5 el valor por defecto. Dicho umbral tiene una importancia crucial en términos de equilibrio entre la comisión y la omisión y los errores. Dado que uno de los puntos más importes de la clasificación de nubes es que intente confundir lo menos posible nubes cuando no las hay, se deberá aumentar el valor del umbral manteniendo el equilibrio de rendimiento para todo tipo de situaciones.

5.1.6. Equilibrio entre clases del conjunto de entrenamiento

En la mayoría de la imágenes de entrenamiento, hay una presencia minoritaria de nubes respecto a los demás terrenos, es decir, hay menos píxeles de la clase nubes en comparación al resto de píxeles. A este fenómeno se le denomina el desequilibrio entre clases y es un tema ampliamente estudiado, ya que es un punto crítico a la hora de abordar problemas de clasificación. Por lo general, suele afectar a los algoritmos de clasificación en su proceso de generalización de la información, perjudicando a la clase con menor presencia de píxeles, en este caso, las nubes. Por ejemplo, si la red debe clasificar 10 píxeles de nubes y 90 píxeles de resto, lo más probable es que la red se limite a decir que solo hay píxeles de resto puesto que así se asegura porcentajes de acierto de más del 90 %. Es decir, midiendo la efectividad del modelo por la cantidad de aciertos que tuvo, se puede llegar a tener una falsa sensación de que el modelo funciona bien. Existen diversas estrategias para reducir el efecto del desequilibrio entre clases. A continuación, pueden verse algunas de las estrategias más comunes:

- **Modificación de la base de datos:** Se podrían eliminar imágenes que contienen mayoritariamente píxeles de la clase mayoritaria, en este caso nubes, con el fin de reducir y equilibrar el conjunto de entrenamiento. Hacerlo de esta forma implica un cierto riesgo en el sentido de poder eliminar imágenes importantes que aportan información, empeorando el resultado del modelo. Para evitar lo máximo posible que esto suceda, se deben seleccionar siguiendo algún criterio como, por ejemplo, eliminar aquellas imágenes que se asemejen a otras que ya existen y sean mayoritariamente de nubes. También se pueden duplicar aquellas imágenes que mayoritariamente son de nubes, equilibrando el conjunto de datos, pero no es muy recomendable, ya que se pude llegar a sobreajustar el modelo.
- **Creación de muestras artificiales:** Se podrían crear muestras sintéticas que sean muy parecidas a las existentes en la base se datos, sin que sean idénticas. Para ello, se podrían hacer uso de diversos algoritmos que intentan seguir la tendencia de la clase minoritaria. Para el caso de la detección de nubes, se podrían añadir nubes artificialmente en las imágenes que no hay o hay muy poca presencia de nubes. El problema de crear muestras artificiales es que se puede alterar la distribución real de las nubes, llegando a confundir el modelo en su clasificación. En [111], aumentan el número de imágenes generando nubes artificiales. Esta estrategia también se podría utilizar para evitar el etiquetado de nubes, muy útil cuando de dispongan de las imágenes reales producidas por la iSIM y se desee reentrenar el modelo con ellas.
- **Ajustando la métrica de pérdidas del modelo:** Consiste en ajustar la métrica de pérdidas para que penalice en mayor media la clase mayoritaria, en este caso, todo lo que no sea nube denominado resto

Para intentar balancear este desequilibrio, se ha optado por la primera opción y se han seleccionado manualmente las imágenes siguiendo el criterio de semejanza de imágenes. Las imágenes de entrenamiento seleccionadas han sido la 1, 2, 3, 6, 9, 10, 12, 18, 20 y 21, siendo 10 en total. De esta manera, se consigue de media que en cada batch haya un 40 % de presencia de nubes, siendo suficiente para considerar que están en relativo equilibrio.

5.1.7. Características de entrada

La normalización es una técnica que se aplica a menudo como parte del preacondicionamiento de los datos de entrada en el aprendizaje automático. El objetivo de la normalización es cambiar los valores del conjunto de datos de entrada a una escala común, sin distorsionar las diferencias en los rangos de valores. Para el aprendizaje automático, todos los conjuntos de datos no requieren normalización, sólo es necesaria cuando las características tienen rangos diferentes. En el caso de la detección de nubes, sí que es muy recomendable la normalización para evitar que las características de los vectores de entrenamiento tengan rangos muy distintos, puesto que la reflectividad espectral de cada banda puede oscilar entre 0 y más de 11.000.

Por otra parte, para establecer el tamaño de las imágenes de entrada se ha tenido en cuenta el tamaño de imagen y la resolución que ofrecen las cámaras de Satlantis. El tamaño de cada imagen fijado por Satlantis es de 56 x 244 píxeles. Para que el modelo tenga como entrada el mismo ancho y largo, se divide entre 4 produciendo mini-parches de 56 x 56. Para el entrenamiento, dicho tamaño se debe ajustar, ya que la resolución de las imágenes de Satlantis es aproximadamente seis veces mayor que las que se utilizan para el entrenamiento. De este modo, el modelo debe ser entrenado con $(56 \cdot 6)$ x $(56 \cdot 6)$, es decir, con tamaño de imagen de 336 x 336 píxeles. A pesar de ello, posteriormente en el análisis de hiperparámetros se probará el modelo propuesto con un tamaño menor para comprobar su eficacia, dado que un tamaño menor de imagen implica un modelo con menor número de parámetros.

Para aumentar la eficiencia del entrenamiento, se han divido las imágenes de Sentinel-2 al tamaño de imagen propuesto de 336 x 336 píxeles. De esta manera, se evita que el algoritmo tenga que cargar la imagen completa de 10980 x 10980 por cada paso del entrenamiento, debido al muy alto coste computacional. La división en imágenes pequeñas se ha realizado varias veces, ya que al tener en cuenta la parte espacial, depende de dónde se corte la imagen, la red puede interpretar distintas situaciones. Además, haciéndolo de esta manera, se aumenta el número de imágenes que se disponen en el entrenamiento. Las divisiones realizadas han sido las siguientes, produciendo un total de 40.960 imágenes de 336 x 336 píxeles de entrenamiento y 20.480 de validación:

- División empezando en (0, 0).
- División empezando en (168, 0).
- División empezando en (0, 168).
- División empezando en (168, 168).

5.1.8. Resultados del modelo U-Net básico con imágenes multiespectrales para la detección de nubes

Una vez estudiada la arquitectura de una U-Net clásica, se procede a su implementación para la detección de nubes. El objetivo principal de implementar este modelo es conocer cual es el rendimiento que ofrece la U-Net clásica sin ninguna modificación y a partir de ahí, ir modificando y simplificando el modelo hasta obtener uno que tenga un equilibrio óptimo entre rendimiento y complejidad computacional. Se utiliza como base la U-Net ya que en el estado del arte se ha visto que la mayoría de modelos para la detección de nubes en imágenes multiespectrales están basado en ello [98, 99, 101, 112].

La arquitectura de la U-Net (Figura 5.1) consta de 4 capas de codificación y de otras 4 de decodificación. Cada capa de codifición a su vez está compuesta por 2 capas de convolución seguidas de una de *max-pooling*. La capa de activación de ambas convoluciones es de tipo ReLU con la operación de relleno (padding) activada, haciendo que tanto la entrada como la salida de la capa tengan el mismo tamaño. Normalmente, para conseguirlo se añaden filas y columnas de ceros en la entrada. El número de filtros de convolución va aumentando en un factor de 2 por cada capa de codificación, empezando por filtros de 8. El tamaño de los filtros de convolución es de 3 x 3 y su desplazamiento (stride) es de un píxel, es decir, el filtro se desplaza un píxel de manera sucesiva a lo largo y ancho de la imagen. La capa de *max-pooling* tiene un tamaño de reducción (pool-size) de 2 x 2 que se va aplicando a toda la imagen tomando el valor máximo por cada ventana de entrada, haciendo que se reduzca las dimensiones espaciales de la imagen (altura y anchura) en un factor de 2. Antes de aplicar las capas de decodificación, se realizan dos operaciones seguidas de convolución sin que se realice la operación de *max-pooling*.

Cada capa de decodificación está compuesta por 5 capas, comenzando con una capa de muestreo ascendente (upsampling) seguida de una convolución con tamaño de filtro de 5 x 5. La capa de muestreo aumenta en 2 el tamaño de la imagen y la capa de convolución es la que se encarga de aprender el reescalado. La capa de convolución para el sobremuestreo conviene que esté compuesta por solo de una capa para evitar artefactos en la reconstrucción de la imagen. Es por ello que el tamaño del filtro es de 5 x 5, en vez dos capas de convolución seguidas de 3 x 3. A continuación, se concatena la salida de la capa de convolución del reescalado con la capa de salida de la codificación y se aplican 2 convoluciones seguidas 3 x 3. En la última capa de decodificación, se añade una capa de convolución cuyo número de filtros debe corresponderse al número de clases, en este caso, 1. La capa de activación es una sigmoidea para darle significado de probabilidad a los datos de salida de la última capa del modelo.

El entrenamiento se realiza con una GPU Nvidia GeForce RTX 3090 de 24 GB de memoria dedicada (VRAM). El dato más importante a tener en cuenta es la memoria dedicada disponible, ya que influye en el tamaño máximo por lote (batch size) que se puede introducir por cada etapa (step). Cuanto mayor sea el tamaño del lote, mayor número de muestras se podrán entrenar a la vez, reduciendo el tiempo de entrenamiento. Además, es importante que por cada lote que se introduzca, estén presentes todo tipo de terrenos para no deshacer lo entrenado en etapas anteriores. Como esta selección se hace de manera aleatoria, cuanto mayor sea el tamaño del lote, mayor probabilidad de tener todo tipo de terrenos. En el caso de la U-Net, se ha podido entrenar con un tamaño por lote de 64. El número de etapas (steps) se debe escoger teniendo en cuenta el número de imágenes que se tienen para entrenar, en este caso, 150. El número de épocas marca cuántas veces se repite el entrenamiento con los nuevos pesos previamente calculados. Dicho valor puede escogerse de manera arbitraría, finalizando el entrenamiento cuando la función de pérdidas empieza a incrementarse durante algunas épocas seguidas, aunque hay que evitar entrenar en exceso el modelo para evitar el sobreajuste. Para evitar el sobreajuste, también se tiene la etapa de validación que permite escoger en cada época la relación de pesos que minimiza la función de pérdidas del conjunto de validación.

Los resultado experimentales pueden observarse en la Tabla 5.1 correspondiente a las métricas descritas en el capítulo 4. También se añade el número de parámetros del modelo y el tiempo de ejecución de la inferencia en CPU. El objetivo de calcular el tiempo de inferencia en CPU es para comparar el coste computacional de los diferentes modelos entre sí, aunque posteriormente los modelos elegidos deberán ser calculados en el propio MPSoC.

Los resultados muestran que el modelo tiene un bajo rendimiento, sobre todo en aque-

Tabla 5.1: Resultados del modelo U-Net 3x3.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	99,10	46,06	72,83	50,82	0,4832	0,3186	0,6548	0,4787
2	98,74	78,03	88,54	55,65	0,6496	0,4811	0,7342	0,6434
3	98,88	54,12	76,84	64,98	0,5905	0,4190	0,7038	0,5849
4	87,65	93,19	88,76	78,11	0,8499	0,7389	0,7746	0,7464
5	86,84	19,29	58,70	63,02	0,2954	0,1733	0,5190	0,2448
6	93,30	31,31	64,60	52,55	0,3924	0,2441	0,5878	0,3593
7	90,13	30,72	64,10	60,15	0,4067	0,2553	0,5765	0,3590
8	71,71	34,52	64,37	78,35	0,4793	0,3152	0,4949	0,3231
9	72,21	5,59	52,62	87,05	0,1051	0,0554	0,3864	0,0724
10	88,45	-	-	-	-	-	0,8845	-
11	77,05	39,22	64,54	56,70	0,4637	0,3018	0,5235	0,3238
12	29,29	-	-	-	-	-	0,2929	-
media	82,78	43,21	69,59	64,74	0,4716	0,3303	0,5944	0,4136

Tabla 5.2: Número de parámetros y tiempo de inferencia medio del modelo U-Net 3x3.

model param.	inf. CPU (s)
714.509	0,0714

llas imágenes de test que contienen nieve. En las Figuras 5.2 puede visualizarse los resultados de las imágenes de test, mostrando en naranja los píxeles de nube que no ha detectado (falsos negativos) y en rojo aquellos píxeles que ha detectado de nubes sin que lo sean en la realidad (falsos postivos). Con tamaños de filtro 3x3 puede verse cómo la nieve la está detectando como nubes. Además, se pude ver cómo aparecen patrones en rejilla en la imagen debido a la unión de los diferentes parches que se calculan. Como el tamaño de la imagen es mucho mayor que el tamaño fijado como entrada del modelo (en este caso, 336 píxeles), la imagen debe trocearse en pequeños parches y, posteriormente, agruparse formando la imagen completa. Al hacerlo de esta manera, dependiendo del modelo, puede aparecer este fenómeno en los bordes de la imagen debido a las convoluciones. La solución más común a este problema es coger más parches, produciendo un solapamiento entre los mismos. De esta manera, con los píxeles en común de los diferentes parches, se selecciona entre nube o no nube la clase más repetida. El problema de este método es que se requiere realizar más inferencias y, además, consume tiempo computacional al calcular la clase más frecuente de cada píxel.

En cambio, con filtros 9x9 se puede ver que desaparece este efecto no deseado al hacer las convoluciones de ventanas más grandes, mejorando mucho la precisión respecto al modelo con filtros 3x3. El problema de utilizar filtros tan grandes es que el número de parámetros del modelo aumenta considerablemente, siendo el tiempo de calcular la inferencia en CPU aproximadamente 11 veces superior al modelo U-Net con tamaño de filtros de convolución 3x3 (Tabla 5.4). Además, aunque el modelo con filtros 9x9 mejora en todos los aspectos en lo referente a las métricas respecto al 3x3 (Tabla 5.3), sigue siendo insuficiente para los requerimientos de este proyecto, ya que sigue confundiendo la nieve con las nubes. Por lo tanto, debe buscarse un modelo que sea capaz de solventar estas dificultades.

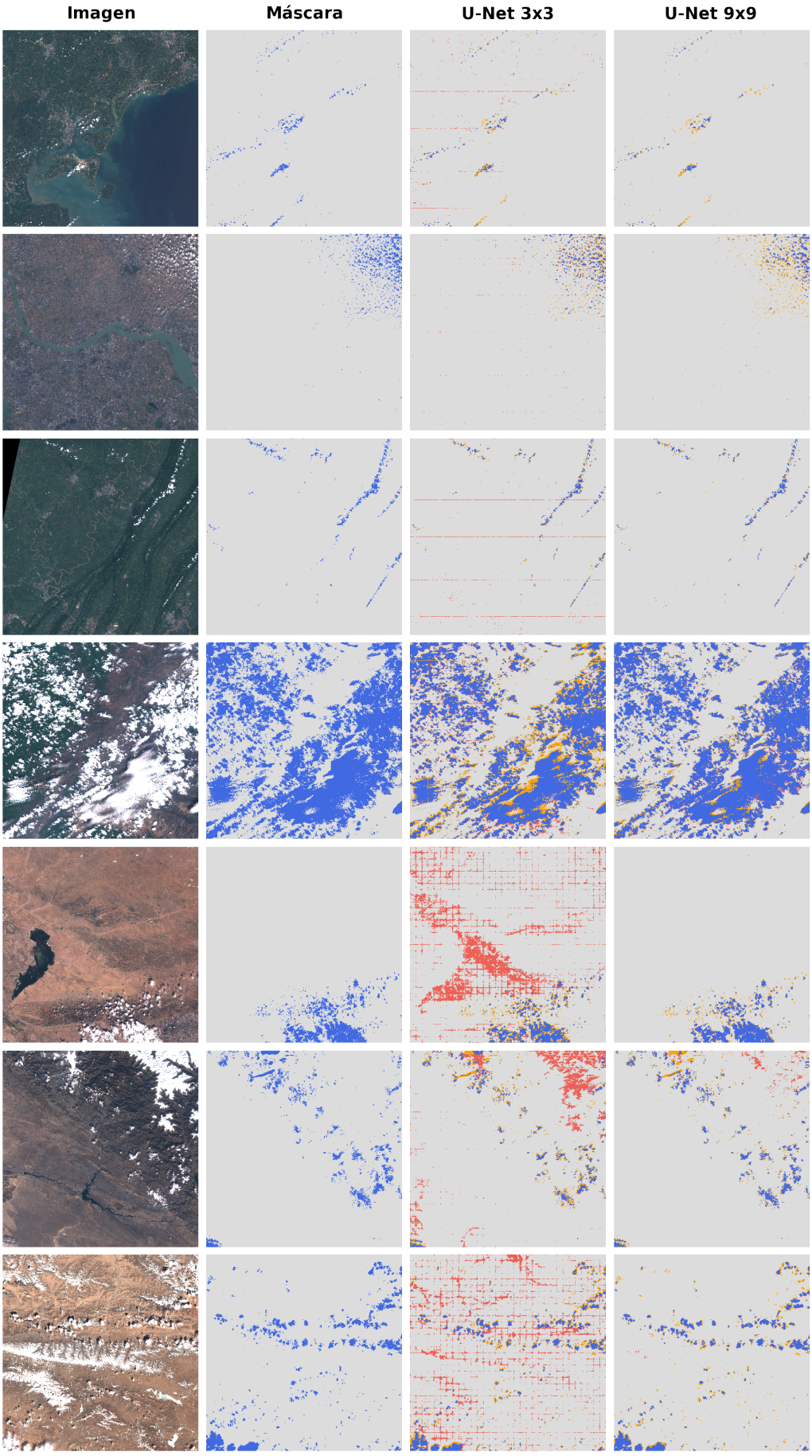


Imagen
Máscara
U-Net 3x3
U-Net 9x9

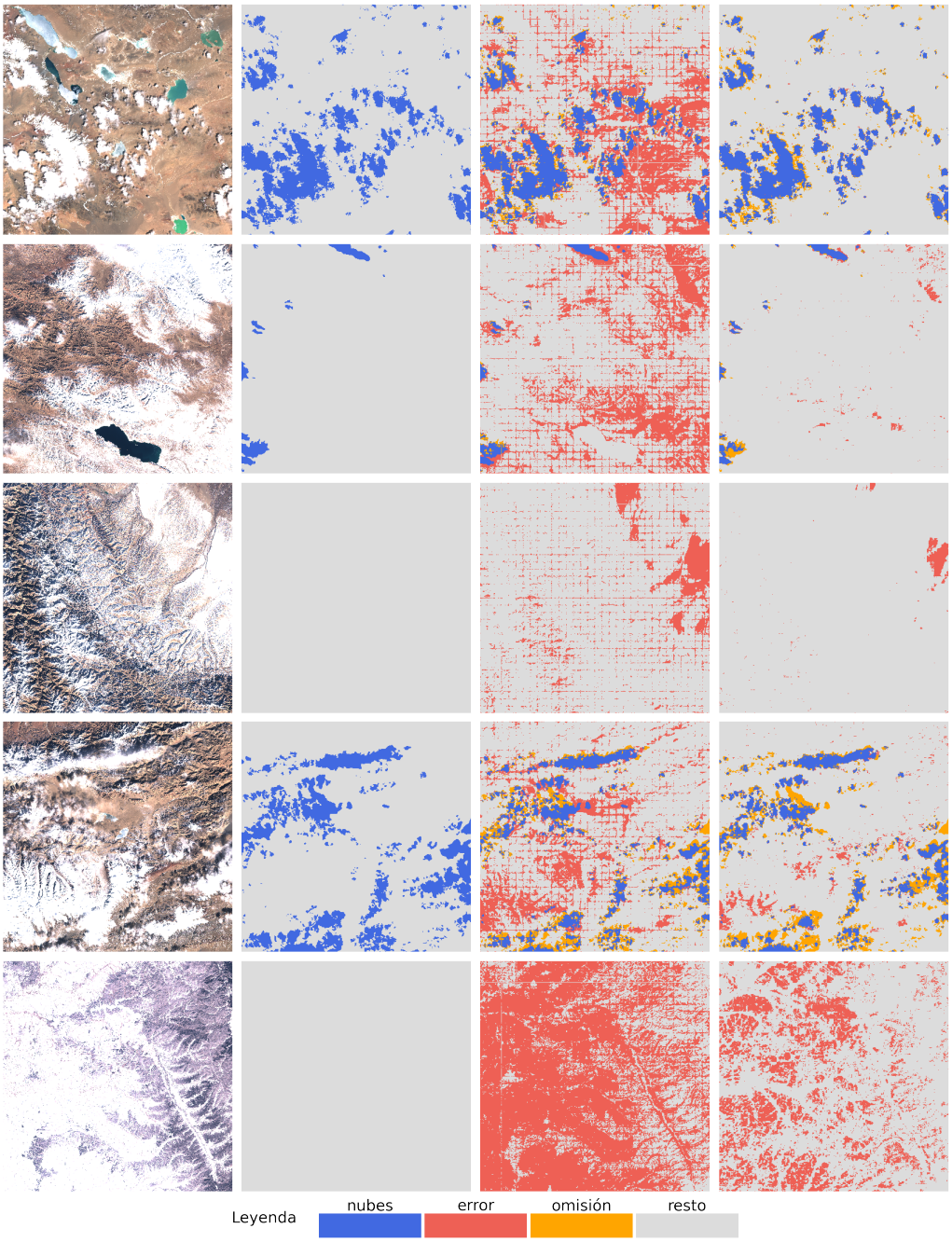


Figura 5.2: Comparativa visual del modelo U-Net con tamaño de filtros de 3x3 y de 9x9.

Tabla 5.3: Resultados del modelo U-Net 9x9.

img	OA (%)	OP (%)	AP (%)	Recall (%)	FI-score	IoU	MIoU	kappa
1	99,54	95,37	97,46	46,02	0,6208	0,4501	0,7227	0,6188
2	98,76	93,39	96,10	44,04	0,5986	0,4271	0,7073	0,5931
3	99,58	97,56	98,58	68,12	0,8022	0,6698	0,8328	0,8002
4	93,72	95,16	93,90	90,56	0,9280	0,8657	0,8800	0,8723
5	98,94	95,99	97,52	79,01	0,8668	0,7649	0,8769	0,8613
6	97,79	77,09	87,82	65,95	0,7108	0,5514	0,7644	0,6994
7	97,93	94,58	96,32	66,98	0,7842	0,6451	0,8118	0,7737
8	95,74	97,15	96,34	76,61	0,8566	0,7492	0,8502	0,8320
9	98,17	50,77	75,09	69,43	0,5865	0,4149	0,6982	0,5774
10	98,74	-	-	-	-	-	0,9874	-
11	88,59	74,64	82,60	52,67	0,6176	0,4468	0,6606	0,5529
12	74,80	-	-	-	-	-	0,7480	-
media	95,19	87,17	92,17	65,94	0,7372	0,5985	0,7950	0,7181

Tabla 5.4: Número de parámetros y tiempo de inferencia medio del modelo U-Net 9x9.

model param.	inf. CPU (s)
4.247.693	0,7947

5.2. Modificaciones U-Net

Después de estudiar las características que componen el modelo de U-Net clásico e implementarlo para la detección de nubes, se realiza una búsqueda bibliográfica específica sobre propuestas para mejorar los índices de precisión de la segmentación de nubes de las redes FCN. Como se ha visto en el apartado anterior, interesan especialmente aquellas modificaciones que permitan reducir el tamaño del modelo sin perder mucho rendimiento y aquellas que mejoren la clasificación en entornos de alto albedo como puede ser la nieve.

Una de las técnicas que se utiliza para diferenciar nubes de la nieve es aplicar la fusión de características en multiescala (MFF) [105]. El MFF puede capturar y agregar características a varias escalas, favoreciendo que las características semánticas de alto nivel extraídas de las nubes y la nieve sean más distintivas. El módulo básico llamado CBR está compuesto por una capa de convolución, una capa de normalización por lotes (batch normalization) y una capa de activación ReLU. La capa de normalización por lotes es una capa que se utiliza para mantener la salida media cercana a 0 y la desviación estándar de salida cercana a 1, consiguiendo en algunos casos, reducir el número de épocas necesarias en el entrenamiento y proporcionando cierta regularidad, reduciendo el error de generalización [113]. Es importante destacar que la normalización por lotes funciona de forma diferente durante el entrenamiento y durante la inferencia. En el entrenamiento, la capa normaliza su salida utilizando la media y la desviación estándar del lote (batch) actual. En cambio, en la inferencia, lo hace utilizando una media móvil de la media y la desviación estándar calculada en el entrenamiento. El módulo CR es idéntico al CBR, salvo que carece de la capa de normalización por lotes.

En la primera etapa, el MFF utiliza el módulo básico CBR con tamaño de filtro de 3 × 3 para cambiar el número de canales del mapa de características. En segundo lugar, se realiza la extracción de características a diferentes escalas añadiendo en canales paralelos los módulos básicos CR de tamaño 1 × 1, 3 × 3 y doble 3 × 3. En la última etapa, se concatenan

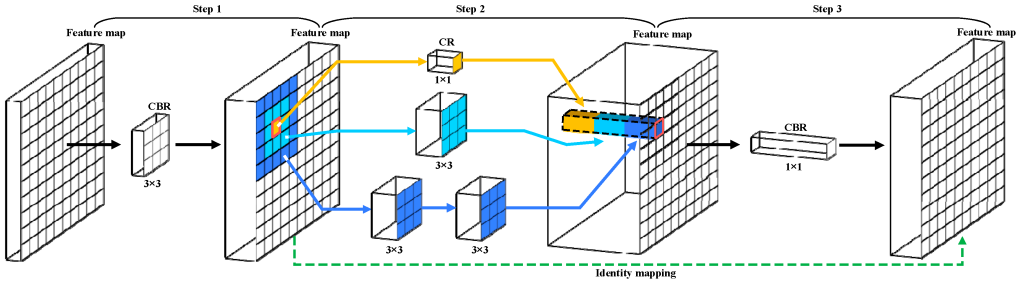


Figura 5.3: Arquitectura general del MFF [105].

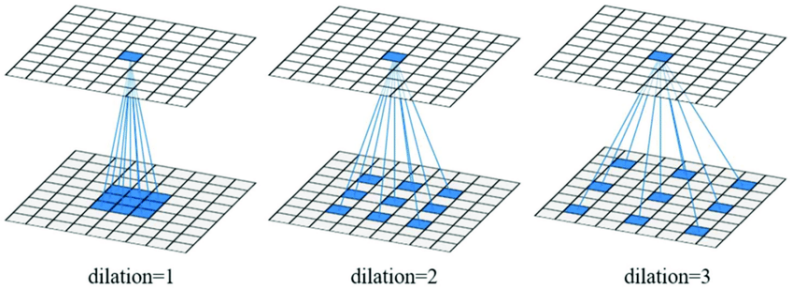


Figura 5.4: Representación gráfica de diferentes tasas de dilatación [115].

la salida de los 3 módulos y se conducen al módulo básico CBR de tamaño 1×1 para reducir la profundidad del mapa de características, consiguiendo comprimir la información de las diferentes escalas. De esta manera, el módulo es capaz de extraer y fusionar características de diversa granularidad, permitiendo mejorar los enlaces entre características de diferentes escalas [114]. Por último, se añade una conexión residual entre los mapas de características generados en la primera y tercera etapa creando el mapa de identidad, es decir, garantizando que la salida de la red sea igual a su entrada [83]. Con todo ello, se consigue que la red sea capaz de extraer más información semánticamente discriminativa con el fin de minimizar el impacto de introducir tipos de terreno similares, tanto a nivel espectral como espacial. En la Figura 5.3 puede observase de manera gráfica la arquitectura general del módulo MFF.

El módulo MFF se aplica tanto en la etapa de codificación como en la decodificación, cambiando de ubicación la conexión de salto (skip connection) y la profundidad de los mapas de características. Además, entre ambas etapas se añade un módulo llamado MFF dilatado, compuesto por dos módulos MFF modificados de codificación y decodificación. El módulo MFF dilatado se diferencia del básico en que no se añade la capa de *max-pooling* ni la capa de normalización por lotes. La tasa de dilatación en las capas de convolución es de 1 para el MFF de codificación y de 2 para el de decodificación. El factor de dilatación indica lo grandes que son los huecos entre los elementos de un mapa de características sobre el que se aplica el filtro de convolución. Por defecto, se toma en uno, es decir, se aplica el filtro completo, sin huecos. En cambio, si la dilatación es de 2, aplicando el filtro se deja un hueco de tamaño 1 y así sucesivamente (Figura 5.4). La ventaja de utilizar este tipo de filtros es que se consigue un campo de visión más amplio con el mismo coste computacional. El la Figura 5.5 puede verse en detalle los módulos MFF de codificación y de decodificación.

Los resultado experimentales del modelo de fusión de características en multiescala descritos en este apartado están recogidos en la Tabla 5.5. Cabe destacar que para referenciar este modelo más adelante, se utilizará el nombre de SatCloud. En esta ocasión, se observa que en todo tipo de imágenes se obtienen unos resultados mejores que para el caso de la U-Net base incluso con filtros 9x9, especialmente en nieve, donde se consigue mantener

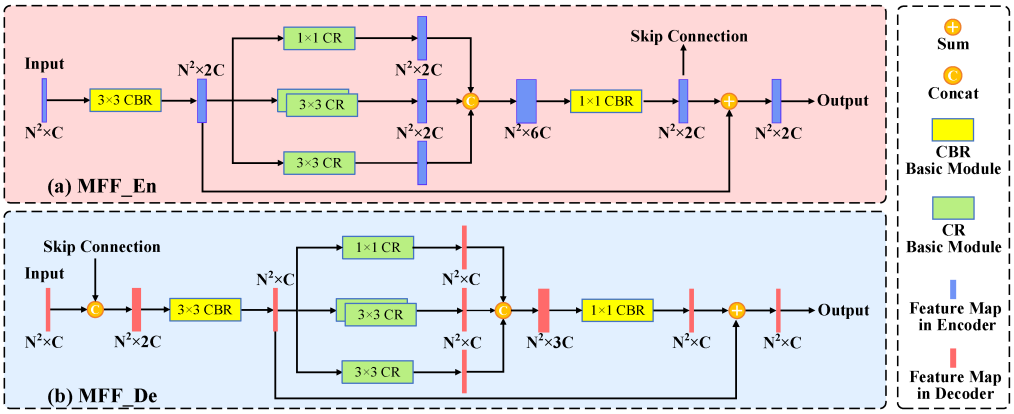


Figura 5.5: Diagrama de bloques de los módulos MFF de codificación y de decodificación [105].

Tabla 5.5: Resultados del modelo SatCloud.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	99,61	97,61	98,62	54,57	0,7001	0,5385	0,7673	0,6983
2	98,82	97,37	98,10	44,91	0,6147	0,4437	0,7159	0,6095
3	99,60	98,30	98,96	68,98	0,8107	0,6817	0,8388	0,8088
4	94,57	98,20	95,14	89,51	0,9366	0,8807	0,8951	0,8893
5	99,06	99,03	99,04	79,21	0,8802	0,7860	0,8881	0,8753
6	98,60	95,60	97,14	69,10	0,8022	0,6697	0,8276	0,7951
7	98,31	98,13	98,22	71,36	0,8263	0,7040	0,8432	0,8176
8	96,45	98,01	97,10	80,24	0,8824	0,7896	0,8743	0,8617
9	99,07	84,73	92,00	61,28	0,7112	0,5519	0,7712	0,7066
10	99,64	-	-	-	-	-	0,9964	-
11	90,23	87,29	88,93	51,69	0,6493	0,4807	0,6867	0,5968
12	92,64	-	-	-	-	-	0,9264	-
media	97,22	95,43	96,33	67,09	0,7814	0,6526	0,8359	0,7659

el rendimiento del modelo en valores más que aceptables para la aplicación requerida. La exactitud está por encima del 90 % en todas las imágenes de test y las métricas que penalizan más la detección de falsos positivos, como son el F1 y el IoU, también se mantienen en valores altos. Respecto al número de parámetros (Tabla 5.6), se comprueba que el modelo SatCloud es el más complejo de los vistos hasta ahora, pero los tiempos de inferencia en CPU se mantienen en valores aceptables. La explicación de que los tiempos sean bajos a pesar del número de parámetros, viene motivada por el hecho de que las convoluciones multiescala se aplican en paralelo, es decir, en el caso de las CPUs pueden calcularse las diferentes escalas en varios núcleos de forma independiente. Este tipo de arquitecturas son un buen ejemplo de que no en todos los casos el número de parámetros está directamente relacionado con los tiempos de inferencia, sino en el propio diseño del modelo. Comparando este resultado con los obtenidos en la U-Net (Tabla 5.2 y Tabla 5.4), puede verse que el número de parámetros del modelo multiescala es prácticamente el doble que el de filtros de tamaño 9x9 pero el tiempo de inferencia es 3,66 veces menor. La U-Net de tamaño 3x3 sigue siendo la que consigue el menor tiempo de inferencia pero no se puede tener en cuenta debido a la escasa capacidad de diferenciar la nieve de las nubes y el problema de los bordes al calcular las convoluciones de los parches.

Como se ha visto hasta ahora, el modelo SatCloud a nivel de métricas ofrece unos re-

Tabla 5.6: Número de parámetros y tiempo de inferencia medio del modelo SatCloud.

model param.	**inf. CPU (s)**
7.463.425	0,2172

sultados más que suficientes para la aplicación requerida. Sin embargo, la complejidad del modelo es elevada, comprometiendo la viabilidad de su implementación en un sistema embebido para su ejecución en tiempo real. Además, teniendo en cuenta que la ejecución se llevará a cabo en un satélite, cuanto menos complejo sea el modelo, menor consumo implicará, siendo un punto importante en aplicaciones aeroespaciales. Así, en lo que sigue, se describen algunos experimentos realizados a este fin.

En primer lugar, se ha estudiado la posibilidad de eliminar bandas espectrales del espacio de entrada del sistema. La idea tras esta simplificación se apoya en el hecho de que, según el análisis de separabilidad espectral realizado previamente, los buenos índices de clasificación obtenidos con SatCloud podrían ser producto fundamentalmente de la información espacial multiescalar extraída por las técnicas de convolución incorporadas en el modelo. Así, con está prueba se ha querido analizar en qué medida la información espectral está siendo relevante en la capacidad de clasificación de estas redes de tipo FCN frente a la información espacial extraída en las capas de convolución ya que, en caso de resultar poco relevante, podrían eliminarse algunas bandas para aligerar la complejidad computacional de los modelos. Para realizar este experimento se ha escogido la banda azul, puesto que como se ha visto en el capítulo 3, sin tener en cuenta las bandas SWIR, la región espectral azul es la más relevante en términos de detección de nubes. Como se observa en la Tabla 5.8, los resultados utilizando solo la banda espectral azul ya no son consistentemente buenos en comparación a utilizar las cuatro bandas VNIR. Es especialmente relevante la métrica IoU puesto que se sitúa de media en 0,42 pero en una imagen de test, en la que hay presencia de nubes y nieve, llega a situarse en 0,11. Además, el número de parámetros y el tiempo de inferencia no se ven significativamente reducidos respecto al que utiliza las cuatro bandas espectrales (Tabla 5.8). Este hecho es debido a que el modelo solo ve reducido el número de parámetros en la primera etapa, es decir, en la entrada de la primera convolución. Después no se ve afectado, ya que prevalece el número de filtros de convolución escogidos sin que esté condicionado por el número de bandas espectrales introducidas. Con todo ello, se puede concluir que el modelo con solo la banda espectral azul no es una opción válida para la simplificación del modelo, demostrando además que la información espectral aportada por las cuatro bandas VNIR, si bien insuficientes por sí mismas, sí son relevantes cuando se combinan con la información espacial extraída en las FCN.

5.3. Optimización de los hiperparámetros del modelo

En este apartado, se describe el procedimiento seguido para la optimización de los hiperparámetros del modelo SatCloud de referencia. El objetivo es doble: por un lado, explorar si es posible mejorar los resultados de la detección de nubes, por otro, analizar en qué medida puede simplificarse el modelo sin comprometer la precisión. Para realizar una búsqueda exhaustiva (grid search) del espacio de diseño, se han tomado los siguientes hiperparámetros: tamaño de los filtros de convolución, profundidad de la red y tamaño de los parches de las imágenes de entrada. En la Tabla 5.9 se muestran los valores posibles asignados a cada parámetro. Así, en total se ha definido un espacio de búsqueda de 18 posibles combinaciones.

Tabla 5.7: Resultados del modelo SatCloud utilizando únicamente la banda espectral azul.

img	**OA (%)**	**OP (%)**	**AP (%)**	**Recall (%)**	**F1-score**	**IoU**	**MIoU**	**kappa**
1	99,49	96,73	98,11	39,48	0,5608	0,3896	0,6922	0,5586
2	98,65	95,01	96,85	37,93	0,5421	0,3719	0,6791	0,5366
3	99,23	96,00	97,62	39,83	0,5630	0,3918	0,6920	0,5598
4	87,00	99,93	90,45	70,99	0,8301	0,7096	0,7595	0,7296
5	98,24	97,25	97,76	61,56	0,7539	0,6050	0,7935	0,7453
6	95,94	50,92	74,19	40,29	0,4499	0,2902	0,6245	0,4291
7	93,39	44,31	71,18	68,46	0,5380	0,3680	0,6496	0,5041
8	90,80	71,12	83,05	75,14	0,7307	0,5757	0,7353	0,6753
9	87,14	11,36	55,53	86,11	0,2007	0,1115	0,4904	0,1733
10	93,58	-	-	-	-	-	0,9358	-
11	78,91	43,36	67,75	67,20	0,5271	0,3579	0,5594	0,3994
12	56,83	-	-	-	-	-	0,5683	-
media	89,93	70,60	83,25	58,70	0,5696	0,4171	0,6817	0,5311

Tabla 5.8: Número de parámetros y tiempo de inferencia medio del modelo SatCloud utilizando únicamente la banda espectral azul.

model param.	**inf. CPU (s)**
7.462.561	0,2223

Tabla 5.9: Combinaciones para la optimización de los hiperparámetros de la red.

Hiperparámetros	**Valores**		
Tamaño de los parches de las imágenes de entrada	168 x 168	336 x 336	
Tamaño de los filtros de convolución	1 x 1	3 x 3	5 x 5
Profundidad de la red	1	2	3

Tabla 5.10: Resultados del análisis de hiperparámetros.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
SatCloud 168 1 1	79,89	59,89	77,71	59,53	0,5224	0,3810	0,5879	0,4728
SatCloud 168 1 2	79,79	64,23	79,63	56,01	0,5244	0,3826	0,5876	0,4758
SatCloud 168 1 3	82,81	66,89	81,07	56,78	0,5587	0,4170	0,6226	0,5168
SatCloud 168 3 1	87,08	72,23	83,86	57,01	0,5698	0,4214	0,6587	0,5324
SatCloud 168 3 2	86,67	66,23	80,73	53,23	0,5117	0,3655	0,6350	0,4690
SatCloud 168 3 3	85,87	69,28	82,66	60,43	0,5821	0,4385	0,6551	0,5462
SatCloud 168 5 1	88,84	77,22	86,49	58,06	0,6051	0,4569	0,6849	0,5726
SatCloud 168 5 2	90,21	77,18	86,70	61,33	0,6325	0,4872	0,7086	0,6035
SatCloud 168 5 3	90,71	78,59	87,25	59,13	0,6256	0,4782	0,7098	0,5960
SatCloud 336 1 1	80,28	63,32	79,30	56,24	0,5268	0,3863	0,5910	0,4814
SatCloud 336 1 2	79,67	63,16	79,64	64,67	0,5625	0,4247	0,6024	0,5160
SatCloud 336 1 3	81,33	65,04	80,26	58,47	0,5436	0,4000	0,6041	0,5004
SatCloud 336 3 1	87,77	65,76	80,93	60,25	0,5769	0,4302	0,6663	0,5425
SatCloud 336 3 2	89,65	75,11	85,53	60,10	0,6118	0,4642	0,6969	0,5794
SatCloud 336 3 3	89,43	73,74	84,90	59,93	0,6101	0,4640	0,6934	0,5789
SatCloud 336 5 1	91,31	81,20	88,69	60,62	0,6503	0,5043	0,7239	0,6234
SatCloud 336 5 2	93,01	84,39	90,23	60,05	0,6667	0,5159	0,7434	0,6408
SatCloud 336 5 3	91,71	81,46	89,07	64,58	0,6841	0,5421	0,7411	0,6602

Para llevar a cabo el proceso de búsqueda en un tiempo razonable, se han limitado los entrenamientos a 300 épocas, ya que esto es suficiente para poder realizar un estudio comparativo. Una vez se decidan los posibles candidatos que mejor cumplen los propósitos de este apartado, se llevará a cabo un entrenamiento completo de estos modelos y se tomará la decisión final sobre cuál es el modelo que mejor relación entre precisión y complejidad muestra. En la Tabla 5.10 se visualizan los resultados de todas las combinaciones posibles con los diferentes hiperparámetros tomados de la red. El resultado que se muestra es la media entre las 12 imágenes de test para cada modelo. El nombre de los modelos describe los hiperparametros que se han tomado, siendo en primer lugar el tamaño de los parches, seguido del tamaño de los filtros de convolución y finalmente la profundidad de la red.

Analizando en detalle la relación entre los hiperparámetros, se observa que reducir el tamaño de los parches disminuye el rendimiento del modelo en todas las métricas, aunque esta reducción tampoco es excesiva. El tiempo de inferencia en CPU sí que se ve reducido a la mitad pero, en la aplicación final, puede que no existan estas diferencias tan significativas, ya que el procesado final se llevará a cabo en un MPSoC, posibilitando la paralelización de las distintas partes de la red (Tabla 5.11). Respecto a la profundidad de la red, entre escoger una de 3 o de 2 no se aprecia ninguna mejora e incluso en alguna métricas empeora el modelo al aumenta la profundidad de 2 a 3. Reducir a 1 sí que se observa un decremento del rendimiento, salvo para filtros de 5 x 5 que se ve muy poco reducida. Este hecho puede deberse a que el entrenamiento al haber sido relativamente corto, no ha dado tiempo a que desarrolle una compresión superior con una profundidad mayor, ya que el entrenamiento con filtros 5 x 5 es más complejo. En relación al tamaño de los filtros, si que se observa una implicación directa en el rendimiento, siendo mayor cuanto más grande sea el tamaño de los filtros. Lo malo de incrementar el tamaño de los filtros es que aumenta de manera muy significativa el número de parámetros, incrementando los tiempos de inferencia de igual manera.

El modelo con parches de 336 x 336, filtros de 5x5 y profundidad de 2 es el que presenta el mayor valor en las métricas que destacan por castigar más la detección de falsos positivos, es decir, las métricas F1-score, IoU, MIoU y kappa. Respecto a la exactitud y precisión de modelo, no es el que presenta el valor más alto pero si que destaca entre los mejores. Res-

Tabla 5.11: Número de parámetros y tiempo de inferencia medio de cada modelo entrenado para el análisis de hiperparámetros.

img	model param.	inf. CPU (s)
SatCloud 168 1 1	115.521	0,0774
SatCloud 168 1 2	1.491.329	0,0867
SatCloud 168 1 3	6.004.225	0,1027
SatCloud 168 3 1	427.841	0,0770
SatCloud 168 3 2	1.836.417	0,0987
SatCloud 168 3 3	7.463.425	0,1147
SatCloud 168 5 1	1.052.481	0,0950
SatCloud 168 5 2	4.492.673	0,1324
SatCloud 168 5 3	18.246.145	0,1516
SatCloud 336 1 1	115.521	0,1291
SatCloud 336 1 2	1.491.329	0,1708
SatCloud 336 1 3	6.004.225	0,1886
SatCloud 336 3 1	427.841	0,1630
SatCloud 336 3 2	1.836.417	0,1952
SatCloud 336 3 3	7.463.425	0,2604
SatCloud 336 5 1	1.052.481	0,2448
SatCloud 336 5 2	4.492.673	0,3281
SatCloud 336 5 3	18.246.145	0,3747

pecto al tiempo de inferencia, esta combinación de hiperparámetros es una en las que más tiempo se debe invertir para realizar la segmentación pero sigue siendo un tiempo asumible para las especificaciones requeridas en este proyecto. El número de parámetros se sitúa en la zona media debido a que el mayor cambio que afecta a este valor es la profundidad de la red y esta se instancia en 2. Por todo ello, este es el modelo de hiperparametrización que se ha escogido como el más adecuado para su implementación final.

5.4. Análisis del modelo final propuesto para implementación

Por último, se propone el modelo final en base al análisis de hiperparámetros realizado. Como se ha detallado en la sección anterior, se ha optado por implementar el modelo con parches de 336 x 336, filtros de 5x5 y profundidad de 2. Ya que el módulo MFF dilatado utiliza capas convolucionales dilatadas, se prueba si manteniendo los filtros de convolución 3x3 en solo ese módulo se logra mantener el buen rendimiento del modelo, reduciendo su complejidad.

En la Tabla 5.12 se observan las métricas del modelo por cada imagen de test. El modelo mantiene unos buenos resultados en la mayoría de imágenes salvo en las que hay presencia de nieve, especialmente en la imagen de test 12 donde la exactitud global decrece hasta un 62,65 %. Es por ello que este modelo se ha considerado que no puede ser válido para el objetivo de este proyecto, ya que el mal resultado que ofrece para la imagen 12 puede ser un indicativo de que no es tan robusto en cualquier tipo de situación. Descartado este modelo, se implementa el modelo con filtros 5x5 en todos los módulos del modelo, manteniendo una profundidad de red de 2.

En la Tabla 5.14 puede verse el rendimiento del modelo propuesto. En esta ocasión, si que muestra unos buenos resultados en todas las imágenes de test. Respecto al modelo con

Tabla 5.12: Resultados del modelo SatCloud 5x5 excepto en el módulo MFF dilatado que es de 3x3 y profundidad 2.

img	**OA (%)**	**OP (%)**	**AP (%)**	**Recall (%)**	**F1-score**	**IoU**	**MIoU**	**kappa**
1	99,60	97,98	98,79	52,42	0,6830	0,5186	0,7573	0,6811
2	98,89	91,07	95,03	52,40	0,6652	0,4984	0,7436	0,6600
3	99,54	98,00	98,78	64,51	0,7781	0,6368	0,8161	0,7759
4	92,93	98,14	93,86	85,84	0,9158	0,8446	0,8649	0,8554
5	98,91	98,74	98,83	75,97	0,8587	0,7524	0,8705	0,8531
6	98,05	84,82	91,65	63,97	0,7294	0,5740	0,7770	0,7195
7	97,87	94,34	96,18	66,10	0,7774	0,6358	0,8068	0,7665
8	95,92	96,74	96,26	78,07	0,8641	0,7607	0,8569	0,8404
9	97,30	38,64	69,08	75,07	0,5102	0,3425	0,6575	0,4978
10	98,66	-	-	-	-	-	0,9866	-
11	88,26	71,43	81,15	54,80	0,6202	0,4495	0,6598	0,5522
12	62,65	-	-	-	-	-	0,6265	-
media	94,05	86,99	91,96	66,92	0,7402	0,6013	0,7853	0,7202

Tabla 5.13: Número de parámetros y tiempo de inferencia medio del modelo SatCloud 5x5 excepto en el módulo MFF dilatado que es de 3x3 y profundidad 2.

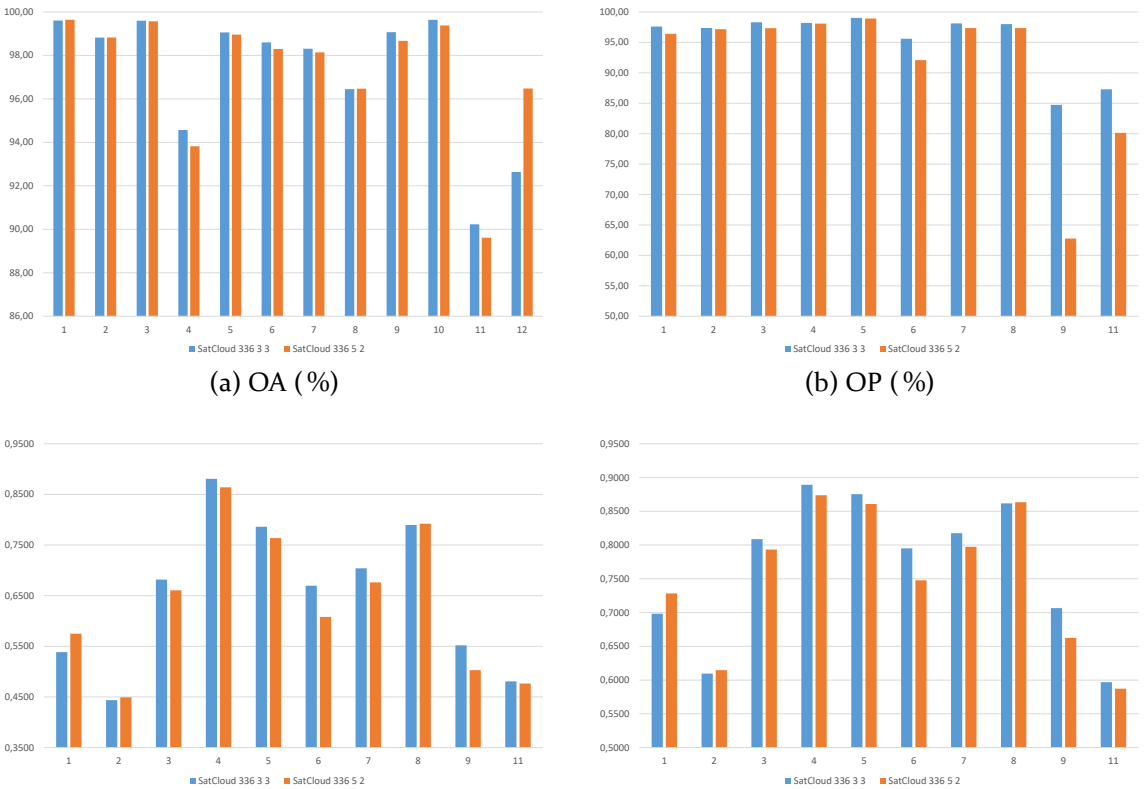
model param.	**inf. CPU (s)**
2.526.593	0,2877

Tabla 5.14: Resultados del modelo SatCloud 5x5 y profundidad 2.

img	**OA (%)**	**OP (%)**	**AP (%)**	**Recall (%)**	**F1-score**	**IoU**	**MIoU**	**kappa**
1	99,64	96,41	98,03	58,75	0,7301	0,5749	0,7857	0,7284
2	98,83	97,17	98,01	45,50	0,6198	0,4491	0,7186	0,6146
3	99,57	97,34	98,47	67,26	0,7955	0,6605	0,8281	0,7934
4	93,82	98,10	94,53	87,89	0,9271	0,8641	0,8811	0,8737
5	98,96	98,93	98,94	77,02	0,8661	0,7638	0,8765	0,8607
6	98,30	92,10	95,29	64,16	0,7563	0,6081	0,7953	0,7478
7	98,14	97,35	97,76	68,89	0,8069	0,6763	0,8285	0,7974
8	96,47	97,36	96,84	80,95	0,8840	0,7921	0,8756	0,8634
9	98,67	62,77	81,11	71,65	0,6692	0,5028	0,7447	0,6624
10	99,38	-	-	-	-	-	0,9938	-
11	89,61	80,12	85,50	54,02	0,6453	0,4764	0,6808	0,5872
12	96,48	-	-	-	-	-	0,9648	-
media	97,32	91,76	94,45	67,61	0,7700	0,6368	0,8311	0,7529

Tabla 5.15: Número de parámetros y tiempo de inferencia medio del modelo SatCloud 5x5 y profundidad 2.

model param.	**inf. CPU (s)**
4.492.673	0,3016



(a) OA (%)

(b) OP (%)

(c) IoU

(d) kappa

Figura 5.6: Comparativa de los modelos SatCloud 336 3 3 y SatCloud 336 5 2.

filtros 3x3 probado en el apartado 5.2, se señalan unos resultados un poco peores en todas las métricas (salvo en la exactitud global) pero la diferencia es mínima, siendo prácticamente despreciable. Gráficamente puede verse en la Figura 5.6, dónde se representan las principales métricas de ambos modelos. Respecto al tamaño del modelo, se logra reducir el número de parámetros de 7.463.425 a 4.492.673 (Tabla 5.15). En las Figuras 5.7 pueden verse de manera visual las diferencias del modelo propuesto respecto al de filtros de tamaño 3x3.

5.5. Validación del modelo con otra base de datos

Como se ha explicado, los clasificadores desarrollados con técnicas de ML pueden verse condicionados por los datos utilizados para su entrenamiento. Si bien la técnica de separar los datos en conjuntos de entrenamiento/validación y test, además de otras técnicas de regularización que puedan implementarse en los propios algoritmos de aprendizaje, tienen como objetivo garantizar la capacidad de generalización de los modelos obtenidos, en este proyecto se ha considerado oportuno realizar una validación cruzada del modelo SatCloud con alguna base de datos totalmente diferente a la utilizada durante la experimentación para su desarrollo. Tomando como referencia las bases de datos que utilizan en CMIX [116] para la comparación de los algoritmos de detección de nubes, se utiliza la base de datos *PixBox S2* que está disponible de forma libre para todos los investigadores [117]. Las máscaras de nubes provienen del algoritmo que utiliza la ESA para Sentinel-2 y están disponibles en el nivel 1C de procesamiento. Dichas máscaras cumplen el propósito de detectar nubes para el descartado automático de imágenes pero no son tan precisas como las que se han utilizado de referencia para los entrenamientos (WHUS2-CD+), ya que estas estaban clasifi-

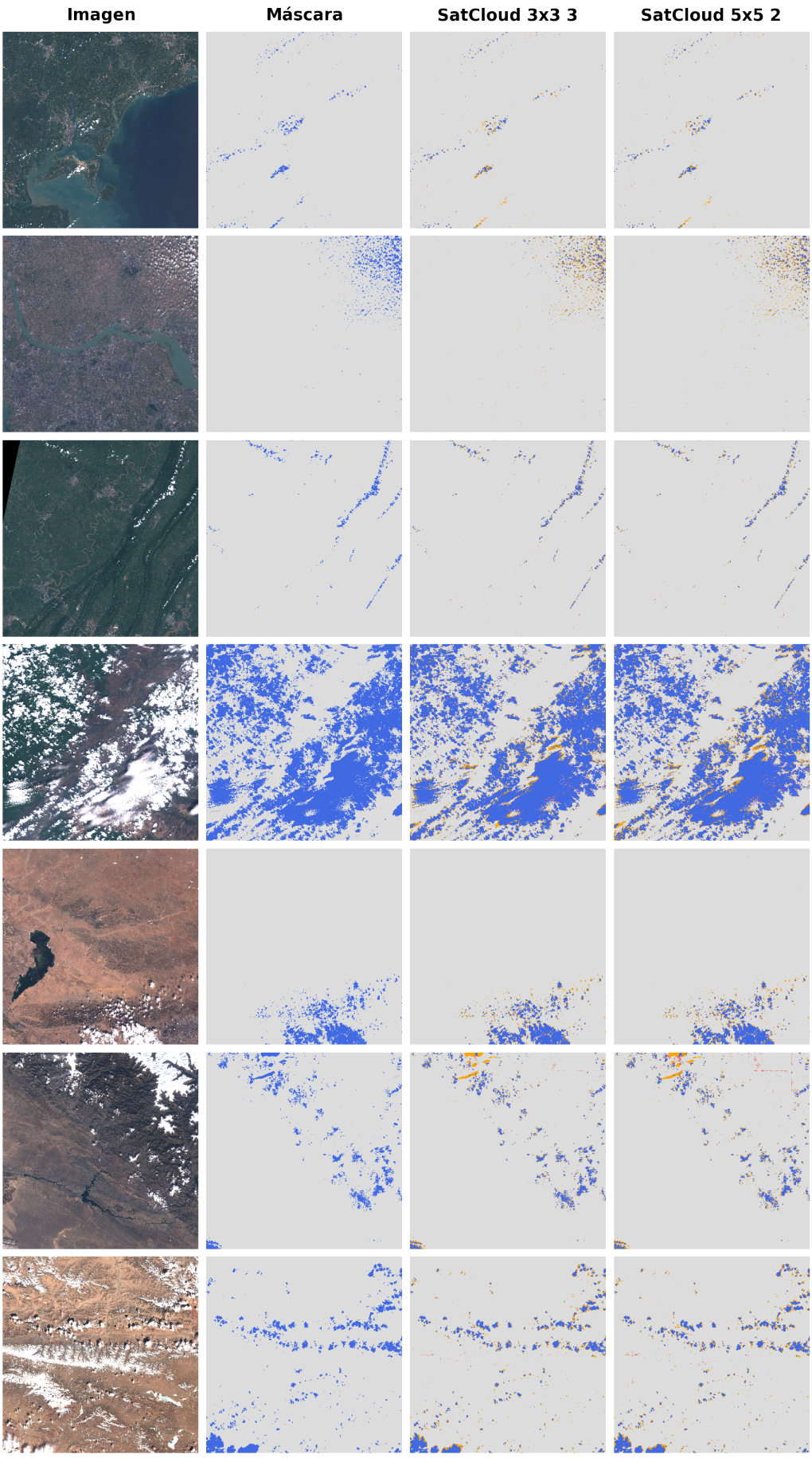


Imagen
Máscara
SatCloud 3x3 3
SatCloud 5x5 2

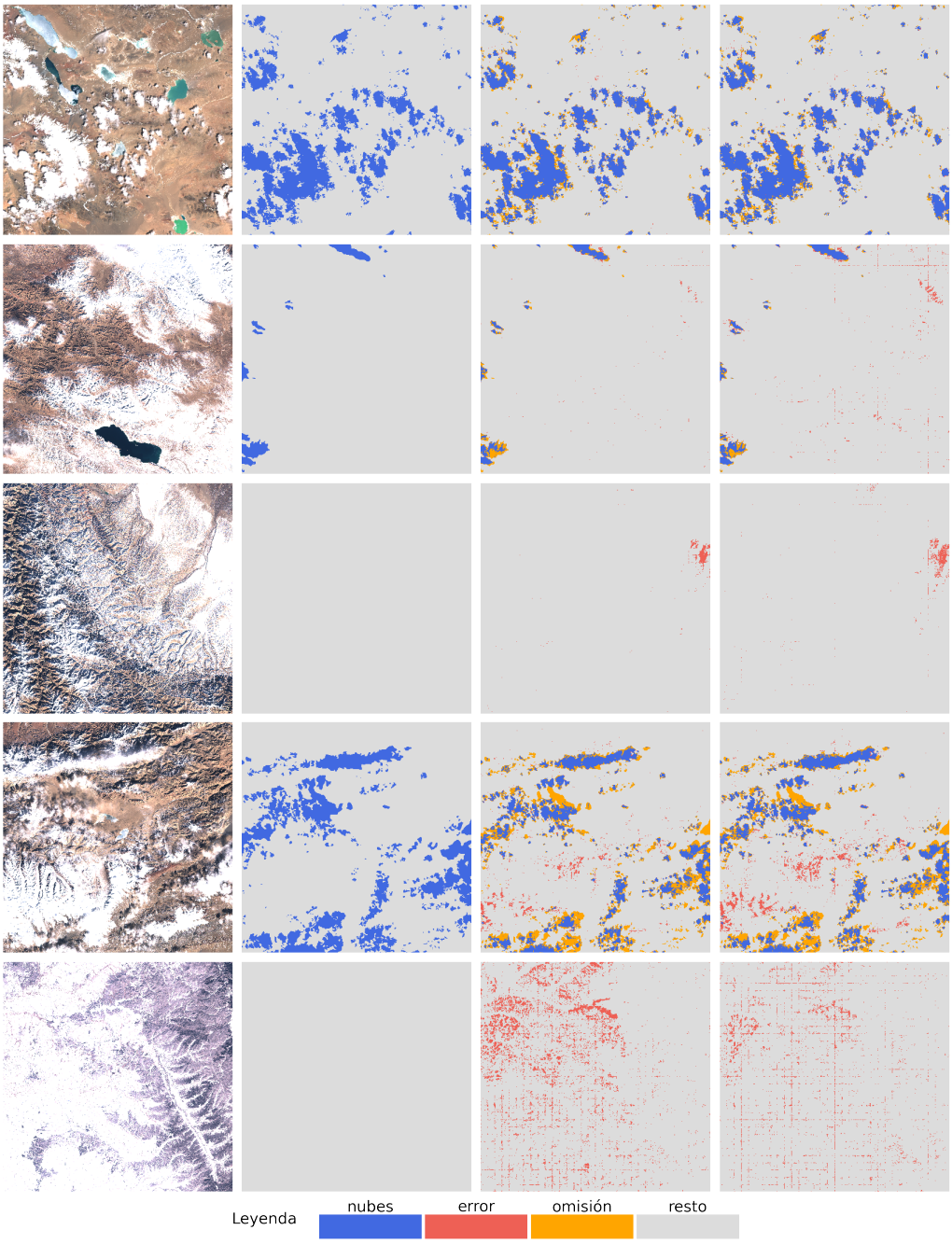


Figura 5.7: Comparativa visual del modelo SatCloud con tamaño de filtros de 3x3 profundidad de 3, y con tamaño de filtros de 5x5 profundidad de 2.

Tabla 5.16: Resultados del modelo SatCloud propuesto utilizando la base de datos PixBox S2.

img	OA (%)	OP (%)	AP (%)	Recall (%)	F1-score	IoU	MIoU	kappa
1	96,89	0,73	50,34	32,69	0,0143	0,0072	0,4880	0,0130
2	96,19	74,33	86,21	77,07	0,7567	0,6087	0,7841	0,7361
3	95,49	79,58	88,05	59,81	0,6829	0,5185	0,7356	0,6592
4	87,94	79,08	86,42	89,25	0,8386	0,7220	0,7732	0,7429
5	82,50	67,02	78,22	73,84	0,7027	0,5416	0,6605	0,5791
6	87,59	81,51	87,11	90,42	0,8574	0,7503	0,7762	0,7480
7	92,90	28,69	61,77	14,37	0,1915	0,1059	0,5172	0,1587
8	90,47	94,04	85,90	93,76	0,9390	0,8851	0,7634	0,7207
9	89,05	86,84	88,33	74,81	0,8037	0,6719	0,7654	0,7284
10	91,04	95,72	78,90	93,99	0,9484	0,9019	0,6963	0,6070
11	67,70	77,08	66,85	67,73	0,7210	0,5638	0,5046	0,3412
12	79,10	81,97	79,64	69,33	0,7512	0,6015	0,6481	0,5732
13	29,33	63,89	45,67	4,55	0,0850	0,0444	0,1568	-0,0121
14	96,15	96,26	96,01	98,79	0,9750	0,9513	0,8978	0,8907
15	91,26	71,60	84,44	88,92	0,7933	0,6574	0,7762	0,7387
16	83,18	83,42	83,23	72,00	0,7729	0,6299	0,6971	0,6404
17	93,74	2,25	49,70	2,84	0,0251	0,0127	0,4750	-0,0068
18	86,26	79,05	87,71	96,83	0,8704	0,7705	0,7578	0,7274
19	85,19	86,05	85,47	65,65	0,7448	0,5933	0,7022	0,6431
20	59,15	69,18	60,85	49,07	0,5742	0,4027	0,4195	0,2030
21	96,05	81,48	90,26	94,48	0,8750	0,7778	0,8660	0,8517
22	81,41	86,61	81,57	77,58	0,8185	0,6927	0,6863	0,6291
23	92,10	68,24	83,65	95,51	0,7960	0,6612	0,7839	0,7487
24	87,01	66,53	82,55	96,32	0,7870	0,6488	0,7389	0,6980
25	89,98	57,17	78,37	97,56	0,7209	0,5636	0,7243	0,6648
26	86,53	66,26	82,36	96,19	0,7847	0,6457	0,7336	0,6914
27	92,11	79,13	87,62	86,24	0,8253	0,7026	0,8028	0,7745
28	91,16	78,67	87,57	90,46	0,8415	0,7264	0,8055	0,7807
29	63,94	77,69	70,16	16,61	0,2736	0,1585	0,3858	0,1514
media	84,88	71,04	78,45	71,26	0,6818	0,5696	0,6732	0,5663

cadas manualmente. Debido a ello, en los resultados de las métricas hay que tener en cuenta una desviación producida por las máscaras, ya que el modelo propuesto en determinadas ocasiones puede detectar mejor las nubes que el algoritmo de Sentinel-2.

Como puede verse en la Tabla 5.16, los resultados muestran una muy buena segmentación de imágenes con presencia de nubes, siendo superior a los modelos propuestos en CMIX (Tabla 5.17), a excepción de algunas imágenes. Para poder determinar la causa de que algunas imágenes no estén en los valores de métrica esperados, se visualizan las imágenes (Figuras 5.8). La imagen número 13 es la que ofrece peores resultados, siendo la exactitud global de 29,33 % y el coeficiente kappa negativo. Visualizando la imagen, puede verse que ha detectado toda la imagen prácticamente sin presencia de nubes. Cuando la imagen es completamente blanca, el modelo no puede determinar si hay presencia de nubes o de nieve, debido a que las características espaciales que puede extraer el modelo son prácticamente nulas. A pesar de ello, ante la duda la respuesta que da el modelo es que no hay presencia de nubes, evitando los falsos positivos. Por ello, se puede determinar que el modelo cumple correctamente para este tipo de imágenes en las que no es posible determinar de forma confiable la presencia de nubes. Del mismo modo ocurre para la imagen 29, aun-

Tabla 5.17: Resultados de los modelos propuestos por CMIX utilizando la base de datos PixBox S2 [116].

Algoritmos	**OA (%)**	**OP (%)**	**AP (%)**	**Recall (%)**
ATCOR* (27/29)	76,60	85,30	76,20	62,50
CD-FCNN	80,50	89,90	79,70	66,00
Fmask 4.0 CCA	84,50	86,50	84,20	79,40
FORCE	80,20	78,90	80,10	79,00
Idepix	75,70	69,70	76,30	85,90
InterSSIM	84,60	93,20	84,00	72,70
LaSRC	66,40	59,90	67,50	86,80
MAJA* (14/29)	85,10	80,20	85,50	88,60
S2cloudless	86,30	89,50	85,90	80,20
Sen2Cor	81,20	83,60	80,80	74,70
media	80,10	81,70	80,00	77,60
deviación estandar	5,70	9,60	5,30	8,30

que las métricas se ven menos afectadas, ya que hay menos nubes en la imagen. Para el resto de imágenes, las mayores diferencias están en los bordes de las imágenes y en algunas zonas de las imágenes, aunque como se ha mencionado anteriormente, gran parte de estas diferencias vienen causadas por las máscaras de referencia que no son lo suficientemente precisas. El número de parámetros y el tiempo de inferencia es el mismo que el calculado en el apartado anterior, ya que viene determinado por el modelo, no por la base de datos que se esté utilizando. Por todo ello, se puede concluir que el modelo multiescalar SatCloud con parches de 336 x 336, filtros de 5x5 y profundidad de 2 es válido y adecuado para los propósitos de este proyecto.

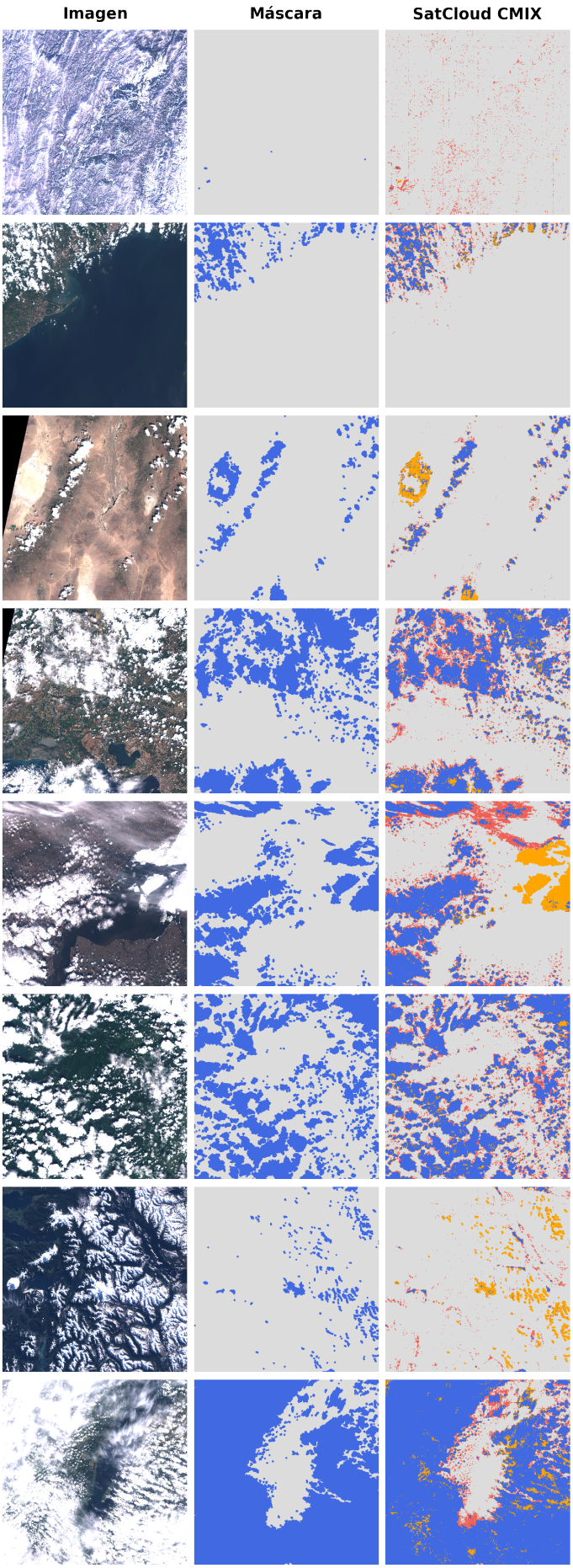
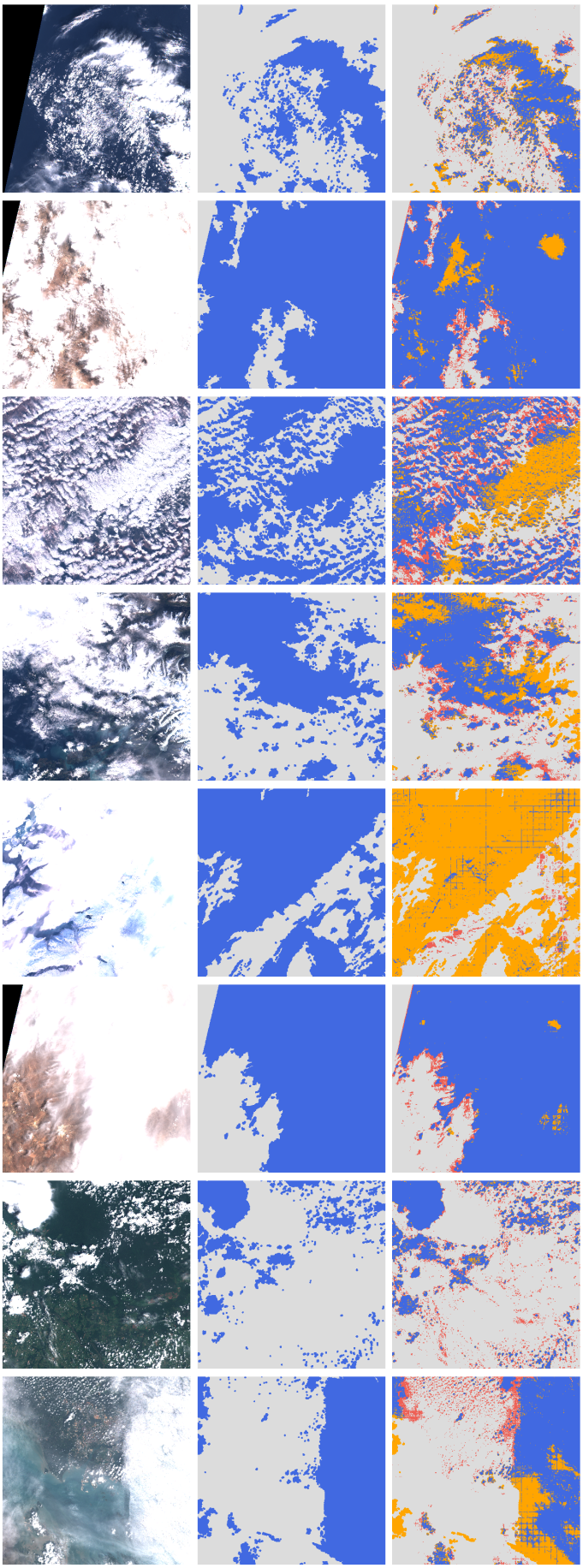
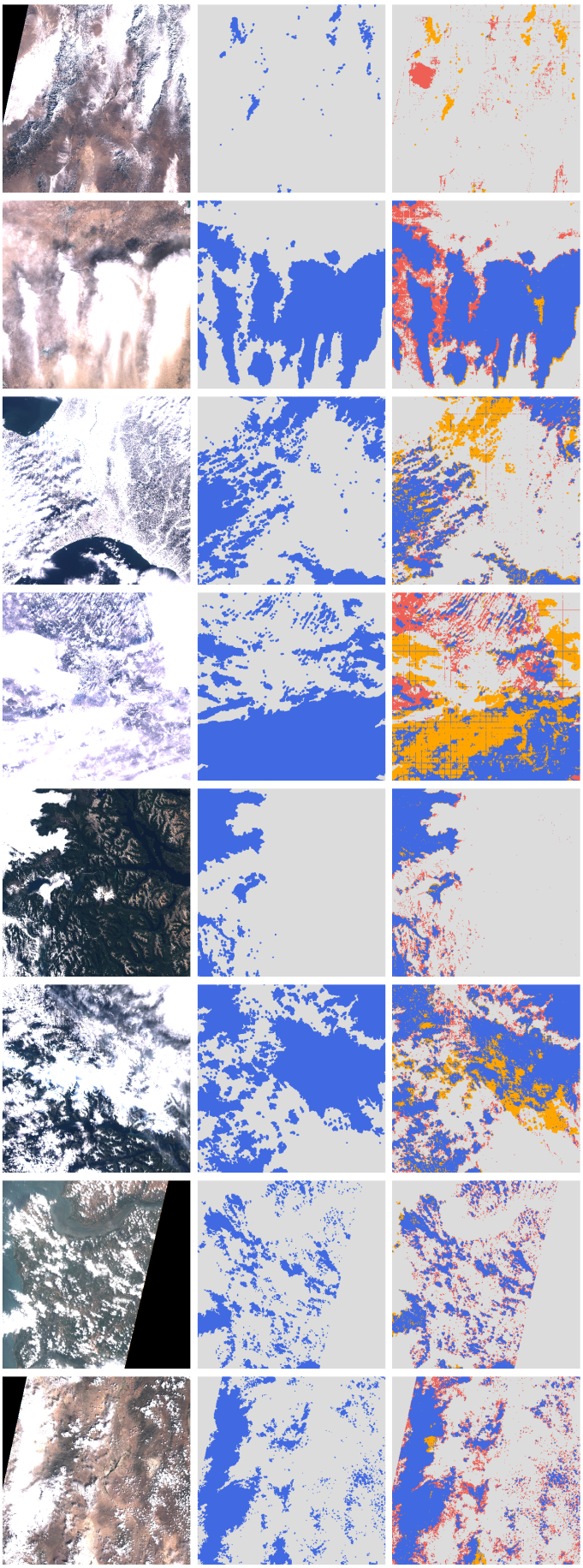


Imagen
Máscara
SatCloud CMIX





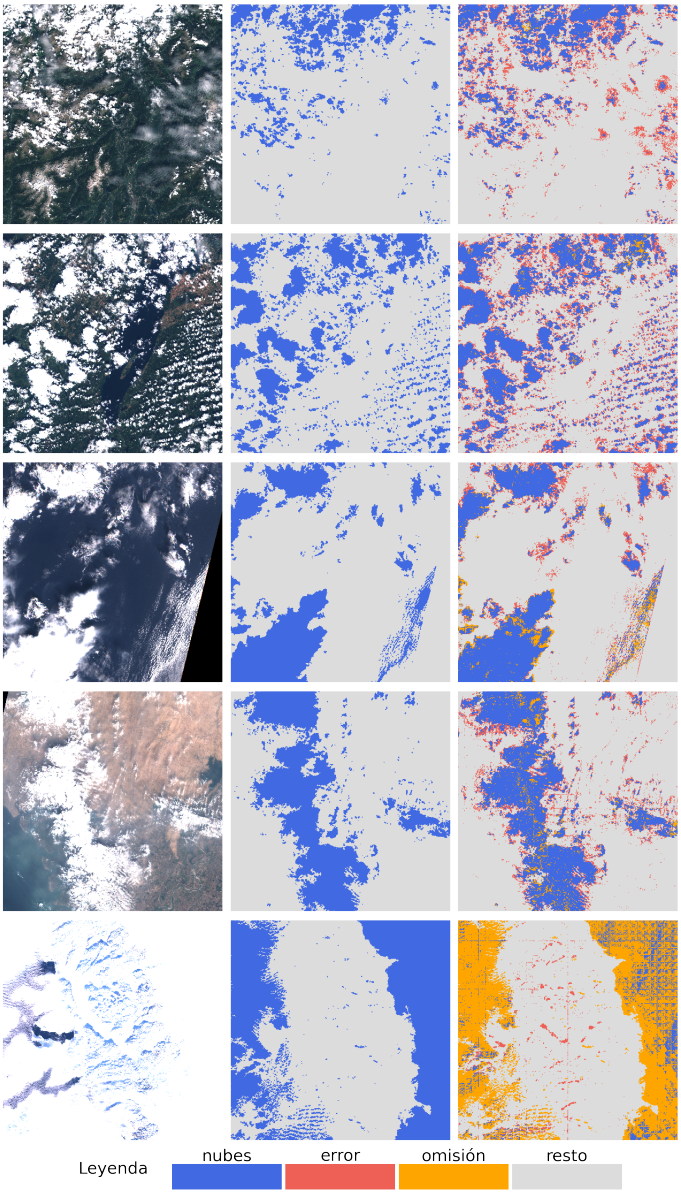


Figura 5.8: Comparativa visual del modelo SatCloud propuesto con la base de datos PixBox S2.

Capítulo 6

Implementación en MPSoC del modelo de detección de nubes

En este capítulo, se lleva a cabo la implementación de un procesador para la inferencia del modelo SatCloud en un MPSoC. Para ello, se hace uso de las herramientas de alto nivel (Vitos AI) que proporciona Xilinx para sus dispositivos con el propósito de realizar la implementación en sus MPSoC a través de las DPUs (unidades de procesamiento profundo). El objetivo de este capítulo es evaluar la implementabilidad del modelo SatCloud en un dispositivo MPSoC, el cual dispone las cámaras iSIM de Satlantis, y examinar los tiempos de ejecución de inferencia para determinar si es posible su ejecución en tiempo real para la aplicación requerida.

6.1. Integración de los modelos de aprendizaje profundo en un sistema embebido

Uno de los retos más importes a la hora de diseñar un modelo de aprendizaje profundo es conocer la plataforma de ejecución en la que se implemente la inferencia del modelo. En la actualidad, debido a la complejidad que supone su incorporación, muy pocos satélites llevan integrado un sistema de aprendizaje profundo en tiempo real. En [3], puede verse la primera implementación conocida de una red neuronal profunda a bordo de un satélite para la observación de la tierra. Todo el procesamiento y control de la EoT (eyes of things, aplicaciones de visión) lo realiza la VPU Intel Movidius Myriad 2 capaz de realizar inferencias rápidas manteniendo el consumo de energía muy por debajo de los 2 W. Uno de los puntos más fuertes de esta implementación es que separa la tarea de detección de nubes del resto de sistemas de cómputo que lleva, como es el MPSoC (Figura 6.1). De esta manera, puede separar del diseño global esta tarea, reduciendo la complejidad del sistema final.

Debido a los requerimientos de este proyecto, se ha optado por utilizar las DPUs, ya que soporta una integración directa de los modelos implementados con Tensorflow en Python y, además, se puede llevar a cabo en los propios dispositivos electrónicos que ya incluyen las cámaras de Satlantis. Este hecho es importante debido a que el satélite de Satlantis que incorpora las cámaras iSIM fue lanzado el pasado mes de mayo. Por lo que la inclusión de estas nuevas características deberá ser realizada en órbita. En próximas actualizaciones de las cámaras iSIM, sería conveniente valorar si la implementación basada en VPUs puede ser una mejor opción.

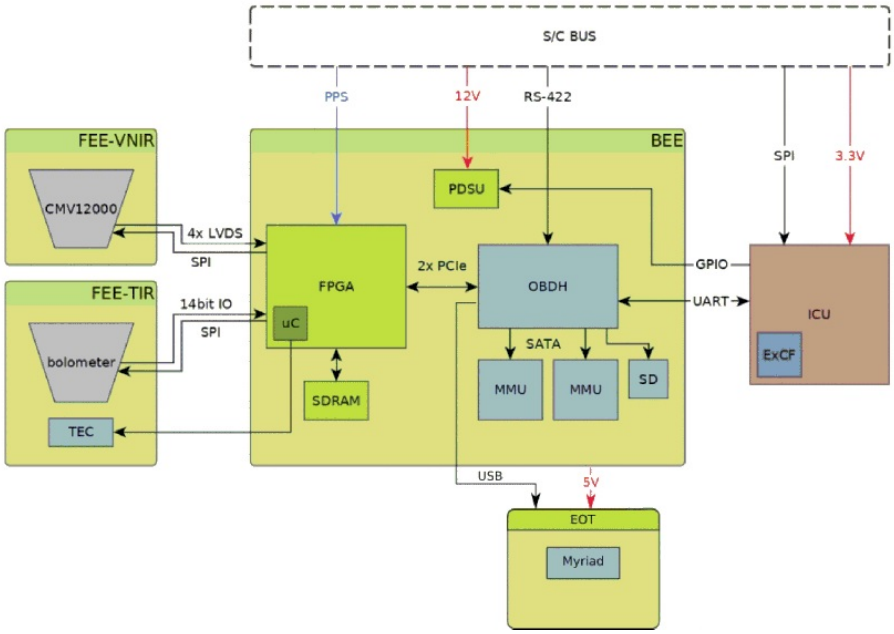


Figura 6.1: Diagrama de bloques de la arquitectura del HyperScout 2 [3].

Para realizar las primeras pruebas, se utiliza el kit de evaluación Xilinx Zynq UltraScale MPSoC ZCU104 disponible en el laboratorio del Grupo de Electrónica Digital (GDED). Con ella, se comprobará la viabilidad del modelo propuesto, se estudiarán los recursos lógicos y de memoria necesarios, y se calcularán los tiempos de ejecución de la inferencia. En una fase más avanzada del proyecto de investigación, se integrará en el diseño de Satlantis utilizando sus MPSoCs.

6.2. Unidad de procesamiento de aprendizaje profundo (DPU)

La unidad de procesamiento de aprendizaje profundo (DPU) es un motor de cálculo configurable y optimizado para redes neuronales convolucionales. Su implementación se realiza en el diagrama de bloques de Vivado mediante un IP llamado DPUCZDX8G diseñado para las Zynq UltraScale+ MPSoC. El grado de paralelismo utilizado en el motor es un parámetro de diseño y puede seleccionarse según el dispositivo y la aplicación de destino. A alto nivel, la DPU es un motor de cálculo microcodificado que cuenta con un conjunto de instrucciones eficiente y optimizado, y que puede soportar la inferencia de la mayoría de las redes neuronales convolucionales.

El IP DPUCZDX8G se implementa en la lógica programable (PL) del dispositivo Zynq UltraScale+ MPSoC seleccionado mediante conexiones directas al sistema de procesamiento (PS). El DPUCZDX8G ejecuta microcódigo compilado generado a partir de un gráfico de red neuronal, y requiere ubicaciones de memoria accesibles para las imágenes de entrada, así como datos temporales y de salida. También se necesita un programa que se ejecute en la unidad de procesamiento de aplicaciones (APU) para atender las interrupciones y coordinar las transferencias de datos. El diagrama de bloques de nivel superior del DPUCZDX8G se muestra en la Figura 6.2.

Se pueden utilizar múltiples núcleos de DPUCZDX8G para lograr un mayor rendimiento, a costa de un mayor uso de recursos de la lógica programable. Este parámetro es configurable, siendo 4 el número máximo que puede tener. Cada núcleo genera una interrupción

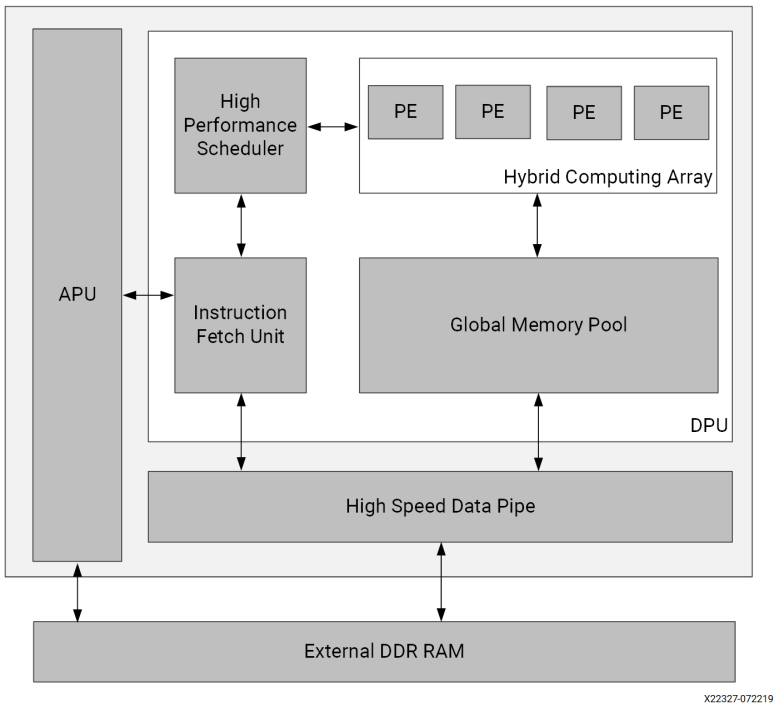


Figura 6.2: Diagrama de bloques del IP DPUCZDX8G [118].

para señalar la finalización de una tarea, pudiendo ser conectadas directamente al PS.

6.2.1. Ejemplo de integración en MPSoC

En la propia documentación de Xilinx del IP, se encuentra un ejemplo de aplicación de las DPUs [118]. Este ejemplo es muy interesante, ya que ilustra un sistema de adquisición de imágenes capturados por un dispositivo Zynq UltraScale+ MPSoC con una cámara conectada. El DPUCZDX8G se integra en el sistema a través de una interconexión AXI para realizar tareas de inferencia de aprendizaje profundo como son la clasificación de imágenes, la detección de objetos, la segmentación semántica, o como en el caso de este proyecto, para la detección de nubes. Con este diagrama se puede comprobar que es posible una adaptación directa con la arquitectura de Satlantis. Al igual que en la Figura 6.3, Satlantis lleva a cabo la captura de las imágenes mediante sus cámaras iSIM y realiza una serie de operaciones en la lógica programable (PL) hasta que los envía al sistema de procesamiento (PS) mediante DMAs. Además, dispone de interconexiones AXI que van directamente al PS, siendo posible conectar en ellos las DPUs, tal y como se muestra en la figura.

6.2.2. Arquitectura de las DPUs

El IP de DPUs puede ser configurado entre varias arquitecturas de convolución, de tal manera que se pueda escoger la complejidad de la de la implementación. La elección de la arquitectura está relacionada con el paralelismo de la unidad de convolución, siendo estas tres las dimensiones que se pueden configurar:

- **Paralelismo de píxeles (PP):** Indica cuantos píxeles se generan en paralelo.

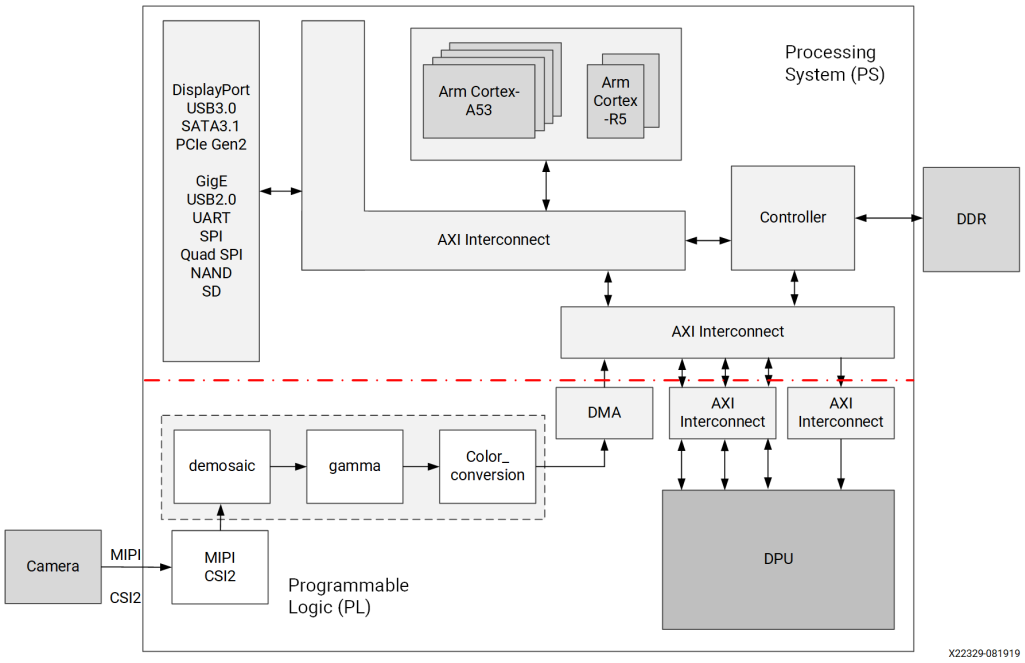


Figura 6.3: Ejemplo de sistema con DPUCZDX8G integrado [118].

Tabla 6.1: Diferentes arquitecturas de convolución dependiendo del paralelismo [118].

DPU Architecture	Pixel Parallelism (PP)	Input Channel Parallelism (ICP)	Output Channel Parallelism (OCP)	Peak Ops (operations/per clock)
B512	4	8	8	512
B800	4	10	10	800
B1024	8	8	8	1024
B1152	4	12	12	1150
B1600	8	10	10	1600
B2304	8	12	12	2304
B3136	8	14	14	3136
B4096	8	16	16	4096

- **Paralelismo del canal de entrada (ICP):** Indica cuántos canales de entrada se calculan en un ciclo.
- **Paralelismo del canal de salida (OCP):** Indica cuántos canales de salida se generan en un ciclo, es decir, cuántos núcleos se utilizan en un ciclo.

En la Figura 6.4 puede verse de manera gráfica la implicación de cada parámetro. El paralelismo del canal de entrada es siempre igual al paralelismo del canal de salida. Las diferentes arquitecturas requieren diferentes recursos lógicos programables. Las arquitecturas más grandes pueden lograr un mayor rendimiento utilizando más recursos. En la Tabla 6.1 pueden verse las diferentes arquitecturas dependiendo del nivel de paralelismo. En cada ciclo de reloj, la matriz de convolución realiza una multiplicación y una acumulación, que se cuentan como dos operaciones. Es por ello que el número máximo de operaciones por ciclo es igual a la multiplicación de los 3 parámetros de paralelismo de convolución multiplicado por 2, es decir, $PP \times ICP \times OCP \times 2$.

La utilización de recursos de lógica programable por parte de las DPUs viene determinada por la arquitectura de convolución que se elija y por algunos extras que se pueden

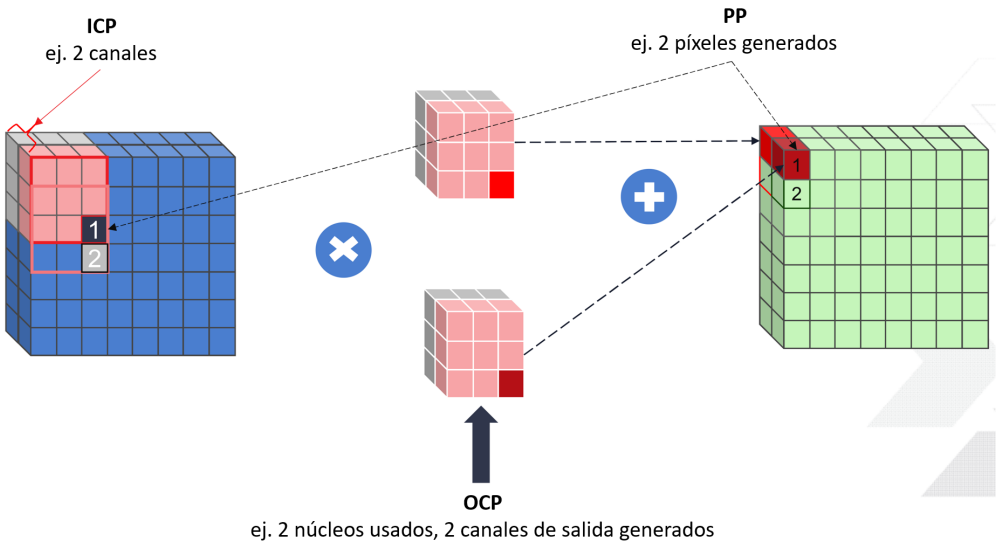


Figura 6.4: Arquitectura de la DPU en relación al paralelismo de convolución

Tabla 6.2: Recursos lógicos dependiendo de la arquitectura de la DPU [118].

DPU Architecture	LUT	Register	Block RAM	DSP
B512 (4x8x8)	27893	35435	73.5	78
B800 (4x10x10)	30468	42773	91.5	117
B1024 (8x8x8)	34471	50763	105.5	154
B1152 (4x12x12)	33238	49040	123	164
B1600 (8x10x10)	38716	63033	127.5	232
B2304 (8x12x12)	42842	73326	167	326
B3136 (8x14x14)	47667	85778	210	436
B4096 (8x16x16)	53540	105008	257	562

añadir, como pude ser, el cálculo de la capa Softmax en la propia DPU. Utilizando un solo núcleo de DPU y basándose en la plataforma ZCU102, equivalente en la utilización de recursos a la ZCU104, en la Tabla 6.2 se observan el número de recursos utilizados. Dichos valores pueden tomarse como una aproximación dada por Xilinx sobre el número de recursos utilizados, pudiendo variar en la implementación real.

6.2.3. Conexión del IP DPUCZDX8G en el diagrama de bloques del MPSoC

El IP DPUCZDX8G sólo contiene una interfaz esclava (S_AXI). Cada núcleo del IP tiene tres interfaces maestras, una para la obtención de instrucciones (DPUX_M_AXI_INSTR) y las otras dos para el acceso a los datos (DPUX_M_AXI_DATA0 y DPUX_M_AXI_DATA1). El IP DPUCZDX8G puede conectarse al sistema de procesamiento (PS) con un IP de interconexión AXI siempre que la DPUCZDX8G pueda acceder correctamente al espacio de memoria DDR. Generalmente, cuando los datos se transfieren a través de un IP de interconexión, la latencia de la transacción de datos aumentará, reduciendo el rendimiento del DPUCZDX8G. Por lo tanto, Xilinx recomienda que cada interfaz maestra en la DPUCZDX8G se conecte al PS a través de una conexión directa en lugar de utilizar un IP de interconexión AXI, siempre que haya suficientes puertos esclavos AXI disponibles en el PS. Teniendo en cuenta la arquitectura de Satlantis, se deberá conectar a través de un IP de interconexión AXI, ya que no hay suficientes disponibles para que se puedan conectar directamente.

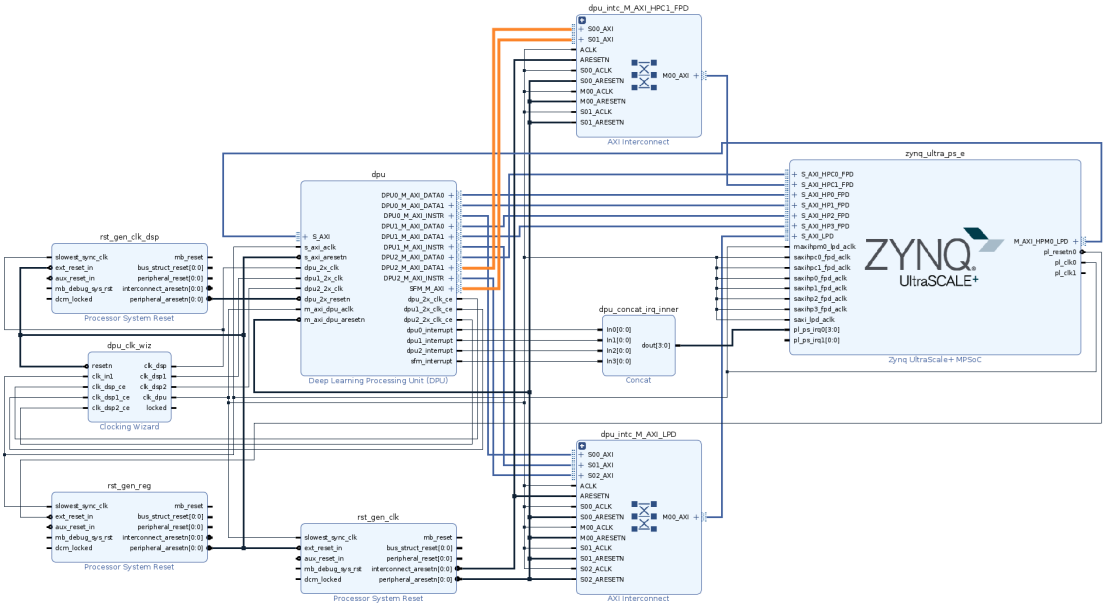


Figura 6.5: Conexión entre el IP DPUCZDX8G y el PS para la integración en MPSoC [118].

El la Figura 6.5 se muestra una conexión de referencia entre el DPUCZDX8G y el PS en el Zynq UltraScale+ MPSoC. El número de núcleos del IP DPUCZDX8G se establece en 3.

6.3. Vitis AI

Para generar un primer prototipo del modelo SatCloud sobre una MPSoC de Xilinx utilizando DPUs se ha utilizado la herramienta de desarrollo de alto nivel que proporciona Xilinx, denominada *Vitis AI*. Dicha herramienta es una completa plataforma de desarrollo de inferencia de IA en dispositivos, placas y tarjetas de aceleración para dispositivos de Xilinx. Consta de un gran conjunto de modelos de IA, núcleos IP de DPUs optimizados, herramientas, bibliotecas y diseños de ejemplo para la IA. Está diseñado teniendo en cuenta la alta eficiencia y la facilidad de uso, liberando todo el potencial de la aceleración de la IA en las FPGAs de Xilinx y los SoCs adaptativos. Admite los principales marcos y los últimos modelos capaces de realizar diversas tareas de aprendizaje profundo, como son las CNN, RNN y NLP.

La herramienta *Vitis AI* viene incluida en un contenedor *docker*, de manera que las distintas fases del proceso de desarrollo vienen protegidas para que no pueda conocerse cómo se realiza la implementación. El primer paso de la herramienta es activar el entorno de *conda* con el workflow que se esté utilizando. Actualmente, *Vitis AI* soporta Tensorflow 1.x y 2.x, Caffe, Neptune y PyTorch. Para este proyecto se ha utilizado el workflow de Tensorflow 2, ya que el modelo propuesto (SatCloud) se ha entrenado del mismo modo.

Actualmente, *Vitis AI* soporta todas las capas que componen el modelo SatCloud, pero deben definirse consecutivamente, es decir, no soporta capas personalizadas ni unión de capas a través de *tf.keras.Sequential*. Además, es conveniente guardar los modelos con la extensión *.h5*, es decir, en formato binario HDF5, ya que *Vitis AI* interpreta mejor los modelos con el formato de guardado original que utilizaban los modelos con keras antes de formar parte de la librería de tensorflow. Las diferentes partes que forman el proceso de *Vitis AI* se ejecutan en Python, debido a que la herramienta está incluida como librerías de Python, aunque como se ha explicado antes, deben ser ejecutas en el contenedor de *docker*

y en el entorno de *conda* correspondiente.

6.3.1. Vitis AI Quantizer

La familia de aceleradores de ML de Xilinx DPU ejecuta modelos y redes que tienen sus parámetros en formato entero, por lo que se debe convertir el modelo entrenado de punto flotante, en un modelo entero de punto fijo. El modelo de aprendizaje profundo de punto fijo requiere menos ancho de banda de memoria, por lo que proporciona una mayor velocidad y una mayor eficiencia energética que el modelo de punto flotante. Este proceso se conoce como cuantificación.

Para llevar a cabo la cuantificación, se debe seleccionar un número de batch size y de steps, de la misma forma en la que se definen durante el entrenamiento. La multiplicación de ambos parámetros tiene como resultado el número de imágenes que se van a utilizar para la calibración. El conjunto de imágenes no tiene por qué estar etiquetado, ya que no se lleva a cabo ningún proceso de optimización de parámetros del modelo.

El cuantificador puede reducir la complejidad del cálculo sin perder prácticamente nada la precisión a la hora de calcular la predicción, convirtiendo los pesos y las activaciones de punto flotante a punto fijo, como puede ser en *int8*. Una vez realizada la cuantificación, es muy recomendable comprobar el antes y el después de las predicciones que realiza modelo para conocer cuanta precisión ha perdido.

6.3.2. Vitis AI Compiler

El compilador de IA de Xilinx, asigna al modelo entrenado y cuantizado, un conjunto de instrucciones y un flujo de datos de alta eficiencia. También realiza sofisticadas optimizaciones, como la fusión de capas y la programación de instrucciones, y reutiliza la memoria del chip en la medida de lo posible. El proceso de compilación va asociado a una placa, por lo que se debe especificar la plataforma de ejecución, en este caso, la ZCU104.

En el registro de los informes del compilador, se puede ver una línea que indica cómo se ha dividido el modelo compilado en subgráficos:

[UNILOG][INFO] Total device subgraph number 3, DPU subgraph number 1

Esta línea indica que el modelo compilado contiene un total de 3 subgráficos, de los cuales uno es un subgráfico de DPU (y por tanto se ejecutan en el acelerador de DPU) y los otros dos subgráficos son de CPU o subgráficos de usuario. Para conocer cómo se conectan los diferentes subgráficos entre sí, se genera una imagen de los subgráficos (Figura 6.6). Las cajas con contornos verdes significan subgráficos de usuario y son los que aparecen a la entrada del modelo. Los que son de contorno azul indican que se ejecutan en las DPUs y los de contorno rojo los que se ejecutan en CPU. Solamente se ejecuta en CPU el último bloque, cuya función es convertir el resultado que se calcula en las DPUs con punto fijo a punto flotante.

6.3.3. Aplicación de Python para ejecutar la inferencia del modelo

Para poder realizar el test y validación del procesador implementado, se debe crear una aplicación en Python que permita la ejecución posterior en el MPSoC. Esta aplicación debe ser capaz de leer imágenes de test, mandarlas ejecutar a las DPUs y guardar el resultado

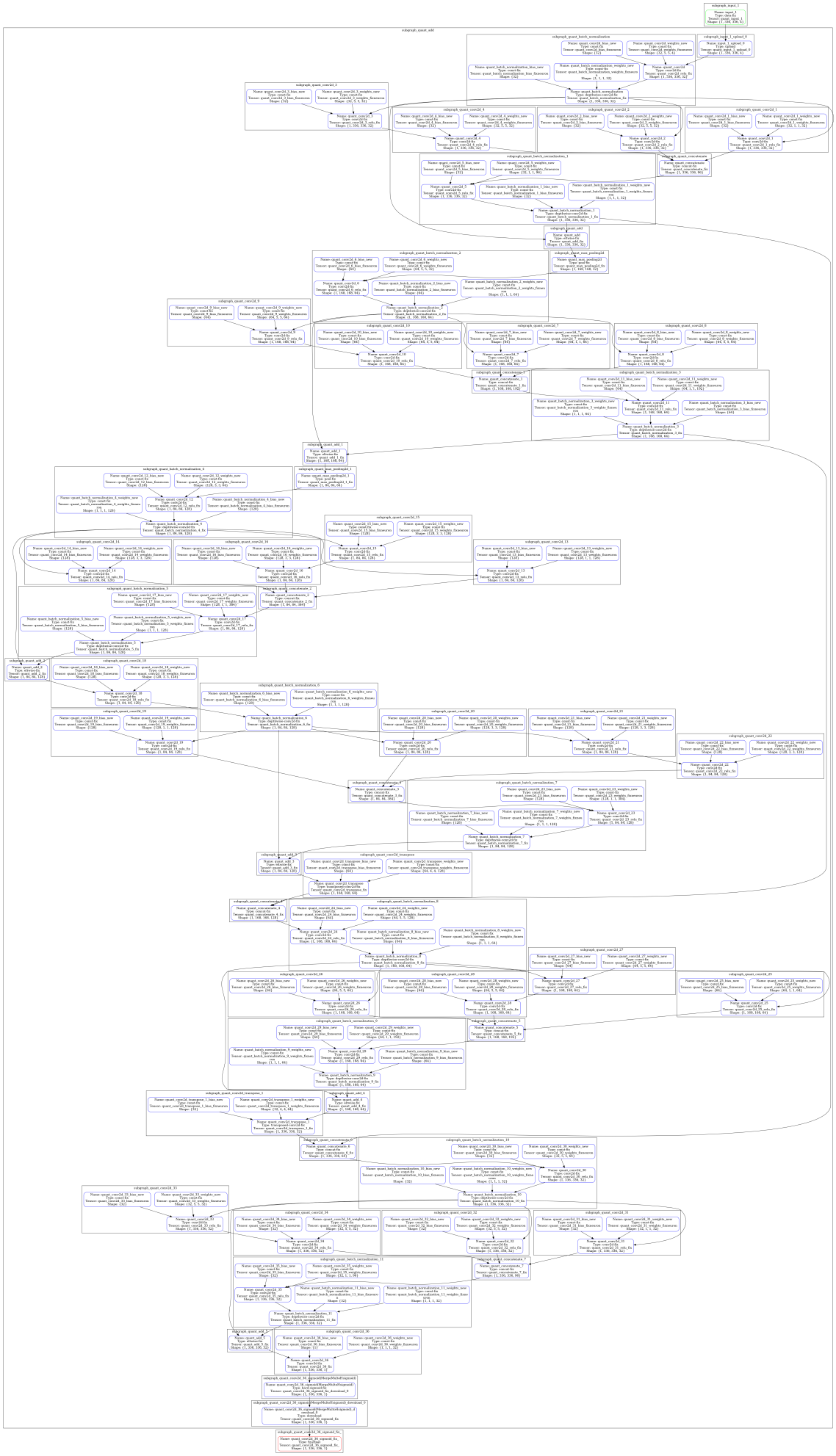


Figura 6.6: Imagen de los subgráficos generados por *Vitis AI Compiler*.

de la máscara de nubes. Además, debe ser capaz de calcular el tiempo de ejecución de la inferencia en el propio MPSoC.

Al ser un script de Python el que se ejecuta en el propio MPSoC, la lectura y escritura de las imágenes se realiza del mismo modo en el que se hacía cuando se realizaba la inferencia en la CPU o en la GPU. Lo que cambia es la manera en la que se ejecuta en las DPUs. Lo primero que se debe hacer es crear un runner de DPU por cada hilo. El usuario puede seleccionar el número de hilos de CPU con los que desea que se ejecute este proceso, hasta un máximo de 8 que es lo que permite el kit de evaluación Xilinx Zynq UltraScale MPSoC ZCU104, es decir, al disponer 4 núcleos de CPU en multihilo, suman un total de 8 hilos disponibles. Para crear el runner se utiliza la librería VART (Vitis AI RunTime) de la siguiente forma:

```
import vart

all_dpu_runners = []
    for i in range(threads):
        all_dpu_runners.append(vart.Runner.create_runner(subgraphs[0], 'run'))
```

Una vez inicializado el runner, la inferencia de cada imagen se ejecuta en la DPU de la siguiente forma:

```
job_id = dpu.execute_async(inputData,outputData)

dpu.wait(job_id)
```

A la hora de automatizar el proceso y que se puedan calcular para un número *n* de imágenes, es necesario añadir más comandos relacionados con la gestión de hilos en Python, pero los principales a la hora de comprender su funcionamiento general son los vistos hasta ahora. Para llevar al target, en este caso el MPSoC, todo lo necesario para que se pueda ejecutar la inferencia del modelo, se crea un script de Python que se encarga de copiar en un fichero las imágenes de test, el modelo compilado y la aplicación de Python.

6.3.4. Ejecución en el MPSoC

Para realizar la ejecución en el MPSoC, en necesario crear una imagen en la SD que contenga un bitstream que incluya los IPs de DPU y un Linux apropiado. En la página oficial de Xilinx hay disponibles imágenes SD para el kit de evaluación Xilinx Zynq UltraScale MPSoC ZCU104, por lo que, para estas pruebas, es suficiente con flashear directamente la SD con esta imagen. El bitstream que viene creado por defecto es el visto en el apartado 6.2 cuando se ha analizado el diseño de las DPUs, a excepción del número de núcleos, que en este caso son 2. El sistema operativo embebido que lleva incorporado es un *PetaLinux* con los parámetros que vienen por defecto.

Una vez flasheada la SD, se introduce en el MPSoC y se arranca desde este dispositivo. Se introduce en el MPSoC el fichero generado en el apartado 6.3.3, donde van incluidas las imágenes de test, el modelo compilado y la aplicación de Python. Para ejecutar la aplicación, se introduce una librería llamada *vaitrace* que se encarga de realizar un seguimiento de las diferentes partes del proceso de inferencia. De esta manera, se guardan una serie de ficheros que pueden ser analizados posteriormente mediante la herramienta de Xilinx llamada Vitis Analyzer. La aplicación se ejecuta de la siguiente forma:

```
python3 -m vaitrace_py ./app_mt.py
```

Tabla 6.3: Tiempos de inferencia en el MPSoC.

DPU Summary

Kernel	Compute Unit	Runs	Min Time (ms)	Avg Time (ms)	Max Time (ms)
subgraph_quant_add	DPUCZDX8G_1:batch-1	48	126.411	128.439	132.792
subgraph_quant_add	DPUCZDX8G_2:batch-1	52	128.126	133.688	141.405
vart::TensorBuffer::copy_tensor_buffer	CPU	200	1.654	5.220	14.975
xir::XrtCu::run	CPU	100	126.393	131.144	141.366

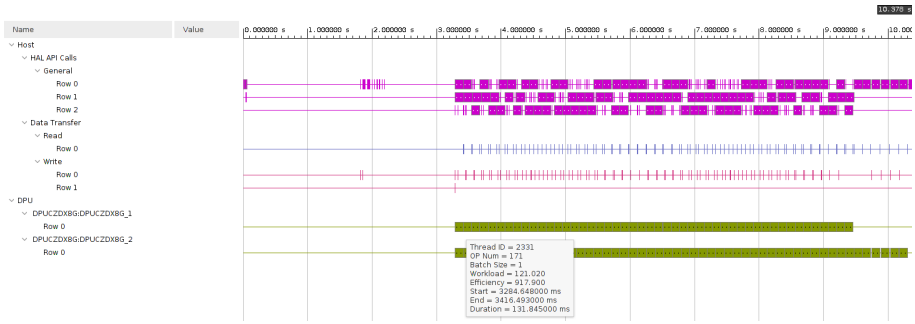


Figura 6.7: Tiempos de inferencia detallados en el MPSoC.

siendo *app_mt.py* la aplicación de Python para la ejecución de la inferencia del modelo.

Para esta prueba se han utilizado 100 imágenes de test, obteniendo los resultados que pueden verse en la propia terminal.

Throughput=14.14 fps, total frames = 100, time=7.0701 seconds

A partir de estos resultados, se puede confirmar que la implementación de la inferencia de los modelos para la detección de nubes utilizando las DPUs en viable. Se han utilizado los 8 hilos del procesador, aunque se ha comprobado que a partir del cuarto hilo apenas mejoran los tiempos de inferencia. Cabe destacar que este throughput es para parches de 336 x 336, siendo necesario 4 parches para completar una imagen proveniente de las cámaras iSIM de Satlantis (apartado 5.1.7). Por lo tanto, el throughput para una imagen completa sería de 3,535 fps.

6.3.5. Vitis Analyzer

El *Vitis Analyzer* es una herramienta de Xilinx que permite ver y analizar los informes generados mientras se construye y ejecuta la aplicación en el MPSoC. Para ello, se deben pasar los informes generados en el MPSoC a un PC en el que esté disponible la herramienta. En la Tabla 6.3 vienen de forma resumida descritos los tiempos de inferencia en el MPSoC. Como se puede ver, se ha realizado la inferencia a 100 imágenes de test, de las cuales 48 se han ejecutado en una DPU y 52 en la otra. El tiempo medio entre las dos DPUs ha sido de 131,06 ms, siendo 141,41 ms el máximo tiempo que ha tardado. El tiempo en CPU no se tiene en cuenta, ya que en la CPU no se realiza la inferencia, solamente se acomodan los datos para introducirlos en las DPUs y para recoger los resultados. En la Figura 6.7 puede verse con mayor detalle la cronología seguida por la aplicación.

Capítulo 7

Conclusiones

En este trabajo se han estudiado los métodos más avanzados para la implementación en tiempo real de algoritmos de detección de nubes para cámaras multiespectrales de alta resolución. El objetivo ha sido conocer los métodos más actuales y proponer un método preciso y robusto capaz de proporcionar máscaras fiables de presencia de formaciones nubosas que además fuera implementable en un dispositivo de tipo MPSoC y que proporcionara un troughput o rendimineto temporal acorde con las necesidades de la plataforma iSIM de Satlantis.

La selección del algoritmo de detección de nubes más óptimo se ha divido principalmente en dos partes. Por un lado, se ha realizado un estudio de las características espectrales para saber el grado de separabilidad entre bandas. Para ello, se ha seleccionado la base de datos de Sentinel-2 llamada *WHUS2-CD+* que contiene los principales tipos de cobertura terrestre y cuyas imágenes fueron tomadas entre abril de 2018 y mayo de 2020, cubriendo las diferentes estaciones del año. Las máscaras de nubes de referencia de la base de datos están etiquetadas manualmente a una resolución espacial de 10m. Para realizar el análisis de separabilidad, se ha etiquetado manualmente la nieve para comprobar que realmente con solo las bandas VNIR es muy difícil distinguir las nubes de terrenos de alto albedo, como es la nieve. Los resultados, basándose en la distancia Jeffreys-Matusita (JM), muestran que utilizando las bandas SWIR se logra una buena separabilidad entre las tres clases, nubes, nieve y resto. En cambio, utilizando únicamente las bandas VNIR, tan solo se logra una separabilidad moderada entre las clases. Realizando el histograma de las reflectancias, se corrobora el gran solapamiento entre las clases de nieve y nubes, a excepción de las bandas SWIR. Por otro lado, se ha buscado la implementación de los algoritmos, que a su vez se ha divido en dos clases principales, los métodos más clásicos basados en umbrales derivados de observaciones físicas o empíricas y los basados en técnicas de ML.

Los algoritmos de detección de nubes clásicos están basados en umbrales píxel a píxel, sin tener en cuenta las características espaciales. Se ha analizado el sistema de detección de nubes de Sentinel-2 basado en umbrales para conocer la metodología utilizada, a pesar de utilizar las bandas SWIR. De esta manera, se demuestra que efectivamente la banda SWIR es la base de los algoritmos de detección de nubes gracias a su separabilidad espectral. Para conocer numéricamente y saber si es posible la utilización de algoritmos basados en umbrales, se realiza la detección de nubes utilizando los árboles de decisión. Los resultados muestras que utilizando solo las bandas VNIR no es posible detectar correctamente las nubes, especialmente en terrenos con nieve. La métrica IoU, una de las más representativas de la robustez del algoritmo, obtiene una media de 0,3654, disminuyendo hasta un 0,0322 en una imagen con presencia de nieve. En cambio, haciendo uso de las bandas SWIR, se logra

un IoU medio de 0,5212. De este análisis, se puede concluir que, para el propósito de este proyecto, realizar una clasificación basada en umbrales utilizando solo las bandas VNIR no es adecuado.

En la actualidad, los algoritmos basados en aprendizaje profundo son los más sofisticados de la literatura científica. La U-Net es la arquitectura en la que se basan la mayoría de los métodos de detección de nubes, gracias a su capacidad de adaptación y su flexibilidad a la hora de extraer características tanto espectrales como espaciales. El mayor problema de este tipo de arquitecturas es que el aprendizaje puede ralentizarse en las capas intermedias de los modelos más profundos, por lo que existe cierto riesgo de que la red aprenda a ignorar las capas en las que se representan las características abstractas. En la práctica, se ha visto que la U-Net clásica en la detección de nubes tiene un rendimiento moderado, sobre todo en terrenos con nieve donde no es capaz de distinguirlo de las nubes. Además, aparece el efecto que producen las convoluciones en los bordes de la imagen. Aumentando el tamaño de los filtros se logra que desaparezca este efecto, pero la complejidad del algoritmo aumenta de manera significativa sin que su desempeño en terrenos con nieve sea suficiente. Por todo ello, en este proyecto se ha propuesto un método basado en las U-Nets que puede capturar y agregar características a varias escalas (multiescala), garantizando que las características semánticas de alto nivel extraídas de los terrenos de alto albedo, como la nieve y las nubes, sean más distintivas. Gracias a una serie de opciones de diseño y técnicas de entrenamiento que mejoran el rendimiento, el modelo propuesto, que hemos denominado SatCloud, muestra un rendimiento de vanguardia, comparable al de otros métodos presentados en la literatura científica, una gran velocidad y una gran solidez ante todo tipo de situaciones. Con el objetivo de analizar si es posible simplificar el modelo y hacerlo computacionalmente más ligero sin comprometer en exceso la precisión, se propone una optimización de algunos hiperparámetros de la red. De este análisis, se concluye que el modelo con parches de 336 x 336, filtros de 5x5 y profundidad de 2 es la mejor opción para la implementación final. Presenta el mayor valor en las métricas que destacan por castigar más la detección de falsos positivos, es decir, las métricas F1-score, IoU, MIoU y kappa. Respecto a la exactitud y precisión de modelo, no es el que presenta el valor más alto, pero sí que destaca entre los mejores. Realizando el entrenamiento completo, se obtienen unos resultados muy buenos, siendo el IoU medio de 0,6368. Además, la exactitud global es de 97,32 % con una precisión de 91,76 %. El desempeño en terrenos de alto albedo también es satisfactorio, siendo de 89,61 la exactitud global más baja de todas las imágenes de test. Además, se ha comparado con otras bases de datos, mostrando que el modelo no está sesgado a la base de datos que se ha utilizado como referencia.

Por último, se lleva a cabo la implementación de la inferencia en un MPSoC utilizando el modelo multiescala propuesto. Mediante la herramienta de Xilinx llamada *Vitis AI*, se realiza el desarrollo a través de las unidades de procesamiento profundo (DPUs). Dicha herramienta, permite realizar la cuantización y la compilación a partir de un modelo generado en Python con la librería Tensorflow. La experimentación se realiza en el kit de evaluación Xilinx Zynq UltraScale MPSoC ZCU104, de las mismas características al utilizado por las cámaras iSIM de Satlantis, utilizando dos DPUs. Se ha podido demostrar la viabilidad de implementar la inferencia de los modelos para la detección de nubes en tiempo real utilizando las DPUs, logrando un tiempo medio entre las dos DPUs de 131,06 ms, siendo 141,41 ms el máximo tiempo que ha tardado. Además, teniendo en cuenta la disposición del diseño de la arquitectura por parte de Satlantis, la implementación de la inferencia a través de DPUs es posible mediante AXI Interconnect.

Como líneas de investigación futuras, se podría buscar alguna otra modificación de la arquitectura U-Net que permita aumentar el rendimiento de la red o disminuir su complejidad. También se podría buscar una alternativa a las redes tipo encoder-decoder. Actual-

mente, la redes generativas adversarias (GANs) son los modelos que más proyección futura están demostrando, aunque todavía las investigaciones no están en una fase de implementación comercial debido a su complejidad en el desarrollo y a nivel de cómputo. Respecto a la implementación del algoritmo, se podría estudiar el uso de VPUs para realizar la inferencia a bordo del satélite. De esta manera, se lograría separar la tarea de detección de nubes del resto de sistemas de cómputo que se realizan en el MPSoC.

Bibliografía

[1] H. Ishida, Y. Oishi, K. Morita, K. Moriwaki, and T. Y. Nakajima, "Development of a support vector machine based cloud detection method for modis with the adjustability to various conditions," *Remote sensing of environment*, vol. 205, pp. 390–407, 2018.

[2] J. Li, Z. Wu, Z. Hu, C. Jian, S. Luo, L. Mou, X. X. Zhu, and M. Molinier, "A lightweight deep learning-based cloud detection method for sentinel-2a imagery fusing multiscale spectral and spatial features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–19, 2022.

[3] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ-sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[4] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827021000670

[5] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021.

[6] Y. Zhang, C. Peng, L. Peng, Y. Xu, L. Lin, R. Tong, Z. Peng, X. Mao, H. Hu, Y.-W. Chen, and J. Li, "Deeprecs: From recist diameters to precise liver tumor segmentation," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 614–625, 2022.

[7] M. Gazzea, M. Pacevicius, D. O. Dammann, A. Sapronova, T. M. Lunde, and R. Arghandeh, "Automated power lines vegetation monitoring using high-resolution satellite imagery," *IEEE Transactions on Power Delivery*, vol. 37, no. 1, pp. 308–316, 2022.

[8] K. Kamirul, W. Hasbi, P. R. Hakim, and A. H. Syafrudin, "Automatic ship recognition chain on satellite multispectral imagery," *IEEE Access*, vol. 8, pp. 221 918–221 931, 2020.

[9] S. Mahajan and B. Fataniya, "Cloud detection methodologies: variants and development—a review," *Complex & Intelligent Systems*, vol. 6, no. 2, pp. 251–261, dec 2019.

[10] M. Sawant, M. K. Shende, A. E. Feijóo-Lorenzo, and N. D. Bokde, "The state-of-the-art progress in cloud detection, identification, and tracking approaches: A systematic review," *Energies*, vol. 14, no. 23, 2021.

[11] D. López-Puigdollers, G. Mateo-García, and L. Gómez-Chova, "Benchmarking deep learning models for cloud detection in landsat-8 and sentinel-2 images," *Remote Sensing*, vol. 13, no. 5, 2021.

[12] R. W. Saunders and K. T. Kriebel, "An improved method for detecting clear sky and cloudy radiances from avhrr data," *International Journal of Remote Sensing*, vol. 9, no. 1, pp. 123–150, 1988.

[13] S. Ackerman, K. Strabala, P. Menzel, R. Frey, C. Moeller, L. Gumley, B. Baum, C. Schaaf, and G. Riggs, "Discriminating clear-sky from cloud with modis. algorithm theoretical basis document (mod35)," vol. 103, 01 2006.

[14] W. Li, S. Fang, J. Guo *et al.*, "Cloud detection in modis data based on spectrum analysis," *Geomatics and Information Science of Wuhan University*, vol. 30, no. 5, pp. 435–438, 2005.

[15] X. Liu, Y. Wang, H. SHI, Z. Long, and Z. Jiang, "Cloud detection over the southeast china basing on statistical analysis," *Journal of Image and Graphics*, vol. 15, pp. 1783–1789, 2010.

[16] C.-j. Li, L.-y. Liu, and J. Wang, "Automatic detection and removal of thin haze based on own features of landsat image," *Journal of Zhejiang University (Engineering Science)*, vol. 40, no. 1, p. 10, 2006.

[17] Z. Li, H. Shen, H. Li, G. Xia, P. Gamba, and L. Zhang, "Multi-feature combined cloud and cloud shadow detection in gaofen-1 wide field of view imagery," *Remote sensing of environment*, vol. 191, pp. 342–358, 2017.

[18] G. Cappelluti, A. Morea, C. Notarnicola, and F. Posa, "Automatic cloud detection from modis images," in *Remote Sensing of Clouds and the Atmosphere VIII*, vol. 5235. International Society for Optics and Photonics, 2004, pp. 574–585.

[19] K. Kawano and J.-I. Kudoh, "Cloud detection method for noaa avhrr images by using local area parameters," in *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, vol. 5. IEEE, 2001, pp. 2155–2157.

[20] A. Dybbroe, K.-G. Karlsson, and A. Thoss, "Nwcsaf avhrr cloud detection and analysis using dynamic thresholds and radiative transfer modeling. part i: Algorithm description," *Journal of Applied Meteorology and Climatology*, vol. 44, no. 1, pp. 39–54, 2005.

[21] Q. Xiong, Y. Wang, D. Liu, S. Ye, Z. Du, W. Liu, J. Huang, W. Su, D. Zhu, X. Yao, and X. Zhang, "A cloud detection approach based on hybrid multispectral features with dynamic thresholds for gf-1 remote sensing images," *Remote Sensing*, vol. 12, no. 3, 2020.

[22] Z.-g. Liu, Y.-x. Li, and F. Huang, "Cloud detection of modis satellite images based on dynamical cluster," *Remote Sensing Information*, vol. 7, pp. 33–35, 2007.

[23] S. Sun, "A multi-spectral remote sensing imagery cloud detection algorithm based on spectral angle principle," *Microcomput. Its Appl*, vol. 36, pp. 16–18, 2017.

[24] Z. Liu, L. Han, P. Zhou, X. Wang, and T. Wu, "A method for cloud interpretation in zy-3 satellite imagery and its application," *Remote Sens. Inf*, vol. 32, pp. 41–46, 2017.

[25] Z. Jin, L. Zhang, S. Liu, and F. Yi, "Cloud detection and cloud phase retrieval based on bp neural network," *Opt. Optoelectron. Technol*, vol. 14, pp. 74–77, 2016.

[26] H. Fu, Y. Shen, J. Liu, G. He, J. Chen, P. Liu, J. Qian, and J. Li, "Cloud detection for fy meteorology satellite based on ensemble thresholds and random forests approach," *Remote Sensing*, vol. 11, no. 1, 2019.

[27] R. M. Malladi, A. Nizami, M. S. Mahakali, and B. G. Krishna, "Cloud masking technique for high-resolution satellite data: An artificial neural network classifier using spectral & textural context," *Journal of the Indian Society of Remote Sensing*, vol. 47, no. 4, pp. 661–670, 2019.

[28] A. Taravat, F. Del Frate, C. Cornaro, and S. Vergari, "Neural networks and support vector machine algorithms for automatic cloud classification of whole-sky ground-based images," *IEEE Geoscience and remote sensing letters*, vol. 12, no. 3, pp. 666–670, 2014.

[29] P. Li, L. Dong, H. Xiao, and M. Xu, "A cloud image detection method based on svm vector machine," *Neurocomputing*, vol. 169, pp. 34–42, 2015.

[30] M. Segal-Rozenhaimer, A. Li, K. Das, and V. Chirayath, "Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (cnn)," *Remote Sensing of Environment*, vol. 237, p. 111446, 2020.

[31] J. H. Jeppesen, R. H. Jacobsen, F. Inceoglu, and T. S. Toftegaard, "A cloud detection algorithm for satellite imagery based on deep learning," *Remote Sensing of Environment*, vol. 229, pp. 247–259, 2019.

[32] W. Xie, J. Yang, Y. Li, J. Lei, J. Zhong, and J. Li, "Discriminative feature learning constrained unsupervised network for cloud detection in remote sensing imagery," *Remote Sensing*, vol. 12, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/3/456

[33] A. Henderson-Sellers, "De-fogging cloud determination algorithms," *Nature*, vol. 298, no. 5873, pp. 419–420, jul 1982.

[34] W. Rossow, F. Mosher, E. Kinsella, A. Arking, M. Desbois, E. Harrison, P. Minnis, E. Ruprecht, G. Seze, C. Simmer *et al.*, "Isccp cloud algorithm intercomparison," *Journal of Applied Meteorology and Climatology*, vol. 24, no. 9, pp. 877–903, 1985.

[35] R. R. Irish, J. L. Barker, S. N. Goward, and T. Arvidson, "Characterization of the landsat-7 etm+ automated cloud-cover assessment (acca) algorithm," *Photogrammetric engineering & remote sensing*, vol. 72, no. 10, pp. 1179–1188, 2006.

[36] Y. Luo, A. P. Trishchenko, and K. V. Khlopenkov, "Developing clear-sky, cloud and cloud shadow mask for producing clear-sky composites at 250-meter spatial resolution for the seven modis land bands over canada and north america," *Remote Sensing of Environment*, vol. 112, no. 12, pp. 4167–4185, 2008.

[37] L. Oreopoulos, M. J. Wilson, and T. Várnai, "Implementation on landsat data of a simple cloud-mask algorithm developed for modis land bands," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 4, pp. 597–601, 2011.

[38] Z. Zhu and C. E. Woodcock, "Object-based cloud and cloud shadow detection in landsat imagery," *Remote sensing of environment*, vol. 118, pp. 83–94, 2012.

[39] Z. Zhu, S. Wang, and C. E. Woodcock, "Improvement and expansion of the fmask algorithm: Cloud, cloud shadow, and snow detection for landsats 4–7, 8, and sentinel 2 images," *Remote Sensing of Environment*, vol. 159, pp. 269–277, 2015.

[40] *EarthExplorer*, U. S. Geological Survey (USGS) Std. [Online]. Available: https://earthexplorer.usgs.gov/

[41] S. Qiu, B. He, Z. Zhu, Z. Liao, and X. Quan, "Improving fmask cloud and cloud shadow detection in mountainous area for landsats 4–8 images," *Remote Sensing of Environment*, vol. 199, pp. 107–119, 2017.

[42] S. Qiu, Z. Zhu, and B. He, "Fmask 4.0: Improved cloud and cloud shadow detection in landsats 4–8 and sentinel-2 imagery," *Remote Sensing of Environment*, vol. 231, p. 111205, 2019.

[43] D. Candra, S. Phinn, and P. Scarth, "Cloud and cloud shadow masking using multitemporal cloud masking algorithm in tropical environmental." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 41, 2016.

[44] Y. Zhang, B. Guindon, and J. Cihlar, "An image transform to characterize and compensate for spatial variations in thin cloud contamination of landsat images," *Remote Sensing of Environment*, vol. 82, no. 2-3, pp. 173–187, 2002.

[45] D. S. Candra, S. Phinn, and P. Scarth, "Automated cloud and cloud-shadow masking for landsat 8 using multitemporal images in a variety of environments," *Remote Sensing*, vol. 11, no. 17, p. 2060, 2019.

[46] B.-C. Gao and Y. J. Kaufman, "Selection of the 1.375-μm modis channel for remote sensing of cirrus clouds and stratospheric aerosols from space," *Journal of Atmospheric Sciences*, vol. 52, no. 23, pp. 4231–4237, 1995.

[47] A. Hollstein, K. Segl, L. Guanter, M. Brell, and M. Enesco, "Ready-to-use methods for the detection of clouds, cirrus, snow, shadow, water and clear sky pixels in sentinel-2 msi images," *Remote Sensing*, vol. 8, no. 8, p. 666, 2016.

[48] C. C. Henken, M. J. Schmeits, H. Deneke, and R. A. Roebeling, "Using msg-seviri cloud physical properties and weather radar observations for the detection of cb/tcu clouds," *Journal of Applied Meteorology and Climatology*, vol. 50, no. 7, pp. 1587–1600, 2011.

[49] M. J. Hughes and D. J. Hayes, "Automated detection of cloud and cloud shadow in single-date landsat imagery using neural networks and spatial post-processing," *Remote Sensing*, vol. 6, no. 6, pp. 4907–4926, 2014.

[50] S. Skakun, E. F. Vermote, J.-C. Roger, C. O. Justice, and J. G. Masek, "Validation of the lasrc cloud detection algorithm for landsat 8 images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2439–2446, 2019.

[51] S. Foga, P. L. Scaramuzza, S. Guo, Z. Zhu, R. D. Dilley Jr, T. Beckmann, G. L. Schmidt, J. L. Dwyer, M. J. Hughes, and B. Laue, "Cloud detection algorithm comparison and validation for operational landsat data products," *Remote sensing of environment*, vol. 194, pp. 379–390, 2017.

[52] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *international conference on machine learning*. PMLR, 2017, pp. 2847–2854.

[53] Y. Yuan and X. Hu, "Bag-of-words and object-based classification for cloud extraction from satellite imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 8, pp. 4197–4205, Aug 2015.

[54] A. Ahmad and S. Quegan, "Cloud masking for remotely sensed data using spectral and principal components analysis," *Engineering, Technology & Applied Science Research*, vol. 2, no. 3, pp. 221–225, 2012.

[55] J. Wei, W. Huang, Z. Li, L. Sun, X. Zhu, Q. Yuan, L. Liu, and M. Cribb, "Cloud detection for landsat imagery by combining the random forest and superpixels extracted via energy-driven sampling segmentation approaches," *Remote Sensing of Environment*, vol. 248, p. 112005, 2020.

[56] R. Cilli, A. Monaco, N. Amoroso, A. Tateo, S. Tangaro, and R. Bellotti, "Machine learning for cloud detection of globally distributed sentinel-2 images," *Remote Sensing*, vol. 12, no. 15, p. 2355, 2020.

[57] M. Main-Knorn, B. Pflug, J. Louis, V. Debaecker, U. Müller-Wilm, and F. Gascon, "Sen2Cor for Sentinel-2," in *Image and Signal Processing for Remote Sensing XXIII*, L. Bruzzone, Ed., vol. 10427, International Society for Optics and Photonics. SPIE, 2017, pp. 37 – 48.

[58] O. Hagolle, M. Huc, D. V. Pascual, and G. Dedieu, "A multi-temporal method for cloud detection, applied to formosat-2, venμs, landsat and sentinel-2 images," *Remote Sensing of Environment*, vol. 114, no. 8, pp. 1747–1755, 2010.

[59] J. Lee, R. C. Weger, S. K. Sengupta, and R. M. Welch, "A neural network approach to cloud classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 5, pp. 846–855, 1990.

[60] B. Tian, M. A. Shaikh, M. R. Azimi-Sadjadi, T. H. V. Haar, and D. L. Reinke, "A study of cloud classification with neural networks using spectral and textural features," *IEEE transactions on neural networks*, vol. 10, no. 1, pp. 138–151, 1999.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[62] Y. Shendryk, Y. Rist, C. Ticehurst, and P. Thorburn, "Deep learning for multi-modal classification of cloud, shadow and land cover scenes in planetscope and sentinel-2 imagery," *ISPRS Journal of photogrammetry and remote sensing*, vol. 157, pp. 124–136, 2019.

[63] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[64] M. Shi, F. Xie, Y. Zi, and J. Yin, "Cloud detection of remote sensing images by deep learning," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 701–704.

[65] F. Xie, M. Shi, Z. Shi, J. Yin, and D. Zhao, "Multilevel cloud detection in remote sensing images based on deep learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 8, pp. 3631–3640, 2017.

[66] G. Mateo-García, L. Gómez-Chova, and G. Camps-Valls, "Convolutional neural networks for multispectral image cloud masking," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017, pp. 2255–2258.

[67] Y. Zi, F. Xie, and Z. Jiang, "A cloud detection method for landsat 8 images based on pcanet," *Remote Sensing*, vol. 10, no. 6, p. 877, 2018.

[68] W. Wang and Z. Shi, "An all-scale feature fusion network with boundary point prediction for cloud detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[69] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[70] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[71] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017.

[72] S. Mohajerani, T. A. Krammer, and P. Saeedi, "Cloud detection algorithm for remote sensing images using fully convolutional neural networks," *arXiv preprint arXiv:1810.05782*, 2018.

[73] Z. Li, H. Shen, Y. Wei, Q. Cheng, and Q. Yuan, "Cloud detection by fusing multi-scale convolutional features," *Proceedings of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science, Beijing, China*, pp. 7–10, 2018.

[74] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 152, pp. 166–177, 2019.

[75] A. Karpatne, W. Watkins, J. S. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *ArXiv*, 2017.

[76] M. Le Goff, J.-Y. Tourneret, H. Wendt, M. Ortner, and M. Spigai, "Deep learning for cloud detection," in *8th International Conference of Pattern Recognition Systems (ICPRS 2017)*, July 2017, pp. 1–6.

[77] S. Mohajerani and P. Saeedi, "Cloud-net: An end-to-end cloud detection algorithm for landsat 8 imagery," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 1029–1032.

[78] Y. Zhan, J. Wang, J. Shi, G. Cheng, L. Yao, and W. Sun, "Distinguishing cloud and snow in satellite images via deep convolutional network," *IEEE geoscience and remote sensing letters*, vol. 14, no. 10, pp. 1785–1789, 2017.

[79] Z. Yan, M. Yan, H. Sun, K. Fu, J. Hong, J. Sun, Y. Zhang, and X. Sun, "Cloud and cloud shadow detection using multilevel feature fused segmentation network," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 10, pp. 1600–1604, 2018.

[80] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[81] Z. Shao, Y. Pan, C. Diao, and J. Cai, "Cloud detection in remote sensing images based on multiscale features-convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 6, pp. 4062–4076, 2019.

[82] Z. Li, H. Shen, Q. Cheng, Y. Liu, S. You, and Z. He, "Deep learning based cloud detection for medium and high resolution remote sensing images of different sensors," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 197–212, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271619300565

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

[84] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[85] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[86] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[87] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, March 2010.

[88] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2019, pp. 1–9.

[89] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking tpu, gpu, and cpu platforms for deep learning," *arXiv preprint*, 2019.

[90] G. Dinelli, G. Meoni, E. Rapuano, G. Benelli, and L. Fanucci, "An fpga-based hardware accelerator for cnns using on-chip memories only: Design and benchmarking with intel movidius neural compute stick," *International Journal of Reconfigurable Computing*, vol. 2019, 2019.

[91] L. Sterpone, S. Azimi, and B. Du, "A selective mapper for the mitigation of sets on rad-hard rtg4 flash-based fpgas," in *2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, 2016, pp. 1–4.

[92] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[93] S. Rivas-Gomez, A. J. Pena, D. Moloney, E. Laure, and S. Markidis, "Exploring the vision processing unit as co-processor for inference," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 589–598.

[94] G. Furano, G. Meoni, A. Dunne, D. Moloney, V. Ferlet-Cavrois, A. Tavoularis, J. Byrne, L. Buckley, M. Psarakis, K.-O. Voss, and L. Fanucci, "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, 2020.

[95] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "Cloudscout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sensing*, vol. 12, no. 14, 2020.

[96] E. Rapuano, G. Meoni, T. Pacini, G. Dinelli, G. Furano, G. Giuffrida, and L. Fanucci, "An fpga-based hardware accelerator for cnns inference on board satellites: Benchmarking with myriad 2-based solution for the cloudscout case study," *Remote Sensing*, vol. 13, no. 8, 2021.

[97] A. Vala, A. Patel, R. Gosai, J. Chaudharia, H. Mewada, and K. Mahant, "A low-cost and efficient cloud monitoring camera system design for imaging satellites," *International Journal of Remote Sensing*, vol. 40, no. 7, pp. 2739–2758, 2019.

[98] S. Ghassemi and E. Magli, "Convolutional neural networks for on-board cloud screening," *Remote Sensing*, vol. 11, no. 12, 2019.

[99] Z. Zhang, A. Iwasaki, G. Xu, and J. Song, "Cloud detection on small satellites based on lightweight u-net and image compression," *Journal of Applied Remote Sensing*, vol. 13, no. 2, p. 026502, 2019.

[100] W. Li, Z. Zou, and Z. Shi, "Deep matting for cloud detection in remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 12, pp. 8490–8502, Dec 2020.

[101] A. Francis, P. Sidiropoulos, and J.-P. Muller, "Cloudfcn: Accurate and robust cloud detection for satellite imagery with deep learning," *Remote Sensing*, vol. 11, no. 19, 2019.

[102] ESA, "Spatial resolution," *Sentinel-2 User Guides*. [Online]. Available: https://sentinels.copernicus.eu/

[103] K. Wada, "Image polygonal annotation with python," *Python library*. [Online]. Available: https://github.com/wkentaro/labelme

[104] G. Forestier, J. Inglada, C. Wemmert, and P. Gançarski, "Comparison of optical sensors discrimination ability using spectral libraries," *International Journal of Remote Sensing*, vol. 34, no. 7, pp. 2327–2349, 2013. [Online]. Available: https://doi.org/10.1080/01431161.2012.744488

[105] G. Zhang, X. Gao, Y. Yang, M. Wang, and S. Ran, "Controllably deep supervision and multi-scale feature fusion network for cloud and snow detection based on medium- and high-resolution imagery dataset," *Remote Sensing*, vol. 13, no. 23, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/23/4805

[106] W. McCelloch and W. Pitts, "A logical calculus of the idea immanent in neural nets," *Bulletin ofMathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[107] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.

[108] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[109] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

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

[111] S. Mohajerani and P. Saeedi, "Pragmatic augmentation algorithms for deep learning-based cloud and cloud shadow detection in remote sensing imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[112] K. Hu, D. Zhang, and M. Xia, "Cdunet: Cloud detection unet for remote sensing imagery," *Remote Sensing*, vol. 13, no. 22, 2021.

[113] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[114] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[115] X. Cui, K. Zheng, L. Gao, D. Yang, and J. Ren, "Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification," *Remote Sensing*, vol. 11, p. 2220, 09 2019.

[116] S. Skakun, J. Wevers, C. Brockmann, G. Doxani, M. Aleksandrov, M. Batič, D. Frantz, F. Gascon, L. Gómez-Chova, O. Hagolle, D. López-Puigdollers, J. Louis, M. Lubej, G. Mateo-García, J. Osman, D. Peressutti, B. Pflug, J. Puc, R. Richter, J.-C. Roger, P. Scaramuzza, E. Vermote, N. Vesel, A. Zupanc, and L. Žust, "Cloud mask intercomparison exercise (cmix): An evaluation of cloud masking algorithms for landsat 8 and sentinel-2," *Remote Sensing of Environment*, vol. 274, p. 112990, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425722001043

[117] M. Paperin, K. Stelzer, C. Lebreton, C. Brockmann, and J. Wevers, "Pixbox sentinel-2 pixel collection for cmix (version 1.0)," *Zenodo*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5036991

[118] *Product Guide: DPUCZDX8G for ZynqUltraScale+ MPSoCs*, Xilinx, 2022. [Online]. Available: https://docs.xilinx.com/viewer/book-attachment/H0IrnZTG1MgvDJfinrXCdw/ycA91e542EnF7iRsc2B0tA