Destilación Bloque a Bloque para un Aprendizaje Eficiente con Datos Escasos

Irene Trigueros-Lorca[2][0009−0001−8836−2481], Leonardo Concepción[2][0000−0002−4515−2038], Christian Wagner[3][0000−0002−6121−9722], Isaac Triguero[1,2][0000−0002−0150−0651] y Daniel Molina[1,2][0000−0002−4175−2204]

[1] Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada

[2] Instituto Andaluz Interuniversitario en Ciencia de Datos e Inteligencia Computacional, Granada, 18016, España

[3] Lab for Uncertainty in Data and Decision Making (LUCID), School of Computer Science, University of Nottingham, Nottingham, UK

{irenetrigueros,leonar2cp}@ugr.es

Christian.Wagner@nottingham.ac.uk

{triguero,dmolina}@decsai.ugr.es

Resumen El auge de las Redes Neuronales Convolucionales ha traído consigo modelos cada vez más grandes y costosos de ejecutar. La Destilación de Conocimiento (KD) permite obtener modelos más ligeros transfiriendo lo aprendido por una red de gran tamaño (profesor) a una más pequeña (estudiante), con la ventaja añadida de necesitar menos datos de entrenamiento. Aunque KD se aplica tradicionalmente solo al final del modelo, su rendimiento se ve limitado en escenarios con pocas instancias por clase, como los conjuntos de entrenamiento de grano fino; KD en capas intermedias, vía poco explorada, alivia este problema proporcionando una supervisión más rica. Este trabajo plantea el diseño de un estudiante basado en bloques homogéneos simples, uno por cada bloque del profesor, destilando el conocimiento entre bloques correspondientes. Los experimentos confirman que esta destilación por bloques mejora los resultados en escenarios con pocos datos. También se observa que, aunque ajustar el profesor da los mejores resultados, basta con ajustar solo el estudiante para ganar rendimiento de forma más eficiente. Este trabajo muestra que aplicar destilación en bloques intermedios es clave para construir modelos compactos sin sacrificar precisión.

Palabras clave: Aprendizaje Profundo · Destilación de Conocimiento · Redes Neuronales Convolucionales · Metodología basada en bloques.

1. Introducción y Marco Teórico

Las técnicas de Aprendizaje Profundo (*Deep Learning*, DL) [6] han transformado al Inteligencia Artificial (IA). En particular, las Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNN) se han convertido en la herramienta de referencia para el procesamiento de imágenes [25]. Las CNN pueden

adaptarse a nuevas tareas reutilizando su estructura base y ajustando solo las últimas capas, proceso que el Aprendizaje por Transferencia [23] permite acelerar. Aún así, especializarlas en dominios concretos sigue siendo costoso en términos computacionales y de datos, lo que motiva la búsqueda de modelos más ligeros.

La Destilación de Conocimiento (*Knowledge Distillation*, KD) [7] permite obtener modelos reducidos a partir de otros de mayor tamaño mediante una arquitectura profesor-estudiante [9]: el estudiante se entrena guiado por las salidas del profesor. Esta transferencia mejora notablemente el aprendizaje, sobre todo cuando los datos escasean [3,17], como ocurre en clasificación de grano fino. Aunque desde sus inicios se planteó aplicar KD en capas intermedias [8], permitiendo una destilación más granular y rica, la mayoría de propuestas la aplican solo al final [1]. Además, muchas CNN populares se construyen apilando bloques repetidos y homogéneos, dando lugar a familias de modelos con complejidad creciente.

En este trabajo proponemos un estudiante basado en bloques homogéneos simples, con el mismo número de bloques que el profesor, analizando el impacto de aplicar KD entre bloques correspondientes a distintos niveles. Frente a propuestas recientes que requieren optimizaciones complejas [16] o restricciones en los canales [14], nuestro enfoque es más flexible y evita ambas limitaciones.

Para ser más explícitos, en este trabajo abordamos las siguientes cuestiones, o preguntas de investigación (PI):

- **PI1:** ¿En qué medida una arquitectura homogénea de estudiante logra capturar, mediante KD, el conocimiento de un profesor preentrenado?
- **PI2:** ¿Es necesario el ajuste fino del modelo profesor, o puede aplicarse directamente al modelo estudiante?
- **PI3:** ¿Cuál es el enfoque más efectivo para clasificación de grano fino: aprender del profesor al final del modelo, en cada bloque, o un enfoque intermedio?

Este trabajo se organiza como sigue: en la Sección 2 se proporciona una descripción detallada de nuestra propuesta de destilación de conocimiento homogénea mediante bloques. En la Sección 3 se presentan los resultados experimentales. Finalmente, en la Sección 4 se resumen las principales conclusiones obtenidas.

2. KD usando bloques homogéneos

En esta sección proporcionamos una descripción detallada de nuestra propuesta. Comenzamos describiendo el esquema general, seguido de una explicación de la flexibilidad del tipo de KD que ofrece nuestra propuesta. A continuación describimos la arquitectura del bloque estudiante y el proceso de entrenamiento.

2.1. Esquema Global

Siguiendo las arquitecturas habituales de las CNN populares, el modelo profesor se divide en bloques de capas consecutivas con la misma arquitectura. Para

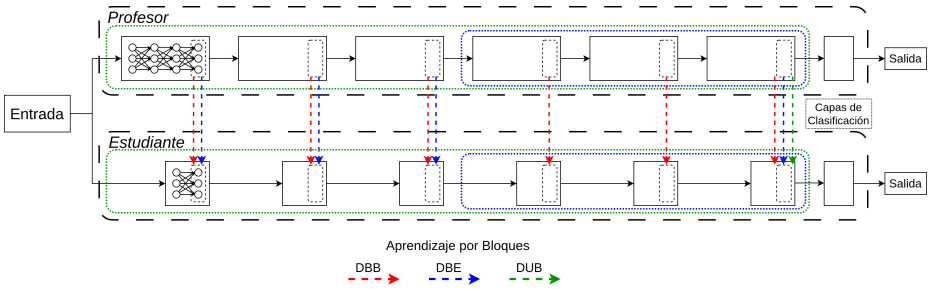


Figura 1: Diferentes niveles de KD. Marcado en rojo, **DBB**: KD se aplica a todos los bloques. En azul, **DBE**: KD se aplica a un subconjunto específico de bloques (1, 2, 3 y 6). En verde, **DUB**: KD se aplica únicamente al último bloque.

poder aplicar KD a cada uno de los bloques, el estudiante también está compuesto por el mismo número de bloques, como se muestra en la Fig. 1. En los modelos analizados en este trabajo se sigue una metodología de KD por bloques en la que las últimas capas (de clasificación) no se incluyen en el proceso de KD.

2.2. Estrategias de Destilación por Bloques

El enfoque de KD por bloques presentado ofrece un marco flexible que puede adaptarse a diferentes modelos de KD, como se ilustra en la Fig. 1. Al aplicar KD únicamente al bloque final, entrenando los bloques restantes mediante retropropagación, como se muestra en la Fig. 1-DUB, se obtiene un esquema comúnmente adoptado, donde solo se tiene en cuenta la salida final del profesor. Aplicar KD a todos los bloques, como se muestra en la Fig. 1-DBB, debería de producir el modelo con el mayor grado de alineación funcional entre las salidas del profesor y del estudiante en cada bloque. Además, pueden concebirse aplicaciones intermedias de KD, como lo mostrado en la Fig. 1-DBE, donde se plantea aplicar KD a un subconjunto específico de bloques (p. ej., bloques 1, 2, 3 y 6).

2.3. Estructura del Bloque Estudiante

Los bloques del estudiante están compuestos por varios módulos Residuales Invertidos [20]. Cada módulo consiste en una expansión de canales de 16 a 96, seguida de normalización, una convolución *depthwise* de 7×7 que mantiene 96 canales con normalización, y un módulo *Squeeze-and-Excitation* [10] que reduce la dimensión de canales de 96 a 4 y luego la expande de nuevo a 96. Finalmente, una convolución 1×1 proyecta las características de 96 a 16 canales, seguida de normalización. Solo cuando las formas de entrada y salida coinciden, se añade una conexión residual de salto.

2.4. Proceso de Entrenamiento

En esta estrategia, utilizamos un modelo preentrenado como profesor, ajustado o no al conjunto de datos específico. Los bloques del estudiante se entrenan para minimizar la diferencia entre sus salidas y las del bloque correspondiente del profesor, ignorando el error de clasificación. La diferencia entre los bloques correspondientes de cada modelo (primero, segundo, ...) se cuantifica con el Error Cuadrático Medio, y se utiliza como función de pérdida para entrenar el bloque del estudiante seleccionado por KD, como se muestra en la Fig. 1-DBB. Si un bloque no es seleccionado para aplicar KD, como se describe en la Sección 2.2, se entrena únicamente mediante retropropagación para dar soporte al siguiente bloque seleccionado para KD. Un ejemplo visual está en la Fig. 1-DBE, que muestra que los primeros 3 bloques se entrenan de forma individual y secuencial, y los últimos 3 bloques se entrenan como si fueran uno solo, utilizando las salidas de la última capa. La etapa final, compuesta por las capas de clasificación, se entrena siempre considerando una función de pérdida para minimizar el error en el problema de clasificación. Una vez finalizado el proceso de KD, el estudiante puede, o no, ser ajustado al conjunto de datos específico.

3. Sección Experimental

En esta sección presentamos los resultados experimentales, analizando la influencia del ajuste fino y el impacto de aplicar KD a diferentes subconjuntos de bloques, con especial atención a los conjuntos de datos para entrenamiento de grano fino.

3.1. Marco Experimental

Para nuestros experimentos, hemos seleccionado EfficientNet-B0 [21] como profesor, un modelo CNN reconocido por su rendimiento y disponible en las bibliotecas de DL. El profesor ha sido preentrenado en ImageNet [13], por ser ampliamente utilizado en numerosos problemas del mundo real.

Hemos llevado a cabo los experimentos con 11 conjuntos de datos de clasificación diferentes. Los siguientes conjuntos de datos clásicos:

- **CIFAR10** y **CIFAR100**, referencias de clasificación de imágenes con 10 y 100 categorías de objetos, respectivamente [12].
- **FashionMNIST**, clasificación de imágenes de ropa con 10 clases [24].
- **MNIST**, reconocimiento de dígitos escritos a mano [15].
- **EMNIST**, versión extendida de MNIST, incluye letras escritas a mano [5].
- **Food101**, clasificación de imágenes de comida con 101 clases [2].
- **SVHN**, clasificación de dígitos en imágenes naturales de vista de calle, más realista que MNIST o EMNIST [18].

Para evaluar con conjuntos de datos escasos para un entrenamiento de grano fino, comparamos también con:

Tabla 1: Conjuntos de datos usados con sus características principales

	Conjuntos	#Entrenamiento	#Test	#Clases	$\frac{\#Entrenamiento}{\#Clases}$
Clásicos	CIFAR10 [12]	50,000	10,000	10	5,000
	CIFAR100 [12]	50,000	10,000	100	500
	EMNIST [5]	112,800	18,800	47	2,400
	FashionMNIST [24]	60,000	10,000	10	6,000
	Food101[2]	75,750	25,250	101	750
	MNIST [15]	60,000	10,000	10	6,000
	SVHN [18]	73,257	26,032	10	-
Grano Fino	CUB200 [22]	5,994	5,794	200	30-60
	ISIC [4]	1,849	459	8	220
	OxfordPets [19]	3,680	3,669	37	99
	StanfordCars [11]	8,144	8,041	196	24-68

- **CUB200**, clasificación de especies de aves [22].
- **ISIC**, referencia de clasificación de lesiones cutáneas [4].
- **OxfordPets**, clasificación de razas de perros y gatos [19].
- **StanfordCars**, clasificación de modelos de coches [11].

Sus propiedades se resumen en la Tabla 1, centrándonos en el número de ejemplos, clases y ejemplos por clase. Para varios conjuntos de datos desbalanceados, dicha proporción es un intervalo, ya que depende de la clase. SVHN está tan desbalanceado que cualquier valor podría resultar confuso.

Para cada conjunto de datos, el conjunto de entrenamiento se ha dividido en un conjunto de entrenamiento real (80 %) y un conjunto de validación (20 %) para la parada temprana. Siguiendo la práctica habitual, los tamaños de lote se han ajustado según las características de cada conjunto de datos: para ISIC es 16; para el resto de conjuntos de entrenamiento de grano fino, Food101 y MNIST, 32; para el resto de conjuntos de datos, 128. Para la etapa final de entrenamiento, hemos establecido el número máximo de épocas en 300, con una paciencia de parada temprana de 15. Para asegurarnos de que el entrenamiento no se detenga en mínimos relativos, también hemos establecido el número mínimo de épocas en 100. A continuación, hemos entrenado cada modelo 5 veces y calculado el *accuracy* medio sobre el conjunto de prueba para cada conjunto de datos.

En [16] se proponen bloques Residuales Invertidos con distinto número de módulos, y sobre éstos se aplica una búsqueda para optimizar, para el problema, el número de módulos para cada bloque. En nuestro caso buscamos que los bloques sean homogéneos, así que hemos llevado a cabo una pequeña experimentación con un número diferente de módulos por bloque, de 1 a 4. Comparando todos estos modelos, observamos que usar 2 produce resultados similares a los del mejor estudiante obtenido mediante su compleja búsqueda, reduciendo el tiempo necesario para definirlo y entrenarlo. El resto de parámetros utilizados por la propuesta son los hiperparámetros establecidos en [16].

Los modelos a comparar según el subconjunto de bloques son los siguientes:

- **Destilación del Último Bloque** (**DUB**): KD se aplica únicamente al último bloque, como se muestra en la Fig. 1-DUB. Este es muy similar a la KD tradicional, en la que solo la salida final (antes de la capa de clasificación) se entrena mediante KD.
- **Destilación de Bloques Específicos** (**DBE**): KD se aplica a un subconjunto de bloques; en este trabajo se eligen los bloques 1, 2, 3 y 6 (Fig. 1-DBE), elegidos en base a experimentos que no se incluyen por motivos de espacio. Esta es una propuesta intermedia entre DUB y DBB.
- **Destilación Bloque a Bloque** (**DBB**): KD se aplica a todos los bloques, como se muestra en la Fig. 1-DBB. Este algoritmo es la propuesta que aplica la técnica de forma más exhaustiva.

En algunos experimentos, indicado explícitamente, aplicamos Aumento de Datos (*Data Augmentation*, DA), permitiendo que cada imagen original genera-se cuatro variaciones y aumentando efectivamente el tamaño del subconjunto en cinco veces. Las transformaciones se seleccionaron para preservar las características distintivas de cada clase. Para OxfordPets, aplicamos volteos horizontales, variación de color agresiva, escala de grises aleatoria y desenfoque gaussiano. Para ISIC, empleamos transformaciones afines aleatorias, volteos bidireccionales y variación de color moderada. Para CUB200 y StanfordCars, usamos aumentaciones más conservadoras (pequeñas rotaciones aleatorias, volteos horizontales y ligera variación de color) para evitar distorsionar características discriminativas.

En la tablas de resultados hemos marcado en negrita para cada conjunto de datos el mejor valor de *accuracy* y subrayado el segundo mejor.

Hemos implementado nuestra arquitectura mediante PyTorch (2.6.0). Las especificaciones de la máquina utilizada en nuestros experimentos incluyen un nodo con dos procesadores AMD EPYC 7742 de 64 núcleos, 1 TiB de RAM y 8 GPUs Nvidia A100, cada una con 40 GB de VRAM. Sin embargo, cada modelo ha sido entrenado en una única GPU a la vez.

3.2. Influencia del Ajuste Fino en los Modelos Profesor y Estudiante

En esta sección, abordamos la decisión sobre si es mejor realizar el ajuste fino al modelo profesor o al modelo estudiante. Los algoritmos comparados, con su correspondiente notación, son los siguientes:

- ***Baseline*:** Resultados del modelo profesor (EfficientNetB0), preentrenado con ImageNet y ajustado para cada problema.
- **Profesor Ajustado (*Fine-tuned Teacher*, FT):** Resultados de KD utilizando un modelo profesor ajustado, sin realizar el ajuste fino del modelo estudiante.
- **Estudiante Ajustado (*Fine-tuned Student*, FS):** Resultados de KD utilizando un modelo profesor preentrenado y sin ajuste fino, y posteriormente realizando el ajuste fino del modelo estudiante.
- **Profesor y Estudiante Ajustados (*Fine-tuned Teacher and Student*, FTS):** Resultados de KD utilizando un modelo profesor ajustado, y posteriormente realizando el ajuste fino del estudiante.

Tabla 2: *Accuracy* medio según la estrategia de ajuste fino

(a) KD en el último bloque (DUB) (b) KD en todos los bloques (DBB)

	Conjuntos	*Baseline*	FT	FS	FTS	*Baseline*	FT	FS	FTS
Clásicos	CIFAR10	**0.914**	**0.914**	0.873	0.902	**0.914**	0.898	0.889	0.894
	CIFAR100	**0.797**	0.774	0.733	0.763	**0.797**	0.774	0.743	0.776
	EMNIST	0.903	**0.907**	0.897	0.897	0.903	**0.905**	0.896	0.900
	FashionMNIST	0.942	**0.944**	0.936	0.938	**0.942**	0.937	0.934	0.935
	Food101	0.803	**0.812**	0.751	0.783	**0.803**	0.784	0.734	0.774
	MNIST	0.995	**0.996**	0.995	**0.996**	**0.995**	**0.995**	**0.995**	**0.995**
	SVHN	0.967	**0.970**	0.957	0.964	**0.967**	0.965	0.955	0.965
Grano Fino	CUB200	**0.722**	0.413	0.364	0.383	**0.722**	0.672	0.659	0.678
	ISIC	**0.684**	0.590	0.488	0.535	0.684	**0.694**	0.687	0.687
	OxfordPets	**0.891**	0.530	0.462	0.488	**0.891**	0.845	0.797	0.815
	StanfordCars	**0.696**	0.424	0.449	0.403	0.696	0.703	0.744	**0.747**

Tabla 3: Tiempos de entrenamiento medios según la estrategia de ajuste fino

Medida	*Baseline*	FT	FS	FTS
#Parámetros (millones)	4.020	**2.707**	**2.707**	**2.707**
Tiempo Entrenamiento (DUB)	04h08m	05h15m	02h02m	05h56m
Tiempo Entrenamiento (DBB)	04h08m	06h27m	02h57m	07h17m

Los resultados obtenidos relativos a aplicar KD solo al bloque final (DUB) se muestran en la Tabla 2a, mientras que los relativos a aplicar KD a todos los bloques (DBB) se presentan en la Tabla 2b; y el tiempo de procesamiento medio y complejidad de ambos, en la Tabla 3.

Respecto a DUB (Tabla 2a), ajustar solo el profesor (FT) supera a las demás opciones; ajustar también el estudiante solo incrementa el tiempo de procesamiento (1 hora más de media) con peores resultados. Aunque FT mejora claramente frente a FS, requiere más tiempo (5 horas frente a 2 de media), por lo que FS resulta preferible con recursos limitados. Frente a la *baseline*, nuestra propuesta la supera en 6 de los 7 conjuntos de datos clásicos, demostrando ser muy competitiva con menos parámetros a costa de 1 hora adicional de entrenamiento, lo que da respuesta a la PI1. Sin embargo, en conjuntos de entrenamiento de grano fino se observa una clara degradación del rendimiento.

En DBB (Tabla 2b) se mantienen las mismas tendencias: ajustar solo el profesor (FT) supera a ajustar ambos modelos (FTS) en 6 casos, y solo el estudiante (FS) en 9, aunque FS sigue siendo un enfoque interesante por su menor coste (3 horas frente a 7). Comparando FT con la *baseline*, aunque esta última es superior en 7 de 11 conjuntos, el modelo KD obtiene resultados competitivos con bastante menos parámetros, reforzando la respuesta a PI1. A diferencia de DUB,

no observamos una degradación en los conjuntos de entrenamiento de grano fino. Este análisis resuelve la PI2.

Con base en estos resultados, en el resto de este trabajo nos centramos en dos opciones de ajuste fino: FT, que ofrece los mejores resultados, y FS, que logra un rendimiento competitivo a un menor coste computacional.

3.3. Efecto del Número de Bloques en la KD

En esta sección comparamos los modelos Destilación del Último Bloque con los Destilación Bloque a Bloque en más detalle. Los resultados obtenidos se muestran en la Tabla 4 (reorganizada a partir de la Tabla 2 para facilitar esta comparación directa), y sus tiempos de procesamiento medios y complejidades en la Tabla 3. Observamos que, en los conjuntos de datos clásicos, los modelos DUB son claramente los de mejor rendimiento, aunque los modelos DBB también presentan resultados competitivos. En cambio, en los conjuntos de datos de entrenamiento de grano fino, los modelos DBB obtienen mejores resultados que los modelos DUB. Respecto al tiempo de procesamiento, entrenar modelos DUB requiere, de media, aproximadamente 1 hora menos que entrenar modelos DBB.

Tabla 4: *Accuracy* medio de usar KD en solo el último bloque (DUB) vs en todos los bloques (DBB)

			DUB		DBB	
	Conjuntos	*Baseline*	FT	FS	FT	FS
Clásicos	CIFAR10	**0.914**	**0.914**	0.873	0.898	0.889
	CIFAR100	**0.797**	0.774	0.733	0.774	0.743
	EMNIST	0.903	**0.907**	0.897	0.905	0.896
	FashionMNIST	0.942	**0.944**	0.936	0.937	0.934
	Food101	0.803	**0.812**	0.751	0.784	0.734
	MNIST	0.995	**0.996**	0.995	0.995	0.995
	SVHN	0.967	**0.970**	0.957	0.965	0.955
Grano Fino	CUB200	**0.722**	0.413	0.364	0.672	0.659
	ISIC	0.684	0.590	0.488	**0.694**	0.687
	OxfordPets	**0.891**	0.530	0.462	0.845	0.797
	StanfordCars	0.696	0.424	0.449	0.703	**0.744**

3.4. Análisis de Problemas de Grano Fino

Los resultados sugieren que aplicar KD en todos los bloques supera a aplicarlo solo en el último en conjuntos de datos con pocas instancias.Para investigar si esta característica es la causa subyacente, aplicamos Aumento de Datos (DA) a dichos conjuntos, limitando su uso a estos por su elevado coste temporal.

Tabla 5: *Accuracy* medio usando y sin usar DA

(a) KD en el último bloque (DUB)

Conjuntos	FT	FT_DA	FS	FS_DA
CUB200	0.413	0.645	0.364	**0.667**
ISIC	0.590	**0.713**	0.488	0.705
OxfordPets	0.530	**0.798**	0.462	0.758
StanfordCars	0.424	0.729	0.449	**0.732**

(b) KD en todos los bloques (DBB)

FT	FT_DA	FS	FS_DA
0.672	**0.696**	0.659	0.646
0.694	0.712	0.687	**0.743**
0.845	**0.870**	0.797	0.841
0.703	0.696	**0.744**	0.646

Tabla 6: Tiempos de entrenamiento para modelos DUB y DBB usando y sin usar Aumento de Datos

	Destilación del Último Bloque		Destilación Bloque a Bloque	
Datos	FT	FS	FT	FS
No DA	00h42m	01h12m	01h23m	00h47m
DA	05h03m	04h42m	07h35m	07h08m

Las Tablas 5a y 5b muestran los resultados de aplicar Aumento de Datos a los modelos DUB y DBB, respectivamente. En la Tabla 5a, podemos observar que el aplicar DA permite a los modelos DUB reducir la diferencia de resultados con los modelos DBB. Esto confirma que el rendimiento superior de DBB sobre los modelos DUB en problemas fino se debía al reducido número de instancias por clase en estos conjuntos de datos. Por tanto, los modelos DBB superan a los modelos DUB, especialmente en conjuntos de datos con pocos ejemplos por clase, es decir, conjuntos de datos de entrenamiento de grano fino, porque el conocimiento almacenado en los bloques intermedios parece más relevante en dichos escenarios.

Aunque la incorporación de DA al entrenamiento de los modelos DUB reduce las diferencias, aplicarlo a los modelos DBB también mejora sus resultados (como se muestra en la Tabla 5b), lo que indica que estos modelos de KD también se benefician de DA. Sin embargo, como se muestra en la Tabla 6, DA aumenta el tiempo de procesamiento en un factor de cuatro en el mejor caso, lo que no es una opción viable en muchos casos. En la siguiente sección, aplicamos KD a un subconjunto de bloques, buscando un equilibrio entre el sólido rendimiento de los modelos DUB en conjuntos de datos con datos suficientes y el mejor rendimiento de los modelos DBB en conjuntos de datos de entrenamiento grano fino.

3.5. Mejora de Conjuntos de Datos de Problemas de Grano Fino con un Enfoque Intermedio

En esta sección, introducimos un enfoque intermedio entre los modelos DUB y DBB, al que nos referimos como modelo "**Destilación de Bloques Específicos**" (**DBE**). En esta metodología, aplicamos KD a los bloques 1, 2, 3 y 6

(Fig. 1-DBE). Se ha elegido aplicar KD a estos bloques en base a experimentos que no se incluyen por motivos de espacio. Este enfoque mejora los resultados en conjuntos de datos de entrenamiento de grano fino manteniendo un rendimiento competitivo en los restantes. Por claridad, presentamos únicamente los resultados con profesor ajustado (FT), al ser la estrategia con mejores resultados.

Tabla 7: *Accuracy* medio de todos los modelos con profesor ajustado (FT)

	Conjuntos	*Baseline*	DUB	DBE	DBB
Clásicos	CIFAR10	**0.914**	**0.914**	0.911	0.898
	CIFAR100	**0.797**	0.774	0.783	0.774
	EMNIST	0.903	**0.907**	**0.907**	0.905
	FashionMNIST	0.942	**0.944**	0.943	0.937
	Food101	0.803	**0.812**	0.801	0.784
	MNIST	0.995	**0.996**	0.995	0.995
	SVHN	0.967	**0.970**	**0.970**	0.965
Grano Fino	CUB200	**0.722**	0.413	0.620	0.672
	ISIC	0.684	0.590	0.690	**0.694**
	OxfordPets	**0.891**	0.530	0.790	0.845
	StanfordCars	0.696	0.424	0.657	**0.703**

La Tabla 7 muestra los resultados del modelo DBE. Puede observarse que este, aunque obtiene peores resultados en conjuntos de datos de entrenamiento de grano fino que el modelo DBB, supera claramente al modelo DUB en ellos. En conjuntos de datos más clásicos, el modelo DBE es más competitivo que el modelo DBB pero menos que el modelo DUB. Por tanto, se puede considerar que el modelo DBE ofrece un mejor equilibrio entre el *accuracy* en conjuntos de datos clásicos y en problemas de grano fino, dando así respuesta a la PI3. Además, DBE presenta un rendimiento similar a la *baseline*, lo que lo convierte en una metodología competitiva.

4. Conclusiones

En este trabajo, hemos propuesto un enfoque de KD profesor-estudiante para CNN que aprovecha su estructura secuencial de bloques para crear arquitecturas de estudiante más simples y homogéneas. Además, es flexible, ya que puede aplicar KD a diferentes subconjuntos de bloques. Usando EfficientNet-B0 como profesor, hemos evaluado tres estrategias de KD: aplicar KD únicamente al último bloque (DUB), a todos los bloques (DBB) y a un subconjunto intermedio (DBE). En conjuntos de datos clásicos, DUB obtiene los mejores resultados con solo el 67 % de los parámetros, superando incluso a la *baseline*. Por tanto, nuestra estructura homogénea de estudiantes es muy competitiva. Sin embargo, DUB se

degrada significativamente en problemas de grano fino. Hemos demostrado mediante DA que la limitación de datos de entrenamiento causa esta degradación, y hemos mostrado que el uso de conocimiento intermedio en DBE proporciona un rendimiento robusto, tanto en conjuntos de datos clásicos como de entrenamiento de grano fino. Entre las diferentes estrategias de ajuste fino, ajustar únicamente el profesor produce los mejores resultados, mientras que ajustar únicamente el estudiante es más rápido y ofrece un enfoque más realista. A través del estudio de diferentes métodos de KD, hemos demostrado que nuestra propuesta de KD con modelos intermedios por bloques es altamente competitiva y que los bloques intermedios conducen a un mejor rendimiento en conjuntos de datos con pocos ejemplos por clase. Al destacar los beneficios y limitaciones de los distintos enfoques de KD, esperamos inspirar una mayor investigación y avances en este campo, especialmente para conjuntos de datos con pocas instancias por clase.

Agradecimientos. Esta investigación es parte del Proyecto TSI-100927-2023-1, financiado por Plan de Recuperación, Transformación y Resiliencia Europeo, de la Unión Europea-Next Generation EU a través del Ministerio para la Transformación Digital y de la Función Pública y está parcialmente financiada por el Ministerio de Ciencia y Tecnología de España en el marco del proyecto PID2023-149128NB-I00, financiado por MICIU/AEI/10.13039/501100011033 y por FEDER, UE.

Declaración de Intereses. Los autores declaran que no tienen intereses en conflicto relevantes para el contenido de este artículo.

Referencias

1. Alkhulaifi, A., Alsahli, F., Ahmad, I.: Knowledge Distillation in Deep Learning and Its Applications. PeerJ Computer Science **7**, e474 (2021)
2. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – Mining Discriminative Components with Random Forests. In: Computer Vision – ECCV 2014, vol. 8694, pp. 446–461. Springer International Publishing (2014)
3. Cheraghian, A., Rahman, S., Fang, P., Roy, S.K., Petersson, L., Harandi, M.: Semantic-Aware Knowledge Distillation for Few-Shot Class-Incremental Learning. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2534–2543. IEEE (2021)
4. Codella, N.C.F., Gutman, D., Celebi, M.E., Helba, B., Marchetti, M.A., Dusza, S.W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., Halpern, A.: Skin Lesion Analysis towards Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), hosted by the International Skin Imaging Collaboration (ISIC). In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). pp. 168–172 (2018)
5. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: EMNIST: Extending MNIST to handwritten letters. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 2921–2926 (2017)
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, 1 edn. (2016)
7. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge Distillation: A Survey. International Journal of Computer Vision **129**(6), 1789–1819 (2021)

8. Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. In: NeurIPS Deep Learning and Representation Learning Workshop (2015)
9. Hu, C., Li, X., Liu, D., Wu, H., Chen, X., Wang, J., Liu, X.: Teacher-Student Architecture for Knowledge Distillation: A Survey (2023), arXiv:2308.04268
10. Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7132–7141 (2018)
11. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D Object Representations for Fine-Grained Categorization. In: IEEE International Conference on Computer Vision Workshops (ICCVW) (2013)
12. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images (2009)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 25 (2012)
14. Lan, X., Zeng, Y., Wei, X., Zhang, T., Wang, Y., Huang, C., He, W.: Counter-clockwise Block-by-Block Knowledge Distillation for Neural Network Compression. Scientific Reports **15**(1), 11369 (2025)
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
16. Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X.: Block-Wisely Supervised Neural Architecture Search With Knowledge Distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1986–1995 (2020)
17. Liu, K., Ye, X., Sun, B., Yang, H., Li, H., Xu, R., Wang, Z.: Low-Resolution Few-Shot Learning via Multi-Space Knowledge Distillation. Information Sciences **677**, 120968 (2024)
18. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading Digits in Natural Images with Unsupervised Feature Learning. In: NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011 (2011)
19. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and Dogs. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3498–3505 (2012)
20. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4510–4520 (2018)
21. Tan, M., Le, Q.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: Proceedings of the 36th International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
22. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
23. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A Survey of Transfer Learning. Journal of Big data **3**(1), 9 (2016)
24. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms (2017), arXiv:1708.07747
25. Younesi, A., Ansari, M., Fazli, M., Ejlali, A., Shafique, M., Henkel, J.: A Comprehensive Survey of Convolutions in Deep Learning: Applications, Challenges, and Future Trends. IEEE Access **12**, 41180–41218 (2024)