Reemplazo de la función de pooling de Redes Neuronales Convolucionales por combinaciones lineales de funciones crecientes

1º Iosu Rodríguez-Martínez
dept. de Estadística, Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
iosu.rodriguez@unavarra.es

2º Julio Lafuente
dept. de Estadística, Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
lafuente@unavarra.es

3º Mikel Sesma
dept. de Estadística, Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
mikel.sesma@unavarra.es

4º Francisco Herrera
dept. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada
Granada, España
herrera@decsai.ugr.es

5º Pablo Ursúa-Medrano
dept. de Estadística, Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
pablo.ursua@unavarra.es

6º Humberto Bustince
dept. de Estadística, Informática y Matemáticas
Universidad Pública de Navarra
Pamplona, España
bustince@unavarra.es

***Abstract*—Las redes convolucionales llevan a cabo un proceso automático de extracción y fusión de características mediante el cual obtienen la información más relevante de una imagen dada. El proceso de submuestreo mediante el cual se fusionan características localmente próximas, conocido como "pooling", se lleva a cabo tradicionalmente con funciones sencillas como el máximo o la media aritmética, ignorando otras opciones muy populares en el campo de la teoría de agregaciones. En este trabajo proponemos reemplazar dichas funciones por otra serie de órdenes estadísticos, así como por la integral de Sugeno y una nueva generalización de la misma. Además, basándonos en trabajos que emplean la combinación convexa del máximo y la media, presentamos una nueva capa que permite combinar varias de las nuevas agregaciones, mejorando sus resultados individuales.**

***Index Terms*—Redes Neuronales Convolucionales, Función de pooling, Combinaciones Lineales, Integral de Sugeno Generalizada**

I. Introducción

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) han cobrado gran popularidad desde que Krizhevsky et al. ganaran la competición ImageNet de 2012 mediante su modelo AlexNet [1], y actualmente constituyen el estado del arte en aplicaciones de tratamiento de imagen.

Inicialmente, el foco se puso en el desarrollo de arquitecturas más y más profundas que buscaban mejorar su capacidad de clasificación mediante la adición de más parámetros [2], [3]. Actualmente, el objetivo contrario ha crecido también en interés, puesto que el coste de entrenar estos modelos puede resultar prohibitivo en ciertos dominios de aplicación (p. e. "smartphones" o vehículos autónomos). Con este objetivo en mente, se han ido introduciendo arquitecturas más modestas aunque competitivas [4]–[6], y estrategias como la transferencia de conocimiento o el podado de modelos se han postulado como estrategias valiosas [7], [8].

Pese a la cantidad de investigación desarrollada en torno a este campo apodado "Deep Learning", las CNNs siguen operando en base a las funciones clásicas en torno a las cuales fueron diseñadas: la convolución, que extrae características locales a partir de una imagen; y el "pooling", que agrega características próximas espacialmente para reducir la dimensionalidad de la salida generada. A pesar de que, con respecto a esta última, se han presentado algunas alternativas [9]–[11], la mayoría de los modelos siguen empleando el máximo o la media aritmética.

En [12] los autores presentan la combinación convexa del máximo y la media aritmética como una función de pooling que aprovecha la información proporcionada por cada una mejorando su comportamiento individual. Creemos que este hecho, combinado a los avances en teoría de agregación de los últimos años, pueden ayudarnos a generar funciones que superen el rendimiento de las opciones tradicionales.

En este artículo, reemplazamos la función de pooling de

varios modelos de CNN por combinaciones lineales crecientes de estas funciones así como de otros órdenes estadísticos y de la integral de Sugeno, que permite modelar la interacción entre los valores por medio del concepto de medida difusa. Nuestros resultados demuestran que combinar distintas funciones puede mejorar su comportamiento individual y que, en particular, una generalización de esta integral tiende a obtener los mejores resultados cuando se combina con la media aritmética o el máximo.

El artículo se estructura de la siguiente forma: en la sección 2 se recuerdan algunos conceptos relacionados con las CNNs y las funciones de agregación propuestas; la sección 3 compone el estudio teórico que demuestra las condiciones a cumplir por nuestras combinaciones con el fin de comportarse como una función creciente; en la sección 4 demostramos la utilidad del método al emplearlo en varias arquitecturas de Deep Learning estándares en la literatura; la sección 5 cierra el artículo con algunas conclusiones y líneas futuras a explorar.

II. Preliminares

A. Redes Neuronales Convolucionales

Las CNNs son un tipo de red neuronal artificial diseñada para tratar con datos cuya información local es relevante, como series temporales [13] o imágenes [1]. Estos modelos se componen de dos partes diferenciadas: una primera que se encarga de extraer las características más relevantes a partir de los datos de entrada de manera automatizada y una segunda, que emplea estas características para resolver un problema específico de su dominio de aplicación. En el caso a tratar, el de la clasificación de imagen, se tratará de un clasificador que, para una imagen de entrada, devolverá la probabilidad de pertenencia a varias posibles clases predefinidas.

Es el primer proceso el que da su nombre a las CNN, dado que la extracción de características se produce mediante la convolución de varios filtros de características sobre una entrada dada. Por entrada entendemos una matriz $X \in \mathbb{R}^{h\times w\times c}$, que puede representar, bien una imagen de tamaño $h \times w$, donde cada pixel $X_{ij} \in \mathbb{R}^c$ cuenta con c canales de color ($c = 1$ para imágenes en gris, $c = 3$ para imágenes en color), o bien una imagen de características, donde cada canal $X^d \in \mathbb{R}^{h\times w}$ representa la presencia o ausencia de una característica concreta en las distintas partes de la imagen. Cada uno de esos canales habrá sido producido, a su vez, por la convolución de otro filtro de características que representará una característica concreta sobre otra entrada anterior.

Para ser más precisos, la salida producida por un filtro de características $W \in \mathbb{R}^{k_1\times k_2\times c}$ para una ventana $k_1 \times k_2$ (con $k_1, k_2 \in \mathbb{N}$ números impares) de una entrada $X \in \mathbb{R}^{h\times w\times c}$ centrada en el pixel $X_{m,n}$ se calcula mediante la siguiente expresión:

$$Y_{m,n} = \sum_{d=1}^{c}\sum_{j=1}^{k_2}\sum_{i=1}^{k_1} W_{ij}^d \cdot X^d_{m-1+i-\frac{k_1-1}{2},n-1+j-\frac{k_2-1}{2}} \quad (1)$$

Los valores de estos filtros se "aprenden" de manera análoga a los pesos de un perceptrón multicapa: se inicializan aleatoriamente y sus valores se optimizan de manera iterativa mediante alguna variación del algoritmo de descenso por gradiente, comúnmente el descenso por gradiente estocástico [14].

Las CNN se componen de varias capas convolucionales (además de otros tipos de capas) que operan secuencialmente sobre los datos de entrada tal que la salida generada por la capa l-ésima se introduce como entrada a la capa $(l+1)$-ésima. Cada capa convolucional está formada por c_l filtros de características que se aplican a la entrada recibida, generando c_l imágenes de características que se envían a la siguiente capa.

Pese a que el objetivo del extractor de características es el de reducir la imagen, el proceso anterior tiene el efecto contrario. Para lograr esta reducción se añaden capas de pooling en distintos puntos de la red. Cuando una capa de pooling recibe una entrada $X \in \mathbb{R}^{h\times w\times c}$, cada canal independiente se separa en ventanas disjuntas de tamaño $k_1 \times k_2$. Los valores de cada ventana se agregan mediante alguna función, tradicionalmente el máximo o la media aritmética.

B. Funciones crecientes

Aquí recordamos y fijamos la notación para algunas funciones crecientes que usaremos más adelante.

De ahora en adelante, asumimos que $2 \leq n \in \mathbb{N}$, $1 \leq r \in \mathbb{N}$.

Decimos que una función $\mathbf{A}\colon \mathbb{R}^n \to \mathbb{R}$ es creciente si $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \leq \mathbf{y}$ implica que $\mathbf{A}(\mathbf{x}) \leq \mathbf{A}(\mathbf{y})$

1) Estadísticos de orden:

Notación II.1. Fijamos $N = \{1, \ldots, n\}$ y Σ_n como el grupo de todas las posibles permutaciones de N. Si $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ y $\sigma \in \Sigma_n$, lo denotamos mediante $\mathbf{x}_\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$; si $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}$, lo denotamos mediante $\sigma \in \mathbf{x}_{(\nearrow)}$.

Sea $r \in \{1, \ldots, n\}$. Denotamos mediante $\mathbf{OS}_r$ al r-ésimo estadístico de orden, esto es, la función $\mathbf{OS}_r\colon \mathbb{R}^n \to \mathbb{R}$ dada por, si $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $\mathbf{OS}_r(\mathbf{x}) = x_{\sigma(r)}$, donde $\sigma \in \mathbf{x}_{(\nearrow)}$. En particular $\mathbf{OS}_n$ es el máximo y $\mathbf{OS}_1$ el mínimo.

2) Media aritmética: Denotamos mediante $\mathbf{AM}$ a la media aritmética, esto es la función $\mathbf{AM} : \mathbb{R}^n \to \mathbb{R}$, dada por $\mathbf{AM}(\mathbf{x}) = (x_1 + \cdots + x_n)/n$ si $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$.

3) Integral de Sugeno y generalizaciones:

Definición II.1. Una medida difusa en N es una relación $\nu\colon 2^N \to [0, +\infty)$ tal que

1) $\nu(\emptyset) = 0$ y
2) $S \subseteq T \subseteq N$ implica $\nu(S) \leq \nu(T)$.

Definición II.2. La *integral de Sugeno* asociada a una medida difusa ν es la función

$$\mathbf{S}_\nu\colon \mathbb{R}^n \to \mathbb{R}$$

dada por, para $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$,

$$\mathbf{S}_\nu(\mathbf{x}) = \mathbf{max}\left(\mathbf{min}(x_{\sigma(1)}, \nu(N_1^\sigma)), \ldots, \mathbf{min}(x_{\sigma(n)}, \nu(N_n^\sigma))\right),$$

donde $\sigma \in \mathbf{x}_{(\nearrow)}$ y $N_i^\sigma = \{\sigma(i), \ldots, \sigma(n)\}$.

Una posible extensión que utilizaremos más adelante fue presentada en [15].

III. Combinación de funciones crecientes

A. Construcción de funciones crecientes mediante combinaciones lineales de funciones crecientes

A continuación estudiamos combinaciones lineales de funciones crecientes de $\mathbb{R}^n \to \mathbb{R}$, en algunos casos de $[0,+\infty)^n \to [0,+\infty)$.

Fijamos $\mathbf{0} = (0,\dots,0)$, $\mathbf{1} = (1,\dots,1)$, $\mathbf{e}_i = (0,\dots,\underset{i}{1},\dots,0)$.

Notación III.1. Sean $\mathbf{A}_1,\dots,\mathbf{A}_r\colon \mathbb{R}^n \to \mathbb{R}$ funciones crecientes. Fijamos

$$\mathcal{I}(\mathbf{A}_1,\dots,\mathbf{A}_r) = \\ = \{(\alpha_1,\dots,\alpha_r) \in \mathbb{R}^n \mid \sum_{i=1}^{r} \alpha_i \mathbf{A}_i\colon \mathbb{R}^n \to \mathbb{R} \text{ es una función creciente}\}.$$

B. Combinaciones de estadísticos de orden y la media aritmética

Para que la combinación lineal de estadísticos de orden sea una función creciente, se debe cumplir la siguiente propiedad

Proposición III.2. *Sea $i_1,\dots,i_r \in N$, $i_1 < \cdots < i_r$. Entonces*

$$\mathcal{I}(\mathbf{OS}_{i_1},\dots,\mathbf{OS}_{i_r}) = \{(\alpha_1,\dots,\alpha_r) \mid \alpha_1,\dots,\alpha_r \geq 0\}$$

Por definición $\mathbf{AM} = \frac{1}{n}(\mathbf{OS}_1 + \cdots + \mathbf{OS}_n)$, tenemos

Proposición III.3. *Sea $i_1,\dots,i_r \in N$, $i_1 < \cdots < i_r$, $r < n$. Entonces*

$$\mathcal{I}(\mathbf{AM},\mathbf{OS}_{i_1},\dots,\mathbf{OS}_{i_r}) = \\ = \{(\alpha,\beta_1,\dots,\beta_r) \mid \alpha,\alpha+n\beta_1,\dots,\alpha+n\beta_r \geq 0\}.$$

Análogamente

Proposición III.4. *Tenemos*

$$\mathcal{I}(\mathbf{AM},\mathbf{OS}_1,\dots,\mathbf{OS}_n) = \\ = \{(\alpha,\beta_1,\dots,\beta_n) \mid \alpha+n\beta_1,\dots,\alpha+n\beta_n \geq 0\}.$$

C. Combinaciones con la Integral de Sugeno

Escribimos $\nu_S = \nu(S)$ para $S \subseteq N$, $\mathbf{x}_\sigma = (x_{\sigma(1)},\dots,x_{\sigma(n)})$, $\nu_i^\sigma = \nu(N_i^\sigma)$, $N_i = \{i,\dots,n\}$ y $\nu_i = \nu(N_i)$. Por lo tanto $\mathbf{S}_\nu(\mathbf{x}) = \bigvee_{i=1}^n (x_{\sigma(i)} \wedge \nu_i^\sigma)$. Sabemos que

$$S_\nu(x_1,\dots,x_n) = \bigvee_{S\subseteq N} [\nu_S \wedge (\bigwedge_{i\in S} x_i)].$$

Lemma III.5. *Si $S \subseteq N$, entonces existe $\sigma \in \Sigma_n$, $i \in N$ tal que $N_i^\sigma = S$ (y por tanto, para una medida difusa $\nu\colon 2^N \to [0,+\infty)$, $\nu_i^\sigma = \nu_S$.).*

Definición III.6. Diremos que una medida difusa $\nu\colon 2^N \to [0,+\infty)$ es *estricta en $k \in N$* si, o bien $k = n$ o existe $\sigma \in \Sigma_n$ tal que $\nu_k^\sigma > \nu_{k+1}^\sigma$. Diremos que ν es *estricta* si es esctricta en k para todo $k \in N$.

Proposición III.7. *Sea $\nu\colon 2^N \to [0,+\infty)$ una medida difusa. Si*

$$\alpha_1,\dots,\alpha_n,\alpha_1+\beta,\dots,\alpha_n+\beta \geq 0,$$

entonces $\alpha_1\mathbf{OS}_1 + \cdots + \alpha_n\mathbf{OS}_n + \beta\mathbf{S}_\nu$ es creciente. Si ν es estricta en $k \in N$, entonces $\alpha_k + \beta \geq 0$; por lo tanto si ν es estricta,

$$\mathcal{I}(\mathbf{OS}_1,\dots,\mathbf{OS}_n,\mathbf{S}_\nu) = \\ = \{(\alpha_1,\dots,\alpha_n,\beta) \mid \alpha_1,\dots,\alpha_n,\alpha_1+\beta,\dots,\alpha_n+\beta \geq 0\}.$$

Proposición III.8. *Sea $\nu\colon 2^N \to [0,+\infty)$ una medida difusa. Tenemos*

$$\mathcal{I}(\mathbf{AM},\mathbf{S}_\nu) = \{(\alpha,\beta) \in \mathbb{R}^2 \mid \alpha,\alpha+n\beta \geq 0\}.$$

D. Generalizaciones de la integral de Sugeno y combinaciones

Definición III.9. Sea $\mathbb{U}$ un subconjunto conexo de $\mathbb{R}$ tal que $0 \in \mathbb{U}$. Una *medida $\mathbb{U}$-difusa en N* es una función $\nu\colon 2^N \to \mathbb{U}$ tal que

1) $\nu(\emptyset) = 0$ y
2) $S \subseteq T \subseteq N$ implica $\nu(S) \leq \nu(T)$.

Obviamente $\operatorname{im}\nu \subseteq [0,+\infty)$, pero necesitamos esta suposición más general en $\mathbb{U}$ para lo siguiente.

Definición III.10. Sean $\mathbb{U}$ y $\mathbb{I}$ dos subconjuntos conexos de $\mathbb{R}$ tal que $0 \in \mathbb{U} \subseteq \mathbb{I}$. Sea $\nu\colon 2^N \to \mathbb{U}$ una medida $\mathbb{U}$-difusa.

Diremos que las funciones $\mathrm{F}\colon \mathbb{I}\times\mathbb{U} \to \mathbb{I}$ y $\mathrm{G}\colon \mathbb{I}^n \to \mathbb{U}$ son *ν-admisibles* si la función $\mathbf{A}\colon \mathbb{I}^n \to \mathbb{I}$ dada por

$$\mathbf{A}(x_1,\dots,x_n) = \mathrm{G}(\mathrm{F}(x_{\sigma(1)},\nu(N_1^\sigma)),\dots,\mathrm{F}(x_{\sigma(n)},\nu(N_n^\sigma))),$$

donde $\sigma \in \mathbf{x}_{(\nearrow)}$ y $N_i^\sigma = \{\sigma(i),\dots,\sigma(n)\}$, están bien definidas (esto es, el resultado no cambia para $\sigma,\sigma' \in \mathbf{x}_{(\nearrow)}$) y son crecientes. Fijamos entonces $\mathbf{A} = \mathbf{A}(\mathrm{F},\mathrm{G},\nu)$ y la nombramos *$(\mathrm{F},\mathrm{G},\nu)$-función Sugeno-semejante*

Nos vamos a centrar en concreto en el caso $\mathbf{D}_\nu = \mathbf{A}(\Pi,\Sigma,\nu)$

Para el resto de la sección $\nu\colon 2^N \to \mathbb{R}$ es una medida difusa simétrica.

Resuta inmediato observar que $\mathbf{D}_\nu(c\,\mathbf{x}) = c\,\mathbf{D}_\nu(\mathbf{x})$ para todo $\mathbf{x} \in \mathbb{R}^n$, $c \in [0,+\infty)$, esto es, $\mathbf{D}_\nu$ es homogénea positiva.

Proposición III.11. *Las siguientes afirmaciones son equivalentes*

1) $\nu_1 + \cdots + \nu_n = 1$.
2) $\mathbf{D}_\nu$ *es idempotente.*
3) $\mathbf{D}_\nu$ *es invariante a traslación.*

Por supuesto esta situación no aparece si $\nu_1 = 1$, excepto en el caso trivial en que $\nu_2 = \cdots = \nu_n = 0$.

Proposición III.12. *Sea $M \subseteq N$ y fijemos $M' = N \setminus M$. $\sum_{i\in M} \alpha_i\mathbf{OS}_i + \beta\mathbf{D}_\nu$ es creciente si y solo si $\alpha_i + \beta\nu_i \geq 0$ si $i \in M$ y $\beta\nu_i \geq 0$ si $i \in M'$.*

Corolario III.13. *$\alpha\mathbf{AM} + \beta\,\mathbf{D}_\nu$ es creciente si y solo si, o bien $\beta \geq 0$ y $\alpha + n\beta\nu_n \geq 0$, o $\beta \leq 0$ y $\alpha + n\beta\nu_1 \geq 0$.*

Proposición III.14. *Sea ν estricta. Entonces $\alpha\,\mathbf{D}_\nu + \beta\,\mathbf{S}_\nu$ es creciente si y solo si $\alpha \geq 0$ y $\alpha\nu_n + \beta \geq 0$.*

E. Capa CombPool: Combinación de funciones crecientes como función de pooling

Ahora procedemos a introducir un reemplazo para las capas de pooling máximo y media aritmética clásicas, a la que llamamos capa combinación de poolings (o, abreviada, capa CombPool). El proceso es el siguiente: inicialmente escogemos r funciones crecientes $\mathbf{A}_1, \ldots, \mathbf{A}_r$ tales que $\mathbf{A}_i : \mathbb{R}^n \to \mathbb{R}\ \forall i = 1, \ldots, r$ y generamos r coeficientes $\alpha_1, \alpha_2, \ldots, \alpha_r \in \mathbb{R}, \alpha_i \geq 0\ \forall i = 1, \ldots, r$. Al reducir los $\mathbf{x} = (x_1, \ldots, x_n)$ valores de cada ventana disjunta de tamaño $k_1 \times k_2$, con $k_1 \cdot k_2 = n$ de un canal de una imagen de características X^c, utilizamos las n funciones crecientes, generando n salidas diferentes $y_1, y_2, \ldots, y_n$. La salida producida por nuestra capa se termina de calcular mediante la siguiente combinación de funciones crecientes:

$$y = \sum_{i=1}^{r} \alpha_i^2 \cdot \mathbf{A}_i(\mathbf{x}) = \sum_{i=1}^{r} \alpha_i^2 \cdot y_i \tag{2}$$

La Fig. 1 ilustra el proceso anterior mediante un ejemplo.

Observe que, con el fin de garantizar que la función resultante sea creciente, no ponderamos cada salida y_i directamente con el valor de α_i, sino con el de su potencia al cuadrado. De este modo garantizamos que la diferenciabilidad y monotonía se conservan para todas las posibles combinaciones.

A la hora de escoger un conjunto de coeficientes existen distintas opciones, tal y como se presenta en [12]: el mismo conjunto para todas las ventanas y canales de la misma capa; un conjunto diferente para cada ventana diferente (compartido por todos los canales); un conjunto diferente por cada canal (compartido por todas las ventanas de cada canal); un conjunto diferente por cada ventana de cada canal. Aunque hemos hecho pruebas con todas las variantes, consideramos que los resultados no difieren lo suficiente como para justificar reportar todas las combinaciones posibles, y nos hemos decantado por la opción de utilizar un conjunto de coeficientes distinto para cada canal, un punto intermedio entre capacidad de representabilidad del modelo y número de parámetros adicionales.

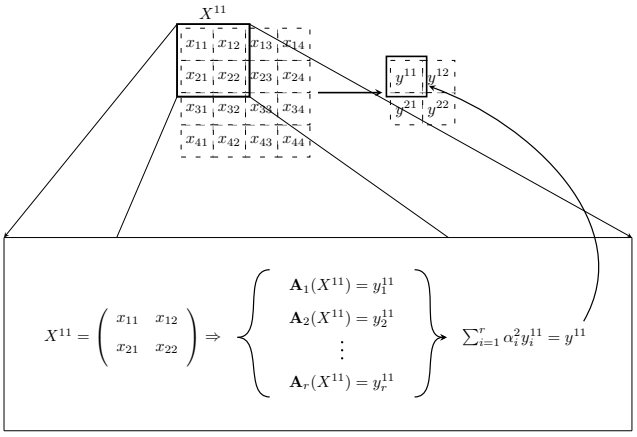

Fig. 1. Ejemplo de la combinación de funciones de pooling. Calculamos r reducciones por cada ventana, que posteriormente ponderamos mediante r parámetros $\alpha \in \mathbb{R}$.

IV. Experimentos

A. Marco experimental

En esta sección ilustramos la utilidad de la capa CombPool evaluando el rendimiento de varios modelos en los que reemplazamos sus capas de pooling por nuestra propuesta. A continuación introducimos brevemente cada una de las arquitecturas y el dataset tenido en cuenta para la evaluación.

1) Conjunto de datos: Todas las pruebas reportadas se han llevado a cabo sobre el dataset CIFAR10 [16]. Este dataset está compuesto por 50000 imágenes a color de tamaño 32x32 píxeles. Los ejemplos están completamente balanceados entre 10 clases que representan objetos del mundo real y animales.

2) Arquitecturas de CNN:

- **Arquitectura 1: LeNet-5**
  La estructura de esta red, presentada en [17] por LeCun et al. es la que se ha tomado como CNN canónica. Está compuesta por dos bloques de capas de convolución y pooling (tradicionalmente pooling media), seguido de un perceptrón de 3 capas que actúa como clasificador. Hemos fijado la función ReLU como función de activación para las capas intermedias, y la función Softmax para la capa de salida. También hemos intercalado capas de "Batch Normalization" tras cada capa de pooling [18]. Los detalles de la arquitectura se detallan en la Tabla I.
- **Architecture 2: Network in Network**
  Esta arquitectura es idéntica a la presentada en [12], y la empleamos por lo similar entre nuestra propuesta y la suya. Se basa, a su vez, en el concepto de "Network in Network" que reemplaza las capas de convolución por perceptrones multicapa y el clasificador final por una capa de Pooling Promedio Global [19].
  Adicionalmente, esta variante del modelo incluye "supervisión de capas ocultas" [20], una estrategia que intercala varios clasificadores en distintos puntos de la red para reforzar el flujo de los gradientes durante el proceso de entrenamiento y mitigar el problema de "fuga de gradiente" que tienden a sufrir los modelos profundos. Los detalles de la arquitectura pueden consultarse en la Tabla II.
- **Architecture 3: DenseNet**
  La última red que hemos empleado es una adaptación directa de la arquitectura DenseNet presentada en [6]. Las DenseNet iteran sobre el concepto de conexión identidad que las arquitecturas ResNet incluyen como método para evitar problemas de "fuga de gradiente" [3], conectando la salida de una capa con la entrada de todas las siguientes.
  Una DenseNet está formada por una serie de "bloques densos", formados por repeticiones de capas de Batch Normalization, ReLU, Convolución y, opcionalmente, Dropout [21]. Los autores se refieren mediante el factor k al número de imágenes de características que cada capa de convolución añade al conocimiento global de la red. Como medida para evitar que los filtros de capas

siguientes tengan que ser cada vez más grandes, se anteponen convoluciones 1×1 para reducir el número de canales de la entrada de la siguiente convolución a un factor fijo.

Las capas de pooling se sitúan entre bloques densos para reducir la dimensionalidad de las características extraídas. Además, tras estas capas de pooling se pueden introducir nuevamente convoluciones 1×1 para reducir el número de canales del conocimiento global de la red G por un factor θ. La Tabla III detalla la arquitectura.

TABLE I
DESCRIPCIÓN DE LA ARQUITECTURA 1.

Layer type	Output size	Kernel size
Conv1	$32 \times 32 \times 64$	3×3
ReLU	$32 \times 32 \times 64$	-
Pool1	$16 \times 16 \times 64$	2×2
BatchNorm1	$16 \times 16 \times 64$	-
Conv2	$16 \times 16 \times 64$	3×3
ReLU	$16 \times 16 \times 64$	-
Pool2	$8 \times 8 \times 64$	2×2
BatchNorm2	$8 \times 8 \times 64$	-
Flatten	4096	-
MLP1	384	-
MLP2	192	-
MLP3	10	-

B. Estudio experimental

Hemos probado a reemplazar las funciones de pooling por las funciones individuales estudiadas previamente así como por capas CombPool que combinan dichas funciones. Además del pooling media y máximo, hemos considerado el mínimo, la mediana, la integral de Sugeno y la generalización $\mathbf{D}_\nu$. La tasa de acierto media sobre 5 ejecuciones independientes de cada modelo se reporta en las Tablas IV y V.

Es interesante observar que los mejores resultados se obtienen al utilizar la función $\mathbf{D}_\nu$. El mejor global para todas las arquitecturas utiliza dicha función y, para las dos últimas, todas las combinaciones que la incluyen superan la tasa de acierto media obtenida con el resto de funciones. Curiosamente, pese a que la arquitectura 1 y 3 ofrecen resultados pobres al emplear esta función de manera individual, la arquitectura 2 obtiene su mejor resultado en este caso, lo que parece indicar que la mejor función es dependiente de la arquitectura.

Si nos fijamos en la arquitectura 2, la empleada en [12], vemos que varias de las funciones que incluyen a la integral de Sugeno superan a la propuesta presentada en dicho artículo.

Otra conclusión interesante es que los modelos más complejos parecen tener una mayor capacidad de adaptación a las funciones menos comunes. En el caso de la arquitectura 1, media y máximo ofrecen un rendimiento muy superior al del resto de funciones individuales, y para que la estrategia de combinarlas surta efecto al menos una de las dos debe estar presente. Los modelos más complejos no sufren este problema, probablemente dado que su mayor número de parámetros les permite incorporar más fácilmente estas nuevas funciones.

TABLE II
DESCRIPCIÓN DE LA ARQUITECTURA 2. LA SALIDA DE LAS CAPAS "HLSUPERVISION" NO SE USAN COMO ENTRADA DE LA CAPA SIGUIENTE. ESTAS CAPAS RECIBEN SU ENTRADA A PARTIR DE LA ANTERIOR A LA CAPA "HLSUPERVISION".

Layer type	Output size	Kernel size
Conv1	$32 \times 32 \times 128$	3×3
ReLU	$32 \times 32 \times 128$	-
HLSupervision1	10	-
Conv2	$32 \times 32 \times 128$	3×3
ReLU	$32 \times 32 \times 128$	-
HLSupervision2	10	-
Mlpconv1	$32 \times 32 \times 128$	1×1
Pool1	$16 \times 16 \times 128$	3×3
Conv3	$16 \times 16 \times 192$	3×3
ReLU	$16 \times 16 \times 192$	-
HLSupervision3	10	-
Conv4	$16 \times 16 \times 192$	3×3
ReLU	$16 \times 16 \times 192$	-
HLSupervision4	10	-
Mlpconv2	$16 \times 16 \times 192$	1×1
Pool2	$8 \times 8 \times 192$	3×3
Conv5	$8 \times 8 \times 256$	3×3
ReLU	$8 \times 8 \times 256$	-
HLSupervision5	10	-
Conv6	$8 \times 8 \times 256$	3×3
ReLU	$8 \times 8 \times 256$	-
HLSupervision6	10	-
Mlpconv3	$8 \times 8 \times 256$	1×1
Mlpconv4	$8 \times 8 \times 10$	1×1
GlobalAvgPool	10	8×8

V. CONCLUSIONES Y LÍNEAS FUTURAS

En este artículo hemos presentado las condiciones necesarias para garantizar que la combinación de funciones crecientes genere una función creciente. Además, presentamos una capa de pooling alternativa, la capa CombPool que hace uso de estas combinaciones, cuya aplicabilidad hemos probado sobre varias arquitecturas de Deep Learning de distinta complejidad.

Así mismo, hemos comprobado que la aplicación de funciones no estándar en la literatura, en concreto una generalización de la integral de Sugeno mejora en general el funcionamiento de los modelos más complejos, por lo que recomendamos su aplicación en modelos profundos de muchas capas.

A futuro, nos gustaría probar la efectividad de otras funciones de la teoría de agregaciones y su efecto al combinarlas en la capa CombPool con las ya exploradas.

AGRADECIMIENTOS

Este trabajo ha sido financiado por los proyectos de investigación PID2019-108392GB-I00 (AEI/10.13039/501100011033) de la Agencia Estatal de Investigación, PC095-096 FUSIPROD y P18-FR-4961.

REFERENCIAS

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

TABLE III
DESCRIPCIÓN DE LA TERCERA ARQUITECTURA. EN NUESTROS EXPERIMENTOS FIJAMOS k A 12 Y EL FACTOR DE COMPRESIÓN θ A 0.5.

Layer type		Output size	Kernel size
Initial block	Conv	$32 \times 32 \times 2k$	3×3
	BatchNorm	$32 \times 32 \times 2k$	-
	ReLU	$32 \times 32 \times 2k$	-
Dense block 1 ($\times 16$)	BatchNorm	$32 \times 32 \times G$	-
	ReLU	$32 \times 32 \times G$	-
	Conv	$32 \times 32 \times 4k$	1×1
	BatchNorm	$32 \times 32 \times 4k$	-
	ReLU	$32 \times 32 \times 4k$	-
	Conv	$32 \times 32 \times k$	3×3
Transition block 1	BatchNorm	$32 \times 32 \times G$	
	ReLU	$32 \times 32 \times G$	
	Conv	$32 \times 32 \times G/2$	1×1
	Pool	$16 \times 16 \times G/2$	
Dense block 2 ($\times 16$)	BatchNorm	$16 \times 16 \times G$	-
	ReLU	$16 \times 16 \times G$	-
	Conv	$16 \times 16 \times 4k$	1×1
	BatchNorm	$16 \times 16 \times 4k$	-
	ReLU	$16 \times 16 \times 4k$	-
	Conv	$16 \times 16 \times k$	3×3
Transition block 2	BatchNorm	$16 \times 16 \times G$	
	ReLU	$16 \times 16 \times G$	
	Conv	$16 \times 16 \times G/2$	1×1
	Pool	$8 \times 8 \times G/2$	
Dense block 3 ($\times 16$)	BatchNorm	$8 \times 8 \times G$	-
	ReLU	$8 \times 8 \times G$	-
	Conv	$8 \times 8 \times 4k$	1×1
	BatchNorm	$8 \times 8 \times 4k$	-
	ReLU	$8 \times 8 \times 4k$	-
	Conv	$8 \times 8 \times k$	3×3
GlobalAvgPool		G	8×8
MLP		10	-

TABLE IV
TASA DE ACIERTO PARA LAS ARQUITECTURAS USANDO FUNCIONES INDIVIDUALES.

	A1	A2	A3
AM	77.08	87.65	**89.25**
Max	**77.39**	87.85	87.99
Min	70.24	87.61	88.28
Median	70.62	87.07	88.76
$\mathbf{S}_\nu$	72.47	86.79	88.97
$\mathbf{D}_\nu$	73.42	**88.70**	87.20

TABLE V
TASA DE ACIERTO PARA MODELOS QUE UTILIZAN CAPAS COMBPOOL.

	A1	A2	A3
Min + Max	**77.02**	87.42	88.55
Min + Max + Median	76.91	**87.43**	**89.77**
AM + Min	75.04	87.23	89.48
AM + Max	**77.23**	**87.78**	86.99
AM + Min + Max	77.17	87.25	88.52
AM + Min + Max + Median	77.04	87.57	**89.83**
$\mathbf{S}_\nu$ + Min	72.41	87.46	88.58
$\mathbf{S}_\nu$ + Max	77.09	87.60	89.48
$\mathbf{S}_\nu$ + Min + Max	**77.30**	87.01	89.42
$\mathbf{S}_\nu$ + Min + Max + Median	77.03	87.29	**89.66**
$\mathbf{S}_\nu$ + AM	76.93	**88.23**	86.99
$\mathbf{D}_\nu$ + Min	72.19	88.51	89.03
$\mathbf{D}_\nu$ + Max	76.80	88.20	89.58
$\mathbf{D}_\nu$ + Min + Max	**77.81**	**88.61**	89.83
$\mathbf{D}_\nu$ + Min + Max + Median	76.15	88.30	89.75
$\mathbf{D}_\nu$ + AM	76.39	88.40	**89.87**
$\mathbf{D}_\nu$ + $\mathbf{S}_\nu$	74.92	88.03	89.68

[2] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[5] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[8] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[9] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[10] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.

[11] J. Forcén, M. Pagola, E. Barrenechea, and H. Bustince, "Learning ordered pooling weights in image classification," *Neurocomputing*, vol. 411, pp. 45–53, 2020.

[12] C. 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, no. 4, pp. 863–875, 2018.

[13] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition," in *INTERSPEECH*, 2013.

[14] S. Ruder, "An overview of gradient descent optimization algorithms," 09 2016.

[15] F. Bardozzo, B. De La Osa, L. Horansk, J. Fumanal-Idocin, M. delli Priscoli, L. Troiano, R. Tagliaferri, J. Fernandez, and H. Bustince, "Sugeno integral generalization applied to improve adaptive image binarization," *Information Fusion*, vol. 68.

[16] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

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

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, ser. ICML'15, vol. 37. JMLR.org, 2015, pp. 448–456.

[19] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2014.

[20] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*, 2015, pp. 562–570.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.