# Modulated Convolutional Networks

Xiaodi Wang and Baochang Zhang
School of Automation Science and Electrical Engineering
Beihang University, Beijing, China
bczhang@buaa.edu.cn

Ce Li
Department of Computer Science and Technology
China University of Mining & Technology, Beijing, China
celi@cumtb.edu.cn

Rongrong Ji*
School of Information Science and Engineering
Xiamen University, Fujian, China
correspondence rrji@xmu.edu.cn

Jungong Han
School of Computing & Communications
Lancaster University, LA1 4YW, UK
jungonghan77@gmail.com

Xianbin Cao
School of Electronic and Information Engineering
Beihang University, Beijing, China
xbcao@buaa.edu.cn

Jianzhuang Liu
Noah's Ark Lab
Huawei Technologies Co. Ltd., China
liu.jianzhuang@huawei.com

## Abstract

*Despite great effectiveness of very deep and wide Convolutional Neural Networks (CNNs) in various computer vision tasks, the significant cost in terms of storage requirement of such networks impedes the deployment on computationally limited devices. In this paper, we propose new modulated convolutional networks (MCNs) to improve the portability of CNNs via binarized filters. In MCNs, we propose a new loss function which considers the filter loss, center loss and softmax loss in an end-to-end framework. We first introduce modulation filters (M-Filters) to recover the unbinarized filters, which leads to a new architecture to calculate the network model. The convolution operation is further approximated by considering intra-class compactness in the loss function. As a result, our MCNs can reduce the size of required storage space of convolutional filters by a factor of 32, in contrast to the full-precision model, while achieving much better performances than state-of-the-art binarized models. Most importantly, MCNs achieve a comparable performance to the full-precision Resnets and WideResnets. The code will be available publicly soon.*

## 1. Introduction

Deep convolutional neural networks (DCNNs) have gained much attention with their capability of learning powerful feature representations directly from raw pixels, thereby facilitating many computer vision tasks. Despite a purely data-driven technique that can learn robust representations from data, DCNNs usually come with the cost of expensive training and complex model parameters. For instance, the sizes of most DCNNs' models for vision applications are easily beyond hundreds of megabytes, which makes them impractical for most embedded platforms. This is fundamentally attributed to the way of filter designing [2, 19], generating many redundant parameters.

Binary filters instead of using real-value weights have been investigated in DCNNs to compress the deep models [15, 4, 3, 9]. A complete binarization process for DCNNs is exploited to approximate floating-point weights with binaries [15, 4, 3]. Inspired by the well-known local binary pattern (LBP), local binary convolution (LBC) layers are presented in [9] that approximate the non-linearly activated response of a standard convolutional layer. While in [3], BinaryConnect uses the real-valued version of the weights as a key reference for the binarization process. Later based on BinaryConnect, BinaryNet is introduced to train CNNs

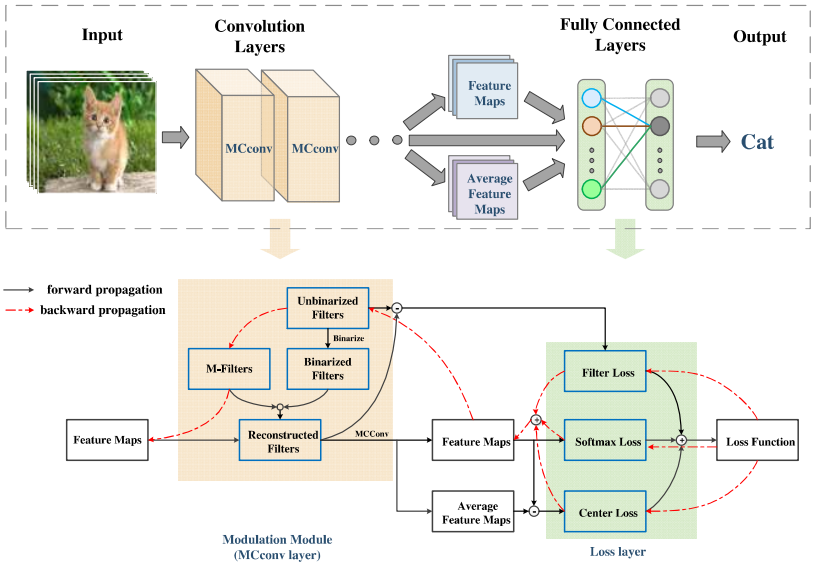



Figure 1: Modulated Convolutional Networks (MCNs). MCNs are designed based on the binarized convolutional filters and the modulation filters (M-Filters). M-Filters are particularly designed to approximate the unbinarized convolutional filters in the end-to-end framework. Due to the operation of M-Filter (matrix) can be shared at each layer, the model size of MCNs is rather small. To alleviate the disturbance caused by the binarized process, the intra-class compactness based on the center loss function is further deployed to enhance the performance. The red arrows are used to show the back propagation process. By considering the filter loss, center loss and softmax loss in an unified framework, we achieve much better performance than state-of-the-art binarized models. Most importantly, our MCNs based on a highly compressed model also achieve a comparable performance than well-known full-precision Resnets and WideResnets. The reconstructed filters are only used for easy presentation of the computation, which are not saved for testing, while the M-Filters and binarized filters are kept in the final model.

with binary weights, where the activations are triggered at running time while parameters are computed at training time. In [15], XNOR-Network is presented where both the weights and inputs attached to the convolution are approximated with binary values, which allows an efficient way of implementing convolutional operations, particularly by reconstructing the unbinarized filters with a single scaling factor. From the concept perspective, our idea is similar to those methods, because all attempt to simplify the convolution procedure via binarized filters and/or approximate the original unbinarized filters. However, the ways of approximating the unbinarized filters are clearly different, where the unbinarized filters are reconstructed using binary filters with a single scaling factor [15] while ours accomplishes it by a set of M-Filters together with the binary filters. On the one hand, M-Filters with the format of a matrix, rather than a single value representing the scaling factor, lead to an unique architecture with more precise estimate of the original convolutional filters through minimizing the filter loss. On the other hand, the approximation based on M-Filters can allow us to consider the intra-class compactness during the optimization procedure, enabling to further improve the convolution operations.

Aiming to improve the portability of DCNNs, this paper reduces the model storage space via binarized filters. Unlike previouse work based on a single scaling factor as shown in [15], we incorporate modulation filter (M-Filter), into DCNNs so as to better approximate the convolution. To this end, a simple and unique modulation process is designed, which is replicable at each layer and can be solved within the same pipeline of the back propagation algorithm. In addition, we further consider the intra-class compactness in the loss function, and obtain the modulated convolutional networks (MCNs) as shown in Fig. 1. Both the M-Filters and binarized filters can be jointly optimized and obtained in an end-to-end learning framework, leading to a compact and portable deep learning architecture. Thanks to the low model complexity, such an architecture is less prone to be over-fitting and suitable for resource-constrained environments. To be specific, our MCNs reduce the required storage space of a full-precision model by a factor of 32, while achieving the best performance so far, as compared to the

existing binarized filters based CNNs. In summary, the contributions of this paper are as follows:

(1) The M-Filters are employed to reconstruct the unbinarized filters, leading to a new architecture to calculate the CNNs model. The convolution operation is further improved by considering the intra-class compactness in the loss function, making the performance comparable to the full-precision model.

(2) We manage to solve MCNs in an end-to-end framework. This highly compressed model outperforms all the state-of-the-art binarized models and is comparable to the well-known full-precision ResNet on CIFAR, MNIST, SVHN and 100-class ImageNet databases in terms of classification accuracy.

## 2. Modulated Convolutional Networks

We design an architecture in MCNs based on the binarized convolutional filters and M-Filters. M-Filters are particularly designed to approximate the unbinarized convolutional filters in the end-to-end framework. Each layer shares only one M-Filter and it leads to significant reduction of the network model. To alleviate the disturbance caused by the binarized process, the intra-class compactness based on the center loss function is further deployed to enhance the performance. With two schemes mentioned above, the performance drop is marginal even when the learnable network parameters are highly compressed.

### 2.1. Loss Function of MCNs

In order to constrain CNNs to have binarized weights, we introduce a new loss function in MCNs. Two aspects are considered: the unbinarized convolutional filters are reconstructed based on binarized filters; the intra-class compactness is incorporated based on the output features. In addition to Table 1, we further introduce the variables used in this section: $C_i^l$ are the unbinary filters of the $l$th convolutional layer, $l \in \{1, ..., N\}$; $\hat{C}_i^l$ denote the binarized filters corresponding to $C_i^l$; $M^l$ denotes the modulation filter (M-Filter) shared by all $C_i^l$ in the $l$th convolutional layer and $M_j^l$ represents the $j$th plane of $M^l$; $\circ$ is a new plane based operation (Eq. 3) which is defined in the next section. We then have the first part of the loss function for minimization:

$$L_M = \frac{\theta}{2} \sum_{i,l} \|C_i^l - \hat{C}_i^l \circ M^l\|^2 + \frac{\lambda}{2} \sum_m \|f_m(\hat{C}, \vec{M}) - \overline{f}(\hat{C}, \vec{M})\|^2, \quad (1)$$

where $\vec{M} = \{M^1, ..., M^N\}$ are the M-Filters, and $\hat{C}$ is the binarized filter set across all layers. The operation $\circ$ defined in Eq. 3 is used to approximate the unbinarized filters based on the binarized filters and M-Filters, leading to the filter loss shown in the first term on the right of Eq. 1. The second term on the right is similar to the center loss used to evaluate the intra-class compactness, which is used for the reason that the binarization process causes feature variations. $f_m(\hat{C}, \vec{M})$ denotes the feature map of the last convolutional layer for the $m$th sample, and $\overline{f}(\hat{C}, \vec{M})$ denotes the class-specific mean feature map of previous samples. Note that we do not approximate the original features, since MCNs are an end-to-end deep model, based on which the original feature maps are unavailable, and thus only the center loss is deployed. To reduce the storage space, after training we only keep the binarized filters and the shared M-Filters (quite small) to calculate the feature maps. Finally, we define the complete loss function $L$ as:

$$L = L_S + L_M, \quad (2)$$

where $L_S$ is the conventional loss function, e.g., softmax loss.

### 2.2. Forward Propagation with Modulation

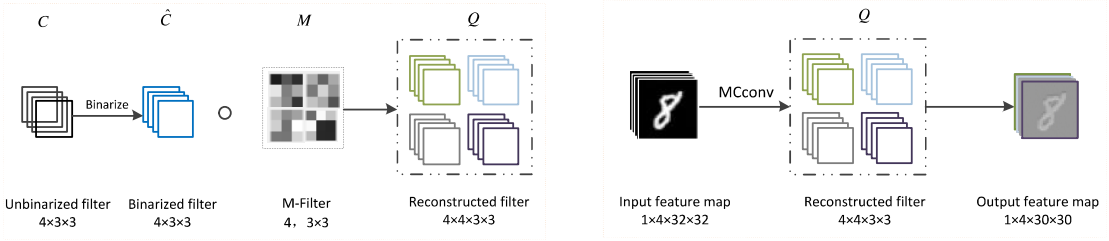
#### 2.2.1 Reconstructed Filters

We first design the specific convolutional filters used in our MCNs. We deploy the 3D filter across all layers with the size of $K \times W \times W$ (one filter), which has $K$ planes, and each of the planes is a $W \times W$-sized 2D filter. To use such kind of filters, we extend the input channels of the network, e.g., from RGB+X, where X can be any of RGB for $K = 4$. By doing so, we can easily implement our MCNs in the Torch platform. After the extension process, we directly deploy our filters in the convolution process, whose details concerning the MCNs convolution are illustrated in Fig. 2(b).

To reconstruct the unbinarized filters, we introduce a modulated process based on the M-Filters and the binarized filters. An M-Filter is a matrix serving as the weight of the binarized filters, which is also with the size of $K \times W \times W$. Let $M_j$ be the $j$th plane of the M-Filter. We define the operation $\circ$ for a given layer as:

$$\hat{C}_i \circ M = \sum_j^K \hat{C}_i * M_j^{'}, \quad (3)$$

where $M_j^{'} = (M_j, ..., M_j)$ is a 3D matrix built based on $K$ copies of the 2D matrix $M_j$ with $j = 1, ..., K$. $*$ is the element-wise multiplication operator, also named Schur product operation. In Eq. 3, $M$ is a learned weight matrix, which is used to reconstruct the convolutional filters $C_i$ based on $\hat{C}_i$ and the operation $\circ$. And it leads to the filter loss in Eq. 1. An example of the filter modulation is shown in Fig. 2(a). In addition, the operation $\circ$ results in a new matrix (named reconstructed filter) , i.e., $\hat{C}_i * M_j^{'}$, which is elaborated in the following. We define:




(a) Modulation process based on an M-Filter

(b) MCNs Convolution (MCconv)

Figure 2: Left shows the modulation process based on an M-Filter to obtain a reconstructed Filter $Q$. Right illustrates an example of MCNs convolution with $K = 4$ planes. The number of planes of the M-Filter is the same as the number of channels of the feature map. In this paper, a feature map is defined as a 3D matrix with 4 channels.

Table 1: A brief description of variables and operators used in the paper

| | | |
|---|---|---|
| $C$: unbinarized filter | $\hat{C}$: binarized filter | $M$: modulation filter (M-Filter) |
| $Q$: reconstructed filter | $\vec{M}$: M-Filters across all layers | |
| $i$: filter index | $j$: plane index | $K$: number of planes for each filter |
| $m$: sample index | $l$: layer index | $N$: number of layers |
| $g$: input feature map index | $h$: output feature map index | |

$$Q_{ij} = \hat{C}_i * M_j^{'}, \tag{4}$$

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

In testing, $Q_i$ is not pre-defined but is calculated based on Eq. 4. An example is shown in Fig. 2(a). $Q_i$ is introduced to approximate the unbinarized filters $C_i$ to alleviate the information loss problem caused by the binarized process. In addition, we further require $M \geq 0$ to simplify the reconstructed process.

The binarized filter in Eq. 4 is calculated based on the nearest neighbor method as:

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

where $c_i$ is an element of $C_i$, and $\hat{c}_i$ denotes the corresponding element in $\hat{C}_i$. In Eq. 6, $a_1$ and $a_2$ are calculated by an offline k-means clustering algorithm on the data of unbinarized filters after 10 epochs. Though $c_i$ is a floating number, it can be represented as a binary value ($a_1$ or $a_2$) to save the storage space. For other quantization process (see the experimental section), the same nearest neighbor method is also deployed.

#### 2.2.2 Forward Propagation of MCNs based on the MCconv Module

In MCNs, the reconstructed filters $Q^l$ in the $l$th layer are used to calculated the output feature maps $F^{l+1}$ as:

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

where $MCconv$ denotes the convolution operation implemented as a new module. A simple example of forward convolutional process is described in Fig. 2(b), where there is one input feature map with one generated output feature map. In MCconv, the channels of one output feature map are generated as follows:

$$F_{h,k}^{l+1} = \sum_{i,g} F_g^l \otimes Q_{ik}^l, \tag{8}$$

$$F_h^{l+1} = (F_{h,1}^{l+1}, ..., F_{h,K}^{l+1}), \tag{9}$$

where $\otimes$ denotes the convolution operation; $F_{h,k}^{l+1}$ is the $k$th channel of the $h$th feature map in the $(l+1)$th convolutional layer. $F_g^l$ denotes the $g$th feature map in the $l$th convolutional layer. In Fig. 2(b), $h = 1$ and $g = 1$, where after MCconv with one reconstructed filter, the number of the channels of the output feature map is the same as that of the input feature map.

Fig. 3 illustrates another example of MCNs convolution with multiple feature maps. One output feature map is the sum of the convolution between all the 10 input feature maps and 10 reconstructed filters in the corresponding group. For example, for the first output feature map, $h = 1, i = 1, ..., 10, g = 1, ..., 10$, and for the second output feature map, $h = 2, i = 11, ..., 20, g = 1, ..., 10$.

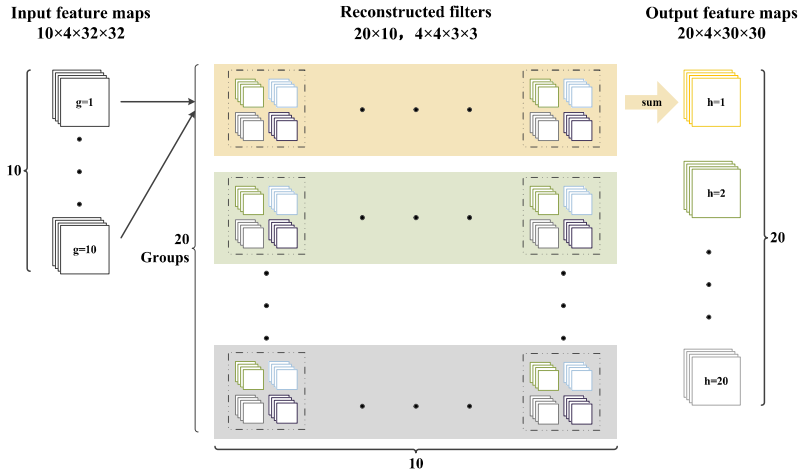



Figure 3: MCNs Convolution (MCconv) with multiple feature maps. There are 10 and 20 feature maps in the input and the output, respectively. The reconstructed filters are divided into 20 groups and each group contains 10 reconstructed filters, corresponding to the number of feature maps and MC feature maps respectively.

When the first convolutional layer is considered, the input size of the network is $32 \times 32$ [1]. First, each channel of the image is copied $K = 4$ times, resulting in the new input of size $4 \times 32 \times 32$ to the whole network.

The advantages of our MCconv module lie in that the numbers of the input and output channels in every feature map are the same, making such a module be replicated and so MCNs be easily implemented.

## 2.3. Back-propagation Updating

In MCNs, what need to be learned and updated are the unbinarized filters $C_i$ and M-Filters $M$. These two kinds of filters are jointly learned. In each convolutional layer, MCNs update the unbinarized filters first, and then the M-Filters.

### 2.3.1 Updating Unbinarized Filters

We define $\delta_C$ as the gradient of the unbinarized filter $C_i$, and have:

$$\delta_C = \frac{\partial L}{\partial C_i} = \frac{\partial L_S}{\partial C_i} + \frac{\partial L_M}{\partial C_i} + \theta(\widehat{C}^{[k]} + \eta_1 \delta_{\widehat{C}}^{[k]} - C^{[k]}), \quad (10)$$

where $\widehat{C}^{[k]} + \eta_1 \delta_{\widehat{C}}^{[k]} - C^{[k]}$ is used to further regularize the loss function, and we have:

$$C_i \leftarrow C_i - \eta_1 \delta_C, \quad (11)$$

[1] We only use one channel of gray-level images ($3 \times 32 \times 32$)

where $L$ is the loss function, and $\eta_1$ is the learning rate. Further we have:

$$\frac{\partial L_S}{\partial C_i} = \frac{\partial L_S}{\partial Q} \cdot \frac{\partial Q}{\partial C_i} = \sum_j \frac{\partial L_S}{\partial Q_{ij}} \cdot M_j', \quad (12)$$

$$\frac{\partial L_M}{\partial C_i} = \theta \sum_j (C_i - \hat{C}_i \circ M_j), \quad (13)$$

where $\hat{C}_i$ is the binarized convolutional filter corresponding to $C_i$.

### 2.3.2 Updating M-Filters

We further update the M-Filter $M$ with $C$ fixed. $\delta_M$ is defined as the gradient of $M$, and we have:

$$\delta_M = \frac{\partial L}{\partial M} = \frac{\partial L_S}{\partial M} + \frac{\partial L_M}{\partial M}, \quad (14)$$

$$M \leftarrow |M - \eta_2 \delta_M|, \quad (15)$$

where $\eta_2$ is the learning rate. Further we have:

$$\frac{\partial L_S}{\partial M} = \frac{\partial L_S}{\partial Q} \cdot \frac{\partial Q}{\partial M} = \sum_{i,j} \frac{\partial L_S}{\partial Q_{ij}} \cdot C_i, \quad (16)$$

Based on Eq. 1 and we have:

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

The details about the derivatives with respect to the center loss can be found from [16]. The above derivations show that MCNs are learnable with the BP algorithm.

## 3. Implementation and Experiments

We evaluate the effectiveness of MCNs on 5 datasets, MINST, SVHN, CIFAR-10, CIFAR-100 and ImageNet. It can be applied to any CNNs. Particularly, our new modulation module is applied to Wide-ResNets [17]. In our experiments, we use MCNs with four convolutional layers on MINIST. In our experiments, 4 NVIDIA GeForce GTX 1080ti GPUs are used. In what follows, the term U-MCNs is short for the unbinarized MCNs that are the full-precision MCNs implemented only based on $L_S$ without the binarized process involved.

### 3.1. Datasets and Implementation Details

**Datasets:** The MINIST [13] dataset is composed of a training set of 60,000 and a testing set of 10,000 $32 \times 32$ gray-scale images of hand-written digits from 0 to 9.

CIFAR-10 [10] is a natural image classification dataset containing a training set of 50,000 and a testing set of 10,000 $32 \times 32$ color images across the following 10 classes: airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships, and trucks. Differently CIFAR-100 consists of 100 classes.

The Street View House Numbers (SVHN) dataset [14] is a real-world image dataset taken from Google Street View images. It contains MNIST-like 32x32-sized images centered around a single character, which however includes a plethora of challenges like illumination changes, rotations and complex backgrounds. The dataset consists of 600,000 digit images: 73,257 digits for training, 26,032 digits for testing, and 531,131 additional images. The additional images are not used in the training of MCNs.

The ImageNet ILSVRC-2012 classification dataset [5] consists of 1000 classes, with 1.28 million images for training and 50,000 images for validation. Different from MNIST, SVHN and CIFAR, ImageNet consists of images with much higher resolutions. In addition, each image usually contains more than one attribute, which may have a large impact on the classification accuracy. We follow LBCNN to use a 100-class subset of ImageNet2012 [5] to evaluate our proposed method. The 100 classes are randomly selected from the full ImageNet dataset, and similar subsets is also used in [1][9][11].

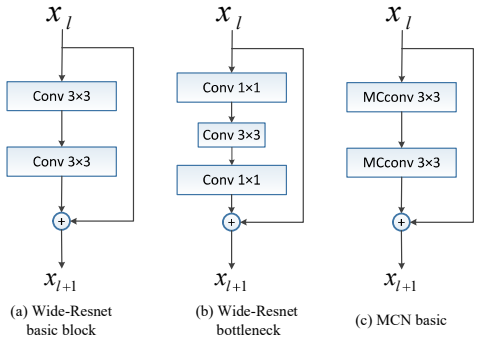



Figure 4: Residual blocks. (a) and (b) are for Wide-ResNets. (c) A basic block for MCNs.

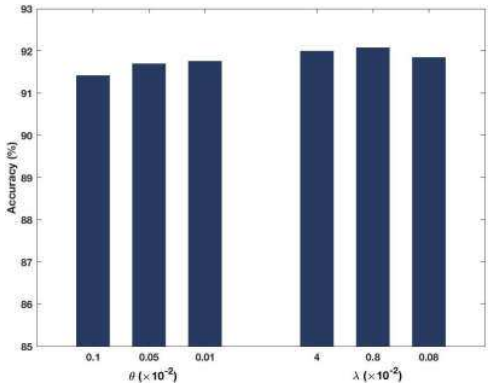



Figure 5: Accuracy with different $\theta$ and $\lambda$.

**Implementation Details:** On all the datasets, the size of each M-Filter and also the convolutional filters is $4 \times 3 \times 3$ ($K = 4$). We replace the spatial convolution layers with MCconv modules, as shown in Fig. 2. In all the experiments, we adopt Max-pooling and ReLU after the convolution layers, and a dropout layer [8] after the FC layer to avoid over-fitting. On CIFAR-10, CIFAR-100, SVHN and 100-class ImageNet datasets, we evaluate MCNs based on Wide-ResNets. The basic blocks in Wide-ResNets and MCNs are shown in Fig. 4. The Wide-ResNets divide the whole network into 4 stages. The bottleneck structure is not used in MCNs since the 1x1 kernel does not propagate any M-filter information. The structures of both the Wide-ResNets and MCNs are the same except that the Conv in Wide-ResNets is replaced by MCconv. The initial values of $\eta_1$ and $\eta_2$ are set to 0.1 and 0.01 respectively. The learning weight decay is set to 0.2.

**$\theta$ and $\lambda$:** In Eq. 2, $L_S$ and $L_M$ are balanced by $\theta$ and $\lambda$ which are related to the filter loss and center loss. The effect of the parameters $\theta$ and $\lambda$ are evaluated on the CIFAR-10 for a 20-layer MCN with width 16-16-32-64, the architecture detail of which can be found in [17]. The Adadelta optimization algorithm [18] is used during the training process, with the batch size 128. Using different values of $\theta$, the performance of the MCNs is shown in Fig. 5. First only the effect of $\theta$ is evaluated and then the center loss is implemented based on a finetuning process. It is observed that the performance is stable with varying $\theta$ and $\lambda$.

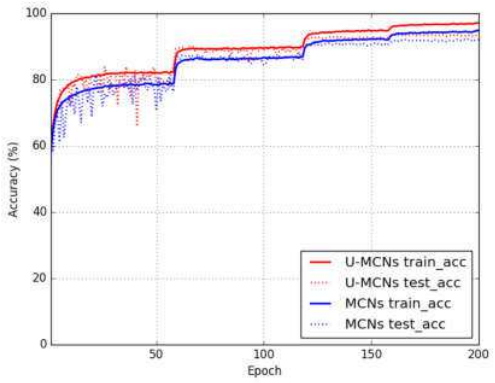



Figure 6: Training and testing curves.

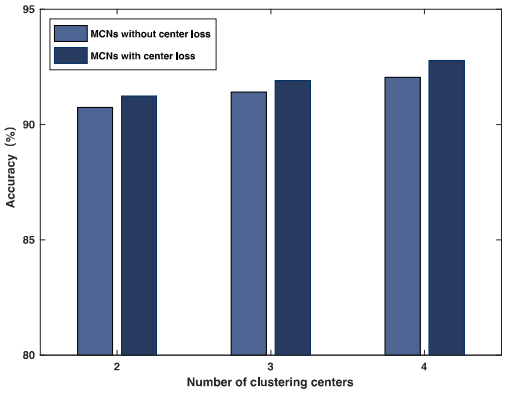



Figure 7: Accuracy with different numbers of clustering centers for 20-layer MCNs with width 16-16-32-64.

**Illustration of learning convergence:** The MCNs model is based on a binarized process, which is implemented on Torch platform. For a 20-layer MCN with width 16-16-32-64 that is trained after 200 epochs, the training process takes about 3 hours with two 1080ti GPUs. We plot the training and testing accuracy of MCNs and U-MCNs in Fig. 6. The architecture of U-MCNs is the same as that of MCNs. Fig. 6 clearly shows that MCNs (the blue curves) converge in similar speeds to its unbinarized counterpart (the red curves).

**The number of clustering centers:** In Section 2.2.1, we show the binary quantization with 2 clustering centers in Eq. 6. In this experiment, we investigate the effect of more numbers of clustering centers with MCNs on CIFAR-10.

The results are shown in Fig. 7, where we can see that the accuracy increases with the number of clustering centers and the center loss can also be used to improve the performance. However, to save storage space and to compare with other binary networks, we focus on using two clustering centers for MCNs.

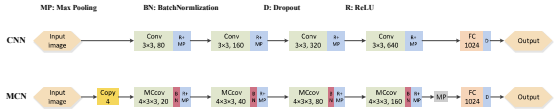



Figure 8: Network architectures of CNNs and MCNs.

### 3.2. Results on MINST, SVHN, and CIFAR-10/100

Table 2 demostrates the image classification results from our experiments on various datasets. MCNs are compared to state-of-the-art methods such as LBCNN [9], BinaryConnect [3], Binarized Neural Networks (BNN) [4], XNOR-Net [15], ResNet-101 [7], Maxout Network [6], and Network in Network (NIN) [12]. For each dataset, the training methods and parameters of the MCNs models are described in the following sections.

**MNIST**: Fig. 8 shows the details of the network architectures of MCNs used on MINIST. Due to the easy task on MNIST, the architectures of our MCNs are based on a simple CNN. The MCNs contain four MCconv layers and one full-connected layer. For this model, we adopt Max-pooling and ReLu after the convolution layers, and a dropout layer [8] after the FC layer to avoid over-fitting. We report the performance of our algorithm on a test set after 200 epochs on the average over 5 predictions. The results are shown in Table 2. It is observed from the experiments that the MCNs achieve 99.52% accuracy on the MINST test, which is better than other binary methods and comparable to other full-precision models.

**SVHN**:The network depth is set to 28 and the stage is set to 64-64-128-256. The total training epochs are 200 and the learning rate is reduced per 30 epochs. The results are list in Table 2. The LBCNN utilized on SVHN has 80 convolutional layers (40 LBCNN modules), 512 LBC filters, 16 output channels, and 512 hidden units in the fully connected layer. Compared to LBCNN, MCNs obtains a better performance with 1.2% improvement. Note that we only use a subset of the whole SVHN to train our MCNs, while other models use the whole set (including the additional images) to do their training.

**CIFAR-10/100**: The models and training parameters of MCNs used on CIFAR-10 and CIFAR-100 datasets are the same. The architecture of MCNs has 34 layers with the basic block(c) in Fig. 4, 64-64-128-256 network stage, and 512 hidden units in the fully connected layer. The accuracy of MCNs on CIFAR-10 and CIFAR-100 reaches to 95.39% and 78.13% respectively. Table 2 shows that MCNs obtain the best performance compared with the other state-of-art binary methods and other CNNs on both CIFAR-10 and CIFAR-100. Compared with ResNet-101, MCNs also achieve better performances, which further validates the effectiveness of our model. Besides, the bracket shows the

Table 2: Classification accuracy (%) on 4 datasets.

| Method | **MCNs(U-MCNs)** | LBCNN [9] | BinaryConnect [3] | BNN [4] | XNOR-Net [15] | ResNet-101 [17] | Maxout [6] | NIN [12] |
|---|---|---|---|---|---|---|---|---|
| MINIST | 99.52 | 99.51 | 98.99 | 98.60 | – | – | 99.55 | 99.53 |
| CIFAR-10 | 95.39(95.75) | 92.99 | 91.73 | 89.85 | 89.83 | 93.57 | 90.65 | 91.19 |
| SVHN | 96.87 | 94.50 | 97.85 | 97.49 | – | – | 97.53 | 97.65 |
| CIFAR-100 | 78.13 | – | – | – | – | 74.84 | 61.43 | 64.32 |

accuracy of corresponding U-MCN. Note that the accuracy of MCNs only decreases a little when the binarized filters are used.

Table 3: Classification accuracy (%) on 100-class ImageNet

| Method | LBCNN | MCNs | WRNs |
|---|---|---|---|
| top-1 | 63.24 | 83.82 | 84.96 |
| top-5 | – | 94.8 | 95.64 |

### 3.3. Results on 100-class ImageNet

To further show the effectiveness of the proposed MCNs method, we evaluate it on the 100-class ImageNet [5] dataset. For the experiment, we train a 34-layer MCN with width 32-64-128-256. The corresponding WRNs are used as baselines. The MCNs is trained after 120 epochs. The learning rate is initialized to 0.1 and decreases to 0.1 times per 30 epochs. Top-1 and Top-5 errors are evaluated and the results are shown in Table 3. The testing error curves are shown in Fig. 9, where we can see that both have similar convergence rates after 30 epochs. Meanwhile, the best result of LBCNN [9] on 100-ImageNet is presented to compare with ours. The LBCNN has 48 convolutional layers (24 LBC modules), 512 LBC filters, 512 output channels, 0.9 sparsity, and 4096 hidden units in the fully connected layer. It is observed that our MCNs gain an advantage over LBCNN by 20.58%. Compared with WRNs, our MCNs have only a little performance drop with the same architecture.

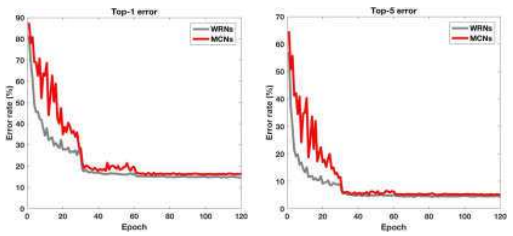



Figure 9: Testing error curves for the 100-class ImageNet experiment.

## 4. Conclusion

We have developed a new deep learning model, modulated convolutional networks (MCNs), which can significantly reduce the storage requirement on computationally limited devices. MCNs are implemented by a set of binary filters and the proposed M-Filters. In MCNs, we use M-Filters to build an end-to-end framework and a new architecture to calculate the network model. Both the binarized filters and M-Filters are obtained in the same pipeline as in the back propagation algorithm. The convolution operation is further approximated via the center loss method. MCNs can reduce the storage by a factor of 32, in contrast to the full-precision models, while achieving a much better performance than state-of-the-art binarized models. Our MCNs based on a highly compressed model also achieve a comparable performance to well-known full-precision Resnets or Wide-Resnets.

As a general convolutional layer, the M-Filters can also be used on other deep models and various tasks, which is our future work.

## Acknowledgment

The work was supported by the Natural Science Foundation of China under Contract 61601466, 61672079 and 61473086. This work was supported by the National Basic Research Program of China (2015CB352501), the Open Projects Program of National Laboratory of Pattern Recognition, and Shenzhen Peacock Plan. Baochang Zhang is also corresponding author. Xiaodi Wang and Baochang Zhang have the equal contribution to the paper.

## References

[1] A. Banerjee and V. Iyer. Cs231n project report—tiny imagenet challenge. http://cs231n.stanford.edu/2015/reports.html, 2015. 6

[2] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the International Conference on Machine Learning*, pages 111–118, 2010. 1

[3] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015. 1, 7, 8

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 1, 7, 8

[5] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6, 8

[6] I. J. Goodfellow, D. Wardefarley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, 2013. 7, 8

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 7

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 6, 7

[9] F. Juefei-Xu, V. N. Boddeti, and M. Savvides. Local binary convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 6, 7, 8

[10] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 6

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 6

[12] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations*, 2014. 7, 8

[13] Y. Lcun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 5, 2011. 6

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 2016. 1, 2, 7, 8

[16] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 6

[17] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016. 6, 8

[18] M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 6

[19] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1