FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 44 (2019) No. 3
ISSN 0867-6356
DOI: 10.2478/fcds-2019-0016 e-ISSN 2300-3405

Implications of Pooling Strategies in Convolutional Neural Networks: A Deep Insight

Shallu Sharma*, Rajesh Mehra

Abstract. Convolutional neural networks (CNN) is a contemporary technique for computer vision applications, where pooling implies as an integral part of the deep CNN. Besides, pooling provides the ability to learn invariant features and also acts as a regularizer to further reduce the problem of overfitting. Additionally, the pooling techniques significantly reduce the computational cost and training time of networks which are equally important to consider. Here, the performances of pooling strategies on different datasets are analyzed and discussed qualitatively. This study presents a detailed review of the conventional and the latest strategies which would help in appraising the readers with the upsides and downsides of each strategy. Also, we have identified four fundamental factors namely network architecture, activation function, overlapping and regularization approaches which immensely affect the performance of pooling operations. It is believed that this work would help in extending the scope of understanding the significance of CNN along with pooling regimes for solving computer vision problems.

Keywords: Pooling strategies, convolutional neural network, visual recognition, regularization, overfitting.

1. Introduction

Computational models of neural network have evolved more than half a century. McCulloch and Pitts developed the very first model in 1943 which is known as linear threshold gate [33, 37, 52]. To train such neural models, Hebbian contributed a learning algorithm known as the Hebbian learning rule. The Hebbian rule performs well only when all the input patterns are orthogonal [20]. The presence of orthogonality in input patterns is a serious demand for a good performance of this rule [40]. In order to surpass this demand, a more powerful learning rule, i.e. Delta rule came into existence. However, Delta rule is

*National Institute of Technical Teachers' Training and Research (NITTTR), Chandigarh-160019, India, e-mail: shallu.ece@nitttrchd.ac.in

unable to solve the problems that are not linearly separable [10]. To overcome the entire problem associated with the above-discussed learning rules, backpropagation emerged as a more complex learning algorithm. Backpropagation having the ability to learn many layers arbitrarily and can approximate any computable function [10, 39]. Backpropagation is a conventional method to train a Feed-Forward Neural Network (FFNN). The simplest FFNN composed of at least one hidden layer sandwiched between an input and output layer. In an FFNN, each pair of neurons has an acyclic and directed connection between each other as depicted in Figure 1.

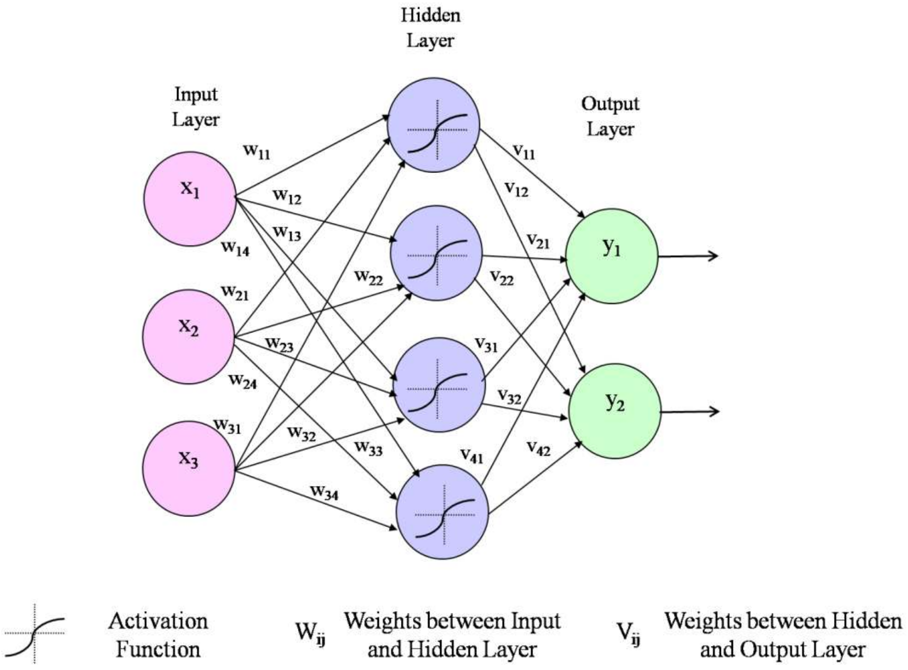


Figure 1. Architecture of Feed Forward Neural Network (FFNN)

In multilayer FFNN, a weighted summation process is used to specify the flow wherein every layer is fully connected to the next layer. So that, each neuron can be capable enough to send its current activation to any connected unit. The transmitted activation is multiplied by the weight of the connection and at the receiving neuron moved through some squashing function (like sigmoid, ReLU, tan h) in order to introduce nonlinearities [15, 51]. The learning is performed by updating the weights to minimize an error function which defined as the difference between desired and the actual output activation vector [22]. Usually, the backpropagation algorithm is used to accomplish the task of learning by taking a partial derivative of the error with respect to the weights of the last layer and then used to modify the weights. Similarly, the partial derivative is computed for the errors with respect to the weights of the second to last layer, and this process is repeated until all the weights connected to the input layer get updated. In spite of being a universal function approximator, FFNNs are poor in dealing with many forms of practical problems such as object and face recognition [5, 34-35, 49, 57]. The reason is full connectivity of the network due to which the number of weights grows rapidly with the input dimension. In addition, the disconcerting fact about the FFNN is their spatial ignorance [9], because separate weights are involved in learning the same object at different location instead of weight transferring. All this happens because every pair of neurons between two layers has their weights. In order to eradicate the problems related with the implemenation of FFNN, the convolutional neural networks (CNNs) are then come into existence. CNNs can exploit the two dimensional spatial constraints imposed by the input modality and at the same time reduce the number of parameters involved in the training process.

1.1 General Convolutional Network Architecture

CNNs are a special class of deep network which has been applied successfully to the data with grid-like topology (image data and time-series data) [26-27, 42, 44]. CNN mainly composed of convolutional, pooling, activation and fully connected layer. The convolutional layer is the integral part of a CNN, where the convolution operation is applied to the input. Convolution operation leverages the three important ideas named as equivalent representations, sparse connectivity and tied weights which play an essential role in the improvement of machine learning systems specifically to solve the tasks related to computer vision [43]. Since the architecture of CNN consists of multiple convolutional layers so as for images. In this context, a bank of filters is applied to an image (at every convolutional layer), and the output is obtained in a piled manner. The piling of outputs increases the abstract feature and makes the pixel-wise analysis more complicated. In order to alleviate this complexity, pooling layers are inserted after the convolutional layers. A schematic representation of the most commonly used CNN architecture with a pooling operation is depicted in Figure 2 with an explanation.

1.1.1 Pooling

Pooling is a non-linear transformation that permits to summarize the output of a net at a certain location with a single value. This single value is obtained from the statistic of the neighboring outputs which makes the feature descriptions more robust and invariant to small translations of the input data [27, 53]. The pooling layer progressively cuts down the dimensionality of the input, consequently reduces the demand of memory for storing the parameters, and improves the statistical efficiency [3, 16-18, 25, 28-30, 41, 53-54]. In deep networks, over-fitting is also controlled with the application of the pooling layers [16, 54]. Numerous application fields of CNN such as visual recognition, tracking, object detection, and face recognition have been used pooling operation to create invariance to small shifts and distortions of the input data. Average, max, fractional-max out, stochastic pooling and mixed pooling are some popular pooling regimes which have been used in numerous variant of CNN to solve the problems related to computer vision. Not only CNN, the Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradient (HOG) methods of feature extraction also utilize pooling regimes to design a robust object detection system that can firmly recognize the objects in clutter and occlusion [6, 32].

In general, choice of pooling regime matters a lot in solving the computer vision problem as its foremost objective is to convert the joint feature representation (mapped by convolution) into useable one. Thus, the pooling operation plays a significant role in various computer vision architectures. Pooling operation also reduces computational cost as it cut down the resolution of feature map while preserving the useful features required for the task to perform. In short, the results desired from pooling operation are compact representation, robustness to noise and clutter, and invariance to shift, skew or lightening condition. Many research communities are working in the direction of development of advanced pooling mechanisms to make efficient use of these pivotal features of pooling regimes. The motivation behind this study is to appraise the readers from the regular advancements in the development of pooling regimes. Additionally, the survey provides a common platform to discuss all the popular pooling schemes with their pros and cons.

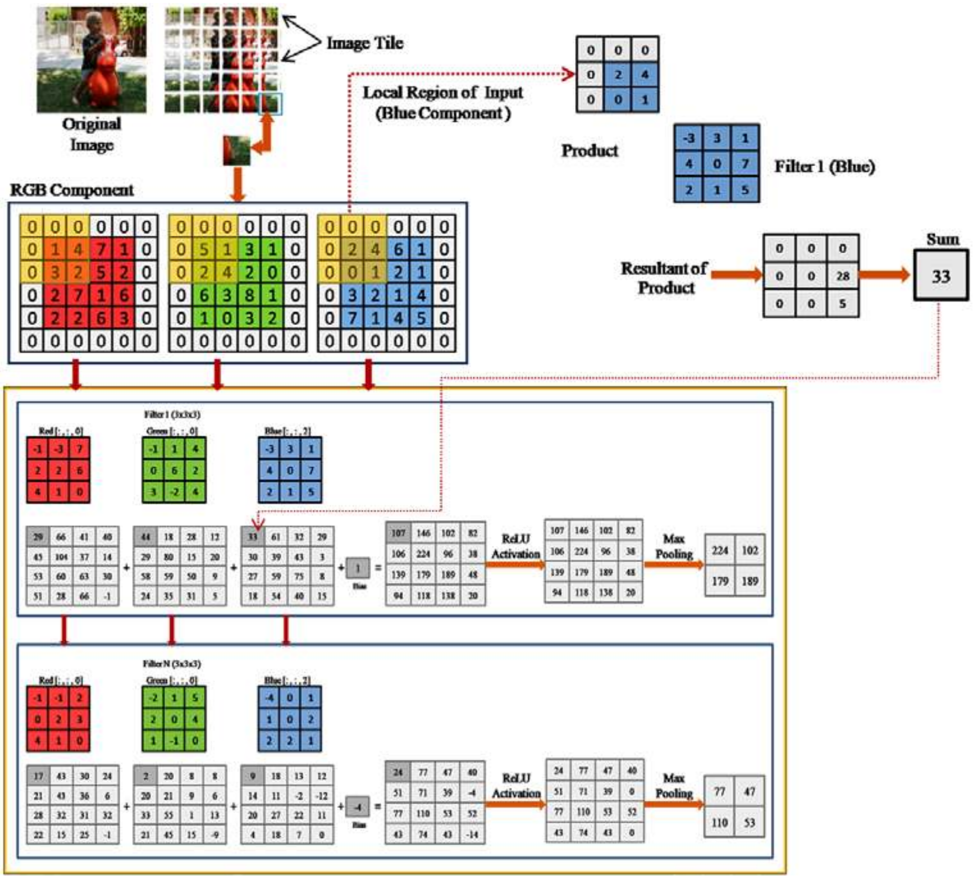


Figure 2. Illustration of CNN architecture. The image is break down in overlapped tiles by using a sliding window approach, where each pixel of the image is represented by a number value for its three channels: red, green and blue. Zero padding is applied to each tile in order to maintain the size of final output. To perform a convolution operation, image tiles are fed to a CNN which is composed of N filters with size (3x3x3) and the filter is slid over

each channel of the image tile by considering a local region of size 3x3. A dot product is computed between the local region of image tile and the filter, which results in single value. This value is then put down in the left most cell of the local region. The filter is then moved with stride 1 to the right and looks at the next local region. After filling the cells, the values of all the three channels are summed up which represents the contribution of this filter for the final output and known as feature map. Similarly, the remaining filters also provide the feature maps as their contribution to the output of CNN. An activation function (ReLU) is then applied to introduce non-linearity in the network, where negative values in the feature maps are replaced by '0'. After activation, max-pooling operation is performed to obtain the feature map with reduced dimensionality by considering the highest value from each window of size 2x2.

2. Pooling Regimes

In this section, the pooling regimes are discussed that are important and applied to several computer visions related tasks. Few popular pooling regimes are illustrated with the help of schematic representations for better understanding and the important characteristics are summarized in Table 1.

2.1 Average Pooling

In this pooling strategy, the input image is partitioned into a set of the disjoint rectangular box. The output for each rectangle is the average of entries in that box [28-29]. A pictorial representation of average pooling is depicted in Figure 3. Mathematically, the average pooling can be expressed as [26]:

$$f_{ave}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

where $\boldsymbol{x}$ is a vector consist of activation values from a rectangular area of $\boldsymbol{N}$ pels (for example: dimensions of the rectangular area in Figure 3 is 2x2) in an image or a channel. Earlier, the use of average pooling was common but their use has been limited with the advent of max pooling operation [3]. The loss of informations in term of contrast is the major rationale behind their failure. In the computation of mean, all the activation values which are present in rectangular box are considered. If the magnitude of all the activations is low, the computed mean would also be low and give rise to reduced contrast. The situation will be worst when the most of the activations in the pooling area come with a zero value. In that case, feature map characteristic would reduce by a large amount.

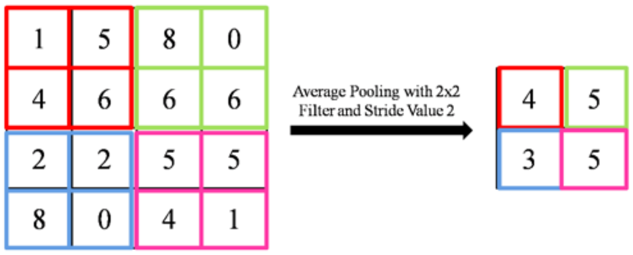


Figure 3. Illustration of average pooling with a pooling area of size 2x2 and stride of 2.

2.2 Max Pooling

In this pooling strategy, activation with the maximum value is selected from all the activations that present in a rectangular field, as shown in Figure 4. This regime is widely applied in most of the architecture which are similar to CNN's [16, 30, 41]. Max pooling strategy trims down the computation of upper layers with the elimination of non-maximal components. Max pooling provides a better performance with sparse coding and simple linear classifiers. Due to this reason, it has gained popularity in the past few years [3]. The statistical properties of max pooling make it considerably fit sparse representations, which is another appeasement for its fame. Mathematical expression for max-pooling is given as [30]:

$$f_{max}(x) = max_i(x_i) \tag{2}$$

The major drawback of max pooling is that the pooling operator considered only the maximum element from the pooling area and ignores other elements. If the majority of the elements in the pooling area would be of high magnitudes, the discerning features get disappeared after performing max pooling operation. As a matter of this fact, the situation leads to unacceptable results due to the loss of information.

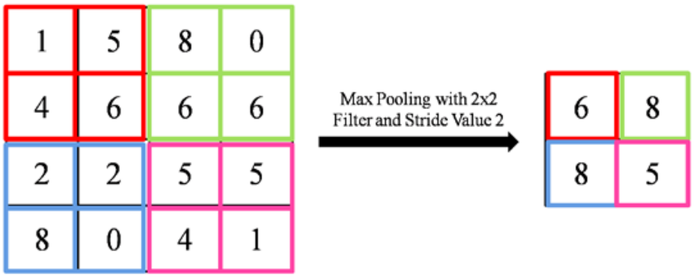


Figure 4. Illustration of max pooling with a pooling area of size 2x2 and stride of 2.

2.3 'Mixed' Max-Average Pooling

Mixed pooling is a blend of max and average pooling, so termed as max-average pooling. This pooling scheme is stochastic because of the random employment of max and average pooling during the training of CNN [30, 53]. Stochastic nature of the mixed pooling helps in prevention of over-fitting to some extent. In this regime, mixing proportion parameters (a) are learned in various fashions (for each net, for each layer, layer/region, layer/channel or layer/region/channel combinations) [30]. A graphical representation of this regime is illustrated in Figure 5(a). The mathematical expression for mixed pooling is [30]:

$$\mathrm{f}_{mix}(\mathrm{x}) = a * f_{max}(x) + (1-a) * f_{avg}(x) \tag{3}$$

where $a \in [0, 1]$ is a scalar factor which determine the blending of max and average pooling and termed as mixing proportion. In order to learn each mixing proportion automatically, we have to define the output loss function (E). For this learning, vanilla back-propagation is represented as [30]:

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial \mathrm{f}_{mix}(\mathrm{x})}\frac{\partial \mathrm{f}_{mix}(\mathrm{x})}{\partial a} = \delta\left(\max_{\mathrm{i}}(\mathrm{x_i}) - \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i}\right) \tag{4}$$

Note: To solve this, $\partial \mathrm{f}_{mix}(\mathrm{x})$ factor is multiplied in numerator and denominator. Later, partial differentiation is performed on $\frac{\partial E}{\partial \mathrm{f}_{mix}(\mathrm{x})}$ and $\frac{\partial \mathrm{f}_{mix}(\mathrm{x})}{\partial a}$ which correspond to δ (the error which is propagated back from the succeeding layers) and $\max_{\mathrm{i}}(\mathrm{x_i}) - \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i}$, respectively.

Moreover, it is necessary to compute the error signal that propagated back to former layer because pooling layers are generally placed in the midst of a deep network.

$$\frac{\partial E}{\partial \mathrm{x_i}} = \frac{\partial E}{\partial \mathrm{f}_{mix}(\mathrm{x_i})}\frac{\partial \mathrm{f}_{mix}(\mathrm{x_i})}{\partial \mathrm{x_i}} = \delta\left[\mathrm{a} * 1[\mathrm{x_i} = \max_{\mathrm{i}}(\mathrm{x_i}) + (1-a) * \frac{1}{\mathrm{N}}\right] \tag{5}$$

The mixed pooling strategy is insensitive towards the important features in the pooled area due to the fixed mixing proportion. Non-responsive behaviour is the major drawback of this pooling scheme. However, Lee et al. experimentally demonstrated a significant improvement in the classification performance of CNN with the mixed-pooling which is also compared with the max pooling and average pooling for the standard datasets namely-SVHN, MNIST, CIFAR10 and CIFAR100 [30]. The details are tabulated in Table 2.

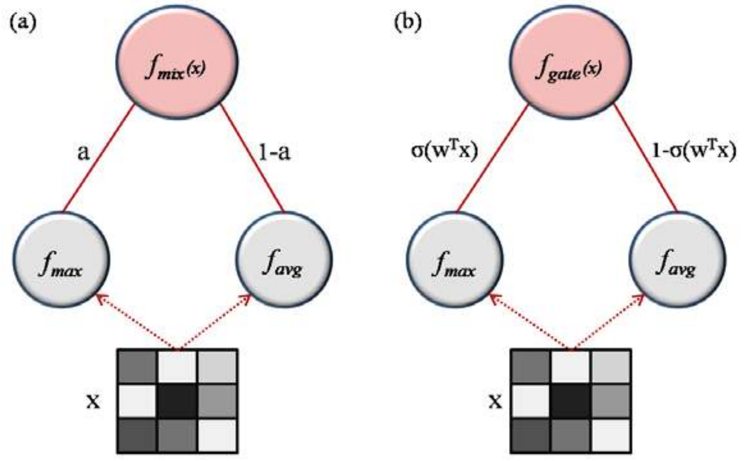


Figure 5. Mixed Pooling, (a) "Mixed" Max-Average Pooling where, 'a' is the mixing proportion (fixed, once learned). The range in which mixing proportion lie is 0 to 1. (b) "Gated" Max-Average Pooling where, 'w' is the gating mask, multiplied with the region being pooled. The resultant of product passed through a sigmoid 'σ' to get the value of mixing proportion (adaptive to pooling region's characteristics). Modified and regenerated from [30]

2.4 'Gated' Max-Average Pooling

'Gated' max-average pooling is similar to mixed max-average pooling as both originates from the combinations of max pooling and average pooling (Figure 5(b)). However, responsive nature of "gated" max-average pooling makes it different from the mixed max-average pooling. In this pooling strategy, a dot product is made between the "gating mask" and the "region being pooled", where both are having the same spatial dimension. The resultant scalar from this product is run through a sigmoid that yields a value known as mixing proportion. According to the strategy, the actual mixing proportion can adapt itself to the features that are present in the pooling region. Similar to "Mixed" max-average pooling, "Gated" max-average pooling strategy has the same options to learn mixing proportion parameters. The mathematical representation for gated max-average pooling is expressed beneath [30]:

$$\mathrm{f}_{gate}(\mathrm{x}) = \sigma(w^T x) * f_{max}(x) + (1 - \sigma(w^T x)) * f_{avg}(x) \tag{6}$$

The gradient concerning "gating mask" (w) can be computed by adopting the same procedure which is used in "Mixed" max-average pooling:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \mathrm{f}_{gate}(\mathrm{x})} \frac{\partial \mathrm{f}_{gate}(\mathrm{x})}{\partial w} = \delta \left[\sigma(w^T x) * (1 - \sigma(w^T x)) x \left(\max_{\mathrm{i}}(\mathrm{x_i}) - \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i} \right) \right]$$

$$\frac{\partial E}{\partial \mathrm{x_i}} = \frac{\partial E}{\partial \mathrm{f}_{gate}(\mathrm{x_i})} \frac{\partial \mathrm{f}_{gate}(\mathrm{x_i})}{\partial \mathrm{x_i}}$$

$$= \delta \left[\sigma(w^T x) * (1 - \sigma(w^T x)) w_i \left(\max_\mathrm{i}(\mathrm{x_i}) - \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i} \right) + \sigma(w^T x) * 1[\mathrm{x_i} = \max_\mathrm{i}(\mathrm{x_i}) + (1 - \sigma(w^T x) * \frac{1}{\mathrm{N}} \right] \tag{7}$$

The experimental work by Lee et al. is an evident that "Mixed" strategy involves fewer parameters as compared to the "Gated" strategy but still "Gated" network surpasses the "Mixed" option on MNIST, CIFAR10, and CIFAR100 except for the SVHN dataset [30] (Table 2).

2.5 Tree Pooling

In this pooling scheme, the learning of pooling operation is carried out in form of the values in pooling filters and further learning is performed to combine these learned filters responsively. Both the learning procedures are executed within a binary tree that consists of leaf nodes. In this tree-like representation, each leaf node (child node) is related to a "pooling filter" with an area of pooling ($x \in \mathbb{R}^N$) and denoted by ($v_m \in \mathbb{R}^N$), where m is an indexing for the node. The values of two child nodes are further combined into a parent node with single value. At each parent node, the mixture is learned responsively with learned "gating masks" and denoted by ($w_m \in \mathbb{R}^N$) which is similar to "gated max-average" pooling. The parent nodes are also known as internal nodes of a tree. Eventually, the parent values are combined into the root node value in a responsive manner to represent the overall output of the tree (Figure 6).

Purposely, the tree pooling is proposed to perform the following actions:

1. Pooling filters are learned directly from the input dataset.
2. It learns how to "blend" pooling filters in a differentiable manner.
3. The former characteristics are brought together inside a tree with hierarchal structure.

$$f_m(x) = \begin{cases} v_m^T x & If\ leaf\ node \\ \sigma(w_m^T x) f_{m,\ left}(x) + \left(1 - \sigma(w_m^T x)\right) f_{m,\ right}(x) & If\ internal\ node \end{cases} \tag{8}$$

Note: A tree pooling function with "2" leaf nodes and "3" internal nodes can be specified as: ($f_{tree}(x) = \sigma(w_3^T x) v_1^T x + (1 - \sigma(w_3^T x)) v_2^T x$). The chain rule is applied on the tree pooling function to compute the gradient with respect to leaf node pooling filters ($\mathrm{v_1}$ & $\mathrm{v_2}$) and internal node gating mask ($\mathrm{w_3}$), which is expressed as [30]:

$$\frac{\partial E}{\partial \mathrm{v_1}} = \frac{\partial E}{\partial \mathrm{f}_{tree}(\mathrm{x})} \frac{\partial \mathrm{f}_{tree}(\mathrm{x})}{\partial \mathrm{v_1}} = \delta[\sigma(w_3^T x) x] \tag{9}$$

$$\frac{\partial E}{\partial \mathrm{v_2}} = \frac{\partial E}{\partial \mathrm{f}_{tree}(\mathrm{x})} \frac{\partial \mathrm{f}_{tree}(\mathrm{x})}{\partial \mathrm{v_2}} = \delta[(1 - \sigma(w_3^T x)).x] \tag{10}$$

$$\frac{\partial E}{\partial \mathrm{w}_3} = \frac{\partial E}{\partial \mathrm{f}_{tree}(\mathrm{x})}\frac{\partial \mathrm{f}_{tree}(\mathrm{x})}{\partial \mathrm{w}_3} = \delta\left[\sigma(w_3^T x)\left(1-\sigma(w_3^T x)\right)x(v_1^T - v_2^T)x\right] \tag{11}$$

Whereas, the error signal back-propagated to the former layer is given as:

$$\frac{\partial E}{\partial \mathrm{x}} = \frac{\partial E}{\partial \mathrm{f}_{tree}(\mathrm{x})}\frac{\partial \mathrm{f}_{tree}(\mathrm{x})}{\partial x} = \delta\left[\sigma(w_3^T x)\left(1-\sigma(w_3^T x)\right)\mathrm{w}_3(v_1^T - v_2^T)x + \sigma(w_3^T x)\mathrm{v}_1 + (1 - \sigma(w_3^T x))\mathrm{v}_2\right] \tag{12}$$

The tree pooling is different from the traditional decision tree in which decisions makes by the tree are "hard decisions". The hard decision functions are neither differentiable nor continuous with respect to its input as the mixing proportion can take value either 0 or 1. This makes the traditional decision tree non-useful for backpropagation in every prospect. Due to all these reasons, sigmoid "gate" function (i.e., $\sigma(w_m^T x) \in [0, 1]$) is used by the internal node in tree pooling; which makes tree pooling differentiable with respect to inputs as well as its parameters. Lee et al. [30] have observed through their experiment that the tree pooling is useful at the lower network layers where the feature responses are dense. However, the "mixed" and "gated" max-average pooling are more advantageous where the feature responses are sparse which present at the higher network layers. From this observation, an idea of a combined strategy which consist of tree pooling at lower layers and max-average pooling at higher layers is implemented on various data sets such as MNIST, CIFAR10, CIFAR100 and SVHN, that outperforms not only baseline but also a network with tree pooling method alone. This scheme is termed as "tree + max-average pooling".

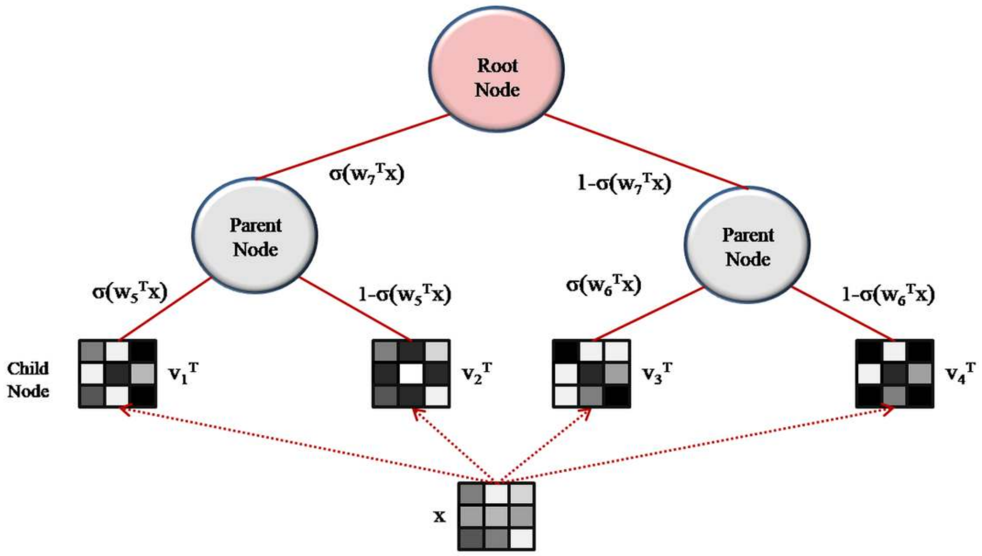


Figure 6. Tree pooling. The child node composed of pooling filters (v_1^T, v_2^T, v_3^T, v_4^T) that learn directly from the input. Like "Gated" max-average pooling, gating mask are used to determine the blending of the child node's output at parent node. The procedure is repeated until the root node is attained, where the mixture of parent node is also computed by gating mask. Modified and regenerated from [30]

2.6 Spatial Pyramid Pooling (SPP)

Spatial pyramid pooling is also known as spatial pyramid matching (SPM). This pooling scheme has a unique characteristic to eliminate the need of input image with fixed size [17, 25], which is a foremost requirement in CNN [18]. The constraint of fixed size is imposed by the fully-connected layers, not the convolution layer. Before the existence of pyramid pooling, cropping [23, 55] and warping [8, 13] are the only methods to fix the size of an image in order to fit within the CNN. But cropping and warping process lead to content loss and geometric distortion in an image, respectively. To obviate the involvement of cropping and warping, the SPP layer is added on top of the last convolution layer which gives rise to a new network, SPP-net [18]. The schematic representation of SPP operation is shown in Figure 7.

In the SPP-net, the SPP layer is made adaptive to the size of feature maps and the numbers of spatial bins are kept constant at every pyramid level. However, the size of bins may vary. Consider an image of size 128x128 and 4 numbers of bins at first level of pyramid and then create a patch of size "32x32" with 4 bins. At second level of pyramid, since the number of bins is 16, patches of size "8x8" are created and maximum value within each bin is taken as an activation value. The pooling of activations for each bin give rise to a vector of fixed dimension as an output of SPP layer which has a fixed length, equal to the multiplication of the number of filters in the last convolutional layer and the number of bins in the SPP layer. Thus, the SPP technique has the ability to generate an output of fixed length without considering the input size. The involvement of multi-level bins is an asset in SPP scheme which makes it robust to the object deformation [25]. In addition, SPP have flexibility toward input image scales during the testing and training phases which enhance the scale-invariance property and reduce the problem of overfitting in the network [19]. SPP technique has been proved as state-of-the-art techniques for classification on Caltech101 [12] and Pascal VOC 2007 [11] and also surpassed even the R-CNN in object detection with faster computation [13].

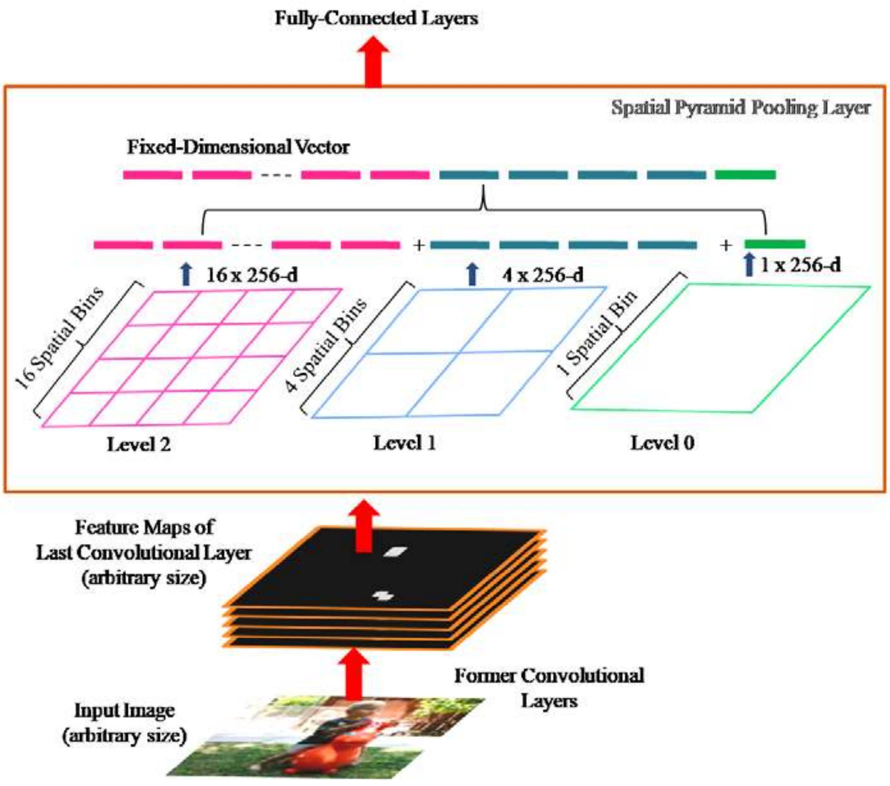

Fully-Connected Layers
Spatial Pyramid Pooling Layer
Fixed-Dimensional Vector
16 x 256-d
4 x 256-d
1 x 256-d
16 Spatial Bins
4 Spatial Bins
1 Spatial Bin
Level 2
Level 1
Level 0
Feature Maps of Last Convolutional Layer (arbitrary size)
Former Convolutional Layers
Input Image (arbitrary size)

Figure 7. SPP-net. SPP layer is inserted between the Conv$_4$ (the last convolutional layer of considered network) and the fully connected layer. The feature maps of arbitrary size from the Conv$_4$ fed to the SPP layer which consists of three level of pyramid: level 0, level 1, and level 2 and the number of bins in each level is 1, 4, and 16, respectively. Remember that the number of bins is fixed in each level but the size of bins may vary. In each bin, we pool the response of each filter present in the Conv$_4$ layer which gives rise to a vector of fixed dimension i.e. equal to the product of number of bins and the number of filters in the Conv$_4$ layer. In this example, 256 depicting the filter number of the Conv$_4$ layer. According to this, the dimension of the output vector would be 21*256. Modified and regenerated from [19]

2.7 Stochastic Pooling

The key idea behind this strategy is to replace the conventional forms of pooling with a stochastic process [54]. In the stochastic pooling, the pooled map response is selected randomly from a multinomial distribution of activations which are obtained from each pooling region after the application of a linear rectification function (ReLU). This strategy involved only the non-negative activations and suppressed the negative activations to zero. Therefore, only the strong activations are considered for sampling. Concurrently, it also assures the accountability of non-maximal activations due to a random selection of activations from a multinomial distribution which helps in prevention of network over-fitting. But before the selection, the probability (p) for each region (R_m) is computed by normalizing the activations and then the selection is performed on the basis of (p) to pick up a location (l) within the region [54]:

$$p_i = \frac{a_i}{\sum_{k \in R_m} a_k} \tag{13}$$

The pooled activation is then set as:

$$s_m = a_l \text{ where } l \sim P(p_1, \dots\dots, p|R_m|) \tag{14}$$

The major drawback of stochastic pooling lies in the constraint to use non-negative activation only. Further, it can be observed from the equation 13 that the strategy is not applicable to negative activations. Since the ReLU activation function set the negative activations to zero, so the stochastic pooling is used with ReLU activation function only. In addition, stochastic pooling leads to the problem of overfitting with the limited training dataset due to high probability in selection of strong activations during the training process. These entire problems in stochastic pooling are known as scale problem.

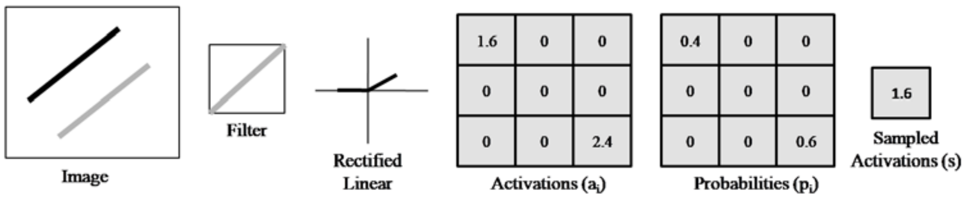


Figure 8. Stochastic Pooling. The selection of sample is carried out from the multinomial distributions of the activations on the basis of the probability computed by normalizing the activations. The larger the activation, the larger is the possibility to be chosen. In this example, there are two activations 1.6 and 2.4 which are having the probability of 40% and 60%, respectively. However, 1.6 is chosen as sampled activation, despite the higher probability of 2.4. Therefore, the important features may be lost in stochastic pooling as one cannot determine that which part of the input will be chosen. Modified and regenerated from [54]

2.8 S3Pool

The S3Pool strategy is a new method of pooling and proposed by Zhai et al. [56] in 2017. In this scheme, the pooling operation is performed in two steps. In step 1, the max-pooling operation is performed on the entire feature maps (received from the convolutional layer) with stride (s) 1. Whereas in step 2, down-sampling is performed on the output of step 1 which is carried out stochastically by first partitioning the feature map of size $x \times y$ into specified number of horizontal (h) and vertical (v) strips, respectively. Where, $h = {}^{x}/_{g}$ and $v = {}^{y}/_{g}$. The 'g' is a hyper-parameter known as grid size which decides the number of horizontal and vertical strips in a feature map. Once the partitioning is over, ${}^{g}/_{s}$ rows and ${}^{g}/_{s}$ columns are then selected randomly from each of the horizontal and vertical strip.

Eventually, a down-sampled feature map of size $x/_s \times y/_s$ is obtained. A schematic of S3Pooling is shown in Figure 9.

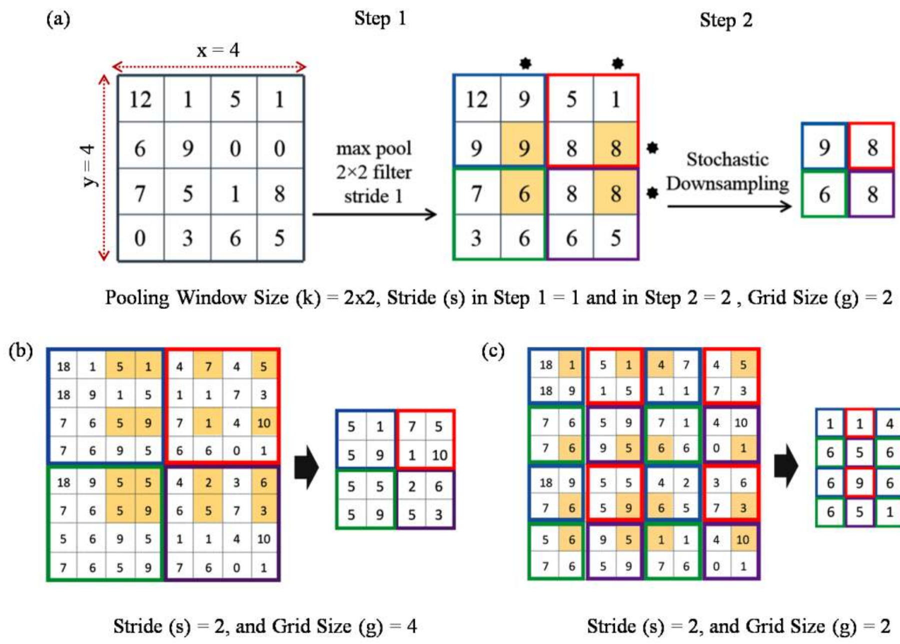

(a)
Step 1
Step 2
x = 4
y = 4
max pool
2×2 filter
stride 1
Stochastic
Downsampling
Pooling Window Size (k) = 2x2, Stride (s) in Step 1 = 1 and in Step 2 = 2 , Grid Size (g) = 2
(b)
(c)
Stride (s) = 2, and Grid Size (g) = 4
Stride (s) = 2, and Grid Size (g) = 2

Figure 9. (a) S3Pool, in this example the size of feature map is 4x4 where, x = 4 and y = 4. In step 1, zero padding is applied at the edges and max-pooling operation is performed with stride 1, while the size of grid and stride will be taken as 2 for step 2. Therefore, the number of horizontal (h) and vertical (v) strips will be 2 according to the scheme. In step 2, the stars are depicting the randomly selected rows and columns in order to obtain downsampled feature map. (b) and (c) Flexibility to control the distortion or stochasticity by the varying the grid size in step 2. Modified and regenerated from [56]

In S3Pool, a random distortion is introduced in the feature maps at each epoch for the same training instance since the sampling performed in step 2 is stochastic. These spatially distorted feature maps give rise to a “virtual” data augmentation at the intermediate layers due to which S3Pool emerged as a firm regularizer. S3Pool also provides flexibility to control the distortion just by changing the grid size to suit different architectures and applications. In order to determine the effectiveness of S3Pool in comparison of other pooling methods, Zhai et al. [56] performed experiments for CIFAR-10, CIFAR-100 and SIT datasets using two architectures: network in network (NIN) and residual network (ResNet). From the experimentation results, the authors have been observed that S3Pool without any data augmentation outperforms the NIN and ResNet with dropout as well as with stochastic pooling even when employed with flipping and cropping technique of data augmentation during testing phase. In terms of computational costs, S3Pool has been found as an efficient method as it provides a computational overhead of 8% and 4% in case of NIN and ResNet, respectively. While for the same networks, stochastic pooling increases the training time by 66% (NIN) and 27% (ResNet). Hence, the S3Pool technique has a good generalization ability and less computational cost as compared to stochastic pooling.

2.9 Rank-Based Pooling (RBP)

In 2016, a new regime of pooling “rank based pooling” was proposed by Shi et al. [45] to alleviate the scale problems which is confronted by value based pooling methods such as stochastic pooling. In RBP, a rank is assigned to all the activations in a pooling region, where the activations are sorted in descending order and rank are assigned in ascending order (start from one) (Figure 10). Mathematically, rank assigning procedure can be represented as [45]:

$$a(i) > a(j) \Rightarrow r(i) < r(j) \tag{15}$$

where ‘a’ is the activations in the pooling region, ‘i’ and ‘j’ defined the position of activations in the pooling region, r(i) and r(j) denotes the rank of activation at ‘i’ and ‘j’, respectively.

If the two activations are of same value, then the rank will be assigned as:

$$a(i) = a(j) \wedge i < j \Rightarrow r(i) < r(j) \tag{16}$$

Further, weights are allocated to all the activations and summed up to get the final output. On the basis of different weighting mechanisms, the RBP is categorized into three new pooling mechanisms namely; rank-based average pooling (RAP), rank-based stochastic pooling (RSP), and rank-based weighted pooling (RWP).

2.9.1 Rank-based Average Pooling (RAP)

In RAP, top (t) highest activations are considered, while the remaining activations are discarded. The highest activations are then averaged to obtain the pooling output. The mathematical expression for the RAP can be given as[45]:

$$s_m = \frac{1}{t} \sum_{i \in R_m, r_i \le t} a_i \tag{17}$$

Where t is the rank threshold (a hyper-parameter to decide how many activations would be involved in averaging) and R_m denotes the pooling area ‘m’ in the feature map. It is important to consider that the RAP turns into max-pooling for t = 1 and average pooling for t = n (the size of pooling area). So the selection of rank threshold should be proper, neither too small nor too large. According to Shi et al. [45], the setting ‘t’ to median value result in satisfactory performance and provides a good trade-off between max-pooling and average pooling. Therefore, RAP is a wonderful mixture of max and average pooling as well as has better discriminating ability than the conventional methods of pooling.

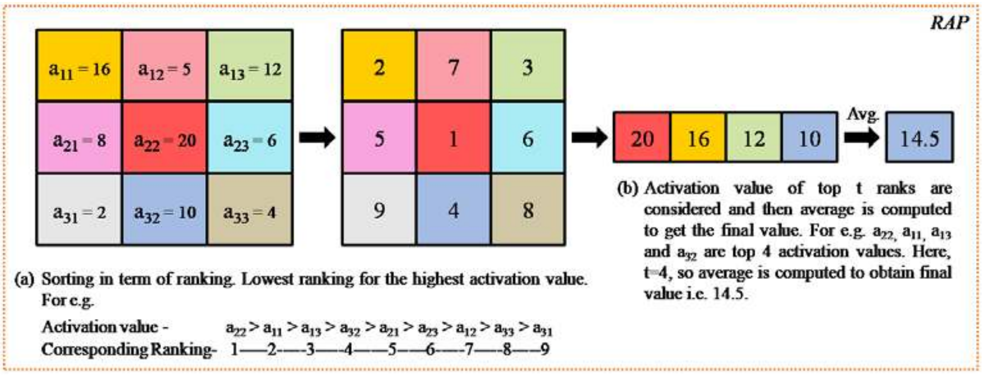

RAP
a11 = 16
a12 = 5
a13 = 12
a21 = 8
a22 = 20
a23 = 6
a31 = 2
a32 = 10
a33 = 4
2
7
3
5
1
6
9
4
8
20
16
12
10
Avg.
14.5
(a) Sorting in term of ranking. Lowest ranking for the highest activation value. For e.g.
Activation value - a22 > a11 > a13 > a32 > a21 > a23 > a12 > a33 > a31
Corresponding Ranking- 1—2—3—4—5—6—7—8—9
(b) Activation value of top t ranks are considered and then average is computed to get the final value. For e.g. a22, a11, a13 and a32 are top 4 activation values. Here, t=4, so average is computed to obtain final value i.e. 14.5.

Figure 10. Rank-based average pooling scheme. The activations in a pooling area are sorted in descending order and ranking is assigned in ascending order. Since t=4, four highest activations are averaged to obtain the pooling output. Modified from [45]

2.9.2 Rank-based Weighted Pooling (RWP)

The RWP strategy is based on the idea that each region in an image is not having an equal importance as the image features are not spatially fixed. Therefore, the reasonable weights are assigned to all the activation in RWP. The smallest weight is assigned to the lowest activation and the largest weight to the highest activation (Figure 11). The distribution of weights is carried out on the basis of computed probability for all the activations by using formula [45]:

$$p_r = \propto (1-\propto)^{r-1}, \text{ where r = 1, 2, … , n} \quad (18)$$

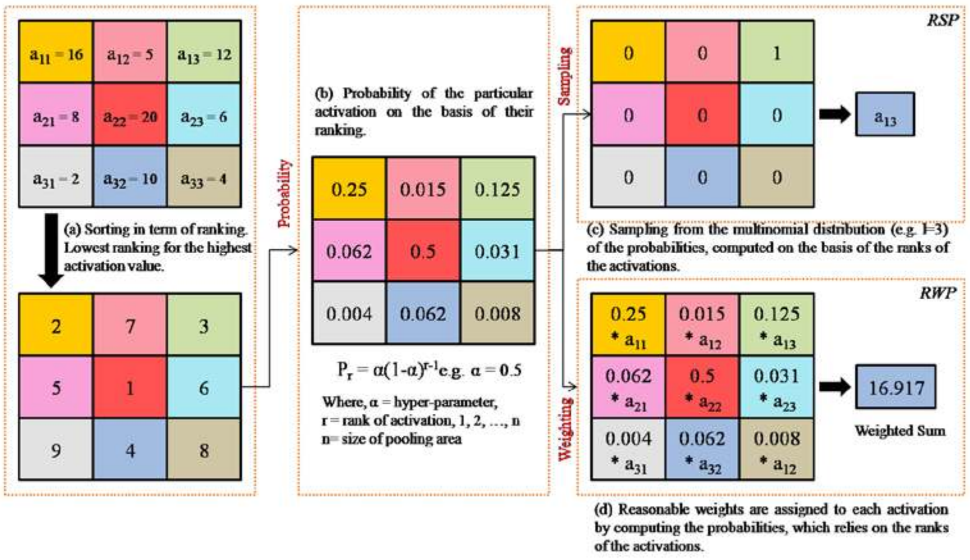
$$\sum_{r=1}^{n} p_r = 1-(1-\propto)^n \text{ , when } 0 < \propto < 1 \quad (19)$$

$$\lim_{n\to+\infty} \sum_{r=1}^{n} p_r = 1 \quad (20)$$

Where the computed probability relies on the rank of the activations and ∝ is a hyper-parameter which controls the probability of the highest activation. Later, the activations within the pooling area are weighted by the probability and then summed up.

$$s_m = \sum_{i \in R_m} p_i a_i \quad (21)$$

According to Shi et al. [45], RWP preserved more discriminating information as compared to max, average, value-based stochastic pooling and also RAP which confirms their good discriminate ability. It all happens due to assigning of reasonable weights to the activations within a pooling area. However, equal importance is given to all the activations in RAP, which became the major reason behind the lack in their discriminate ability.



$a_{11}=16$
$a_{12}=5$
$a_{13}=12$
$a_{21}=8$
$a_{22}=20$
$a_{23}=6$
$a_{31}=2$
$a_{32}=10$
$a_{33}=4$
(a) Sorting in term of ranking. Lowest ranking for the highest activation value.
2 7 3
5 1 6
9 4 8
Probability
(b) Probability of the particular activation on the basis of their ranking.
0.25 0.015 0.125
0.062 0.5 0.031
0.004 0.062 0.008
$P_r = \alpha(1-\alpha)^{r-1}$ e.g. $\alpha = 0.5$
Where, α = hyper-parameter,
r = rank of activation, 1, 2, ..., n
n = size of pooling area
Sampling
RSP
0 0 1
0 0 0
0 0 0
a_{13}
(c) Sampling from the multinomial distribution (e.g. l=3) of the probabilities, computed on the basis of the ranks of the activations.
Weighting
RWP
0.25 * a_{11}
0.015 * a_{12}
0.125 * a_{13}
0.062 * a_{21}
0.5 * a_{22}
0.031 * a_{23}
0.004 * a_{31}
0.062 * a_{32}
0.008 * a_{12}
16.917
Weighted Sum
(d) Reasonable weights are assigned to each activation by computing the probabilities, which relies on the ranks of the activations.

Figure 11. Rank-based stochastic pooling (RSP) and rank-based weighted pooling (RWP). Activations are sorted in term of ranking and probability is computed for the entire activations that present within the pooling area. In RSP, random selection of activation is performed on the basis of calculated rank based probabilities and in RWP, reasonable weights are assigned to the activations according to the computed probabilities. Modified and regenerated from [45]

2.9.3 Rank-based Stochastic Pooling (RSP)

RSP is just similar to the value based stochastic pooling, where the activations are selected from the multinomial distribution of the probabilities. The only difference lies in the computation of probability which is based on the rank of the activation (equation 18), not on the value of the activation (Figure 11). More randomness in selection of the activation is the major advantage of this regime. There is a single hyper-parameter ($\propto$) which controls the probability of the highest activation. According to Shi et al. [45], the setting ($\propto= 0.5$) leads to the adequate performance by introducing more randomness. Better capacity to preserve more diverse information is another advantage of RSP as it preserves more appropriate and task-specific frequencies in the feature maps. In addition, the RSP regime has potential to avoid the scale problem confronted by stochastic pooling, where the negative activations are substituted by zero to compute the probability. Although, the negative activations are not suppressed to zero in RSP as the probabilities are based on the rank. Thus, no constraint of activation function is imposed on the implementation of RSP which provides the flexibility to choose the activation function.

2.10 Fractional Max-out Pooling

As the name suggests "fractional max-out pooling" is a fractional version of max-pooling in which multiplicative factor ($\propto$) is allowed to take non-integer value within a range of $1 < \propto < 2$. In this regime, size of hidden layers is reduced by a fractional factor which provides an opportunity to view an image at a different scale. The visualization of an image at a "proper" scale make it easy to recognize the discerning features that used in identification of an object [16]. Fractional max pooling introduces randomness that associated with the choice of pooling region due to its stochastic nature. The region of pooling can either be overlapped or disjoint which can be selected either randomly or pseudo-randomly with the use of dropout and training data augmentation. According to Graham B. et al. [16], the use of fractional max pooling with overlapped region of pooling works better than the disjoint one. Besides, they have observed that the pseudorandom selection of pooling region with data augmentation performs better as compared to random selection.

Table 1. Evaluation of pooling strategies in terms of upsides and downsides

Pooling	Upsides	Downsides	Reference
Average	• Easy to understand. • Implementation is simple.	• Deterministic in nature. • Resulted in reduced contrast if low magnitudes are taken into consideration.	[3, 28-29]
Max	• Statistical properties make it fit sparse representations. • Perform better when coupled with sparse coding and simple linear classifiers. • Reduce computation for upper layers with elimination of non-maximal components.	• Deterministic in nature. • The discerning features get disappeared when majority of the elements in the pooling area are present with high magnitudes.	[16, 30, 41]
'Mixed' Max-Average	• Stochastic in nature • Helps in prevention of the problem of over-fitting.	• Unresponsive to the characteristics or features of the area being pooled as mixing proportion remain fixed once it learned.	[30]
'Gated' Max-Average	• Responsive in nature. • Adaptive in nature as the mixing proportion can adapt according to the features present within the pooling region.	• Resulted in additional parameters for training.	[30]
Tree	• Responsive in nature. • Differentiable with respect to inputs as well as parameters. • Useful at the lower layers of the network.	• Non-useful for the dense layers of network.	[30]
Pyramid	• Ability to handle input of arbitrary size. • Multi-level spatial bins. • Flexibility towards input image scales.	• Complex implementation during training stage in deep networks.	[19]

Stochastic	• Stochastic in nature. • Non-maximal activations can be utilized. • Possible combination with any other regularization approach like dropout, data augmentation, weight decay etc. • No hyper-parameter to tune • Negligible computation overhead.	• Difficult to understand. • Inapplicable to the negative activations. • Lead overfitting when training data is limited due to participation of strong activations only in process updating. • Scale problem.	[4, 54]
S3Pool	• Easy to understand and implement. • Fast to compute during training. • Flexibility to change the level of distortion. • Introduce data-augmentation at pooling layer level which provides it good generalization ability. • Introduce computational overhead by little in comparison of max-pooling.	• In each pooling layer, setting of grid size should be proper that depends upon the application for which it being used. • Higher grid size may lead to increment in testing error.	[56]
RBP	• Stochastic in nature. • Avoid scale problem. • Introduce more randomness in selection of activations than stochastic pooling. • Single parameter to control the probability of the maximum activation. • Preserve more diverse information.	• Poor performance with respect to fractional max-pooling. • No fixed rule to set the value of hyper-parameter.	[45]
Fractional Max	• Stochastic in nature. • Randomness or pseudo-randomness in selection of pooling region. • Good performance of pseudorandom selection with data augmentation. • Superior result obtained by overlapping instead of disjoint fractional max pooling	• Besides data augmentation, random selection of the pooling region reduces the model performance. • Performance degradation occurs with the disjoint fractional max pooling	[16]

3. Performance Analysis of Pooling Strategies

In this section, the performance of latest pooling strategies has been reviewed and explored for the task of image classification. This paper enlightened the idea of applying pooling operation in the various CNN based architectures. However, the utilization of pooling operation is negated by Geoffrey Hinton. According to him “the pooling operation used in convolutional neural networks is a big mistake and the fact is that it works so well is a disaster”[1]. In this context, he proposed “capsule networks” to discard the pooling layer which are different from the CNN and utilized only a fraction of the data to achieve state-of-the-art performance as compared to the CNN. We note that the goal of our study is not to determine the best architecture in terms of classification, rather to provide a fair analysis on the implications of the pooling strategies in the CNNs.

The performance comparison of different pooling schemes on the various benchmark dataset such as MNIST, CIFAR-10, CIFAR-100 and SVHN is presented in Table 2. The table highlights the network and the type of activation functions which have been utilized during the implementation of these strategies. It has been observed from the Table 2 that the average pooling showed the lowest performance with the error rate of 0.83% for the MNIST dataset. While the gated pooling outperformed the other pooling strategies in which the max-pooling and average pooling are mixed responsively. Further, the performance of gated pooling is followed by the mixed, tree-max-average pooling, and fractional max-pooling with a difference of 0.01%, consecutively. A good performance of these pooling strategies confirms their firm regularization and generalization ability. Since the NIN and maxout networks also claimed a good performance with an error rate of 0.45% and 0.47%, respectively. But the performance is still lower than that achieved by employing the pooling methods. The rank based pooling provides the error rates for MNIST dataset ranging from 0.42% to 0.59% and observed that the error rate provided by the RSP is higher than that provided by the stochastic pooling for the same network with the ReLU activation function. However, the performance of RSP improved when the ReLU activation function is replaced with the parametric ReLU (PReLU) and leaky ReLU (LReLU).

Table 2. Comparison of different pooling strategies on various standard datasets

Pooling Strategies	**Network**	**Activation Function**	**Classification Error Rate (%) on Dataset**				**Ref.**
			MNIST	**CIFAR-10**	**CIFAR-100**	**SVHN**	
Gated	6 Conv. Layer (3x3)	ReLU	0.29	7.90	33.22	----	[30]
Mixed	6 Conv. Layer (3x3)	ReLU	0.30	8.05	33.35	----	
Tree + Max-Avg	6 Conv. Layer (3x3)	ReLU	0.31	7.62	32.37	1.69	
Max-pooling	6 Conv. Layer (3x3)	ReLU	0.39	9.10	34.21	1.91	
Fractional Max-Pooling	Sparse Conv. Network	Leaky ReLU	0.32	3.47	26.39	----	[16]
Spatial Pyramid	----	----	0.64	----	54.23	----	
Stochastic	3 Conv. Layer + 64 filters (5x5)	ReLU	0.47	15.13	42.51	2.8	[54]
Avg-pooling	6 Conv. Layer (3x3)	ReLU	0.83	19.24	47.77	3.72	
RAP	3 Conv. Layer + 64 filters (5x5)	ReLU	0.56	18.68	46.22	----	[45]
RWP	3 Conv. Layer + 64 filters (5x5)	ReLU	0.50	19.05	48.19	----	
RSP	3 Conv. Layer + 64 filters (5x5)	ReLU	0.50	15.44	46.83	----	
RAP	3 Conv. Layer + 64 filters (5x5)	Leaky ReLU	0.59	17.97	45.66	----	
RWP	3 Conv. Layer + 64 filters (5x5)	Leaky ReLU	0.53	19.92	46.69	----	
RSP	3 Conv. Layer + 64 filters (5x5)	Leaky ReLU	0.45	13.84	43.91	----	
RAP	3 Conv. Layer + 64 filters (5x5)	Parametric ReLU	0.58	18.52	45.82	----	
RWP	3 Conv. Layer + 64 filters (5x5)	Parametric ReLU	0.53	18.91	47.05	----	
RSP	3 Conv. Layer + 64 filters (5x5)	Parametric ReLU	0.42	14.90	44.79	----	
RAP	NIN	ReLU	----	9.78	34.81	----	
RWP	NIN	ReLU	----	10.08	35.28	----	
RSP	NIN	ReLU	----	9.44	36.23	----	
RAP	NIN	Leaky ReLU	----	9.43	32.17	----	
RWP	NIN	Leaky ReLU	----	9.84	32.16	----	
RSP	NIN	Leaky ReLU	----	9.26	32.15	----	
RAP	NIN	Parametric ReLU	----	8.73	34.82	----	
RWP	NIN	Parametric ReLU	----	8.91	34.48	----	

RSP	NIN	Parametric ReLU	----	8.67	34.42	----	
RSP (With data augmentation)	NIN	ReLU	----	8.76	----	----	
RSP (With data augmentation)	NIN	Leaky ReLU	----	8.54	30.41	----	
RSP (With data augmentation)	NIN	Parametric ReLU	----	7.76	33.67	----	
----	NIN	ReLU	0.47	10.41	35.68	----	
----	Maxout Network	ReLU	0.45	11.68	----	----	
----	Densely Supervised Network	ReLU	----	9.78	34.57	----	
S3Pool (With flip+crop)	NIN+dropout	ReLU	----	7.71	30.90	----	[56]
----	NIN+dropout (With flip+crop)	ReLU	----	9.34	32.36	----	
S3Pool (With flip+crop)	ResNet	ReLU	----	7.09	29.36	----	
----	ResNet (With flip+crop)	ReLU	----	7.72	30.88	----	
----	ALL-CNN (With data augmentation)	ReLU	----	7.25	----	----	[46]
----	ALL-CNN (Without data augmentation)	ReLU	----	9.08	33.71	----	

In case of the dataset CIFAR-10 and CIFAR-100, fractional-max-pooling provided the outstanding performance, followed by the S3Pool (ResNet+data augmentation). Fractional-max-pooling and S3Pool, both of the pooling regimes surpassed nearly all the architectures that have been developed to discard the pooling layer such as NIN and ALL-CNN. The worst performance for CIFAR-10 and CIFAR-100 has been shown by RWP (LReLU activation) and the spatial pyramid pooling, respectively. While, the average pooling holds the second position as a worse performer for both the datasets. In addition, it has been observed that the employment of rank based pooling within the NIN network (with or without data augmentation) for different activation functions provided an acceptable performance and better than the NIN, maxout network, and densely supervised network alone. However, the use of rank based pooling within the network of 3 convolutional layer and 64 filters of size 5x5 decline the performance considerably. Through these empirical observations, it is clear that the average pooling is the worst pooling strategies in comparison of other pooling methods. The performance of pooling strategies largely depends on the network and the activation function which are chosen for the implementation. Moreover, the techniques of data augmentation also influence the overall performance of the network, where the RSP provided a lower error rate with data augmentation as compared to no augmentation for CIFAR-10 and CIFAR-100 datasets (Table 2).

4. Pooling – A Firm Prior and Regularizer

In general, the prior is the distribution of probability for an uncertain quantity which shows an impression about the quantity before the acquisition of relevant evidence. Simply, the prior distribution of the probability over the parameters of a model is performed to estimate a reasonable model for a specified task, before seeing the testing data. Prior can be weak or strong which depends on the probability density concentrated in the prior. A prior with low entropy (low variance) is considered as strong prior and with high entropy (high variance) as weak prior. In neural networks, pooling can be used as an infinitely strong prior. An infinitely strong prior distributed zero probability to some parameters and even excluded from the use, regardless of the data. By implementing the pooling operation, each unit in the network is supposed to be invariant to small translations in the input data. Hence, pooling act as a strong prior for example: max-pooling, where only the maximum value is considered, whereas the other values are discarded. But in some cases, pooling may cause underfitting if the assumption made by the prior is inaccurate. For instance, image classification and object detection task greatly depends on preserving the spatial information and the application of pooling on all the features may results in information loss, which ultimately increase the training errors. In order to alleviate this problem of underfitting, some convolutional networks are designed to get highly invariant features and features that will not under fit when the pooling prior is inaccurate [48]. The networks are designed in such a way that they use pooling on few channels instead of all the channels to minimize the training error.

Regularization is another key role of the pooling operation in the CNN. In deep networks, regularization approach is used to trim down the test error [2]. Dropout [21, 47], weight decay [14, 24, 38], data augmentation [31], weight tying [36] and cutout [7] are different forms of regularization technique which assists the model in achieving high-quality performance on the test set and prevent the model from overfitting. Dropout is the most popular regularization approach in which some nodes in the neural network are dropped out randomly. But, dropout is only applicable to the fully connected layers, not to the convolutional layers. Therefore, stochastic pooling emerges as a new regularization approach for convolutional layers [54] in which activations are picked randomly on the basis of multinomial distribution of probability during training. Max-pooling-dropout [50] is another regularization approach in which dropout is applied to the input of the max-pooling layers. The max-pooling-dropout is similar to stochastic pooling in term of activation picking and inspired by the dropout regularization approach. In addition, both the strategies adopted probabilistic weighting for model averaging at test time. Despite of these similarities, max-pooling-dropout is different from stochastic pooling in performance for retaining probability. The max-pooling-dropout outperforms the stochastic pooling by a significant margin when the retaining probability (typically around 0.5) is neither too small nor too large [50]. Data augmentation is the easiest and one of the simplest methods of regularization to reduce overfitting on the test data. However, S3Pool is only the pooling regime which implicitly augments the data and improves the generalization ability of the model. The results from the experimental work of Zhai et al. [56] confirmed that S3Pool have ability to surpass even the dropout (with data augmentation) and the stochastic pooling approach with the marginal increment in the training time. It has also been observed from the Table 2 that the pooling strategies act as a stronger regularizer and even able to surpass the other popular regularization approaches (dropout, data augmentation, maxout network), except cutout. The cutout is a variant of dropout technique in which the nodes are dropped at the input stage instead of hidden layers [7]. The cutout approach of regularization also has the ability to augment the data implicitly by generating occluded versions of the input sample and overcome the problem of occlusion encountered in the most of computer vision tasks. The cutout approach has provided a state-of-the-art performance for the CIFAR-10 and CIFAR-100 dataset (with and without data augmentation). Therefore, the pooling strategies can be combined easily with the other regularization approaches in order to enhance the performance of the network (Table 2). Hence, the designing of new methodologies comprises of pooling methods in conjunction with different regularization approaches can be an interesting topic of research in future. The influence of different combinations of pooling strategies on the performance of the model can also be another aspect of future work.

5. Conclusions

The pooling is a powerful concept in deep architecture and widely used in CNN's to solve the task related to computer vision. Most of the studies focused on max-pooling due to its easy implementation and sparse representation, but its deterministic nature is its major drawback. We find stochasticity as an important asset for a variety of pooling regimes such as 'mixed' max-average, stochastic, S3Pool, RSP and fractional max pooling. Since stochasticity in these regimes, introduces randomness to select activation or pooling region that helps in reducing the overfitting and improve generalization ability of the model. In addition, pooling operations assist the deep architectures in terms of computational cost by reducing the number of parameters involved in training by virtue of its dimensionality reduction property. However, the "Gated" Max-Average pooling regime is an exception which increases the computational overhead by introducing additional parameters for training. Based on the detailed analysis of the pooling regimes and their performance for the task of classification, we found that the pooling operations largely depend upon the network architecture and the activation function for their performance. Overlapping is another factor that affects the performance of the network as it decreases the chances of information loss during the pooling operation. While the step size of overlapping is very necessary to consider because a blind increment in step size can drop the model performance significantly. We also found that S3Pool act as the strongest regularization approach because it augments the data implicitly. Moreover, it is able to form possible combinations with other regularization methods. Thus, it would not be wrong if we say that the choice of pooling operation is a kind of empiricism, but we believe that our work could pave a leading step in the better understanding of pooling methods and factors involved in the improvement of pooling performance.

References

[1] *Reddit, Machine Learning*. Available: https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/

[2] Achille A., Soatto S., Information dropout: Learning optimal representations through noisy computation, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2018, 2897-2905.

[3] Boureau Y.-L., Le Roux N., Bach F., Ponce J., LeCun Y., Ask the locals: multi-way local pooling for image recognition, in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, 2651-2658.

[4] Cai M., Shi Y., Liu J., Stochastic pooling maxout networks for low-resource speech recognition, in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, 3266-3270.

[5] Cheng Y., Zhao X., Cai R., Li Z., Huang K., Rui Y., Semi-Supervised Multimodal Deep Learning for RGB-D Object Recognition, in *IJCAI*, 2016, 3345-3351.

[6] Dalal N., Triggs B., Histograms of oriented gradients for human detection, in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, 886-893.

[7] DeVries T., Taylor G. W., Improved regularization of convolutional neural networks with cutout, *arXiv preprint arXiv:1708.04552,* 2017, 1-8.

[8] Donahue J., Jia Y., Vinyals O., Hoffman J., Zhang N., Tzeng E., Darrell T., Decaf: A deep convolutional activation feature for generic visual recognition, in *International conference on machine learning*, 2014, 647-655.

[9] Dumpala S. H., Chakraborty R., Kopparapu S. K., k-FFNN: A priori knowledge infused Feed-forward Neural Networks, *arXiv preprint arXiv:1704.07055,* 2017, 1-9.

[10] Ellacott S., An analysis of the delta rule, in *International Neural Network Conference*, 1990, 956-959.

[11] Everingham M., Van Gool L., Williams C. K., Winn J., Zisserman A., The pascal visual object classes (voc) challenge, *International journal of computer vision,* **88**, 2, 2010, 303-338.

[12] Fei-Fei L., Fergus R., Perona P., Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, *Computer vision and Image understanding,* **106**, 1, 2007, 59-70.

[13] Girshick R., Donahue J., Darrell T., Malik J., Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, 580-587.

[14] Goodfellow I., Bengio Y., Courville A., *Deep learning*1, MIT press Cambridge, 2016.

[15] Goodfellow I. J., Warde-Farley D., Mirza M., Courville A., Bengio Y., Maxout networks, *arXiv preprint arXiv:1302.4389,* 2013, 1-9.

[16] Graham B., Fractional max-pooling, *arXiv preprint arXiv:1412.6071,* 2014, 1-10.

[17] Grauman K., Darrell T., The pyramid match kernel: Discriminative classification with sets of image features, in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, 1458-1465.

[18] He K., Zhang X., Ren S., Sun J., Spatial pyramid pooling in deep convolutional networks for visual recognition, in *European conference on computer vision*, 2014, 346-361.

[19] He K., Zhang X., Ren S., Sun J., Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **37**, 9, 2015, 1904-1916.

[20] Hebb D., "The organization of behavior: a neuropsychological theory. Mahwah, NJ: L," ed: Erlbaum Associates, 1949.

[21] Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., Improving neural networks by preventing co-adaptation of feature detectors, *arXiv preprint arXiv:1207.0580,* 2012, 1-18.

[22] Khan Z. H., Alin T. S., Hussain M. A., Price prediction of share market using artificial neural network (ANN), *International Journal of Computer Applications,* **22**, 2, 2011, 42-47.

[23] Krizhevsky A., Sutskever I., Hinton G. E., Imagenet classification with deep convolutional neural networks, in *Advances in neural information processing systems*, 2012, 1097-1105.

[24] Lang K. J., Hinton G. E., Dimensionality reduction and prior knowledge in e-set recognition, in *Advances in neural information processing systems*, 1990, 178-185.

[25] Lazebnik S., Schmid C., Ponce J., Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in *null*, 2006, 2169-2178.

[26] LeCun Y., Generalization and network design strategies, *Connectionism in perspective,* 1989, 143-155.

[27] LeCun Y., Bengio Y., Hinton G., Deep learning., *Nature* **521**, 2015, 436-444.

[28] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D., Backpropagation applied to handwritten zip code recognition, *Neural computation,* **1**, 4, 1989, 541-551.

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

[30] Lee C.-Y., Gallagher P. W., Tu Z., Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree, in *Artificial Intelligence and Statistics*, 2016, 464-472.

[31] Lemley J., Bazrafkan S., Corcoran P., Smart Augmentation Learning an Optimal Data Augmentation Strategy, *IEEE Access,* **5**, 2017, 5858-5869.

[32] Lowe D. G., Distinctive image features from scale-invariant keypoints, *International journal of computer vision,* **60**, 2, 2004, 91-110.

[33] McCulloch W. S., Pitts W., A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics,* **5**, 4, 1943, 115-133.

[34] Mehdipour Ghazi M., Kemal Ekenel H., A comprehensive analysis of deep learning based representation for face recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, 34-41.

[35] Nagpal S., Singh M., Vatsa M., Singh R., Regularizing deep learning architecture for face recognition with weight variations, in *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, 2015, 1-6.

[36] Nowlan S. J., Hinton G. E., Simplifying neural networks by soft weight-sharing, *Neural computation,* **4**, 4, 1992, 473-493.

[37] Piccinini G., The First computational theory of mind and brain: a close look at mcculloch and pitts's "logical calculus of ideas immanent in nervous activity", *Synthese,* **141**, 2, 2004, 175-215.

[38] Plaut D. C., Experiments on Learning by Back Propagation, 1986, 1-49.

[39] Rumelhart D. E., Hinton G. E., Williams R. J., Learning representations by back-propagating errors, *nature,* **323**, 6088, 1986, 533-536.

[40] Rumelhart D. E., McClelland J. L. (1986). *Parallel distributed processing: explorations in the microstructure of cognition. volume 1. foundations.*

[41] Scherer D., Müller A., Behnke S., Evaluation of pooling operations in convolutional architectures for object recognition, Springer, 2010, 92-101.

[42] Shallu, Mehra R., Automatic Magnification Independent Classification of Breast Cancer Tissue in Histological Images Using Deep Convolutional Neural Network, Singapore, 2019, 772-781.

[43] Shallu, Mehra R., Kumar S., "An insight into the convolutional neural network for the analysis of medical images," presented at the Nanotechnology for Instrumentation and Measurement Workshop 2017.

[44] Sharma S., Mehra R., Breast cancer histology images classification: Training from scratch or transfer learning?, *ICT Express,* **4**, 4, 2018, 247-254.

[45] Shi Z., Ye Y., Wu Y., Rank-based pooling for deep convolutional neural networks, *Neural Networks,* **83**, 2016, 21-31.

[46] Springenberg J. T., Dosovitskiy A., Brox T., Riedmiller M., Striving for simplicity: The all convolutional net, *arXiv preprint arXiv:1412.6806,* 2014,

[47] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research,* **15**, 1, 2014, 1929-1958.

[48] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A., Going deeper with convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, 1-9.

[49] Szegedy C., Toshev A., Erhan D., Deep neural networks for object detection, in *Advances in neural information processing systems*, 2013, 2553-2561.

[50] Wu H., Gu X., Max-pooling dropout for regularization of convolutional neural networks, in *International Conference on Neural Information Processing*, 2015, 46-54.

[51] Xu B., Wang N., Chen T., Li M., Empirical evaluation of rectified activations in convolutional network, *arXiv preprint arXiv:1505.00853,* 2015, 1-5.

[52] Yadav N., Yadav A., Kumar M., Preliminaries of Neural Networks, *An introduction to neural network methods for differential equations,* 2015, 17-42.

[53] Yu D., Wang H., Chen P., Wei Z., Mixed pooling for convolutional neural networks, in *International Conference on Rough Sets and Knowledge Technology*, 2014, 364-375.

[54] Zeiler M. D., Fergus R., Stochastic pooling for regularization of deep convolutional neural networks, *arXiv preprint arXiv:1301.3557,* 2013, 1-9.

[55] Zeiler M. D., Fergus R., Visualizing and understanding convolutional networks, in *European conference on computer vision*, 2014, 818-833.

[56] Zhai S., Wu H., Kumar A., Cheng Y., Lu Y., Zhang Z., Feris R. S., S3Pool: Pooling with Stochastic Spatial Sampling, in *CVPR*, 2017, 4003-4011.

[57] Zhou B., Khosla A., Lapedriza A., Oliva A., Torralba A., Object detectors emerge in deep scene cnns, *arXiv preprint arXiv:1412.6856,* 2014, 1-12.

Received 22.09.2018, Accepted 29.04.2019