REGP: A NEW POOLING ALGORITHM FOR DEEP CONVOLUTIONAL NEURAL NETWORKS

O. Yildirim[*], *U.B. Baloglu*[†]

Abstract: In this paper, we propose a new pooling method for deep convolutional neural networks. Previously introduced pooling methods either have very simple assumptions or they depend on stochastic events. Different from those methods, RegP pooling intensely investigates the input data. The main idea of this approach is finding the most distinguishing parts in regions of the input by investigating neighborhood regions to construct the pooled representation. RegP pooling improves the efficiency of the learning process, which is clearly visible in the experimental results. Further, the proposed pooling method outperformed other widely used hand-crafted pooling methods on several benchmark datasets.

Key words: *convolutional neural networks, deep learning, pooling*

Received: July 15, 2018 **DOI:** 10.14311/NNW.2019.29.004
Revised and accepted: March 4, 2019

1. Introduction

Deep learning is an evolving methodology, which represents more abstract concepts to discover better learning algorithms less dependent on feature engineering [3]. Deep learning implementations extract high-level features of complex data sets automatically through a special multi-layer neural network structure [10]. These structures usually have multiple hidden layers instead of a single one, and these layers are processed with different techniques to train the deep neural network.

Convolutional neural networks (CNNs) are a special type of artificial neural networks, and they learn very expressive features in an adaptive manner [2]. CNNs have receptive fields with weight attributes called as convolutional units which are shifted step by step across a 2-dimensional array of input values [22]. This structure has recently successfully applied to various research areas from image recognition to bio-medical image segmentation [6, 9, 13, 16, 19, 25, 27]. CNNs have a linear mathematical operation of convolution and parameters of the convolution layer contain learnable filters or kernels. In CNN structures both feature extraction and classification processes are performed in a conventional training phase,

[*]Ozal Yildirim; Computer Engineering Department, Munzur University, 62000, Tunceli, Turkey, E-mail: yildirimozal@hotmail.com

[†]Ulas Baran Baloglu – Corresponding author; Department of Computer Science, University of Bristol, BS8 1UB, Bristol, UK, E-mail: ulasbaloglu@gmail.com

where several convolutional and pooling layers are applied successively to obtain hierarchical properties of the input data. Additional layers are needed because a deep CNN generalize better when having a more compact representation [12]. A sufficient number of convolution and pooling layers is necessary to represent the features of the input precisely.

Pooling layers work as a kind of down-sampling step, where semantically corresponding representation of the input is determined. Using all properties of the input will produce a very high computational cost so that pooling is used on the output matrix obtained from the convolution layer. In the pooling layer, the data is reduced in size by applying various techniques to the regions in predetermined dimensions. The applied technique usually gives the name to the pooling process, such as Average pooling (for taking averages) and Max pooling (for taking maximum values). Pooling is a fundamental component of deep CNNs. Nevertheless, the most widely used pooling methods, Average pooling, and Max pooling have very simple assumptions, and they do not provide optimal solutions [14]. A similar problem occurs with methods which depend on probability values, such as Stochastic pooling [31]. Many applications follow previously created models without investigating or optimizing the pooling layer, but it is clear that there is a need to develop more feasible methods to be used in the pooling layer of deep CNNs.

In this paper, we propose a new pooling method, which is as fast as popular hand-crafted pooling methods. More specifically, we present the RegP pooling algorithm to be used in deep CNNs. Contributions of this study include: (1) a new pooling algorithm is proposed; (2) the proposed algorithm, which has same computational complexity with other hand-crafted pooling methods, significantly decreases learning time in CNNs according to various experiments; (3) the proposed algorithm is compared with widely used standard pooling methods on three different benchmark datasets.

The remainder of the paper is organized as follows. A review of the related literature is given in Section 2, followed by the details of the proposed RegP pooling in Section 3. Comprehensive analysis of the performed experiments is given in Section 4 to demonstrate the efficiency of the proposed method. Finally, concluding remarks are given.

2. Related works

There are different types of layers used in the construction of CNNs, and only a small number of studies in the literature have explicitly investigated the pooling layer and developed pooling methods. In most of the studies on the CNNs, usually one of Max pooling or Average pooling methods has been used, and it has been unfortunately ignored that a pooling method superior to these methods may improve the overall model performance. Previous studies done in this area could be divided into three categories: hand-crafted pooling, learning-based pooling and probabilistic pooling [26]. Different than this categorization, we can classify spatial pooling methods in a fourth category by separating them from hand-crafted pooling methods.

Most known methods Max pooling and Average pooling are hand-crafted pooling methods. In the Max pooling method, the largest activation value in a pooling

region is selected as the pooling sample. In a similar fashion, the average value of the region is selected as the pooling sample in the Average pooling method. Both of these methods assume that local features are independently distributed, and they cannot successfully detect the underlying difference. Because of their implementation simplicity and speed of operation, these two methods are frequently used in deep CNN studies. The proposed RegP pooling method is as simple as these two methods, and it further achieves faster learning in some experiments than these widely used methods.

One of the important studies carried out in this area is the Stochastic pooling method of Zeiler and Fergus [31]. In this study, a stochastic method, which randomly selects the pooling sample, was proposed. A multinomial distribution was used as the selection criteria. Since the probability values are calculated by dividing the values by the sum of the values of the field, the larger the values, the greater the chance of being selected. The major drawback of this pooling method is the outcome naturally linked to the chance factor, and a model may produce different results at each execution.

In another recent study, a pooling method named as Rank-based pooling was developed [24]. This method has different variations in which activation values in each pooling region are first sorted according to their values. The sort operation is needed as a preprocessing operation to determine the ranks. The major weakness of this method is the additional computational complexity of sorting process, which will eminently decrease the overall performance of the method. It is also questionable whether it is worth to this performance loss or not because this approach is not superior to Average pooling and Max pooling in presented experimental results. There is also another rank based pooling study in the literature for capturing video sequence data which aggregates the relevant information from a video sequence by using learning-based ranking [5]. Other researchers have also investigated the learning-based approaches and tried to train the pooling layer within CNNs. LEAP (LEArning Pooling) [26], which is proposed by Sun et al., is one of these methods. In the LEAP method, the weights are learned from the training data, and pooling operator calculates aggregated information within the local region for each feature. In another study, representation learning and classifier training are implicitly combined in a framework named as Task-Driven Feature Pooling [30]. This method includes optimization processes so that it is not a simple and computationally feasible method. Cross-convolutional-layer pooling also works as a learning-based approach and examines activations of two consecutive convolutional layers [15]. This approach requires a pre-trained model and also it is complicated to implement like other learning-based approaches.

There are spatial pooling studies in the literature, which are specifically used for face image classification and object-based surveillance [1, 8, 18, 23]. Class-Specific Pooling Shapes (CSPS) is a recently proposed method, which can learn compact geometric information within the input image [29]. Another study adopts a category-specific distribution matrix to determine how patches of an image are pooled together [17]. Both weighted and content-based spatial pooling methods usually have computational performance problems, and they are very dependent on the input.

3. The proposed method

In this section, we present the details of RegP pooling method and evaluate its computational complexity by comparing it with hand-crafted pooling methods.

3.1 RegP Pooling

The proposed RegP pooling analyzes an activation value by examining its surrounding activation values. Hand-crafted pooling methods assume that local features are independently distributed, and these methods determine the pooling sample accordingly. However, with this approach, an ambiguity may arise when different overlapping objects represented in an image and these pooling methods may have difficulty in separating those objects. RegP pooling treats activation values in a different manner. For each activation value, the activation values which are in the neighborhood are examined. As expected, the maximum number of neighborhood activation values is 9. For the activation value being examined, a counter value is calculated according to the same (or similar) activation values around. In a pooling window, activation value with the largest counter value is selected as the pooling sample. If more than one activation value has the same counter value, the average of these activation values is calculated to determine the value of the pooling sample.

In Fig. 1, for a pooling region how RegP pooling calculates the pooling sample is illustrated with a toy example. For the first activation value, there are three neighborhood activation values. Since two of them have same (or close) values, the counter value is set as 2, and the current pooling sample is set as 0. The second activation value also has two neighbors with the same activation values. According to the RegP pooling algorithm, the new value of the current pooling

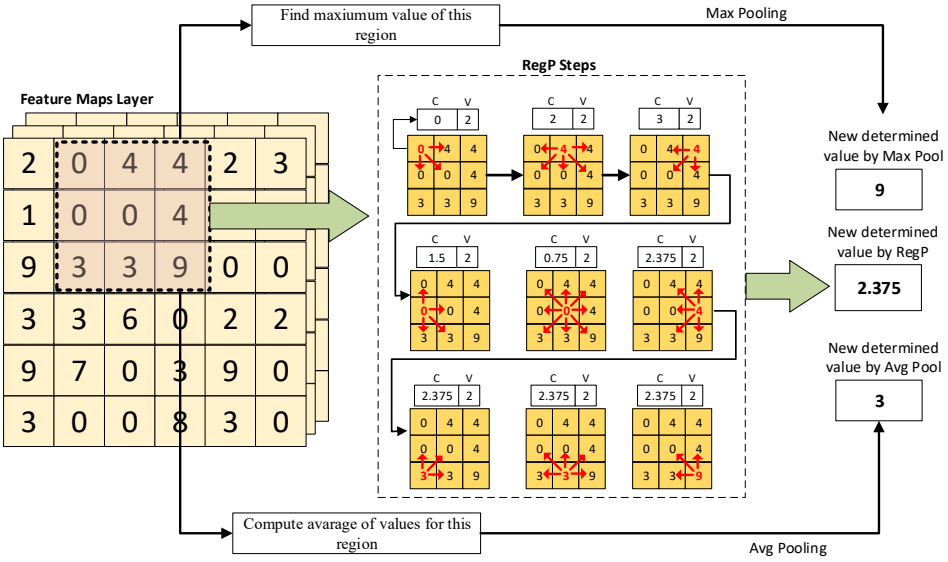

Fig. 1 *A toy example to illustrate the proposed RegP pooling method.*

sample is determined by calculating the average of the current activation value and the previous value of the current pooling sample. After completing 9 steps in a similar manner, the pooling sample is calculated as 2.375. For this pooling region, pooling sample is calculated as 9 with Max pooling and 3 with Average pooling. The algorithm of the proposed RegP pooling is given below. As it is difficult to find exact same values in surrounding pixels, min and max values are used to create an acceptable range.

3.2 Computational Complexity

The proposed RegP pooling method examines the activation values in a single loop structure like hand-crafted pooling methods, such as Average pooling and Max pooling. For each activation value, a comparison is made using a total of k neighborhood activation values. For these comparisons part, another loop structure is not needed, and the comparison could be made by using the selection structures. The maximum value of k is 8, and the activation values at the pooling window edge are have smaller values than 9. From this perspective, the computational complexity of the method is $\Theta(kn^2)$. In the worst-case scenario, the value of k is equal to 8, and computational complexity of the method becomes $\Theta(n^2)$. Hand-crafted pooling methods Max pooling and Average pooling also have $\Theta(n^2)$ computational complexity. Other pooling methods usually have worse computational complexity as they try to process the values in the pooling window.

Algorithm 1 RegP Pooling.

Input: pooling window coordinates $x1, y1, x2, y2$
Output: pooling sample p

```
maxCounter = 0;
p = 0;
set min and max values
for (i = y1; i ≤ y2; i++) do
  for (j = x1; j ≤ x2; j++) do
    read ActivationValue valA;
    counter = 0;
    investigate neighborhood activation values nVal
    if (nVal*min ≤ valA ≤ nVal∗max) then
      increment counter;
    end if
    if (counter > maxCounter) then
      maxCounter = counter;
      p = valA;
    else if (counter = maxCounter) then
      p = (p+ valA)/2;
    end if
  end for
end for
return p
```

4. Experiments

For the performance evaluation of the proposed pooling method, widely used MNIST and CIFAR-10 datasets are used. In addition to these datasets, a model for the Drexel texture dataset was created. This model was used to obtain results of Max pooling, Average pooling, and the proposed RegP pooling.

4.1 MNIST Dataset

The MNIST [13] dataset contains handwritten digits, and it consists of 60000 training and 10000 test data. This dataset is very suitable for testing machine learning algorithms on real-world data. It is derived from original NIST database, and images have 28×28 dimension. In Fig. 2, various data samples from the MNIST are given.

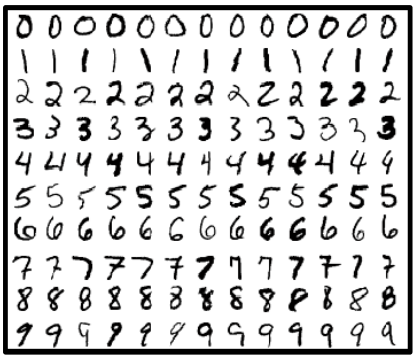

Fig. 2 *Various digit samples from the MNIST dataset.*

Convolutional Neural Networks [4, 7, 13, 20] were previously implemented on MNIST for the recognition process. There are several LeNet models used for recognition on the MNIST dataset. In this study, the CNN model from the Matconvnet [28], which is depicted in Fig. 3, is used for the MNIST dataset.

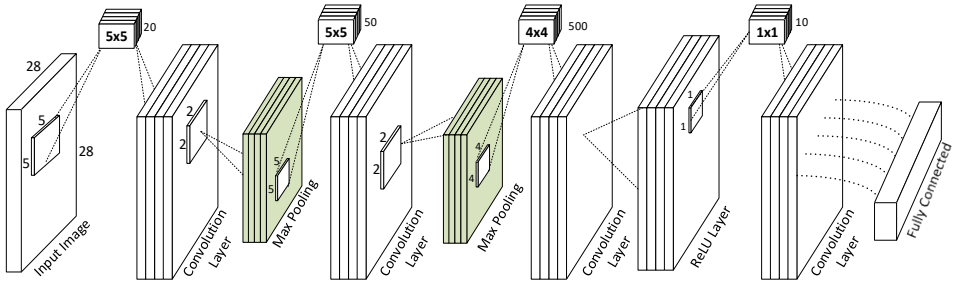

Fig. 3 *Original CNN model from the Matconvnet for the MNIST dataset.*

The MNIST LeNet architecture consists of 8 layers in total. The first layer is the convolution layer, which contains 20 kernels in 5×5 dimension. The second layer

is the max pooling layer for performing 2×2 subsampling process. The third layer is again a convolution layer but with 50 kernels in 5×5 size, and a 2×2 dimension max pooling layer follows this layer. The fifth layer consists of a convolution layer with 500 kernels in 4×4 dimension, and an activation map forming ReLU layer follows as the sixth layer of the model. The seventh layer is another convolution layer with 1×1 dimension 10 kernels, which contain activation values. The fully connected layer is the eighth and final layer. In this model, there are two pooling layers, which separately perform the subsampling operations. Both of these pooling layers implement the Max pooling method.

For the performance evaluation of the proposed RegP pooling, the pooling layers in the Lenet model were arranged as Max, Avg, and RegP, and experimental results were obtained for 40 epochs. Fig. 4 shows the performance results of each pooling method on the MNIST dataset. According to the obtained results, it is seen that the most successful results on the MNIST dataset are obtained according to the original model parameters when both pooling layers use Max pooling. The arrangement of the original model parameters (kernel size, bias size, etc.) according to the Max pooling method is influential on this result. However, it is clear that the proposed RegP pooling method has more successful results than the Average pooling method on the MNIST dataset. Tab. I shows average error rates obtained from all pooling methods at various epoch intervals.

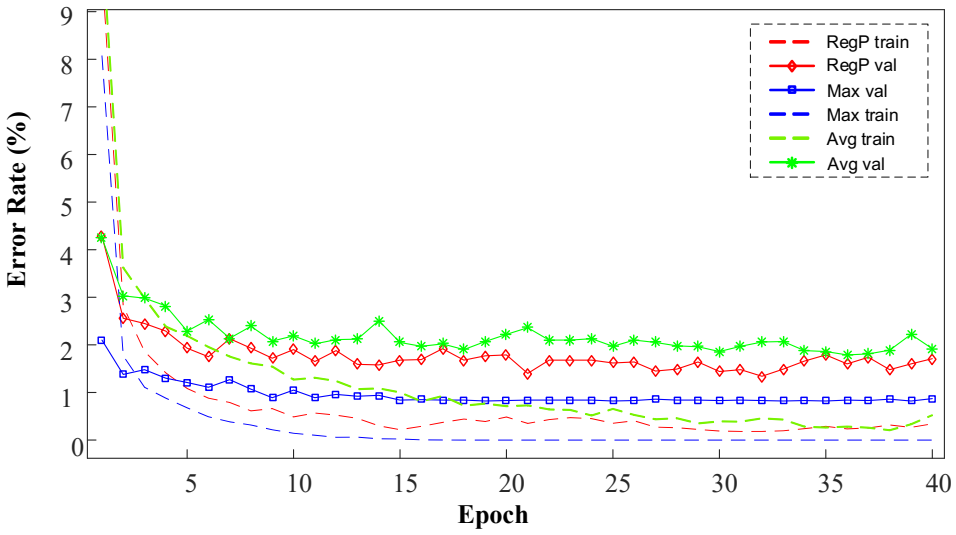

Fig. 4 *Performance graphs of the pooling methods on the MNIST dataset.*

4.2 CIFAR-10 Dataset

The CIFAR-10 [11] dataset contains 60000 images of 10 different classes in 32×32 dimensions. For each class, 50000 of these 60000 color images are allocated for the training, and the remaining 10000 images are allocated for the testing. This dataset is divided into five training batches and one test batch. The data in each

Pooling Method	Train Error Rate [%] 20-25 Epoch	25-30 Epoch	30-35 Epoch	35-40 Epoch	Mean	Validation Error Rate [%] 20-25 Epoch	25-30 Epoch	30-35 Epoch	35-40 Epoch	Mean
Max Pool	0	0	0	0	0	0.83	0.84	0.83	0.84	0.83
Avg Pool	0.65	0.47	0.36	0.31	0.45	2.15	1.99	1.95	1.91	2.01
RegP Pool	0.42	0.28	0.21	0.28	0.30	1.64	1.55	1.53	1.65	1.59

Tab. I *Error rates of pooling methods on the MNIST dataset for different epoch intervals.*

Pooling Method	Train Error Rate [%] 20-25 Epoch	25-30 Epoch	30-35 Epoch	35-40 Epoch	Mean	Validation Error Rate [%] 20-25 Epoch	25-30 Epoch	30-35 Epoch	35-40 Epoch	Mean
Max Pool	13.61	8.51	8.11	7.73	9.58	25.77	24.83	24.91	25.07	25.17
Avg Pool	21.26	18.27	18.05	17.83	18.90	30.02	29.02	28.98	28.99	29.27
RegP Pool	16.51	13.40	13.21	13.03	14.09	25.23	24.27	24.33	24.36	24.56

Tab. II *Error rates of pooling methods on the Cifar-10 dataset at different epoch intervals.*

Pooling Method	Train Error Rate [%] 20-40 Epoch	40-60 Epoch	60-80 Epoch	80-100 Epoch	Mean	Validation Error Rate [%] 20-40 Epoch	40-60 Epoch	60-80 Epoch	80-100 Epoch	Mean
Max Pool	52.52	40.92	29.65	24.94	37.11	51.50	41.47	31.30	25.42	37.44
Avg Pool	37.82	21.14	9.38	3.94	18.07	43.20	30.29	20.05	14.02	26.92
RegP Pool	3.79	0.08	0	0	1.00	14.65	8.57	8.40	8.43	10.07

Tab. III *Error rates of pooling methods on the Drexel texture dataset.*

batch was randomly selected. Image classes in the CIFAR-10 dataset and randomly selected image samples from these classes are given in Fig. 5.

Fig. 5 *Randomly selected image samples from the CIFAR-10 dataset.*

For the CIFAR-10 dataset, a model consisting of 12 layers is used in the CNN-based classification process. This model includes 5 convolution layers, 3 pooling layers, and 4 activation layers. The 12-layer CNN architecture used in the study for the CIFAR-10 dataset is given in Fig. 6. The first pooling layer in the CIFAR-10 model of the Matconvnet is the Max pooling, which is the second layer of the model and has a size of 3×3. The following pooling methods are again having a size of 3×3. Average pooling is used in layers 6 and 9. After the Max pooling layer, a subsampling process was applied to activation layers by Average pooling layers.

No parameters have been changed in the original model to evaluate the performance of the three pooling methods. Only corresponding pooling layers of the model is replaced with the proposed pooling method. Values of the training error rate and test error rate for each pooling method were examined for 50 epochs. Fig. 7 shows the performance results of each pooling method.

4.3 Drexel Texture Dataset

The Drexel texture dataset [21] contains 20 different texture images. These images were obtained 2000 times with multiple distances, different light angles, and different in-plane and out-plane angles. In this study, grayscale textures were created for images selected from the Drexel dataset. These textures are in 64×64 dimensions with distance $= 1$, in-plane rotation: 0^o out-of-plane rotation: 0^o. 100 sub-images are derived for each class. Thus, there is a total of 2000 64×64 texture data in

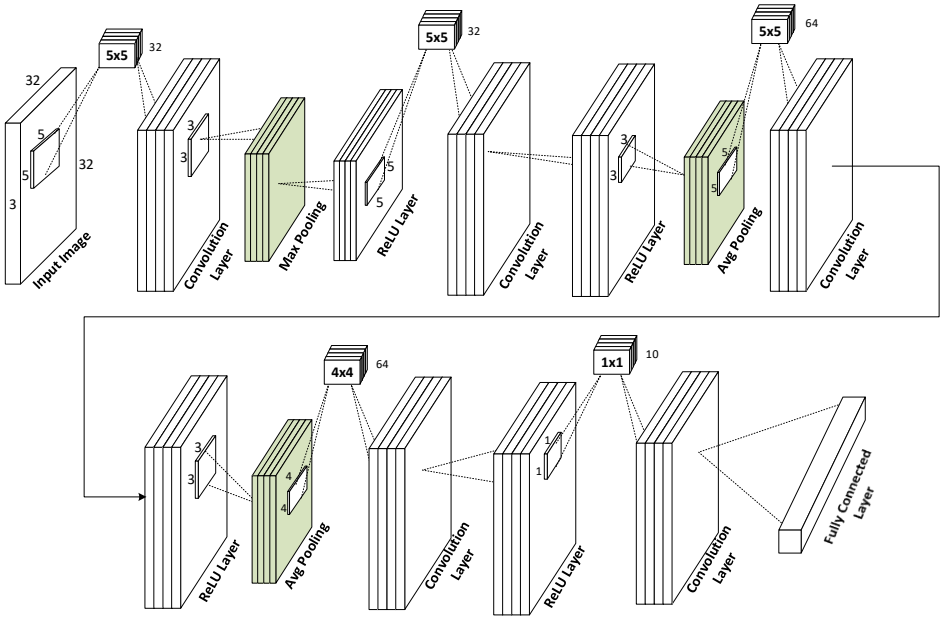

Fig. 6 *Original CNN model in Matconvnet for CIFAR-10 dataset.*

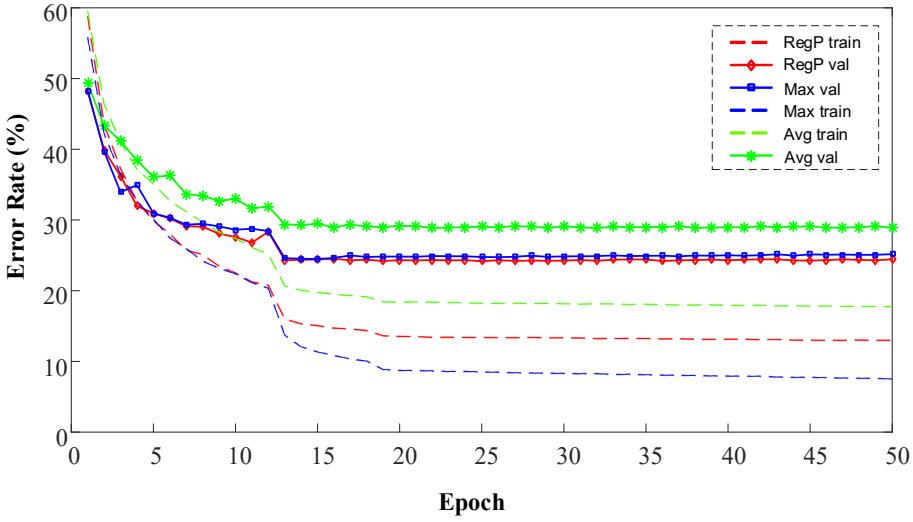

Fig. 7 *Performance graphs of the pooling methods on the CIFAR-10 dataset.*

the dataset. 25 % of these textures ($20 \times 25 = 500$) were used during the training phase, and the remaining 75 % ($20 \times 75 = 1500$) were used for testing. Samples of Drexel texture images are shown in Fig. 8.

A new simple CNN model is designed to compare the performance of the three pooling methods on the Drexel texture dataset. This developed model has four

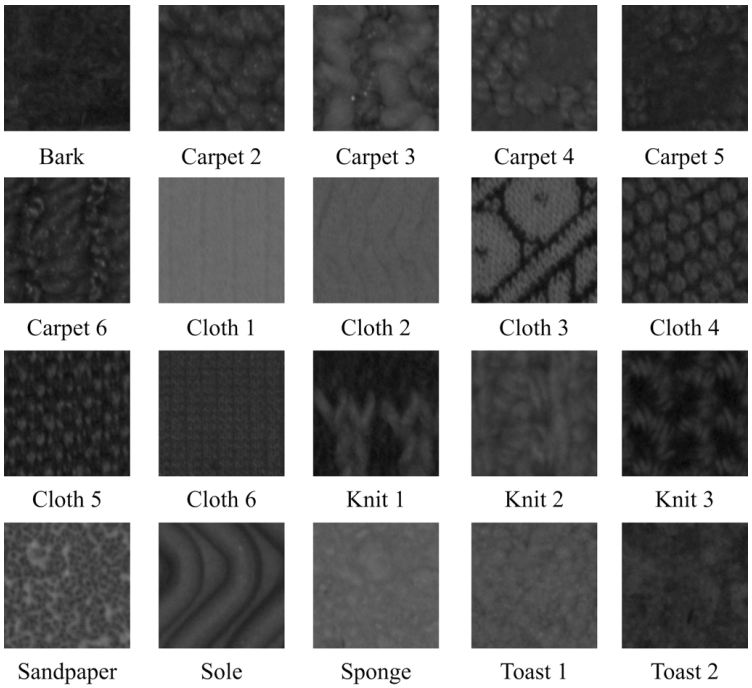

Fig. 8 *Examples of Drexel texture classes used in the study.*

layers, two of which are convolution layers. Remaining two layers are pooling layer and fully-connected layer. This CNN model developed for the Drexel texture dataset is shown in Fig. 9.

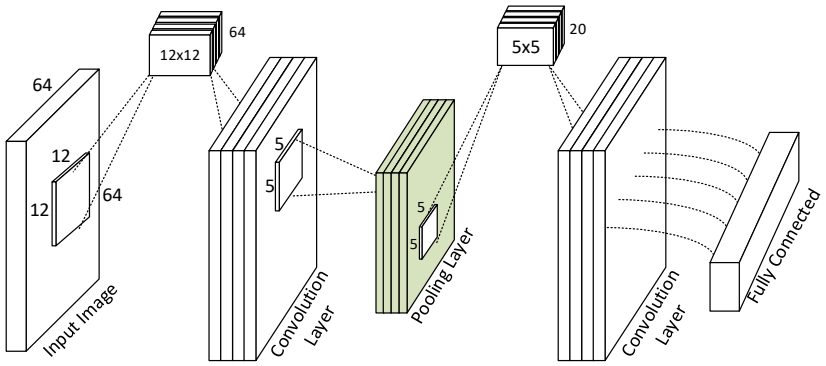

Fig. 9 *The developed CNN model for the Drexel texture dataset.*

The first layer is a convolution layer which has 64 kernels in 12×12 dimension. The following layer is a pooling layer in 5×5 dimension with a stride value 5. In the 3rd layer, there is again a convolution layer but having 20 kernels in 5×5 dimension. Finally, there is a fully-connected layer with softmax function. In this model,

only the pooling layer was changed to test each pooling method. Performance measurements for 100 epochs were obtained, and the results are given in Fig. 10.

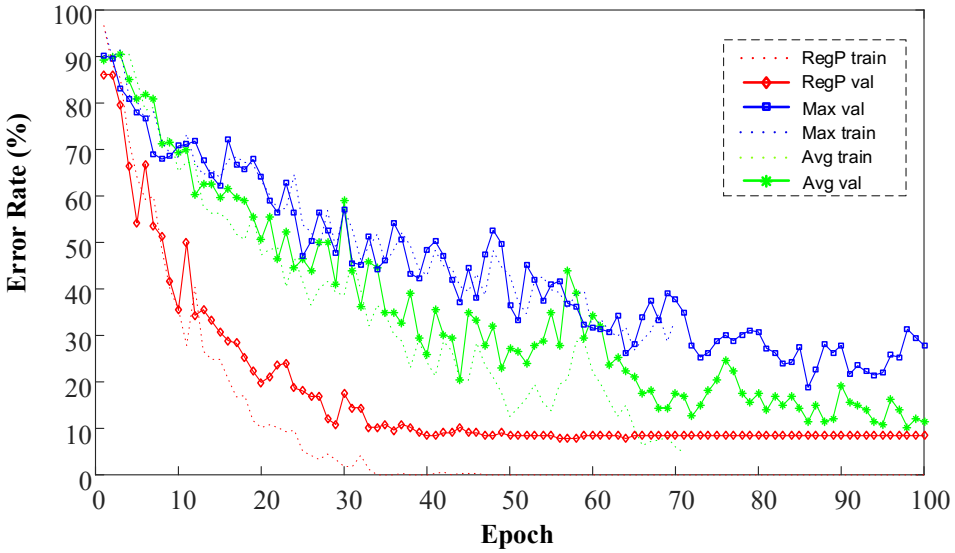

Fig. 10 *Performance graphs of the pooling methods on the Drexel dataset.*

The experimental results show that the proposed RegP pooling method provides a considerable advantage over other methods. The RegP pooling method has completed the learning process in a short period which is around 40 epochs. Average pooling, which gives the closest result to the RegP pooling, has finished the learning process around 100 epochs. Performance values of all pooling methods for various epoch ranges are given in Tab. III.

The classification of the test data was performed after completion of the CNN model's training. CNN model is separately trained for each pooling method, and classification results are given in Tab. IV. As it can be seen from these test results, the CNN model with the proposed RegP pooling method showed the highest performance with 96.80 %. The second successful CNN model was using the Average pooling, and it has 94.86 % success on the test data. The lowest performance is seen on the model with Max pooling method, which is 83.13 %.

5. Conclusions

In this paper, we propose a new pooling method, named as RegP, to be used in deep CNNs. Although the pooling layer seems less attractive for researchers compared with other layers, we can still obtain recognition and performance improvements. The proposed method has the same computational complexity with hand-crafted Average pooling and Max pooling methods. Similar to these methods, the proposed method is easy to implement, and very efficient in texture recognition tasks. We show that the RegP pooling is superior to other hand-crafted pooling methods

Textures	Max Pooling			Avg Pooling			RegP Pooling		
	True	False	Mean [%]	True	False	Mean [%]	True	False	Mean [%]
Bark	75	0	100	75	0	100	75	0	100
Carpet1	63	12	84	74	1	99	75	0	100
Carpet2	75	0	100	60	15	80	64	11	85
Carpet3	60	15	80	75	0	100	75	0	100
Carpet4	6	69	8	75	0	100	75	0	100
Carpet5	56	19	75	53	22	71	65	10	87
Cloth1	75	0	100	75	0	100	75	0	100
Cloth2	22	53	29	75	0	100	75	0	100
Cloth3	75	0	100	59	16	79	62	13	83
Cloth4	75	0	100	75	0	100	75	0	100
Cloth5	75	0	100	74	1	99	75	0	100
Cloth6	75	0	100	75	0	100	75	0	100
Knit1	53	12	71	75	0	100	75	0	100
Knit2	56	19	75	74	1	99	75	0	100
Knit3	46	29	61	59	16	79	64	11	85
Sandpaper	75	0	100	75	0	100	75	0	100
Sole	60	15	80	75	0	100	75	0	100
Sponge	75	0	100	71	4	95	72	3	96
Toast1	75	0	100	75	0	100	75	0	100
Toast2	75	0	100	74	1	99	75	0	100
	Overall Success = 83.13 %			Overall Success = 94.86 %			Overall Success = 96.80 %		

Tab. IV *Performance comparison of pooling methods for the Drexel texture dataset.*

on the Drexel texture dataset and CIFAR-10 dataset. The main reason for the improved performance could be the ability to construct a better relation to pixels with the surrounding pixels which would be fruitful for the feature extraction task. The popular hand-crafted pooling methods do not construct such relations during their operations which is found very problematic and meaningless by some researchers. This problem already led to the development of new neural network structures such as capsule networks.

The proposed RegP pooling method showed an acceptable performance on the MNIST dataset, where the data is very suitable for the success of the Max pooling. Due to its simplicity and efficiency, the proposed method can potentially be applied in many other deep learning studies. Despite the rapid progress in deep learning research, this study has revealed that the overall performance of the proposed CNN models could be improved with layer tunings. In future, we will apply the proposed method on domain-specific problems and investigate parameter optimizations for neighborhood similarity values.

Acknowledgement

All funding sources have been acknowledged.

References

[1] AHMAD J., MEHMOOD I., BAIK S.W. Efficient object-based surveillance image search using spatial pooling of convolutional features, *Journal of Visual Communication and Image Representation*, 45, 2017, pp. 62–76. doi: 10.1016/j.jvcir.2017.02.010.

[2] BENGIO Y. Learning deep architectures for AI, *Foundations and trends in Machine Learning*, 2(1), 2009, pp. 1–127. doi: 10.1561/2200000006.

[3] BENGIO Y., LEE H. Editorial introduction to the neural networks special issue on deep learning of representations, *Neural Networks*, 64(C), 2015, pp. 1–3. doi: 10.1016/j.neunet.2014.12.006.

[4] CIRESAN D.C., MEIER U., GAMBARDELLA L.M., SCHMIDHUBER J. Convolutional neural network committees for handwritten character classification, In: *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 1135–1139. doi: 10.1109/ICDAR.2011.229.

[5] FERNANDO B., GAVVES E., ORAMAS J., GHODRATI A., TUYTELAARS T. Rank pooling for action recognition, *IEEE transactions on pattern analysis and machine intelligence*, 39(4), 2017, pp. 773–787. doi: 10.1109/TPAMI.2016.2558148.

[6] GONG Y., WANG L., GUO R., LAZEBNIK S. Multi-scale orderless pooling of deep convolutional activation features, In: *European conference on computer vision*, 8695, 2014, pp. 392–407. doi: 10.1007/978-3-319-10584-0_26.

[7] GUO L., DING S. A hybrid deep learning cnn-elm model and its application in handwritten numeral recognition, *Journal of Computational Information Systems*, 11(7), 2015, pp. 2673-2680. doi: 10.12733/jcis13987.

[8] 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, pp. 1904–1916. doi: 10.1109/TPAMI.2015.2389824.

[9] KARPATHY A., TODERICI G., SHETTY S., LEUNG T., SUKTHANKAR R., FEI-FEI L. Large-scale video classification with convolutional neural networks, In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732. doi: 10.1109/CVPR.2014.223.

[10] KIM S., CHOI Y., LEE M. Deep learning with support vector data description, *Neurocomputing*, 165, 2015, pp. 111–117. doi: 10.1016/j.neucom.2014.09.086.

[11] KRIZHEVSKY A., HINTON G. Learning multiple layers of features from tiny images, Technical report, University of Toronto, 2009.

[12] LE ROUX N., BENGIO Y. Representational power of restricted Boltzmann machines and deep belief networks, *Neural computation*, 20(6), 2008, pp. 1631–1649. doi: 10.1162/neco.2008.04-07-510.

[13] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER, P. Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, 86(11), 1998, pp. 2278–2324. doi: 10.1109/5.726791.

[14] LI T., MENG Z., NI B., SHEN J., WANG M. Robust geometric ℓ p-norm feature pooling for image classification and action recognition, *Image and Vision Computing*, 55(2), 2016, pp. 64–76. doi: 10.1016/j.imavis.2016.04.002.

[15] LIU L., SHEN C., VAN DEN HENGEL A. Cross-convolutional-layer pooling for image recognition, *IEEE transactions on pattern analysis and machine intelligence*, 39(11), 2017, pp. 2305–2313. doi: 10.1109/TPAMI.2016.2637921.

[16] LIU L., SHEN C., WANG L., VAN DEN HENGEL A., WANG C. Encoding high dimensional local features by sparse coding based fisher vectors, In: *Advances in neural information processing systems*, 2014, pp. 1143–1151.

[17] LIU Y., ZHANG Y.M., ZHANG X.Y., LIU C.L. Adaptive spatial pooling for image classification, *Pattern Recognition*, 55, 2016, 58–67. doi: 10.1016/j.patcog.2016.01.030.

[18] MA B., HU H., SHEN J., LIU Y., SHAO L. Generalized pooling for robust object tracking, *IEEE Transactions on Image Processing*, 25(9), 2016, pp. 4199–4208. doi: 10.1109/TIP.2016.2588329.

[19] NGO T.A., LU Z., CARNEIRO G. Combining deep learning and level set for the automated segmentation of the left ventricle of the heart from cardiac cine magnetic resonance, *Medical image analysis*, 35, 2017, pp. 159–171. doi: 10.1016/j.media.2016.05.009.

[20] NIU X.X., SUEN C.Y. A novel hybrid CNN–SVM classifier for recognizing handwritten digits, *Pattern Recognition*, 45(4), 2012, pp. 1318–1325. doi: 10.1016/j.patcog.2011.09.021.

[21] OXHOLM G., BARIYA P., NISHINO K. The scale of geometric texture, *In: European conference on computer vision*, 7572, 2012, pp. 58–71. doi: 10.1007/978-3-642-33718-5_5.

[22] SCHMIDHUBER J. Deep learning in neural networks: An overview, *Neural networks*, 61, 2015, pp. 85–117. doi: 10.1016/j.neunet.2014.09.003.

[23] SHEN F., SHEN C., ZHOU X., YANG Y., SHEN H.T. Face image classification by pooling raw features, *Pattern Recognition*, 54, 2016, pp. 94–103. doi: 10.1016/j.patcog.2016.01.010.

[24] SHI Z., YE Y., WU Y. Rank-based pooling for deep convolutional neural networks, *Neural Networks*, 83, 2016, pp. 21–31. doi: 10.1016/j.neunet.2016.07.003.

[25] SILVER D., et al. Mastering the game of Go with deep neural networks and tree search, *Nature*, 529, 2016, pp. 484–489. doi: 10.1038/nature16961.

[26] SUN M., SONG Z., JIANG X., PAN J., PANG Y. Learning pooling for convolutional neural network, *Neurocomputing*, 224, 2017, pp. 96–104. doi: 10.1016/j.neucom.2016.10.049.

[27] VARDHANA M., ARUNKUMAR N., LASRADO S., ABDULHAY E., RAMIREZ-GONZALEZ G. Convolutional neural network for bio-medical image segmentation with hardware acceleration, *Cognitive Systems Research*, 50, 2018, pp. 10–14. doi: 10.1016/j.cogsys.2018.03.005.

[28] VEDALDI A., LENC K. Matconvnet: Convolutional neural networks for matlab, In: *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 689–692. doi: 10.1145/2733373.2807412.

[29] WANG J., WANG W., WANG R., GAO W. CSPS: An adaptive pooling method for image classification, *IEEE Transactions on Multimedia*, 18(6), 2016, pp. 1000–1010. doi: 10.1109/TMM.2016.2544099.

[30] XIE G.S., ZHANG Y., SHU X., YAN S., LIU C.L. Task-driven feature pooling for image classification, In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1179-1187. doi: 10.1109/ICCV.2015.140.

[31] ZEILER M.D., FERGUS, R. Stochastic pooling for regularization of deep convolutional neural networks, arXiv preprint: 1301.3557, 2013.