Additional Feature Layers from Ordered Aggregations for Deep Neural Networks

I. Dominguez-Catena, D. Paternain, M. Galar
Institute of Smart Cities
Public University of Navarre
Pamplona, Spain
{iris.dominguez, mikel.galar, daniel.paternain}@unavarra.es

***Abstract*—In the last years we have seen huge advancements in the area of Machine Learning, specially with the use of Deep Neural Networks. One of the most relevant examples is in image classification, where convolutional neural networks have shown to be a vital tool, hard to replace with any other techniques. Although aggregation functions, such as OWA operators, have been previously used on top of neural networks, usually to aggregate the outputs of different networks or systems (ensembles), in this paper we propose and explore a new way of using OWA aggregations in deep learning. We implement OWA aggregations as a new layer inside a convolutional neural network. These layers are used to learn additional order-based information from the feature maps of a certain layer, and then the newly generated information is used as a complement input for the following layers. We carry out several tests introducing the new layer in a VGG13-based reference network and show that this layer introduces new knowledge into the network without substantially increasing training times.**

***Index Terms*—Neural Nets, RNN, deep learning, OWA operator**

I. Introduction

One of the most common problems to be solved with Machine Learning are classification tasks, either supervised or unsupervised. Among supervised classification problems, an up-to-date example is image classification [1], [2]. In these kinds of problems, we try to create a mathematical model that learns over a set of labelled data (images), which can then label new unseen data appropriately. Currently, the most common technique for image classification is the use of deep Convolutional Neural Networks (CNNs), which is the focus of this paper.

The use of aggregation functions (see [3]–[6] for more details about aggregation functions) in image processing and computer vision is very common, specially focused on preprocessing steps, such as noise reduction, filters, image enhancement or image reduction and compression [7], [8]. A well-known family of aggregation functions are Ordered Weight Averaging (OWA) operators [9], which are very common in fuzzy logic and machine learning as a parametric class of aggregations. In fact, it is well known that depending on the specific weighting vector we can recover important aggregation functions such as the minimum, the median, etc. Moreover, OWA operators can be classified depending on its behavior by means of an orness function, which shows whether an OWA operator is closer to the pure OR operator (maximum) or, on the contrary, is close to the minimum operator [9], [10].

This work was partially supported by the Public University of Navarre under the projects PJUPNA13 and PJUPNA1926 .

In previous works, OWAs have been mainly used in deep learning as a way to combine the outputs of different networks as an ensemble [11]–[14]. OWA operators have also been used with interesting results on the pooling layers of CNNs [15], [16].

Different from the previous works, we attempt to employ OWAs on the inner layers of a CNN to increase the amount of information available for the next layer, adding very few parameters and with a negligible increase in the computational cost. Our aim is to generate cost-free information on the network, getting the feature maps at a certain layer and adding derived information that would be hard to get with regular convolutions. For example, a global view of the activation of a feature map is not easy to be computed and used in a CNN, since the convolutions tend to provide local information, and only stacking layers allows the network to capture more global information. Therefore, we try to take advantage of OWAs and use this type of global information to generate additional order-based information. These two factors mean that the network could benefit from information that would otherwise be difficult to obtain.

This idea is inspired by the work done in [17], where a similar approach is applied to image segmentation. Apart from the target problem itself, the other main differences with our work are the particular OWAs used and the layer architecture. The original paper fixed by hand the weights to 6 specific OWAs, while we will allow the model to learn the weights. Also, while in the original approach, the information from the previous layer was fully replaced, we will just append new information derived by the OWA operators.

In order to test the goodness of our proposal we have considered a VGG13 architecture [18], and inserted different OWA layers on it. We test different configurations, based on the insertion point (the layer of the original network where we insert the new layer), the additional feature maps depth and the order measure used in the new layer. Additionally, we study the kind of OWAs that are learned by the network. For implementation, we chose to use Fastai, a Pytorch based library. This library is flexible enough to allow for an easy implementation of the new layer, and at the same time implements some of the latest developments in Deep Learning. In particular, this allows us to use the 1cycle policy for learning rate control [19] to speed up training. Thanks to this robust base, we can run comprehensive tests with 50 repetitions for each configuration, asserting the statistical significance of the improvements of this new technique. For the tests we use the well-known CIFAR10 and CIFAR100 datasets [20].

Although this is just a preliminary study of how to approach the insertion of OWA based layers on CNNs, we believe that the results show the goodness of OWA-based layers in this specific experimental framework and target problem proposed. We think that this new approach opens a new line of work to be explored.

The remainder of this work is organized as follows. Section II describes related work relevant to our proposal. Section III recalls some preliminaries on both OWAs and CNNs. Then, Section IV specifies our methodology for the insertion of an OWA layer and defines how the layer works. Section V presents the experiments that we have designed to test the OWA layers and our experimental framework. In Section VI we gather the experimental results and analyze them. Finally, Section VII concludes this work and proposes some future work.

II. Related Work

In the literature several ways of combining OWAs and neural networks have been previously explored [11]–[16]. The most common way is using OWAs at the output of the network [11]–[14], but other approaches try to replace the aggregation in the pooling layers of a network [15], [16].

Using fuzzy measure-based aggregation operators [21], such as the Choquet or Sugeno integrals, to aggregate an ensemble of neural networks has been explored multiple times in the recent literature [11]–[14]. Recall that OWA operators are particular cases of Choquet integrals based on symmetric measures. On these systems, the underlying classifiers are trained independently and can potentially be of any type, even though CNNs are the most commonly used classifier when dealing with image classification. After all the predictions from the classifiers are collected, the results are aggregated, and here is where OWA operators or other fuzzy measure-based aggregation operators are used. For example, in [11], Fuzzy Integrals are used to improve the ensemble performance over more classical voting methods, such as simple voting or arrogance.

Another explored technique is using Fuzzy Measure-based operators in pooling layers of CNNs [15], [16]. In this case, the usual aggregations used on pooling layers are replaced by Fuzzy Measure-based operators. The idea is to have more meaningful representations of an image after the size reduction that happens in a pooling layer. More specifically, in [16], the authors show how a Choquet-like integral can improve the results of a mean and max aggregated pooling layer inside a CNN.

Finally, the main inspiration for this work can be found in [17], where the authors propose the creation of what they call a "Fuzzy layer". This layer, inserted on different points of a network, performs six predetermined OWA operations (max, min, soft-max, soft-min, average and a random operator) over the channels of the network, sorted by a measure of the entropy of the channel. In our experience, such a specific approach does not translate well to more general classification problems. While the authors applied their methodology to a semantic segmentation problem [22], following the same procedure in the image classification task did not achieve the expected performance. We think that, specially for classification CNNs, there is a large amount of information encoded in the order of the feature maps of a layer that is lost when applying an OWA operator. Therefore, we design our system with the idea of augmenting the feature maps of a layer instead of replacing them, concatenating the new feature maps to the previous ones. This way, we use the output of our OWA layers as a complement to the output of regular convolutions, giving the next layers information that would be difficult to extract via convolution (global information derived from channel metrics). Furthermore, we also include the OWA weights as parameters of the network, learning them, instead of keeping them fixed as in [17].

It is also worth mentioning the work developed by Veal et al. [23], where they developed a Linear Order Statistic Neuron, an artificial neuron based on an OWA operator. This is another interesting approach to the idea of implementing OWAs on neural networks. This proposal is still a work in progress showing the potential of OWAs in this area. However, we focus on CNN architectures aiming to augment the existing information, rather than substituting or changing the convolution operator.

III. Preliminaries

In this section we present some basic theory on which we base the rest of our work. Section III-A recalls OWA operators, and Section III-B CNNs.

A. OWA operators

Ordered Weighted Averaging operators (OWAs for short) were proposed first by Yager [9]. OWA operators are mappings $F\colon \mathcal{R}^n \to \mathcal{R}$, based on a collection of weights $W = [w_1, \dots, w_n]$, with the condition $w_i \in [0,1]$ for every $i = 1, \dots, n$ and $\sum_{i=1}^{n} = 1$, and defined by:

$$F(a_1, \dots, a_n) = \sum_{j=1}^{n} w_j b_j \tag{1}$$

where b_j represents the j-th largest element of a_i.

Some notable examples of OWA operators would be `max` ($W = [1, 0, \dots, 0]$), `min` ($W = [0, \dots, 0, 1]$), and the `arithmetic mean` ($W = [\frac{1}{n}, \dots, \frac{1}{n}]$).

B. Deep Convolutional Neural Networks

Neural networks (NN) [24] are a set of algorithms designed to recognize patterns. The basic building block for most NNs is the perceptron, an algorithm that processes an input by performing first a linear combination (a simple aggregation) of the components of the input, and then applying a nonlinear activation function to the single output. This simple perceptron is quite limited on its own, but we are able to build more complex networks with it, where multiple perceptrons work over the same input composing a layer, and several layers of perceptrons are stacked on top of each other.

CNNs modify the regular architecture to specialize on spatial based data [25], [26]. The most common use, image processing, means recognizing the 2D spatial relationship of the information, and employing a convolutional operation that operates only between neighboring pixels of the image. This also means losing the large scale information, as the output of a convolutional layer on a pixel is unaware of the information of the image outside of its neighborhood.

Another important feature of CNNs are the pooling layers [1]. This layers, as convolutions, recognise the spatial structure of images and feature maps, but instead of aggregating maps together (as convolutions do), they operate on a single channel, summarizing the information in blocks and reducing the size of the image.

Some well-known CNN architectures that have been developed for image classifications are LeNet [27], the VGG family [18], ResNet [2] or DenseNet [28]. On this paper we will focus on VGG, but our methodology could be potentially extrapolated to most other CNN architectures.

IV. Methodology

In this section, we present the proposed OWA layer and the way it fits into a regular CNN. In section IV-A, we present the basic building block of our proposal, the OWA Layer. In Section IV-B, we present the sorting functions that we are going to use to apply the OWAs. Finally, In Section IV-C, we will explain how the aggregation is performed and how the aggregation weights are initialized.

A. OWA Layer

Our proposed OWA layer will work by taking an input of N images, with a resolution of I rows and J columns and C_{in} channels (input feature maps), and appending C_f new channels to those, $C_f \in [0, C_{in}]$. The output will be N images with the same $I \times J$ resolution, but $C_{out} = C_{in} + C_f$ channels of depth, $C_{out} \geq C_{in}$. To generate those C_f new feature maps, we will apply C_f OWA operators over the input channels. These OWA operators will share the same ordering function, which will use channel metrics to reorder the channels. Then, each one of these OWA operators will generate a new feature map as a linear combination of the sorted channels, based on its own independent weighting vector. These weighting vectors, one for each C_f learned OWA, will be learned and updated as parameters of the network. We explain more about this in Section IV-C. The general architecture of the layer is depicted in Figure 1. It is important to mention that all these operations are performed on a per-channel basis.

This methodology is different from that originally proposed in [17], since we realized early in our experiments that trying to replace the source C_{in} channels with a number of C_{out} channels, all of them obtained from the aggregation of the C_{in} channels by different OWA operators (as was originally proposed in [17]), resulted in high penalties on the training of the network, achieving very poor results in this application. Thus, we opted for this alternative scheme where the new channels just augment the input, without removing information from it.

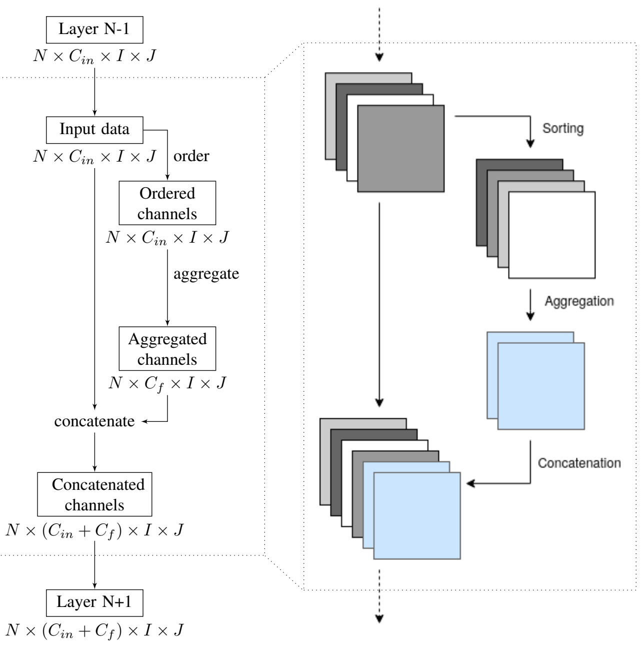
Layer N-1
$N \times C_{in} \times I \times J$
Input data
$N \times C_{in} \times I \times J$
order
Ordered channels
$N \times C_{in} \times I \times J$
aggregate
Aggregated channels
$N \times C_f \times I \times J$
concatenate
Concatenated channels
$N \times (C_{in} + C_f) \times I \times J$
Layer N+1
$N \times (C_{in} + C_f) \times I \times J$
Sorting
Aggregation
Concatenation

Fig. 1. Proposed OWA layer structure.

B. Sorting function

As we are working on a per-channel basis (not on a per-pixel basis), we need to firstly define the ordering function, that is, how to order the C_{in} channels in a decreasing way. Given a channel X of size $I \times J$, we consider the following channel metrics:

- *Channel entropy*. We use the Shannon entropy [29] formula applied to the pixel values of the channel across all rows and columns,

$$H(X) = -\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij} \log x_{ij} \tag{2}$$

As this function is designed to work in vectors of elements in the range $x_{ij} \in [0,1]$ and $\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij} = 1$, we first apply the softmax function to the input X to normalize the input feature map,

$$\text{Softmax}(x_{kl}) = \frac{e^{x_{kl}}}{\sum_{i=1}^{I}\sum_{j=1}^{J} e^{x_{ij}}} \tag{3}$$

Intuitively, the entropy is a measure of disorder, of the amount of information codified in a channel. A higher value of entropy means more uniformity in the input data, while a smaller entropy value means more contrast and information in that input.

- *Sum of channel values*. We also consider the sum of all the channel pixel values,

$$S(X) = \sum_{i=1}^{I} \sum_{j=1}^{J} x_{ij} \tag{4}$$

- *Total variation* [30]. Considering the spatial characteristics of the image, we calculate the differences between each pixel and its horizontal and vertical neighbors, and aggregate the absolute differences over the image.

$$TV_v(X) = \sum_{i=2}^{I} \sum_{j=1}^{J} |x_{i,j} - x_{i-1,j}| \tag{5}$$

$$TV_h(X) = \sum_{i=1}^{I} \sum_{j=2}^{J} |x_{i,j} - x_{i,j-1}| \tag{6}$$

$$TV(X) = TV_v(X) + TV_h(X) \tag{7}$$

The Total Variation (TV), as defined in [30], is a measure that considers the spatial nature of images, and tell us how much variance is between a pixel and its neighbors. The TV measure will be high for images with a lot of crisp borders, where there are high differences between a pixel and its neighbors, and low for channels where every pixel is surrounded by pixels with similar values.

- *Median of channel values*. Another common OWA operator is the median,

$$M(X) = median(x_{11}, \dots, x_{IJ}) \tag{8}$$

where the median operator returns the $ceil(I \cdot J/2)$ largest element of X if $I \cdot J$ is odd or the arithmetic mean of the $I \cdot J/2$ and the $I \cdot J/2 + 1$ largest elements of X, if $I \cdot J$ is even.

- *Maximum of channel values*. In this context, the most activated pixel of the channel,

$$MAX(X) = \max(x_{11}, \dots, x_{IJ}) \tag{9}$$

We have also used two additional sorting methods which are not based on the channel themselves:

- *No sorting*. As a reference system, and in order to check the impact of the additional complexity (added layers) on the system, we have implemented an identity function that does not sort the layers, and returns them as they are given.
- *Random sorting*. We randomly sort the values, to have a reference of the impact of random noise on the system (that could have some potential regularization effect).

C. *Weighted aggregation*

Once we have the input sorted, we need to define the weighting vectors of the C_f OWA operators, each of them composed by C_{in} weights. In our proposal, we will randomly initialize the weights of the OWA layer following a uniform distribution $U(0, 1)$. We treat the OWA layer weights as parameters of the NN, and hence they are learned through back-propagation at the same time as the rest of the parameters.

These weights are not directly constrained and, as they are learned, can take any real value. To comply with the definition of an OWA given in Section III-A, where for each $w_i, i \in 1, \dots, C_f$, we require that $w_i \in [0, 1]$ and $\sum_{i=1}^{C_f} w_i = 1$, we apply some transformations. First, we apply a ReLU function on the raw weights to ensure that they are all positive. We then normalize the matrix by dividing each weight by the total sum of the positive weights to ensure that they add up to one.

$$ReLU(x) = max(x, 0) \tag{10}$$

$$w_j = \frac{ReLU(x_j)}{\sum_{i=1}^{C_f} ReLU(x_i)} \tag{11}$$

The result is a proper OWA weighting vector, that can be directly applied.

V. Experimental Framework

In this section, we explain how we have performed our experiments to test the validity of our new methodology. In Section V-A, we present our chosen datasets, and the reasons that support this choice. Then, in Section V-B, we explain the base architecture, a modification of VGG13, and how we insert our OWA layers in it. Section V-C explains more about our specific implementation, tools used and learning hyperparameters. In Section V-D, we explain our evaluation methodology. Finally, in Section V-E, we present the configurations tested in our three experiments.

A. *Dataset*

For testing, we have opted to use the CIFAR10 and CIFAR100 datasets [20]. CIFAR10 is a well-known dataset composed of 60,000 color images in a 32x32 pixel resolution, sorted into 10 different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks) with 6,000 examples each. CIFAR100 is a similar dataset composed of another 60,000 color images of 32x32 pixel resolution, but this time distributed in 100 classes, each one with 600 examples. Both datasets are already split in train and test partitions, with 50,000 training examples and 10,000 testing examples each, with an even class distribution.

The choice of dataset is motivated by the amount of configurations that we want to try, making a small dataset like this one faster to test multiple hypothesis. Also, it is important to note that most simple architectures have room for improvement on this dataset, unlike on smaller ones like MNIST (where error rates of under 1% are already common, making hard to appreciate new improvements).

B. Architecture

For testing, we decided to employ the VGG family of architectures [18]. Our idea was to use an architecture that already gave good results on our datasets, but still had room for improvement. The VGG architectures are also relatively quick to train, allowing us to do all the experiments on a reasonable time-frame with untrained networks. Also, because of their linear structure, they give us a simple choice of insertion points for the OWA layers.

As the base we have chosen the VGG13 configuration, a good middle point between the lightweight VGG11 and the largest and more common VGG16 presented on the original paper [18]. In our initial tests, we have found that the VGG13 achieved similar results for our dataset to those of the VGG16 network with a much smaller training time, so we settled up for the VGG13 configuration.

The CIFAR10 dataset has a small image size (32×32 pixels), while the initial VGG experiments where run on larger 224×224 pixel images. Thus, we have slightly adapted the VGG architecture, replacing the final 3 layer classifier stage with a simpler classifier, with only one fully connected layer, in the same way as in [31]. In our initial tests, this reduced training times at no cost for the accuracy for our specific dataset. As we will only modify the convolutional blocks, this classifier will be the same for our reference and modified results.

The network itself has a total of 10 convolutional layers, with an additional linear layer for classification. The 10 convolutions are distributed on 5 blocks, each one finished by a MaxPool layer that halves the size of the image. For our experiments, we will consider potential insertion points for the OWA layers just before each of the convolutional blocks, with the exception of the first convolution. This results in 9 potential insertion points, presented in Table I.

In this configuration, each convolutional layer is composed of the convolution itself, followed by a batch normalization layer and a ReLU layer.

C. Implementation Details

The implementation of these experiments was made using PyTorch 1.3.1 and Fastai 1.0.58.

For all the configurations we use the same hyperparameters, namely, a maximum learning rate of $1e^{-2}$ with a 1cycle policy (as described in [19] and implemented in Fastai). This parameter was decided by using the *lr_finder* tool in Fastai on the reference network (without OWA layers). Later, we tested again on the modified versions of the network to check that it was also an adequate parameter for those networks. We used 1024 as batch size for all the experiments. In this particular implementation, no dropout layers have been used.

We have also used basic data augmentation, as described in [32]. The particular augmentations are left-right flipping with a 0.5 probability, and a random padding of 4 pixels (using mirroring to fill the padding), followed by a random crop to the initial size of 32×32 pixels.

TABLE I
NETWORK ARCHITECTURE*.

Name	Kernel Size	Stride	Output Size
input_data	-	-	$32 \times 32 \times 3$
conv1_1	3×3	1	$32 \times 32 \times 64$
OWA_1	-	-	$32 \times 32 \times (64 + C_f)$
conv1_2	3×3	1	$32 \times 32 \times 64$
maxpool	2×2	2	$16 \times 16 \times 64$
OWA_2	-	-	$16 \times 16 \times (64 + C_f)$
conv2_1	3×3	1	$16 \times 16 \times 128$
OWA_3	-	-	$16 \times 16 \times (128 + C_f)$
conv2_2	3×3	1	$16 \times 16 \times 128$
maxpool	2×2	2	$8 \times 8 \times 128$
OWA_4	-	-	$8 \times 8 \times (128 + C_f)$
conv3_1	3×3	1	$8 \times 8 \times 256$
OWA_5	-	-	$8 \times 8 \times (256 + C_f)$
conv3_2	3×3	1	$8 \times 8 \times 256$
maxpool	2×2	2	$4 \times 4 \times 256$
OWA_6	-	-	$4 \times 4 \times (256 + C_f)$
conv4_1	3×3	1	$4 \times 4 \times 512$
OWA_7	-	-	$4 \times 4 \times (512 + C_f)$
conv4_2	3×3	1	$4 \times 4 \times 512$
maxpool	2×2	2	$2 \times 2 \times 512$
OWA_8	-	-	$2 \times 2 \times (512 + C_f)$
conv5_1	3×3	1	$2 \times 2 \times 512$
OWA_9	-	-	$2 \times 2 \times (512 + C_f)$
conv5_2	3×3	1	$2 \times 2 \times 512$
maxpool	2×2	2	$1 \times 1 \times 512$
flatten	-	-	512
linear	-	-	10

* The layers marked as OWA_x are the possible insertion points for the new OWA layers.

D. Evaluation

For evaluation purposes, we have run a large set of experiments with different configurations. All the experiments have been run 50 times, each one training for 30 epochs over the dataset. From these results, we use the test accuracy on the last epoch of each repetition and get both the average and standard deviation of those accuracy values. In order to test whether a configuration works better than the reference, we also perform a statistical test. As we cannot assert the normality of the distribution of results, we consider the non-parametric Mann-Whitney U test [33]. We run this test against the reference, with then null hypothesis that the reference configuration works better or equal than the modified version, for additional statistical confirmation.

The reference for all the experiments is an unmodified version of the network, without OWA layers. In the last experiment, where we compare the order functions, we also include an additional reference, performing the same aggregation on the unsorted channels. This way, we can check whether the improvement comes from the increase in the number of parameters or the usage of OWA layers.

E. Configuration of experiments

Since there is a large combination of possible configurations, we have decided to run 3 experiments. Each experiment corresponds to an important parameter in OWA layers: position, feature depth and the order metric. For each experiment, we run it both on CIFAR10 and CIFAR100:

1) Layer position configurations: In the first experiment, we try a predefined feature depth ($C_f = 16$) across all the possible OWA layer positions (OWA$_1$ to OWA$_9$), with two different order functions, activation sum and total variation.

2) Feature configurations: In the second experiment, we try variable feature depths ($C_f = 4$, $C_f = 8$, $C_f = 16$ and $C_f = 32$) across two of the better performing OWA layer insertion points of Experiment 1, using the activation sum as order function.

3) Order configurations: For the third experiment, we try to focus on the order functions. We fix the configuration to the best layer and feature depth combination for each dataset, and try the full array of order functions. These include two "reference" methods, random sorting (sort the layers randomly) and no sorting (do not sort the layers, just apply the linear aggregation). These two reference functions should show the impact of the additional parameters on the network if we were to use regular unordered aggregations instead of OWAs. We also use this experiment to analyze the weight matrices learned by the OWA operators.

VI. Experimental Study

In this Section, we present the results found on the experiments detailed in Section V-E. Section VI-A refers to the first experiment on the influence of the layer position. Section VI-B refers to the second experiment, testing different feature depths for the OWA layers. Section VI-C covers the last experiment on order configurations. Finally, in Section VI-D, we study the patterns developed on weight matrices learned by the OWA layers under different ordering methods.

A. Insertion point for the OWA layer

The results of the first experiment are summarized in Table II. We can observe a high dependence of the result on the insertion point chosen for the OWA layer. On the CIFAR10 dataset we see that all of the configurations on layers OWA$_2$ to OWA$_5$ improve the reference accuracy, with the best one at OWA$_4$. On the CIFAR100 dataset we can observe similar results, with the best configurations at OWA$_3$ and OWA$_5$. All of the configurations with activation sum as order metric and insertion point between OWA$_2$ and OWA$_5$ have a better accuracy than the reference with statistical significance (p-value < 0.05). In general, we observe better performance on the configurations that have activation sum as order metric than on those with total variance.

We suspect that this tendency to perform better on the lower insertion points is tied to the small image size of our dataset, at just 32×32 pixels. This, in combination with the VGG architecture halving the image size after every block, makes the images at the upper layers very small (from OWA$_6$ to OWA$_9$ we have image sizes of 4×4 and 2×2 pixels), making our channel metrics much less meaningful.

B. Feature configurations

The results of the second experiment are summarized in Table III. From the results of Experiment 1, we chose to stick with the insertion points at OWA$_3$ and OWA$_4$ and activation sum as ordering metric, which give good results for both CIFAR10 and CIFAR100 datasets.

On this second experiment, we can observe some variance between the different feature depths for both layers and datasets. We can observe a tendency to favor more features, with the best results at $C_f = 16$, but it does not seem to hold in all cases. In this experiment, we can observe that all the configurations of $C_f = 8$ and $C_f = 16$ for both CIFAR10 and CIFAR100 have greater accuracy than the reference with p-value < 0.05.

TABLE II
Layer configuration results.

Order	Layer	CIFAR10 acc	CIFAR100 acc
reference	-	92.44 ± 0.17	69.74 ± 0.27
activ_sum	OWA$_1$	92.40 ± 0.19	$69.85 \pm 0.29^\bullet$
	OWA$_2$	$92.53 \pm 0.19^\bullet$	$69.87 \pm 0.32^\bullet$
	OWA$_3$	$92.52 \pm 0.18^\bullet$	$69.97 \pm 0.27^\bullet$
	OWA$_4$	$\mathbf{92.55 \pm 0.18}^\bullet$	$69.95 \pm 0.24^\bullet$
	OWA$_5$	$92.51 \pm 0.17^\bullet$	$\mathbf{69.97 \pm 0.28}^\bullet$
	OWA$_6$	92.45 ± 0.20	69.75 ± 0.33
	OWA$_7$	92.44 ± 0.17	69.82 ± 0.25
	OWA$_8$	92.44 ± 0.18	69.79 ± 0.30
	OWA$_9$	92.49 ± 0.20	69.78 ± 0.25
total_var	OWA$_1$	92.45 ± 0.18	$69.87 \pm 0.26^\bullet$
	OWA$_2$	$92.53 \pm 0.15^\bullet$	$69.86 \pm 0.31^\bullet$
	OWA$_3$	$92.52 \pm 0.18^\bullet$	$69.91 \pm 0.24^\bullet$
	OWA$_4$	92.47 ± 0.17	69.79 ± 0.23
	OWA$_5$	$92.51 \pm 0.19^\bullet$	$69.87 \pm 0.31^\bullet$
	OWA$_6$	92.43 ± 0.20	69.81 ± 0.27
	OWA$_7$	92.45 ± 0.18	69.82 ± 0.28
	OWA$_8$	92.43 ± 0.18	69.82 ± 0.29
	OWA$_9$	92.45 ± 0.19	69.76 ± 0.26

Results marked with $^\bullet$ have better accuracy than the reference with p-value < 0.05.

TABLE III
Feature configuration results.

Layer	Feat	CIFAR10 acc	CIFAR100 acc
reference	-	92.44 ± 0.17	69.74 ± 0.27
OWA$_3$	4	92.49 ± 0.19	69.82 ± 0.32
	8	$92.51 \pm 0.21^\bullet$	$69.88 \pm 0.26^\bullet$
	16	$92.52 \pm 0.18^\bullet$	$\mathbf{69.97 \pm 0.27}^\bullet$
	32	$\mathbf{92.57 \pm 0.16}^\bullet$	69.82 ± 0.28
OWA$_4$	4	92.47 ± 0.17	69.81 ± 0.33
	8	$92.50 \pm 0.19^\bullet$	$69.91 \pm 0.31^\bullet$
	16	$92.55 \pm 0.18^\bullet$	$69.95 \pm 0.24^\bullet$
	32	$92.51 \pm 0.17^\bullet$	$69.90 \pm 0.29^\bullet$

Results marked with $^\bullet$ have better accuracy than the reference with p-value < 0.05.

C. Order configurations

The results of this final experiment are presented in Table IV. Here, we test the different order configurations with the best configurations of Experiment 2, OWA$_3$ and $C_f = 32$ for CIFAR10 and OWA$_3$ and $C_f = 16$ for CIFAR100. We can observe that the activation sum clearly outperforms the rest of the measures, closely followed by the total variation of the layer, on both CIFAR10 and CIFAR100. The entropy measure, the one that was originally used in [17], performs poorly on this application.

The two reference measurements, random and no sorting, perform similarly to the global reference, proving that the improvement that we are observing is not due to the increased complexity of the network, but to the sorting nature of the aggregation. The activation sum and total variation configurations for both datasets outperform the reference accuracy with statistical significance, p-value < 0.05.

TABLE IV
ORDER CONFIGURATION RESULTS.

Order	CIFAR10 acc	CIFAR100 acc
reference	92.44 ± 0.17	69.74 ± 0.27
activ_sum	$\mathbf{92.57 \pm 0.16}^{\bullet}$	$\mathbf{69.97 \pm 0.27}^{\bullet}$
total_var	$92.55 \pm 0.19^{\bullet}$	$69.91 \pm 0.24^{\bullet}$
max_activ	$92.51 \pm 0.21^{\bullet}$	69.74 ± 0.28
median_activ	92.48 ± 0.19	69.84 ± 0.31
entropy	92.47 ± 0.16	69.80 ± 0.25
random	92.45 ± 0.17	69.76 ± 0.26
no_sorting	92.43 ± 0.19	69.79 ± 0.30

Results marked with $^{\bullet}$ have better accuracy than the reference with p-value < 0.05.

D. Weight matrices

There is an interesting analysis to be carried out from the weight matrices of the learned OWAs, specially for the third experiment, where we can compare different ordering measures. In Figure 2, we show the value of the 8 learned weighting vectors (each one composed of 64 weights) after 30 epochs, all obtained from the same configuration (CIFAR10, insertion point in OWA_2, $C_f = 8$) but using different ordering functions. The size of these matrices is 8×64, being $C_f = 8$ the number of OWA operators in the OWA layer and $C_{in} = 64$ the number of weights in each weighting vector.

We can appreciate that the system converges to very clear patterns for most of the aggregations. These patterns tend to be soft-min and soft-max operators, characterized by a set of weights where one of the extremes has a maximum weight, and the weight decreases towards the other extreme, where it usually reaches zero.

Specifically, we can observe that the system converges to soft-min OWAs for the total variation and activation sum orderings (with one feature converging to a soft-max in the case of activation sum), and soft-max OWAs for the entropy ordering. In the case of the median activation, we can observe a very steep soft-max, and more diffuse soft-mins for the rest of the features. If we compare with the no sorting reference, we can observe how it learns regular features, and on the random sorting matrix we can observe that it does not converge to any recognizable pattern.

E. Computational Complexity

To observe the impact of the OWA layer insertion, we run several configurations on a reference machine, equipped with a GeForce RTX 2060 GPU, 20GB of RAM, an Intel Core i5-8500 CPU, and running Ubuntu 18.04. In Table V, we gather the mean epoch times of the network on the CIFAR10 dataset (the number of images is the same for CIFAR100, obtaining equivalent epoch times). The configurations are run with the different order functions and number of features, whereas the insertion point is fixed in OWA_3 for all of the runs.

We can observe no substantial variations between configurations, neither because of the ordering function nor because of the number of new features learned. There is an overall increase in execution time of approximately 10%. The uniform distribution of the time increases with respect to the reference time hints at just a non-optimal implementation of the technique, that we expect to improve in the future. Although there is an increase to be expected, because of the new parameters of the network, the fact that there is no difference between execution times between different number of features indicates that its not a relevant factor here.

TABLE V
EXECUTION TIMES.

Order	Feat	Epoch time (s)
reference	-	9.67
activation_sum	4	10.70
activation_sum	8	10.77
activation_sum	16	10.77
activation_sum	32	10.73
activation_sum	16	10.77
total_variation	16	11.00
max_activation	16	10.86
median_activation	16	11.33
entropy	16	10.93
random	16	11.10
no_sorting	16	11.27

VII. CONCLUSION & FUTURE WORK

In this work we have proposed the insertion of OWA operators in deep CNNs as a method for feature map augmentation. Although the results we have found are not enough to directly push the state-of-the-art, we consider that this experiments show the validity of an interesting approach towards the use of OWAs and other complex aggregations as an additional source of information for regular CNNs.

We have seen some improvements on simple networks, and the approach seems to be quite sensible to the specific configuration used, but once set up, it gives consistent improvements at little computational cost. The approach itself is highly extensible (we have only tried a few ordering functions, for example), and the same experiments could be performed on different problems and different network architectures. In particular, we suspect that higher image sizes could result in larger improvements, thanks to the large scale information provided by the sorting functions.

Additionally, it should be tested whether this approach could be fine-tuned and applied to regular building blocks of complex networks, in the fashion of ResNet [2] and other architectures. Further research is needed about the way the networks are taking advantage from this new addition, and we hope that further understanding could help to stabilize and increase the results reported in this work.

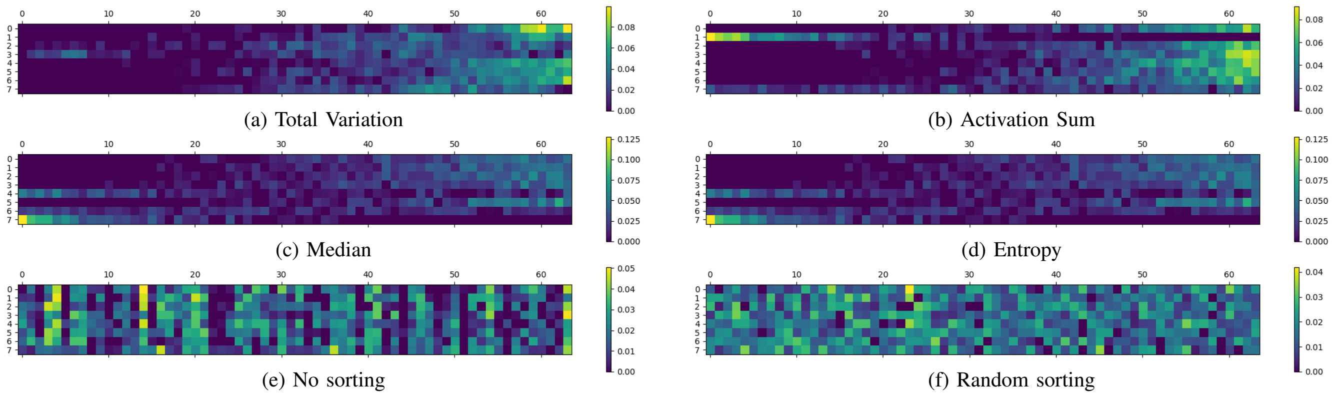

Fig. 2. Learned OWA weights of the proposed layer for different ordering measures. The vertical axis represents the new feature maps, and the horizontal axis the input feature maps.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions*. Cambridge University Press, 2009.

[4] T. Calvo, G. Mayor, and R. Mesiar, *Aggregation Operators. New Trends and Applications*. Physica-Verlag, 2002.

[5] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*. Springer, 2007.

[6] G. Beliakov, H. Bustince, and A. Pradera, *A Practical Guide to Averaging Functions*, 2nd ed. Springer, 2015.

[7] D. Paternain, J. Fernandez, H. Bustince, R. Mesiar, and G. Beliakov, "Construction of image reduction operators using averaging aggregation functions," *Fuzzy Sets and Systems*, vol. 261, pp. 87 – 111, 2015, theme: Aggregation operators.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (4th Edition)*. Pearson, 2018.

[9] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

[10] R. R. Yager, "Families of owa operators," *Fuzzy Sets and Systems*, vol. 59, no. 2, pp. 125 – 148, 1993.

[11] G. J. Scott, R. A. Marcum, C. H. Davis, and T. W. Nivin, "Fusion of deep convolutional neural networks for land cover classification of high-resolution imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1638–1642, 2017.

[12] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 9, pp. 1451–1455, 2018.

[13] D. T. Anderson, G. J. Scott, M. Islam, B. Murray, , and R. Marcum, "Fuzzy choquet integration of deep convolutional neural networks for remote sensing," in *Computational Intelligence for Pattern Recognition*. Springer, 2018, pp. 1–28.

[14] X. Du and A. Zare, "Multiple instance choquet integral classifier fusion and regression for remote sensing applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 5, pp. 2741–2753, 2019.

[15] C. A. Dias, J. C. S. Bueno, E. N. Borges, S. S. C. Botelho, G. P. Dimuro, G. Lucca, J. Fernandéz, H. Bustince, and P. L. J. Drews Junior, "Using the choquet integral in the pooling layer in deep learning networks," in *Fuzzy Information Processing*, G. A. Barreto and R. Coelho, Eds. Springer, 2018, pp. 144–154.

[16] C. A. Dias, J. C. S. Bueno, E. N. Borges, G. Lucca, H. Santos, G. P. Dimuro, H. Bustince, P. L. J. D. Junior, S. S. C. Botelho, and E. Palmeira, "Simulating the behaviour of choquet-like (pre) aggregation functions for image resizing in the pooling layer of deep learning networks," in *International Fuzzy Systems Association World Congress*. Springer, 2019, pp. 224–236.

[17] S. R. Price, S. R. Price, and D. T. Anderson, "Introducing fuzzy layers for deep learning," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2019, pp. 1–6.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 2014.

[19] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," *arXiv 1803.09820*, 2018.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images." *Master's thesis, Department ofComputer Science, University of Toronto*, 2009.

[21] J. Keller, D. Liu, and D. Fogel, *Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation*. Wiley, 2016.

[22] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[23] C. Veal, A. Yang, A. Hurt, M. A. Islam, D. T. Anderson, G. Scott, J. M. Keller, T. C. Havens, and B. Tang, "Linear order statistic neuron," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2019, pp. 1–6.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[25] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[26] 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, no. 4, pp. 541–551, 1989.

[27] 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.

[28] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[29] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[30] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.

[31] 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.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *arXiv 1603.05027*, 2016.

[33] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.