MAX-MIN CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Michael Blot, Matthieu Cord, Nicolas Thome

Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris

ABSTRACT

Convolutional neural networks (CNN) are widely used in computer vision, especially in image classification. However, the way in which information and invariance properties are encoded through in deep CNN architectures is still an open question.

In this paper, we propose to modify the standard convolutional block of CNN in order to transfer more information layer after layer while keeping some invariance within the network. Our main idea is to exploit both positive and negative high scores obtained in the convolution maps. This behavior is obtained by modifying the traditional activation function step before pooling. We are doubling the maps with specific activations functions, called MaxMin strategy, in order to achieve our pipeline. Extensive experiments on two classical datasets, MNIST and CIFAR-10, show that our deep MaxMin convolutional net outperforms standard CNN.

Index Terms— Convolutional neural network, pooling, activation function, invariance, image representation, classification

1. INTRODUCTION

Computer vision has for a long time great interaction with artificial intelligence and machine learning. One of the main examples studied in this article is image classification.

For a while, state of the art algorithms for image classification were based on bag of words (BoW) models [1] [2]. Those algorithms build a visual dictionary from local image descriptors. For any image, local detection values of each word are pooled together to represent the final image feature, which is used as input for a SVM [3] classifier. Several attempts have been made to improve the coding or pooling steps [4, 5, 6, 7].

Recently, a major breakthrough has been revealed with convolutional neural networks (CNN) [8] beating all others models with a huge gap on the ILSVR2012 competition [9]. An appealing feature of deep learning is the ability to learn representations from raw pixels. Today CNN introduced for the first time in [10] and popularized by [11] are widely used is computer vision and stand as the state of the art in image classification on many popular datasets. Recently, attempts have been made to tackle low resolution images [12] and training deep CNNs in a weakly supervised manner [13, 14].

The CNN architecture can be described as a succession of convolutional blocks followed by some fully connected layers. Classically, one convolution block successively applies linear filters, non-linear activation functions (like $ReLU$), and local pooling operations [8].

In BoW, the pooling phase enables to manage many invariances but it is also responsible for a loss of spatial information that make the learning difficult. In deep convolutional architectures, activation functions and pooling may be also questioned. For instance, using a $ReLU$ activation function, all negative information is removed from the convolutional map considered. On the contrary, we propose to modify the CNN layers by introducing a new block to preserve this information from strong negative detections. We assume that keeping both positive and negative evidences for each filter may be of interest for the whole architecture. Inspired by [15, 7], we propose a new pipeline doubling the output map for each linear filter in order to independently process these maps with dedicated activation functions. The resulting deep CNN architecture, called MaxMin CNN, is evaluated against simple CNN on two benchmarks.

2. RELATED WORKS AND MOTIVATIONS

The standard deep CNN introduced in [8] is composed with 5 convolutional blocks followed by 3 fully connected layers with a Softmax function at the end. Each convolutional block has a convolutional filter layer applying a number of filters to the input and concatenates the resulting maps of convolution. This output passes through an activation function that is responsible for increasing the representational power of the network. In image classification the mostly used function is $ReLU$ [8]. Then frequently comes a pooling layer [16]. Aggregating information in a local neighborhood, this step ensures a CNN invariance to small translations. Optionally, others kind of layers such as local response normalisation may be used [8].

Since [8], many CNN architecture improvements have been proposed to boost CNN performances. For instance [17] and [18] improved greatly the performances of CNN on ImageNet by using a deeper and optimized architecture.

Thanks to DGA for funding.

Concerning the pooling function [16] questions about the nature of the information transmitted and proposed parametric functions that include max and average. Varying the parameters they tried to optimise the pooling function but obtained no better results that average or max pooling showing that it is difficult to improve the pooling function itself. They also made a theoretical analysis explaining the conditions on the input distribution involving max pooling working better than average pooling. Still on pooling [19] proposes a method selecting random smaller pooling windows enabling to filter less information that will be exploited with more convolutional layers.

Regarding $ReLU$, it is a non-linear function that sets to zero all negative values after the convolutional layer. This filtering is supposed to facilitate the exploitation of discriminative information by de-noising filter detections. [20] introduced the $PReLU$ as an extension of $ReLU$. It multiplies negative values with a learnable coefficient instead of setting them to zero. This method enables to filter less information at activation layer while keeping the non linear property of $ReLU$ although at the cost of additional learnable parameters.

We explore in this paper a different strategy to improve convolutional block at the $ReLU$ step. As mentioned, $ReLU$ and pooling filter a lot of information from their input. We modify the convolutional block in order to keep more information after the $ReLU$. Indeed, the information from strong negative detection is important and is totally left out after the $ReLU$. Keeping and exploiting this special information is the goal of our method. Inspired by [15, 7] which modifies the pooling by using a histogram of pattern detection instead of just keeping the maximum, we propose to keep high positive and high negative values obtained after filtering in a double $ReLU$ scheme (MaxMin) that we detail in the following.

3. MAXMIN CNN

3.1. MaxMin Net Architecture

With the $ReLU$ strong negative detections give the same information as weak negative detections. Unfortunately strong negative detections can be discriminant. Similarly than with geometrical shapes the semantic of a pattern can be invariant to the negative transformation. Thus it seems natural to presume that for many patterns that have important discriminant power the network learns the filters that detects the patterns as well as their opposites. Our method aims at transmitting directly the negative detections from a filter in order to prevent the network to learn the opposite filter. Indeed for a filter h the negative filter $h^- = -h$ verifies for all input $\mathbf{x}$, $\mathbf{x} * h^- = \mathbf{x} * (-h) = -(\mathbf{x} * h)$. Then if the pattern filtered by h^- is strongly detected on $\mathbf{x}$ then $\mathbf{x} * h^-$ will be high and positive while $\mathbf{x} * h$ will be symmetrically low and negative. Thus conserving the information from the convolution with h either the result is positive or negative will give the information about h^- and enables to use less filter in the architecture.

To implement our strategy, we duplicate the convolutional filter maps (represented is blue in Fig. 1) and multiply the copy by $\times -1$ resulting in the negative version of the detections (in red in Fig. 1). We then concatenate the original maps and their negative copy as shown on Fig. 1. This operation increases the depth of the convolutional layer's output by two compared to classical CNN. We then apply the $ReLU$ normally to the concatenated output and optionally process a pooling operation. This way negative values are not filtered and can be exploited by following layers.

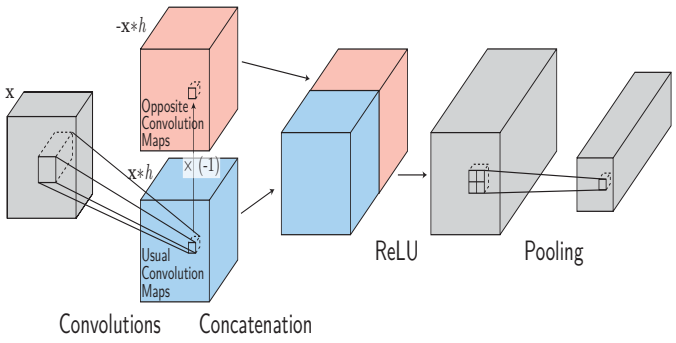

Fig. 1. MaxMin scheme. After the convolutional layer, the resulting maps (in blue) are duplicated negatively (in red). Both stacked maps are concatenated to get the MaxMin output that will pass through the ReLU before pooling.

3.2. Relation with max pooling

After the $ReLU$ is frequently applied max pooling map by map. Let first remark that $ReLU \circ max = max \circ ReLU$. It is possible to commute the two layers without changing the block's output. For our method we notice the fact that

$$\Big(max(ReLU(X)),\ max(ReLU(-X))\Big) = \Big(ReLU(max(X)),\ ReLU(-min(X))\Big)$$

where X is a vector. Thus our method can be interpreted as simply adding an additional information at pooling function with a bi-dimensional output when applied before the $ReLU$. This additional information is the minimum detection on the window taken negatively.

3.3. Discussion

Here we discuss the interest of the method in term of information modeling the generalization improvement.

Modeling: As the size of the output of the convolutional layer is doubled within the convolutional block it is necessary that the size of the filter of the following block is doubled

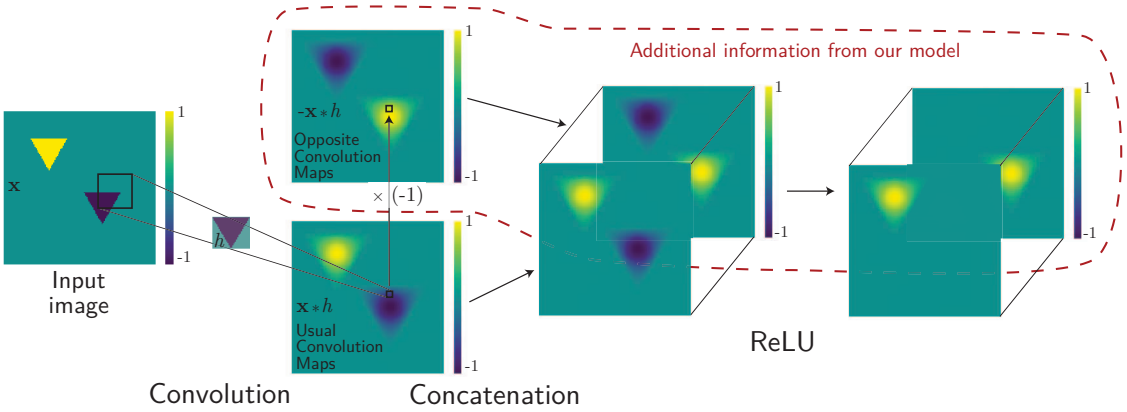

Fig. 2. MaxMin CNN block for one input image with specific patterns. When considering input with 2 specific triangles and a convolution with the filter h, we obtain a map with 2 strong positive and negative answers. Thanks to our MaxMin, we get an additional information (red dotted line) that makes possible to keep both extreme information after ReLU in a joint double map.

too resulting on additional parameters on following filters. It worth mentioning that if those additional parameters are all set to zero the network is equivalent to a simple CNN and our method does not reduce CNN representational power.

At the convolutional layer MaxMin network enforces the transmission of the detection from both the filters and their negative versions. The learning of the negative version of all filters is then not necessary. This is very useful for pattern that are invariant to negative transformations. Fig. 2 shows that different versions of a same shape (yellow and blue triangles) can be transmitted with MaxMin network with only 1 filter, whereas 2 filters are necessary for standard CNN. Thus we can reduce the number of filters and keep a reasonable amount of parameters.

Generalization: On a spatial window, the maximum and the minimum pooling outputs are rarely simultaneously positively high and negatively low. Therefore, $ReLU$ filters very often one of the two values (min or max). This ensures a sparse activation of the network's neurons. This property is known to enhance the generalization performance of neural networks in computer vision as studied in [21] and [22].
Moreover, we claim that our method is able to learn more efficiently the convolutional filters than standard CNN. MaxMin method enforces discriminant patterns to be learnt by back-propagating the error from both positive detections and negative ones. MaxMin networks learn each pattern both from its positive and negative occurrences in the dataset. They learn the pattern filters more accurately and faster than a classical CNN that would learn positive and negative filters independently.

4. EXPERIMENTS

We test our method on two well-known datasets, MNIST and CIFAR-10, and we compare our results with classical CNN.

Learning protocol: All the models and the learning framework are implemented in Lua using Torch 7 library in [23] [1]. For the training, we use the same gradient descent as in [8] with 0.9 of momentum and fixed weight decay (10^{-3} for MNIST and 10^{-4} for CIFAR). We use an equal learning rate for all layers that is being manually reduced when validation error stops decrease. All weights of convolutional filters are initialized from a zero-mean Gaussian distribution with standard deviation 0.01. The number of training epochs depends on the network but is between 30 and 120 for CIFAR-10 and around 250 for MNIST. Top-1 accuracy on test set is computed to evaluate performances.

4.1. MNIST RESULTS

MNIST is a 60,000 images dataset representing 9 handwritten digits. Images are 32×32 pixels in one channel. See [24] for more information. We use the same protocol as [24]: 50,000 images for training and the 10,000 remaining images for testing. To compare our method, we use a LeNet like network with pooling layers. It is composed of three convolutional layers with $ReLU$ activations all followed by max pooling and a local contrast normalisation layers. The filters are of size 5×5 with a stride of 1. There is 64 of them at each convolutional layers. The pooling windows are of size 3×3 for a stride of 2. Those layers are followed by one fully connected layers before a softmax.

For the MaxMin network setting, we simply include a MaxMin layer before the $ReLU$ activation function. We keep the same number of filters paying attention to double their size when needed.

The baseline network obtains a score of 99.34% accuracy. With our MaxMin network, we obtain a score of 99.39% accuracy. Our strategy has 31 errors on the validation set against

[1] Code will be released on Gitub after publication.

# Parameters	MaxMin-CNN	Simple-CNN
$\approx 30K$	73.81	69.98
$\approx 15K$	78.13	74.44
$\approx 60K$	80.03	77.01
$\approx 2M$	81.68	78.11
$\approx 5M$	82.07	79.48
$\approx 15M$	82.69	80.13
$\approx 45M$	82.98	80.41

Table 1. CIFAR-10 results for simple CNN and our MaxMin CNN. Accuracy scores are reported for different deep architectures parametrized by their number of parameters. MaxMin network has systematically better performances than simple CNN for equal complexity (# Parameters).

36 errors for the baseline CNN. It corresponds to an relative improvement of 13.9%. Note that the performance scores on this dataset are very high, and any small performance gain is difficult to obtain.

4.2. CIFAR-10 RESULTS

CIFAR-10 dataset consists of 60,000 32×32 color images (50,000 training images and 10,000 test images) representing 10 objects class like airplanes, trucks or birds. Images vary greatly within each class. See [24] for more information. The baseline network is the one implemented in [25] where we have replaced the average pooling layers with max pooling. The network has three convolutional layers with $ReLU$ activation, all followed by max pooling. All filters have size 5×5 and stride 1 with 32 filters at the first two layers and 64 at the last one. The pooling windows have size 3×3 and stride 2 at all layers. The last max pooling layer is followed by two fully connected layers before a softmax.

The accuracy score obtained using this simple CNN is 74.44% on the test set. When we train an equivalent MaxMin network, we reach an accuracy of 78.62%, showing an improvement of more than 4% of the results.

Robustness to parameter number: As mentioned earlier, our MaxMin layer imposes to double up the size of following filters compared to classical CNN. At fixed number of filters, this implies to increase the number of parameters when including MaxMin layers to CNN. As the number of parameters directly impacts the performances of a CNN, we now compare both methods with constant number of parameters. We use the previous network and modify the number of filters on different convolutional layers in order to vary the total number of parameters. For the MaxMin network, we adapt the number of filters to get comparable amount of parameters as CNN networks. As we intend to compare the quality of the exploitation of the filters by CNN and MaxMin, we pay attention to get the same amount of neurons in the fully connected layers. Only the number of convolutional filters varies from a method to another. We report in table 1 the accuracies obtained for different numbers of parameters for both simple CNN and MaxMin networks. We observe that the MaxMin network systematically outperforms the CNN whatever the number of parameters is with a best performance of 82.98%. This shows the robustness of our method to different network parametrization and demonstrates the ability of MaxMin networks to exploit filter information.

Boosting performances: Current deep models on CIFAR-10 use several kinds of learning tricks such as data augmentation, image preprocessing, dropout, to improve final classification performances.

Our MaxMin deep convolutional model may benefit of the same tricks. As the learning becomes quite more complex and much more time consuming, we do perform only once these optimizations for our MaxMin architecture model on CIFAR-10. We thus apply some translations and flips for data augmentation, zca whitening referenced in [26] to preprocess the images, and dropout from [22] to boost the learning. We use the same network described above with additional local contrast normalisation layer after each pooling.

With this training of our network, we do obtain the accuracy of 90.03% on the test set. This score beats the results reported by [8] that uses a deeper architecture (89%) and reported by [24] that uses 8 networks for prediction (88.79%). On the contrary, our MaxMin scores are obtained with only one network. Our method could also benefit from the use of several nets for prediction in order to reach the performances of recent very deep architectures as [27] or [28] with an accuracy of 92.40%.

5. CONCLUSION

In this paper, we present our new deep CNN architecture, MaxMin-CNN, to better encode both positive and negative filter detections in the net. MaxMin strategy aims at preserving and propagating significant negative detection values through the net. This difference with standard CNN enables us to preserve and transfer more information through the network.

We evaluate and compare our strategy to classical deep CNNs on two benchmarks CIFAR-10 and MNIST. Results demonstrate that our MaxMin networks perform better than CNNs, whatever the configuration considered.

Finally, our MaxMin strategy reaches very good performances on CIFAR-10 outperforming several recent and much more complex deep architectures. The principle is very simple and many deep networks could benefit of this MaxMin strategy with only minor adaptations.

6. REFERENCES

[1] Manjunath B.S., "Netra: a toolbox for navigating large image databases," *ICIP*, October 1997.

[2] J. Sivic, B.C. Russell, A. Efros, A.A.and Zisserman, and W.T Freeman, "Discovering objects and their location in images," *ICCV*, 2005.

[3] V. Vapnik., "Statistical learning theory," *Wiley-Interscience*, 1998.

[4] Florent Perronnin and Christopher R. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, 2007.

[5] Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim, "Top-down regularization of deep belief networks," in *NIPS*, December 2013, pp. 1878–1886.

[6] Hanlin Goh, Nicolas Thome, M. Cord, and Joo-Hwee Lim, "Learning Deep Hierarchical Visual Feature Coding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2212–2225, Dec 2014.

[7] Sandra Eliza Fontes de Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and Arnaldo de Albuquerque Araújo., "Pooling in Image Representation: the Visual Codeword Point of View," *CVIU*, vol. 117, no. 5, pp. 453–465, May 2013.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," *NIPS*, November 2012.

[9] A. Berg, J. Deng, , and L. Fei-Fei, "Large scale visual recognition challenge 2010," *www.imagenet.org/challenges*, November 2010.

[10] K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics, 36(4):193202*, December 1980.

[11] Y. LeCun, L. Bottou, Y. Bengio, , and P. Haffner, "Gradientbased learning applied to document recognition. proceedings," *IEEE, 86(11):22782324*, November 1998.

[12] Marion Chevalier, Nicolas Thome, Matthieu Cord, Jérôme Fournier, Gilles Henaff, and Élodie Dusch, "LR-CNN For Fine-grained Classification with Varying Resolution," in *International Conference on Image Processing (ICIP)*, 2015.

[13] Thibaut Durand, Nicolas Thome, and Matthieu Cord, "MANTRA: Minimum Maximum Latent Structural SVM for Image Classification and Ranking," in *International Conference on Computer Vision (ICCV)*, 2015.

[14] Thibaut Durand, Nicolas Thome, and Matthieu Cord, "WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] S. Avila, N. Thome, M. Cord, E. Valle, and A. de A. Arajo, "Bossa: extended bow formalism for image classification," *ICIP*, September 2011.

[16] Y-Lan Boureau, Jean Ponce, and Yann Lecun, "A theoretical analysis of feature pooling in visual recognition," *ICML*, 2010.

[17] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, November 2015.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," *CVPR*, 2015.

[19] Ben Graham, "Fractional max-pooling," *arXiv:1412.6071*, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv:1502.01852v1*, 2015.

[21] Bruno A. Olshausen and David J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images?," *Nature*, 1996.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *NIPS*, August 2012.

[23] "torch 7," http://torch.ch/.

[24] Dan Cires an, Ueli Meier, and Jurgen Schmidhuber, "Multi-column deep neural networks for image classification," *CVPR*, December 2012.

[25] "matconvnet," http://www.vlfeat.org/matconvnet/.

[26] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," *JMLR*, 2013.

[27] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *NIPS*, 2015.

[28] Rupesh Kumar Srivastava, Klaus Greff, and Jrgen Schmidhuber, "Training very deep networks," *NIPS*, 2015.