Redes convolucionales basadas en escalado uniforme de dimensiones para la clasificación de imágenes multiespectrales

Nicolás Vilela-Pérez[1], Álvaro G. Dieste[1], Dora B. Heras[1,2] y Francisco Argüello[2]

Resumen— **En el campo de la clasificación de imágenes multi e hiperespectrales se ha popularizado en la última década el uso de técnicas basadas en aprendizaje profundo, en particular las basadas en redes convolucionales. Dado que su coste computacional es más alto que el de las técnicas de aprendizaje automático más clásicas, recientemente se han propuesto nuevos esquemas de clasificación que tienen como objetivo reducir el tiempo de ejecución requerido por ellas. Un modelo de red que ha sido aplicado con éxito a problemas de clasificación de imágenes en la literatura es la llamada red EfficientNet. Este tipo de redes de aprendizaje profundo se caracterizan por realizar un escalado uniforme de las tres dimensiones que componen la red: profundidad, anchura y resolución. De esta forma, se obtiene un modelo con buenas precisiones de clasificación a la vez que se cumplen las restricciones computacionales impuestas por el problema. En este trabajo se propone la adaptación de las redes tipo EfficientNet para resolver la clasificación de imágenes multiespectrales de alta resolución. Se analizan diferentes configuraciones de este tipo de redes variando el escalado de la arquitectura, tanto en términos de precisión de la clasificación del esquema resultante como de tiempo de ejecución. Los experimentos realizados sobre imágenes multiespectrales de alta resolución espacial obtenidas mediante vehículos aéreos no tripulados han mostrado que algunas variantes específicas de EfficientNet permiten una reducción importante del coste computacional sin degradar la precisión de clasificación.**

Palabras clave— **multiespectral, clasificación de imágenes, redes convolucionales, GPU, aprendizaje profundo**

AA Average Accuracy
Adam Adaptive Moment Estimation
CNN Convolutional Neural Network
DWConv Depthwise Convolution
DWSConv Depthwise Separable Convolution
κ Coeficiente kappa de Cohen
MBConv Mobile Inverted Bottleneck
MSI Multispectral Imaging
OA Overall Accuracy
PWConv Pointwise Convolution
ResNet Residual Network
SE Squeeze-and-Excitation
TTPE Training Time Per Epoch
WP Waterpixels

[1]Centro Singular de Investigación en Tecnologías Inteligentes (CiTIUS), Universidad de Santiago de Compostela, 15782, Santiago de Compostela, España. E-mails: {nicolas.vilela.perez, alvaro.goldar.dieste, dora.blanco}@usc.es.

[2]Departamento de Electrónica y Computación, Universidad de Santiago de Compostela, 15782, Santiago de Compostela, España. E-mail: francisco.arguello@usc.es.

I. Introducción

En el campo de teledetección es habitual resolver el problema de clasificación, que consiste en la asignación de una clase a cada píxel de la imagen [1]. Para realizar esta tarea, es común en la actualidad utilizar métodos de clasificación basados en redes de aprendizaje profundo, ya que obtienen mejores métricas de precisión que los métodos de aprendizaje automático tradicionales [2][3]. Dentro de las técnicas de aprendizaje profundo para la clasificación de imágenes destacan en número las basadas en redes neuronales convolucionales –Convolutional Neural Network (CNN)– [4], que usan capas convolucionales para la extracción de características espaciales y/o espectrales. Existen diversos tipos de CNNs, tales como las redes residuales –Residual Network (ResNet)– [5], que son una elección habitual hoy en día para la clasificación de imágenes [6]. Este tipo de redes se caracteriza por la inserción de conexiones que saltan capas, facilitando así el entrenamiento de redes profundas al evitar el problema del desvanecimiento del gradiente [7].

En los últimos años, un grupo importante de avances en el diseño de esquemas de clasificación se ha centrado en la creación de modelos que no consuman muchos recursos computacionales, con el objetivo, entre otros, de usar los esquemas en dispositivos móviles con poca capacidad de cómputo [8] o de obtener un tiempo de ejecución reducido.

Un modelo de red creado con el propósito de reducir el coste computacional de la clasificación es el propuesto por Google en 2019: la arquitectura EfficientNet [9]. La principal novedad introducida con esta arquitectura es que se pueden crear diferentes redes variando uniformemente el tamaño de las tres dimensiones que las definen: profundidad, anchura y resolución. Esto se hace a través de un único coeficiente de escalado llamado coeficiente compuesto. Este coeficiente se define como un exponente que afecta a las tres dimensiones mencionadas, permitiendo una adaptación a los recursos computacionales del dispositivo usado y/o a los requerimientos en tiempo de computación a la vez que se alteran mínimamente las precisiones de clasificación. Gracias a su arquitectura, la EfficientNet iguala e incluso supera las precisiones de clasificación de imágenes obtenidas mediante otras redes basadas en CNN, como son las redes tipo ResNet [9].

En este trabajo se propone una técnica de clasificación de imágenes multiespectrales –Multispectral

Imaging (MSI)– de alta resolución basada en aplicar la arquitectura EfficientNet sobre imágenes segmentadas en regiones homogéneas. Se analizan diferentes configuraciones basadas en el escalado de la arquitectura de la red, tanto en términos de precisión de la clasificación del esquema resultante como de tiempo de ejecución.

El resto del artículo se organiza como sigue. El Apartado II presenta los detalles de la técnica de clasificación propuesta. El Apartado III detalla la técnica de clasificación propuesta y sus variantes. El Apartado IV discute los experimentos realizados y que permiten alcanzar las conclusiones discutidas en el Apartado V.

II. Contexto científico

En esta sección se detalla el método de clasificación propuesto en el contexto de la bibliografía.

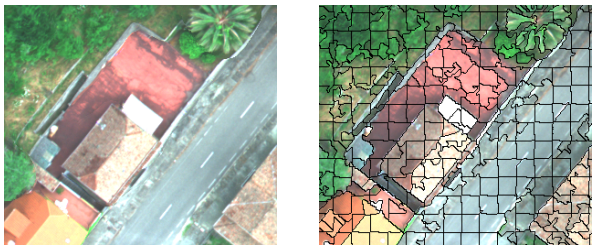
A. Métodos de segmentación basados en superpíxeles

Las imágenes MSI en las que se centra este trabajo son de resolución espacial alta, del orden de unos pocos centímetros por píxel, y cubren un área considerable, haciendo el procesado a nivel de píxel impracticable computacionalmente en un tiempo razonable. En estos casos es común realizar un preprocesado sobre las imágenes que permita extraer información espacial mediante la extracción de bordes, construcción de perfiles basados en operaciones morfológicas o de difusión, o mediante segmentación [10]. En particular, en este trabajo nos centramos en aplicar segmentación produciendo segmentos homogéneos en cuanto a etiqueta de clasificación. De este modo, es posible realizar la clasificación a nivel de segmento en lugar de a nivel de píxel, reduciendo así el coste computacional.

La técnica de segmentación que aplicamos está basada en detectar superpíxeles, que son agrupamientos de píxeles contiguos con características similares, dando lugar así a una región homogénea. En las imágenes multi e hiperespectrales, las características de un píxel las define el vector espectral de cada uno de ellos, que contiene los valores de las diferentes bandas presentes en la imagen. No es necesario que los superpíxeles tengan una forma concreta y/o regular, sino que tienen que adaptarse a la imagen para agrupar los píxeles con características similares. Para ello se varían las propiedades de compacidad de los superpíxeles, dando lugar a segmentaciones como la mostrada en la Figura 1.

En este trabajo se usará un algoritmo de segmentación basado en superpíxeles llamado Waterpixels (WP) [11], para facilitar la comparación con otras técnicas de clasificación ya publicadas, así como por su buen funcionamiento y bajo tiempo de computación [12].

En la clasificación cada superpíxel es un elemento a procesar. La entrada a la red de aprendizaje profundo asociada a cada superpíxel se denomina parche, y se obtiene mediante los siguientes pasos:



(a) Imagen de entrada (b) Segmentación resultante

Fig. 1: Resultado de una segmentación en superpíxeles.

1. Se determina el cuadrilátero mínimo que contiene al superpíxel en su interior.
2. A partir del cuadrilátero, se determina el píxel central como el punto central de dicho cuadrilátero.
3. Se selecciona como parche una región de $N \times N$ píxeles centrada en el píxel central cuya clase asociada es la de dicho píxel.

B. Métodos de clasificación de imágenes de teledetección basados en aprendizaje profundo

Como se ha comentado anteriormente, en la clasificación de imágenes multi e hiperespectrales la arquitectura neuronal más popular es la red neuronal convolucional –CNN– [13].

Una CNN consiste en una sucesión de pares de capas convolucionales y de agrupamiento (*pooling*), para posteriormente aplanar todas las características extraídas e introducirlas en una red densamente conectada, que se encarga de realizar la clasificación propiamente dicha. Cada capa convolucional aplica una serie de convoluciones (filtros) a la información de entrada para extraer distintos tipos de características de la misma. Cada capa de agrupamiento reduce la dimensionalidad espacial de las características extraídas por las capas convolucionales anteriores y mantiene las más relevantes, para mejorar así la capacidad de generalización. Este segundo tipo de capa divide el espacio de características en regiones no solapadas para después extraer un único valor representativo de cada una de ellas. En la Figura 2 se muestra un ejemplo de las tres variantes de capas de agrupamiento más utilizadas. Al combinar de forma reiterada los dos tipos de capas mencionados, una CNN comienza extrayendo características que pueden resultar más evidentes sobre la imagen para posteriormente extraer características de un nivel de abstracción mayor.

Así, cuanto más profunda es una CNN, características más complejas puede extraer de los datos de entrada. Sin embargo, no suele ser suficiente con introducir más capas a la red, sino que hay que buscar estructuras más complejas para un correcto funcionamiento, ya que si se introducen más capas arbitrariamente puede aparecer el problema del desvanecimiento del gradiente, que impediría que el entrenamiento de las redes se realice de manera eficaz [7]. A partir de esta dificultad ha surgido el concepto de aprendizaje residual, que introduce "atajos" entre capas, de forma que como salida de una capa se tiene la información proce-

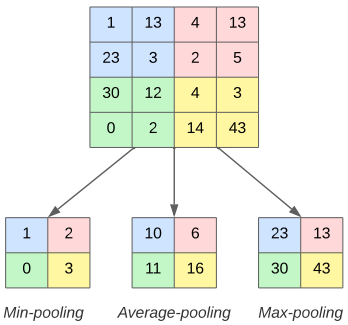


Fig. 2: Aplicación de capas de agrupamiento sobre una entrada de dos dimensiones. Los distintos colores indican las distintas regiones no solapadas sobre las que se aplican los agrupamientos.

sada por la misma y también información disponible con anterioridad a esta, facilitando así el aprendizaje de la red. En la Figura 3 se muestra la arquitectura de un bloque residual. Las arquitecturas neuronales que siguen este principio se denominan redes residuales –ResNet– [5][14].

Desde que la arquitectura ResNet fue introducida, la comunidad investigadora ha tratado de mantener las precisiones de las clasificaciones disminuyendo el coste computacional. Surgen así las MobileNets [8], diseñadas para su ejecución en dispositivos móviles y caracterizadas por el uso de un tipo especial de convolución llamada "convolución separable en profundidad" –Depthwise Separable Convolution (DWSConv)– [15]. Esta reduce la cantidad de operaciones computacionales requeridas, resultando así en un modelo más rápido y eficiente sin perjudicar las precisiones.

A partir de las redes MobileNets y las operaciones DWSConv se ha propuesto una familia de arquitecturas neuronales llamadas EfficientNets [9], caracterizadas por realizar un escalado uniforme de las tres dimensiones que tiene una CNN (profundidad, anchura y resolución). El objetivo es tener una arquitectura que sea adaptable en coste computacional pero que mantenga precisiones competitivas. En la bibliografía se muestra cómo logran una muy buena precisión en la clasificación de imágenes RGB [16][17] con menor coste computacional que las ResNets. Mostraremos en este artículo los resultados para imágenes multiespectrales.

III. Técnica de clasificación propuesta

A. *EfficientNet: escalado uniforme de una CNN*

La técnica de clasificación que proponemos está basada en una red tipo EfficientNet [9], cuyas propiedades se describen a continuación.

La EfficientNet es una CNN que se caracteriza por realizar un escalado uniforme de las tres dimensiones de las CNNs a través de un coeficiente compuesto, adaptándose así a las limitaciones computacionales y/o temporales impuestas al mismo tiempo que se obtiene un esquema de clasificación con precisiones competitivas. Dichas dimensiones son anchura, profundidad y resolución, que se corresponden con el número de canales por capa convolucional, el número de capas de la red, y el tamaño de las muestras de entrada, respectivamente.

El coeficiente compuesto relaciona la proporcionalidad de las tres dimensiones, definiendo ciertos valores de partida para dichas proporciones. Estos valores se corresponden a la denominada línea base de EfficientNet (EfficientNet-B0), a partir de la cual, según se vaya incrementando el coeficiente compuesto, se crean las redes de mayor tamaño EfficientNet-B1 a EfficientNet-B8.

El bloque principal de las EfficientNets es el bloque de cuello de botella invertido móvil –Mobile Inverted Bottleneck (MBConv)– [18]. Este bloque consta de dos fases:

1. **Expansión a través de una convolución de un punto:** el bloque comienza con una capa convolucional con filtro de tamaño 1×1 llamada convolución de un punto –Pointwise Convolution (PWConv)–. El objetivo de esta convolución es aumentar la dimensionalidad de entrada, proyectándola en un espacio de alta dimensionalidad para capturar representaciones más complejas. Para determinar el avance de dicha expansión se usa un factor de expansión e, siendo el número de canales de salida el número de canales de entrada al bloque multiplicado por e.
2. **Convolución separable en profundidad (DWSConv):** el resultado de la fase anterior pasa después por una convolución separable en profundidad, que, como ya se ha comentado, reduce significativamente el coste computacional en comparación con una convolución tradicional al evitar las interacciones intensivas entre canales. Esta operación consta a su vez de dos partes:
   a) Convolución en profundidad –Depthwise Convolution (DWConv)–: aplica un filtro de convolución separado a cada canal de entrada, identificando así patrones espaciales de forma independiente a cada canal, preservando el número de canales pero reduciendo las dimensiones espaciales.
   b) Reducción a través de una convolución de un punto: es la última operación del bloque y se aplica para aprender representaciones más complejas combinando la información espectral de los canales de la operación anterior, que fueron tratados de forma independiente. A su vez, este bloque reduce el número de canales de salida para el procesado en capas posteriores.

En las EfficientNets, los bloques MBConv incluyen además un bloque Squeeze-and-Excitation (SE) [19], que consta también de dos fases:

1. ***Squeeze*:** en esta fase se realiza una operación de agrupación global para reducir espacialmente las dimensiones de los mapas de características. Esto significa que se obtiene un valor representativo de cada mapa de características como el resumen global de todas las características presentes en él. Cabe comentar que no es necesario que la

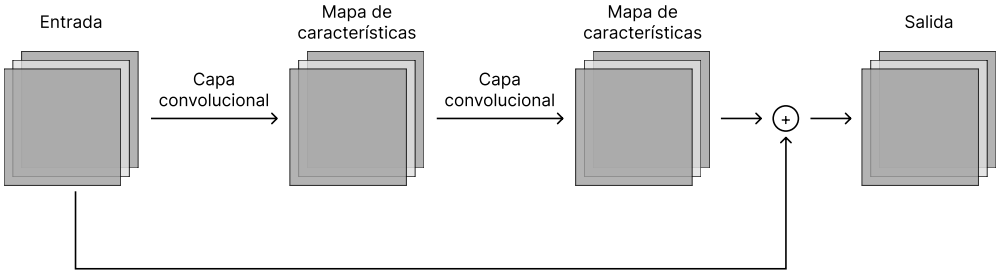


Fig. 3: Estructura de un bloque residual.

reducción se haga sobre todos los canales, sino que se puede hacer solamente sobre un porcentaje de estos, definido como tasa de reducción *se*.

2. ***Excitation*:** se usan capas densamente conectadas para aprender los pesos que se aplicarán a las características obtenidas en la fase anterior. Estos pesos se usan para recalibrar las características en función de su importancia relativa.

Se muestra en la Figura 4 como sería un bloque SE, y en la Tabla I como va evolucionando la dimensionalidad a lo largo del bloque MBConv con SE integrado, que es el usado en las EfficientNets.

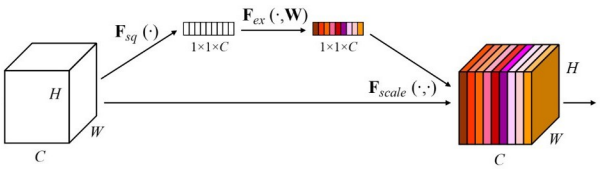


Fig. 4: Estructura de un bloque SE [19]: la operación $\mathbf{F}_{sq}$ se corresponde a la fase *squeeze*; y la operación $\mathbf{F}_{ex}$, a la fase *excitation*. Posteriormente se realiza un reescalado ($\mathbf{F}_{scale}$) de cada mapa de características, multiplicando cada uno por el valor correspondiente de la salida de $\mathbf{F}_{ex}$ para dicho mapa. H, W y C indican el alto, ancho y número de canales de la muestra, respectivamente.

Tabla I: Arquitectura del bloque MBConv usado en EfficientNet, de K a K' canales con altura H y anchura W, y con desplazamiento $s_H \times s_W$, factor de expansión e, y tasa de reducción $se \in [0, 1]$.

#	Operador	Dim. entrada	Dim. saída
1	PWConv (expansión)	$K@H \times W$	$(e \cdot K)@H \times W$
2	DWConv	$(e \cdot K)@H \times W$	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$
3.1	Bloque SE *Squeeze*	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$	$(se \cdot K)@1 \times 1$
3.2	Bloque SE *Excitation*	$(se \cdot K)@1 \times 1$	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$
4	PWConv (reducción)	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$	$K'@\frac{H}{s_H} \times \frac{W}{s_W}$

Por un lado, al usar el bloque MBConv, las CNNs logran alta eficiencia computacional ya que este bloque reduce el número de cálculos en comparación con las convoluciones tradicionales. El bloque MBConv logra esto mediante el uso de las DWSConvs, al mismo tiempo que permite la extracción de características significativas al combinar la información de diferentes canales y localizaciones espaciales.

Por otro lado, al integrar el bloque SE, las CNNs aprenden a enfocarse en las características más relevantes y atenuar las que menos lo son, a la vez que no se añade prácticamente coste computacional. En las EfficientNets, estos bloques tienen una tasa de reducción de canales del 25 %, lo que quiere decir que después de la fase *squeeze* se conservará el 25 % de los canales de entrada, descartando el resto.

Una vez presentado el bloque principal de la EfficientNet, se muestra en la Tabla II la arquitectura detallada de su configuración básica, denominada línea base (B0).

Tabla II: Arquitectura de la EfficientNet-B0. Las dimensiones son las correspondientes a una entrada inicial con dimensiones $H \times W \times B = 32 \times 32 \times 5$. C indica el número de clases a clasificar.

#	Operador	# capas	Dim. salida	Tam. filtro	Desplazamiento
1	Convolución 2D	1	$32@16 \times 16$	3×3	2×2
2	MBConv $e = 1$, $se = 0{,}25$	1	$16@16 \times 16$	3×3	1×1
3	MBConv $e = 6$, $se = 0{,}25$	2	$24@8 \times 8$	3×3	2×2
4	MBConv $e = 6$, $se = 0{,}25$	2	$40@4 \times 4$	5×5	2×2
5	MBConv $e = 6$, $se = 0{,}25$	3	$80@2 \times 2$	3×3	2×2
6	MBConv $e = 6$, $se = 0{,}25$	3	$112@2 \times 2$	5×5	1×1
7	MBConv $e = 6$, $se = 0{,}25$	4	$192@1 \times 1$	5×5	2×2
8	MBConv $e = 6$, $se = 0{,}25$	1	$320@1 \times 1$	3×3	1×1
	Convolución 2D		$1280@1 \times 1$	1×1	1×1
9	*Pooling*	1	$1280@1 \times 1$	-	-
	Fully connected		$C \times 1$	-	-

B. *Escalado de la EfficientNet*

Las EfficientNets fueron creadas pensando en probarse contra conjuntos de datos muchísimo más grandes que los usados en teledetección, como ImageNet [20]. Cuando operan sobre entradas de menor tamaño es probable que la gran complejidad del modelo pueda llevar a un sobreajuste, no generalizando bien y, por lo tanto, no obteniendo valores altos en precisión de la clasificación. Para evitarlo, a partir de la versión B0 proponemos crear modelos más pequeños. Para obtenerlos, propondremos a continuación diversas modificaciones para reducir aún más la complejidad de dicha versión.

La primera modificación es una eliminación de capas de la red. Esta eliminación se justifica porque, como se puede ver en la Tabla II, a partir de la etapa 7 las convoluciones no extraen características espaciales, pues la dimensión de entrada es 1×1. Es esta la dimensión de entrada porque se ha elegido un tamaño de parche de 32×32 como entrada de la red. Así entonces, se eliminan las etapas 2, 3 y 4 para que de esta forma las convoluciones finales puedan extraer características espaciales de la entrada. La arquitectura de la red resultante tras esta eliminación se muestra detalladamente en la Tabla III y gráficamente en la Figura 5.

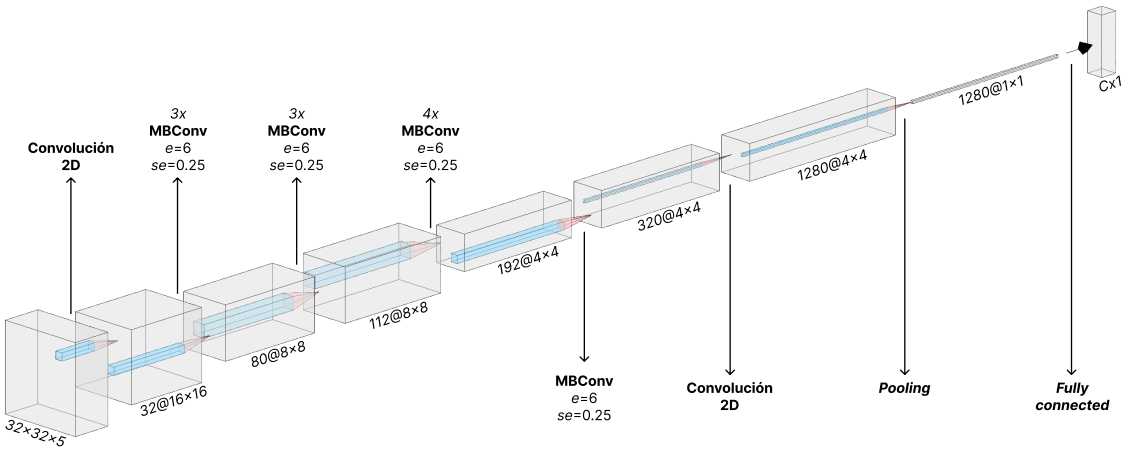


Fig. 5: Estructura de la versión reducida de la red EfficientNet-B0. Los detalles se muestran en la Tabla III.

Tabla III: Arquitectura de la versión reducida de la EfficientNet-B0. Las dimensiones son las correspondientes a una entrada inicial con dimensiones $H \times W \times B = 32 \times 32 \times 5$. C indica el número de clases a clasificar.

#	Operador	# capas	Dim. salida	Tam. filtro	Desplazamiento
1	Convolución 2D	1	$32@16 \times 16$	3×3	2×2
2	MBConv $e = 6$, $se = 0{,}25$	3	$80@8 \times 8$	3×3	2×2
3	MBConv $e = 6$, $se = 0{,}25$	3	$112@8 \times 8$	5×5	1×1
4	MBConv $e = 6$, $se = 0{,}25$	4	$192@4 \times 4$	5×5	2×2
5	MBConv $e = 6$, $se = 0{,}25$	1	$320@4 \times 4$	3×3	1×1
	Convolución 2D		$1280@4 \times 4$	1×1	1×1
6	*Pooling*	1	$1280@1 \times 1$	-	-
	Fully connected		$C \times 1$	-	-

C. *Manipulando la EfficientNet reducida*

A partir de la versión reducida de la EfficientNet-B0 (Tabla III) que proponemos en el subapartado anterior, hay numerosos parámetros que se pueden modificar. Para mejorar la eficiencia computacional proponemos centrarnos en los bloques MBConv. Los parámetros modificados son los siguientes:

- **Número de repeticiones de cada etapa:** en la Tabla III se muestra una columna que indica el número de capas de cada etapa (**# capas**). Teniendo en cuenta que la versión reducida tiene para las etapas con bloques MBConv un número de 3, 3, 4 y 1 capas, respectivamente, se van a disminuir dichos números por si todavía se está produciendo sobreajuste debido a la complejidad del modelo con respecto al tamaño del conjunto de datos que se clasifica. Esta disminución se hará respetando la relación de número de capas entre las distintas etapas de los bloques MBConv, de forma que el número de capas será r_1, r_1, r_2 y 1, respectivamente, tal que $r_1 \leq r_2$.
- **Factor de expansión *e*:** como ya se ha comentado anteriormente, este factor es el que indica en que proporción se multiplican los canales de entrada dentro del bloque MBConv, tal y como se puede ver en la Tabla I. Teniendo en cuenta que la versión reducida tiene $e = 6$ y que la intención es reducir, se va a disminuir dicho factor de expansión. Para respetar la relación, e será el mismo para las cuatro etapas de bloques MBConv.
- **Tasa de reducción *se*:** como ya se ha comentado, cada bloque MBConv integra un bloque SE con tasa de reducción del 25 %. Para variar este porcentaje respetando la relación entre las distintas etapas, *se* será el mismo para las cuatro etapas de bloques MBConv.

De esta forma se pueden crear muchos esquemas de clasificación a partir de la versión reducida de la EfficientNet-B0, que tendrían la arquitectura mostrada en la Tabla IV. Cabe señalar que, para ajustarse a la de la EfficientNet-B0, los valores por defecto de los parámetros modificables son $(r_1, r_2, x, y) = (3, 4, 6, 0{,}25)$. A partir de la variación de estos parámetros se pueden obtener distintas arquitecturas de EfficientNet, que serán las que se probarán a posteriori.

Tabla IV: Arquitectura de la versión reducida de la EfficientNet-B0 con parámetros modificables r_1, r_2, x e y. Las dimensiones son las correspondientes a una entrada inicial con dimensiones $H \times W \times B = 32 \times 32 \times 5$. C indica el número de clases a clasificar.

#	Operador	# capas	Dim. salida	Tam. filtro	Desplazamiento
1	Convolución 2D	1	$32@16 \times 16$	3×3	2×2
2	MBConv $e = x$, $se = y$	r_1	$80@8 \times 8$	3×3	2×2
3	MBConv $e = x$, $se = y$	r_1	$112@8 \times 8$	5×5	1×1
4	MBConv $e = x$, $se = y$	r_2	$192@4 \times 4$	5×5	2×2
5	MBConv $e = x$, $se = y$	1	$320@4 \times 4$	3×3	1×1
	Convolución 2D		$1280@4 \times 4$	1×1	1×1
6	*Pooling*	1	$1280@1 \times 1$	-	-
	Fully connected		$C \times 1$	-	-

D. *EfficientNet V2: modelos más pequeños y rápidos*

Apenas dos años más tarde, los mismos autores de la EfficientNet han propuesto una segunda versión de la misma: la EfficientNet V2 [21]. Esta red se caracteriza por la reducción de parámetros respecto a la primera versión, manteniendo aun así las precisiones. La principal diferencia de la arquitectura de esta red es la modificación del bloque principal MBConv, pasando a ser un bloque MBConv fundido (*Fused* MBConv), introducido en [22]. Se muestra en la Figura 6 una comparativa de ambos bloques.

El bloque *Fused* MBConv se diferencia del bloque MBConv en la sustitución de las dos primeras fases (PWConv y DWConv) por una convolución 2D estándar, quedando la arquitectura de este bloque tal y como se detalla en la Tabla V. Pese a lo que se ha

comentado en el Subapartado III-A, la convolución 2D estándar puede resultar más rápida que PWConv + DWConv en tiempo de ejecución debido al mejor aprovechamiento de la localidad espacial de los datos procesados, de forma que el sobrecoste de la operación estándar queda anulado por dicha ventaja. Se espera que la mayor complejidad de esta operación, en cuanto a número de parámetros se refiere, no lleve al modelo a caer en sobreajuste, comprobándose esto en las pruebas experimentales correspondientes.

Tabla V: Arquitectura del bloque *Fused* MBConv usado en la EfficientNet V2, de K a K' canales con altura H y anchura W, y con desplazamiento $s_H \times s_W$, factor de expansión e, y tasa de reducción $se \in [0, 1]$.

#	Operador	Dim. entrada	Dim. salida
1	Convolución 2D (expansión)	$K@H \times W$	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$
2.1	Bloque SE *Squeeze*	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$	$(se \cdot K)@1 \times 1$
2.2	Bloque SE *Excitation*	$(se \cdot K)@1 \times 1$	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$
3	PWConv (reducción)	$(e \cdot K)@\frac{H}{s_H} \times \frac{W}{s_W}$	$K'@\frac{H}{s_H} \times \frac{W}{s_W}$

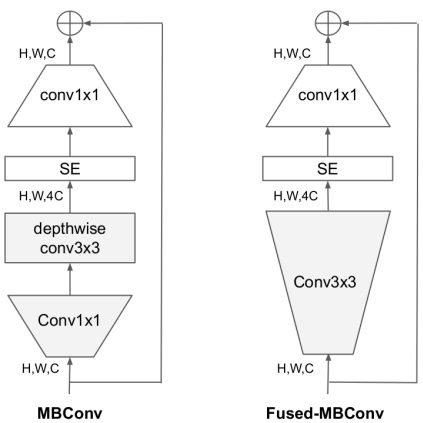


Fig. 6: Estructura de los bloques MBConv (izquierda) y *Fused* MBConv (derecha), con factor de expansión $e = 4$ [21]. Los detalles de los bloques están en las Tablas I y V, respectivamente.

E. Reduciendo la EfficientNet V2

Una vez introducida la EfficientNet V2 y siguiendo la filosofía del Subapartado III-C, proponemos la creación de dos esquemas de clasificación con parámetros modificables a partir de esta red. Partiendo de la arquitectura de parámetros modificables y de la aproximación de [21] sobre la proporcionalidad del uso entre bloques MBConv y *Fused* MBConv, las dos arquitecturas creadas serán: una con la mitad de los bloques *fused*, y otra con todos ellos.

Los detalles de dichas arquitecturas están en la Tabla VI, cuyos parámetros modificables son los mismos que los explicados en el Subapartado III-C. Cabe comentar que, para ajustarse a la EfficientNet-B0, los valores por defecto de los parámetros modificables son $(r_1, r_2, x, y) = (3, 4, 6, 0{,}25)$.

IV. Resultados experimentales

Para evaluar los esquemas de clasificación propuestos y otros a modo de comparación, se usarán conjuntos de datos de imágenes multiespectrales de cuencas de ríos gallegos, creados con el objetivo de monitorizar la interacción de masas de vegetación nativa con estructuras artificiales y especies invasoras.

En concreto, estos son ocho imágenes de muy alta resolución multiespectral que se han usado en [12]. Estas imágenes han sido capturadas en 2018, 2019 y 2020 volando un vehículo aéreo no tripulado a 120 m de altitud sobre varias cuencas de ríos en Galicia, resultando en una resolución espacial de 10 cm/px. El vehículo llevaba una cámara multiespectral MicaSense RedEdge-MX, capturando cinco bandas correspondientes a las longitudes de onda de 475 nm (azul), 560 nm (verde), 668 nm (rojo), 717 nm (*rededge*), y 840 nm (infrarrojo cercano).

En la Figura 7 se puede observar la imagen en color compuesto correspondiente a una de las ocho imágenes junto con su información de referencia, y un mapa de clasificación resultante. La Tabla VII enumera las diez clases identificables en la información de referencia, detallando el número de muestras en cada conjunto de datos. Estas clases van desde vegetación nativa a estructuras hechas por el humano, como carreteras o construcciones. Cabe destacar los grandes desbalances que hay en todos los conjuntos de datos entre el número de muestras de las distintas clases, existiendo, a veces, diferencias de varios órdenes de magnitud. Estos desbalances introducen un sesgo hacia las clases mayoritarias que podría impedir una precisión de clasificación equilibrada entre las distintas clases.

Todos los conjuntos de datos han sido segmentados en superpíxeles usando el algoritmo WP con un tamaño medio de 400 px/superpíxel, con un mínimo de 100 px/superpíxel, y usando un factor de compacidad de 0.5 puntos, siguiendo el enfoque de [12]. Los parches extraídos tienen una dimensión espacial de $N \times N = 32 \times 32$ px. Adicionalmente, todos los datos han sido normalizados en el rango [-1,1].

Una vez presentados los conjuntos de datos, se presentan a continuación las métricas que se usarán para evaluar el desempeño de los distintos esquemas de clasificación, tanto en términos de precisión de la clasificación como de rendimiento computacional.

En cuanto a las métricas de precisión, se usarán las siguientes tres métricas, estándares en la clasificación en teledetección:

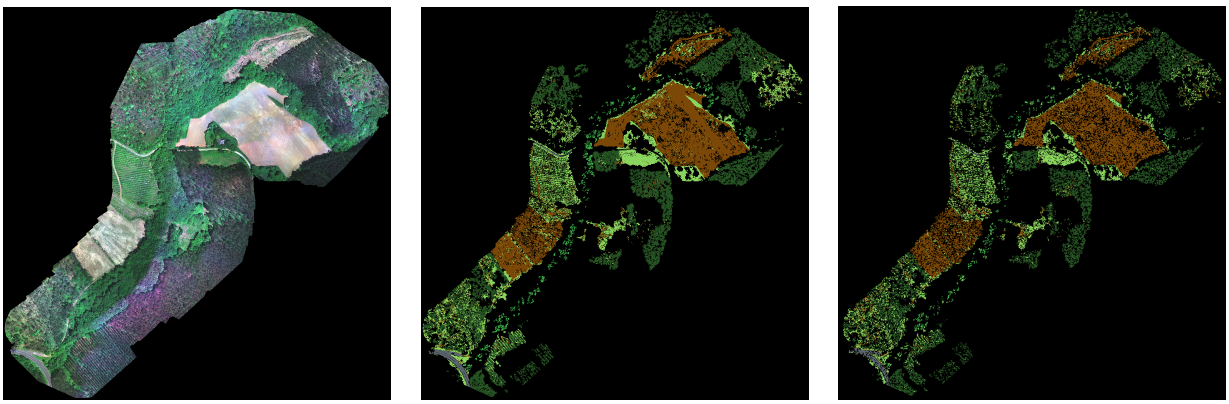
- Overall Accuracy (OA): porcentaje total de píxeles correctamente clasificados.
- Average Accuracy (AA): media de los porcentajes de píxeles correctamente clasificados dentro de cada clase.
- Coeficiente kappa de Cohen (κ): porcentaje total de píxeles correctamente clasificados corregido tras tener en cuenta cuantos aciertos se pueden producir por mero azar.

Para las tres métricas anteriores, cuanto mayor sea el porcentaje, mejor es la métrica, pues el esquema clasifica con más precisión.

En cuanto a las métricas de rendimiento, el entre-

Tabla VI: Arquitecturas de las versiones reducidas de EfficientNet V2 con parámetros modificables r_1, r_2, x e y. La versión 1 (izquierda de las etapas 4 y 5) tiene dos bloques *Fused* MBConv, mientras que la versión 2 (derecha de las etapas 4 y 5) tiene cuatro. Las dimensiones son las correspondientes a una entrada inicial con dimensiones $H \times W \times B = 32 \times 32 \times 5$. C indica el número de clases a clasificar.

#	Operador	# capas	Dim. salida	Tam. filtro	Desplazamiento
1	Convolución 2D	1	$24@16 \times 16$	3×3	2×2
2	*Fused* MBConv $e = x,\ se = y$	r_1	$80@8 \times 8$	3×3	2×2
3	*Fused* MBConv $e = x,\ se = y$	r_1	$112@8 \times 8$	5×5	1×1
4	MBConv $e = x,\ se = y$ | *Fused* MBConv $e = x,\ se = y$	r_2	$192@4 \times 4$	5×5	2×2
5	MBConv $e = x,\ se = y$ | *Fused* MBConv $e = x,\ se = y$	1	$320@4 \times 4$	3×3	1×1
	Convolución 2D		$1280@4 \times 4$	1×1	1×1
6	*Pooling*	1	$1280@1 \times 1$	-	-
	Fully connected		$C \times 1$	-	-



(a) Imagen de entrada (b) Información de referencia (c) Mapa de clasificación

Fig. 7: Imagen de entrada en color compuesto (7a), información de referencia (7b) y mapa de clasificación resultante con una red EfficientNet-B0 (7c) obtenidos a partir de una imagen MSI de 5 bandas de la cuenca del río Ferreiras, con 10 clases distintas y con una resolución de 9 219 × 9 335 px. Las clases correspondientes a cada color en 7b y 7c están descritas en la Tabla VII.

Tabla VII: Clases identificables de los conjuntos de datos. El número de muestras está en superpíxeles, obtenido a través de segmentación con WP.

#	Color	Clase	Río de las Mestas	Embalse de Eiras	Riachuelo Ermidas	Río Ferreiras	Río Mera	Río Oitavén	Río Ulla	Cuenca del Xesta
1		Agua	0	3 557	890	0	0	1 515	0	488
2		Tierra	7 076	867	874	24 377	18 535	978	495	411
3		Piedras	0	2 923	1 846	0	0	749	0	4 632
4		Asfalto	0	340	3 597	685	395	202	32	700
5		Hormigón	0	182	126	0	10	774	0	0
6		Tejados	0	47	608	0	18	448	0	0
7		Prados	23 417	5 722	13 052	22 364	43 191	9 376	494	52 276
8		Árboles nativos	2 973	13 466	4 559	4 830	13 241	11 300	2 243	23 711
9		Pinos	0	687	1 527	14	0	2 109	284	0
10		Eucaliptos	23 592	74	8 984	30 586	49 752	8 387	0	0
		Muestras totales:	57 058	27 865	36 063	82 856	125 142	35 838	3 548	82 218

namiento de los esquemas será evaluado en cuanto al tiempo de ejecución requerido. Concretamente:

- Se medirá el tiempo de entrenamiento por época – Training Time Per Epoch (TTPE)–. Este valor es el resultado del tiempo que tarda el esquema de clasificación en entrenarse dividido por las épocas de entrenamiento realizadas. Cuanto menor sea el tiempo, mejor es la métrica, pues el esquema tarda menos tiempo en entrenarse.
- Se determinarán las aceleraciones (*speedups*) experimentadas en el entrenamiento respecto a un clasificador base preestablecido. Esta métrica indica cuantas veces un tiempo optimizado t_o es más rápido que un tiempo base t_b, tal que $speedup = \frac{t_b}{t_o}$. Cuanto mayor sea la aceleración, mejor es la métrica, pues el esquema optimizado es rápido en mayor medida que el esquema base.

El equipo usado para la realización de los experimentos tiene un procesador Intel Core i5-8400, 48 GB de RAM, y una tarjeta gráfica NVIDIA GeForce RTX 3070 Ti con 8 GB de VRAM. En cuanto al software, el equipo tiene el sistema operativo Ubuntu 20.04.6 LTS junto con Docker 23.0.6. Para la ejecución de los experimentos se ha levantado en el equipo un contenedor Docker basado en la imagen de NVIDIA para PyTorch 22.04, que tiene sistema operativo Ubuntu 20.04.4 LTS, Python 3.8.13, y CUDA 11.6.2 con cuDNN 8.4.0.27.

El lenguaje de programación a usar será Python, sobre el que conviene destacar los siguientes paquetes principales: PyTorch 1.12.0, para la creación, entrenamiento y prueba de los esquemas de clasificación; NumPy 1.22.3, para la manipulación de los conjuntos de datos; scikit-learn 0.24.2, para el preprocesado de los conjuntos de datos y la obtención de κ; torchinfo 1.7.2, para la obtención de la información detallada de la arquitectura de los esquemas; y Guild AI 0.8.1, para el registro de los experimentos.

Durante la experimentación, las EfficientNets van a ser comparadas con dos esquemas de clasificación mencionados en [23], debido a que se ha trabajado con los mismos conjuntos de datos. Estos esquemas son una CNN simple, con cuatro bloques convolucionales y dos densamente conectados, y una ResNet profunda, compuesta por tres capas residuales seguidas de una capa de agrupamiento y otra densamente conectada. La CNN representa las primeras arquitecturas de aprendizaje profundo usadas para clasificación de imágenes multi e hiperespectrales, mientras que la ResNet constituye el "diseño por defecto" después de las CNNs simples para este tipo de problema.

Todos los conjuntos de datos usados han sido segmentados usando el algoritmo WP con los parámetros mencionados anteriormente. Una vez realizada la segmentación, se selecciona el centro de cada segmento como el píxel de referencia de este, a partir del cual se elige el parche, que será la entrada de la red. Para todos los conjuntos de datos se usará un subconjunto de entrenamiento del 15 % y un subconjunto de validación del 5 %, este último para monitorizar el progreso del entrenamiento con el objetivo de identificar problemas potenciales tales como el sobreajuste. De esta forma, queda para el subconjunto de prueba el 80 % restante. Por último, y como ya se ha comentado también anteriormente, todas las muestras se normalizarán en el rango $[-1, 1]$ para una comparación más rigurosa con los resultados obtenidos en [23].

Con el objetivo de analizar el desempeño de las diferentes EfficientNets propuestas en el Apartado III se van a proponer 38 entrenamientos sobre cada uno de los 8 conjuntos de datos, detallados a continuación:

1. 9 con cada una de las líneas base de EfficientNet (Subapartado III-A), que van desde la B0 a la B8.
2. 23 a partir de la EfficientNet reducida con parámetros modificables (Subapartado III-C), en los que se usarán los valores por defecto, se variará independientemente cada parámetro, y se probarán dos de las configuraciones más prometedoras a partir de los resultados de los entrenamientos anteriores.
3. 6 con las versiones reducidas de la EfficientNet V2 (Subapartado III-E). Se entrenarán, para cada una de las dos versiones, la configuración con los valores de los parámetros modificables por defecto y las dos combinaciones consideradas como las más prometedoras en el punto 2 de esta enumeración.

Todos los entrenamientos seguirán un proceso iterativo de 100 épocas, suministrando en cada una de ellas las muestras de entrenamiento en lotes de 100. Un parámetro relevante de la red es la tasa de aprendizaje, que controla la medida en la que se actualizan los parámetros internos de la red en el proceso de entrenamiento. El valor de este parámetro es crucial, ya que en caso de no asignar uno ideal puede que la red no sea capaz de converger. En estos entrenamientos se ha comenzado con una tasa de aprendizaje de 0.001, que se irá reduciendo mediante un mecanismo que evitará el estancamiento, siendo este una parte del propuesto en [24]. Además, como algoritmo de optimización de la red se ha usado el más que conocido Adaptive Moment Estimation (Adam) [25], y como función de pérdida, la entropía cruzada.

Con el objetivo de comparar las EfficientNets, se han reejecutado los entrenamientos de los otros esquemas de clasificación (CNN y ResNet) para obtener las métricas de rendimiento, ajustando el número de épocas y el tamaño del lote al de los entrenamientos de las EfficientNets.

Por simplificación se mostrarán los resultados medios tras realizar los experimentos con los ocho conjuntos de datos, y solamente las configuraciones que han sido relevantes en el desarrollo de la investigación.

La Tabla VIII muestra las distintas métricas presentadas, tanto de precisión como de rendimiento, para varias configuraciones relevantes a lo largo de la investigación de este trabajo. El guion (-) indica que el parámetro de la posición correspondiente toma su valor por defecto, siendo estos valores los siguientes: $(r_1, r_2, x, y) = (3, 4, 6, 0{,}25)$. Los valores de *speedup* se han obtenido a través de la fórmula presentada anteriormente, siendo t_0 el TTPE del esquema de clasificación de la fila actual y t_b el TTPE de la ResNet, ya que, de los mencionados aquí de los de [23], es la CNN que mejores métricas de precisión obtiene.

En lo que a las líneas base de la EfficientNet V1 se refiere, se puede observar en los resultados como según se va aumentando la complejidad de la red (de la EfficientNet-B0 a la EfficientNet-B8), las métricas de precisión empeoran debido al sobreajuste consecuencia directa de este aumento de complejidad, además de que solo se usa el 15 % de las muestras de los conjuntos de datos para entrenamiento y hay grandes desbalances entre clases, y a mayor complejidad de red se necesitaría mayor cantidad de datos.

Al crear la versión reducida de la EfficientNet-B0 se puede apreciar como tanto las métricas de precisión como las de rendimiento mejoran, comportamiento esperado bajo la hipótesis expuesta, disminuyendo la complejidad del esquema de clasificación y desapareciendo así el sobreajuste al conjunto de entrenamiento. En esta versión reducida y tras realizar toda la experimentación, se ha concluido que las dos configuraciones de parámetros mostradas en la tabla además de la de los valores por defecto son las mejores, pues han mejorado considerablemente las métricas de rendimiento sin apenas empeorar las de precisión. De entre estas dos, se puede decir que la mejor configuración es la

Tabla VIII: Resultados medios de algunos de los esquemas de clasificación mencionados y propuestos tras realizar todas las pruebas experimentales sobre los ocho conjuntos de datos descritos. El mejor valor de cada una de las métricas de los esquemas de clasificación basados en EfficientNet aparece resaltado.

Esquema de clasificación			OA (%)	AA (%)	κ (%)	TTPE (s)	*Speedup*
CNN [23]			$92{,}17 \pm 0{,}40$	$73{,}83 \pm 1{,}62$	$86{,}47 \pm 0{,}67$	1,54	1,83×
ResNet [23]			$94{,}61 \pm 0{,}26$	$83{,}00 \pm 1{,}25$	$90{,}86 \pm 0{,}44$	2,82	-
EfficientNet V1	B0 (Tabla II) [9]		$85{,}51 \pm 0{,}48$	$63{,}83 \pm 1{,}36$	$73{,}85 \pm 0{,}69$	2,98	0,95×
	B8 [9]		$82{,}02 \pm 3{,}26$	$54{,}78 \pm 6{,}16$	$65{,}40 \pm 7{,}69$	14,21	0,20×
	Versión reducida de la B0 con parámetros (r_1, r_2, x, y) (Tabla IV)	$(-,-,-,-)$	$88{,}00 \pm 1{,}29$	$70{,}74 \pm 1{,}86$	$78{,}29 \pm 3{,}09$	3,37	0,84×
		$(1,1,-,0)$	$87{,}97 \pm 2{,}95$	$72{,}32 \pm 2{,}07$	$81{,}37 \pm 3{,}61$	1,14	2,47×
		$(1,1,5,0)$	$84{,}97 \pm 2{,}66$	$70{,}75 \pm 2{,}14$	$77{,}87 \pm 3{,}16$	1,01	2,79×
EfficientNet V2	Versión reducida 1 con parámetros (r_1, r_2, x, y) (Tabla VI)	$(-,-,-,-)$	$92{,}77 \pm 0{,}68$	$80{,}04 \pm 1{,}95$	$88{,}54 \pm 1{,}12$	2,74	1,03×
		$(1,1,-,0)$	$92{,}67 \pm 0{,}35$	$80{,}29 \pm 2{,}12$	$88{,}36 \pm 0{,}62$	0,89	3,16×
		$(1,1,5,0)$	$92{,}47 \pm 0{,}39$	$78{,}65 \pm 2{,}56$	$88{,}01 \pm 0{,}62$	0,81	3,50×
	Versión reducida 2 con parámetros (r_1, r_2, x, y) (Tabla VI)	$(-,-,-,-)$	$94{,}05 \pm 0{,}28$	$83{,}06 \pm 2{,}52$	$90{,}66 \pm 0{,}45$	3,41	0,83×
		$(1,1,-,0)$	$93{,}40 \pm 0{,}37$	$83{,}60 \pm 0{,}98$	$89{,}62 \pm 0{,}57$	0,90	3,13×
		$(1,1,5,0)$	$93{,}17 \pm 0{,}45$	$83{,}25 \pm 1{,}64$	$89{,}24 \pm 0{,}74$	0,81	3,49×

versión reducida de la B0 con parámetros $(1,1,-,0)$, ya que alcanza el segundo mejor TTPE con métricas de precisión muy competitivas respecto a la otra configuración. Esta otra configuración con parámetros $(1,1,5,0)$ alcanza un TTPE ligeramente mejor, pero a costa de empeorar las métricas de precisión en torno a 3 puntos porcentuales, por lo que no se podría considerar que este sea el mejor esquema, aunque tenga el menor TTPE de esta versión reducida.

Centrándonos ahora en las versiones reducidas de la EfficientNet V2, se puede observar como la versión reducida 2 obtiene mejores métricas de precisión que la versión reducida 1, demostrando así que, sobre los conjuntos de datos de este trabajo, funcionan mejor los bloques *Fused* MBConv en todas las etapas del esquema de clasificación (ver arquitectura del bloque en la Tabla V). Dentro de esta versión reducida 2, el esquema con mejores métricas de rendimiento es el correspondiente a los parámetros $(1,1,5,0)$, pero tiene unas métricas de precisión sutilmente peores que la configuración con parámetros $(1,1,-,0)$, la cual a su vez tiene también un TTPE ligeramente peor. De esta forma se llega a una solución de compromiso entre preferir una mejora sutil en métricas de precisión o una ligera reducción en el TTPE.

Observando los resultados conjuntamente, se puede concluir que los esquemas de clasificación basados en EfficientNet V2 son mejores, tanto en términos de precisión como de rendimiento, que los basados en EfficientNet V1. Concretamente, el mejor esquema de EfficientNet V2 obtiene unas métricas de precisión con unos 6, 11 y 8 puntos porcentuales más que el mejor esquema de EfficientNet V1, en OA, AA y κ, respectivamente. Además, este esquema de EfficientNet V2 disminuye el TTPE del esquema de EfficientNet V1 en casi un 30 % (1,14 s $\rightarrow$ 0,81s). También se puede observar claramente como las mejores versiones de EfficientNet obtienen unas métricas de precisión muy similares a las de la ResNet, pero con un menor TTPE, siendo estas versiones hasta $3,49$ veces más rápidas que la ResNet según el TTPE.

V. Conclusiones

En este trabajo se propone una técnica de clasificación para imágenes MSI de alta resolución basada en una etapa inicial de segmentación en superpíxeles mediante el algoritmo WP seguida por una etapa de clasificación mediante CNNs tipo EfficientNet. Se han realizado modificaciones sobre diferentes arquitecturas de este tipo de redes y se han adaptado para su operación eficiente sobre conjuntos de datos de teledetección multiespectrales de cuencas de ríos gallegos. Posteriormente se ha realizado una comparación de las configuraciones propuestas con otros esquemas de aprendizaje profundo usados para la clasificación MSI.

Con todo esto, las principales contribuciones de este trabajo son las siguientes:

1. Se ha propuesto una técnica de clasificación para imágenes multiespectrales de alta resolución basada en una segmentación en superpíxeles mediante el algoritmo Waterpixels seguida de la aplicación de una red neuronal convolucional tipo EfficientNet para realizar la clasificación. Ello ha exigido adaptar el esquema de EfficientNet para imágenes multiespectrales.
2. Se ha comprobado experimentalmente que las arquitecturas basadas en EfficientNet son demasiado complejas para los conjuntos de datos de teledetección usados (imágenes multiespectrales de 5 bandas y resolución de 10 cm/px), siendo necesario reducirlas para conseguir un aprendizaje adecuado.
3. Se han propuesto diferentes variantes de la arquitectura EfficientNet reduciendo sus dimensiones y modificando su configuración. En particular, se han reducido el número de repeticiones, el factor de expansión y la tasa de reducción de cada etapa, y se han combinado estas tres modificaciones. También se ha propuesto cambiar el bloque principal MBConv a *Fused* MBConv.
4. Se ha comprobado experimentalmente como las mejores configuraciones de EfficientNet igualan en métricas de precisión a la red estándar en cla-

sificación de imágenes multiespectrales ResNet, reduciendo además el coste computacional con respecto a esta.

En cuanto a posibles ampliaciones, se han abierto tres vías a explorar:

- Analizar diferentes configuraciones de EfficientNet para el problema bajo estudio: inicialización de pesos, funciones de activación, funciones de pérdida, técnicas de adaptación de la tasa de aprendizaje, distintas arquitecturas, y otros parámetros.
- Integrar las EfficientNet en arquitecturas más complejas de clasificación MSI, con el objetivo de mantener la alta calidad de la clasificación en cuanto a precisiones obtenidas pero con un menor coste computacional.
- Definir una cadena de procesado para clasificación MSI que pueda ser ejecutada a bordo del dron que captura las imágenes, para conseguir así procesado cercano al tiempo real mediante el uso de plataformas computacionales con prestaciones limitadas, por ejemplo, en consumo de potencia.

Agradecimientos

Este trabajo ha sido financiado en parte por la Agencia Estatal de Investigación, Gobierno de España, a través de los contratos PID2019–104834GB–I00, PID2022–141623NB–I00 y TED2021–130367B–I00 financiados por MCIN/AEI/10.13039/501100011033 y por los fondos European Union NextGenerationEU/PRTR; en parte por la Consellería de Cultura, Educación, Formación Profesional e Universidades, Xunta de Galicia, a través de la ayuda de acreditación de Centro de Investigación de Galicia ED431G 2019/04, y acreditación de Grupo de Investigación competitivo ED431C 2022/16; todos ellos están cofinanciados por el Fondo Europeo de Desarrollo Regional (FEDER).

Referencias

[1] D Chutia, D K Bhattacharyya, K K Sarma, R Kalita, and S Sudhakar, "Hyperspectral remote sensing classifications: A perspective survey," *Transactions in GIS*, vol. 20, no. 4, pp. 463–490, 2016.

[2] M.E. Paoletti, J.M. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 158, pp. 279–317, 2019.

[3] Shutao Li, Weiwei Song, Leyuan Fang, Yushi Chen, Pedram Ghamisi, and Jón Atli Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6690–6709, Sep. 2019.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2015.

[6] Waseem Rawat and Zenghui Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[7] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 196–207, 2020.

[8] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017.

[9] Mingxing Tan and Quoc V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2020.

[10] Harmanpreet Kaur, "Review of remote sensing image segmentation techniques," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 4, no. 4, pp. 1667–1674, 2015.

[11] Vaïa Machairas, Matthieu Faessel, David Cárdenas-Peña, Théodore Chabardes, Thomas Walter, and Etienne Decencière, "Waterpixels," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3707–3716, 2015.

[12] Francisco Argüello, Dora B. Heras, Alberto S. Garea, and Pablo Quesada-Barriuso, "Watershed monitoring in Galicia from UAV multispectral imagery using advanced texture methods," *Remote Sensing*, vol. 13, no. 14, 2021.

[13] Shiqi Yu, Sen Jia, and Chunyan Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, 2017.

[14] Zilong Zhong, Jonathan Li, Lingfei Ma, Han Jiang, and He Zhao, "Deep residual networks for hyperspectral image classification," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 1824–1827.

[15] François Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017.

[16] Pimpisa Charoenchittang, Petarpa Boonserm, Kazuki Kobayashi, and Nagul Cooharojananone, "Airport buildings classification through remote sensing images using EfficientNet," in *2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2021, pp. 127–130.

[17] Haikel Alhichri, Asma S. Alswayed, Yakoub Bazi, Nassim Ammour, and Naif A. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE Access*, vol. 9, pp. 14078–14094, 2021.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2019.

[19] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu, "Squeeze-and-Excitation networks," 2019.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2015.

[21] Mingxing Tan and Quoc V. Le, "EfficientNetV2: Smaller models and faster training," 2021.

[22] Suyog Gupta and Mingxing Tan, "EfficientNet-EdgeTPU: Creating accelerator-optimized neural networks with AutoML," en línea [fecha de consulta: 9 enero 2024].

[23] Álvaro G. Dieste, Francisco Argüello, and Dora B. Heras, "ResBaGAN: A residual balancing GAN with data augmentation for forest mapping," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 1–21, 2023.

[24] Ilya Loshchilov and Frank Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2017.

[25] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.