Redes adversariales para la adaptación de dominio sobre imágenes de sensado remoto

Alberto S. Garea[1], Eloi Corral López[2], Francisco Argüello[3] y Dora B. Heras[1]

Resumen— **El sensado remoto es un campo esencial en múltiples aplicaciones científicas y tecnológicas como la detección de cambios en el uso del suelo o la gestión de recursos naturales. No obstante, uno de sus principales retos es la variabilidad entre los dominios de entrenamiento (origen) y test (objetivo) de los datos adquiridos, fenómeno conocido como desplazamiento de dominio y cuya presencia puede deberse a factores como variaciones atmosféricas, diferencias en las condiciones de iluminación, cambios estacionales o divergencias en las características de los sensores utilizados. En este trabajo se estudia una técnica de adaptación de dominio conocida como Adversarial Discriminator Domain Adaptation (ADDA) para la clasificación de imágenes de sensado remoto. Esta técnica se basa en descubrir una función de mapeo de los datos originales a un espacio de características común independiente de su dominio de origen. Para verificar la viabilidad de esta técnica en problemas de sensado remoto, se llevaron a cabo experimentos con diferentes propuestas de arquitectura del modelo encargado del mapeo. A su vez, se analizó la posible influencia del uso de segmentación en superpíxeles de las imágenes y la inclusión de un clasificador final basado en SVM sobre el espacio de características común. Las pruebas realizadas permitieron concluir que esta técnica presenta potencial de cara a enfrentar el problema de adaptación de dominio.**

Palabras clave— **sensado remoto, ADDA, adaptación de dominio, clasificación, red adversarial, SVM , segmentación en superpíxeles, desplazamiento de dominio, CNN, LeNet, ResNet.**

I. Introducción

En los últimos años los avances en las tecnologías de adquisición de imágenes de sensado remoto han impulsado notablemente la demanda de geoinformación, tanto en el ámbito público (agencias de gobierno y centros de investigación) como en el privado [1]. Este crecimiento ha facilitado el acceso a grandes volúmenes de datos, pero también ha planteado nuevos retos, especialmente en la clasificación automática de imágenes. Entre estos desafíos destaca la escasez de datos de referencia etiquetados, lo que dificulta el uso de técnicas de aprendizaje supervisado, dado que el proceso de etiquetado manual resulta costoso y lento [2]. Además, la utilización de modelos entrenados en un dominio fuente directamente sobre un dominio objetivo con características distintas suele conllevar una degradación significativa del rendimiento, debido a lo que se conoce como desplazamiento de dominio. Este fenómeno puede originarse por factores como las condiciones atmosféricas o las diferencias de temperatura o presión de los instrumentos de medida [3]. Estos hechos ponen de manifiesto la necesidad de desarrollar técnicas de adaptación de dominio que permitan reutilizar el conocimiento adquirido en un dominio fuente para aplicarlo eficazmente en un dominio objetivo, minimizando la pérdida de precisión.

El presente trabajo se enmarca dentro del aprendizaje por transferencia transductiva, según la taxonomía propuesta en [4], donde la tarea de clasificación se realiza sobre ambos dominios, pero solo se dispone de datos etiquetados en el dominio fuente. Las diferencias inherentes entre los dominios, unidas a la falta de etiquetas en el dominio objetivo, exigen estrategias específicas para abordar el problema.

Existen diversas aproximaciones para entrenar modelos robustos que permitan afrontar el problema del desplazamiento de dominio [5]. Una opción consiste en identificar y emplear características invariantes entre ambos dominios durante el entrenamiento [6]. Otra alternativa es el uso de técnicas semi-supervisadas, incorporando una fracción de datos etiquetados del dominio objetivo [7], o bien recurrir al aprendizaje activo, donde se selecciona un pequeño subconjunto de muestras del dominio objetivo para ser etiquetadas [8], esta última alternativa podría considerarse un caso especial del uso de técnicas semi-supervisadas. Finalmente, un enfoque ampliamente adoptado es la adaptación basada en la extracción de características (FE), cuyo objetivo es aprender una función de mapeo que proyecte los datos de ambos dominios (fuente y objetivo) en un espacio común donde las diferencias se reduzcan al mínimo.

Dentro de las técnicas de FE, destacan aquellas que emplean redes neuronales para aprender representaciones comunes. Entre ellas se encuentran los Denoising Autoencoders (DAE) [9], que comprimen los datos y los transforman a una nueva representación, y las arquitecturas adversariales [10], [11], que entrenan modelos de forma antagónica para optimizar la función de mapeo entre dominios. Este trabajo se centra en el análisis de una arquitectura adversarial, evaluando su capacidad para la adaptación de dominio en imágenes de sensado remoto.

El uso de redes neuronales profundas, aunque efectivo, implica elevados costes computacionales, lo que ha motivado el desarrollo de alternativas como PCA-Net [12], KPCA [13], ScatNet [14] o Transfer Component Analysis (TCA) [15], que buscan reducir la complejidad del entrenamiento sin sacrificar preci-

[1]Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, Spain, e-mail: {jorge.suarez.garea,dora.blanco}@usc.es.

[2]Escola Técnica Superior de Enxeñaría (ETSE), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, Spain, e-mail: eloi.corral@rai.usc.es.

[3]Dpto. de Arquitectura y Tecnología de Computadores, Universidade de Santiago de Compostela, 15782, Santiago de Compostela, e-mail: francisco.arguello@usc.es.

sión. Si bien este grupo de técnicas permiten reducir los costes computacionales en el proceso de entrenamiento para la adaptación de dominio, estas todavía presentan cierto margen de mejora en la clasificación de imágenes de sensado remoto [15].

En este artículo se evalúa la arquitectura propuesta en [10] y se analizan sus resultados para determinar su idoneidad a la hora de resolver problemas de adaptación de dominio sobre imágenes de sensado remoto. El resto del artículo se organiza de la siguiente manera: la Sección 2 presenta la arquitectura propuesta, la Sección 3 expone en detalle los métodos y herramientas utilizados, la Sección 4 describe el proceso de evaluación experimental y analiza los resultados obtenidos, y finalmente, la Sección 5 recoge las principales conclusiones .

II. Redes Generativas Antagónicas (GAN)

En esta sección se introducirá el concepto de Red Generativa Antagónica (GAN, por sus siglas en inglés) y la arquitectura base utilizada en este articulo.

Las redes GAN son un tipo de arquitectura introducida en [16] como un marco innovador para el aprendizaje generativo no supervisado. Esencialmente, se compone de dos redes neuronales profundas: un generador y un discriminador, que compiten entre sí en un proceso de entrenamiento adversarial. El generador intenta producir muestras sintéticas que imiten los datos reales, mientras que el discriminador se encarga de distinguir entre muestras reales y generadas.

A lo largo del proceso de entrenamiento, el generador mejora su capacidad para producir datos indistinguibles de los reales, en tanto que el discriminador se perfecciona en su tarea de diferenciación. Esta dinámica de juego de suma cero permite que ambos modelos se refuercen mutuamente, llevando al generador a aprender una distribución de datos próxima a la real. La Figura 1 ilustra la arquitectura típica de una GAN.

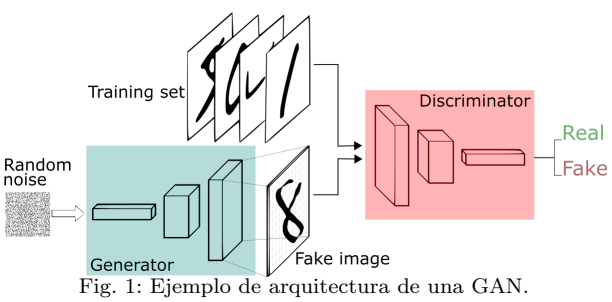

Fig. 1: Ejemplo de arquitectura de una GAN.

En esencia, estas redes logran transformar vectores aleatorios en muestras representativas del dominio definido por los datos etiquetados. Este mecanismo de transformación resulta especialmente útil en problemas de adaptación de dominio, ya que sugiere la posibilidad de reemplazar el vector aleatorio de entrada por imágenes que se desean adaptar. Así, mediante una arquitectura similar, el generador podría aprender una función de mapeo entre dominios. En este nuevo contexto, la red dejaría de tener un carácter generativo en sentido estricto, convirtiéndose en una arquitectura adversarial para la adaptación de dominio.

A. *Adversarial Discriminator Domain Adaptation (ADDA)*

En esta subsección se describe la arquitectura propuesta en [10], denominada Adversarial Discriminator Domain Adaptation (ADDA), orientada a resolver problemas de adaptación de dominio y que es la arquitectura base utilizada para los experimentos de este artículo.

En un problema típico de adaptación de dominio, se cuenta con datos provenientes de dos dominios diferentes, que en adelante se denominarán dominio fuente (o de partida) y dominio objetivo. Los datos del dominio fuente se denotarán como X_s con distribución conjunta $p_s(x_s, y_s)$ con $x_s \in X_s$ e $y_s \in Y_s$, y los del dominio objetivo como X_t con distribución $p_t(x_t, y_t)$ con $x_t \in X_t$ e $y_t \in Y_t$. La motivación principal de la adaptación de dominio es permitir la transferencia de modelos entrenados sobre un dominio a otro diferente, evitando la necesidad de contar con etiquetas en este último. Así, mientras se dispone de las etiquetas Y_s del dominio fuente, no se cuenta con etiquetas Y_t del dominio objetivo.

La estrategia propuesta se fundamenta en mapear los datos de ambos dominios a un espacio de características común sobre el cual pueda operar un clasificador único. Para ello, se construyen funciones de mapeo específicas para cada dominio: M_s para el dominio fuente y M_t para el objetivo. Este mapeo asimétrico, aunque conlleva un mayor número de parámetros y, por ende, mayor complejidad, permite optimizar cada función de forma independiente, facilitando el proceso de entrenamiento. El objetivo de estas funciones es minimizar la divergencia entre las distribuciones de salida $M_s(X_s)$ y $M_t(X_t)$, posibilitando el uso de un clasificador compartido $C = C_s = C_t$ que ha sido entrenado exclusivamente con datos etiquetados del dominio fuente.

El entrenamiento se divide en dos etapas claramente diferenciadas, representadas en la Figura 2.

En la primera etapa, denominada preentrenamiento, se entrenan simultáneamente la función de mapeo del dominio fuente M_s y el clasificador C. M_s está constituida por una red convolucional tradicional compuesta por capas convolucionales y de pooling, mientras que C se implementa como una red densa que produce probabilidades de pertenencia a cada clase. El entrenamiento se realiza mediante retropropagación utilizando las etiquetas Y_s. Al finalizar esta etapa, los pesos de M_s y C quedan fijados para las siguientes fases del proceso.

La segunda etapa, denominada entrenamiento, tiene como finalidad ajustar la función de mapeo del dominio objetivo M_t, de modo que las representaciones extraídas se alineen con las obtenidas por M_s. Para ello se emplea una arquitectura adversarial similar a las GAN. En este esquema, se introduce un modelo discriminador cuya tarea es identificar si una

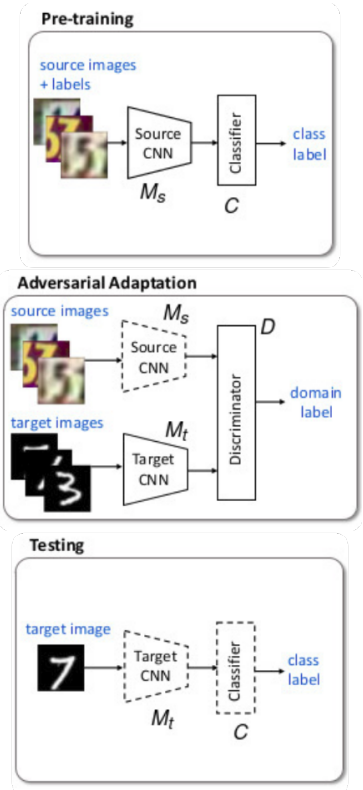

Fig. 2: Esquema de la arquitectura propuesta en [10].

muestra procesada proviene del dominio fuente o del objetivo, tras ser mapeada mediante M_s o M_t, respectivamente.

El discriminador se entrena mediante la función de coste definida en la ecuación 1, donde D representa la salida del discriminador y E representa el error cometido. De esta forma, $E_{x_s \sim X_s}[log\ D(M_s(x_s))]$ se corresponderá con el error cometido por el discriminador al identificar muestras mapeadas desde el dominio fuente o de partida y $E_{x_t \sim X_t}[log(1-D(M_t(x_t)))]$ el cometido para las mapeadas desde el dominio objetivo.

$$L_{adv_D}(X_s, X_t, M_s, M_t) = -E_{x_s \sim X_s}[\log D(M_s(x_s))] - E_{x_t \sim X_t}[\log(1 - D(M_t(x_t)))] \quad (1)$$

Por su parte, el modelo generativo en esta etapa es M_t, cuyo propósito es engañar al discriminador. A diferencia de las GAN tradicionales, no se requiere generar imágenes, sino que basta con transformar los datos del dominio objetivo a un espacio de características invariante. M_t comparte arquitectura con M_s y se inicializa con los pesos obtenidos durante el preentrenamiento, lo que favorece la convergencia y coherencia entre dominios.

Para el entrenamiento de M_t se contemplan dos estrategias de función de coste. La primera (ecuación 2) consiste en adoptar una estrategia tipo minimax, utilizando el gradiente negativo de la función de coste del discriminador. Sin embargo, esta puede provocar desvanecimiento del gradiente si el discriminador converge prematuramente.

$$L_{adv_{M_t}} = -L_{adv_D} \quad (2)$$

Como alternativa más robusta, se propone invertir las etiquetas del discriminador (ecuación 3), lo cual proporciona gradientes más intensos durante el entrenamiento, mejorando la estabilidad del proceso de optimización.

$$L_{adv_{M_t}}(X_s, X_t, D) = -E_{x_t \sim X_t}[\log D(M_t(x_t))] \quad (3)$$

A través de este esquema de entrenamiento adversarial, se busca que M_t aprenda a proyectar los datos del dominio objetivo en un espacio de características indistinguible del proyectado por M_s. Si el entrenamiento se ha llevado a cabo correctamente, el clasificador C, entrenado únicamente con datos etiquetados del dominio fuente, podrá generalizar adecuadamente sobre el dominio objetivo sin una merma significativa en su desempeño.

B. Codificadores

En esta subsección se describen las dos arquitecturas candidatas para efectuar el mapeo de los datos desde los dominios fuente y objetivo hacia un espacio común de características. Ambos mapeos, M_s y M_t, aunque distintos entre sí (mapeo asimétrico), comparten una misma arquitectura subyacente. En este trabajo, dichas arquitecturas se implementan mediante codificadores, los cuales permiten transformar la representación original de los datos de entrada, resaltando únicamente la información más relevante. Los codificadores considerados a continuación tienen en común el uso predominante de capas convolucionales.

B.1 Codificador básico

Esta arquitectura se basa en la extracción jerárquica de características mediante capas convolucionales, complementadas con capas de normalización por lotes (*Batch Normalization*) que estabilizan y aceleran el proceso de entrenamiento. La reducción progresiva de la dimensionalidad se logra a través de operaciones de muestreo máximo (*MaxPool*). Finalizada esta etapa, los mapas de características resultantes se aplanan en un vector unidimensional, el cual se somete a una capa de regularización por abandono (*Dropout*) para mitigar el sobreajuste, y a continuación se introduce en una capa densa. La salida resultante es un vector de tamaño 256×1, que será posteriormente utilizado como entrada del clasificador. La Figura 3 muestra un esquema ilustrativo de esta arquitectura, mientras que la Tabla I detalla la configuración de sus capas. En las tablas de esta sección, se utilizarán las abreviaturas *Conv.* para capas convolucionales y *Batch Norm.* para normalización por lotes.

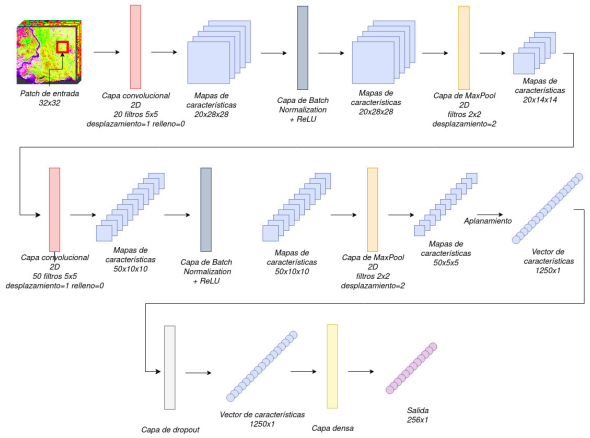

Fig. 3: Diagrama de la estructura del codificador básico.

Nº	Tipo	Salida	Filtro	Desplazamiento	Relleno	Activación
1	Conv. 2D	20x28x28	5x5	1x1	0	-
2	Batch Norm. 2D	20x28x28	-	-	-	ReLU
3	MaxPool 2D	20x14x14	2x2	2x2	-	-
4	Conv. 2D	50x10x10	5x5	1x1	0	-
5	Batch Norm. 2D	50x10x10	-	-	-	ReLU
6	MaxPool 2D	50x5x5	2x2	2x2	-	-
7	Dropout	1250x1	-	-	-	-
8	Densa	256x1	-	-	-	-

Tabla I: Estructura de capas codificador básico.

B.2 Codificador ResNet

Aunque las redes neuronales profundas ofrecen un potencial considerable, incrementar su profundidad conlleva desafíos como la atenuación del gradiente durante la retropropagación y la degradación del rendimiento conforme se añaden capas [17]. En respuesta a estas limitaciones, se introdujeron las Redes Neuronales Residuales (ResNet) [17], que incorporan conexiones de salto (*skip connections*) entre capas. Estas conexiones permiten definir bloques residuales, en los que la salida del bloque corresponde a la suma de la salida de su última capa con su entrada original. Esto facilita el flujo del gradiente, mitigando su desaparición y permitiendo la construcción de redes mucho más profundas.

En la Figura 4 se muestra un ejemplo típico de bloque residual. En nuestro diseño, se emplean dos tipos de bloques: los bloques sin reducción de dimensionalidad, en los que se preserva el tamaño del mapa de características mediante filtros 3×3 con desplazamiento 1×1 y relleno 1; y los bloques con reducción de dimensionalidad, en los que se aplica un desplazamiento de 2×2 para disminuir la resolución. En este segundo caso, la entrada al bloque también se transforma mediante una convolución con igual desplazamiento, de modo que ambas ramas tengan dimensiones compatibles al realizar la suma.

La arquitectura completa del codificador ResNet comienza con una capa convolucional estándar seguida de normalización por lotes. A continuación, se organizan tres capas residuales, cada una formada por dos bloques. La primera capa (Figura 5) contiene únicamente bloques sin reducción de dimensionalidad. La segunda (Figura 6) y la tercera (Figura 7) incluyen un bloque con reducción seguido de uno sin ella. Finalmente, se aplica una operación de *Average*

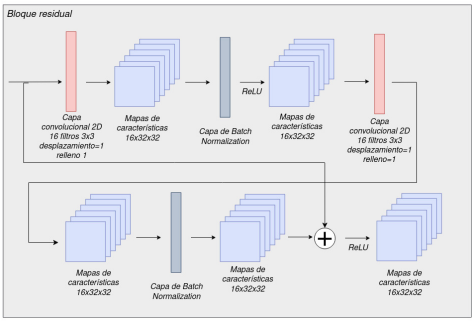

Fig. 4: Ejemplo de bloque residual.

Pooling de 8×8, que promedia los valores dentro de cada ventana en lugar de tomar el máximo, seguida del aplanamiento del resultado en un vector de 256×1.

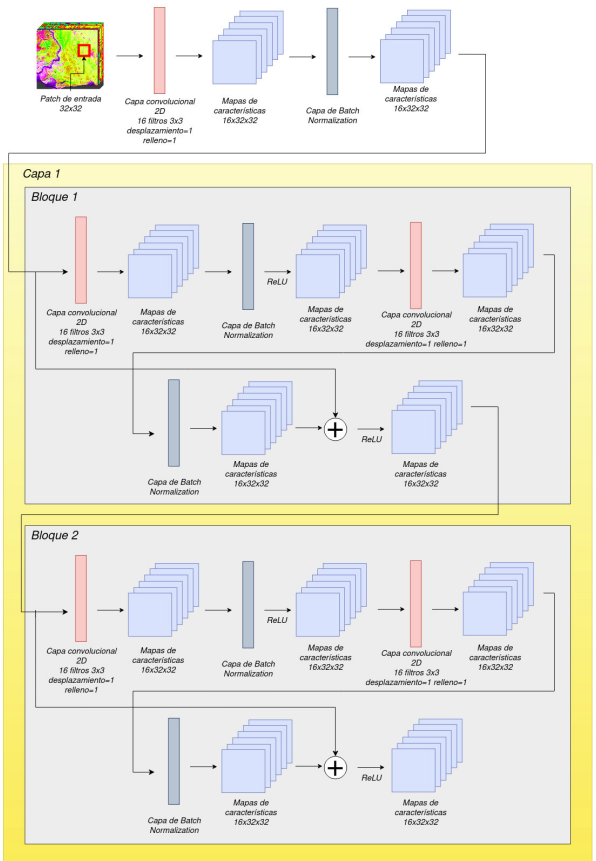

Fig. 5: Capa 1 del codificador ResNet conformada por dos bloques sin reducción de dimensionalidad.

III. Métodos y herramientas

En esta sección se detallarán tanto las características del hardware empleado en los experimentos como las medidas de precisión y tiempo utilizadas para evaluarlos. También se describirán los conjuntos de datos utilizados en los experimentos.

Para la realización de las pruebas se ha empleado en un PC con 2 procesadores Intel Xeon E52623v4, 128GB de memoria RAM, sistema operativo Ubuntu 18.04 y dos tarjetas Nvidia GP102GL (Tesla P40) de 24GB de memoria GDDR5. Se ha usado la versión 12.1 del conjunto de herramientas CUDA para Linux con soporte para cuDNN y las librerías Pytorch 2.0.0,

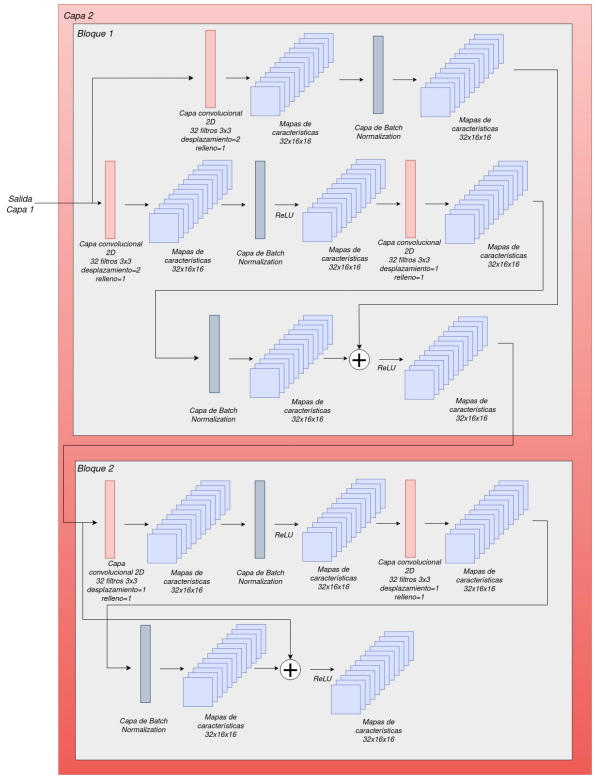

Fig. 6: Capa 2 del codificador ResNet conformada por dos bloques, el primero con reducción de dimensionalidad y el segundo sin esta.

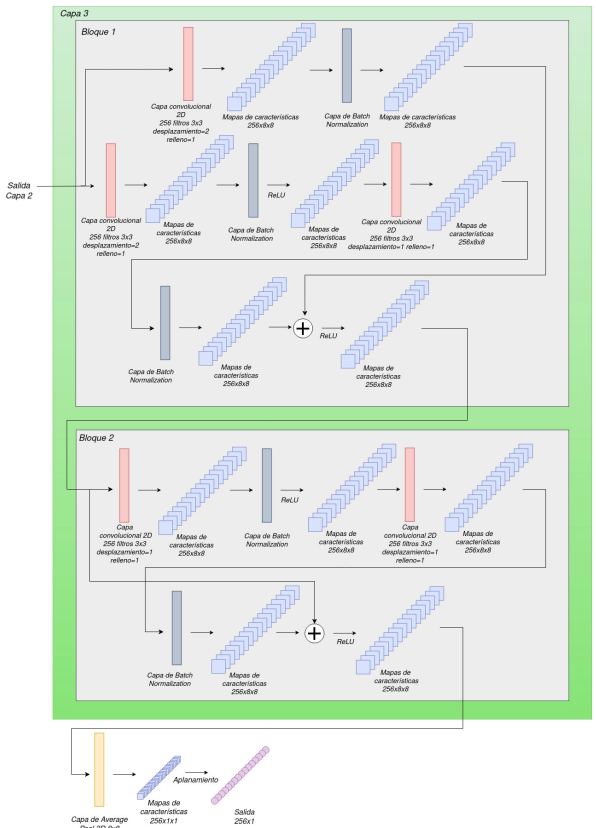

Fig. 7: Capa 3 del codificador ResNet conformada por dos bloques, el primero con reducción de dimensionalidad y el segundo sin esta.

Numpy 1.24.2 y Scikit-learn 1.2.2. Para la ejecución del código se utilizó un entorno Conda con la versión 3.10.9 de Python.

La precisión de los resultados obtenidos en los distintos experimentos está expresada en términos de precisión total (OA), que es el porcentaje de píxeles correctamente clasificados y precisión media (AA), que se obtiene como la media de las precisiones de las diferentes clases. Todos los resultados han sido calculados como el valor medio de 12 ejecuciones independientes.

A. Conjunto de datos

Para el desarrollo de la fase experimental, se dispuso de una imagen multiespectrales correspondiente con una zona del río Oitavén de coordenadas 42°22'15.48" N 8°25'47.07" W. Esta imagen fue particionada a su vez en dos subimágenes. Esta división permitió establecer dos dominios diferenciados: un dominio fuente y un dominio objetivo.

- Imagen Fuente Zona 1 (Figura 8): sus dimensiones son 2501 × 1898 (alto × ancho).
- Imagen Objetivo Zona 1 (Figura 9): sus dimensiones son 6689 × 4701 (alto × ancho).

Fig. 8: Imagen del dominio fuente o de partida de la Zona 1 con las tres bandas clásicas de RGB y su respectivo mapa de etiquetas.

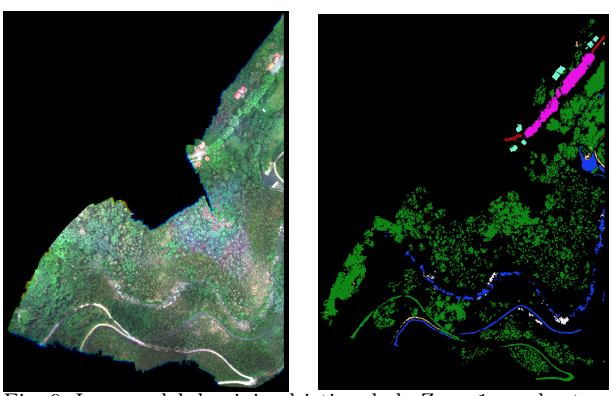

Fig. 9: Imagen del dominio objetivo de la Zona 1 con las tres bandas clásicas de RGB y su respectivo mapa de etiquetas.

La imagen empleada en este estudio fue adquirida durante el año 2018 utilizando una cámara multiespectral MicaSense RedEdge-MX, la cual fue instalada a bordo de un vehículo aéreo no tripulado (UAV) que operó a una altitud aproximada de 120 metros. Esta cámara se distingue por su capacidad para capturar imágenes compuestas por cinco bandas espectrales. En concreto, registra las bandas convencionales del espectro visible: rojo (668 nm de longitud de onda), verde (560 nm) y azul (475 nm), así como dos bandas adicionales: el infrarrojo cercano (842 nm) y la denominada *red-edge* (717 nm), esta última de especial relevancia para tareas de clasificación en imágenes que contienen vegetación [18]. El

proceso de etiquetado de la imagen fue llevado a cabo por el equipo de investigación en imágenes de sensado remoto del Centro Singular de Investigación en Tecnologías Inteligentes (CiTIUS), en colaboración con otros expertos en la materia.

En las Figuras 8 y 9 se recogen las imágenes, incluyendo tanto su vista en RGB como su mapa de etiquetas. A su vez, se puede observar un resumen de la distribución de las clases en la Tabla II. Se presenta tanto su distribución a nivel de píxeles como la resultante tras aplicar segmentación por superpíxeles.

	Fuente		Objetivo	
Clase	Píx.	Seg.	Píx.	Seg.
1. Agua	3944	43	276159	2880
2. Tejado	16787	178	60347	669
3. Asfalto	11845	124	26188	272
4. Tierra	31578	336	28776	371
5. Roca	138440	1535	233494	2847
6. Cemento	5414	75	40309	462
7. Vegetación	2212402	22300	2838245	30976
Total:	2420410	24591	3503518	38477

Tabla II: Distribución de los datos por clase en ambas imágenes de la Zona 1, tanto a nivel de píxel como de segmentos.

IV. Resultados experimentales

En esta sección se presentan los resultados obtenidos a partir de los distintos experimentos realizados con el objetivo de evaluar el rendimiento de la red ADDA en tareas de clasificación de imágenes de teledetección, considerando diversas configuraciones y arquitecturas de codificadores. Asimismo, se examina el impacto que tiene la segmentación mediante superpíxeles y la incorporación de un clasificador SVM sobre el espacio de características adaptado.

Durante la fase experimental, se llevaron a cabo evaluaciones tanto a nivel de píxel como a nivel de superpíxel. Para este último, se empleó el algoritmo de segmentación *Waterpixels* [19], configurado con los siguientes parámetros: longitud de segmento de 20 píxeles, regularización con un valor de 0.5, tamaño mínimo de segmento de 100 píxeles y una conectividad de 8 con los píxeles adyacentes.

En cuanto al entrenamiento de las distintas redes de Deep Learning (DL), se adoptaron configuraciones específicas según la fase. En el preentrenamiento se realizaron 50 épocas, mientras que en el entrenamiento adversarial esta cifra se duplicó, alcanzando las 100 épocas. El tamaño del lote (*batch size*) se fijó en 128 muestras. Para la optimización de los modelos se empleó el algoritmo Adam (Adaptive Moment Estimation) [20], ampliamente reconocido en el ámbito del aprendizaje profundo por su capacidad para ajustar individualmente la tasa de aprendizaje de cada parámetro, considerando además el momento de los gradientes pasados en el proceso de actualización. La configuración de Adam incluyó un valor de *weight decay* de $2{,}5 \times 10^{-5}$. Además, durante el preentrenamiento, los coeficientes β_1 y β_2 se establecieron en 0.9 y 0.999, respectivamente; sin embargo, durante el entrenamiento adversarial estos valores se ajustaron a 0.5 y 0.999.

Respecto a la inicialización de los pesos, se adoptaron estrategias diferenciadas según el componente de la red. Para el clasificador, el discriminador y el codificador básico se utilizó la inicialización propuesta por Kaiming [21], mientras que para el codificador basado en ResNet se recurrió a la inicialización aleatoria predeterminada de la biblioteca PyTorch.

Por último, en lo que concierne al clasificador SVM, se empleó una función de base radial (RBF) con un valor de $\gamma = 2 \times 10^{-10}$ y un parámetro de regularización $C = 32$.

A. Experimentos

Con el objetivo de evaluar el rendimiento de las distintas configuraciones arquitectónicas, se diseñó un conjunto de experimentos que pueden agruparse en tres bloques principales. El primer grupo de pruebas se centró en el ajuste de los hiperparámetros asociados a las tasas de aprendizaje de la red adversarial, dado que encontrar un equilibrio adecuado entre el aprendizaje del codificador y del discriminador es crucial para lograr una adaptación efectiva.

Para ello, se definieron diversas combinaciones de tasas de aprendizaje. En particular, se consideraron los valores 0,0001, 0,00001, 0,000005 y 0,000001 para el discriminador, y 0,0001, 0,00001 y 0,000005 para el codificador. Para cada combinación se evaluaron tanto variantes con tasa de aprendizaje adaptativa como no adaptativa. Estas combinaciones se aplicaron a las dos arquitecturas de codificadores seleccionadas, considerando además la presencia o ausencia de segmentación en el entrenamiento.

Una vez obtenidos los resultados de este bloque, se seleccionaron las configuraciones de tasas de aprendizaje que ofrecieron el mejor desempeño. La elección se basó principalmente en aquellas combinaciones que reportaron un incremento significativo en la métrica *Overall Accuracy* (OA) con respecto al caso base sin codificador adaptado. No obstante, también se consideraron los valores de *Average Accuracy* (AA), de modo que si una configuración con OA óptima no presentaba mejoras relevantes en AA, y existía otra ligeramente inferior en OA pero con un desempeño notable en AA, se optaría finalmente por esta última.

A partir de las configuraciones óptimas identificadas en esta fase, se procedió a ejecutar un segundo bloque de experimentos. En este caso, se realizaron múltiples ejecuciones por combinación de uso o no de segmentación y codificador, con su respectiva configuración óptima. Esta estrategia permitió obtener una medida estadísticamente más representativa del rendimiento de cada arquitectura, minimizando la influencia de la aleatoriedad inherente al proceso de entrenamiento. En concreto, se llevaron a cabo 12 ejecuciones completas por cada combinación, y se reportaron los valores medios de las métricas obtenidas.

Finalmente, se repitió el procedimiento anterior in-

corporando una etapa adicional de clasificación basada en SVM. Esta etapa final se entrenó utilizando los datos del dominio fuente junto con el codificador correspondiente, previamente ajustado mediante el esquema ADDA. Para la fase de test, se utilizaron los datos del dominio objetivo, también con su codificador entrenado. Cabe destacar que el clasificador SVM, implementado mediante la librería `scikit-learn`, se ejecuta sobre CPU, lo cual implica un coste computacional considerable. Por este motivo, se optó por separar esta etapa del bloque anterior, ya que repetir 12 ejecuciones por combinación, sumando los tiempos asociados al entrenamiento y prueba del modelo SVM, resultaría excesivamente costoso en términos de tiempo.

Con respecto a la partición de los datos en conjuntos de entrenamiento y prueba, a lo largo de todo el proceso se aplicó una selección aleatoria de muestras por clase. En concreto, se seleccionaron 200 muestras por clase, una cantidad razonable que permite mantener el equilibrio entre eficiencia computacional y representatividad del conjunto de datos. Esta elección asegura un conjunto balanceado en el caso del uso de píxeles, dado que se dispone de más de 200 muestras por clase. Sin embargo, en el caso de los superpíxeles, algunas clases están escasamente representadas incluso con segmentaciones finas. A pesar de ello, se mantuvo el criterio de 200 muestras por clase también en la versión con segmentación, dado que la mayoría de las clases disponen de suficientes superpíxeles para cumplir dicho umbral.

B. Resultados

En este apartado se presentan los resultados obtenidos tras la ejecución del proceso experimental previamente descrito.

En la Tabla III se muestra la selección final de tasas de aprendizaje para la red adversarial, así como las mejoras alcanzadas en las métricas de *Overall Accuracy* (OA) y *Average Accuracy* (AA) en comparación con el escenario sin adaptación. De manera general, se observa que la utilización de tasas de aprendizaje adaptativas resultó beneficiosa en la mayoría de los casos evaluados (3 de 4). No obstante, no se identifica un patrón uniforme en cuanto a los valores óptimos de estas tasas, destacando la disparidad entre las combinaciones más efectivas. En términos de precisión, se constata una mejora en ambos clasificadores respecto al escenario sin adaptación.

	Codificador Básico			Codificador ResNet		
	Configuración	OA	AA	Configuración	OA	AA
Píxeles	lrd=0.000001 lrg=0.0001 No adaptativo	+26.04	+4.70	lrd=0.0001 lrg=0.00001 Adaptativo	+24.16	+21.90
Segmentación	lrd=0.00001 lrg=0.000005 Adaptativo	+12.69	+3.18	lrd=0.000001 lrg=0.000005 Adaptativo	+15.78	+7.96

Tabla III: Selección de mejores configuraciones tras la búsqueda de hiperparámetros junto a la mejora de OA y AA obtenida con respeto a no realizar adaptación. La abreviatura *lrd* representa la tasa de aprendizaje del discriminador mientras que *lrg* será la del codificador.

La Tabla IV recoge los promedios obtenidos a partir de 12 ejecuciones para cada configuración, considerando tanto el uso como la ausencia de adaptación. Adicionalmente, se incluyen los resultados medios obtenidos al incorporar un clasificador SVM. En la mayoría de los casos, se detecta una ligera disminución en el rendimiento con respecto al uso de codificadores sin adaptación, salvo en el caso del codificador ResNet aplicado a nivel de píxel, donde se logra una mejora del 12 % en OA y del 10 % en AA. Estas cifras se incrementan aún más al aplicar un modelo SVM sobre las representaciones extraídas por los codificadores finales, alcanzando mejoras del 14 % en OA y del 17 % en AA.

Un análisis más detallado permite ofrecer una visión más profunda de los resultados. La Tabla IV incluye también la desviación estándar asociada a cada métrica promedio. Al observar estos valores, se evidencia que en muchos casos el rendimiento experimentó una notable variabilidad a lo largo de las 12 ejecuciones. Esta fluctuación sugiere que, aunque no siempre se alcanzó una mejora sustancial, en ciertos ensayos la técnica de adaptación de dominio resultó efectiva.

		Codificador Básico		Codificador ResNet	
		OA	AA	OA	AA
Sin Adaptación	Píxeles	58.29+5.84	47.52+5.52	57.78+4.04	45.27+4.98
	Segmentación	61.33+4.05	48.36+5.03	59.01+6.71	47.03+4.89
Con Adaptación	Píxeles	49.20+15.07	49.28+7.67	69.91+5.93	55.62+7.12
	Segmentación	52.99+11.24	47.58+3.62	46.68+13.50	47.68+7.92
SVM	Píxeles	54.15+9.40	47.43+0.88	71.10+8.60	62.09+10.11
	Segmentación	34.57+7.69	25.61+6.73	46.74+0.82	27.35+3.72

Tabla IV: Resultados promedio de las 12 ejecuciones.

Los casos más exitosos se corresponden con aquellas configuraciones que mostraron resultados menos inestables entre ejecuciones. Esta observación indica que, en situaciones donde las métricas con y sin adaptación son similares, es posible que algunas ejecuciones hayan alcanzado una adaptación efectiva, incluso si los promedios no reflejan mejoras significativas. A continuación, se detallan estos casos, con el objetivo de ilustrar el potencial de la técnica de adaptación de dominio en contextos en los que se logre un entrenamiento más consistente.

La Tabla V presenta los mejores resultados obtenidos en una única ejecución para cada configuración. Se constata que en todos los casos analizados, al menos una ejecución logró una mejora en OA mediante la adaptación. En particular, destacan las mejoras obtenidas por los codificadores a nivel de píxel, alcanzando incrementos del 25 % y 23 % en OA, respectivamente. Estos resultados refuerzan la idea de que, si bien el entrenamiento de esta técnica presenta desafíos y no garantiza consistencia entre ejecuciones, sí es posible alcanzar desempeños notables de adaptación de dominio para distintas arquitecturas de codificadores.

		Codificador Básico		Codificador ResNet	
		OA	AA	OA	AA
Sin Adaptación	Píxeles	51.60	42.19	52.35	58.59
	Segmentación	57.96	53.18	47.26	45.40
Con Adaptación	Píxeles	76.30	44.55	75.64	66.91
	Segmentación	72.19	47.75	57.81	45.75

Tabla V: Promedio de precisión (media de 12 ejecuciones).

En relación con la influencia de la segmentación en el rendimiento, se observa que el codificador básico se benefició significativamente de su uso. En contraste, el codificador ResNet evidenció una pérdida de rendimiento, posiblemente atribuida a la reducción de información causada por el proceso de segmentación. Esto podría deberse a la mayor complejidad de dicha arquitectura, que parece requerir una mayor riqueza informativa, tal como la ofrecida por los datos a nivel de píxel.

La Tabla VI muestra los tiempos de ejecución correspondientes a las distintas configuraciones de entrenamiento y test. Se evidencia que la arquitectura ResNet implica una mayor carga computacional, duplicando aproximadamente el tiempo de procesamiento en comparación con el codificador básico cuando se trabaja a nivel de píxel y se aplica únicamente ADDA. Asimismo, se destaca el ahorro de tiempo asociado al uso de superpíxeles, que representa aproximadamente una cuarta parte del tiempo necesario respecto al enfoque basado en píxeles para la aplicación de la adaptación, sin considerar la clasificación SVM adicional. Cuando esta última se incluye, los tiempos aumentan considerablemente, tanto por el mayor volumen de datos como por la limitación de la implementación de SVM en `scikit-learn`, que solo permite ejecución en CPU.

Estos factores, sumados al modesto desempeño del clasificador SVM en comparación con los clasificadores basados en redes neuronales, sugieren que la mejor alternativa es optar por un clasificador basado en DL. Por último, cabe destacar que la reducción del tiempo de entrenamiento gracias al uso de segmentación, junto con el hecho de que —salvo en el caso de ResNet— las versiones segmentadas ofrecen resultados comparables o incluso superiores, refuerzan la idea de que el uso de imágenes segmentadas en superpíxeles constituye una práctica recomendable en problemas de adaptación de dominio y clasificación de imágenes de sensado remoto.

		Codificador Básico	Codificador ResNet
ADDA	Píxeles	674.72s	1028.33s
	Segmentos	197.12s	280.10s
ADDA + SVM	Píxeles	3169.45s	4466.02s
	Segmentos	268.24s	403.16s

Tabla VI: Tiempos conjuntos de entrenamiento y test en segundos.

V. Conclusiones

En este artículo se presenta un estudio sobre la aplicación de la red adversarial ADDA en el contexto de adaptación de dominio para tareas de clasificación de imágenes de sensado remoto. Esta técnica permite aprender una función de mapeo que proyecta datos provenientes de dos dominios distintos —fuente y objetivo— hacia un espacio común de características, sobre el cual puede operar un modelo de clasificación. De este modo, se atenúan los efectos provocados por el desplazamiento de dominio. Una de las principales ventajas de este enfoque radica en que no requiere datos etiquetados del dominio objetivo, lo que posibilita el entrenamiento del modelo utilizando exclusivamente información etiquetada del dominio fuente. Esta propiedad resulta especialmente relevante dada la elevada carga de trabajo y el coste económico asociados al proceso de etiquetado manual por parte de expertos en teledetección.

La red ADDA ha demostrado ser efectiva a la hora de incrementar la precisión respecto a escenarios sin adaptación, alcanzando, en determinados casos, mejoras notables que evidencian el potencial de esta metodología para investigaciones futuras. No obstante, se ha identificado cierta variabilidad en los resultados entre distintas ejecuciones, lo que resalta la necesidad de profundizar en la estabilidad y robustez del método.

Durante la fase experimental se evaluaron dos arquitecturas de codificadores: una basada en redes neuronales convolucionales (CNN) y otra que incorpora una estructura más compleja inspirada en la arquitectura ResNet. Los resultados indicaron un rendimiento global comparable entre ambas configuraciones, sin que se identificara una superioridad clara. Sin embargo, el denominado codificador básico evidenció un mayor potencial en términos de precisión máxima alcanzada.

Adicionalmente, los experimentos confirmaron que, en la mayoría de los casos, la segmentación mediante superpíxeles no sólo no compromete la precisión del modelo, sino que incluso puede contribuir a mejorarla. Este enfoque también permitió una reducción significativa del coste computacional, lo cual lo convierte en una alternativa atractiva en escenarios con recursos limitados.

Como trabajo futuro, se plantea la exploración de nuevas arquitecturas de codificadores, abarcando tanto modificaciones incrementales de las empleadas en este estudio como la adopción de enfoques más innovadores, tales como codificadores basados en redes neuronales recurrentes, que podrían ofrecer una mejor capacidad de modelado en tareas de adaptación de dominio complejas.

Agradecimientos

Este trabajo fue financiado en parte por las subvenciones PID2022-141623NB-I00 y TED2021-130367B-I00 financiadas por MCIN/AEI/10.13039/501100011033 y por ≪European Union NextGenerationEU/PRTR≫. También fue financiado por la Xunta de Galicia - Consellería de Educación, Ciencia, Universidades e Formación Profesional [acreditación Centro de investigación de Galicia 2024-2027 ED431G-2023/04 y acreditación Grupo Competitivo de Referencia, ED431C-2022/16], y por ≪FEDER/UE≫.

Referencias

[1] Haroon Sajjad and Pavan Kumar, "Future challenges and perspective of remote sensing technology," in *Applications and Challenges of Geospatial Technology: Potential and Future Trends*, pp. 275–277. Springer, 2018.

[2] Muhammad Ahmad, "Ground truth labeling and sam-

ples selection for hyperspectral image classification," *Optik*, vol. 230, pp. 166267, 2021.
[3] Guorui Jia et al., "Detection and correction of spectral shift effects for the airborne prism experiment," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, pp. 6666–6679, 2017.
[4] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
[5] Devis Tuia, Claudio Persello, and Lorenzo Bruzzone, "Domain adaptation for the classification of remote sensing data: An overview of recent advances," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 41–57, 2016.
[6] Emma Izquierdo-Verdiguier et al., "Encoding invariances in remote sensing image classification with svm," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 5, pp. 981–985, 2012.
[7] Lorenzo Bruzzone and Diego Fernàndez Prieto, "Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 2, pp. 456–460, 2001.
[8] Devis Tuia, E. Pasolli, and William J. Emery, "Using active learning to adapt remote sensing image classifiers," *Remote Sensing of Environment*, vol. 115, no. 9, pp. 2232–2242, 2011.
[9] Pascal Vincent et al., "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.
[10] Eric Tzeng et al., "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176.
[11] Y. Ganin et al., "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.
[12] Tsung-Han Chan et al., "Pcanet: A simple deep learning baseline for image classification?," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.
[13] Mathieu Fauvel, Jocelyn Chanussot, and Jon Atli Benediktsson, "Kernel principal component analysis for the classification of hyperspectral remote sensing data over urban areas," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–14, 2009.
[14] Wojciech Czaja, Ilya Kavalerov, and Weilin Li, "Scattering transforms and classification of hyperspectral images," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIV. SPIE*, 2018, pp. 125–136.
[15] S. Alberto Garea, Dora B. Heras, and Francisco Argüello, "Tcanet for domain adaptation of hyperspectral images," *Remote Sensing*, vol. 11, no. 19, pp. 2289, 2019.
[16] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
[17] Kaiming He et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[18] P. J. Zarco-Tejada et al., "Understanding the temporal dimension of the red-edge spectral region for forest decline detection using high-resolution hyperspectral and sentinel-2a imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 137, pp. 134–148, 2018.
[19] Vaïa Machairas et al., "Waterpixels," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3707–3716, 2015.
[20] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[21] Kaiming He et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.