A Deep Convolutional Neural Network Module that Promotes Competition of Multiple-size Filters

1. Introduction

The use of competitive activation units in deep convolutional neural networks (ConvNets) is generally understood as a way of building one network by the combination of multiple sub-networks, with each one being capable of solving a simpler task when compared to the complexity of the original problem involving the whole dataset [1]. Similar ideas have been explored in the past using multi-layer perceptron models [2], but there is a resurgence in the use of competitive activation units in deep ConvNets [3, 1]. For instance, rectified linear unit (ReLU) [4] promotes a competition between the input sum (usually computed from the output of convolutional layers) and a fixed value of 0, while maxout [5] and local winner-take-all (LWTA) [3] explore an explicit competition amongst the input units. As shown by Srivastava et al. [1], these competitive activation units allow the formation of sub-networks that respond consistently to similar input patterns, which facilitates training [4, 5, 3] and generally produces superior classification results [1].

In this paper, we introduce a new module for deep ConvNets composed of several convolutional filters of multiple sizes that are joined by a maxout activation unit, which promotes competition amongst these filters. Our idea has been inspired by the recently proposed inception module [6], which currently produces state-of-the-art results on the ILSVRC 2014 classification and detection challenges [7]. The gist of our proposal is depicted in Fig. 1, where we have the data in the input layer filtered in parallel by a set of convolutional filters of multiple sizes [8, 6, 9]. Then the output of each filter of the convolutional layer passes through a batch normalisation unit (BNU) [10] that weights the importance of each filter size and also pre-conditions the model (note that the pre-conditioning ability of BNUs in ConvNets containing piece-wise linear activation units has been empirically shown in [11]). Finally, the multiple size filter outputs, weighted by BNU, are joined with a maxout unit [5] that reduces the dimensionality of the joint filter outputs and promotes competition amongst the multiple size filters. We empirically show that the competition amongst filters of multiple size prevents filter co-adaptation and allows the formation of multiple sub-networks. Furthermore, we show that the introduction of our proposal module in a typical deep ConvNet produces competitive results in the field for the benchmark datasets MNIST [12], CIFAR-10 [13], CIFAR-100 [13], street view house number (SVHN) [14], and ImageNet ILSVRC 2012 [7].

2. Literature Review

One of the main reasons behind the outstanding performance of deep ConvNets is attributed to the use of competitive activation units in the form of piece-wise linear functions [15, 1], such as ReLU [4], maxout [5] and LWTA [3] (see Fig. 2). In general, these activation functions enable the formation of sub-networks that respond consistently to similar input patterns [1], dividing the input data points (and more generally the training space) into regions [15], where classifiers and regressors can be learned more effectively given that the sub-problems in each of these regions are simpler than the original problem involving the whole training set. In addition, the joint training of the sub-networks present in such deep ConvNets represents a useful regularisation method [4, 5, 3]. In practice, ReLU, maxout and LWTA allows the division of the input space in exponentially many regions as a function of the number of layers and the number of input nodes to each competitive activation unit, so this means that maxout and LWTA can estimate exponentially complex functions more effectively than ReLU because of the larger number of sub-networks that are jointly trained. An important aspect of deep ConvNets with competitive activation units is the fact that the use of batch normalisation units (BNU) helps not only with respect to the convergence rate [10], but also with the pre-conditioning of the model by promoting an even distribution of the input data points, which results in the maximisation of the number of the regions (and respective sub-networks) produced by the piece-wise linear activation functions [11].

Furthermore, training ConvNets with competitive activation units [1, 11] usually involves the use of dropout [16] that consists of a regularisation method that prevents filter co-adaptation [16], which is a particularly important issue in such models, because filter co-adaptation can lead to a severe reduction in the number of the sub-networks that can be formed during training.

Another aspect of the current research on deep ConvNets is the idea of making the network deeper, which has been shown to improve classification results [17]. However, one of the main ideas being studied in the field is how to increase the depth of a ConvNet without necessarily increasing the complexity of the model parameter space [18, 6]. For the Szegedy et al.'s model [6], this is achieved with the use of 1×1 convolutional filters [19] that are placed before each local filter present in the inception module in order to reduce the input dimensionality of the filter. In Simonyan et al.'s approach [18], the idea is to use a large number of layers with convolutional filters of small size (e.g., 3×3). In this work, we restrict the complexity of the deep ConvNet with the use of maxout activation units, which selects only one of the input nodes, as shown in Fig, 2.

Finally, the use of multiple size filters in deep ConvNets is another important implementation that is increasingly being explored by several researchers [8, 6, 9]. Essentially, multiple size filtering follows a neuroscience model [20] that suggests that the input image data should be processed by filters of different sizes (which can lead to filters of different scales) and then pooled together, so that the deeper processing stages can become more robust to scale changes [6]. We explore this idea in our proposal, as depicted in Fig. 1, but we hypothesise (and show supporting evidence) that the multiple size of the filters prevents their co-adaptation during training, leading to better generalisation. We also hypothesise and show evidence that what is driving this better generalisation is the fact that the multiple sizes of the filters promote the learning of features that are more different from each other within competitive units when compared to the single-size filters.

3. Methodology

Assume that an image is represented by $\mathbf{x} : \Omega \rightarrow \mathbb{R}$, where Ω denotes the image lattice, and that an image patch of size $(2k-1) \times (2k-1)$ (for $k \in \{1, 2, ..., K\}$) centred at position $i \in \Omega$ is represented by $\mathbf{x}_{i \pm (k-1)}$. The models being proposed in this paper follow the structure of the NIN model [19], and is in general defined as follows:

$$f(\mathbf{x}, \theta_f) = f_{out} \circ f_L \circ ... \circ f_2 \circ f_1(\mathbf{x}), \tag{1}$$

where $\circ$ denotes the composition operator, θ_f represents all the ConvNet parameters (i.e., weights and biases), $f_{out}(.)$ denotes an averaging pooling unit followed by a softmax activation function [19], and the network has blocks represented by $l \in \{1, ..., L\}$, with each block containing a composition of N_l modules with $f_l(\mathbf{x}) = f_l^{(N_l)} \circ ... \circ f_l^{(2)} \circ f_l^{(1)}(\mathbf{x})$. Each module $f_l^{(n)}(.)$ at a particular position $i \in \Omega$ of the input data for block l is defined by:

$$\begin{aligned} f_l^{(n)}(\mathbf{x}_i) = \sigma\big(&\mathrm{BN}_{\gamma_1,\beta_1}(\mathbf{W}_1^\top \mathbf{x}_i),\ \mathrm{BN}_{\gamma_3,\beta_3}(\mathbf{W}_3^\top \mathbf{x}_{i\pm1}), ..., \\ &\mathrm{BN}_{\gamma_{2k-1},\beta_{2k-1}}(\mathbf{W}_{2k-1}^\top \mathbf{x}_{i\pm(k-1)}), \\ &\mathrm{BN}_{\gamma_p,\beta_p}(\mathbf{W}_1^\top p_{3\times3}(\mathbf{x}_{i\pm1}))\big), \end{aligned} \tag{2}$$

where $\sigma(.)$ represents the maxout activation function [5]$\sigma(x) = \max_{j\in\{1,2,...,J\}} x_j$, the convolutional filters of the module are represented by the weight matrices $\mathbf{W}_{2k-1}$ for $k \in \{1, ..., K_l\}$ (i.e., filters of size $2k-1 \times 2k-1 \times \#filters$, with $\#filters$ denoting the number of 2-D filters present in $\mathbf{W}$), which means that each module n in block l has K_l different filter sizes and $\#filters$ different filters, $\mathrm{BN}_{\gamma,\beta}$ represent the batch normalization transformation with scaling and shifting parameters [10], and $p_{3\times3}(\mathbf{x}_{i\pm1})$ represents a max pooling operator on the 3×3 subset of the input data for layer l centred at $i \in \Omega$, i.e.$\mathbf{x}_{i\pm1}$. The batch normalisation transformation $\mathrm{BN}_{\gamma,\beta}$ [10]

is computed as

$$\begin{aligned}
\mu_{\mathcal{B}} &= \frac{1}{m}\sum_{i=1}^{m} x_i, \\
\sigma_{\mathcal{B}}^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2, \\
\hat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \\
\mathrm{BN}_{\gamma,\beta}(x_i) &= \gamma \hat{x}_i + \beta,
\end{aligned} \tag{3}$$

where $\mathcal{B} = \{x_1, ..., x_m\}$ represents a mini-batch of inputs, γ and β are two learn-able parameters, and ϵ is a small constant.

Using the ConvNet module defined in (2), our proposed models differ mainly in the presence or absence of the node with the max-pooling operator within the module (i.e., the node represented by $\mathrm{BN}_{\gamma_p,\beta_p}(\mathbf{W}_1^\top p_{3\times3}(\mathbf{x}_{i\pm1}))$. When the module does not contain such node, it is called **Competitive Multiple-size Convolution** (see Fig. 3-(a)), but when the module has the max-pooling node, then we call it **Competitive Inception** (see Fig. 3-(b)) because of its similarity to the original inception module [6]. The original inception module is also implemented for comparison purposes (see Fig. 3-(c)), and we call this model the **Inception Style**, which is similar to (1) and (2) but with the following differences: 1) the function $\sigma(.)$ in (2) denotes the concatenation of the input parameters; 2) a 1×1 convolution is applied to the input $\mathbf{x}$ before a second round of convolutions with filter sizes larger than or equal to 3×3; and 3) a ReLU activation function [4] is present after each convolutional layer.

An overview of all models with the structural parameters is displayed in Fig. 3. Note that all models are inspired by NIN [19], GoogLeNet [6], and MIM [11]. In particular, we replace the original 5×5 convolutional layers of MIM by multiple size filters of sizes 1×1, 3×3, 5×5, and 7×7. For the inception style model, we ensure that the number of output units in each module is the same as for the competitive inception and competitive multiple size convolution, and we also use a 3×3 max-pooling path in each module, as used in the original inception module [6]. Another important point is that in general, when designing the inception style network, we follow the suggestion by Szegedy et al. [6] and include a relatively larger number of 3×3 and 5×5 filters in each module, compared to filters of other sizes (e.g., 1×1 and 7×7). An important distinction between the original GoogLeNet [6] and the inception style network in Fig. 3-(c) is the fact that we replace the fully connected layer in the last layer by a single 3×3 convolution node in the last module, followed by an average pooling and a softmax unit, similarly to the NIN model [19]. We propose this modification to limit the number of training parameters (with the removal of the fully connected layer) and to avoid the concatenation of the nodes from different paths (i.e., maxpooling, 1×1 convolution filter, and etc.) into a number of channels that is equal to the number of classes (i.e., each channel is averaged into a single node, which is used by a single softmax unit), where the concatenation would imply that some of the paths would be directly linked to a subset of the classes. Therefore, the last two layers of block 3 of all architectures in Fig. 3 contain an average pooling and a softmax unit to ensure a fair comparison amongst these networks.

3.1. Competitive Convolution of Multiple Size Filters Prevents Undesirable Filter Convergence and Filter Co-adaptation

The main reason being explored in the field to justify the use of competitive activation units [4, 5, 3] is the fact that they build a network formed by multiple underlying sub-networks [1]. More clearly, given that these activation units consist of piece-wise linear functions, it has been shown that the composition of several layers containing such units, divide the input space in a number of regions that is exponentially proportional to the number of network layers [15], where sub-networks will be trained with the samples that fall into one of these regions, and as a result become specialised to the problem in that particular region [1], where overfitting can be avoided because these sub-networks must share their parameters with one another [1]. It is worth noting that these regions can only be formed if the underlying convolutional filters do not converge to similar features, otherwise all input training samples will fall into only one region of the competitive unit, which degenerates into a simple linear transform, preventing the formation of the sub-networks.

A straightforward solution to avoid such undesirable convergence can be achieved by limiting the number of training samples in a mini-batch during stochastic gradient descent. These small batches allow the generation of "noisy" gradient directions during training that can activate different maxout gates, so that the different linear pieces of the activation unit can be fitted, allowing the formation of an exponentially large number of regions. However, the drawback of this approach lies in the determination of the "right" number of samples per mini-batch. A mini-batch size that is too small leads to poor convergence, and if it is too large, then it may not allow the formation of many sub-networks. Recently, Liao and Carneiro [11] propose a solution to this problem based on the use of BNU [10] that distributes the training samples evenly over the regions formed by the competitive unit, allowing the training to use different sets of training points for each region of the competitive unit, resulting in the formation of an exponential number of sub-networks. However, there is still a potential problem with that approach [11], which is that the underlying convolutional filters are trained using feature spaces of the same size (i.e., the underlying filters are of fixed size), which can induce the filters to converge to similar regions of the feature space, also preventing the formation of the sub-networks.

Our proposed module that promotes competition amongst filters of multiple sizes represents a way to prevent the undesirable convergence mentioned above [11]. To evaluate this hypothesis, we examine the similarity between convolutional filters within each competitive unit by empirically showing that the multiple sizes of the convolutional filters within a competitive unit promotes the learning of different features. This demonstration is based on measuring the orthogonality between filters within the same competitive unit, where filters that have similar responses will have orthogonality measures closer to one, and different responses will lead to orthogonality measures close to zero. The orthogonality between two filters, represented by $\mathbf{w}_1$ and $\mathbf{w}_2$, is measured by $\frac{|\mathbf{w}_1^\top \mathbf{w}_2|}{\|\mathbf{w}_1\|_2 \|\mathbf{w}_2\|_2}$, where zero padding along the filter border is used in order to allow the measure of orthogonality between filters of different sizes. We show the mean value of the orthogonality measures between pairs of single-size and multiple-size filters in a competitive unit in Table 1, where results are gathered from the first competitive layer of the first two blocks of the models trained on CIFAR-10 (see Sec. 4.1 for competitive single-size network architecture). The results from Table 1 provide evidence that the filter responses from the competitive multiple-size modules are more orthogonal to each other than the responses from the competitive single-size module.

Multiple-size (Block 1)				
	1x1	3x3	5x5	7x7
1x1	1	0.12	0.15	0.13
3x3	0.12	1	0.21	0.12
5x5	0.15	0.21	1	0.12
7x7	0.13	0.12	0.11	1

Multiple-size (Block 2)				
	1x1	3x3	5x5	7x7
1x1	1	0.05	0.03	0.02
3x3	0.05	1	0.05	0.04
5x5	0.03	0.05	1	0.07
7x7	0.02	0.04	0.07	1

Single-size (Block 1)				
	7x7	7x7	7x7	7x7
7x7	1	0.22	0.22	0.21
7x7	0.22	1	0.19	0.20
7x7	0.22	0.19	1	0.23
7x7	0.21	0.20	0.23	1

Single-size (Block 2)				
	7x7	7x7	7x7	7x7
7x7	1	0.09	0.10	0.09
7x7	0.09	1	0.09	0.09
7x7	0.10	0.09	1	0.09
7x7	0.09	0.09	0.09	1

Table 1: Mean orthogonality measure between pairs of filters within competitive multiple-size convolution modules (up), and filters from competitive single-size convolution modules (down), gathered from the first competitive layer of the first two blocks of the models trained on CIFAR-10.

Our second hypothesis is that the competition amongst multiple size filters within a convolution module prevents co-adaptation throughout the ConvNet model. The evidence for such hypothesis is displayed in Fig 4, which shows Yosinski et al.'s test [22] that assesses the transferability of ConvNet filters. The idea of this experiment is that the ConvNet with more co-adapted filters will produce larger testing errors when the higher level layers are removed because it is unlikely that the re-trained layers will be able to discover the complex co-adaptations present in the removed layers. To conduct this experiment, we train a model using one dataset (CIFAR-100), remove a subset of the higher level convolutional layers (e.g., from layers 7 to the last layer, where the indexing used in the horizontal axis of Fig. 3 is consistent with the first two competitive/inception layers to remove), and re-train only these layers for the CIFAR-10 dataset, while keeping the remaining lower level layers fixed (i.e., we do not re-train these lower layers). Fig. 4 shows that the competitive modules with filters of multiple sizes have the lowest testing error in all evaluated configurations, when compared to the single size competitive module and the competitive inception module. This result supports our hypothesis that competition amongst filters of multiple sizes in a ConvNet module represents a way to prevent co-adaptation.

4. Experiments

We quantitatively measure the performance of our proposed models **Competitive Multiple-size Convolution** and **Competitive Inception** on five computer vision benchmark datasets: CIFAR-10[13], CIFAR-100[13], MNIST [12], SVHN [14], and ImageNet ILSVRC 2012 [7]. We first describe the experimental setup, then using CIFAR-10 and MNIST, we show a quantitative analysis (in terms of classification error, number of model parameters and train/test time) of the two proposed models, the Inception Style model presented in Sec. 3, and two additional versions of the proposed models that justify the use of multiple-size filters, explained in Sec. 3.1. Finally, we compare the performance of the proposed Competitive Multiple-size Convolution and Competitive Inception with respect to the current state-of-the-art results in the four benchmark datasets mentioned above.

The CIFAR-10 [13] dataset contains 60000 images of 10 commonly seen object categories (e.g., animals, vehicles, etc.), where 50000 images are used for training and the rest 10000 for testing, and all 10 categories have equal volume of training and test images. The images of CIFAR-10 consist of 32×32-pixel RGB images, where the objects are well-centered in the middle of the image. The CIFAR-100 [13] dataset extends CIFAR-10 by increasing the number of categories to 100, whereas the total number of images remains the same, so the CIFAR-100 dataset is considered as a harder classification problem than CIFAR-10 since it contains 10 times less images per class and 10 times more categories. The well-known MNIST [12] dataset contains 28×28 grayscale images comprising 10 handwritten digits (from 0 to 9), where the dataset is divided into 60000 images for training and 10000 for testing, but note that the number of images per digit is not uniformly distributed. The Street View House Number (SVHN) [14] is also a digit classification benchmark dataset that contains 600000 32×32 RGB images of printed digits (from 0 to 9) cropped from pictures of house number plates. The cropped images is centered in the digit of interest, but nearby digits and other distractors are kept in the image. SVHN has three sets: training, testing sets and a extra set with 530000 images that are less difficult and can be used for helping with the training process. We do not consider data augmentation on MNIST and SVHN dataset but we show results with and without data augmentation (we only use simple data augmentation, i.e., image translation and mirroring [23, 24]) on CIFAR-10/100. Finally, the ImageNet ILSVRC 2012 [7] dataset contains more than 1 million training images and 50000 validation images of 1000 classes. The performance is evaluated by computing the top1 and top5 (i.e., one of the top 5 predictions is the ground truth label) accuracies on the validation set.

In all these benchmark datasets we minimize the softmax loss function present in the last layer of each model for the respective classification in each dataset, and we report the results as the proportion of misclassified test images. We use an initial learning rate of 0.1 and follow a multiple-step decay to a final learning rate of 0.001 (in 80 epochs for CIFAR-10 and CIFAR-100, 50 epochs for MNIST, and 40 epochs for SVHN). For ImageNet dataset, we decay the final learning rate to 1e-5 in 90 epochs. The stopping criterion is determined by the convergence observed in the error on the validation set. The mini-batch size for CIFAR-10, CIFAR-100, and MNIST datasets is 100, and 128 for SVHN and ImageNet dataset. The momentum and weight decay are set to standard values 0.9 and 0.0005, respectively. For each result reported (except the ImageNet results), we compute the mean and standard deviation of the test error from five separately trained models, where for each model, we use the same training set and parameters (e.g., the learning rate sequence, momentum, etc.), and we change only the random initialization of the filter weights and randomly shuffle the training samples. We also show the best achieved error of the five models as in [25, 23].

We use the GPU-accelerated ConvNet library MatConvNet [26] and Torch-7 [27] based fb.resnet.torch package [28] to perform the experiments specified in this paper. Our experimental environment is a desktop PC equipped with i7-4770 CPU, 24G memory and a 12G GTX TITAN X graphic card. Using this machine, we report the mean training and testing times of our models.

4.1. Model Design Choices

In this section, we show the results from several experiments that show the design choices for our models, where we provide comparisons in terms of their test errors, the number of parameters involved in the training process and the training and testing times. Table 2 shows the results on CIFAR-10 and MNIST for the models **Competitive Multiple-size Convolution**, **Competitive Inception**, and **Inception Style** models, in addition to other models explained below. Note that all models in Table 2 are constrained to have the same numbers of input channels and output channels in each module, and all networks contain three blocks [19], each with three modules (so there is a total of nine modules in each network), as shown in Fig. 3. These models are trained without data augmentation.

We argue that the multiple-size nature of the filters within the competitive module is important to avoid the co-adaptation issue explained in Sec. 3.1. We assess this importance by comparing both the number of parameters and the test error results between the proposed models and the model **Competitive Single-size Convolution**, which has basically the same architecture as the Competitive Multiple-size Convolution model represented in Fig. 3-(a), but with the following changes: the first two blocks contain four sets of 7×7 filters in the first module, and in the second and third modules, two sets of 3×3 filters; and the third block has three filters of size 5×5 in the first module, followed by two modules with two 3×3 filters. Notice that this configuration implies that we replace the multiple-size filters by the filter of the largest size of the module in each node, which is a configuration similar to the recently proposed MIM model [11]. The configuration for the Competitive Single-size Convolution has around two times more parameters than the Competitive Multiple-size Convolution model and takes longer to train, as displayed in Table 2. The idea behind the use of the largest size filters within each module is based on the results obtained from the training of the batch normalisation units of the Competitive Multiple-size Convolution modules, which indicates that the highest weights (represented by γ in (2)) are placed in the largest size filters within each module, as shown in Fig. 5. The classification results of the Competitive Single-size Convolution, shown in Table 2, demonstrate that it is consistently inferior to the Competitive Multiple-size Convolution model.

Another important point that we test in this section is the relevance of dropping connections in a deterministic or stochastic manner when training the competitive convolution modules. In particular, we are interested if the deterministic masking provided by our proposed Competitive Multiple-size Convolution module is more effective at avoiding undesirable filter convergence and filter co-adaptation than the stochastic masking provided by DropConnect [29]. We run a quantitative analysis of the **Competitive DropConnect Single-size Convolution**, where we take the Competitive Single-size Convolution proposed before and randomly drop connections using a rate, which is computed such that it has on average the same number of parameters to learn in each round of training as the Competitive Multiple-size Convolution, but notice that the Competitive DropConnect Single-size Convolution has in fact the same number of parameters as the Competitive Single-size Convolution. Using Fig. 6, we see that the DropConnect rate is 0.57 for the module 1 of blocks 1 and 2 specified in Fig. 3. The results in Table 2 show that it has around two times more parameters, takes longer to train and performs significantly worse than the Competitive Multiple-size Convolution model.

We also notice that the competitive multiple-size module has better generalisation ability amongst the evaluated models, which is shown in Fig. 7, where the competitive multiple-size module has the highest training error in CIFAR-10 and CIFAR-100 (amongst competitive models), but the smallest testing error. Finally, the reported training and testing times in Table 2 show a clear relation between the number of model parameters and those times.

4.2. Comparison with the State of the Art

We show the performances of the proposed Competitive Multiple-size Convolution (CMSC) and Competitive Inception Convolution models on CIFAR-10, CIFAR-100, MNIST and SVHN, and compare them with the current state of the art in the field. To further explore the usefulness of our module in deeper learning models, we introduce the **CMSC-V2** model which adds one more CMSC layer with filter size up to 5×5 in between the two sequential 3×3 CMSC layers in each block of the original CMSC design (see Fig. 3-a), resulting in a 12-layer model with 7 million parameters.

We now give a brief introduction to the state of the art methods. **Stochastic Pooling** [30] proposes a regularization based on a replacement of the deterministic pooling (e.g., max or average pooling) by a stochastic procedure, which randomly selects the activation within each pooling region according to a multinomial distribution, es-

CIFAR-10

Method	No. of Params	Test Error best (mean $\pm$ std dev)	Train Time (h)	Test Time (ms)
Competitive Multiple-size Convolution	4.48 M	6.80 $(6.87 \pm 0.05)\%$	6.4 h	2.7 ms
Competitive Inception	4.69 M	6.67 $(7.13 \pm 0.31)\%$	7.6 h	3.1 ms
Inception Style	0.61 M	8.42 $(8.50 \pm 0.06)\%$	3.9 h	1.5 ms
Competitive Single-size Convolution	9.35 M	6.98 $(7.15 \pm 0.12)\%$	8.0 h	3.2 ms
Competitive DropConnect Single-size Convolution	9.35 M	8.88 $(9.12 \pm 0.17)\%$	7.7 h	3.1 ms

MNIST

Method	No. of Params	Test Error best (mean $\pm$ std dev)	Train Time (h)	Test Time (ms)
Competitive Multiple-size Convolution	1.13 M	0.29 $(0.33 \pm 0.04)\%$	1.5 h	0.8 ms
Competitive Inception	1.19 M	0.38 $(0.40 \pm 0.02)\%$	1.9 h	1.0 ms
Inception Style	0.18 M	0.43 $(0.44 \pm 0.01)\%$	1.4 h	0.7 ms
Competitive Single-size Convolution	2.39 M	0.33 $(0.37 \pm 0.03)\%$	1.7 h	0.9 ms
Competitive DropConnect Single-size Convolution	2.39 M	0.32 $(0.35 \pm 0.03)\%$	1.6 h	0.9 ms

Table 2: Results on CIFAR-10 and MNIST of the proposed models, in addition to the Competitive Single-size Convolution and Competitive DropConnect Single-size Convolution that test our research questions posed in Sec. 3.1.

timated from the activation of the pooling unit. **Spectral Pooling** [31] proposes to perform pooling by truncating the representation in the frequency domain, which preserves considerably more information than other pooling methods and enables flexible choices of pooling output dimensionality. **BinaryConnect** [32] introduces a method to train a DNN with binary weights during the forward and backward propagations, and the Binary Connect method acts as regularizer. **Maxout Networks** [5] introduces a piece-wise linear activation unit that is used together with dropout training [16] and is introduced in Fig. 2-(c). The **Network in Network** (NIN) [19] model consists of the introduction of multilayer perceptrons as activation functions to be placed between convolution layers, and the replacement of a final fully connected layer by average pooling, where the number of output channels represent the final number of classes in the classification problem. **Deeply-supervised nets** [33] introduce explicit training objectives to all hidden layers, in addition to the back-propagated errors from the last softmax layer. The use of a recurrent structure that replaces the purely feed-forward structure in ConvNets is explored by the model **RCNN** [34]. An extension of the NIN model based on the use of maxout activation function instead of the multilayer perceptron is introduced in the **MIM** model [11], which also shows that the use of batch normalization units are crucial for allowing an effective training of several single-size filters that are joined by maxout units. Clevert et al. [35] introduces the exponential linear unit (**ELU**) which alleviates the vanishing gradient problem and speeds up the learning, leading to better classification performance. **FitNet** [36] learns a student network from the softmax output of a large teacher network or ensemble networks. Residual network (**ResNet**) [23] implements a residual connection to bypass learning blocks, showing competitive results on both CIFAR [13] datasets and Imagenet [7]. **Stochastic depth** [37] is a method that dynamically drops learning blocks of the ResNet in order to improve the generalisation ability of the ResNet model. **WideResNet** [38] model aims to reduce the depth of the ResNet while increasing the capacity of each residual block. Finally, the **FractalNet** [24] shows a network module with interactive sub-paths, which can be used as an alternative to the residual module to compose a very deep network.

The comparison on CIFAR-10 and CIFAR-100 [13] datasets are shown in Table 3, where we separate the results by whether they are obtained with or without data aug-

Method	CIFAR-10			CIFAR-100	
	No. of Params	noAug	Aug	noAug	Aug
CMSC	4.48 M	6.80 (6.87 $\pm$ 0.05)%	6.36 (6.52 $\pm$ 0.18)%	26.87 (27.56 $\pm$ 0.49)%	25.52 (25.89 $\pm$ 0.30)%
CMSC-V2	7.01 M	6.73 (6.80 $\pm$ 0.12)%	5.39 (5.61 $\pm$ 0.16)%	29.24 (29.63 $\pm$ 0.44)%	24.43 (25.37 $\pm$ 0.54)%
Competitive Inception	4.69 M	6.67 (7.13 $\pm$ 0.31)%	5.98 (6.18 $\pm$ 0.21)%	27.73 (28.17 $\pm$ 0.25)%	26.30 (26.61 $\pm$ 0.38)%
FractalNet-20layers [24]	38.6 M	7.27%	4.68%	29.05%	24.32%
ELU [35]	-	-	6.55%	-	24.28%
FitNet [36]	2.50 M	-	8.39%	-	35.04%
MIM [11]	1.93 M	8.52 $\pm$ 0.20%	-	29.20 $\pm$ 0.20%	-
Spectral pooling [31]	-	8.60%	-	31.60%	-
RCNN [34]	-	8.69%	7.09%	31.75%	-
Deeply-supervised nets [33]	-	9.69%	7.97%	34.57%	-
Network in Network [19]	0.97 M	10.41%	-	35.68%	-
Maxout Networks [5]	-	11.68%		38.57%	-
ResNets:					
WideResNet [38]	36.5 M	-	4.17%	-	20.50%
ResNet (Stochastic depth) [37]	1.70 M	11.66%	5.25%	37.80%	24.98%
ResNet [23, 37]	1.70 M	13.63%	6.41%	44.74%	27.76%

Table 3: Comparison in terms of classification error between our proposed models (highlighted) and the state-of-the-art methods on CIFAR-10 and CIFAR-100 [13].

Method	Test Error
CMSC	0.29 (0.33 $\pm$ 0.04)%
RCNN [34]	0.31%
MIM [11]	0.35 $\pm$ 0.03%
Deeply-supervised nets [33]	0.39%
Competitive Inception	0.38 (0.40 $\pm$ 0.02)%
Network in Network [19]	0.45%
Conv. Maxout+Dropout [5]	0.47%
Stochastic Pooling [30]	0.47%

Table 4: Comparison in terms of classification error between our proposed models (highlighted) and the state-of-the-art methods on MNIST [12].

mentation. Our proposed methods has the best result among the state-of-the-art methods without data augmentation and show competitive results to the state-of-the-art methods that using data augmentation. Table 4 shows the results on MNIST [12], where it is worth reporting that the best result (over the five trained models) produced by our CMSC model is a test error of 0.29%, which is better than the single result from Liang and Hu [34]. Finally, the comparison on SVHN[14] dataset is shown in Table 5, where our CMSC model is able to achieve the best error of 1.69%.

4.3. ImageNet Evaluation

ImageNet [7] is a large-scale image dataset that is use to benchmark several methods proposed in the field. Naively, our Competitive Multiple-size module includes the use of large size filters, which produce a large number of parameters that do not scale well with the input dimensionality (i.e., the number of channels of input tensor). This

Method	Test Error
ResNet (Stochastic depth) [37]	1.75%
CMSC	1.69 (1.76 $\pm$ 0.07)%
RCNN [34]	1.77%
Competitive Inception	1.70 (1.78 $\pm$ 0.09)%
ResNet [23, 37]	1.80%
FractalNet-20layers [24]	1.87%
Deeply-supervised nets [33]	1.92%
Drop-connect [29]	1.94%
MIM [11]	1.97 $\pm$ 0.08%
BinaryConnect [32]	2.15%
Network in Network [19]	2.35%
Conv. Maxout+Dropout [5]	2.47%
Stochastic Pooling [30]	2.80%

Table 5: Comparison in terms of classification error between our proposed models (highlighted) and the state-of-the-art methods on SVHN [14].

means that implementing a similar ConvNet structure to GoogLeNet [6] in terms of the number of layers and number of units per layer would require more than 70M parameters, as opposed to 7M parameters needed by GoogLeNet. To resolve this issue, we adopt the "bottleneck" design from the ResNet [23] model to reduce the dimensionality of the input tensor to keep Competitive Multiple-size Convolution (CMSC) unit computation manageable. Our proposal is to replace the second 3×3 ReLU activated convolution layer [4] with our CMSC module inside the bottleneck module (See Fig. 8) to compose the bottleneck-CMSC module. We replace all the bottleneck units in the original ResNet model with the bottleneck-CMSC units and name it CMSC-ResNet-V1 model. Since a single CMSC module contains more parameters than a 3×3 convolution module, we use $r < 1$ (uniformly used in all modules) to indicate the reduction rate of the number of channels in the input tensor of the CMSC unit (shown at bottom of Fig. 8).

In Table 6 we show that our CMSC-ResNet-V1-5x5 model outperforms the ResNet in a well-controlled experiment setup (i.e., same number of epochs, learning rate schedule, batch size, etc.). Note that the performance of the ResNet model surpasses the performance of GoogLeNet, so that is why we compare our method with ResNet. In the CMSC-ResNet-V1 models, the $k \times k$ suffix means the maximum filter size (i.e., -5 $\times$ 5 means the CMSC module uses 1×1, 3×3, and 5×5 filters) used in each

Method	No. of Params	Top1	Top5
CMSC-ResNet-V1-5x5 (r=0.6)	23.48 M	26.22%	8.06%
CMSC-ResNet-V1-7x7 (r=0.4)	22.59 M	26.76%	8.46%
CMSC-ResNet-V2-5x5 (r=0.9)	22.23 M	25.50%	7.66%
CMSC-ResNet-V2-7x7 (r=0.9)	22.71 M	25.48%	7.59%
Baseline ResNet (r=1.0)[1]	22.78 M	26.63%	8.34%

Table 6: Comparison between the CMSC-ResNet and the baseline ResNet on ILSVRC 2012 validation set, showing the top1 and top5 validation errors by evaluating only the center crop (224x224) of the validation images. All models have 50 layers.

bottleneck-CMSC module. The CMSC-ResNet-V1-5x5 model shows better result than the baseline ResNet model, which indicates the advantage of using the Competitive Multiple-size module in very deep networks. However, we also notice that the CMSC-ResNet-V1-7x7 does not improve the baseline performance of ResNet model, where our hypothesis is that this issue is due to the 60% reduction of activating units in that model. To verify this hypothesis, we increase the number of units (while keeping a similar number of parameters) and introduce the CMSC-ResNet-V2 models, where the large size bottleneck-CMSC units are only used at the bottom of the model and small bottleneck-CMSC units (CMSC with 1×1 and 3×3 filters) at the top of the model, resulting in further improvements. We show that the best model (CMSC-ResNet-V2-7x7) outperforms the baseline ResNet model by 1.15%.

5. Discussion

In terms of the model design choices in Sec. 4.1, we can see that the proposed Competitive Multiple-size Convolution produces more accurate classification results than the proposed Competitive Inception. Given that the main difference between these two models is the presence of the max-pooling path within each module, we can conclude that this path does not help with the classification accuracy of the model. The better performance of both models with respect to the Inception Style model can be attributed to the maxout unit that induces competition amongst the underlying filters, which helps

[1]Our ResNet baseline model is trained (from scratch) from the default script provided in [28] with a few modifications to suit our computational environment: 1) we use mini-batches of size 128; and 2) we downsize the images to 256x256 and store them in an LMDB database, which speeds up the training but at the cost of reading the same training image sequence in each epoch. In comparison, the official script [28] uses mini-batches of size 256, full size images and randomised image reads as default.

more the classification results when compared with the collaborative nature of the Inception module. Also, a comparison with the Competitive Single-size Convolution in Tab. 2 replicates the results from Sec. 3.1, which shows that the proposed Competitive Multiple-size Convolution produces more accurate classification results on the test set. Considering model complexity, it is important to notice that the relation between the number of parameters and training and testing times is not linear, where even though the Inception Style model has $10\times$ fewer parameters, it trains and tests 2 to $1.5\times$ faster than the proposed Competitive Multiple-size Convolution and Competitive Inception models.

The use of deterministic, as opposed to stochastic, mapping also appears to be more effective in avoiding filter co-adaptation given the more accurate classification results of the former mapping. Nevertheless, the reason behind the worse performance of the stochastic mapping may be due to the fact that DropConnect has been designed for the fully connected layers only [29], while our test bed for the comparison is set in the convolutional filters. To be more specific, we think that a fully connected layer usually encapsulates hundreds to thousands of weights for inputs of similar size of dimensions, thus a random dropping on a subset of weight elements can hardly change the distribution of the outputs pattern. However, the convolution filters are of small dimensions, and each of our maxout unit controls 4 to 5 filters at most, so such masking scheme over small weights matrix could result in "catastrophic forgetting" [39] which explains why the Competitive DropConnect Single-size Convolution performs even worse than Competitive Single-size Convolution on CIFAR-10.

The use of 7×7 filters in the competitive module allows us to capture large size spatial patterns in each computation layer. It may be argued that such large size spatial patterns may not be necessary to obtain good classification performance. To explore this point, we test the exclusion of the 7×7 filters from the proposed Competitive Multi-size Convolution model (see Fig. 3-a), resulting in a model of 3.68 M parameters, where the classification result is $7.21 \pm 0.11\%$, which is worse than the one with the 7×7 filters ($6.87\pm0.05\%$). We further show an experiment that assesses whether filters of larger size within a competitive module can improve the classification accuracy at the expense of having a larger number of parameters to train. We test the inclusion of two more filters of sizes 9×9 and 11×11 in module 1 of blocks 1 and 2, and two more filter sizes 7×7 and 9×9 in module 1 of block 3 (see Fig. 3-a). The classification result obtained is $7.36 \pm 0.16\%$ on CIFAR-10, and number of model parameters is 13.11 M. These experiments show that decreasing and increasing the number of filters of larger sizes can have different effects in the classification results, where we may conclude that some spatial pattern at 7×7 size is helpful to classify the CIFAR-10 classes. On the other hand, the CMSC-ResNet-V2-7x7 model shows slightly better performance to CMSC-ResNet-V2-5x5 model on ImageNet, which suggests that the 5×5 spatial patterns may be as expressive as the 7×7 patterns for ImageNet. To conclude, the filter sizes to be included in the competitive unit is a task dependent design choice, where the large size filters can be omitted if the performance gain does not justify the increase in computational costs.

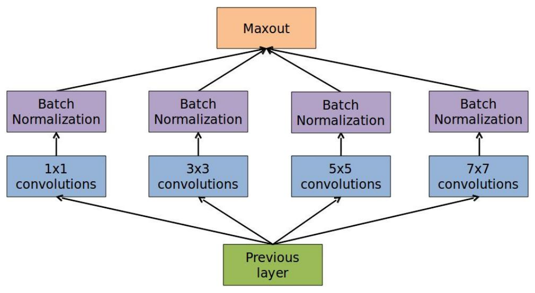
An important modification that can be suggested for our proposed Competitive Multiple-size Convolution model is the replacement of the maxout by ReLU activation, where only the largest size filter of each module is kept and all other filters are removed. One can argue that such model is perhaps less complex (in terms of the number of parameters) and probably as accurate as the proposed model. However, the results we obtained with such model on CIFAR-10 show that this model has 3.28 M parameters (i.e., just slightly less complex than the proposed models, as shown in Table 2) and has a classification test error of $8.16 \pm 0.15\%$, which is significantly larger than for our proposed models. On MNIST, this model has 0.81 M parameters and produces a classification error of $0.37 \pm 0.05\%$, which also shows no advantage over the proposed models.

6. Conclusion

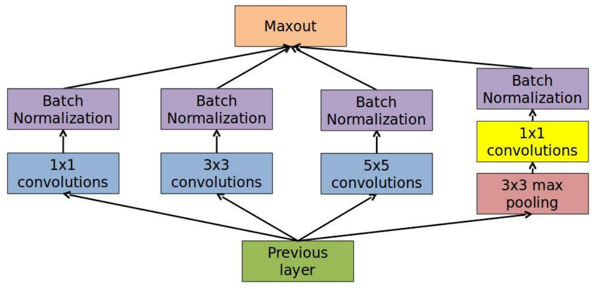
In this paper, we show the effectiveness of using competitive units on modules that contain multiple-size filters. We argue that the main reason of the superior classification results of our proposal, compared with the current state of the art in several benchmark datasets, lies in the following points: 1) the deterministic masking implicitly used by the multiple-size filters avoids the issue of filter co-adaptation; 2) the competitive unit that joins the underlying filters and the batch normalization units promote the formation of a large number of sub-networks that are specialized in the classification problem restricted to a small area of the input space and that are regularized by the fact that they are trained together within the same model; and 3) the maxout unit allows the reduction of the number of parameters in the model. It is important to note that such modules can be applied in several types of deep learning networks, and we plan to apply it to other types of models, such as the recurrent neural network [34].

Acknowledgement

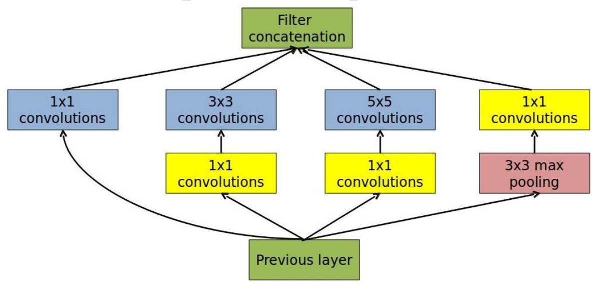
This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).



(a) Competitive multiple-size convolution module



(b) Competitive Inception module



(c) Original inception module

Figure 1: The proposed deep ConvNet modules are depicted in (a) and (b), where (a) only contains multiple size convolutional filters within each module, while (b) contains the max-pooling path, which resembles the original inception module [6] depicted in (c) for comparison.

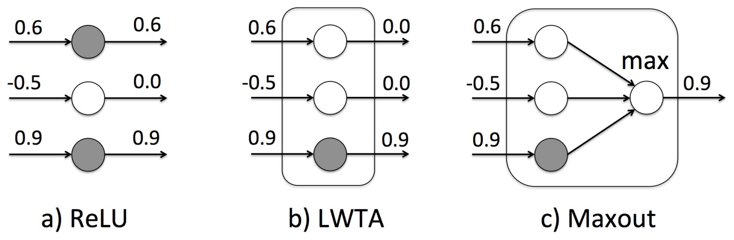
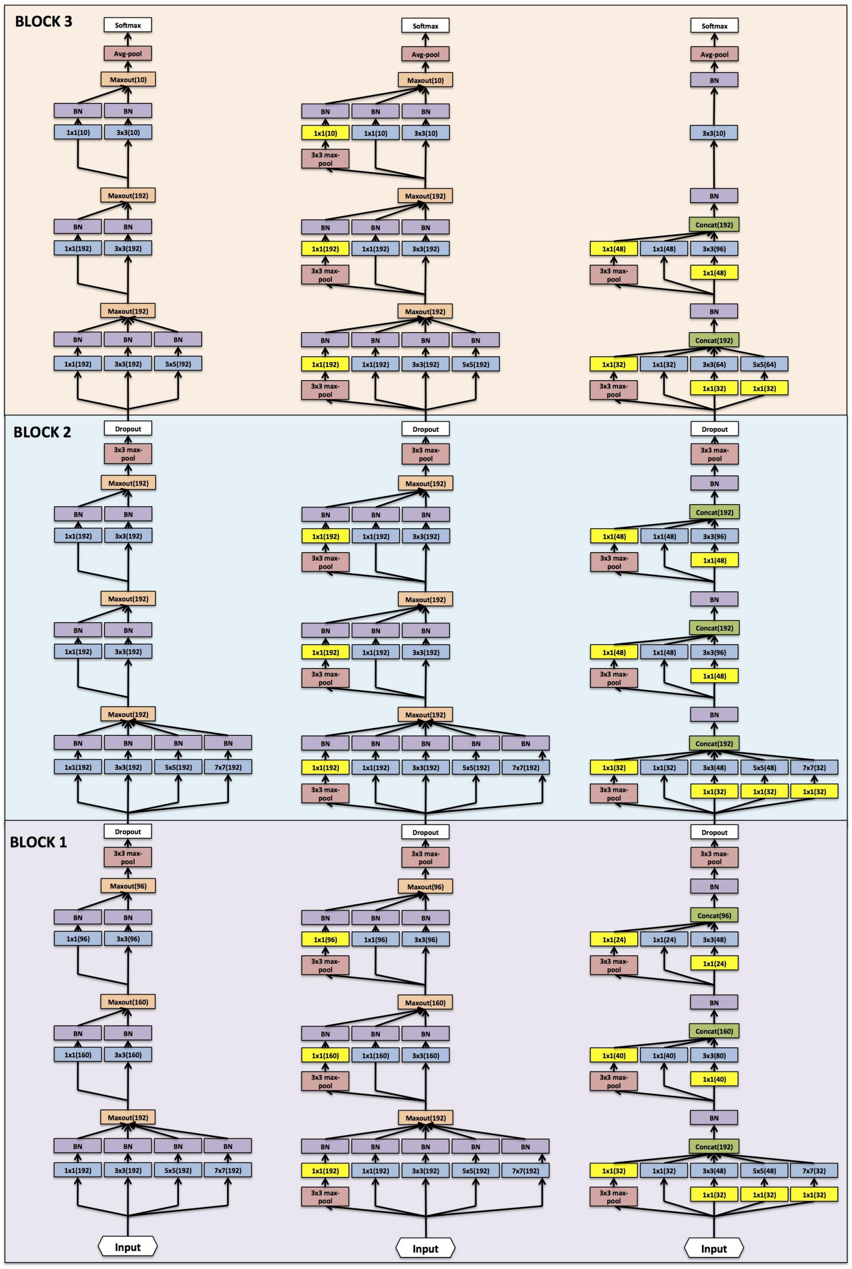


Figure 2: Competitive activation units, where the grey nodes are the active ones, from which errors flow during backpropagation. ReLU [4] (a) is active when the input is bigger than 0, LWTA [3] (b) activates only the node that has the maximum value (setting to zero the other ones), and maxout [5] (c) has only one output containing the maximum value from the input. This figure was adapted from Fig.1 of [1].



(a) Competitive Multiple-size Convolution (b) Competitive Inception (c) Inception style

Figure 3: The proposed competitive multiple-size convolution (a) and competitive inception (b) networks, together with the reference inception style network (c). In these three models, we ensure that the output of each layer has the same number of units. Also note that the inception style model uses ReLU [21] after all convolutional layers, the number of filters per convolutional node is represented by the number in brackets, and these models assume a 10-class classification problem.

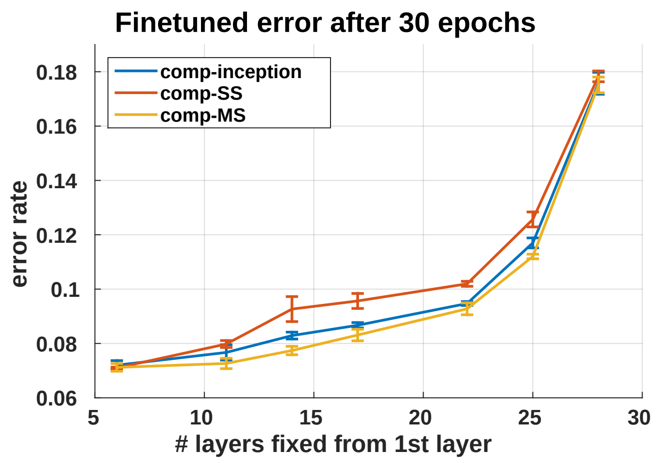


Figure 4: Co-adaptation experiment that tests the transferability of the filters learned for one dataset (CIFAR-10) to another (CIFAR-100) [22] for Competitive inception (**comp-inception**), Single-size (**comp-SS**), and Multiple-size (**comp-MS**) networks. The vertical axis shows the testing error and the horizontal axis represents the index of the first layer of the ConvNet to be re-trained for the new dataset, where the layers below are fixed during the fine-tuning process that lasts 30 epochs.

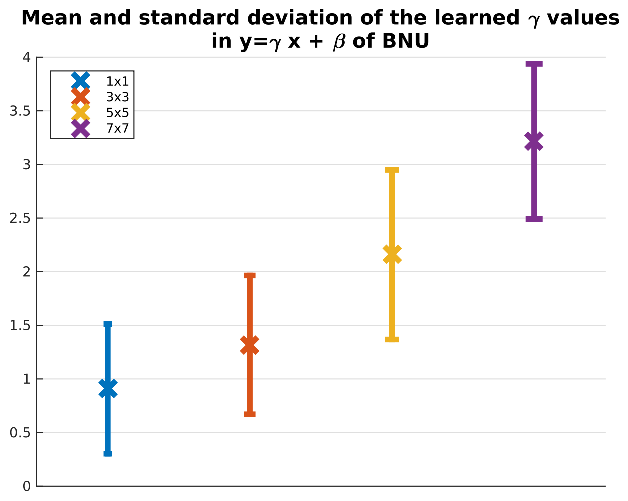
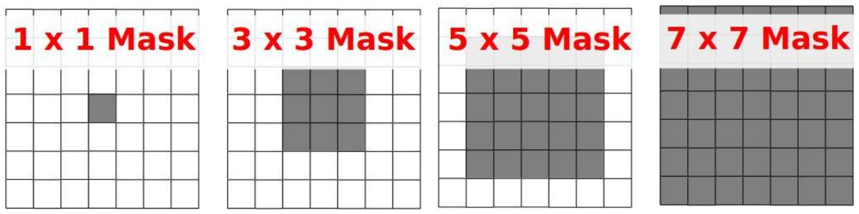
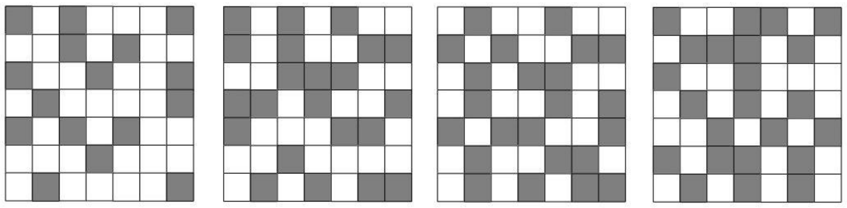


Figure 5: Mean and standard deviation of the learned γ values in the batch normalisation unit of (2) for the Competitive Multiple-size Convolution model on CIFAR-10. This result provides an estimate of the importance placed on each filter by the training procedure.

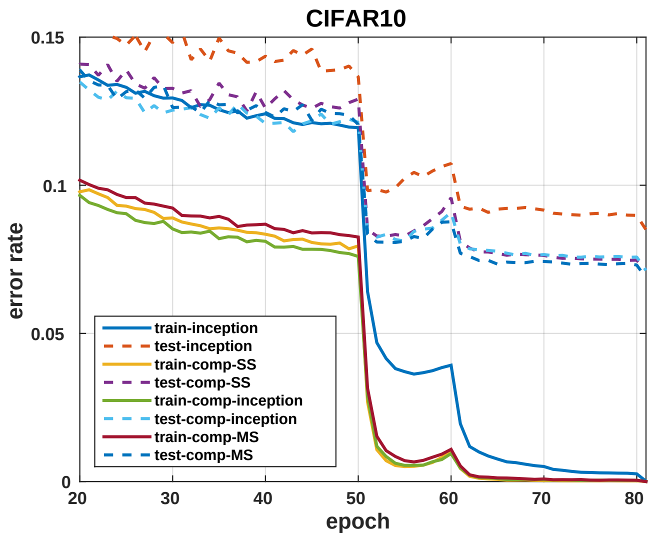


(a) Competitive Multiple-size Convolution Filter Masks



(b) Competitive Single-size DropConnect Convolution Filter Masks

Figure 6: The Competitive Multiple-size Convolution module has filters of size 1×1, 3×3, 5×5, and 7×7, which is equivalent to having four 7×7 filters (with a total of 196 weights) with the masks in (a), where the number of deterministically masked out (or dropped) weights is 112. Using a DropConnect rate of $112/196 \approx 0.57$, a possible set of randomly dropped weights is shown in (b). Note that even though the proportion and number of weights dropped in (a) and (b) are the same, the deterministic or stochastic masking of the weights make a difference in the performance, as explained in the paper.



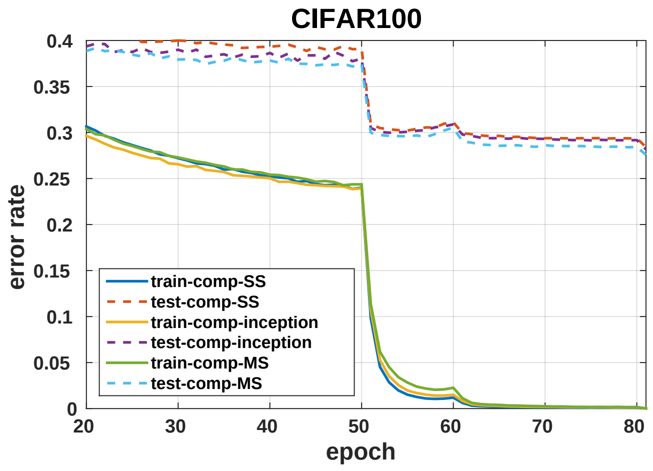


Figure 7: Training and testing errors for CIFAR-10 (up) and CIFAR-100 (down) of the following models: Inception Style (**inception**) , Competitive Single-size Convolution (**comp-SS**), Competitive Inception (**comp-inception**), and Competitive Multiple-size Convolution (**comp-MS**).

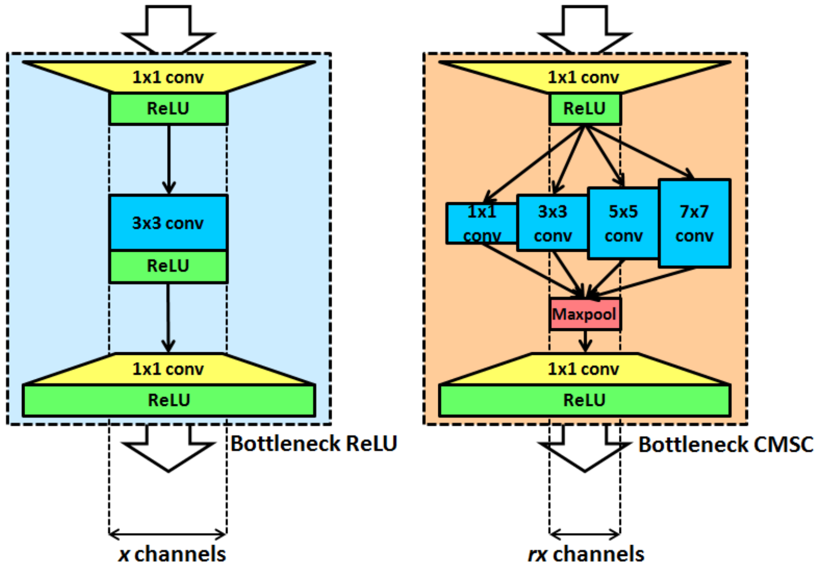


Figure 8: The comparison of the original "bottleneck" ReLU architecture (left) used in ResNet [23] model and our proposed "bottleneck" Competitive Multiple-size Convolution (CMSC) architecture. The width of each computation block illustrates the number of channels of each block, where the first computation block is a 1×1 convolution that is used to reduce the number of input channels for the second computation block. The hyper-parameter r(where $r < 1$) is used to indicate the reduction rate of the number of input channels (as well as the number of CMSC units), compared to the number of input channels/units in the baseline ResNet model. The third computation block is also a 1×1 convolution which restore the number of channels/units.

References

[1] R. K. Srivastava, J. Masci, F. Gomez, J. Schmidhuber, Understanding locally competitive networks, in: International Conference on Learning Representations (ICLR), 2015, pp. 1–11.

[2] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, in: Neural computation, Vol. 3, MIT Press, 1991, pp. 79–87.

[3] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, J. Schmidhuber, Compete to compute, in: Advances in Neural Information Processing Systems (NIPS), 2013, pp. 2310–2318.

[4] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Vol. 15, 2011, pp. 315–323.

[5] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, Y. Bengio, Maxout networks., in: Proceedings of the International Conference on Machine Learning (ICML), Vol. 28, 2013, pp. 1319–1327.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, in: International Journal of Computer Vision, Springer, 2014, pp. 1–42.

[8] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2014, pp. 392–407.

[9] S. Zagoruyko, N. Komodakis, Learning to compare image patches via convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4353–4361.

[10] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the International Conference on Machine Learning (ICML), Vol. 37, 2015, pp. 448–456.

[11] Z. Liao, G. Carneiro, On the importance of normalisation layers in deep learning with piecewise linear activation units, in: IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2016, pp. 1–8.

[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[13] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep. (2009).

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS workshop on Deep Learning and Unsupervised Feature Learning, 2011, p. 5.

[15] G. F. Montufar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 2924–2932.

[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research (JMLR) 15 (1) (2014) 1929–1958.

[17] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, V. Shet, Multi-digit number recognition from street view imagery using deep convolutional neural networks, in: International Conference on Learning Representations (ICLR), 2014, pp. 1–13.

[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICLR), 2015, pp. 1–14.

[19] M. Lin, Q. Chen, S. Yan, Network in network, in: International Conference on Learning Representations (ICLR), 2014, pp. 1–10.

[20] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3).

[21] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of International Conference on Machine Learning, 2010, pp. 807–814.

[22] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 3320–3328.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[24] G. Larsson, M. Maire, G. Shakhnarovich, Fractalnet: Ultra-deep neural networks without residuals, in: International Conference on Learning Representations (ICLR), 2016, pp. 1–11.

[25] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in neural information processing systems (NIPS), 2015, pp. 2377–2385.

[26] A. Vedaldi, K. Lenc, Matconvnet: Convolutional neural networks for matlab, in: Proceedings of the 23rd ACM international conference on Multimedia, ACM, 2015, pp. 689–692.

[27] R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: A matlab-like environment for machine learning, in: BigLearn, NIPS Workshop, EPFL-CONF-192376, 2011.

[28] Resnet training in torch, ResNet training in Torch [Online], accessed: 2017-04-25.

[29] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the International Conference on Machine Learning (ICML), Vol. 28, 2013, pp. 1058–1066.

[30] M. D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: International Conference on Learning Representations (ICLR), 2013, pp. 1–9.

[31] O. Rippel, J. Snoek, R. P. Adams, Spectral representations for convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 2449–2457.

[32] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 3123–3131.

[33] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2015, pp. 562–570.

[34] M. Liang, X. Hu, Recurrent convolutional neural network for object recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3367–3375.

[35] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: International Conference on Learning Representations (ICLR), 2016, pp. 1–14.

[36] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: Hints for thin deep nets, in: International Conference on Learning Representations (ICLR), 2015, pp. 1–13.

[37] G. Huang, Y. Sun, Z. Liu, D. Sedra, K. Q. Weinberger, Deep networks with stochastic depth, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 646–661.

[38] S. Zagoruyko, N. Komodakis, Wide residual networks, in: British Machine Vision Conference (BMVC), 2016, pp. 1–15.

[39] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: The psychology of learning and motivation, Vol. 24, 1989, p. 92.