Channel-Max, Channel-Drop and Stochastic Max-pooling

Yuchi Huang
NEC Labs
Beijing, China
huang_yuchi@nec.cn

Xiuyu Sun
NEC Labs
Beijing, China
sxy008@gmail.com

Ming Lu
Tsinghua University
Beijing, China
lu-m13@mails.tsinghua.edu.cn

Ming Xu
NEC Labs
Beijing, China
xu-ming@nec.cn

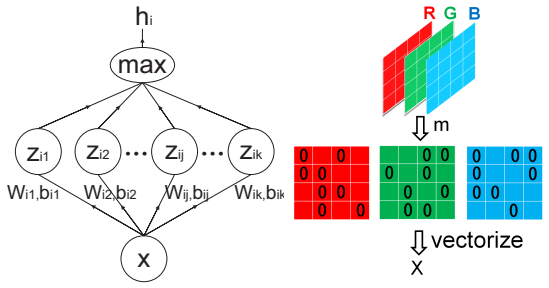
Abstract

We propose three regularization techniques to overcome drawbacks of local winner-take-all methods used in deep convolutional networks. Channel-Max inherits the max activation unit from Maxout networks, but otherwise adopts complementary subsets of input and filters with different kernel sizes as better companions to the max function. To balance the training on different pathways, Channel-Drop is employed to randomly discard half pathways before their inputs are convolved respectively. Stochastic Max-pooling is defined to reduce the overfitting caused by conventional max-pooling, in which half activations are randomly dropped in each pooling region during training and top largest activations are probabilistically averaged during testing. Using Channel-Max, Channel-Drop and Stochastic Max-pooling, we demonstrate state-of-the-art performance on four benchmark datasets: CIFAR-10, CIFAR-100, STL-10 and SVHN.

1. Introduction

Convolutional Neural Network (Convnet) [2, 4, 14] is one of the most powerful tools in applied machine learning to solve various problems. In a Convnet model, convolutional layers are formulated to generate feature maps containing responses to different kernel filters. Then the resulting activations of a convolutional layer are passed to a pooling layer, in which the information in small local regions are subsampled to produce a smaller feature map as input to the next level. There are two conventional choices for the pooling function: average and max. Max-pooling usually works better than Average-pooling empirically.

The size of deep Convnets makes overfitting a significant problem, even with a large number of labeled training examples. Dropout, proposed by Hinton et al. [10], is an effective regularization approach to reduce co-adaption of feature detectors and to improve test performance. During the training of Dropout, activations in each layer are randomly deleted with a probability (usually 50%) and the error is only back-propagated through the remaining activations.



(a) (b)

Figure 1. (a): A max activation unit. (b): How to generate a corrupted x from a color image patch for a maxout unit ($Drop\ Rate = 0.5$).

In a local winner-take-all (LWTA) architecture named Maxout [7], the max function is used in the hidden unit as a companion to Dropout. It is verified that a model contains two or more max activation units is a universal approximator that can approximate any continuous function arbitrarily well, when inputs to the max function are allowed to have arbitrarily high cardinality. Due to this nice property, Maxout outperforms previous methods on various popular datasets for image classification. A similar method based on local competition is proposed to reveal that LWTA structures helps to prevent *catastrophic forgetting* [21], which is common to neural networks when training sets change over time.

According to empirical experiments, when the drop rate is high, the information loss caused by Dropout in maxout networks may result in performance degradation; drop rate should be carefully tuned to achieve the best performance. Another set of heuristic experiments in Section 3 illustrates that the difference between input signal channels is beneficial to the classification task. Recently, An 'inception module' is created in GoogLeNet [23] to achieve the winning classification results for Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). In this method, feature maps produced by convolutional filters with different kernel sizes(e.g. 1×1, 3×3 and 5×5) are combined and

	$k=2$	$k=4$
$Drop\ Rate=0$	22.29	22.26
$Drop\ Rate=0.25$	21.02	20.13
$Drop\ Rate=0.5$	21.39	21.78

Table 1. Comparison of Average Test Errors (%) on CIFAR-10, between Maxout networks with different configurations. k indicates the number of pathways of a maxout hidden unit. We use Net-1 (as introduced in Table 4) in these experiments. Each experiment is run 10 times and average test errors are reported.

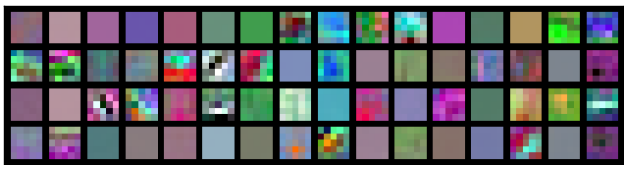


Figure 3. First 16 sets of feature kernels of four pathways of maxout networks ($k=4$), learned from the CIFAR-10 dataset. Each column contains 4 kernels of a max hidden unit. Net-1 (Defined in Table 4) is used in this experiment. Those uniform color patches present under-trained feature kernels.

sent to the next layer. Above work motivates us to introduce better companions to replace the same corrupted input and the same filter size used in a maxout unit. Different from previous methods in which certain subgroups of feature maps [14] or results of different convolutional kernels [23] are simply concatenated, our emphasis is on the competition of feature maps produced by different subsets of input and/or filters of different sizes in the LWTA framework.

We also show that Maxout is prone to unbalanced training, that is, the learned convolutional filters in some pathways of a Maxout unit are over-trained and the convolutional filters in the rest pathways are under-trained. According to the analysis in Section 2, this effect is caused by the winner-take-all mechanism itself. It affects the approximation ability of the max activation function and causes an overfitting problem.

Max-pooling is also a LWTA method in nature: in each local pooling region, it captures the strongest activation and the network error is only back-propagated through the strongest activation during the training. That is, only the pattern corresponding to the strongest activation in a local region is learned and patterns corresponding to other top largest activations are neglected. This processing may lead to overfitting, making Convnets hard to generalize well to diversified test examples.

To sum up, we propose three effective regularization techniques (Channel-Max, Channel-Drop and Stochastic Max-pooling) to overcome above drawbacks of LWTA methods. We empirically verify that each of them can individually improve the optimization accuracy of Convnet models. Moreover, we use the conjunction of these three techniques to set new state-of-the-art results on various benchmark datasets, without improving preprocessing and adopting larger models.

2. Review of Maxout Networks

As shown in Figure 1(a), a max activation function h_i in a hidden layer is formulated as

$$h_i(x) = \max_{j\in[1,k]} z_{ij}, \tag{1}$$

where $x \in \mathbb{R}^n$ and z_{ij} is defined as an affine function

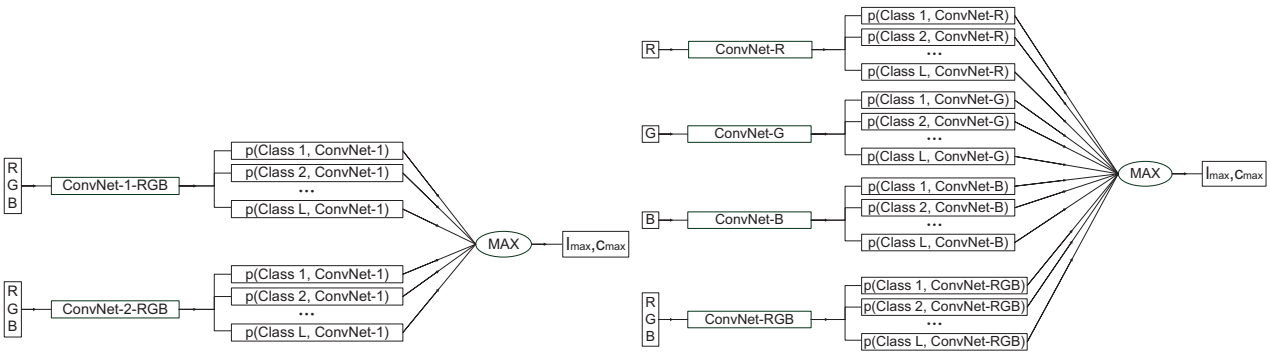
$$z_{ij} = x^T W_{ij} + b_{ij}, \tag{2}$$

in which W_{ij} and b_{ij} are learned parameters w.r.t. a feature detector. In this way, a single hidden unit $h_i(x)$ can be interpreted as a convex piecewise linear function consisting of k locally affine regions on $\mathbb{R}^n$. It is verified in [7] that any continuous function can be approximated arbitrarily well on a compact domain $C \subset \mathbb{R}^n$ by a difference of two max activation functions, with sufficiently large k.

In [7], above max activation function is combined with dropped input to implement maxout networks. As illustrated in the example of Figure 1(b), during the training, the element-wise multiplication is performed between an RGB patch and a randomly generated dropout mask m to produce a corrupted input x. For one maxout unit, the error is only back-propagated through the maximal z_{ij} and only the corresponding W_{ij} and b_{ij} are updated. During the testing, non-corrupted input is used in Equation 2 to calculate z_{ij} and the result of Equation 1 is output to the next layer.

The varying dropout mask often moves the effective inputs far enough to escape the local region surrounding the clean inputs. In this sense, Dropout increases the diversity of training data. However, the information loss caused by a higher dropout rate may harm the training process instead of helping to increase the diversity of training data. For each classification task, drop rate should be carefully tuned to achieve the best performance. For example, as shown in Table 1, Maxout networks with a drop rate of 0.25 achieve better results on the CIFAR-10 dataset when using Net-1 (which is introduced in Section 4.1).

Ideally, a max activation unit can always make the correct decision by choosing the largest z_{ij}, if parameters of all pathways are trained properly. However, in practice, we observe an unbalanced training effect among feature filters w.r.t. different pathways of a maxout unit, as illustrated in Figure 3. Since all the weights and biases in a Convnet model are randomly initialized at the beginning of the training phase, some feature filters may gain an advantage over other filters after some epochs of training. The advantage of stronger feature detectors will continue to be strengthened during the training, due to the winner-take-all mech-



(a) (b)

Figure 2. Two different model combinations used in our heuristic experiment.

Model Combination	Avg. Test Errors
RGB+RGB	20.35
RGB+R+G+B	19.78
HSV+HSV	24.73
HSV+H+S+V	24.33

Table 2. Comparison of Average Test Errors (%) on CIFAR-10, between different model combinations. We use Net-1 (as introduced in Table 4) for these experiments. Each experiment is run 10 times and average test errors are reported.

anism. This is NOT a good thing: since the under-trained feature detectors (relating to different pathways) become a part of the learned piece-wise linear activation function, this learned function can not approximate the actual activation function well. As a result of the unbalanced training, maxout networks easily overfit to the training samples containing patterns represented by those over-trained feature detectors.

3. Our Approach

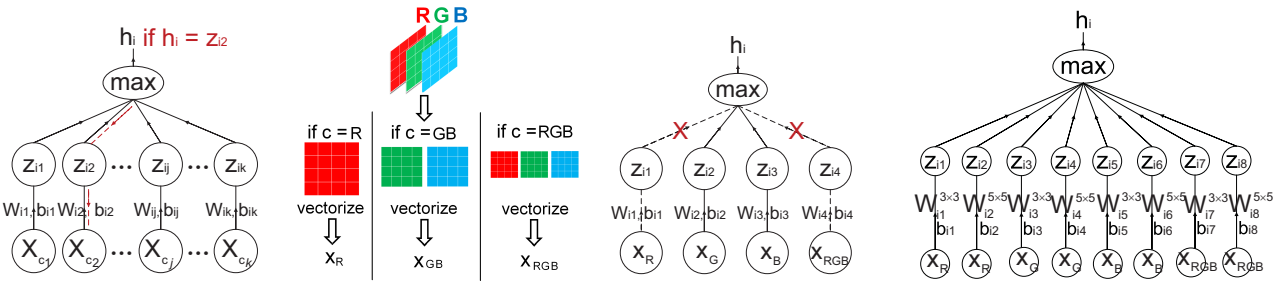
3.1. Channel-Max

Heuristic Experiments. We perform a set of heuristic experiments to compare two model combinations. In the configuration of Figure 2(a), two Convnet models with the same RGB input and the network structure (but the different parameter initializations) are combined; in the configuration of Figure 2(b), one Convnet model using the RGB channels as input and three ConvNet models with distinctive channel (R,G or B) as input are summarized. In the last layer of each model, the softmax processing is operated to generate multi-way classification probabilities. Then the max function is applied on all the output probabilities to obtain the maximum value and the corresponding class label l_{max}:

$$(l_{max}, c_{max}) = \arg\max_{l \in L, c \in C} p(l, c), \tag{3}$$

where L is the set of class labels and C is the set of models in a model combination. For example, $C = \{ConvNet - 1 - RGB, ConvNet - 2 - RGB\}$ for the configuration in Figure 2(a) and $C = \{ConvNet - R, ConvNet - G, ConvNet - B, ConvNet - RGB\}$ for the configuration in Figure 2(b). We also compare two similar combinations in the HSV color spaces. All experiments are implemented using Net-1 (defined in Table 4) here, a small Convnet structure for fast validation of ideas. As shown in Table 2, the configuration in Figure 2(b) consistently outperforms the configuration in Figure 2(a) on CIFAR-10. Similar phenomenon is found on different architectures and on several different datasets. This performance advantage reveals that the competition among models using different subsets of signal channels as input (e.g. $\{R\}$, $\{G\}$,$\{B\}$ and $\{RGB\}$) may be more effective. However, it is quite inefficient to combine different models only in the last layer. Furthermore, compared to the first model, the size of the second one is doubled, thus networks in Figure 2(b) need doubled training and testing time. This is avoidable if we employ the competition between subsets of channels in the local-winner-take-all framework.

Channel-Max. If we replace randomly-corrupted input x in Figure 1(a) by a local feature x_{c_j} only containing information from a subset of channels, the max activation function just provides a natural selection mechanism for each hidden unit to exploit channel information, as shown in Figure 4(a). We refer to this technique as Channel-Max, in contrast with Maxout. Figure 4(b) gives an example to illustrate how a Channel-Max unit in the first convolutional layer works on a color image patch. All channel subsets of an RGB image include $\{R\}$, $\{G\}$, $\{B\}$, $\{RG\}$, $\{RB\}$, $\{GB\}$, $\{RGB\}$. For those mono-channel subsets such as $\{R\}$, $\{G\}$ and $\{B\}$, 2D filters will be applied to perform the convolution; for those multi-channel subsets such as $\{RG\}$, $\{RB\}$, $\{GB\}$, and $\{RGB\}$, 3D filters will be utilized instead. To speed up the computation, usually we only adopt four channel subsets of color images in our experiments: $\{R\}$, $\{G\}$,$\{B\}$ and $\{RGB\}$ (Figure 4(c)). To achieve a robust processing, we perform a cross-feature map normal-



(a) (b) (c) (d)

Figure 4. (a): A Channel-Max unit, in which each pathway is corresponding to a subset of channels. (b): How to generate an x_{c_j} from a color image patch for a Channel-Max unit. (c): The basic Channel-Max model used in this paper contains four pathways w.r.t. four different channel subsets: R, G, B and RGB. Channel-Drop randomly discards each pathway with a probability of 0.5. (d): The extended Channel-Max model in which kernel filters of two different size are utilized.

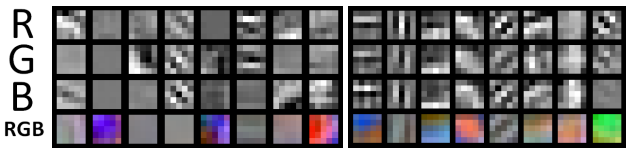
ization on each z_{ij} before inputting it into the max function. When training with Channel-Max, in each Channel-Max unit the error is only back-propagated through a pathway that has produced the maximum value in the last forward propagation step; during the testing phase, for each Channel-Max unit only the most discriminative feature is retained for the recognition task.

Extended Channel-Max. Considering different features may have different spatial layout, we can further strengthen the competition in Channel-Max by introducing filters with various kernel sizes(e.g. 3×3 and 5×5) to explore the optimal local structure of features, as illustrated in Figure 4(d). To make the convolved feature maps (produced by filters with different kernel size) comparable, the input layer should be padded with zeros properly so that those convolved results have the same size.

The idea of connecting units in a convolutional layer to certain subgroups of feature maps [14] [11] and multi-resolution filtering [23] dates far back. Different with prior work that directly concatenates feature maps produced by different subsets of input and/or filters of different sizes, our emphasis is on the competition of those features in each winner-take-all hidden unit. In this way, not only the most discriminative feature is learned, but also the computational bottlenecks are removed: results of different feature subsets and various filters are computed but only a small portion of them are sent to the next layer. That is, we can dynamically learn better features without increasing the computational load of next level.

3.2. Channel-Drop

Channel-Max is still a LWTA method, which causes an unbalanced training problem, as shown in Figure 5(a). In order to reduce the unbalanced training effect, we introduce the Channel-Drop technique, in which each pathway is randomly disconnected with a probability of 0.5 in each training loop, as shown in Figure 4(c). By preventing (randomly selected) half pathways from updating in each loop,



(a) (b)

Figure 5. First 8 sets of learnt feature kernels on R, G, B and RGB pathways of Channel-Max networks without Channel-Drop (a) and those of Channel-Max networks with Channel-Drop (b). All filters are learned from CIFAR-10 and Net-1 (Defined in Table 4) is used to implement both models. Those uniform-gray patches in (a) present under-trained feature kernels, which are rarely found in (b).

Channel-Drop can rebalance the training on different channel subsets (as illustrated in Figure 5) and alleviate the overfitting problem (as illustrated in Figure 7(a)).

Channel-Drop is different from Dropout [10] and DropConnect [24] in both training and testing phases. During the training, the random discarding in Channel-Drop only happens on pathways (corresponding to different feature subsets or filter kernels of different size) inside each max hidden unit. In Dropout, outputs of hidden units are randomly set to zero, while in Channel-Drop outputs of all hidden units are retained. In DropConnect weights on all connections between two layers are randomly set to zero, while in Channel-Drop all local weights corresponding to a disconnected pathway are temporarily disabled. What's more, Channel-Drop is only adopted in the training phase (to bal-

	# of PARs	# of PARs to be updated
A hidden unit with a 3D RGB filter	n+1	n+1
Maxout (k=2)	2n+2	2n+2
Maxout (k=4)	4n+4	4n+4
Channel-Max	2n+4	2n+4
Channel-Max with Channel-Drop	**2n+4**	**n+2**

Table 3. Comparison of the total number of parameters and the number of parameters to be updated between five micro-structures. $k = 4$ denotes four pathways under one maxout hidden unit. The input is an RGB image patch containing n elements in total (each channel contains $n/3$ pixels). Four channel subsets(R, G, B and RGB) are used for the Channel-Max unit.

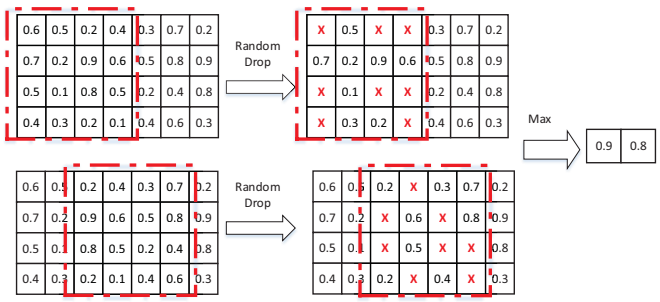


Figure 6. (a): Stochastic Max-pooling during the training. (b): We use the weighted sum of top 3 largest values in each region to compute the pooling value during the test time.

ance the training); because it is used in the winner-take-all framework, the weight scaling (used in Dropout and DropConnect) is not required in the testing phase.

Another benefit brought by Channel-Max and Channel-Drop is decreased number of parameters and training burden. In Table 3.1, we compare the total number of parameters and the number of parameters to be updated in each training loop between five micro-structures: a ConvNet hidden unit with a 3D RGB filter, a Maxout unit ($k = 2$), a Maxout unit ($k = 4$), a Channel-Max unit ($k = 4$, the basic version as illustrated in Figure 4(c)) and a Channel-Max unit with Channel-Drop ($k = 4$, the basic version).

3.3. Stochastic Max-pooling

To relieve the overfitting problem of Max-pooling, we introduce a technique called Stochastic Max-pooling. As illustrated in Figure 6, **during the training**, we stochastically throw away activations in each pooling region with a probability p_a (in this paper, we set $p_a = 0.5$), then utilize the max function on remaining elements to get the pooling value s. In this sense, each feature value generated by Stochastic Max-pooling is a random variable; the probability that the ith largest activation a_i in a region R is picked equals $p_a^{i-1}(1 - p_a)$. **During the test time**, instead of employing above sampling method, we use a probabilistic averaging of top h largest activations to compute the pooling value s of a region R:

$$s = \sum_{i=1}^{h} p_a^{i-1}(1 - p_a)a_i. \quad (4)$$

When h equals the size of the local region, the pooled value s becomes the expectation over all possibly selected activations. In practice, we set $h = 3$ for the best performance: the model with $h = 3$ produces better results than $h = 1$ and $h = 2$; when $h >= 4$, the model performance remains almost the same when h increases. Equation 4 can be viewed as a form of model averaging, in which more patterns from the top h activations of a local region R are learned.

The training of Stochastic Max-pooling is different from Dropout with Max-pooling, because the random selection in our method is done in each local region independently. As in Figure 6(a), considering two neighboring regions with a 2-pixel stride, in our method one activation with a value of 0.9 in the stride area may be dropped in one region but selected in the neighboring region. While in Dropout with Max-pooling, the random dropping is done globally in a feature map.

Our approach is also different from Stochastic Pooling [25], in which a pooled value is sampled from activations in a region, according to probabilities of those activations computed by normalizing the values of activations within the region. In this way, a non-top activation has a much larger probability to be selected, compared with our Stochastic Max-Pooling. Comparison between the results of two methods can be found in Table 6 (Method 10 v.s. Method 11).

4. Experiments

4.1. Overview

Datasets. We evaluate our framework on four benchmark datasets: CIFAR-10 [12], CIFAR-100 [12], STL-10 [3] and SVHN [18]. For CIFAR-10 and CIFAR-100, we apply the same global contrast normalization and ZCA whitening as was used by Goodfellow et al. in the maxout network [7]. For STL-10, we employ a standard normalization preprocessing procedure [13] in which the per-pixel mean value is subtracted from each STL-10 image. For SVHN, we utilize local contrast normalization preprocessing the same way as [25].

Implementation. We construct two Convnet structures – Net-1 (as in Table 4) and Net-2 (as in Table 5) in this paper. Net-1 is a smaller model to fast validate experimental ideas (e.g. experiments listed in Table 9), which may not produce state-of-the-arts results; Net-2 is used to compare our framework to different methods (listed in Table 6) on a fair basis and achieve the best performance. In the experiments of Table 6, we only utilize the basic Channel-Max model (filter size: 5×5) in the first layer of Net-2. In the experiments of Table 7, 8, 10 and 11, we also employ the extended Channel-Max in the first layer and the basic Channel-Max in the third convolutional layer. The total number of parameters of Net-2 is largely decreased, compared with Maxout networks used in Section 5.2 to

	1	2	3	4	5	6	7
Layer	Conv.	Pool.	Conv.	Pool.	Conv.	Pool.	Softmax
Kernel on each channel	5×5	3×3	5×5	3×3	5×5	3×3	–
# of hidden units	32	–	32	–	32	–	–

Table 4. Structure of Net 1, for fast validation of experimental ideas.

	1	2	3	4	5	6	7	8	9
Layer	Conv.	Pool.	Conv.	Conv.	Pool.	Conv.	Pool.	Full Conn.	Softmax
Kernel on each channel	5×5 (and 3×3, in extended Channel-Max)	3×3	3×3	3×3	3×3	3×3	3×3	–	–
# of hidden units	96	–	192	192	–	192	–	512	–

Table 5. Structure of Net 2, for the fair comparison in Table 6 and the best performance.

5.4 of [7]. We implement all the methods in these tables on the super fast cuda-convnet code developed by Alex Krizhevsky [13]. Except those hidden units using the max activation function (like in Channel-Max and Maxout), all other hidden units in these methods adopt the rectified linear function and 3D convolutional filters. We use 3×3 pooling with stride 2 for each pooling layers in all three structures. Each covolutional layer is followed by a processing which normalizes the convolution outputs at each location over a subset of neighboring feature maps. **Except in the experiments of Table 9, we only employ four subsets of the RGB color space ($\{R\}$, $\{G\}$,$\{B\}$ and $\{RGB\}$); the HSV and LAB channels are not used to produce our best results.**

Experiment Protocols. We train our models using stochastic gradient descent with a batch size of 128 examples; we set momentum = 0.9 and weight decay = 0.001; we initialize all the weights from zero-mean Gaussian distribution with a standard deviation 0.005.

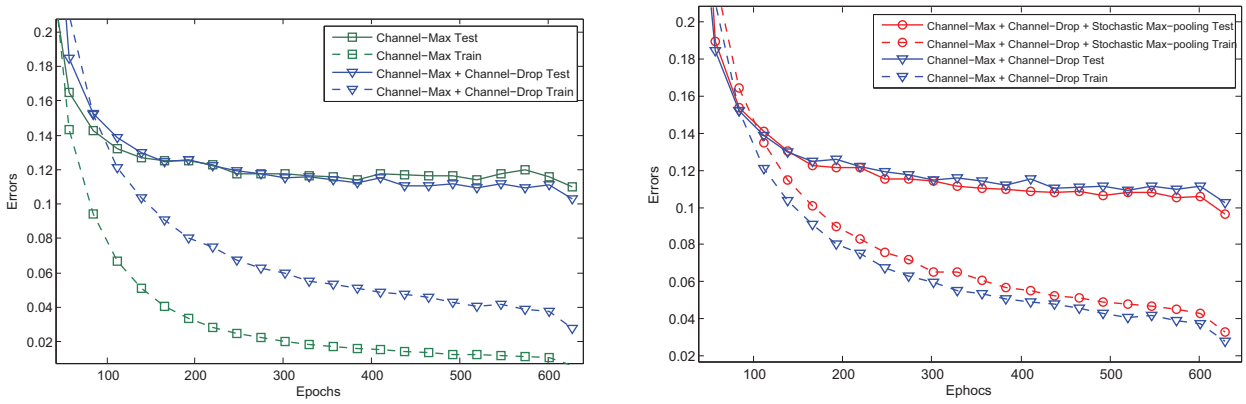
We follow a similar training methodology used in [13]. In the first round of training, we divide the whole training set into the training subset and the validation subset. We use the training subset to train the model (with a learning rate of 0.001 on kernel weights and a learning rate of 0.002 on biases) and monitor the errors on the validation subset. According to our observation, on CIFAR-10, CIFAR-100 and STL-10, the validation errors stop improving before the 600th epoch; on SVHN, the validation errors stop decreasing before the 100th epoch. Therefore, we perform a two-stage training in the second round by using the whole training set. For CIFAR-10, CIFAR-100 and STL-10, in the first stage, we retrain our models from the scratch with the whole training set (including the validation subset) for 600 epochs; for SVHN, we retrain our model from the scratch with the whole training set for 100 epochs. After this, all learning rates are reduced by a factor of 10 (0.0001 on kernel weights and 0.0002 on biases). For all four datasets, we use these decayed learning rates to perform 50$\sim$150 more epochs of training in the second stage and obtain the final models for testing.

4.2. In-depth Analysis on CIFAR-10

Comparison between different configurations, using Net-2. As illustrated in Table 6, we make a fair comparison between different methods by **only changing the configuration in the first two layers**; the rest layers of all the methods are the same, in which the rectified linear units with 3D filters are adopted in convolution; Max-pooling is employed for sub-sampling and Dropout is utilized for regularization. In Method 8 to Method 11, **only the basic Channel-Max with 4 GRB subsets is used**. In Method 4 to Method 7, we choose the Maxout structure with $k = 4$ which contains doubled number of parameters (compared to the basic Channel-Max) and set the drop rate to be 0.25 for the best performance. According to Table 6, it can be shown that: 1) Channel-Max, Channel-Drop and Stochastic Max-pooling can individually improve the optimization accuracy. This can be shown by comparing methods in Table 6, such as the comparison between Method 8 and Method 2/4; the comparison between Method 9 and Method 5; the comparison between Method 9/11 and Method 8; the comparison between Method 5/6/7 and Method 4. 2) The conjunction of these three techniques achieves the best result, according to the result of Method 11.

Comparison to state-of-the-art methods. As shown in Table 7, by only using our configuration in the first two layers of Net-2 (with the basic Channel-Max), we obtain a test error of 9.47%, which has already set the new state-of-the-art performance on CIFAR-10. With data augmentation, this model achieves a test error of 7.88%. Based on Net-2, we also implement a configuration to utilize the extended Channel-Max in the first layer and the basic Channel-Max in the third convolutional layer (in this layer, we divide all feature maps into three sets and also keep a set containing all feature maps. Then we use them as inputs in four different pathways). This larger model obtains a test error of 9.17%; with data augmentation, it achieves a test error of 7.41%. The latter result outperforms those of all other methods by a decent margin, except that of [8] achieved by a much deeper and wider model with (padded) spatially-sparse inputs.

The regularization effect of Channel-Drop and Stochastic Max-pooling. As shown in Figure 7, the over-



(a) (b)

Figure 7. The regularization effect of Channel-Drop (a) and Stochastic Max-pooling (b). With these two techniques, training errors remain higher while test errors continue to decrease.

Method #	The first 2-layer configuration	Train Error %	Test Error %
1	Rectified Conv. + 3D filters+ Max-pooling, without Dropout	0.57	11.06
2	Rectified Conv. + 3D filters + Max-pooling, with Dropout	3.91	11.15
3	Rectified Conv. + 3D filters + Stochastic Max-pooling	0.68	10.65
4	Maxout ($k = 4$) + Max-pooling	3.32	10.72
5	Maxout ($k = 4$) + Channel-Drop + Max-pooling	3.68	10.49
6	Maxout ($k = 4$) + Stochastic Max-pooling	3.57	10.22
7	Maxout ($k = 4$) + Channel-Drop + Stochastic Max-pooling	3.74	10.01
8	Channel-Max + Max-pooling	0.65	10.75
9	Channel-Max + Channel-Drop + Max-pooling	3.52	10.07
10	Channel-Max + Channel-Drop + Stochastic Pooling [25]	3.65	9.80
11	**Channel-Max + Channel-Drop +Stochastic Max-pooling**	3.71	**9.47**

Table 6. Comparison between different configurations, using Net-2.

Method	Test Error %
Conv. maxout + Dropout [7]	11.68
NIN + Dropout [16]	10.41
Deeply Supervised Nets [15]	9.78
Channel-Max + Channel-Drop +Stochastic Max-pooling	**9.47**
Extended Channel-Max + Channel-Drop +Stochastic Max-pooling	**9.17**
CNN + Spearmint + Data Augmentation [19]	9.50
Conv. maxout + Dropout + Data Augmentation [7]	9.38
DropConnect + 12 networks + Data Augmentation [24]	9.32
NIN + Dropout + Data Augmentation [16]	8.81
Deeply Supervised Nets + Data Augmentation [15]	8.22
Spatially-sparse Convolutional Neural Networks + + Data Augmentation [8]	6.28
Channel-Max + Channel-Drop +Stochastic Max-pooling + Data Augmentation	**7.88**
Extended Channel-Max + Channel-Drop +Stochastic Max-pooling + Data Augmentation	**7.41**

Table 7. Comparison to state-of-the-art methods on CIFAR-10.

Method	Test Error %
Stochastic Pooling [25]	42.51
Conv. maxout + Dropout [7]	38.57
Tree based priors [20]	36.85
NIN + Dropout [16]	35.68
Deeply Supervised Nets [15]	34.57
Channel-Max + Channel-Drop +Stochastic Max-pooling	**34.15**
Extended Channel-Max + Channel-Drop +Stochastic Max-pooling	**32.97**
Spatially-sparse Convolutional Neural Networks + Data Augmentation [8]	24.30
Channel-Max + Channel-Drop +Stochastic Max-pooling + Data Augmentation	**30.97**
Extended Channel-Max + Channel-Drop +Stochastic Max-pooling + Data Augmentation	**28.56**

Table 8. Comparison to state-of-the-art methods on CIFAR-100, using Net 2 and the RGB color space.

fitting problem is largely relieved by Channel-Drop and Stochastic Max-pooling. In this figure, training error curves w.r.t. Channel-Drop/Stochastic Max-pooling remain higher while corresponding test errors continue to decrease.

Method	Test Error %
Discriminative Sum-Product Networks [5]	37.7
Hierarchical Matching Pursuit [1]	35.5
Representation Learning with Nonnegativity Constraints [9]	32.1
Deep feedforward networks [17]	32.0
Multi-Task Bayesian Optimization [22]	29.9
Channel-Max + Channel-Drop + Stoch. Max-pooling	**32.6**
Extended Channel-Max + Channel-Drop + Stochastic Max-pooling	**28.7**

Table 10. Comparison to state-of-the-art methods on STL-10, using Net 2 and the RGB color space.

Method	Test Error %
Conv. maxout + Dropout [7]	2.47
NIN + Dropout [16]	2.35
Channel-Max + Channel-Drop +Stochastic Max-pooling	2.31
Multi-digit Number Recognition [6]	2.16
Extended Channel-Max + Channel-Drop +Stochastic Max-pooling	2.01
DropConnect [24]	1.94
Deeply Supervised Nets [15]	**1.92**

Table 11. Comparison to state-of-the-art methods on SVHN, using Net 2 and the RGB color space.

Method and its first layer configuration	RGB	LAB	HSV
Rectified Conv. + 3D filters	20.73	20.37	23.58
Channel-Max + Channel-Drop	**19.04**	**19.61**	**20.15**
Maxout + non-corrupted inputs ($k = 2$)	20.27	20.16	23.03
Maxout + non-corrupted inputs ($k = 4$)	19.52	20.13	22.53

Table 9. Comparison of Test Errors (%) between different methods in 3 color spaces, using Net-1. Stochastic Max-pooling is employed in all the pooling layers of different methods.

Comparison in different color space, using Net-1. We use the Net-1 structure to fast verify the universal advantage of our framework in three different color spaces: RGB, HSV and LAB, as illustrated in Table 9. In this experiment, we only change the configuration in the first layer and keep rest layers of different methods the same; we employ Stochastic Max-pooling in all the pooling layers of different methods.

4.3. Experments on CIFAR-100, STL-10 and SVHN

Results on CIFAR-100. To demonstrate the robustness of our algorithm, we also conduct a comparison on CIFAR-100 [12]. With the same implementation used for CIFAR-10, our framework with the basic Channel-Max in the first layer obtains a test error rate of **34.15%**. Our framework with the extended Channel-Max achieves a test error rate of **32.97%**. These two results surpass the current best performance (as shown in Table 8). For the case with data augmentation, our model with the extended Channel-Max achieves an error of **28.56%**.

Results on STL-10. STL-10 [3] has larger 96×96 pixel images and less labeled data (5,000 training and 8,000 test) than CIFAR-10. A very large set of unlabeled examples is provided in STL-10 to learn image models prior to supervised training. In this experiment, we resize all samples in STL-10 to 32×32 image patches to fit the model used for CIFAR-10 and CIFAR-100. We retrain our model from scratch, only relying on the labeled samples. With augmented inputs, our extended model obtains a test error of 28.7%, the lowest published result as of writing (Table 10).

Results on Street View House Numbers(SVHN). The SVHN dataset [18] contains 630,420 32×32 color images, divided into training set, testing set and an extra set. In this experiment, we follow Goodfellow et al. [7] to perform the data pre-processing. The structure and parameters used in this experiment are the same to those used for CIFAR-10. Our best model obtains a test error of 2.01%, which is comparable to state-of-the-arts results shown in Table 11.

5. Conclusion

We introduce a novel model for image classification tasks, in which three effective techniques are employed to regularize deep convolutional networks. In Channel-Max, we exploit the competition between distinctive features to boost the classification results. By Channel-Drop and Stochastic Max-pooling, the unbalanced training is rectified and the overfitting problem is largely relieved. The effectiveness and robustness of the proposed framework is demonstrated by extensive experimentation on four general purpose image databases.

References

[1] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. In *In International Symposium on Experimental Robotics (ISER)*, 2012. 8

[2] J. Bouvrie. Notes on convolutional neural networks. 2006. 1

[3] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. 5, 8

[4] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4), 1980. 1

[5] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. 8

[6] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *Workshop at International Conference on Learning Representations (ICLR)*, 2013. 8

[7] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013. 1, 2, 5, 6, 7, 8

[8] B. Graham. Spatially-sparse convolutional neural networks. *CoRR*, abs/1409.6070, 2014. 6, 7

[9] T. han Lin and H. T. Kung. Stable and efficient representation learning with nonnegativity constraints. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1323–1331, 2014. 8

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012. 1, 4

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):221–231, Jan. 2013. 4

[12] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 5, 8

[13] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. 5, 6

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998. 1, 2, 4

[15] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. *ArXiv e-prints*, Sept. 2014. 7, 8

[16] M. Lin, Q. Chen, and S. Yan. Network in network. *Workshop at International Conference on Learning Representations (ICLR)*, 2013. 7, 8

[17] B. Miclut, T. Käster, T. Martinetz, and E. Barth. Committees of deep feedforward networks trained with few data. *CoRR*, abs/1406.5947, 2014. 8

[18] Y. Netzer, T. Wang, A. Coates, R. 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. 5, 8

[19] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. 7

[20] N. Srivastava and R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems (NIPS)*. 2013. 7

[21] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in Neural Information Processing Systems*, pages 2310–2318. 2013. 1

[22] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2004–2012. Curran Associates, Inc., 2013. 8

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1, 2, 4

[24] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013. 4, 7, 8

[25] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *International Conference on Learning Representations (ICLR)*, 2013. 5, 7