Coste computacional de una red de aprendizaje profundo basada en EfficientNet sobre dispositivos basados en GPU para computación en el borde

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

Resumen— **El despliegue de redes de aprendizaje profundo para la teledetección en entornos de computación en el borde sigue siendo un gran reto, ya que estas plataformas operan bajo restricciones de energía y de recursos de cómputo. Este desafío exige aplicar técnicas de optimización de redes y realizar un trabajo especialmente cuidadoso en el caso de redes diseñadas para operar con un coste computacional reducido. El problema se agrava en el caso de redes de tamaño medio, ya que la mayoría de las técnicas de optimización están orientadas a redes de gran escala. En este trabajo se analizan el coste computacional y la eficiencia energética generados por técnicas de optimización aplicadas a la red EffBaGAN, una red de tamaño medio basada en EfficientNet, orientada a la clasificación de imágenes multiespectrales de alta resolución espacial para la cobertura terrestre en escenarios de escasez y desequilibrio de datos. Se evalúa el impacto de la precisión mixta automática (AMP), la cuantización durante y después del entrenamiento (QAT y PTQ, respectivamente) y la poda, tanto de forma aislada como combinada, en una plataforma NVIDIA Jetson AGX Orin. Además, se analiza el rendimiento computacional en función de los límites de potencia del dispositivo. Los resultados revelan que la combinación del entrenamiento sensible a la cuantización (QAT) con la poda representa el enfoque más efectivo para lograr reducciones conjuntas en el tamaño de la red y en el número de parámetros y FLOPs, al mismo tiempo que se mantienen métricas de precisión competitivas.**

Palabras clave— **clasificación de imágenes, teledetección, computación en el borde, coste computacional, eficiencia energética, precisión mixta automática, cuantización, poda**

I. Introducción

En el ámbito de la teledetección, uno de los problemas fundamentales es la clasificación de imágenes, una tarea que consiste en asignar una etiqueta o clase a cada píxel para identificar diferentes tipos de elementos en problemas de cobertura terrestre como, por ejemplo, ecosistemas vegetales [1]. En la actualidad, las redes de aprendizaje profundo han desplazado a los algoritmos clásicos de aprendizaje automático para la clasificación, superando sus métricas de precisión. No obstante, la mejora algorítmica conlleva una penalización: una alta exigencia de recursos computacionales y la necesidad de grandes cantidades de datos etiquetados para su entrenamiento adecuado [2]. Por otro lado, en aplicaciones reales, como el etiquetado de componentes vegetales y artificiales en imágenes de cobertura terrestre de teledetección, la obtención de datos etiquetados resulta costosa y compleja, ya que se trata de zonas de difícil acceso y, dependiendo de la resolución espacial, las imágenes pueden representar elementos de más de una clase en un mismo píxel [3]. Esto da lugar a escenarios de escasez de datos y desequilibrio de clases, donde las redes neuronales tradicionales sufren de sobreajuste, ya que aprenden a representar los patrones y características espaciales y espectrales presentes en los datos de entrenamiento en lugar de patrones generales [4]. La red EffBaGAN (Efficient Balancing Generative Adversarial Network) [5] aborda estos dos problemas al integrar una arquitectura generativa para sintetizar muestras artificiales y estabilizar el aprendizaje. Además, esta red ha sido concebida bajo la premisa de reducir el coste computacional, heredando la alta eficiencia de su arquitectura base EfficientNet [6].

A pesar de estas optimizaciones a nivel algorítmico, el despliegue de redes para teledetección en entornos de computación en el borde para su uso en tiempo cercano al real sigue siendo un reto [7]. Las plataformas computacionales heterogéneas basadas en GPU para la computación en el borde son muy comunes actualmente para ejecutar algoritmos de aprendizaje profundo en teledetección [8]. No obstante, estas operan bajo estrictas restricciones de energía, de memoria y de cómputo. Para facilitar la ejecución de redes neuronales complejas en este tipo de hardware, resulta indispensable aplicar técnicas de optimización del coste computacional [9]. Entre estas estrategias, se encuentran en la literatura las siguientes: la precisión mixta automática [10], que combina operaciones en punto flotante de 32 bits (FP32) con operaciones en 16 bits (FP16) para aliviar el uso del ancho de banda de memoria; la cuantización, que proyecta los pesos y activaciones en representaciones enteras de menor precisión (INT8), pudiendo aplicarse simulando el ruido de cuantización durante el entrenamiento [11], o tras finalizarlo [12]; y la poda [13], que transforma la red eliminando conexiones no representativas o grupos enteros de parámetros, como canales o filtros, reduciendo así el número de operaciones necesarias.

[1]Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, España. E-mails: {nicolas.vilela.perez, dora.blanco}@usc.gal.

[2]Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, 15782, Santiago de Compostela, España. E-mail: francisco.arguello@usc.gal.

También existen otras técnicas de mejora enfocadas exclusivamente en el entrenamiento, como LoRA (*Low-Rank Adaptation*) [14], orientadas a la adaptación eficiente de redes preentrenadas de gran escala, lo que reduce drásticamente los requerimientos de memoria y tiempo en esta fase. Sin embargo, dado que estas técnicas no alteran la topología de la red, esta permanece completamente activa durante la inferencia, por lo que no ofrecen ninguna reducción en los costes de despliegue. Por consiguiente, estos enfoques difieren de la compresión estructural de arquitecturas diseñadas desde cero, que es el enfoque de este trabajo.

Más allá de la compresión estructural y algorítmica, el despliegue efectivo de redes exige herramientas orientadas a maximizar la eficiencia en la etapa de inferencia en el hardware de despliegue. Para ello, es una práctica estándar exportar las redes a un formato de representación intermedia interoperable como ONNX (*Open Neural Network Exchange*), lo que permite desacoplar la arquitectura de la red de la sobrecarga del *framework* de entrenamiento [15]. Posteriormente, se emplean motores de inferencia especializados como NVIDIA TensorRT, que actúan como compiladores para GPUs de computación en el borde [16]. Este motor aplica transformaciones a nivel de grafo computacional y realiza un autoajuste para seleccionar las implementaciones a bajo nivel más eficientes según la arquitectura específica de las tarjetas gráficas de destino. Estudios recientes demuestran empíricamente que el uso de estas optimizaciones en plataformas de alto rendimiento, como la NVIDIA Jetson AGX Orin, mejora la velocidad de inferencia y la eficiencia energética frente a otras alternativas [17].

En este trabajo se analiza el coste computacional de aplicar diferentes técnicas de mejora aplicadas a la red de aprendizaje profundo de bajo coste computacional EffBaGAN para la clasificación de imágenes multiespectrales de alta resolución espacial en un dispositivo basado en GPU para computación en el borde. Presentamos un análisis del coste computacional de las diferentes versiones de red optimizada en un dispositivo basado en GPU para computación en el borde: una NVIDIA Jetson AGX Orin. Asimismo, se realiza una evaluación de la eficiencia energética al variar el límite máximo de potencia. Las principales contribuciones de este trabajo son las siguientes:

- **Reducción de coste computacional:** se aplican distintas técnicas de reducción de coste computacional, como la precisión mixta automática, la cuantización durante y después del entrenamiento, y la poda.
- **Análisis del impacto de los distintos perfiles de potencia:** se evalúa cómo escala la red optimizada bajo diferentes restricciones de potencia configurables en el dispositivo, alterando la frecuencia y el número de núcleos activos de la CPU/GPU.

II. Red EffBaGAN para clasificación

Este trabajo analiza la ejecución sobre un dispositivo basado en GPU para computación en el borde de la red EffBaGAN (*Efficient Balancing Generative Adversarial Network*) [5], una red generativa específicamente diseñada para resolver tareas de clasificación en condiciones de escasez de datos y desequilibrio de clases de forma eficiente.

Para abordar la falta de datos, EffBaGAN se basa en redes generativas adversarias (GANs). Una GAN consiste en dos redes neuronales: un generador, que aprende a crear muestras sintéticas realistas a partir de ruido (o de un espacio latente), y un discriminador, que intenta distinguir entre las muestras reales y las generadas. Este proceso iterativo permite generar nuevas muestras sintéticas de alta calidad que enriquecen y equilibran el conjunto de entrenamiento, evitando así el sobreajuste del clasificador. En concreto, EffBaGAN se apoya en una variante denominada red generativa adversaria balanceada (BAGAN) [18], diseñada para estabilizar el aprendizaje en clases minoritarias.

A pesar de su capacidad para estabilizar el aprendizaje, las arquitecturas GAN conllevan una alta demanda de recursos computacionales. Por ello, a diferencia de los enfoques convencionales, el diseño de EffBaGAN ha tenido el bajo coste computacional como criterio guía, combinando las ventajas de BAGAN con la eficiencia computacional de la arquitectura EfficientNet [6]. La esencia de este bajo coste computacional radica en el uso de bloques de cuello de botella invertido móviles (MBConv) como unidad principal de procesamiento del discriminador. Estos bloques, a diferencia de las convoluciones tradicionales, que operan simultáneamente en las dimensiones espaciales y de canal, emplean convoluciones separables en profundidad. Esta técnica desacopla la extracción de características espaciales de la combinación de canales, reduciendo drásticamente las interacciones intensivas y minimizando el coste global de la red.

La figura 1 muestra la arquitectura de EffBaGAN. Para el entrenamiento, la red consta de tres pasos:

1. Primero, se entrena el *autoencoder* sin considerar la clase para aprender una representación compacta de los datos que captura su estructura y distribución subyacentes. Un *autoencoder* [19] es una red que a través de un codificador (*encoder*) y de un decodificador (*decoder*) genera vectores de características comprimidas de las muestras de entrada. Concretamente, el codificador reduce la dimensionalidad de las imágenes originales mapeándolas a un espacio latente de menor tamaño, extrayendo patrones más representativos, mientras que el decodificador toma estos vectores comprimidos y reconstruye la imagen original, ajustando sus pesos para minimizar el error de reconstrucción.
2. Después, se transfieren los parámetros del *autoencoder* que permiten reconstruir las muestras

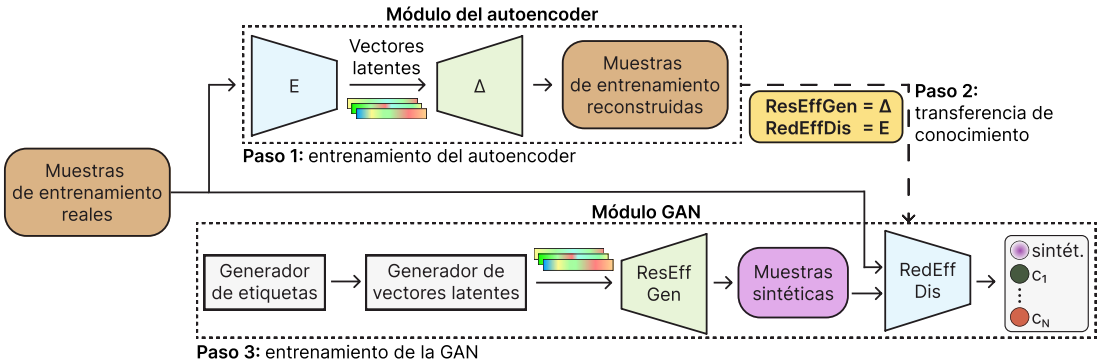

Fig. 1: Arquitectura neuronal de la red EffBaGAN [5].

originales al módulo GAN, adquiriendo este así el conocimiento del *autoencoder* y reduciendo el tiempo de entrenamiento, ya que al partir de información no aleatoria sobre los parámetros, la calidad del entrenamiento del módulo GAN mejora.

3. Por último, se entrena el módulo GAN propiamente dicho. El módulo GAN es el bloque principal de la arquitectura de la red, estando formado por dos componentes: ResEffGen, un generador que debido a la introducción de un camino residual basado en EfficientNet sintetiza muestras de alta calidad; y RedEffDis, un discriminador reducido basado en la versión EfficientNet-B0.

El coste computacional de EffBaGAN es, de partida, bajo, debido a su propia arquitectura. De las dos versiones existentes de EffBaGAN, se elige para este trabajo la de menor coste computacional: EffBaGAN-Small. A pesar de su bajo coste computacional, que no supera los 5 millones de parámetros, los recursos limitados disponibles en plataformas como la NVIDIA Jetson estudiada en este trabajo siguen haciendo necesaria la aplicación de técnicas de mejora del rendimiento.

III. Técnicas de reducción de coste computacional

Aunque la arquitectura de EffBaGAN ya es eficiente, como se explicó en la sección anterior, las restricciones de los dispositivos de computación en el borde requieren optimizaciones adicionales a nivel de software. En este trabajo se evalúa el impacto individual y combinado de las siguientes técnicas para la reducción del coste computacional de la red:

- **Precisión mixta automática** (*Automatic Mixed Precision*, AMP) [10]: esta técnica optimiza el uso de memoria durante el entrenamiento combinando dinámicamente operaciones en precisión simple (FP32), necesarias para garantizar la estabilidad numérica, con operaciones en precisión media (FP16), orientadas a reducir el uso del ancho de banda de la memoria.
- **Entrenamiento sensible a la cuantización** (*Quantization-Aware Training*, QAT) [11]: consiste en reducir la precisión de los pesos y las activaciones de la red a enteros de 8 bits (INT8) simulando los efectos de la cuantización mediante la inyección de ruido durante el entrenamiento. Esto permite que la red se adapte gradualmente a representaciones de menor precisión, logrando una mayor robustez, aunque a costa de aumentar el tiempo de entrenamiento y el uso de memoria en la GPU.
- **Cuantización posentrenamiento** (*Post Training Quantization*, PTQ) [12]: al igual que QAT, proyecta los pesos y las activaciones de la red a una menor precisión (INT8), pero se aplica una vez finalizado el entrenamiento. Su objetivo principal es comprimir la red, acelerar el tiempo de inferencia y reducir el consumo de energía en dispositivos de borde. Es un proceso rápido que requiere un paso de calibración con datos representativos, pero no sobrecarga el tiempo de entrenamiento ni modifica los pesos aprendidos.
- **Poda** [13]: transforma la red densa en una arquitectura más compacta mediante la eliminación permanente de conexiones consideradas redundantes. En este caso se aplica una poda estructurada de filtros en las capas convolucionales, utilizando un criterio basado en la magnitud para eliminar el 25 % de los canales menos significativos, tras lo cual se aplica una etapa de *fine-tuning* para tratar de conseguir la máxima precisión posible en la clasificación en comparación con la red original. Su objetivo principal es reducir el número de parámetros y de operaciones (FLOPs) de la red. Cabe comentar que este tipo de poda se realiza en anchura, al eliminar filtros y no capas completas, por lo que la red resultante tendrá el mismo número de capas y, en estas, menos filtros. Dado que se mantienen las dependencias entre capas y el número de capas, no se esperan mejoras en tiempo de ejecución notables respecto a una poda en profundidad [20].

La siguiente sección evalúa el impacto de aplicar estas técnicas, analizando su viabilidad y rendimiento en el dispositivo de computación en el borde, tanto de forma aislada como combinada.

IV. Experimentación

En esta sección presentaremos inicialmente el contexto experimental y el conjunto de imágenes multi-

espectrales de cobertura terrestre sobre el que se han realizado los experimentos. A continuación mostraremos los resultados de aplicar diferentes técnicas de optimización a la red EffBaGAN así como los resultados bajo diferentes restricciones de potencia.

A. Configuración experimental

Para evaluar el comportamiento de EffBaGAN sobre un dispositivo basado en GPU para computación en el borde se han usado tres conjuntos de datos multiespectrales de alta resolución espacial (10 cm/px) de cuencas de ríos en Galicia, usados anteriormente en [21]. En la tabla I se detallan las 10 clases presentes en la información de referencia de estos tres conjuntos de datos. Las imágenes que componen los conjuntos de datos han sido capturadas con un vehículo aéreo no tripulado a 120 m de altitud equipado con una cámara multiespectral MicaSense RedEdge-MX, que captura 5 bandas correspondientes a las longitudes de onda de 475 nm (azul), 560 nm (verde), 668 nm (rojo), 717 nm (*red-edge*) y 840 nm (infrarrojo cercano).

Tabla I: Número de muestras para cada conjunto de datos expresado en número de superpíxeles (obtenidos mediante el algoritmo de segmentación Waterpixels).

#	Color	Clases	Eiras Dam	Oitavén River	Xesta Basin
1		Agua	1 929	912	210
2		Tierra	424	484	182
3		Piedras	726	349	2 170
4		Asfalto	233	128	467
5		Hormigón	111	458	0
6		Tejados	29	294	0
7		Prados	2 989	6 896	42 821
8		Árboles nativos	8 604	7 573	15 543
9		Pinos	351	955	0
10		Eucaliptos	39	3 599	0
		Muestras totales:	15 435	21 648	61 393

En la figura 2 se puede ver la imagen en color RGB del conjunto de datos Eiras Dam, así como el detalle de una porción de dicha imagen, incluyendo la información de referencia correspondiente, que indica en diferentes colores los elementos presentes. Las zonas negras corresponden a regiones en las que la información de referencia no es concluyente.

Todos los conjuntos de datos han sido segmentados en superpíxeles usando el algoritmo Waterpixels con un tamaño medio de 400 px/superpíxel, un tamaño mínimo de 100 px/superpíxel y un factor de compacidad de 0,5 puntos, tal y como se describe en [21]. La entrada de la red son los parches centrados, cada uno en un superpíxel, con una dimensión espacial de 32×32 px. Adicionalmente, todos los datos han sido normalizados al rango $[-1, 1]$. Para cada conjunto de datos, el 15 % de las muestras corresponden al conjunto de entrenamiento, y el 5 %, al conjunto de validación, siendo el 80 % restante para el conjunto de test.

Para analizar el efecto de las técnicas evaluaremos: precisión de clasificación, coste computacional y eficiencia energética. La precisión se evalúa mediante tres métricas estándar a nivel de píxel: la precisión global (OA), la precisión media por clase (AA) y el coeficiente kappa de Cohen (κ). Los tiempos de ejecución registrados son el tiempo de entrenamiento por época (TEPE) y el tiempo total de test (inferencia). Otras métricas de coste computacional son el pico de memoria requerida por la GPU durante el entrenamiento, el tamaño de la red en disco, el número de parámetros entrenables y el número de millones de operaciones de coma flotante (MFLOPs), siendo estas tres últimas métricas tomadas durante la fase de inferencia y, en consecuencia, solo usando el discriminador. Finalmente, para evaluar la eficiencia energética, se registra la potencia media y se estima la energía total consumida tanto en la fase de entrenamiento como en la de test. Con el fin de garantizar la significancia estadística de los resultados, cada configuración experimental se ha ejecutado de forma independiente un total de 5 veces, reportando el valor medio y la desviación típica de todas las métricas.

Para la ejecución de los experimentos se ha utilizado un kit de desarrollo NVIDIA Jetson AGX Orin, un sistema en chip representativo de los dispositivos basados en GPU para computación en el borde, diseñado para operar bajo restricciones de tamaño, peso y potencia. En la tabla II se presentan las especificaciones técnicas de dicho sistema.

Tabla II: Especificaciones del kit de desarrollo NVIDIA Jetson AGX Orin.

Componente	Especificación
CPU	ARM Cortex-A78AE v8.2 de 12 núcleos
GPU	NVIDIA de arquitectura Ampere
Memoria RAM	32 GB LPDDR5 (unificada)
Almacenamiento	64 GB eMMC 5.1
Potencia	Configurable entre 15 W y 60 W

En cuanto al software, el sistema opera sobre el *software stack* NVIDIA JetPack 6.0, que incluye CUDA 12.2 y cuDNN 8.9.4. El desarrollo y la experimentación se han realizado en un entorno de Conda con Python 3.10.19, pudiéndose resaltar los siguientes paquetes: PyTorch 2.4.0, para el entrenamiento de la red; TensorRT 8.6.2, para la inferencia de alto rendimiento; y Guild AI 0.9.0, para registrar los experimentos.

B. Resultados experimentales

B.1 Impacto de las técnicas de reducción de coste computacional

En esta fase de la experimentación se evalúa el impacto de las diferentes técnicas de reducción del coste de cómputo sobre la red EffBaGAN-Small ejecutada en la GPU, utilizando la configuración de máxima potencia del dispositivo. Los resultados de precisión de clasificación y coste computacional se pueden ver en las tablas III y IV.

El análisis individual de las técnicas muestra que AMP logra reducir el pico de memoria requerida por la GPU durante el entrenamiento en un 27,96 %, así

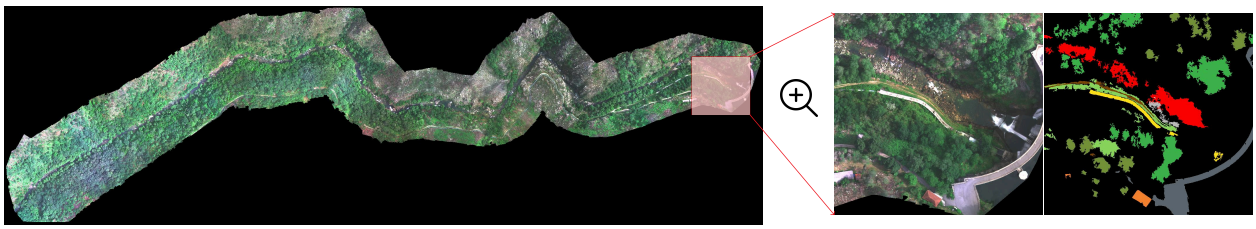

Fig. 2: Imagen en color RGB y ampliación de la misma junto con su información de referencia del conjunto de datos Eiras Dam usado en este trabajo, con una resolución de 18 221 × 5 176 px. La clase correspondiente a cada color de la información de referencia está descrita en la tabla I.

Tabla III: Métricas de precisión y tiempos de ejecución para todos los conjuntos de datos con diferentes técnicas de reducción del coste computacional aplicadas a EffBaGAN-Small en el dispositivo de computación en el borde NVIDIA Jetson AGX Orin.

Técnica	**OA** (%)	**AA** (%)	κ (%)	**TEPE** (s)	**Tiempo de test** (s)
Ninguna	97,32 ± 0,35	90,00 ± 3,10	95,50 ± 0,61	9,47 ± 0,02	265,07 ± 5,00
AMP	97,19 ± 0,36	89,41 ± 2,39	95,31 ± 0,59	9,02 ± 0,03	266,98 ± 4,66
PTQ	96,30 ± 0,70	87,84 ± 2,42	93,83 ± 1,10	9,47 ± 0,02	263,82 ± 6,28
QAT	97,19 ± 0,35	90,75 ± 1,34	95,33 ± 0,57	16,16 ± 0,05	263,43 ± 3,40
Poda	97,63 ± 0,11	91,03 ± 1,73	96,03 ± 0,18	9,48 ± 0,02	263,83 ± 5,18
PTQ + Poda	53,03 ± 23,02	44,58 ± 11,52	36,46 ± 19,72	9,47 ± 0,02	264,25 ± 2,78
QAT + Poda	88,56 ± 11,39	81,06 ± 12,22	84,44 ± 12,68	9,46 ± 0,03	264,45 ± 2,71

Tabla IV: Métricas de coste computacional para todos los conjuntos de datos al aplicar diferentes técnicas a EffBaGAN-Small en el dispositivo de computación en el borde NVIDIA Jetson AGX Orin.

Técnica	**Pico mem. GPU** (MiB)		**Tamaño red** (MiB)		**Parámetros** (#)		**MFLOPs** (#)	
Ninguna	1 314,01		3,78		972 235		27,10	
AMP	946,62	(↓ 27,96 %)	3,78	(=)	972 235	(=)	27,10	(=)
PTQ	1 315,19	(↑ 0,09 %)	1,10	(↓ 70,88 %)	972 235	(=)	26,62	(↓ 1,79 %)
QAT	1 425,14	(↑ 8,46 %)	1,10	(↓ 70,88 %)	972 235	(=)	26,62	(↓ 1,79 %)
Poda	1 315,51	(↑ 0,11 %)	3,22	(↓ 14,99 %)	677 315	(↓ 30,33 %)	18,34	(↓ 32,34 %)
PTQ + Poda	1 314,43	(↑ 0,03 %)	1,34	(↓ 64,60 %)	675 443	(↓ 30,53 %)	17,94	(↓ 33,81 %)
QAT + Poda	1 316,43	(↑ 0,18 %)	1,36	(↓ 63,94 %)	677 315	(↓ 30,33 %)	17,94	(↓ 33,81 %)

Los valores porcentuales indican la diferencia relativa respecto de la configuración sin aplicar ninguna técnica. Todas las métricas de coste computacional registradas, excepto el pico de memoria requerida por la GPU durante el entrenamiento, corresponden a la fase de inferencia; por lo tanto, solo se considera el discriminador al calcularlas.

como disminuir el TEPE de 9,47 s a 9,02 s, siendo un 4,75 % más rápido. Por su parte, ambas técnicas de cuantización (PTQ y QAT) consiguen comprimir el tamaño de la red en disco en más de un 70 %, mientras que la poda logra eliminar más de un 30 % de los parámetros y MFLOPs manteniendo las métricas de precisión estables.

Al evaluar las combinaciones de técnicas de cuantización con poda, resulta evidente la necesidad de un enfoque algorítmico cuidadoso para evitar caídas notables en las métricas de precisión. La combinación de PTQ con poda provoca una caída significativa en las métricas de precisión. Sin embargo, la integración de QAT con poda demuestra ser una estrategia con mayor robustez: consigue aplicar reducciones en el coste computacional de un 63,94 %, 30,33 % y 33,81 % en el tamaño de la red, número de parámetros y número de MFLOPs, respectivamente, a la vez que logra mitigar la caída de precisión en comparación con la combinación con PTQ. Debido a este equilibrio entre precisión y coste, la versión optimizada con QAT y poda se selecciona para el análisis de eficiencia del dispositivo que se realizará a continuación.

B.2 Análisis de perfiles de potencia

Una vez optimizado el coste computacional a nivel de software, en esta sección evaluamos cómo se comporta la red en términos de eficiencia energética. La arquitectura NVIDIA Jetson AGX Orin permite configurar dinámicamente su límite de potencia, ajustando las frecuencias de reloj y los núcleos activos de la CPU y de la GPU.

Para este análisis, se ha ejecutado la versión más eficiente de la red optimizada, la red optimizada con QAT y poda, bajo distintos perfiles de potencia: sin restricciones (MAXN), y con máximos fijados a 50 W, 30 W y 15 W. En la tabla V se muestran los correspondientes tiempos de ejecución y métricas de eficiencia energética de la red optimizada sobre el conjunto de datos Eiras Dam. Debe tenerse en cuenta que las métricas de precisión de la clasificación no se ven afectadas por la modificación de los perfiles de potencia.

Como se puede observar, los perfiles de mayor potencia máxima dan lugar a los tiempos de ejecución más bajos. Por el contrario, los perfiles más restrictivos en términos energéticos penalizan los tiempos de ejecución. Esto demuestra que, para la computación

Tabla V: Tiempos de ejecución y métricas de eficiencia energética, variando el perfil de potencia, para EffBaGAN-Small optimizada con QAT y poda (QAT + Poda) sobre el conjunto de datos Eiras Dam, en el dispositivo de computación en el borde NVIDIA Jetson AGX Orin.

Perfil de potencia	**TEPE** (s)		**Tiempo de test** (s)		**Potencia** (W) Entrenamiento		**Potencia** (W) Test		**Energía** (Wh) Entrenamiento		**Energía** (Wh) Test	
MAXN	4,73		228,18		36,25		15,54		28,59		0,98	
50 W	6,60	(↑ 39,51 %)	300,78	(↑ 31,82 %)	23,83	(↓ 34,27 %)	12,74	(↓ 18,04 %)	26,22	(↓ 8,30 %)	1,07	(↑ 8,33 %)
30 W	11,42	(↑ 141,40 %)	271,62	(↑ 19,04 %)	18,30	(↓ 49,52 %)	13,28	(↓ 14,53 %)	34,84	(↑ 21,86 %)	1,00	(↑ 1,83 %)
15 W	21,54	(↑ 355,26 %)	407,94	(↑ 78,78 %)	12,47	(↓ 65,60 %)	10,34	(↓ 33,43 %)	44,77	(↑ 56,57 %)	1,17	(↑ 19,11 %)

Los valores porcentuales indican la diferencia relativa respecto del perfil de potencia sin restricciones (MAXN).

en el borde, limitar la potencia del hardware puede ser una estrategia eficiente, siempre y cuando se pueda permitir un aumento de los tiempos de ejecución. Para el caso estudiado, si contemplamos un escenario en el que tanto el entrenamiento como el test se ejecutan en el dispositivo en el borde, el perfil de 50 W representa el único compromiso energético viable. Esto se debe a que la energía consumida en el entrenamiento se reduce en un 8,30 %, tal como se puede observar en la tabla V, siendo esta 26,22 Wh frente a 28,59 Wh para el caso MAXN, es decir, sin restricciones. Los perfiles de potencia con restricciones más agresivas (30 W y 15 W) provocan un incremento notable en los tiempos, lo que tiene un impacto directo en la energía total consumida.

Puede resultar llamativo que el perfil de potencia limitado a 50 W tenga un mayor tiempo de test que el limitado a 30 W. Esto es debido a las restricciones de frecuencia de cada perfil: para mantener el límite de consumo, el perfil de 50 W habilita los 12 núcleos de la CPU, pero restringe su frecuencia máxima a 1,50 GHz; por el contrario, el perfil de 30 W trabaja solamente con 8 núcleos, pero a una mayor frecuencia de 1,72 GHz. Dado que la etapa de posprocesado asociada al cálculo de métricas de precisión se ejecuta en la CPU de forma secuencial, esta parte del código se ejecuta más rápido en el núcleo de mayor frecuencia disponible en el escenario del perfil de 30 W.

V. Conclusiones

En este trabajo se ha analizado la aplicación de distintas técnicas para reducir el coste computacional de la red EffBaGAN-Small, una red generativa eficiente orientada a la clasificación de imágenes de teledetección en escenarios de escasez de datos y desequilibrio entre clases. El despliegue de esta red de tamaño medio en una plataforma de computación en el borde NVIDIA Jetson AGX Orin ha permitido cuantificar el impacto real de estas optimizaciones en un hardware con restricciones computacionales.

Los resultados experimentales muestran que, aplicadas de forma aislada, las técnicas evaluadas ofrecen beneficios notables: la precisión mixta automática (AMP) reduce el pico de memoria requerida por la GPU durante el entrenamiento en un 27,96 %, las estrategias de cuantización comprimen el tamaño de la red en más de un 70 %, y la poda logra reducir en más del 30 % el número de parámetros y MFLOPs manteniendo las métricas de precisión de la clasificación. No obstante, el análisis de la interacción entre técnicas revela que la pertinencia de combinar técnicas de optimización requiere un análisis cuidadoso. Por ejemplo, la integración de la técnica de cuantización PTQ con una poda posterior provoca una caída notable en el rendimiento de la clasificación. La combinación de QAT con poda es la estrategia más robusta, lo que reduce el coste de computación sin comprometer significativamente la precisión de la clasificación, que se reduce en torno a 10 puntos porcentuales. La evaluación del comportamiento en escenarios de limitación de potencia indica que restringir el límite penaliza el tiempo de ejecución, pero puede asegurar la viabilidad de la ejecución en escenarios de disponibilidad limitada de energía, como suele ser el caso de dispositivos de computación en el borde.

Resulta pertinente analizar hasta qué punto los resultados y comportamientos expuestos en este trabajo son extrapolables a otras redes. Una de las observaciones es que, si bien técnicas como la poda logran reducir más de un 30 % los parámetros y los MFLOPs para la red, estas mejoras apenas se traducen en una reducción tangible en tiempo de ejecución. Este comportamiento se debe al tamaño de la red estudiada, ya que EffBaGAN-Small es una arquitectura compacta que cuenta con menos de un millón de parámetros en el discriminador. Para la ejecución de redes de tamaño similar, que puede considerarse de tamaño medio en el entorno de aprendizaje profundo, el tiempo de procesamiento en la GPU queda enmascarado por las sobrecargas del entorno de ejecución, ya que estamos utilizando *frameworks* de alto nivel como PyTorch. Como conclusión, podemos decir que para redes con menos de un millón de parámetros, la computación deja de ser el cuello de botella, dominada por estos costes fijos, que incluyen los movimientos de datos transparentes para el usuario, y resulta difícil percibir consecuencias directas en el tiempo de ejecución.

La decisión de aplicar las técnicas analizadas a otra red debe basarse en el tamaño y no solo en el tipo de red. En redes pequeñas o altamente optimizadas desde su diseño mediante bloques eficientes, como es el caso, la aplicación de técnicas de optimización no producirá mejoras notables en los tiempos de ejecución. Por el contrario, en el caso de redes de aprendizaje profundo con un alto número de parámetros (decenas de millones), las optimizaciones se traducirán en una reducción del tiempo de ejecución.

Agradecimientos

Este trabajo ha sido financiado por el proyecto PID2022-141623NB-I00, financiado por MCIN/AEI/10.13039/501100011033, y por la Xunta de Galicia – Consellería de Educación, Ciencia, Universidades e Formación profesional (acreditación de Centro de Investigación de Galicia 2024-2027 ED431G-2023/04 y acreditación de Grupo Competitivo de Referencia ED431C-2022/16), ambos cofinanciados por el FEDER/UE. El trabajo de Nicolás Vilela-Pérez ha sido financiado por la Xunta de Galicia – Consellería de Educación, Ciencia, Universidades e Formación profesional, a través de la ayuda ED481A-2025-015.

Referencias

[1] Enpu Yu, "Review of remote sensing image classification: Technology evolution, method innovation and future challenges," *Applied and Computational Engineering*, vol. 177, pp. 110–116, 07 2025.

[2] Neil Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso, "The Computational Limits of Deep Learning," in *Ninth Computing within Limits 2023*. LIMITS, jun 14 2023, https://limits.pubpub.org/pub/wm1lwjce.

[3] José M. Bioucas-Dias, Antonio Plaza, Nicolas Dobigeon, Mario Parente, Qian Du, Paul Gader, and Jocelyn Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[4] Shweta Sharma and Anjana Gosain, "Addressing class imbalance in remote sensing using deep learning approaches: a systematic literature review," *Evolutionary Intelligence*, vol. 18, no. 1, pp. 23, Jan 2025.

[5] Nicolás Vilela-Pérez, Dora B. Heras, and Francisco Argüello, "EffBaGAN: An efficient balancing GAN for Earth observation in data scarcity scenarios," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 18, pp. 2477–2496, 2025.

[6] Mingxing Tan and Quoc Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR.

[7] Tianyu Wang, Jinyang Guo, Bowen Zhang, Ge Yang, and Dong Li, "Deploying AI on edge: Advancement and challenges in edge intelligence," *Mathematics*, vol. 13, no. 11, 2025.

[8] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello, "Using heterogeneous computing and edge computing to accelerate anomaly detection in remotely sensed multispectral images," *The Journal of Supercomputing*, vol. 80, no. 9, pp. 12543–12563, jun 2024.

[9] Sebastián A. Cajas Ordóñez, Jaydeep Samanta, Andrés L. Suárez-Cetrulo, and Ricardo Simón Carbajo, "Intelligent edge computing and machine learning: A survey of optimization and applications," *Future Internet*, vol. 17, no. 9, 2025.

[10] Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al., "Mixed precision training," in *Int. Conf. on Learning Representation*, 2017.

[11] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[12] Raghuraman Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *CoRR*, vol. abs/1806.08342, 2018.

[13] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[15] Lukas Stäcker, Juncong Fei, Philipp Heidenreich, Frank Bonarens, Jason Rambach, Didier Stricker, and Christoph Stiller, "Deployment of deep neural networks for object detection on edge AI devices with runtime optimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1015–1022.

[16] Omais Shafi, Chinmay Rai, Rijurekha Sen, and Gayathri Ananthanarayanan, "Demystifying TensorRT: Characterizing neural network inference engine on Nvidia edge devices," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 226–237.

[17] Ishrak Jahan Ratul, Yuxiao Zhou, and Kecheng Yang, "Accelerating deep learning inference: A comparative analysis of modern acceleration frameworks," *Electronics*, vol. 14, no. 15, 2025.

[18] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi, "BAGAN: Data augmentation with balancing GAN," in *International Conference on Machine Learning*, 2018.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] Sara Elkerdawy, Mostafa Elhoushi, Abhineet Singh, Hong Zhang, and Nilanjan Ray, "To filter prune, or to layer prune, that is the question," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[21] 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.