Adaptación automática del operador de *pooling* aprendiendo pesos de medias ponderadas ordenadas en Redes Neuronales Convolucionales

Juan I. Forcén
Das-nano — Veridas
Poligono Industrial Talluntxe II
Tajonar, España
jiforcen@das-nano.es

Miguel Pagola
dept. Estadística Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
miguel.pagola@unavarra.es

Edurne Barrenechea
dept. Estadística Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
edurne.barrenechea@unavarra.es

Humberto Bustince
dept. Estadística Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
bustince@unavarra.es

***Resumen*—Las Redes Neuronales Convolucionales (RNCs) han logrado ser los modelos de mayor precisión en varias tareas de visión por computador. Tienen estructuras cada vez más complejas en las que se deben seleccionar un buen número de parámetros, como el número de capas, los filtros por capa o las capas de re-muestreo conocidas como *pooling*. En la capa de *pooling* las activaciones de los píxeles vecinos se agregan de forma local; el máximo y la media ponderada son las funciones de agregación comúnmente utilizadas. En este trabajo presentamos una nueva forma de agregar a través de medias ordenadas ponderadas, de tal forma que los pesos de las medias se aprenden durante el entrenamiento de la red. De esta forma el operador de *pooling* se selecciona automáticamente durante el entrenamiento de la red. Por lo tanto, la RNC tiene unos hiper-parámetros menos que no debemos seleccionar. En los resultados experimentales mostramos que la precisión y la capacidad de generalizar de las redes que utilizan este tipo de operador es similar a las redes con operadores de *pooling* fijados.**

***Index Terms*—Clasificación, Redes Neuronales Convolucionales, Pooling, Máximo, Medias ponderadas ordenadas**

I. Introducción

Las Redes Neuronales Convolucionales (RNCs) que están inspiradas en cómo funciona la corteza visual, consisten en múltiples capas de filtros convolucionales de una o más dimensiones [1]. En el ámbito de la clasificación de imágenes las RNCs son el estado del arte en cuanto a precisión y han cambiado el paradigma del procesamiento de imágenes tradicional, ya que las características se aprenden durante el proceso de entrenamiento de la red. Las RNCs son utilizadas en reconocimiento de objetos, conducción automática o reconocimiento del habla [2].

La estructura de una RNC es una serie de capas convolucionales seguidas de capas de *pooling* y finaliza con una serie de capas de neuronas completamente conectadas. El objetivo de cada capa convolucional es producir representaciones que reflejen características locales de la imagen. Cada capa convolucional consta de múltiples canales en los que las características están representadas según los valores de activación. En las capas de *pooling* se agregan las activaciones de cada región de cada canal de la capa de convolución precedente. Este paso se utiliza para obtener una representación más compacta, más robusta e invariante a las transformaciones de las imágenes. A pesar de tener una influencia muy grande en el funcionamiento de la red, la operación de *pooling* no ha sido muy estudiada y la mayor parte de modelos utilizan el máximo o la media aritmética [3]. Al utilizar operadores fijos, en las capas de *pooling* no se aprende ningún parámetro, sin embargo, el tipo de operador de *pooling* que se utiliza en cada una de las capas pasa a ser un hiper-paramétro más de la red. Por lo tanto, pasa a ser otro de los problemas de las RNCs, el alto número de hiper-paramétros [4] (número de capas de la red, número de filtros por capa, el tamaño de los filtros, ratio de aprendizaje, tamaño del subconjunto de entrenamiento, etc.). Debido al alto número de hiper-parámetros, la selección para un problema dado de la arquitectura de red más adecuada tiene un alto coste computacional.

En este trabajo proponemos utilizar como operador de *pooling* medias ponderadas ordenadas [5], cuyos pesos son aprendidos en la fase de entrenamiento. Al ser una media ponderada aseguramos que el resultado de la agregación está siempre entre el máximo y el mínimo valor de los elementos agregados. Por lo tanto, durante del proceso de entrenamiento los valores de los pesos pueden ajustarse a que el operador resultante sea el máximo, la media aritmética o cualquier otro operador que resulte en una red con mejor precisión en el conjunto de entrenamiento.

En la validación experimental hemos comprobado que (en las arquitecturas que hemos probado) si reemplazamos los operadores de *pooling* clásicos por nuestra propuesta, se

alcanza una precisión igual o superior en varios conjuntos de imágenes (CIFAR-10, CIFAR-100, FMNIST).

El trabajo se divide en las siguientes secciones: en la sección II realizamos un análisis de la operación de *pooling* y repasamos los trabajos relacionados. En la sección III describimos el método propuesto y en la sección IV presentamos los resultados obtenidos en los diferentes experimentos. Acabamos con las conclusiones y posibles trabajos futuros.

II. Análisis de la operación de *pooling*

La operación de *pooling*, a veces conocida como de remuestreo, es un paso crucial en varios sistemas de reconocimiento de imágenes como el modelo *Bag-of-Words*, el método VLAD [6], el Super Vector [7] o las RNCs. Dos factores definen la operación de *pooling*: uno es la región de la imagen cuyas características locales se van a agregar y el otro factor es la operación de agregación que define la forma combinar dichas características locales.

En este trabajo nos vamos a centrar en la forma de agregar los valores de las características. Los operadores más comunes son el máximo, utilizado en las arquitecturas de redes más conocidas (AlexNet [8], VGG [9]) y la media aritmética que se utiliza en otros modelos (p.e. en Network in Network [11] o GoogleNet [10]). Recordemos que en la operación de *pooling* simplemente tomamos una ventana de tamaño $n \times n$ de una imagen de características y obtenemos un único valor. Deslizando la ventana de n en n píxeles a lo largo de toda la imagen obtendremos una nueva imagen de menor dimensión que la original.

Boureau et. al. [3], analizó el método conocido como *Spatial Pyramid* [12] e identificó mucho mejor rendimiento en clasificación en varios conjuntos de imágenes utilizando el máximo frente a la media aritmética. Más aún, en [13] Boureau et. al. desarrollaron un estudio teórico en el que demuestran que utilizar el máximo en la fase de agregación de vectores de características es adecuado cuando las características que definen a una clase tienen poca probabilidad de activación. Este resultado fue validado experimentalmente en [14] y en [15]. En ambos trabajos, el operador máximo obtiene los mejores resultados cuando se utilizan vectores dispersos de características muy grandes. En los vectores de características dispersos, cada característica tiene poca probabilidad de activación (es decir que su valor sea mayor que cero) y tiene mucho poder discriminatorio. Sin embargo, cuando se utilizan operadores de *pooling* que suavizan el valor máximo (el máximo esperado en [13] y otras metodologías en [14]) se obtienen todavía mejores resultados que con el máximo. Similar resultado obtuvimos en [16], donde estudiamos la cardinalidad del operador de *pooling* que realiza la media sobre los $\mathcal{N}$ valores mayores (utilizando *Bag-of-Words* y *Spatial Pyramid*). Además en [17] comprobamos que las medias ponderadas ordenadas también dan buenos resultados en dicha metodología de clasificación de imágenes. En [14] y [13] se argumenta que las conclusiones obtenidas se pueden trasladar a la operación de *pooling* de las RNCs, ya que si aplicamos el método de *Bag-of-Words* en diferentes capas, el modelo global es equivalente a una RNC.

Pensemos en una red con una estructura con varias capas. Si en la imagen que estamos tratando existe, por ejemplo, una esquina muy definida, habrá un valor de activación muy alto en una imagen de características en la que se representen las esquinas. Por lo tanto, si en las restantes capas de *pooling* utilizamos el operador máximo, este valor (que es muy alto) se va a ir propagando por la arquitectura de la red y llegará a formar parte de la representación final de características de la imagen. Si dicha esquina es representativa de la clase de la imagen, entonces será una característica discriminatoria y servirá para clasificar correctamente la imagen.

Por otro lado, imaginemos que por toda la imagen hay una textura que no está claramente definida, pero que representa a la clase de la imagen. El valor de activación en la imagen de características de esa textura será pequeño. Si el operador de *pooling* en todas las capas es el máximo, el valor que se propaga por la red es pequeño y por lo tanto desaparecerá o no tendrá mucha representación en el vector final. Sin embargo, si el operador utilizado es la media aritmética, al aparecer la textura por toda la imagen, su representación en el vector final será mayor. Esto es debido a que las características que aparecen aisladamente son filtradas por la media y acaban teniendo un valor más pequeño en la representación final.

Teniendo en cuenta que el máximo y la media son operadores adecuados para diferentes tipos de características, en [18] se proponen dos métodos en los que se aprende la función de *pooling*. En la primera estrategia, se aprende un parámetro que mezcla el resultado del valor máximo con la media aritmética. En el segundo método se aprende una función de *pooling* en forma de árbol que va mezclando los resultados de diferentes filtros de *pooling*. Estos filtros también se aprenden y son idénticos a los filtros de convolución. En ambos casos obtiene mejores resultados que si se utilizan operadores de *pooling* fijos. Sin embargo, la segunda estrategia es similar a añadir nuevas capas de convolución, por lo tanto, no puede considerarse una generalización de la operación de *pooling*. En este trabajo proponemos un método de pooling en el que se aprenda una función de *pooling*. El objetivo es aprender unos coeficientes que afecten únicamente a los valores de las activaciones (en el caso de la convolución los pesos están asociados a su posición espacial en el filtro). Para ello utilizaremos medias ordenadas ponderadas, en las que el vector de valores se ordena de mayor a menor y después cada valor es multiplicado por el coeficiente asociado a cada posición del vector ordenado. Los valores de los pesos se aprenderán en la fase de entrenamiento. De esta forma, nos proponemos obtener un operador de *pooling* que propague los valores más altos de las activaciones, teniendo en cuenta también los valores de activaciones medios que aparezcan frecuentemente.

III. *Pooling* basado en medias ponderadas ordenadas

Los operadores estándar de *pooling* son o el máximo $f_{max}(\boldsymbol{x}) = \max_{i \in N}\{\boldsymbol{x}_i\}$ o la media aritmética $f_{med} =$

$\frac{1}{N}\sum_{i\in N}\boldsymbol{x}_i$, donde el vector $\boldsymbol{x}$ contiene los valores de activación de una región de *pooling* local de tamaño N píxeles (típicamente 2x2 o 3x3).

Las medias ponderadas ordenadas son funciones de agregación promedio. Se diferencian de las medias ponderadas en que los pesos no están asociados a unas posiciones del vector de entrada sino a las magnitudes. Fueron propuestas por Yager en 1988 [5].

$$f_{mpo}(\boldsymbol{x}_i \searrow) = \sum_{i\in N} w_i x_i \tag{1}$$

Donde $(\boldsymbol{x}_i \searrow)$ es un vector ordenado con $x_1 \geq x_2 \geq \cdots \geq x_N$. La suma de pesos debe ser igual a uno $\sum_{i=1}^{n} w_i = 1$ y $w_i \geq 0$ para cada $i = 1, ..., n$.

El cálculo de la media ponderada ordenada requiere de la ordenación de los valores que van a ser agregados. Dependiendo de los valores de los pesos podemos obtener las funciones de agregación típicas [19], por ejemplo:

- Si $w = (0, 0, \cdots, 0, 1)$, entones f_w = mínimo.
- Si $w = (1, 0, \cdots, 0, 0)$, entones f_w = máximo.
- Si $w = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}, \frac{1}{n})$, entonces f_w = media aritmética.
- Si $w = (0{,}5, 0{,}5, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, entonces f_w = media de los dos valores mayores.

Cuando utilizamos un operador de *pooling* con medias ponderadas ordenadas tenemos varias posibilidades, aprender unos pesos por red, por capa, por cada canal de cada capa o incluso de cada región de cada canal de cada capa. A continuación vamos a desarrollar las ecuaciones para el caso de aprender un conjunto de pesos en una capa de la red. Sea la función de coste de la red J, creamos una nueva función de coste en la que añadimos tres términos. El primer término fuerza que los valores sean mayores que cero, el segundo obliga a que la suma de pesos sea 1 y el tercer término penaliza las diferencias entre los pesos consecutivos. Al utilizar estos términos (similares a una regularización) obtenemos unos valores de los pesos que serán más interpretables.

$$J_{mpo} = J + C_1 \textstyle\sum_{i=1}^{N} max\{0, -w_i\} +$$
$$+ \quad C_2 \left(\left(\textstyle\sum_{i=1}^{N} w_i\right) - 1\right)^2 + C_3 \left(\textstyle\sum_{i=1}^{N-1} (w_i - w_{i+1})^2\right)$$

Por lo tanto el cálculo del gradiente para utilizarlo en el algoritmo de optimización queda:

$$\Delta_{w_i} J_{mpo} = \Delta_{w_i} J - C_1 +$$
$$+2C_2 \left(\left(\textstyle\sum_{i=1}^{N} w_i\right) - 1\right) w_i + 2C_3(w_i - w_{i+1}) \tag{2}$$

Si $w_i \geq 0$ el término C_1 desaparece. El cálculo del gradiente $\Delta_{w_i} J$ es similar al de una capa de convolución. Teniendo en cuenta que en lugar de la convolución, hay que ordenar los valores de la capa de activación.

IV. RESULTADOS EXPERIMENTALES

En la fase de experimentación queremos comprobar la precisión de RNCs ya conocidas y validadas cuando sustituimos el operador de *pooling* original por el operador de *pooling* basado en medias ordenadas ponderadas. Además queremos comprobar si los pesos convergen hacia unos operadores de *pooling* similares a los operadores originales ($(1, 0, 0, \cdots, 0)$ para el máximo o $(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$ para la media). Los modelos de red que utilizaremos son dos: la red propuesta en [11] que es conocida como Network in Network (NiN) y el modelo VGG [9]. Ambas arquitecturas quedan reflejadas en la tabla I y el tabla II respectivamente (todas las convoluciones van seguidas de activaciones tipo Relu).

Cuadro I
PARÁMETROS DE LA RED NIN

Input		Filtros, canales
32x32		5x5, 192
32x32		1x1, 160
32x32		1x1, 96
32x32	pool1	3x3 Max *pooling*, stride 2
16x16		dropout 0.5
16x16		5x5, 192
16x16		1x1, 192
16x16		1x1, 192
32x32	pool2	3x3 Med *pooling*
8x8		dropout 0.5
8x8		5x5, 192
8x8		1x1, 192
8x8		1x1, 10 o 100
8x8	pool3	8x8 Med *pooling*
10 o 100		Softmax

Cuadro II
PARÁMETROS DE LA RED VGG

Input		Filtros, canales
28x28		3x3, 64
28x28		3x3, 64
28x28	pool1	2x2 Max *pooling*, stride 2
14x14		3x3, 128
14x14		3x3, 128
14x14	pool2	2x2 Max *pooling*, stride 2
7x7		3x3, 256
7x7		3x3, 256
7x7	pool3	2x2 Max *pooling*, stride 2
4x4		3x3, 512
4x4		3x3, 512
4x4	pool4	2x2 Max *pooling*, stride 2
2x2		3x3, 512
2x2		3x3, 512
2x2	pool5	2x2 Max *pooling*, stride 2
-		Flatten
-		2048
-		512
-		10
-		Softmax

Hemos utilizado los conjuntos de imágenes CIFAR-10, CIFAR-100 [20] y Fashion-MNIST [21]. Los datasets CIFAR-10 y CIFAR-100 contienen 50000 imágenes a color para la fase de entrenamiento y 10000 de test. Tienen 10 y 100 categorías respectivamente, mientras que FMNIST contiene 60000 imágenes en escala de grises para entrenamiento y 10000 de test (10 categorías). Las imágenes de CIFAR-10 y

CIFAR-100 tienen una resolución de 32x32 píxeles, mientras que las de FMNIST de 28x28.

En la tabla III mostramos los resultados obtenidos para la red NiN en los conjuntos de datos CIFAR-10 y CIFAR-100. En la primera fila se muestra el porcentaje de acierto en los conjuntos de test. En la fila Max se muestra el resultado cuando los operadores de todas las capas son *pooling* el máximo. En la fila Med todos los operadores de *pooling* son la media aritmética. Las filas con MPO1 corresponden a cuando entrenamos en todas las capas de *pooling* una media ponderada ordenada. En el caso MPO2 entrenamos un operador por cada canal de cada capa. En la fila MPO1ft, se muestran los resultados cuando entrenamos la red con todos los operadores de *pooling* con la media aritmética y cuando el entrenamiento converge, se sustituyen los operadores de *pooling* por medias ponderadas ordenadas, se fijan los pesos de las capas convolucionales y se entrena durante unas pocas épocas para que se ajusten los pesos de las capas de *pooling*. Este método es similar al conocido *fine tunning*. En el resto de filas el sufijo pl significa que el entrenamiento de los pesos del operador de *pooling* se realiza sin añadir los términos de regularización a la función de coste. De tal manera que los pesos convergen a valores que no cumplen las condiciones para ser una media ponderada (es decir, no tienen que ser positivos ni sumar 1). En la tabla IV se muestran los resultados de la red VGG en FMNIST (en el modelo original todos los operadores de *pooling* son el máximo).

Observando los resultados comprobamos que los modelos en los que se utilizan las MPOs alcanzan una precisión parecida al modelo original. Un poco menor en VGG y un poco mayor en el caso de NiN para CIFAR-100. Cuando se entrenan los pesos libres la precisión es un poco mayor a cuando los pesos cumplen las condiciones de las MPOs. También se comprueba que ajustar un operador de *pooling* por cada canal de cada capa no es necesario, de hecho la precisión de MPO2 es más baja que MPO1 en todos los casos. Por lo tanto, podemos utilizar las MPOs para encontrar modelos de red sin tener que identificar manualmente cuáles son los operadores de *pooling* adecuados para cada capa.

Cuadro III
Resultados experimentales de la red NiN

NIN	CIFAR-10	CIFAR-100
Orig	90.24	58.70
Max	88.76	54.89
Med	90.50	58.75
MPO1	90.34	59.33
MPO2	90.32	58.09
MPO1ft	90.39	59.42
MPO1pl	**90.58**	**59.66**
MPO2pl	90.36	58.77
MPO1pl-ft	90.46	59.62

IV-1. Test robustez: Al añadir más parámetros a la red que se entrenan cabe pensar que obtendremos redes que están más ajustadas y por tanto con menos capacidad de generalización. Para comprobar este hecho hemos realizado un test de robustez

Cuadro IV
Resultados experimentales de la red VGG.

VGG	FMNIST
Max	**94.31**
Mean	92.90
MPO1	93.57
MPO2	92.11
MPO1ft	92.87
MPO1pl	94.25
MPO2pl	92.91
MPO1pl-ft	92.770

[18]. En este test, se rotan las imágenes de test entre -8 y 8 grados y se comprueba la precisión (Figuras 1, 2, 3). Comprobamos que en el caso de NiN, el modelo con MPO es el más robusto a los cambios y en VGG es prácticamente igual al modelo original en el que todos los operadores de *pooling* son el máximo. Por lo tanto, ponderar las activaciones teniendo en cuenta únicamente su valor, provoca que las características representativas de la imagen se propaguen por la red hasta la representación final y mejoren la clasificación.

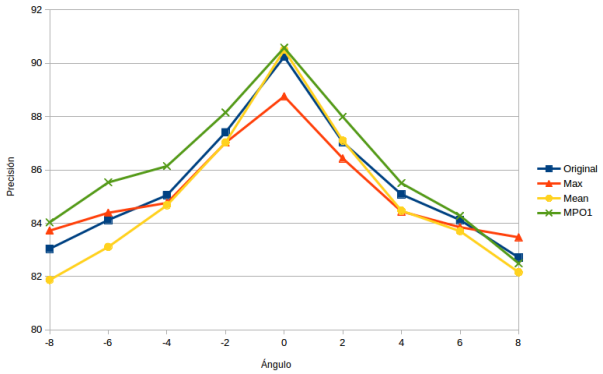

Figura 1. Test de Robustez giro NiN en CIFAR-10.

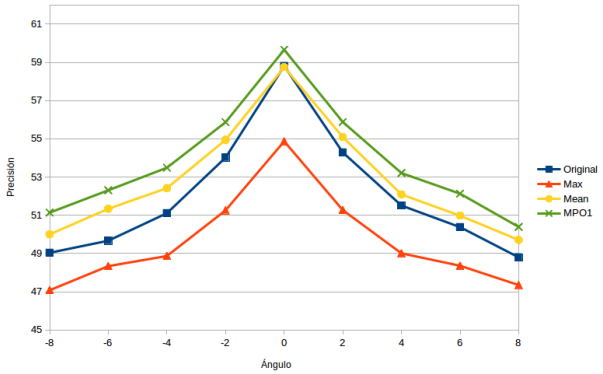

Figura 2. Test de Robustez giro NiN en CIFAR-100.

IV-2. Análisis de los pesos: En la figura 4 vemos los 9 pesos de las dos primeras capas de *pooling* de la red NiN en el caso MPO1 cuando entrenamos la red para CIFAR-10 y CIFAR-100 respectivamente. En el modelo original los

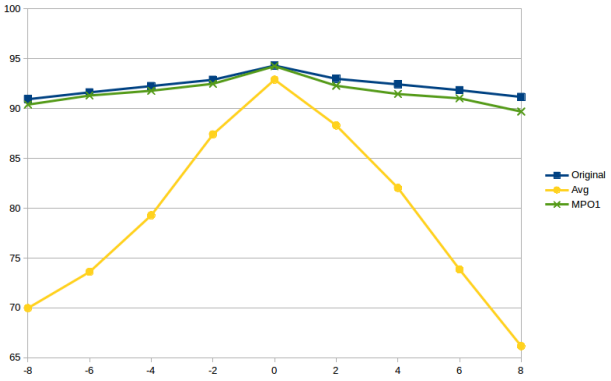

Figura 3. Test Robustez giro VGG en FMNIST.

operadores utilizados son el máximo para la primera capa y la media aritmética para la segunda capa de *pooling*. El coeficiente 1 multiplica al valor mayor y así sucesivamente. En el caso de CIFAR-100, vemos que todos los pesos son prácticamente iguales (ligeramente descendientes), por lo tanto el resultado de estos operadores será muy similar a la media aritmética. Sin embargo, al entrenar la red con CIFAR-10 los coeficientes de cada capa son diferentes. Aunque en ambos casos se obtienen coeficientes mayores para los valores más pequeños, es decir que el valor resultante será incluso más pequeño que la media.

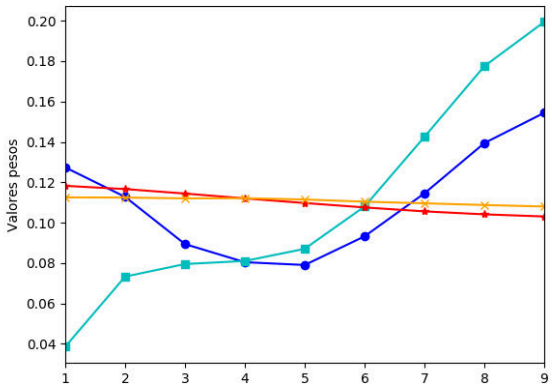

Figura 4. Azul: pesos de la capa pool1 entrenando en CIFAR-10; Cyan pesos de la capa pool2 entrenando en CIFAR-10; Rojo: pesos de la capa pool1 entrenando en CIFAR-100; Naranja pesos de la capa pool2 entrenando en CIFAR-100.

En la figura 5 vemos los 64 pesos de la tercera capa de *pooling* de la red NiN en el caso MPO1 cuando entrenamos la red para CIFAR-10 y CIFAR-100 respectivamente. Los pesos obtenidos con CIFAR-10son similares a las capas pool1 y pool2, los pesos más grandes multiplican a los valores pequeños. Esto significa que se da importancia a las activaciones de menor valor. Para el caso de CIFAR-100 los pesos son parecidos a la media aritmética (los valores negativos se deben a que el factor de regularización no ha penalizado lo suficiente y el entrenamiento ha convergido sin llevar ese término a cero).

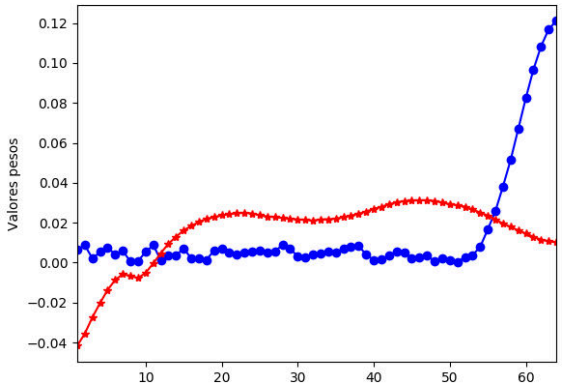

Figura 5. Azul: pesos de la capa pool3 entrenando en CIFAR-10; Rojo: pesos de la capa pool3 entrenando en CIFAR-100.

Al fijarnos en la precisión obtenida por los modelos vemos que MPO1 para CIFAR-100 es bastante mejor que el modelo original y en CIFAR-10 es un poco peor.

En la figura 6 mostramos los 4 pesos obtenidos para las 5 capas de *pooling* de la red VGG (en el modelo original todos los operadores son el máximo). En este caso las capas 1, 3, 4 y 5 convergen a un operador que da más peso a los valores más altos. Sin embargo, los pesos de la segunda capa dan más importancia a los valores pequeños. Puede ser que este sea el motivo de que la precisión del modelo MPO1 es menor que la del modelo original. Por lo tanto, nos queda por comprobar si al lanzar más entrenamientos obtenemos modelos que obtienen una precisión mejor y cuales son los valores de los pesos.

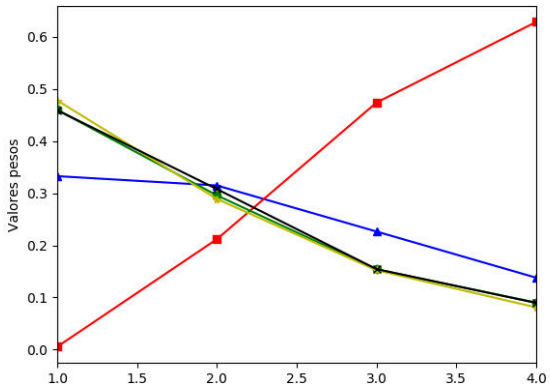

Figura 6. Pesos de las capas pool1 (en azul), pool2 (en rojo), pool3 (en verde), pool4 (en amarillo) y pool5 (en negro) de la red VGG (MPO1) para FMNIST.

IV-3. Análisis de los tiempos: El mayor problema de las medias ponderadas ordenadas es la necesidad de ordenar el vector de valores que queremos agregar. Al introducir esta ordenación en la fase de entrenamiento, hace que los tiempos de

computación crezcan. En el tabla V mostramos la proporción de tiempo de los diferentes modelos con respecto al original. El costo en tiempo del modelo MPO1 es demasiado grande y probablemente sea más rápido probar dos o tres combinaciones de operadores de *pooling* basándonos en nuestra experiencia. Sin embargo, si entrenamos con todos los operadores siendo la media y luego realizamos el ajuste de los pesos de los operadores, simplemente doblamos el tiempo del modelo original y obtenemos un modelo robusto y con una precisión similar al mejor modelo. Por lo tanto, si utilizamos esta metodología, el uso de medias ponderadas puede utilizarse en casos reales en los que no conozcamos la arquitectura correcta e introducimos en la fase de entrenamiento la selección del operador de *pooling*.

Cuadro V
TIEMPOS DE ENTRENAMIENTO

NiN	Orig	Max	Med	MP01	MPO1ft
Tiempo	1	1.02x	0.95x	4.06x	1.69x
VGG	Orig	Max	Med	MPO1	MPO1ft
Tiempo	1	1	0.83x	7.26x	2.11x

V. Conclusiones y trabajos futuros

En este trabajo hemos propuesto aprender los pesos de medias ponderadas ordenadas para adaptar automáticamente el operador de *pooling*. De esta forma eliminamos un hiperparámetro a la hora de seleccionar la estructura de la red. En las pruebas experimentales hemos comprobado que añadiendo muy pocos parámetros, aprendiendo un único operador por capa, el *pooling* basado en medias ponderadas obtiene una precisión similar a los modelos originales, teniendo la red obtenida una mayor precisión frente a variaciones de las imágenes de test. Como trabajo futuro falta por analizar las condiciones iniciales y los parámetros del entrenamiento para comprobar que se ha convergido a la mejor solución o si es posible obtener otros pesos que mejoren la precisión de la red. Otro posible estudio consiste en relacionar el operador de *pooling* a la zona de la imagen original. Teniendo en cuenta que la estructura de la imagen original se mantiene a lo largo de las capas de la red, en lugar de aprender un operador por cada canal, el objetivo es aprender un operador por cada zona.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto TIN2016-77356-P (AEI/FEDER, UE).

Referencias

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, Dec 1989.

[2] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0588–0592, April 2017.

[3] Y. L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2559–2566, 2010.

[4] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pp. 437–478. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[5] R. Yager, "On ordered weighted averaging aggregation operators in multi criteria decision making," *IEEE Trans. Syst. Man Cybern.*, vol. 18, no. 1, pp. 183–190, 1988.

[6] F. Perronnin, M. Douze, S. Jorge, C. Schmid, F. Perronnin, M. Douze, S. Jorge, and P. Patrick, "Aggregating local image descriptors into compact codes Aggregating local image descriptors into compact codes," 2012.

[7] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6315 LNCS, no. PART 5, pp. 141–154, 2010.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[11] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[12] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2169–2178, 2006.

[13] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," *Icml*, pp. 111–118, 2010.

[14] P. Koniusz, F. Yan, and K. Mikolajczyk, "Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection," *Computer Vision and Image Understanding*, vol. 117, pp. 479–492, may 2013.

[15] C. Wang and K. Huang, "How to use Bag-of-Words model better for image classification," *Image and Vision Computing*, vol. 38, pp. 65–74, 2015.

[16] M. Pagola, J. I. Forcen, E. Barrenechea, J. Fernández, and H. Bustince, "A study on the cardinality of ordered average pooling in visual recognition," in *Pattern Recognition and Image Analysis* (L. A. Alexandre, J. Salvador Sánchez, and J. M. F. Rodrigues, eds.), (Cham), pp. 437–444, Springer International Publishing, 2017.

[17] M. Pagola, J. I. Forcen, E. Barrenechea, C. Lopez-Molina, and H. Bustince, "Use of owa operators for feature aggregation in image classification," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, July 2017.

[18] C. Y. Lee, P. Gallagher, and Z. Tu, "Generalizing pooling functions in cnns: Mixed, gated, and tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 863–875, April 2018.

[19] G. Beliakov, H. B. Sola, and T. C. Sánchez, *A Practical Guide to Averaging Functions*. Springer, 2016.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.