Github: https://github.com/Intelligent-Systems-MSc/Modulated-Convolutional-Networks

Modulated Convolutional Networks

Karim HOCINE KARIM.HOCINE@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Anis NEHMAR ANIS.NEHMAR@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Rabah MOULAI RABAH.MOULAI@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Abdelali ISLI ABDELALI.ISLI@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Editor:

Abstract

In the context of our machine learning project, which consists of implementing a new convolutional layer described in a 2018 CVPR article, we had to implement a new convolution method named "Modulated convolution" based on filters obtained by modulating a binary filter with another filter called M-filter. We also propose a novel loss function used to better approximate the convolution operation using the binary weights. The main purpose of this modulated convolution is to improve the memory management of convolutional neural networks, this can offer better portability to CNNs.

1. Introduction

The objective of this work is to try to implement a new type of convolutional neural network presented in [1]. Deep Convolution Neural Networks (DCNNs) are attracting a lot of attention because of their ability to acquire advanced feature representations directly from raw pixels, which makes them useful for a lot of computer vision problems.

Despite the high efficiency DCNNs, they are often linked to the cost of expensive training and complex model parameters making them unsuitable for most embedded devices which are limited in storage space.
In order to improve the size of required storage space used in DCNNs and their portability, Xiaodi Wang and al. [1] introduced a modulation process based on the use of binarized filters instead of weights. Binary weights are used is this case to simplify the convolutional operations by reducing the number of computed parameters while training the model. Modulated filters are incorporated into DCNNs to better approximate the unbinarized convolution. The M-filters and the binarized filtres are combined to reconstruct the unbinarized filter leading to a new architecture to calculate the Convolutional Neural networks model.

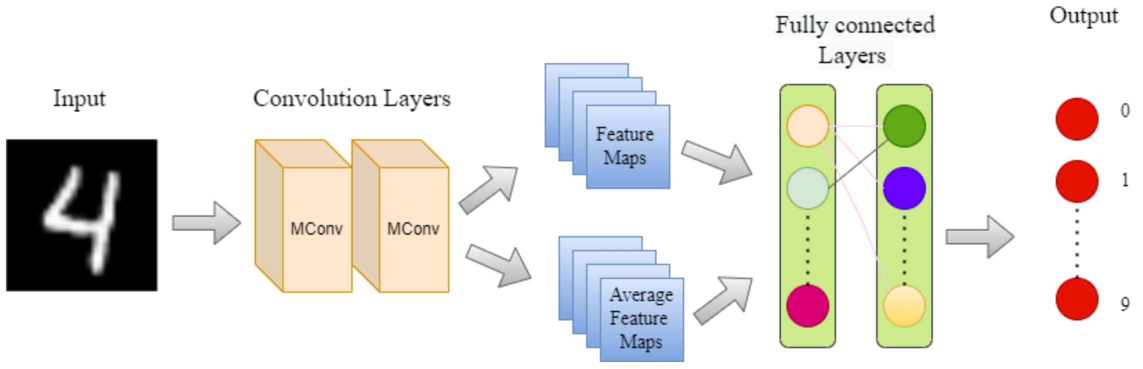

Figure 1: Modulated Convoltional Networks

2. Presentation of the method

Based on binarized convolution filters and M-filters, modulated convolutions networks are constructed as an efficient alternative to basic convolutional networks.Only one M-filter is shared between all the layers of the network which leads to a significant reduction of the number of parameters of the model. Architectures based on MCNs can achieve great performance even if the parameters are highly compressed.

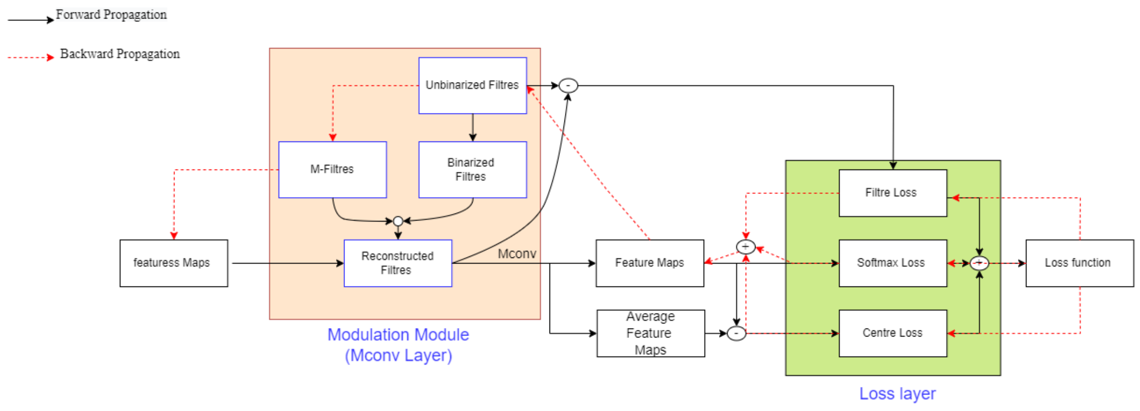

Figure 2: Modulated Convoltional Networks architecture

2.1 Loss functions

The complete loss function is defined as:

$$Loss = L_S + L_M \tag{1}$$

Where L_S is a conventional loss function like the softmax loss function and L_M is a specific loss function introduced to constraint CNNs to have binarized weights [1].

$$L_M = \frac{\theta}{2} . \sum_{i,l} ||C_i^l - \hat{C}_i^l o M^l||^2 + \frac{\lambda}{2} . \sum_m ||f_m(\hat{C}, \vec{M}) - \overline{f_m}(\hat{C}, \vec{M})||^2 \tag{2}$$

- θ and λ: Trained parameters.

- C_i^l: Unbinary filters of the lth convolutional layer.
- $\hat{C}_i^l$: Binary filters of the lth convolutional layer.
- M^l: Modulation filter (M-filter) shared by all C_i^l in the lth convolutional layer.
- $f_m(\hat{C}, \vec{M})$: Feature map of the last convolutional layer.
- $\overline{f_m}(\hat{C}, \vec{M})$: Class specific mean feature map of previous samples.
- $\vec{M} = \{M^1, ..., M^N\}$: Are the M-filters set across all layers.

L_M combines two loss functions. The filter loss (first term) used to evaluate the loss the reconstruction of the unbinarized filter. Due to the disturbance caused by the binarized process a second term is added similar to the center loss. It is used to evaluate the intra-class compactness.

2.2 Forward propagation

2.2.1 Reconstructed Filters

First of all we build the convolutional filters used in our MCNs, the size of this filter is K, W, W and for K it is always and equal to 4, so each time we have to modify the input due to the network, for example it is we have an image at gray level by duplicating the image by 4.

we do operation o between the jth plane of the M-filter and the bianarized filter which also of size K, W, W to reconstruct the non-binarized filters Q.

$$\hat{C}_i o M = \sum_j^k \hat{C}_i * M'_j \tag{3}$$

- $M_j = (M_j, .., M_j)$ is a 3D matrix built based on K copies of the 2D matrix Mj with j=1,..,K.
- M is a learned weight matrix

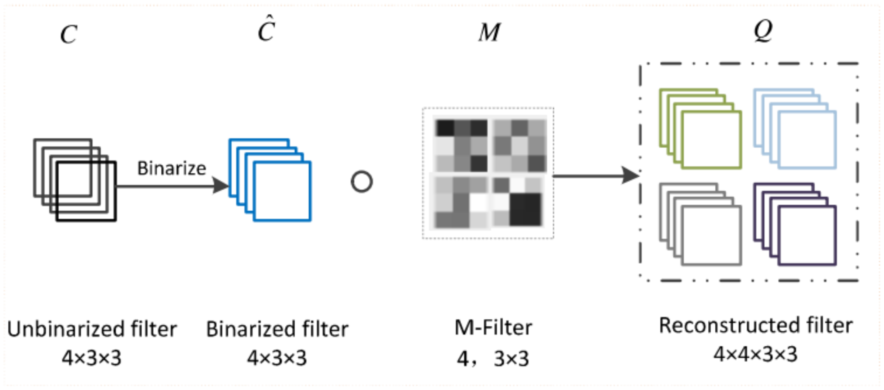

Figure 3: Modulation process

$$Q_i j = \hat{C}_i * M'_j \tag{4}$$

$$Q_i = \{Q_{i1}, ..., Q_{iK}\} \tag{5}$$

Q used to mitigate the in formation loss problem caused by the binarized process, is implemented to approximate the unbinarized filters C_i.

$$\hat{c}_i = \begin{cases} a_1 & \text{if } |c_i - a_1| < |c_i - a_2|, \\ a_2 & \text{otherwise} \end{cases} \tag{6}$$

where :

- c_i in an element of C_i and $\hat{c}_i$ is an element of $\hat{C}_i$.
- a_1 and a_2 are each the center of a cluster calculated by k-means clustering algorithm on the data of unbinarized filters for 10 epochs.

c_i now can be represented as a_1 or a_2 (binary values) to save storage space.

2.2.2 Mconv module

The Mconv consists in the convolution of the *lth* feature map with the Q^l reconstructed filter of the *lth* layer. The result is the feature map of the $(l+1)th$ layer.

$$F^{l+1} = Mconv(F^l, Q^l) \tag{7}$$

Considering the dataset used as the MNIST Dataset, When the network is built the size of the input is 28×28. The first channel of every input is copied $K = 4$ times, resulting in the input size of $4 \times 28 \times 28$.

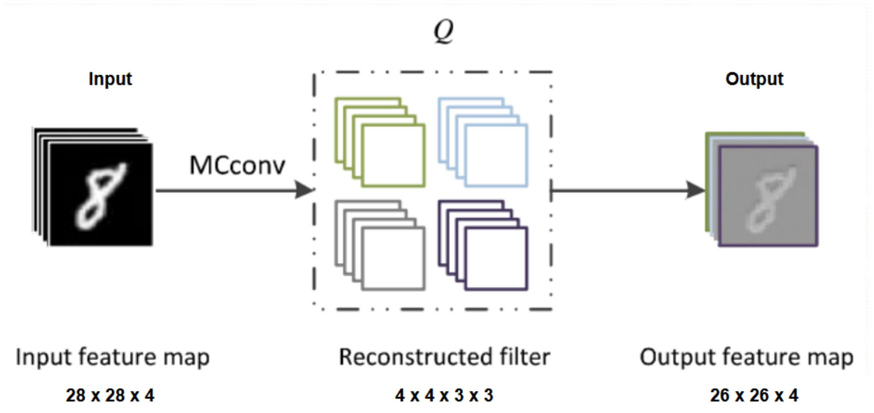

Figure 4: MCNs Convolution (Mconv)

$$F^{l+1}_{h,k} = \sum_{i,g} F^l_g \circledast Q^l_{ik} \tag{8}$$

$$F^{l+1}_h = \{F^{l+1}_{h,1}, ..., F^{l+1}_{h,K}\} \tag{9}$$

where :

- $F^{l+1}_{h,k}$: *kth* channel of the *hth* output feature map in the $(l+1)th$ convolutional layer.

- F_g^l: gth input feature map of the lth convolutional layer.

The simplest case is $h = 1$,$g = 1$ where after Mconv with one reconstructed filter, the number of channels of the output feature map is the same as that of the input feature map.
Another application of the Mconv is the convolution of multiple reconstructed filters where one output feature map is the sum of of the convolution between all the input feature maps and the reconstructed filters in the corresponding group (Fig17)

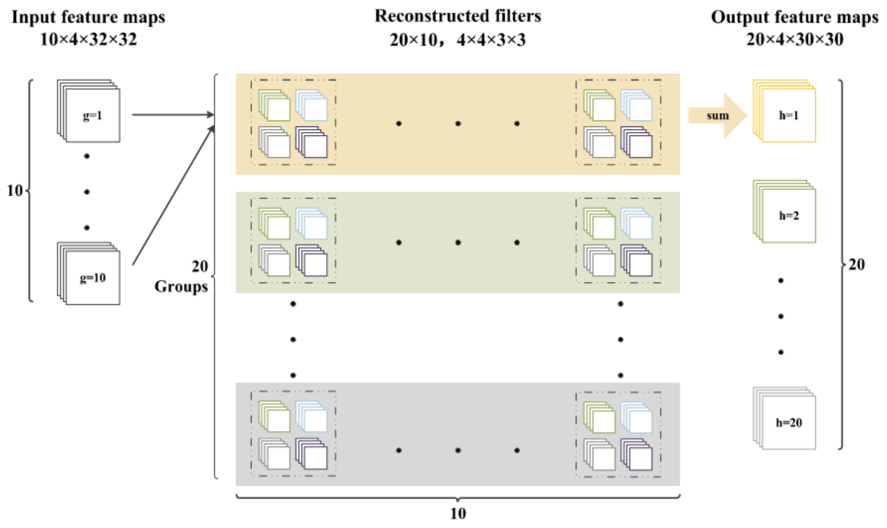

Figure 5: Modulated Convolutional layer with 20 group of filters

2.3 Backward propagation updating

Updating weights in MCNs is done on each layer, the two types of filters used are learned jointly, first unbinarised filter are updated, second the M-filter in each block.

2.3.1 Updating unbinarized filters

Updating unibinarised filters is done with a simple gradient descent:

$$C_i \leftarrow C_i - \eta \delta_C \tag{10}$$

δ_C is the gradient of the loss function over an unibinarised filter C$_i$:

$$\delta_C = \frac{\partial L}{\partial C_i} = \frac{\partial L_S}{\partial C_i} + \frac{\partial L_M}{\partial C_i} \tag{11}$$

$$\frac{\partial L_S}{\partial C_i} = \frac{\partial L_S}{\partial Q}\frac{\partial Q}{\partial C_i} = \sum_j \frac{\partial L_S}{\partial Q_{ij}}.M'_j \tag{12}$$

$$\frac{\partial L_M}{\partial C_i} = \theta \sum_j (C_j - \hat{C}_i o M_{j)} \tag{13}$$

$\hat{C}_i$ is the binarised convolutional filter that correspond to C_i

2.3.2 Updating M-filters

The M-filters are updated with the already updated C filters. Knowing that δ_M is the gradient of M-filters, we proceed as before with a simple gradient descent.

$$\delta_M = \frac{\partial L}{\partial M} = \frac{\partial L_s}{\partial M} + \frac{\partial L_M}{\partial M} \tag{14}$$

η_2 : learning rate

$$M \leftarrow |M - \eta_2 \delta_M| \tag{15}$$

Moreover to calculate the 2 terms of the gradient we have these two expressions :

$$\frac{\partial L_s}{\partial M} = \frac{\partial L_s}{\partial Q} + \frac{\partial Q}{\partial M} = \sum_{i,j} \frac{\partial L_s}{\partial Q_{i,j}}.C_i \tag{16}$$

$$\frac{\partial L_M}{\partial M} = -\theta \sum_{i,j} (C_i - \hat{C}_i o M_j).\hat{C}_i \tag{17}$$

3. Dataset

The MNIST database (Modified National Institute of Standards and Technology database) [2]. The MNIST is dataset of handwritten digits. It has a training set of 60,000 examples and a test set of 10,000 examples. This dataset is one of the most widely used datasets for image classification, and it's available from a variety of places. It consists of 28x28 pixel images of handwritten digits, such as:

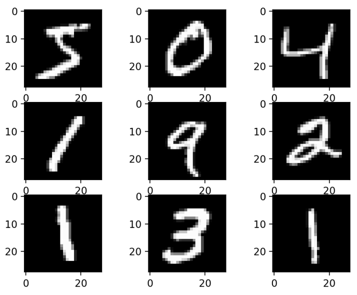

Figure 6: MNIST Handwritten Digit Classification

4. Experiments

This project is an attempt to replicate the results of the work [1]. To do so, we chose to program in object-oriented Python using the numpy library. We started to program the different layers represented in figure 7.

The first step was to code a layer class. From this class all other classes will inherit. This class has two methods "forward" and "backward" that will allow us in the following to realize a forward propagation and a backward propagation. Then from the parent class, we program the Mconv layer, a Dense layer and a batch normalization layer. After that we had to code the different activation functions and the loss functions and their derivatives. After coding the different layers blocks and

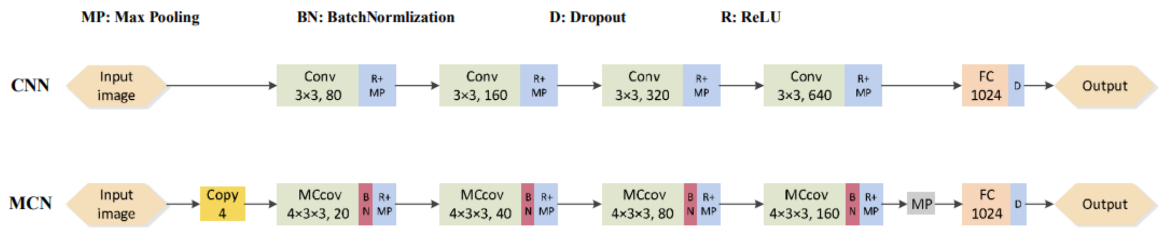

Figure 7: Network architecture of CNNs and MCNs.

activation functions, we have made a pre-processing to our data. This preprocesses consisted in coding the labels in a one hot format. Once, the data are ready to be used we implemented a training loop and a test loop.

5. Conclusion

The above-mentioned Mconv module can be integrated into different neural network architectures. As an alternative to simple convolution layers. The researchers have implemented the architecture presented before to be able to compare the performances of the Mconv module with a standard convolution layer. Other comparisons have been made with more complex architectures such as Unet or resnet to be able to generalize the results obtained. Thanks to the different performance tests, modulated convolutional networks have managed to reduce the memory used by a factor of 32 compared to equivalent CNNs, which greatly improves the portability of convolutional neural networks without compromising on performance. Despite our various attempts to reproduce the results obtained by the researchers, we were unable to train our model.

References

[1] Bertrand Alain. *"La branche armée du féminisme: les Amazones"*. http://revuelabyrinthe.org/document742.html. Mis en ligne le 20 avril 2005, consulté le 23 septembre 2007.

[2] Baochang Zhang XIaodi Wang and al. "Modulated Convolutional Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA* (2016). DOI: `10.1109/CVPR.2018.00094`.