Rank-based pooling for deep convolutional neural networks

Zenglin Shi, Yangdong Ye*, Yunpeng Wu
School of Information Engineering, Zhengzhou University, Zhengzhou, 450052, China

A R T I C L E  I N F O
Article history:
Received 1 September 2015
Received in revised form 6 June 2016
Accepted 4 July 2016
Available online 20 July 2016

Keywords:
Pooling
Deep learning
Image classification
Convolutional neural network

A B S T R A C T
Pooling is a key mechanism in deep convolutional neural networks (CNNs) which helps to achieve translation invariance. Numerous studies, both empirically and theoretically, show that pooling consistently boosts the performance of the CNNs. The conventional pooling methods are operated on activation values. In this work, we alternatively propose rank-based pooling. It is derived from the observations that ranking list is invariant under changes of activation values in a pooling region, and thus rank-based pooling operation may achieve more robust performance. In addition, the reasonable usage of rank can avoid the scale problems encountered by value-based methods. The novel pooling mechanism can be regarded as an instance of weighted pooling where a weighted sum of activations is used to generate the pooling output. This pooling mechanism can also be realized as rank-based average pooling (RAP), rank-based weighted pooling (RWP) and rank-based stochastic pooling (RSP) according to different weighting strategies. As another major contribution, we present a novel criterion to analyze the discriminant ability of various pooling methods, which is heavily under-researched in machine learning and computer vision community. Experimental results on several image benchmarks show that rank-based pooling outperforms the existing pooling methods in classification performance. We further demonstrate better performance on CIFAR datasets by integrating RSP into Network-in-Network.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

CNNs have recently attracted a great deal of attentions due to their success in large-scale visual recognition tasks (Krizhevsky, Sutskever, & Hinton, 2012; Simonyan & Zisserman, 2015; Szegedy et al., 2014; Zeiler & Fergus, 2014). Such excellent performance of CNNs is arguably attributed to their strong ability of learning powerful and interpretable image features (Zeiler & Fergus, 2014). CNN consists of alternating convolutional layers and pooling layers. The cascade of convolution-pooling structure helps to obtain more robust features under translation or small object deformations. A feature extraction stage of this structure is illustrated in Fig. 1.

The aim of convolutional layer is to extract the patterns, found within local regions of input images, that are common throughout the dataset. This is done by applying many filters to the inputs, computing the inner product of the filters at every location in the inputs and outputting these as feature maps. A non-linear function is then applied to each feature map. The resulting activations are then passed to the pooling layer.

In this paper, we focus on the pooling layer. Pooling layer merges semantically similar features into one by preserving task-related information while removing irrelevant details. It provides some degree of invariant to translations of the inputs, which plays an important role in boosting classification performance (Boureau, Ponce, & LeCun, 2010; Boureau, Roux, Bach, Ponce, & LeCun, 2011; LeCun, Bottou, Bengio, & Haffner, 1998). The most commonly used pooling operations are average and max pooling. However, recent works have indicated that neither of these two pooling techniques may be qualified to be optimal (Boureau et al., 2010; Feng, Ni, Tian, & Yan, 2011; Zeiler & Fergus, 2013). Their drawbacks lie in the fact that these two deterministic pooling methods either treat all the activations uniformly or only select the strongest one. Two pooling alternatives, LP pooling (Boureau et al., 2010) and stochastic pooling (Zeiler & Fergus, 2013), have been proposed to address some of the problems with max pooling and average pooling. However, these two pooling strategies are still operated on activation values.

In this work, we desire to bring the rank of activation into the pooling operation. A novel pooling mechanism called rank-based pooling is proposed. Some advantages of utilizing the rank of activation can be summarized from three aspects: (1) Important activations can be easily distinguished by their ranks. (2) Ranking list is invariant under changes of activation values in a pooling region, and thus rank-based pooling operation may achieve more robust performance. (3) The reasonable usage of rank can avoid the
scale problems encountered by value-based methods. For example, whether the activation is ten times larger than the next activation or 0.001%, the rank of activation does not make any difference. All that matters is the rank relative to other individuals.

We consider rank-based pooling as an instance of weighted pooling where a weighted sum of activations is used to generate the pooling output. Three new pooling methods, rank-based average pooling (RAP), rank-based weighted pooling (RWP) and rank-based stochastic pooling (RSP), are introduced according to different weighting strategies. RAP can be regarded as a tradeoff between max pooling and average pooling. A rank threshold \( t \) is used to eliminate some near-zero activations. The weights of those selected activations are set to be \( 1/t \) while others are kept as 0. In RWP, each activation is weighted by a coefficient in range of \((0, 1)\) computed based on its rank, and more important activations have larger weights. RSP replaces the conventional deterministic pooling operations with a stochastic procedure. A multinomial distribution is created from the probabilities \( p \) based on the ranks. The weights of the activations can be got by sampling from this distribution.

Preserving discriminative information is critical to the success of pooling method in recognition tasks. Diverse information and salient information dominate the discriminant ability of the pooled features. From this point of view, average pooling and max pooling are two extreme cases where the former aims at maintaining diverse information while the latter focuses on preserving salient ones. A good tradeoff between salient information and diverse information may boost the discriminating capability of the resultant features for image classification. We propose to compute the entropy of pooled activation to measure the tradeoff. Based on this, we analyze the discriminant ability of various pooling methods.

In general, the main contributions of the current work can be summarized as follows:

- A novel pooling mechanism called rank-based pooling is proposed in this work. The rank of activation in a pooling region can be reasonably used when performing pooling operation.
- Three new pooling methods are introduced to show the advantages of this pooling mechanism. These three methods are applicable to any types of activation functions in the literature and are very effective for solving the problems of the existing pooling methods.
- A novel criterion is provided to analyze the discriminant ability of various pooling methods.
- Experimental results show that the proposed pooling mechanism is superior to existing pooling methods. We further demonstrate better performance on CIFAR datasets by integrating RSP into Network-in-Network (Lin, Chen, & Yan, 2014).

The rest of this paper is organized as follows. We give a description in detail about pooling mechanism used in CNNs and briefly review the existing pooling methods in Section 2. Section 3 then elaborates on rank-based pooling mechanism and provides a novel criterion to analyze the discriminant ability of various pooling methods. Next, Section 4 reports comprehensive experimental results for classification on four benchmark image datasets (MNIST, CIFAR-10, CIFAR-100, and NORB). Finally, Section 5 concludes with a discussion of future work directions.

2. Related work

Spatial pooling is a key mechanism in popular recognition architectures such as convolutional networks (LeCun et al., 1998), Neocognitron (Fukushima & Miyake, 1982), and HMAX (Riesenhuber & Poggio, 1999), providing some invariance to small transformations of the inputs. It can be traced back to the seminal work of Hubel and Wiesel on complex cells in the visual cortex (Hubel & Wiesel, 1962), and it is related to Koenderink et al.’s concept of locally orderless images (Koenderink & Van Doorn, 1999). Pooling also is used in popular feature extraction methods such as SIFT (Lowe, 2004), histograms of oriented gradients (HOG) (Dalal & Triggs, 2005) and so on. In addition, Srivastava et al.’s “Winner Takes All” type of activation functions (Srivastava, Masci, Kazeronian, Gomez, & Schmidhuber, 2013) and Kahou et al.’s temporal pooling (Kahou, Bouthillier, Lamblin, & Gulcehre, 2015) applied pooling mechanism into the multi-layer perceptron. This work is mainly concerned with the pooling used in deep convolutional networks.

Pooling is a crucial component in deep CNN which helps to obtain large increases in performance. The motivation behind pooling is that activations in the pooled map are less sensitive to the precise locations of structures within the image than the original feature map (LeCun et al., 1998). It summarizes the outputs of neighboring groups of neurons in the same kernel map through a pooling function. In the current deep learning literature, popular pooling functions include max and average. Studies have theoretically and empirically shown that the details of the pooling operation can greatly influence the performance (Boureau et al., 2010, 2011). For a given classification problem, the optimal pooling type may be neither max nor average pooling (Boureau et al., 2010). Thus, some new pooling methods have been developed.

A successful variant of pooling is Lp pooling (Boureau et al., 2010; Gulcehre, Cho, Pascanu, & Bengio, 2014). Lp pooling can be viewed as a continuous parametrization transition from average to max pooling. Two special cases of Lp pooling are notable. p = 1 can be regarded as a simple form of averaging, and p = \( \infty \) corresponds to max-pooling. Lp pooling has shown to give large improvements in error rate in computer vision tasks compared to max pooling (Sermanet, Chintala, & LeCun, 2012). Our RAP like Lp pooling detailed in Section 3 provides a more flexible way to transition smoothly from max to average pooling.

Stochastic pooling (Zeiler & Fergus, 2013) is another improved pooling strategy. This method is less prone to over-fitting due to the random strategy which picks the activations within each
pooling region based on the normalized probability of activation values. At test time, probabilistic weighting is adopted. In this paper, Zeiler’s approach is called value-based stochastic pooling (VSP) and value-based weighted pooling (VWP). VSP provides improvements over $L_P$ pooling in error rate (Sainath et al., 2015). However, VSP is only applicable to rectified linear unit (ReLU) (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012; Nair & Hinton, 2010). The non-negative constraint on the elements in the pooling regions makes it unsuitable for some novel generalizations of ReLU such as leaky rectified linear unit ($LReLU$) (Maas, Hannun, & Ng, 2013) and parameter rectified linear unit (PreLU) (He, Zhang, Ren, & Sun, 2015). To alleviate potential problems caused by the hard 0 activation of ReLU, Maas et al. alternatively introduced LReLU: $LReLU(x) = \max(x, ax)$. LReLU allows for a small, non-zero gradient $a$ when activation is saturated and not active. Kaiming et al. proposed a variant of LReLU called PreLU: $PreLU(x) = \max(x, ax)$. In PreLU, the slopes of negative part are learned from data rather than predefined. In general, these generalizations have been shown to have significant benefits over ReLU (Bing Xu et al., 2015). Additionally, VSP reduces to max pooling when the probability of maximum activation is significant larger than others. RSP proposed in Section 3 is different from VSP, in which the probability of each activation is computed based on its rank instead of its raw value and can be controlled by a hyper-parameter.

Some other works on pooling about the hyper-parameter of pooling, such as pooling size and pooling stride, are also meaningful. The size of pooling region plays an important role in pooling operation. Zeiler et al. found that the optimal size is $3 \times 3$, with smaller regions over-fitting and larger regions possibly being too noisy during training (Zeiler & Fergus, 2013). Pooling units are usually spaced at least several pixels apart, so that there are fewer number of pooling units than convolutional layer outputs. Making this spacing smaller than the size of the neighborhood that the pooling units summarize produces overlapping pooling (Krizhevsky et al., 2012). Krizhevsky et al. reported that overlapping pooling can reduce over-fitting. The proposed methods can be easily combined with these works.

3. Proposed rank-based pooling

In rank-based pooling, the activations within each pooling region are firstly sorted according to their activation value in descending order. Consider a single pooling region: Let $a$ be the sorted activations, $a: T \to \{a_{\text{max}}, \ldots, a_{\text{min}}\}$. Then, ranks are assigned to $a(i)$ based on their position in the sorted vector. Let $r$ be the ranking $r: T \to \{1, \ldots, n\}$ where $r(i)$ is the rank of the activation $i$, and $n$ denotes the size of pooling region. Lower ranks are assigned to higher activations.

$$a(i) > a(j) \Rightarrow r(i) < r(j).$$

There is ambiguity when two activations have the same value. To resolve this, we add the following constraint:

$$a(i) = a(j) \land i < j \Rightarrow r(i) < r(j).$$

The rationale behind the proposed method is to exploit a reasonable ranking strategy for the activations in a pooling region when necessary. According to different strategies to assign weights, this pooling mechanism can be realized as RAP, RWP and RSP, respectively. Fig. 2 gives a simple example of RAP, RWP and RSP. The following sections elaborate on each of the three pooling methods.

3.1. Rank-based average pooling

RAP is introduced to alleviate the problem of useful information loss encountered by max pooling and average pooling. Throwing away the non-maximum activations fully in max pooling causes severe information loss. And likewise, discriminative information may lose in average pooling due to the average operation where near-zero negative activations may downplay higher activations. RAP alleviates some of their problems by using an average of top $t$ highest activations. The weights of top $t$ highest activations are set to $1/t$, and other weights are set to 0. The pooled output can be calculated by the following formula:

$$s_j = \frac{1}{t} \sum_{i \in R_j, r_i \leq t} a_i$$

where $t$ denotes the rank threshold that is used to determine which activations are involved in averaging, $R_j$ is pooling region $j$ in feature maps, and $i$ is the index of each activation within it. $r_i$ and $a_i$ represent the rank of activation $i$ and the value of activation $i$, respectively. The value of $t$ should not be too large or too small since $t = 1$ corresponds to max-pooling, and $t = n$ (pooling size) reduces to average-pooling. $t$ should ensure that RAP provides a good tradeoff between max pooling and average pooling. Optimal $t$ may be different in different visual tasks, but we...
suggest that setting \( t \) at medium ranges (i.e., median value) may lead to satisfactory result.

RAP removes low-value or negative activations and only considers the activations with high response. This method is capable of preserving important information in the pooled features and significantly boosts the discriminating capability of the resultant features for image classification.

3.2. Rank-based weighted pooling

Average Pooling assigns the same weight of \( 1/n \) to each activation in the pooling region. However, it is obvious that each region in an image may not be equally important because image features are highly spatially non-stationary. A rank-based weighted pooling may easily remedy this problem by assigning larger weights to higher activations. The ranking method can be a linear function or a nonlinear function. In this paper, we introduce the following exponential ranking method (Michalewicz, 2013):

\[
p_r = \alpha(1 - \alpha)^{r-1}, \quad r = 1, \ldots, n
\]

where \( \alpha \) is a hyper-parameter, \( r \) represents the rank of activations, and \( n \) denotes the size of pooling region. It is easy to prove

\[
\sum_{r=1}^{n} p_r = 1 - (1 - \alpha)^n
\]

when \( 0 < \alpha < 1 \), it can be guaranteed:

\[
\lim_{n \to +\infty} \sum_{r=1}^{n} p_r = 1.
\]

Then, activations in each region are weighted by the probability \( p_i \) (see Eq. (4)) and summed:

\[
s_j = \sum_{i \in R_j} p_i a_i.
\]

RWP assigns a reasonable weight value to each activation in the pooling region, which often significantly improves the performance.

3.3. Rank-based stochastic pooling

The motivation behind rank-based stochastic pooling is the scale problems encountered by value-based stochastic pooling. The scale problems lie in three aspects: (1) This method is only applicable to the non-negative activations, which is too strict for applications. (2) The probability of activations, which is computed by normalizing the activation values, is out-of-control. (3) When training data is limited, strong activations dominate updating process and lead to over-fitting. (Cai, Shi, & Liu, 2014)

RSP computes the probabilities \( p \) of activations by Eq. (4). Then, we sample from the multinomial distribution based on \( p \) to pick an activation \( a_i \):

\[
s_j = a_i, \quad \text{where } i \sim \text{Multinomial}(p_1, \ldots, p_n).
\]

Actually, RSP can be also viewed as an instance of weighted pooling since random sampling result can be used as the weights of activations. Two special cases of RSP are notable. \( \alpha = 1 \) corresponds to a simple max pooling, whereas \( \alpha < 1 \) corresponds to stochastic pooling.

During the backward-propagation, gradients are propagated through the selected activations. The model parameters corresponding to the selected activations are updated, while others are kept frozen. The stochasticity introduced by RSP helps improve performance.

RSP training is similar to model averaging, where many different models are trained on different subsets of the data. However, the difference lies in that each model is trained for only one step and all of the models share parameters. Akin to dropout (Srivastava, 2014; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) and dropconnect (Wan, Zeiler, Zhang, Cun, & Fergus, 2013), this idea is widely accepted in deep learning community. A new model may be created during RSP training phase as the sampling mechanism can produce new connections structure. During testing, RWP, in an effective and implicit manner, provides an estimate of model averaging without explicitly instantiating them.

3.3.1. Solving scale problems

Stochastic pooling randomly picks the activations within each pooling region based on a probability distribution. Value-based stochastic pooling computes the probability \( p \) by normalizing activation values for each region \( j \):

\[
p_i = \frac{a_i}{\sum_{k \in R_j} a_k}.
\]

It is intuitively clear that the formulation requires nonnegative elements. Hence, VSP is only applicable to ReLU, in which negative responses are changed to zeros. The probabilities of some activations are usually overwhelmed by their “significantly larger” counterparts. During training, maximum activation values may exist in different regions and sometimes the probability of maximum activation may approach the value of 1. In this case, VSP reduces to max pooling. RSP solves the scale problem by using the rank of activation. In this work, Eq. (4) is used to map the rank of activation into probability. This function offers twofold benefits. In addition, it is a monotone function of the rank \( r \) (the probability distribute can be seen in Fig. 3), viz., higher activation functions get larger probabilities/weights. On the other hand, this function is conceptually simple. It depends only on \( \alpha \). By setting \( \alpha \) to a proper value (e.g., 0.5), we can ensure that the probability of the maximum activation is approximate equivalent to the sum of the probability of the other activations in a pooling region. In this case, RSP has more randomness when picking an activation, which helps to further control over-fitting. More randomness leads to strong diversity of model average at test time, which again makes the model more robust (Breiman, 1996).

3.4. Discriminative analysis

The capacity for preserving discriminative information plays a central role in pooling. Diverse information and salient information
dominate the discriminant ability of the pooled features. Max pooling preserves the most salient local feature containing discriminative information for image classification. This operation enhances the high frequency components which are retained for the following steps (Wang, 2014). Hence it is more suitable for simple images, as demonstrated in Fig. 4(a), where target objects are more likely to be the high frequency regions. However, for complex images, in which the high frequency regions of image are more likely to be clustered background or irrelevant objects as shown in Fig. 4(b), simple shape and less textured objects in the foreground may gradually disappear in clustered background after several max pooling operations. Therefore, the resulting feature vector may be lacking in presenting foreground objects, and then degenerates the overall recognition performance. Average pooling, on the other hand, preserves diverse information by combining information from several elements. However, this operation may have poor performance on some tasks since the salient information is always coupled by non-salient ones.

Based on the intuitive argument above, we argue that a good tradeoff between salient information and diverse information can boost the discriminating capability of the resultant features for image classification. In this study, Shannon entropy is introduced to measure the tradeoff. Considering pooled activation $s_j$ as a random variable $S$ with $n$ possible values ($n$ is the pooling size) \{a_1, a_2, a_3, \ldots, a_n\} and the corresponding probabilities are given by \{p_1, p_2, p_3, \ldots, p_n\}.

**Definition.** The entropy $H(s)$ of pooled activation $S$ is defined by

$$H(s) = - \sum_{i=1}^{n} p_i \log_2 p_i.$$  \hspace{1cm} (10)

We follow the convention of $0 \log_2 0 = 0$, which is consistent with the limit: $\lim_{p \to 0} p \log_2 p = 0$. Thus, adding terms of zero probability does not change the entropy.

Note that $H(s)$ is a functional of the distribution of $S$, which depends only on the probabilities. Max pooling assigns a probability of 1 to the strongest activation and a probability of 0 to others. The value of $H(s)$ is 0 in max pooling, and there is no uncertainty. Hence, pooled activation obtained by max pooling contains only salient information. RSP computes the probabilities $p$ for each pooling region by Eq. (4). The value of $H(s)$ is $- \sum_{i=1}^{n} \alpha(1 - \alpha)^{i-1} \log \alpha(1 - \alpha)^{i-1}$, and there is more uncertainty. This means that more diverse information is preserved. Since the probability of activation cannot be pre-determined in VSP, the value of $H(s)$ is 0 as the probability of maximum activation is significantly larger than others, and $\log n$ when activations have the same probability. The same analysis is applicable to weighted pooling. The weight in weighted pooling can be regarded as probability. In this case, the uniform distribution in average pooling has the maximum entropy of $\log n$ and makes the pooled activation values contain the strongest uncertainty or, in other words, most diverse information. The nonuniform distribution has smaller entropy than the uniform one. The value of $H(s)$ of RAP and RWP are $\log t$ and $- \sum_{i=1}^{n} \alpha(1 - \alpha)^{i-1} \log \alpha(1 - \alpha)^{i-1}$, respectively, and they are both between 0 and $\log n$. VWP has the same problem as VSP. To better compare these $H(s)$, let $n = 8$ and $t = 4$ and $\alpha = 0.5$. The calculation results are shown in Table 1. The values of $H(s)$ obtained by the proposed methods are more close to the median. From the above analysis, we can see that the proposed pooling methods provide good tradeoff, and thus have better discriminant ability than other pooling methods.

### 4. Experiments

#### 4.1. Overview

We evaluate the proposed methods on four benchmark datasets: MNIST, CIFAR-10, CIFAR-100 and NORB. The networks used for these datasets all consist of three convolutional layers with 5 × 5 filters and 64 feature maps. Pooling layers follow all three convolutional layers. We use $3 \times 3$ pooling with stride 2 for each of the three pooling layers. Additionally, after the first two pooling

Fig. 4. Simple images and complex images. In general, target objects are more likely to be the high frequency regions in simple images. However, for complex images, the high frequency regions of image are more likely to be clustered background or irrelevant objects.

<table>
<thead>
<tr>
<th>Pooling methods</th>
<th>$H(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average pooling</td>
<td>3</td>
</tr>
<tr>
<td>Max pooling</td>
<td>0</td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>2</td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>1.96</td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>[0, 3]</td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>[0, 3]</td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>1.96</td>
</tr>
</tbody>
</table>
layers there is a response normalization layer (as in Hinton et al., 2012) aside from one additional model in Section 4.3. Dropout is applied on the single fully-connected layer as a regularizer. The overall structure can be seen in Fig. 5.

All experiments are conducted by using a deep learning framework called Caffe (Jia et al., 2014). The proposed method is simple and can be easily implemented. We use a single NVIDIA TESLA C2075 GPU to run the experiments. The networks are trained using SGD with mini-batch size of 128 at a fixed constant momentum value of 0.9. Momentum term is used for faster convergence. The loss function of our model is defined as follows:

\[ E = - \sum_t \sum_k x_t^k \log(y_t^k) \]  

(11)

where \( y_t^k \) and \( x_t^k \) are the target and predicted values of the \( t \)th training example at \( k \)th class, respectively. We use weight decay with a fixed value of 0.01 as a regularizer. The update rule for \( W_{ij}^l \) is

\[ W_{ij}^l = W_{ij}^l + \Delta W_{ij}^l \]  

(12)

\[ \Delta W_{ij}^l = \text{momentum} \cdot \Delta W_{ij}^l - \text{decay} \cdot \epsilon \cdot W_{ij}^l - \epsilon \cdot \frac{\partial E}{\partial W_{ij}^l} \]  

(13)

where \( W_{ij}^l \) represents the weight between \( X_{ij}^{l-1} \) and \( X_{ij}^l \), \( \partial W_{ij}^l \) denotes the gradient of \( W_{ij}^l \), and \( \epsilon \) denotes the learning rate. Initial value for learning rate is 0.001, and then the learning rate is annealed during training by a factor of 10 each 100 epochs. The training epoch is taken as 250.

4.2. CIFAR-10

The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) is composed of 10 classes of natural images split into 50,000 train images and 10,000 test images. Each image is a RGB image of size 32 × 32. For this dataset, we follow Hinton et al.’s (Hinton et al., 2012) approach of subtracting the per-pixel mean computed over the dataset from each image. We do not use data augmentation on the dataset.

Using the same network architecture described above, we trained several models and compared their performances. We selected the hyper-parameters by minimizing the error on a validation set consisting of the last 10,000 training samples. Then keeping all hyper-parameters fixed, we continued training on the full 50,000 training samples for the final evaluation. The optimal hyper-parameters found are included in the following list:

- LReLU \( a = 0.33 \)
- PReLU channel-shared version and \( a_i = 0.25 \) as the initialization.
- RWP \( a = 0.4 \)

1 The codes can be available on the web at http://www5.zzu.edu.cn/mlis/singlePageDir/download.html.

The test performances of these models are shown in Table 2. Rank-based average pooling obtains a result of 18.5%, which improves more than one percent compared to the results obtained by max pooling and average pooling. Rank-based weighted pooling obtains the better performance than value-based weighted pooling and average pooling, which demonstrates that the proposed method provides more reasonable weights. Due to the stochastic property introduced by stochastic pooling, RSP and VSP achieve a better performance than RAP and RWP. Rank-based stochastic pooling outperforms value-based stochastic pooling, and the best result of 13.84% is obtained when it is combined with LReLU.

One obvious question is whether we obtain better performance by using new generalizations of ReLU, rather than by the proposed methods. By this experiment, we can see that the proposed methods offer a clear improvement over other pooling approaches with the same activation function. When using LReLU and PReLU, the proposed methods have better performance than max pooling and average pooling. This means that the proposed pooling is more suitable for these novel generalizations.

4.3. MNIST

The MNIST handwritten digit classification task (LeCun et al., 1998) consists of 28 × 28 pixel grayscale images, and each containing a digit 0–9 (10-classes). There are 60,000 training images and 10,000 test images in total. Without extra preprocessing, the image pixels are only divided by 255 so that they are in the range [0, 1]. Rank-based stochastic pooling with PReLU surpasses the result of 0.44% obtained by value-based stochastic pooling, achieving an error rate of 0.42%. We beat the previous state-of-art result of 0.45%. RAP and RWP achieve comparable but not better performance than max pooling and VWP. There may be two reasons for this phenomenon. On the one hand, MNIST has been tuned to a very low error rate, and even a little improvement is not easy to be made. On the other hand, grayscale images provide a straightforward ordering of values from small to large that is intuitively meaningful, so capturing the strongest activation may be a more appropriate choice. Details of the performance comparison can be seen in Table 3.

---

**Table 2** Test set error rates (%) for CIFAR-10 of various methods.

<table>
<thead>
<tr>
<th>Pooling Method</th>
<th>Rectifiers</th>
<th>Average pooling</th>
<th>LReLU</th>
<th>PReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average pooling</td>
<td>19.86</td>
<td>20.04</td>
<td>20.97</td>
<td></td>
</tr>
<tr>
<td>Max pooling</td>
<td>20.47</td>
<td>20.68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>18.68</td>
<td>17.97</td>
<td>18.52</td>
<td></td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>20.41</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>19.05</td>
<td>19.92</td>
<td>18.91</td>
<td></td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>16.96</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>15.44</td>
<td>13.84</td>
<td>14.90</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3** Test set error rates (%) for MNIST of various methods.

<table>
<thead>
<tr>
<th>Pooling Method</th>
<th>Rectifiers</th>
<th>Average pooling</th>
<th>LReLU</th>
<th>PReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average pooling</td>
<td>12.78</td>
<td>13.01</td>
<td>12.97</td>
<td></td>
</tr>
<tr>
<td>Max pooling</td>
<td>12.16</td>
<td>12.38</td>
<td>12.30</td>
<td></td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>11.52</td>
<td>11.28</td>
<td>11.30</td>
<td></td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>12.84</td>
<td>12.50</td>
<td>12.56</td>
<td></td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>11.10</td>
<td>11.28</td>
<td>11.30</td>
<td></td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>10.80</td>
<td>10.50</td>
<td>10.56</td>
<td></td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>10.20</td>
<td>10.00</td>
<td>10.04</td>
<td></td>
</tr>
</tbody>
</table>
Fig. 6. The structure of Network In Network. In this paper, we replace the first two pooling layer with the proposed pooling methods (for example, rank-based stochastic pooling).

### Table 3

<table>
<thead>
<tr>
<th>Pooling</th>
<th>Rectifiers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ReLU</td>
<td>LReLU</td>
<td>PReLU</td>
</tr>
<tr>
<td>Average pooling</td>
<td>0.72</td>
<td>0.89</td>
<td>0.72</td>
</tr>
<tr>
<td>Max pooling</td>
<td>0.49</td>
<td>0.48</td>
<td>0.5</td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>0.56</td>
<td>0.59</td>
<td>0.58</td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>0.48</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>0.50</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>0.44</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>0.5</td>
<td>0.45</td>
<td><strong>0.42</strong></td>
</tr>
<tr>
<td>Network in network [Lin et al., 2014]</td>
<td>0.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maxout network [Goodfellow et al., 2013]</td>
<td>0.45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4

<table>
<thead>
<tr>
<th>Pooling</th>
<th>Rectifiers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ReLU</td>
<td>LReLU</td>
<td>PReLU</td>
</tr>
<tr>
<td>Average pooling</td>
<td>51.62</td>
<td>51.93</td>
<td>50.14</td>
</tr>
<tr>
<td>Max pooling</td>
<td>50.82</td>
<td>51.80</td>
<td>49.76</td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>46.22</td>
<td>45.66</td>
<td>45.82</td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>49.87</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>48.19</td>
<td>46.69</td>
<td>47.05</td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>46.78</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>46.83</td>
<td><strong>43.91</strong></td>
<td>44.79</td>
</tr>
</tbody>
</table>

### Table 5

<table>
<thead>
<tr>
<th>Pooling</th>
<th>Rectifiers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ReLU</td>
<td>LReLU</td>
<td>PReLU</td>
</tr>
<tr>
<td>Average pooling</td>
<td>5.30</td>
<td>5.85</td>
<td>5.68</td>
</tr>
<tr>
<td>Max pooling</td>
<td>5.88</td>
<td>5.45</td>
<td>5.92</td>
</tr>
<tr>
<td>Rank-based average pooling</td>
<td>4.71</td>
<td>4.34</td>
<td>4.57</td>
</tr>
<tr>
<td>Value-based weighted pooling</td>
<td>5.44</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based weighted pooling</td>
<td>5.20</td>
<td>5.04</td>
<td>4.97</td>
</tr>
<tr>
<td>Value-based stochastic pooling</td>
<td>5.03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rank-based stochastic pooling</td>
<td>4.02</td>
<td>3.72</td>
<td><strong>3.50</strong></td>
</tr>
</tbody>
</table>

4.4. CIFAR-100

The CIFAR-100 dataset [Krizhevsky & Hinton, 2009] has the same size and format as the CIFAR-10 dataset, but it contains 100 classes with only one tenth as many labeled examples per class. For CIFAR-100, we found a $\alpha$ of 0.5 to be optimal, and other settings are the same as the CIFAR-10 dataset. In agreement with the CIFAR-10 results, our models have a significant improvement on classification performance. Details of the performance comparison are shown in Table 4.

4.5. NORB

In the final experiment, we evaluate our methods on NORB (jittered–cluttered) dataset [LeCun, Huang & Bottou, 2004], which is a collection of stereo images of 3D models. For each image, one of 6 classes appears on a random background. The objects are centrally placed on randomly chosen backgrounds, and there is also cluttering from a peripherally placed second object. The training set has 10 folds of 29160 images each for a total of 291600 images; the testing set consists of two folds totaling 58320 images. We train on the first two folds and test on the entire testing data. No preprocessing is used for this dataset. The images only are down-sampled from 108 to 48 as in Ciresan, Meier, and Schmidhuber (2012). The same network as in Section 4.4 is used for this experiment (adapted to classify 6 classes). Table 5 shows the classification results. A test error of 3.50% is obtained by RSP with PReLU, which surpasses other methods at least one percent. We beat the previous state-of-art result of 3.57%. Although our result is slightly lower than DropConnect, we should note that DropConnect obtains the result of 3.23% by using data augmentation and ensemble method (5-net), and these techniques are not used in our models.

4.6. Connection with other state-of-the-art solution

Actually our work is orthogonal to many other state-of-the-art solutions. In this section, we show it is easy to further improve their results. In the present work, we embark on Network-in-Network (NIN) [Lin et al., 2014], which is shown to perform well on many tasks. We choose CIFAR-10 and CIFAR–100 classification task to demonstrate our point. NIN is an approach proposed in order to increase the representational power of neural networks. In NIN, the generalized linear convolution filter is replaced with a micro network structure which is a general nonlinear function approximator. The multi-layer perceptron [Silva, de Sá & Alexandre, 2008] is used as the instantiation of the micro network. Instead of adding fully connected layers on top of the feature maps, it takes the average of each feature map via a global average pooling layer, and the resulting vector is fed directly into the softmax layer. Global average pooling can be regarded as a structural regularizer that explicitly enforces feature maps to be confidence maps of categories.

Using the same network architecture as NIN on CIFAR datasets, we only replace the first two pooling layers with the proposed pooling methods. Global average pooling is still used on the top of the network. Fig. 6 illustrates the overall structure of network used in this section. We obtain a test error of 8.67% on CIFAR-10 dataset without data augmentation, which improves more than
Tricks such as sparse network and ensemble method, not used obtained the best results, but should note that FMP used more tricks such as sparse network and ensemble method, not used.

7. We can see that Fractional max-pooling (FMP) (Graham, 2015) obtained the best results, but should note that FMP used more tricks such as sparse network and ensemble method, not used.

in our models. Moreover, our method is not in competition with FMP and can be easily combined with FMP. The experimental results demonstrate that the proposed pooling methods can be incorporated with NIN very well. However, they are not confined to the Network-in-Network model; it is capable of being incorporated into many existing convolutional networks.

4.7. Hyper-parameter selection

In RSP, we introduce a new hyper-parameter $\alpha$. It controls the probability of the maximum activation in a pooling region. We argue that setting it to be around 0.5 may lead to satisfactory performance, which makes RSP have more randomness when picking an activation. We trained a series of models to verify our analysis. Using the same network architecture described above, the models use RSP with various activation functions and $\alpha$. The activation functions include ReLU, LReLU, and PReLU. We first test three values of $\alpha$: 0.4, 0.5 and 0.6 with all them. Details of the performance comparison are shown in Fig. 7(a). The optimal $\alpha$ appears to be 0.4, although the best result of 13.84 is obtained when 0.5 is combined with LReLU. In fact, we argue optimal $\alpha$ should be between 0.4 and 0.5. Fig. 7(a) shows that $\alpha = 0.4$ consistently performs better than 0.5 and 0.6, despite of using ReLU, or LReLU, or PReLU. Thus, we only provide the classification performance of other choice of $\alpha \in \{0.1, 0.2, 0.3, 0.7, 0.8, 0.9\}$ with ReLU. They all give significantly worse performance than $\alpha = 0.4$ as shown in Fig. 7(b). We can see that RSP gradually reduces to max pooling when increasing $\alpha$ to a higher value for all $\alpha > 0.4$.

RAP uses the parameter $t$ to control how many activations are selected to be averaged. We suggest that setting $t$ at medium ranges (i.e., median value) may lead to satisfactory result. Empirical justifications are provided in this section. Using the same network architecture described above and keeping other hyper-parameters fixed, we validate the choice of the parameters $t \in \{0.4, 0.5, 0.6\}$ in RAP.
Fig. 9. CIFAR-10 Classification performance for various pooling methods. (a) Combining max pooling with average pooling. (b) Combining value-based stochastic pooling with average pooling. (c) Combining rank-based stochastic pooling with average pooling.

4.8. Preserving diverse information

We argue that pooling operation should preserve salient information without losing diverse information. The proposed pooling is highly capable of preserving diverse information, which is analyzed in Section 3.4. To further verify our hypothesis, we trained three models on CIFAR-10. The basic architecture of models is the same as described above. We use different pooling operations in the first two pooling layers of three models, and all of the last pooling layers perform average pooling in order to preserve diverse information.

Fig. 9(a) shows that combining max pooling with average pooling has improved performance, and the test error is reduced from 20% to 18.44%. At the beginning of training, max pooling converges fast, but its performance eventually is the same as average pooling. Based on the experimental result, we can see that preserving diverse information is as important as preserving salient information. Fig. 9(b) shows that combining value-based stochastic pooling with average pooling has also yielded a little improvement, and the test error is reduced from 16.96% to 16.86%. This suggests that preserving diverse information can boost classification performance. Fig. 9(c) shows that the proposed pooling has poor performance when it is combined with average pooling, and the test error is increased from 15.44% to 16.46%. The experimental results indicate that the proposed method provides a good tradeoff between salient information and diverse information.

4.9. Visualizations

The discriminant ability of features learned by various pooling methods is different. We desire that pooling operation preserves task-related information while removing irrelevant details. To see what features have been learned by different pooling methods and analyze the effectiveness of the proposed pooling method from the feature level, we extract and directly visualize the feature maps from the first pooling layer of the trained model for CIFAR-10.

We select one example image from each of the ten categories of CIFAR-10 test set, following Lin (Lin et al., 2014) and Chenyu (Lee, Xie, Gallagher, Zhang, & Tu, 2014), run one forward pass, then show the feature maps learned from the first pooling layer in Fig. 10. Feature maps learned by RSP preserve more task-related information and more appropriate frequencies than those learned by other pooling methods. This means that the proposed method can learn more discriminative features.

5. Conclusions and future work

In this work, we propose a novel pooling mechanism called rank-based pooling. RAP, RWP and RSP are introduced to show the advantages of this pooling mechanism. In these methods, the reasonable usage of rank resolves some problems encountered by
conventional pooling methods. From the perspective of entropy, we can see that the proposed methods provide a good tradeoff between diverse information and salient information, which can boost the discriminating capability of the resultant features for image classification. Then, experimental results show that the proposed methods are superior to existing pooling methods on a range of datasets. We also show that the proposed pooling methods can be incorporated with Network-in-Network model very well.

There are several research directions for future work. First, in this paper, the hyper-parameter \( \alpha \) needs to be predefined, so we plan to develop a learned parametric RSP and RWP in the future. Second, we also intend to come up with new activation function, which is more suitable for the proposed pooling approaches. Third, in RAP, \( t \) is fixed in the whole training process. We argue that it can be automatically adjusted according to the value distribution of activations.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 6170223, the National Natural Science Foundation of China under Grant No. 61502432 and the National Natural Science Foundation of China under Grant No. 61502434.

References


