Dominguez-Catena I., Paternain D., Galar M. (2021) Learning Channel-Wise Ordered Aggregations in Deep Neural Networks. In: Kahraman C., Cevik Onar S., Oztaysi B., Sari I., Cebi S., Tolga A. (eds) Intelligent and Fuzzy Techniques: Smart and Innovative Solutions. INFUS 2020. Advances in Intelligent Systems and Computing, vol 1197. Springer, Cham. https://doi.org/10.1007/978-3-030-51156-2_119

Learning Channel-wise Ordered Aggregations in Deep Neural Networks *

Iris Dominguez-Catena[0000−0002−6099−8701], Daniel Paternain[0000−0002−5845−887X], and Mikel Galar[0000−0003−2865−6549]

Institute of Smart Cities, Public University of Navarre, Pamplona, Spain
{iris.dominguez, mikel.galar, daniel.paternain}@unavarra.es

Abstract. One of the most common techniques for approaching image classification problems are Deep Neural Networks. These systems are capable of classifying images with different levels of detail at different levels of detail, with an accuracy that sometimes can surpass even manual classification by humans. Most common architectures for Deep Neural Networks are based on convolutional layers, which perform at the same time a convolution on each input channel and a linear aggregation on the convoluted channels. In this work, we develop a new method for augmenting the information of a layer inside a Deep Neural Network using channel-wise ordered aggregations. We develop a new layer that can be placed at different points inside a Deep Neural Network. This layer takes the feature maps of the previous layer and adds new feature maps by applying several channel-wise ordered aggregations based on learned weighting vectors. We perform several experiments introducing this layer in a VGG neural network and study the impact of the new layer, obtaining better accuracy scores over a sample dataset based on ImageNet. We also study the convergence and evolution of the weighting vectors of the new layers over the learning process, which gives a better understanding of the way the system is exploiting the additional information to gain new knowledge.

Keywords: Neural Nets · RNN · Deep Learning · Ordered Aggregations.

1 Introduction

Image Classification can be considered one of the most common problems to be solved with Deep Learning and Convolutional Neural Networks (CNNs). CNNs are neural networks that allow us to work on data with spatial structures, such as images, where the meaning of a pixel depends strongly on its neighborhood. This is achieved in CNNs by applying convolutional operations that impose local connectivity constraints on the network weights. Most CNN architectures alternate these layers that exploit local information with pooling layers that down-sample

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

the feature maps, making the network resistant to local translations. This architecture means that, for most CNNs, only local information is exploited in early layers of the network, and only by down-sampling will we combine this information into a more global perspective in later layers.

Our main hypothesis is that some of this global information could be recovered via ordered weighted aggregations using global metrics, and used on the convolutional layers of the network to gain new knowledge. Thus, our main objective is to implement a new layer that augments the feature maps of a CNN network, generating information with channel-wise ordered weighted aggregations, and study the impact on the system. We are interested in both the impact on the final performance of the network and the convergence of the weighting vectors of the new aggregations.

The rest of this work is as follows. In Section 2 we will overview the literature related to our work. Then, in Section 3 we will develop our proposed methodology. In Section 4 we will present a case of study, with a specific implementation of the method and the results we have found. Finally, in Section 5 we will conclude this work and propose some future work lines.

2 Related work

In the literature we can find several approaches for applying ordered weighted aggregations, usually Ordered Weighting Averaging operators (OWA for short) and other fuzzy measures to CNNs [1, 3–5, 7, 9, 10, 13].

The most common approach is to use fuzzy measures to aggregate the result of an ensemble of classifiers, in the image classification case usually CNNs [1, 5, 9, 10]. In these systems, we have several independent classifiers, and the aggregation only operates on the results of the classifiers.

Another approach is to use Fuzzy Measure-based operators in the pooling layers of CNN classifiers [3, 4], replacing the common aggregations used for pooling, maximum and average.

A third interesting approach are Linear Order Statistic Neurons [13], where the neuron at the core of neural networks is redesigned based on OWA operators.

Finally, the work presented here is mainly inspired by previous attempts to employ OWA operators to summarize the information on a CNN [7]. The authors proposed the creation a "Fuzzy Layer", 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. Our proposal works in a similar fashion, but with several key differences. The first one is that we will try to augment the information of the network, instead of replacing it. The second one is that we will not use predefined operators, but we will allow the network to learn the weights of the aggregations. Finally, we will not use OWA operators, as the common constraints make the learning progress more difficult, and instead, we will use a generalized version that does not constraint the weights.

3 Proposal

To develop our proposal, we will first present the general layer structure in Section 3.1. The, in Section 3.2 we will present the metric used for sorting image channels. Finally, we will consider the way of aggregating channels in Section 3.3.

3.1 Layer structure

Our proposed augmented layer structure will be a combination of ordering and aggregation, used to augment the information in the network. We should note that by combining ordering and aggregation, we are creating an approximation of a channel-wise OWA operator [14], with some of the restrictions that make them averaging operators lifted. For these aggregations to be real OWA, we would require that $w_i \in [0,1]$ for every $i = 1,\ldots,n$ and $\sum_{i=1}^{n} = 1$. In our experience these constraints impair learning, so we avoid this constraint for the practical implementation of OWAs.

The layer will be placed in a CNN between two layers, and will take as input the image of resolution $I \times J$ with C_{in} channels, each channel being a feature map corresponding to a convolutional filter of the previous layer. As output, the layer will generate a new image of $I \times J$ resolution, but with additional channels making a total of C_{out} number of channels, with $C_{out} = C_{in} + C_{feat}$. To generate the new output, we will take the input channels of an image, sort them according to a channel-wise metric (explained in Section 3.2), and aggregate them with C_{feat} channel-wise weighted aggregations (explained in Section 3.3). Finally, we will concatenate the new C_{feat} generated channels with the original C_{in} channels to preserve the original information of the network.

3.2 Channel-wise ordering metrics

To sort the channels we define an ordering metric, a function that will take as input a single channel image X of size $I \times J$, and output a single $m(X)$ measure value corresponding to that value.

For this work, we have to chosen to use as the ordering metric the Total Variation [8]. This metric can be defined as:

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

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

$$TV(X) = TV_v(X) + TV_h(X) \tag{3}$$

This metric measures the absolute differences between each pixel and its horizontal and vertical neighbors. This will result in a higher TV for images with a lot of crisp borders or peak values, and lower TV for flatter images with constant pixel values.

3.3 Channel-wise weighted aggregation

For the aggregation, we use a simple weighted aggregation, with a single weight per channel. We will perform several of these aggregations, thus taking an input image X of C_{in} channels and a resolution of I rows J, and producing a new image of C_{feat} channels with the same resolution of $I \times J$, generating each channel with an independent weighted aggregation of the input image.

For these aggregations, we impose the restriction that all the weights $W_i > 0$ for $i \in (1, C_{in})$. This restriction is implemented by using a ReLU over the raw weights, where:

$$ReLU(X) = max(x, 0) \tag{4}$$

The matrix of weights $(C_{feat} \times C_{in})$ corresponding to these aggregations (where one row corresponds to one aggregation) are initialized consisting of random numbers following a normal distribution $\mathcal{N}(0, 1)$. Since we work with positive weights, we obtain the absolute values of the generated random numbers.

After that, these weights are considered in the same as the rest of the parameters in the network, in such a way that they are learned by back-propagation.

4 Case of study

As a test for the proposed architecture, we implement the layer inside a VGG13 network [11]. This is a well known CNN architecture with 10 convolutional blocks, each one composed of a convolutional layer, a batch normalization layer and a ReLU activation layer. We test our new proposed layer by inserting it on the points marked as P_1 and P_2, one in the early layers of the network and one in the later layers. The layer structure is presented in the Table 1.

For evaluating our proposal, we consider the Imagenette dataset, a subset of ImageNet [2]. This dataset includes 10 classes from the original ImageNet, each one with around 1,300 training images and 50 test images, for a total of 12,894 training images and 500 test images. These images are in color, with 3 RGB channels, and a variable image size. We will resize all of them to a 256×256 resolution upon loading. The implementation of this experiment is done using PyTorch 1.3.1 and Fastai 1.0.58.

We use as a reference an unmodified version of the network. For both the reference and the test configurations, we run 10 repetitions, each one training the network from scratch for 10 epochs, following the 1cycle policy [12], with a maximum learning rate of 6×10^{-5}. We then get the last accuracy score of each run, and compute both the average, and the non-parametric Mann-Whitney U test [6] comparing each configuration with the reference network. For the statistical test, the null hypothesis is that the configuration has the same performance as the reference, and we will consider the threshold of 0.05 for the resulting p-value.

The accuracy results of the reference and our two test configurations, one with insertion point in P_1 and 64 learned feature maps, and the other with insertion point in P_2 and 16 feature maps, are shown in the Table 2. We choose

Table 1. Network architecture.

Name	Kernel Size	Stride	Output Size
input_data	-	-	$256 \times 256 \times 3$
conv1_1	3×3	1	$256 \times 256 \times 64$
conv1_2	3×3	1	$256 \times 256 \times 64$
maxpool	2×2	2	$128 \times 128 \times 64$
conv2_1	3×3	1	$128 \times 128 \times 128$
P_1	-	-	$128 \times 128 \times (128 + C_{feat})$
conv2_2	3×3	1	$128 \times 128 \times 128$
maxpool	2×2	2	$64 \times 64 \times 128$
conv3_1	3×3	1	$64 \times 64 \times 256$
conv3_2	3×3	1	$64 \times 64 \times 256$
maxpool	2×2	2	$32 \times 32 \times 256$
conv4_1	3×3	1	$32 \times 32 \times 512$
conv4_2	3×3	1	$32 \times 32 \times 512$
maxpool	2×2	2	$16 \times 16 \times 512$
conv5_1	3×3	1	$16 \times 16 \times 512$
P_2	-	-	$16 \times 16 \times (512 + C_{feat})$
conv5_2	3×3	1	$16 \times 16 \times 512$
maxpool	2×2	2	$8 \times 8 \times 512$
linear	-	-	256
linear	-	-	256
linear	-	-	10

this learned feature map numbers to keep a proportion of one generated feature map for every 8 original layers at that point. We can observe an improvement of 1% with both configurations, with a more marked improvement in the second configuration (on a higher insertion point and using more learned features). Both configurations have statistically significant results, with a $p - value \leq 0.05$.

Table 2. Accuracy scores.

Insertion point	C_{feat}	Accuracy	$p - value$
reference	-	89.45 ± 0.52	-
P_1	16	90.48 ± 0.55	0.0079
P_2	64	$\mathbf{90.78 \pm 0.51}$	0.0016

We present some of the weighting matrices in the Figure 1. The first column corresponds to a run of the first configuration (with insertion point in the new layer in P_1 and 64 learned feature maps), while the second column to a run of the second configuration (with insertion point in P_2 and 16 feature maps). On both columns, from top to bottom, the weighting matrices are plotted on each epoch of the training.

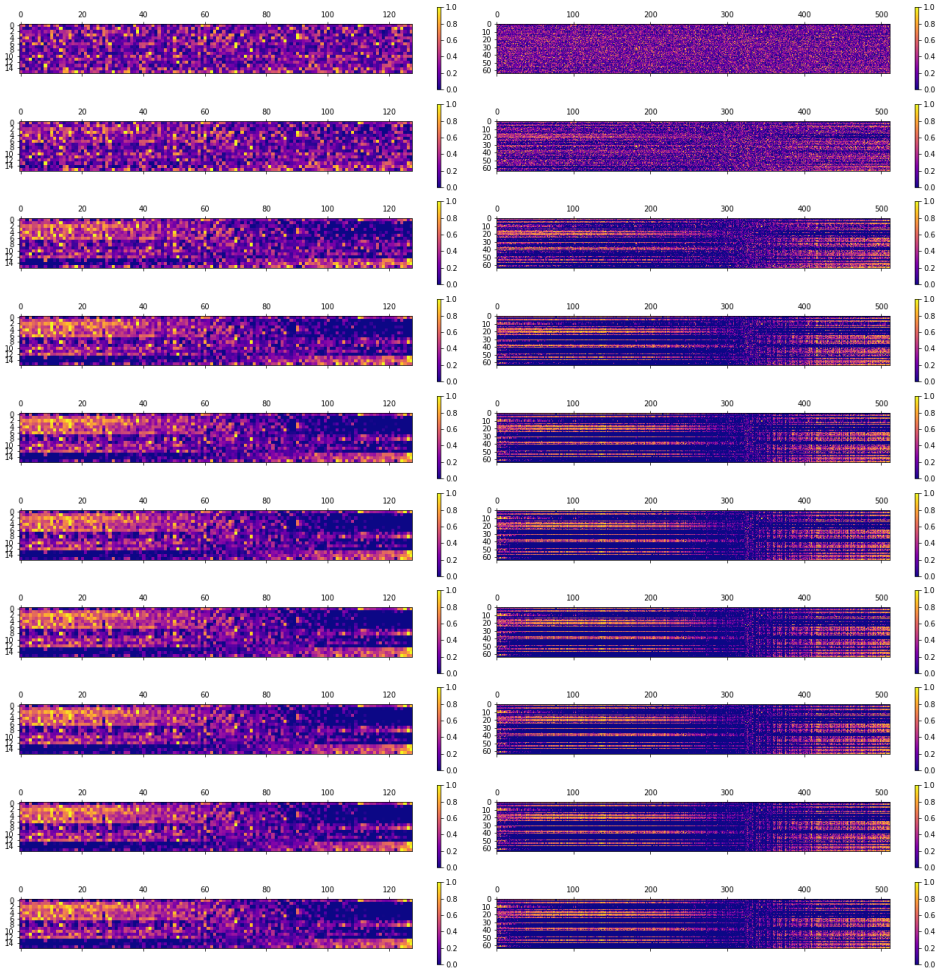
We can observe a clear and fast convergence in both cases, where after the third epoch the matrices remain stable. On both configurations we can observe very clear patterns, similar to soft-max and soft-min operators. These patterns seem to repeat for each aggregation, converging to just two main models, one for the min and one for the max. Both configurations evolve with similar speed and to similar patterns, although with a bit more noise on the configuration with insertion point in P_1. This supports the theory that the network is extracting knowledge from these aggregations, and the fact that the models are smooth proves that the channel metric that we are using is not generating a fixed order (which would defeat the purpose of using ordered aggregations).

5 Conclusion

We have presented and studied the augmentation of a CNN with information derived from the application of ordered weighted aggregations to its feature maps. The study shows that the systems is able to extract knowledge from the new layer, obtaining an improvement in accuracy and showing a clear convergence of the weighting matrices corresponding to the new layer.

Further research is needed to stabilize these results, studying the impact of the layer on different points of the network, different network architectures, and under different circumstances.

We also believe that the general architecture proposed for feature map augmentation could be extended with different techniques, using different fuzzy measures and other methods of generating the new channels.



(a) Insertion point P_1, $C_{feat} = 16$ (b) Insertion point P_2, $C_{feat} = 64$

Fig. 1. Normalized weighting vector evolution.

References

1. Anderson, D.T., Scott, G.J., Islam, M., Murray, B., Marcum, a.R.: Fuzzy choquet integration of deep convolutional neural networks for remote sensing. In: Computational Intelligence for Pattern Recognition, pp. 1–28. Springer (2018)
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
3. Dias, C.A., Bueno, J.C.S., Borges, E.N., Botelho, S.S.C., Dimuro, G.P., Lucca, G., Fernandéz, J., Bustince, H., Drews Junior, P.L.J.: Using the Choquet Integral in the Pooling Layer in Deep Learning Networks. In: Barreto, G.A., Coelho, R. (eds.) Fuzzy Information Processing. pp. 144–154. Springer (2018)
4. Dias, C.A., Bueno, J.C.S., Borges, E.N., Lucca, G., Santos, H., Dimuro, G.P., Bustince, H., Junior, P.L.J.D., Botelho, S.S.C., Palmeira, E.: 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. pp. 224–236. Springer (2019)
5. Du, X., Zare, A.: Multiple Instance Choquet Integral Classifier Fusion and Regression for Remote Sensing Applications. IEEE Transactions on Geoscience and Remote Sensing **57**(5), 2741–2753 (2019)
6. Mann, H.B., Whitney, D.R.: 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)
7. Price, S.R., Price, S.R., Anderson, D.T.: Introducing Fuzzy Layers for Deep Learning. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–6 (2019)
8. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena **60**(1), 259 – 268 (1992)
9. Scott, G.J., Hagan, K.C., Marcum, R.A., Hurt, J.A., Anderson, D.T., Davis, C.H.: Enhanced Fusion of Deep Neural Networks for Classification of Benchmark High-Resolution Image Data Sets. IEEE Geoscience and Remote Sensing Letters **15**(9), 1451–1455 (2018)
10. Scott, G.J., Marcum, R.A., Davis, C.H., Nivin, T.W.: Fusion of Deep Convolutional Neural Networks for Land Cover Classification of High-Resolution Imagery. IEEE Geoscience and Remote Sensing Letters **14**(9), 1638–1642 (2017)
11. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556 (2014)
12. Smith, L.N.: A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. arXiv 1803.09820 (2018)
13. Veal, C., Yang, A., Hurt, A., Islam, M.A., Anderson, D.T., Scott, G., Keller, J.M., Havens, T.C., Tang, B.: Linear Order Statistic Neuron. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–6 (2019)
14. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics **18**(1), 183–190 (1988)