Application of Learned OWA Operators in Pooling and Channel Aggregation Layers in Convolutional Neural Networks

Leonam R. S. Miranda[1], **Frederico G. Guimarães**[2]

[1] Graduate Program in Electrical Engineering
Universidade Federal de Minas Gerais - UFMG
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil

[2]Machine Intelligence and Data Science (MINDS) Laboratory
Universidade Federal de Minas Gerais – UFMG, Belo Horizonte, MG, Brazil

leonamrsm@ufmg.br, fredericoguimaraes@ufmg.br

Abstract. *Promising results have been obtained in recent years when using OWA operators to aggregate data within CNNs pool layers, training their weights, instead of using the more usual operators (max and mean). OWA operators were also used to learn channel wise information from a certain layer, and the newly generated information is used to complement the input data for the following layer. The purpose of this article is to analyze and combine the two mentioned ideas. In addition to using the channel wise information generated by trainable OWA operators to complement the input data, replacement will also be analyzed. Several tests have been done to evaluate the performance change when applying OWA operators to classify images using VGG13 model.*

1. Introduction

Image Classification are one of the most common problems to be solved with Machine Learning, usually being used deep learning and Convolutional Neural Networks (CNNs) to solve them. CNNs have come to be widely applied to computer vision problems when AlexNet [Krizhevsky et al. 2012] won the image classification challenge: ILSVRC 2012 [Russakovsky et al. 2015]. CNNs are neural networks designed to work with data and spatial data, such as images and videos, where any part of the input information is strongly correlated with other nearby parts, which form the neighborhood. In the images this information is the pixels, normally arranged in three channels in RGB scale. CNNs apply layered convolutional operations to detect specific types of features in the input data, where the deeper the layer, the more complex the detected feature.

In CNNs, aggregation operations are applied at convolution, pooling and fully connected layers. The aggregation function is responsible for the process of combining different numeric values returning a single value [Grabisch et al. 2009]. A good example of this averaging functions is the Ordered Weighted Averaging operators (OWA) [Yager 1988]. These OWA operators are a set of parameterized classes of aggregation operators, which have been commonly applied in several fields, such as multicriteria decision making and fuzzy logic [Zhou et al. 2008, Herrera and Martínez 2001].

The Pooling layer performs an aggregation of features by reducing the size of the input sample. Applying pooling is expected to keep relevant information from the input while removing irrelevant and confusing details. The most common operators used

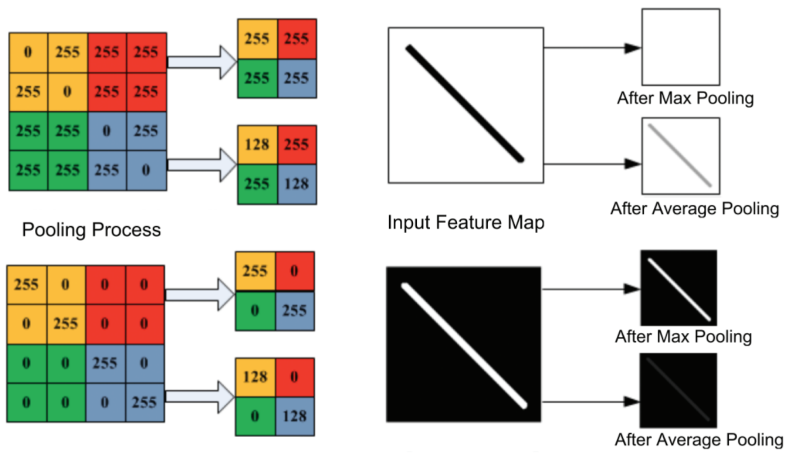


Figure 1. Pooling process of input feature that illustrates the drawbacks of max pooling and average pooling on CNN [Jie and Wanda 2020].

are the maximum and the arithmetic mean. In fact, it is well known that the maximum and arithmetic mean operators are nothing more than special cases of the OWA operators. However, significant information losses may occur when using max pooling. On the other hand, the average pooling gets worse if there are many null elements in the input, greatly reducing the significant features after pooling [Yu et al. 2014]. [Jie and Wanda 2020, Figure 1] illustrates the disadvantages of maximum pooling and average pooling in detecting a diagonal line.

In order to solve the problem of generalizing the pooling layers, the first great inspiration for this article was the work of [Forcen et al. 2020], where a new pooling layer was created using OWA operators, called by the authors **OWA-pooling**. In this layer the aggregation is made by a weighted average of the ordered elements. The OWA weights will be learned in the training phase so that the pooling function propagates the most significant activations for the classification problem. To apply OWA operators, each pooling region of the input data is ordered so that weights are associated not with particular inputs, but with their magnitudes. Another positive point when performing aggregation with trained OWA operators is that it reduces the number of CNNs hyperparameters, due to the fact that hyper-parameter selection is a major drawback in CNNs.

Furthermore, the work [Dominguez-Catena et al. 2020], which also aimed at integrating OWA operators into CNNs, was one of the main sources for the development of this article. In this work another layer was proposed, which in this paper will be called **OWA-channel-aggregation**. This layer will be placed inside the convolutional region of the network and will modify the input data, a 3D matrix of activations, by adding newly generated feature maps using OWA operators. Using OWA operators to aggregate channels based information is an interesting idea as this information is often unavailable for the convolution operation. For this, several filters are convoluted (each kernel represents an OWA operator) over the input channels, ordered in descending order. While training the model, the weights of each applied OWA operator are also learned.

The objective of this work is to jointly apply the **OWA-pooling** and **OWA-channel-aggregation** layers in VGG13 [Simonyan and Zisserman 2014] model, and compare the results with the baseline model, where these new layers are not used. In addition to using the idea of modifying the input data by adding newly generated resource maps using OWA operators, we will also evaluate the cases where the input data will be replaced by the newly generated feature maps created by the fusion of the input data using OWA operators with trainable weights. Merging the input data is expected to improve the overall accuracy performance, allowing more descriptive power to the system. The purpose of testing is not to achieve state-of-the-art performance, but to assess whether or not there was an improvement in the application of the new layers. The models were trained using the CIFAR10 and CIFAR100 data sets. [Krizhevsky et al. 2009].

The rest of the work is organized as follows. Section 2 contains a bibliographic survey of related works. Section 3 presents the OWA operator, **OWA-pooling** and **OWA-channel-aggregation** layers, how each one performs its aggregations, how they were integrated into a CNN and how its owa weights were trained. Section 4 presents the experiments performed and the results obtained. Section 5 analyzes the results obtained in the experiments. The work is concluded in Section 6.

2. Related Work

In addition to the two main references presented in the section 1 Introduction, in the literature OWA operators have been widely used since they were first proposed by Yager [Yager 1988]. Furthermore, the idea of combining OWAs and neural networks has shown promise in recent literature, showing interesting results when used.

One of the most common ways to use OWA operators are in ensemble learning methods [Scott et al. 2017, Anderson et al. 2018]. The idea is to train several models independently and aggregate the results using an aggregation operator in a single output. This is where OWA operators come in to perform this aggregation, or other fuzzy measure-based aggregation operator. Noting that the OWA operators are a special case of aggregation operators based on fuzzy measures [Keller et al. 2016].

Another common application is the use of OWA operators in CNNs' pooling layers. Usually these layers downsample the resolution of the input along its spatial dimensions (height and width) taking the maximum value or average value. However there are several works that have explored the disadvantages of using the usual pooling layers (max and average). In [Boureau et al. 2010] it is provided a detailed theoretical analysis of maximum pooling and average pooling for object recognition tasks. It has been shown that the optimal pooling type for a given classification problem might not be maximum or average clustering, but something in between. With these disadvantages, it became an interesting idea to apply OWA operators to perform the pooling operation in CNNs.

To exemplify the use of OWA operators in the pooling layer, in [Dias et al. 2018] the authors applied a Fuzzy Measure-based operator in a pooling layer of a CNN using Choquet-like integral, showing an improvement in the results compared to the usual average and maximum aggregations. In [Pagola et al. 2017] showed that good results are achieved when using OWA operators with untrained fixed weights in image classification problems.

Last but not least, [Price et al. 2019] proposes the "Fuzzy Layer". This layer was designed to get the information in a certain point of the network and replace it with the result of applying six predefined OWA operators (max, min, soft-max, soft-min, average and a random operator) channel-wise, sorting the channels by entropy.

3. Methodology

In this section, initially, the OWA operator will be presented, how it was implemented in the layers **OWA-pooling** and **OWA-channel-aggregation**, and finally how its weights were trained.

3.1. OWA Operators

Ordered weighted averaging (OWA) operators, proposed by Yager [Yager 1988], belong to the class of averaging aggregation functions. They differ to the weighted arithmetic means in that the weights are associated not with the particular inputs, but with their magnitude. Formally, an OWA operator of dimension **n** is a mapping $f : [0,1]^n \rightarrow [0,1]$ having weight vector $\mathbf{w} = [w_1, ..., w_n]$, with the contidions $w_i \in [0,1]$ and $\sum_{i=1}^{n} w_i = 1$. Specified its conditions, the Eq.1 shows the OWA function.

$$OWA(x_{\searrow}) = \sum_{i=1}^{n} w_i x_i \tag{1}$$

where $x_{\searrow}$ denotes the vector obtained from **x** by arranging its components in descending order $x_{(1)} \geq x_{(2)} \geq ... \geq x_{(n)}$.

From the given definition (1) it is obvious that the calculation of the value of an OWA function involves sorting the array of values to be aggregated. Some notable examples of OWA operators would be $\max$ (corresponding to $\mathbf{w} = [1, 0, ..., 0]$), $\min$ ($\mathbf{w} = [0, ..., 0, 1]$), and the arithmetic mean (for which $\mathbf{w} = 1/n, ..., 1/n$).

3.2. OWA-pooling

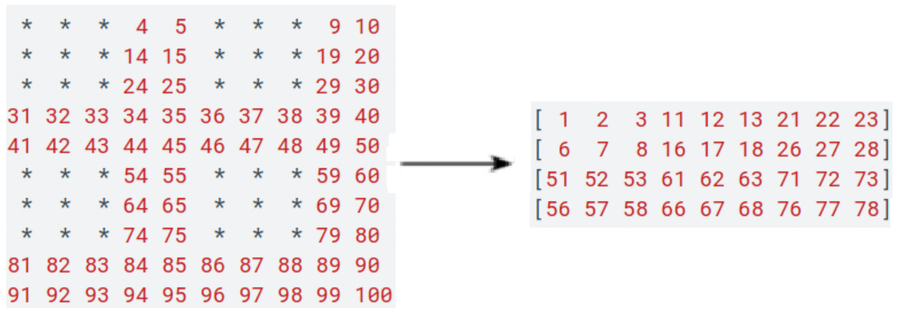
The OWA-pooling layer, like the usual pooling layers, is responsible for downsampling the input its spatial dimensions (height and width). When applying the pooling operation, it is expected to keep relevant information from the input while removing irrelevant and confusing details. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

The pooling operation involves sliding a two-dimensional filter (kernel) over each channel of feature map and summarising the features lying within the region covered by the filter. For a feature map having dimensions $\mathbf{n_h} \times \mathbf{n_w} \times \mathbf{n_c}$, the dimensions of output obtained after a pooling layer is:

$$\mathbf{(nh - f + 1)/s \times (nw - f + 1)/s \times nc}$$

where: $\mathbf{nh}$ - height of feature map; $\mathbf{nw}$ - width of feature map; $\mathbf{nc}$ - number of channels in the feature map; $\mathbf{f}$ - window size of the two-dimensional filter; $\mathbf{s}$ - stride length that specifies how far the pooling window moves for each pooling step.

In this layer, the first operation performed is the extraction of patches from the input feature map. Figure 2 illustrates how patches are extracted from a feature map. After the patches are extracted, their values are sorted in descending order, so that they can be aggregated using an OWA operator. Figure 3 exemplifies how the OWA operator is applied to perform the pooling operation, where each color represents a patch and the result of its respective aggregation.



```
 *  *  *  4  5  *  *  *  9  10
 *  *  * 14 15  *  *  * 19  20
 *  *  * 24 25  *  *  * 29  30
31 32 33 34 35 36 37 38 39  40
41 42 43 44 45 46 47 48 49  50
 *  *  * 54 55  *  *  * 59  60
 *  *  * 64 65  *  *  * 69  70
 *  *  * 74 75  *  *  * 79  80
81 82 83 84 85 86 87 88 89  90
91 92 93 94 95 96 97 98 99 100
```

```
[ 1  2  3 11 12 13 21 22 23]
[ 6  7  8 16 17 18 26 27 28]
[51 52 53 61 62 63 71 72 73]
[56 57 58 66 67 68 76 77 78]
```

Figure 2. Patch extraction with a 3 x 3 window size and stride value equal to 5.

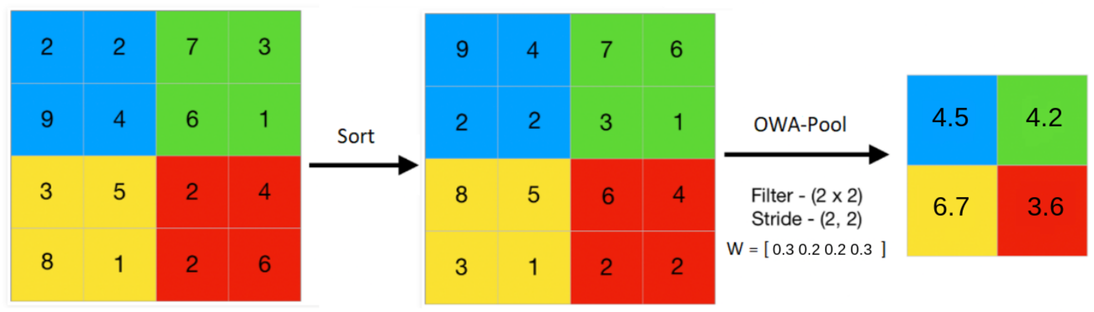


Figure 3. Example of pooling using OWA operator.

Two approaches to using this layer will be considered during the tests. The first is that only a single OWA operator will be applied to all channels from the input feature map. In the second approach, for each channel from the input feature map, a different OWA operator will be used. The weights of the OWA operators will be learned during the training of the model from an iterative process, with the objective of generating more discriminative aggregations than those obtained in the usual pooling layers.

3.3. OWA-channel-aggregation

This special OWA layer[Dominguez-Catena et al. 2020, Dominguez-Catena et al. 2021] will be placed inside the convolutional region of the network and will modify the input data. First of all, this layer reorders the input data channels in a descending way, based on a global ordering metric of the resource maps. After sorting the input data channels, each input, with resolution of I rows and J columns and C_{in} channels, is convoluted with C_f filters formed by OWA operators, each one defined by a weighting matrix of unit height and width and depth equal to C_{in}. As a result, C_f new resource maps are generated. These new feature maps can be concatenated to the input data, or replace the input data, as shown in Fig. 4.

It should be noted that the aggregation does not consider the height and width of the resource maps, as done in the OWA-pooling layer. The aggregation is based on the data in each of the channels, for each value of height $i \in I$ and width $j \in J$.

In [Dominguez-Catena et al. 2020] tests were carried out with different sorting functions for the channels of the input data. The tests were performed on the VGG13

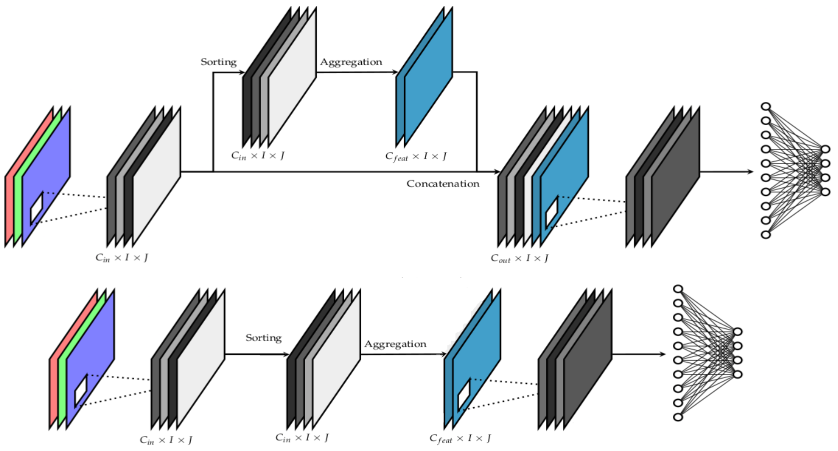

Sorting
Aggregation
$C_{in} \times I \times J$
$C_{feat} \times I \times J$
Concatenation
$C_{in} \times I \times J$
$C_{out} \times I \times J$
$C_{in} \times I \times J$
Sorting
Aggregation
$C_{in} \times I \times J$
$C_{feat} \times I \times J$

Figure 4. OWA layer structure. On the top CNN, the newly generated feature maps are added to the input data, and on the bottom CNN, the replacement is performed. Adapted from [Dominguez-Catena et al. 2021].

network, also using the CIFAR10 and CIFAR100 data sets. Several ordering functions were considered, such as Shannon entropy [Shannon 1948], sum of channel values, total variation of channels values [Rudin et al. 1992], median of channel values and maximum of channel values. At the end of the study, it was concluded that the activation sum clearly outperforms the rest of the measures. As consequence, in this present work, the sum of the channel values will also be used to order the resource maps, according to Eq. 2.

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

Given a channel X of size I × J.

3.4. OWA weights

The values of the weights of the OWA operators were initialized through samples of a uniform distribution U(0,1). These weights are treated as trainable parameters of the CNN and learned by the backpropagation process, at the same time as the rest of the trainable parameters of the model. To ensure that the OWA weights of the **OWA-pooling** and **OWA-channel-aggregation** layers meet the conditions that define an OWA operator, $\mathbf{w} = [w_1, ..., w_n]$, with the contidions $w_i \in [0, 1]$ and $\sum_{i=1}^{n} w_i = 1$, according to subsection 3.1, constraints have been added to the model. These constraints are per-variable projection functions applied to the target variable after each gradient update [Chollet et al. 2015]. Eq. 3 shows the constraint that was applied to the weights of each OWA operators.

$$w_i = \frac{\max(w_i, 0)}{\sum_{i=1}^{f} \max(w_f, 0)}, \forall i \in f \tag{3}$$

Where f is the total number of weights of the OWA operator.

4. Experiments and Results

This section presents the databases used, the model applied, some implementation details, the experiments carried out and the results obtained.

4.1. Dataset

The experiments were performed on the CIFAR10 and CIFAR100 data sets [Krizhevsky et al. 2009]. CIFAR10 is a well-known dataset composed of 60,000 color images in a 32x32 pixel resolution, sorted into 10 different classes with 6,000 examples each. CIFAR100 is a similar dataset composed of another 60,000 color images of 32x32 pixel resolution, with 100 different 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.

4.2. Model Architecture

All tests to evaluate the performance when using the **OWA-pooling** and **OWA-channel-aggregation** layers were performed on the VGG13 network [Simonyan and Zisserman 2014]. This is a well-known CNN architecture with 10 convolutional blocks, each consisting of a convolutional layer, a batch normalization layer, and a ReLU activation layer. However, a small change was made in the fully connected layers of the network, replacing the last 3 dense layers with a single dense layer composed of 512 neurons, in the same way as in [Liu and Deng 2015]. This change greatly reduces the number of trained parameters, without impacting much on performance. Instead of VGG16 or VGG19, which are used more frequently in the literature, VGG13 was chosen because in the tests the models were trained for a few epochs and it was found that when using VGG13 there was no significant loss of performance, while having a smaller training time.

Table 1 shows the modified architecture of VGG13 used in this work. *OWA-ca* layer represents an **OWA-channel-aggregation** layer, where the output has dimensions $16 \times 16 \times 128 + Cf$ if used to add more feature maps, or 16 × 16 × Cf if used to replace the input feature maps. C_f is equal to 32 or 16 for CIFAR10 or CIFAR100 respectively. The network output has dimension 10 or 100 for CIFAR10 or CIFAR100 respectively.

In [Dominguez-Catena et al. 2020] several insertion points of the **OWA-channel-aggregation** layer in VGG13 were considered, each one just before each of the convolution layers, with the exception of the first convolution layer. From the experiments carried out, it was concluded that adding new feature maps just before the 2nd convolution layer of the 2nd VGG block presented better performance. For the CIFAR10 dataset the best results were obtained by adding 32 new feature maps, and for the CIFAR100 dataset the best results were obtained by adding 16 new feature maps. Thus, this work will work only with the best insertion point found in [Dominguez-Catena et al. 2020] generating new 32 and 16 when working with CIFAR10 and CIFAR100 respectively.

4.3. Implementation Details

As the computational cost required to train a convolutional neural network model, it is common to find works that perform only one execution of the same model on the same dataset. The CIFAR10 and CIFAR100 data sets have a training set and test set division,

Table 1. VGG13 Architecture

Name	Kernel Size	Stride	Output Size
input_data	-	-	$32 \times 32 \times 3$
block1_conv1	3×3	1	$32 \times 32 \times 64$
block1_conv2	3×3	1	$32 \times 32 \times 64$
pooling1	2×2	2	$16 \times 16 \times 64$
block2_conv1	3×3	1	$16 \times 16 \times 128$
OWA-ca	1×1	1	$16 \times 16 \times 128 + C_f$ or $16 \times 16 \times C_f$
block2_conv2	3×3	1	$16 \times 16 \times 128$
pooling2	2×2	2	$8 \times 8 \times 128$
block3_conv1	3×3	1	$8 \times 8 \times 256$
block3_conv2	3×3	1	$8 \times 8 \times 256$
pooling3	2×2	2	$4 \times 4 \times 256$
block4_conv1	3×3	1	$4 \times 4 \times 512$
block4_conv2	3×3	1	$4 \times 4 \times 512$
pooling4	2×2	2	$2 \times 2 \times 512$
block5_conv1	3×3	1	$2 \times 2 \times 512$
block5_conv2	3×3	1	$2 \times 2 \times 512$
pooling5	2×2	2	$1 \times 1 \times 512$
dense	-	-	512
dense	-	-	10 or 100

this allows a comparison between the models, however the model performance may vary according to the initial values of the weights and the order of presentation of the images to the CNN.

By replacing the usual pooling layers with OWA-pooling, it was found to be too costly to train the model, as the patch extraction and classification operations are computationally expensive. Thus, in this work, only one execution was performed for each model in each of the data sets, performing hold-out evaluation. For this, the seed used to generate the pseudo-random numbers was fixed, so there is a fair comparison with the reference model, because the same initial weights of the model are generated, the same division into batchs and other values generated randomly during the training.

In all experiments the model was trained from scratch for 30 epochs, with the learning rate initialized to the value of 1×10^{-3}, being halved whenever it stabilizes for 3 epochs. The batch size was set to 32. Stochastic gradient descent was chosen as the optimizer, with momentum equal to 0.9. The weights of the convolution and dense layers were initialized using He uniform distribution [He et al. 2015].

The Python language was used together with the Keras framework [1] for the development of **OWA-pooling** and **OWA-channel-aggregation** layers. The models were trained using the Google Collaboratory [2].

4.4. Configuration of Experiments

Nine different configurations of experiments were defined to be performed, which represent different combinations of layers **OWA-pooling** and **OWA-channel-aggregation**:

- **Orig**: original baseline model with max-pooling
- **OWA-PL**: learn OWA-pooling weights for each layer
- **OWA-PC**: learn OWA-pooling weights for each channel of each layer
- **OWA-A**: learn and add new feature maps
- **OWA-APL**: learn and add new feature maps + learn OWA-pooling weights for each layer
- **OWA-APC**: learn and add new feature maps + learn OWA-pooling weights for each channel of each layer
- **OWA-R**: learn and replaces for new feature maps
- **OWA-RPL**: learn and replaces for new feature maps + learn OWA-pooling weights for each layer
- **OWA-RPC**: learn and replaces for new feature maps + learn OWA-pooling weights for each channel of each layer

The results of the experiments, showing test accuracy, are presented in Table 2.

Table 2. Accuracies in CIFAR10 and CIFAR100 with different model configurations

Configuration	CIFAR10 (% acc)	CIFAR100 (% acc)
Orig	84.810	57.660
OWA-PL	**86.240**	60.180
OWA-PC	85.440	**60.540**
OWA-A	85.090	56.980
OWA-APL	86.040	60.430
OWA-APC	86.110	57.700
OWA-R	84.440	54.040
OWA-RPL	84.830	56.810
OWA-RPC	85.040	57.980

[1] https://keras.io/

[2] colab.research.google.com

5. Analysis of Results

From the results shown in Table 2, it is observed that the best results were obtained by applying only the **OWA-pooling** layer, training an OWA operator for each layer in CIFAR10, and training a OWA operator for each channel of each layer for CIFAR100. It was expected that when using **OWA-pooling** together with **OWA-channel-aggregation** the best results would be obtained, as the model's capacity was increased. In cases where the feature maps generated by **OWA-channel-aggregation** layer were concatenated with the input feature maps, there was a significant performance improvement when compared to the baseline model. Finally, in cases where input feature maps have been replaced by feature maps generated by the **OWA-channel-aggregation** layer, it was observed that there was a significant loss of performance, indicating a large amount of significant information encoded in input data feature maps are lost when aggregating them using an OWA operator.

It should be noted that when using the **OWA-pooling** layer, the training time increases from 10 to 20 times. As GPUs delivered by google Colaboratory have variable performance, training times were not saved.

6. Conclusion

In this work, the joint use of the layers **OWA-pooling** and **OWA-channel-aggregation** was investigated, proposed by [Forcen et al. 2020] and [Dominguez-Catena et al. 2020] respectively. From the experiments it is observed that promising results are achieved when using the layers **OWA-pooling** and **OWA-channel-aggregation**, showing the validity of an interesting approach towards the use of OWAs operators to aggregate information in CNNs.

Despite promising results, a statistical analysis of the results is recommended, where each experiment is repeated several times, in order to be able to say with greater certainty whether or not there was a performance improvement when using the proposed layers.

The best results were obtained using the **OWA-pooling** layer, however the training time increased substantially. Therefore, a future work proposal would be to investigate the use of the **OWA-pooling** layer to fine tune trained models.

Additionally, it should be tested whether this approach could applied in more complex networks, like Resnet [He et al. 2016] and in different datasets.

Acknowledgement

This work has been supported by the Brazilian agencies (i) National Council for Scientific and Technological Development (CNPq), Grant no. 312991/2020-7; (ii) Coordination for the Improvement of Higher Education Personnel (CAPES) and (iii) Foundation for Research of the State of Minas Gerais (FAPEMIG, in Portuguese), Grant no. APQ-01779-21. MINDS Laboratory – `https://minds.eng.ufmg.br/`

References

Anderson, D. T., Scott, G. J., Islam, M. A., Murray, B., and Marcum, R. (2018). Fuzzy choquet integration of deep convolutional neural networks for remote sensing. In *Computational Intelligence for Pattern Recognition*, pages 1–28. Springer.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Chollet, F. et al. (2015). Keras.

Dias, C. A., Bueno, J., Borges, E. N., Botelho, S. S., Dimuro, G. P., Lucca, G., Fernandéz, J., Bustince, H., and Drews Junior, P. L. J. (2018). Using the choquet integral in the pooling layer in deep learning networks. In *North american fuzzy information processing society annual conference*, pages 144–154. Springer.

Dominguez-Catena, I., Paternain, D., and Galar, M. (2020). Additional feature layers from ordered aggregations for deep neural networks. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE.

Dominguez-Catena, I., Paternain, D., and Galar, M. (2021). A study of owa operators learned in convolutional neural networks. *Applied Sciences*, 11(16):7195.

Forcen, J. I., Pagola, M., Barrenechea, E., and Bustince, H. (2020). Learning ordered pooling weights in image classification. *Neurocomputing*, 411:45–53.

Grabisch, M., Marichal, J.-L., Mesiar, R., and Pap, E. (2009). *Aggregation functions*, volume 127. Cambridge University Press.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Herrera, F. and Martínez, L. (2001). A model based on linguistic 2-tuples for dealing with multigranular hierarchical linguistic contexts in multi-expert decision-making. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(2):227–234.

Jie, H. J. and Wanda, P. (2020). Runpool: A dynamic pooling layer for convolution neural network. *Int. J. Comput. Intell. Syst.*, 13(1):66–76.

Keller, J. M., Liu, D., and Fogel, D. B. (2016). *Fundamentals of computational intelligence: neural networks, fuzzy systems, and evolutionary computation*. John Wiley & Sons.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Liu, S. and Deng, W. (2015). Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, pages 730–734. IEEE.

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

Price, S. R., Price, S. R., and Anderson, D. T. (2019). Introducing fuzzy layers for deep learning. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Scott, G. J., Marcum, R. A., Davis, C. H., and Nivin, T. W. (2017). Fusion of deep convolutional neural networks for land cover classification of high-resolution imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(9):1638–1642.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, 18(1):183–190.

Yu, D., Wang, H., Chen, P., and Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology*, pages 364–375. Springer.

Zhou, S.-M., Chiclana, F., John, R. I., and Garibaldi, J. M. (2008). Type-1 owa operators for aggregating uncertain information with uncertain weights induced by type-2 linguistic quantifiers. *Fuzzy Sets and Systems*, 159(24):3281–3296.