Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 23, No. 2, March 2024, pp. 343∼352
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 200/M/KPT/2020
DOI: 10.30812/matrik.v23i2.3763

Improving Convolutional Neural Networks Performance Using Modified Pooling Function

Achmad Lukman[1], **Wahju Tjahjo Saputro**[2], **Erni Seniwati**[3]
[1]Telkom University, Bandung, Indonesia
[2]Universitas Muhammadiyah Purworejo, Purworejo, Indonesia
[3]Universitas Amikom Yogyakarta, Yogyakarta, Indonesia

Article Info

Article history:

Received January 08, 2024
Revised January 23, 2024
Accepted March 01, 2024

Keywords:

Convolutional Neural Network
Visual Geometry Group-16
Qmax pooling function
Qavg pooling function

ABSTRACT

The Visual Geometry Group-16 (VGG16) network architecture, as part of the development of convolutional neural networks, has been popular among researchers in solving classification tasks, so in this paper, we investigated the number of layers to find better performance. In addition, we also proposed two pooling function techniques inspired by existing research on mixed pooling functions, namely Qmax and Qavg. The purpose of the research was to see the advantages of our method; we conducted several test scenarios, including comparing several modified network configurations based on VGG16 as a baseline and involving our pooling technique and existing pooling functions. Then, the results of the first scenario, we selected a network that can adapt well to our pooling technique, which was then carried out several tests involving the Cifar10, Cifar100, TinyImageNet, and Street View House Numbers (SVHN) datasets as benchmarks. In addition, we were also involved in several existing methods. The experiment results showed that Net-E has the highest performance, with 93.90% accuracy for Cifar10, 71.17% for Cifar100, and 52.84% for TinyImageNet. Still, the accuracy was low when the SVHN dataset was used. In addition, in comparison tests with several optimization algorithms using the Qavg pooling function, it can be seen that the best accuracy results lie in the SGD optimization algorithm, with 89.76% for Cifar10 and 89.06% for Cifar100.

Corresponding Author:

Achmad Lukman, +6281342352282,
Faculty of Informatics, Department of Information Technology,
Telkom University, Bandung, Indonesia,
Email: alukman@telkomuniversity.ac.id

How to Cite:
A. Lukman, W. Saputro, and E. Seniwati, ”Improving Convolutional Neural Networks Performance Using Modified Pooling Function”, *MATRIK: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, Vol. 23, No. 2, pp. 343-352, Mar, 2024.

1. INTRODUCTION

Currently, many studies have produced various ways to improve the performance of convolutional neural networks, namely by modifying the network architecture [1], finding the best architectural model to improve accuracy for large-scale image classification, and producing a new architectural model called VGG networks, and also other research related to it, namely the inception network [2], DenseNet [3], and residual network (ResNet) [4]. Furthermore, other research related to network performance improvement is the dropout regularization approach, which is very effective in reducing the overfitting of neural networks [5], and research on improving dropout performance by offering sparsity in the weights rather than the output vector of a layer using generalized dropout [6]. Then, in terms of pooling function modification, a new pooling method called max-min pooling is proposed [7], which explores the positive and negative responses of the resulting convolutional process. In contrast, another approach, mixed pooling operation [8], the pooled map response, is selected probabilistically through sampling from a multinomial distribution formed from the activation of each pooling region. Traditionally, convolutional neural networks use one of the two well-known pooling functions to improve the accuracy of the network, but research [9] revealed that the selection of one of the two pooling functions can be used based on the problem at hand. For this reason, they proposed a linear combination of the two pools in a convolutional neural network architecture called "CombPool." Their results show that their proposed method can outperform existing pooling function methods, including the traditional max pooling and average pooling functions. Some existing research has applied a combination of traditional pooling functions because the shortcomings of the two pools are that the max pooling function only relies on the maximum value in the feature map area. The problem occurs when most of the area has a high magnitude, resulting in the loss of information that stands out in the region. In contrast, average pooling has shortcomings when the feature map area mostly has a value of zero, so the results obtained make the convolution results worse [10]. **This study aims** to find a way to overcome the shortcomings pointed out by previous studies by modifying the existing pooling function.

To overcome the problem, we developed double max pooling and double average pooling methods inspired by the mixed pooling function method offered by [11], the same approach also provided by [8], and we also developed a method similar to paper [9] but in a different way, which combines max pooling with average pooling so that it outperforms the error rate of conventional pooling (e.g., max and average pooling) on CIFAR and SVHN datasets. Then, we also investigate the network architecture model that we modified from the VGG16 network [12], which is the baseline network. Then, we use five modified network architectures from the baseline derivative to find the right match for our method. For more details, **our contributions** can be listed as follows.

a. Modify the VGG16-based network architecture with several variations and compare the performance results for classification tasks. Then, choose the best network architecture to adapt to our pooling function method.
b. We propose a double max and double average pooling function method, as well as a combination of our method with max and average pooling, to see its response to improving the performance of convolutional neural networks.
c. Investigate the best network architecture among the offered variations by testing with Cifar10 and Cifar100 datasets and comparing the results with the baseline (VGG16) and existing state-of-the-art methods.
d. Comparing the accuracy results of the selected network architecture with the Cifar-VGG architecture and combining the network architecture with the pooling function method. It also involves TinyImageNet and SVHN datasets as benchmarks to show the performance of our pooling technique and network architecture.

Comparing the accuracy results of the selected network architecture with the Cifar-VGG architecture and combining the network architecture with the pooling function method. It also involves TinyImageNet and SVHN datasets as benchmarks to show the performance of our pooling technique and network architecture.

The max pooling method works by selecting the largest element in each pooling region Rij, where Rij represents the local neighborhood around position (i,j) that shows in both equations (1) and (2):

$$f(x)_{pq} = max_{(p,q)} \in R_{ij}.X_{cpq} \tag{1}$$

Meanwhile, average pooling uses the average value of the elements in each pooling region.

$$f(x)_{pq} = \frac{1}{\mid R_{ij} \mid} \sum_{(p,q)\in R_{ij}} x_{cpq} \tag{2}$$

Where $\mid Rij \mid$ is the size of the pooling region Rij and Xcpq is the element at (p, q) in the pooling region Rij. In gated pooling [11], a gating mask is used to determine the mixed proportion of max-average pooling to improve the learning response of the processed image features and improve the accuracy of the convolutional neural network. This differs from research [11], where we offer double max pooling and double average pooling methods with formulas similar to gated pooling. Each double max pooling function method contains two max pooling methods whose results will be summed up to get the Qmax pooling function result.

Likewise, the scheme we use to get the Qavg pooling function value contains two average pooling function combinations whose schemes can be seen in Figure 1.

Next, we will present related work on CNN configuration development in session 2, which we developed based on the VGG16 CNN network architecture. Then, in session 3, we will present in detail our proposed method, including our network architecture and the experimental setup results, including datasets, training details, and evaluation, which are discussed in session 4. And finally, in section 5, we will summarize our experimental results and possible future developments.

2. RESEARCH METHOD

The method stages used in this research are double max and double average pooling, combining the Proposed Method with the Existing Pooling Function and Proposed network architectures.

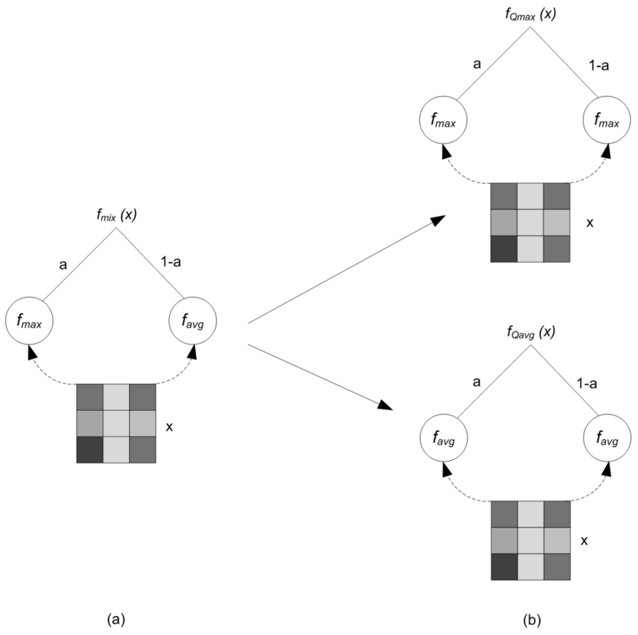

Figure 1. Pooling configuration of (a) a combination of max and average pooling called "mixed pooling"[11] and (b)modification of mixed pooling by breaking it into two, namely the combination of two maxpooling "$fQmax$."

called double max pooling, and the combination of two average pooling "$fQavg$" is called double average pooling.

2.1. Double Max and Double Average Pooling

The pooling operation often used in CNNs as a default is max pooling, which is also often used after max pooling, which is average pooling. Conventionally, the formula used to maximize the pooling operation is called max pooling $f_{max}(x) = max_j.x_j$ or to get the average value of the pooling operation using average pooling $f_{avg}(x) = \frac{1}{N}\sum_{j=1}^{N} x_j$ which activation value is included in the vector x of N pixels of a local pooling region. Previous research [11] used a combination of average pooling and max pooling, which they called mixed pooling, the combination of which is shown in Figure 1.

Based on Figure 1, we make modifications by combining two max pooling and two average pooling using the equations (3) and (4).

$$f_{Qmax}(x) = a_1.f_{max}(x) + (1 - a_1).f_{max}(x) \tag{3}$$

$$f_{Qavg}(x) = a_1.f_{avg}(x) + (1 - a_1).f_{avg}(x) \tag{4}$$

Where a is the activation function, and the subscript l is used "one per layer."

2.2. Combining the Proposed Method with the Existing Pooling Function

The second scenario combines the double max pooling function with the existing max pooling and double average pooling with average pooling, respectively. The scheme is shown in Table 1.

Table 1. Scheme of Pooling Configuration

LP1	**LP2**
PL1 = Qavg pooling	PL1 = Qmax pooling
PL2 = avg pooling	PL2 = max pooling
PL3 = Qavg pooling	PL3 = Qmax pooling
PL4 = avg pooling	PL4 = max pooling
PL5 = Qavg pooling	PL5 = Qmax pooling

Based on Table 1, the pooling layers marked PL1, PL2, PL3, PL4, and PL5 will be filled with combinations in Table 2 to see the effect of our pooling function on improving the performance of several convolutional neural network architectures we defined. The results of this combination will be shown in the experiments and results section.

2.3. Proposed Network Architectures

In this test, we also use our proposed network architecture, as shown in Table 1, where Net A is the VGG 16 architecture [12] as the baseline, while Net B, Net C, Net D, and Net E are the combinations that we use to test the performance of our method (sec. 3.1).

Table 2. Convolutional neural networks configuration, Net A as a baseline VGG-16. We use the variance pooling function in pooling*

Net A	Net B	Net C	Net D	Net E	Net F
Input (32 x 32 x 3)					
Conv3 64	Conv3 64	Conv3 64	Conv3 64	Conv3 64	Conv3 64
Conv3 64	Conv3 64	Conv3 64	Conv3 64	Conv3 64	
Pooling*(PL1)					
Conv3 -128	Conv3 -128	Conv3 -128	Conv3 -128	Conv3 -128	Conv3 -128
Conv3 -128	Conv3 -128	Conv3 -128	Conv3 -128		
Pooling*(PL2)					
Conv3 256	Conv3 256	Conv3 256	Conv3 256	Conv3 256	Conv3 256
Conv3 256	Conv3 256	Conv3 256	Conv3 256	Conv3 256	Conv3 256
Conv3 256	Conv3 256				
Pooling*(PL3)					
Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512
Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512
Conv3 512	Conv3 512	Conv3 512		Conv1 512	Conv1 512
Pooling*(PL4)					
Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv3 512
Conv3 512	Conv3 512	Conv3 512	Conv3 512	Conv1 512	Conv3 512
Conv3 512				Conv3 512	Conv1 512
Pooling*(PL5)					
FC-512					
FC-512					
FC-10/ FC-100					
soft-max					

Based on Table 2, we use a 3232 input dimension with three channels, namely R, G, and B. Then we map the pooling function variably using max pooling, average pooling, Qmaxpooling (ours), and Qavgpooling to see the effectiveness of the VGG architecture variation against the baseline which is VGG-16 architecture on the accuracy results shown. Among the architectures, Net B, Net C, Net D, Net E, and Net F differ in layer reduction in each part of the convolution layer, resulting in different parameter variations shown in Table 4.

2.4. Experimental Setup

This research uses a personal computer equipped with an Intel i3 processor with 12 GB RAM. To speed up the computational process of training and testing convolutional neural networks, we use Invidia GTX 1080 VGA 8Gb memory so that the training process does not burden the computer memory much. The software used is Windows 8.1 OS and Python 3.1 programming with the hard TensorFlow library 1.1.14, along with built-in functions that support deep learning technology to run the algorithm code we offer.

1. Dataset

In this study, we used the Cifar dataset [13] with dimensions of 32 32 consisting of Cifar10, which has ten classes with a total of 60,000 images; Cifar100, which consists of 100 classes with a total of 50,000 images for training and 10,000 for testing. In addition, the TinyImageNet dataset [14] which contains 200 classes with a total of 120,000 images with a size of 64 64, and we also involve the SVHN dataset, which consists of 73,257 digits images with a size of 32 32 to see the reliability of our method and network. We will conduct a systematic testing process where we try to find the appropriate parameters to improve the performance of our network.

3. RESULT AND ANALYSIS

We conducted experiments using a convolutional neural network with VGG16 architecture as the baseline. Then we modified the architecture with several different configurations, as listed in Table 2, as well as several parameters and pooling functions used. In this section, we start by running the training and testing process for all the network configurations we offer (see Table 2) with variations in the pooling method, followed by testing with variations in batch size. In the third section, we run tests with variations in the training algorithm when using the two pooling functions we offer. Then we show the results of tests with combinations of our methods with max pooling and average pooling functions.

3.1. Choosing a Better Network Configuration

Table 4 shows the training and testing results on the network configuration variations that we offer based on the VGG16 architecture and its variations using the cifar10 and cifar100 datasets. This test involves the pooling functions max pooling, average pooling, gated pooling, mixed pooling, and two pooling functions we offer. This experiment uses the Stochastic gradient descent (SGD) training algorithm, a learning rate of 10-5, and a batch size of 128. Net A, VGG16 as a baseline, shows the best training accuracy results on average pooling function and very good testing accuracy on the cifar10 dataset at 91.33%, followed by Qavg pooling with testing results reaching 67.23%.

Furthermore, Net B with architecture VGG-15 shows the highest testing accuracy results on the average pooling function, with 60.87% on cifar100 and 90.10% on cifar10, for Net C with architecture VGG14, which achieves the same performance as Net B with the highest accuracy falling on the average pooling function but slightly lower accuracy than Net B. While Net D shows good accuracy of the two existing pooling functions, where maxpooling is superior on the cifar10 dataset. In comparison, average pooling has better accuracy on the cifar100 dataset but is still slightly lower than Net C's achievement. On Net E, the highest accuracy results fall on our proposed method, namely the Qavg pooling function with the highest accuracy for all results from this network configuration variation, which is 67.89% on the cifar100 dataset. Still, on cifar10, the accuracy is not higher than Net A. Finally, Net F, whose highest accuracy falls on the max pooling function on cifar100, whose accuracy is still lower than Net E.

Based on the results from Table 2, we choose Net E with 200 epochs, and then we investigate which one is better than minibatch, which can improve the performance shown in Table 3.

3.2. Comparing the Result of Batch Size Variation

We conducted tests with several batch size variations to see the effect of accuracy using the pooling function, which can be seen in Table 3. In this experiment, we used Net E because the evaluation results obtained were more adaptable to our pooling technique than other network configurations.

Table 3 shows the effect of batch size on the accuracy of testing the Net E network architecture we chose. When the batch size is small, namely 64, the accuracy obtained is quite good, 88.93% for cifar10 with Qmax pooling and 89.24% for Qavg pooling function, while in testing Cifar100 using Qmax pooling function obtained 66.06% and on Qavg pooling function the accuracy obtained is

66.52%. Then we added batch sizes to 128, 256, 512, and 1024. It can be seen that the best accuracy is obtained at a batch size of 128, namely using Qmax pooling with the Cifar10 dataset with an accuracy of 89.13%, while cifar100 accuracy reaches 66.57%. The accuracy of the Qavg pooling function with the same batch size is 89.89% for cifar10 and 67.89% for cifar100.

Table 3. Top-1 accuracy (%) comparing the result with different batch size configuration using proposed method

Batch size	#Epochs	Qmax pooling				Qavg pooling			
		Cifar 10		Cifar 100		Cifar 10		Cifar 100	
		Training	Testing	Training	Testing	Training	Testing	Training	Testing
64	200	99.91	88.93	99.04	66.06	99.89	89.24	99.04	66.52
128	200	99.94	89.13	99.53	66.57	99.95	89.89	99.58	67.89
256	200	99.89	85.07	99.21	61.68	99.95	89.28	99.51	63.51
512	200	99.95	88.76	98.81	62.28	99.93	88.20	98.67	60.04
1024	200	99.90	87.79	99.04	62.39	99.96	88.17	99.44	63.81

Table 4. -1 accuracy(%) network performance compare with another pooling method with epoch 200

Network conf. (table 1)	**#Params for Cifar10/Cifar100**	**Method**	**Training**		**Testing**	
			Cifar 10	**Cifar 100**	**Cifar 10**	**Cifar 100**
Net A	15,001,418/ 15,047,588	max pooling (baseline)	99.82	98.84	80.15	59.33
		Qmax pooling (ours)	99.89	98.58	80.64	59.27
		Mixed pooling [11]	99.87	98.42	88.93	59.35
		Gated pooling [11]	99.39	92.75	84.41	53.22
		average pooling	**99.92**	98.61	**81.33**	59.42
		Qavg pooling (ours)	99.88	**99.06**	80.94	**67.23**
Net B	12,639,562/ 12,685,732	max pooling	99.86	98.94	87.74	59.83
		Qmax pooling (ours)	99.89	99	88.79	60.7
		Mixed pooling [11]	99.89	98.16	**89.9**	55.21
		Gated pooling [11]	99.57	94.2	87.44	60.57
		average pooling	**99.88**	**98.83**	89.1	**60.87**
		Qavg pooling (ours)	99.91	98.76	89.43	55.28
Net C	12,048,458/ 12,094,628	max pooling	99.91	99.35	88.15	65.54
		Qmax pooling (ours)	99.91	99.37	89.32	65.84
		Mixed pooling [11]	99.93	99.35	89.06	66.78
		Gated pooling [11]	99.66	94.36	86.23	61.13
		average pooling	**99.92**	**99.48**	**90.05**	**67.79**
		Qavg pooling (ours)	99.92	99.39	89.86	65.7
Net D	9,686,602/ 9,732,772	**max pooling**	**99.96**	99.64	**90.35**	64.76
		Qmax pooling (ours)	99.95	99.66	89.59	65.74
		Mixed pooling [11]	99.92	99.65	85.58	65.59
		Gated pooling [11]	99.62	95.65	87.1	61.33
		average pooling	99.94	**99.69**	89.5	**66.06**
		Qavg pooling (ours)	99.93	99.69	85.8	66.02
Net E	**14,262,218/ 14,308,388**	max pooling	99.96	99.5	89.36	66.82
		Qmax pooling (ours)	99.94	99.53	89.13	66.57
		Mixed pooling [11]	99.94	99.22	89.6	64.18
		Gated pooling [11]	99.59	96.5	86.46	61.83
		Avg pooling	99.95	99.55	89.55	67.75
		Qavg pooling (ours)	**99.95**	**99.58**	**89.89**	**67.89**
Net F	14,225,034/ 14,271,204	**max pooling**	99.95	**99.53**	87.97	**64.73**
		Qmax pooling (ours)	99.96	99.57	**88.06**	64.17
		Mixed pooling [11]	99.94	99.59	87.58	62.87
		Gated pooling [11]	99.76	97.16	86.03	61.07
		Avg pooling	99.93	99.52	87.61	63.99
		Qavg pooling (ours)	99.93	99.55	86.86	63.27

Furthermore, along with the addition of batch size, it turns out that the accuracy results tend to decrease. This states that the effect of a larger batch size will increase the loss function value so that it decreases the resulting accuracy, but it should be noted that the selection of batch size must also be adjusted to the size of the dataset used. Furthermore, we show the test results with several optimization algorithms, as listed in Table 3.

3.3. Find a Better Optimization Algorithm

Based on the network performance generated in Table 3, each of which uses Net E to show the comparison of the accuracy results of our two pooling function methods, it can be concluded that the performance of Qavg pooling function is better than Qmax pooling, so we choose Qavg pooling function for further experiments. Table 5 shows the evaluation results of several optimization algorithms in adapting to the Qavg pooling function using 200 epochs. Table 5 compares the test results using each optimization algorithm to see how well our method adapts to these algorithms. We will then select the best of the results from this testing stage to use in subsequent experiments. It can be seen that the best adaptation to our method is SGD (stochastic Gradient descent), with the highest accuracy of 89.76% for Cifar10 and 89.06% for Cifar100 datasets.

Table 5. Comparing to another optimization algorithm using Qavg pooling

optimization algorithm	#Epochs	Cifar 10		Cifar 100	
		Training	Testing	Training	Testing
Adam	200	99.91	88.44	98.47	61.26
SGD	200	99.96	**89.76**	99.85	**89.06**
RMSProp	200	99.82	87.76	94.86	58.95
Nadam	200	99.92	87.90	98.28	60.89
Adamax	200	99.91	86.42	98.73	60.52

3.4. The Effect of Combining the Proposed Methods

We carried out the test using a combination of our two methods. We used several network configurations (see Table 1) and compared them with Net A, which is VGG16 as a baseline that, by default, uses the Maxpooling function. The evaluation results are presented in two tables using the Cifar10 and Cifar100 datasets seen in Tables 6 and 7, respectively.

Table 6. Top-1 (% accuracy) Comparing results proposed method and baseline on Cifar10 datasets.

Table config.	**#Params**	**Testing**				
		Max pooling	**Qavgpooling**	**Qmaxpooling**	**LP1**	**LP2**
Net A (baseline)	14M	86.94	-	-	-	-
Net B	12.9M	-	86.23	85.90	86.49	85.71
Net C	12.3M	-	87.59	86.57	87.16	85.33
Net D	9.9M	-	86.55	87.32	87.27	86.20
Net E	14M	-	88.10	87.74	86.24	88.16
Net F	14M	-	81.79	84.49	83.52	84.19

The evaluation seen in Table 6 shows that the accuracy of Net B cannot outperform the baseline that uses the max pooling function as the default configuration of VGG16 (Net A). In contrast, Net C shows better accuracy than Net A as a baseline and excels at using Qavg pooling function and LP1 pooling combination. In Net D, the accuracy shows a good accuracy trend, especially in the use of Qmaxpooling pooling function and LP1 pooling combination with 87.32% and 87.27%, respectively, but still tends to be lower than the accuracy of Net C, which uses Qavgpooling pooling function which reaches 87.59%. Then a very good accuracy trend is shown by Net E, which can outperform the accuracy of all performance network configurations involved in this section, namely 88.16% when using the LP2 combination, followed by when using the Qavgpooling function with accuracy reaching 88.10%, and when using the Qmaxpooling function which reaches 87.74% accuracy. Then, the accuracy shown for Net F cannot outperform the baseline.

Table 7. Top-1 (% accuracy) Comparing results between baseline and proposed method on cifar 100 datasets

Table config.	#Params	Testing				
		Max pooling	Qavgpooling	Qmaxpooling	LP1	LP2
Net A (baseline)	14M	59.18	-	-	-	-
Net B	12.9M	-	58.88	55.82	60.21	57.79
Net C	12.3M	-	55.58	58.08	58.58	58.03
Net D	9.9M	-	57.36	58.78	55.74	58.95
Net E	14M	-	58.76	55.78	60.38	56.93
Net F	14.5M	-	48.31	55.39	46.99	53.37

The results of the accuracy comparison evaluation using the Cifar100 dataset are shown in Table 7, where, in general, the accuracy results that can beat the performance of Net A as a baseline are Net B using the LP1 combination with an accuracy of 60.21%, followed by the highest performance is Net E with an accuracy of 60.38%.

3.5. Comparison with Others

We conducted a comparison with several existing methods, including mixed pooling [11], gated pooling [11], All-CNN [15], and Network in Network [16], the results of which can be seen in Table 8. Table 9 presents the results of the comparison using Net E and architecture cifar-vgg [2] combined with pooling functions Qmax and Qavg.

In Table 8, the data augmentation test results show that Net E with the combination of LP1 outperforms the baseline and all methods used. Net E + LP1 gets 93.90% accuracy for Cifar10 and 71.10% for Cifar100 datasets. At the same time, the combination of Net E + LP2 and Net C + Qavg pooling can also outperform the baseline, namely getting 93.63% and 93.60% respectively on Cifar10 and also 69.76% and 69.81% respectively for Cifar 100 datasets. This shows that our method is more adaptable to Cifar datasets in general.

In Table 9, we compare several combinations of methods, including Net E + LP1 configuration, and use four datasets, namely Cifar 10, Cifar 100, Tiny Imagenet, and SVHN datasets, to show the adaptivity of our method. Net E combined with LPI showed superiority to other methods on Cifar 10, Cifar 100, and Tiny Imagenet datasets, but the accuracy was rather low when using the SVHN dataset. This is because our architecture cannot generalize well to the SVHN dataset with a dimension of 32 $\times$ 32.

Table 8. Comparison of the testing (%) results with several state-of-the-art on cifar 10 and cifar 100 datasets using Data Augmentation

Method	**Cifar 10**	**Cifar 100**
Baseline	93.58	68.75
Net E + Qavg pooling(ours)	93.6	69.81
Net E + Qmax pooling(ours)	93.28	69.73
Net E + LP1 (ours)	**93.9**	**71.1**
Net E + LP2 (ours)	93.63	69.76
Mixed pooling [11]	90.74	69.1
Gated pooling [11]	92.38	65.79
Network in Network (NiN) [16]	91.19	64.32
All-CNN [15]	92.75	66.29

Table 9. Comparison of the proposed method Testing (%) results with the state-of-the-art model on cifar 10, cifar 100, SVHN, and Tiny Imagenet datasets

No.	Method	Cifar 10	Cifar 100	Tiny Imagenet (200)	SVHN
1	Cifar-VGG [2]	90.94	67.02	51.83	96.18
2	Cifar-VGG [2]+Qmax(ours)	91.72	67.45	51.69	**96.38**
3	Cifar-VGG 2+ Qavg(ours)	91.81	67.75	52.03	95.21
4	**Net E + LP1 (ours)**	**93.9**	**71.1**	**52.84**	95.95

The results of this study found that the techniques we propose, namely Qavg pooling and Qmax pooling function, can relatively improve the shortcomings of the existing pooling functions, namely maxpooling and avg pooling, especially when we do a scheme of pooling configuration that consists of the combination between our pooling and existing pooling functions that the results can be shown in Table 6, 7, 8 and 9, that point out of our techniques superior to another.

4. CONCLUSION

This paper introduces two new pooling function techniques and five network architecture variations derived from the VGG16 architecture. Among the five architectures we offer, one option has a fairly good performance in adapting to the pooling function techniques we offer, namely Net E as a network configuration modified from VGG16 (Net A). In addition, the results of testing optimization algorithms that can be combined with one of the pooling functions we offer lie in the SGD optimization algorithm, which shows the highest accuracy of 89.76% for Cifar10 and 89.06% for Cifar100. In testing the comparison of network performance accuracy using two pooling functions and two combinations of pooling functions that we offer against the baseline shows that Net C, Net D, and Net E are more adaptable to the Cifar10 dataset by showing higher accuracy than the baseline and from other network configurations. While on the Cifar100 dataset, Net B and Net E show a trend of better accuracy when using the LP1 pooling function combination. Furthermore, to see the reliability of the method we offer by doing two scenarios, namely first comparing it with the existing pooling function method using data augmentation, it can be seen that the Net E + LP1 combination can outperform the methods' accuracy. In the second scenario, the result of performance comparison using VGG-cifar network architecture shows superior Net E + LP1 on three benchmarking datasets, namely Cifar10, Cifar100, and TinyImagenet, but tends to be low in accuracy when given the SVHN dataset. Based on the experiments in the results and analysis section, the accuracy obtained is not too high compared to the accuracy of the baseline model and other comparison models. For this reason, in future work, this research can still be developed with several combinations of existing methods, such as using stochastic pooling functions whose accuracy results in the paper are more promising.

5. ACKNOWLEDGEMENTS

This work was supported by Research and Community Service, Telkom university Bandung.

6. DECLARATIONS

AUTHOR CONTIBUTION

Conceptualization and methodology, Achmad lukman and Erni seniwati, implementation, Wahju Tjahjo saputro, writing-original draf preparation, Achmad lukman formal analysis, Achmad Lukman, investigation, Erni seniwati. All authors have read and agreed to the published version of the manuscript.

FUNDING STATEMENT

This research has not been funded.

COMPETING INTEREST

The authors declare no conflict of Interest.

REFERENCES

[1] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for Pruning Very Deep Convolutional Networks With Complicated Structure," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. June. IEEE, jun 2019, pp. 4938–4948, https://doi.org/10.1109/CVPR.2019.00508.

[2] X. Soria, E. Riba, and A. Sappa, "Dense Extreme Inception Network: Towards a Robust CNN Model for Edge Detection," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, mar 2020, pp. 1912–1921, https://doi.org/10.1109/WACV45572.2020.9093290.

[3] J. Hemalatha, S. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An Efficient DenseNet-Based Deep Learning Model for Malware Detection," *Entropy*, vol. 23, no. 3, pp. 1–23, mar 2021, https://doi.org/10.3390/e23030344.

[4] F. He, T. Liu, and D. Tao, "Why ResNet Works? Residuals Generalize," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5349–5362, dec 2020, https://doi.org/10.1109/TNNLS.2020.2966319.

[5] C. Wei, S. Kakade, and T. Ma, "The implicit and explicit regularization effects of dropout," in *37th International Conference on Machine Learning, ICML 2020*, 2020, pp. 10 181–10 192.

[6] X. Liang, L. Wu, J. Li, Y. Wang, Q. Meng, T. Qin, W. Chen, M. Zhang, and T. Y. Liu, "R-Drop: Regularized Dropout for Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 13, 2021, pp. 1–16.

[7] S. K. Roy, M. E. Paoletti, J. M. Haut, E. M. T. Hendrix, and A. Plaza, "A New Max-Min Convolutional Network for Hyperspectral Image Classification," in *2021 11th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, mar 2021, pp. 1–5, https://doi.org/10.1109/WHISPERS52202.2021.9483983.

[8] Q. Zhou, Z. Qu, and C. Cao, "Mixed pooling and richer attention feature fusion for crack detection," *Pattern Recognition Letters*, vol. 145, no. May, pp. 96–102, may 2021, https://doi.org/10.1016/j.patrec.2021.02.005.

[9] I. Rodriguez-Martinez, J. Lafuente, R. H. Santiago, G. P. Dimuro, F. Herrera, and H. Bustince, "Replacing pooling functions in Convolutional Neural Networks by linear combinations of increasing functions," *Neural Networks*, vol. 152, no. August, pp. 380–393, aug 2022, https://doi.org/10.1016/j.neunet.2022.04.028.

[10] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, and S. Almotairi, "A Comparison of Pooling Methods for Convolutional Neural Networks," *Applied Sciences*, vol. 12, no. 17, pp. 1–21, aug 2022, https://doi.org/10.3390/app12178643.

[11] C.-Y. Lee, P. Gallagher, and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 863–875, apr 2018, https://doi.org/10.1109/TPAMI.2017.2703082.

[12] H. Yang, J. Ni, J. Gao, Z. Han, and T. Luan, "A novel method for peanut variety identification and classification by Improved VGG16," *Scientific Reports*, vol. 11, no. 1, pp. 1–17, aug 2021, https://doi.org/10.1038/s41598-021-95240-y.

[13] M. S and E. Karthikeyan, "Classification of Image using Deep Neural Networks and SoftMax Classifier with CIFAR datasets," in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, may 2022, pp. 1132–1135, https://doi.org/10.1109/ICICCS53718.2022.9788359.

[14] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, pp. 1–6, 2015.

[15] M. R. Islam, D. Massicotte, and W.-P. Zhu, "All-ConvNet: A Lightweight All CNN for Neuromuscular Activity Recognition Using Instantaneous High-Density Surface EMG Images," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, may 2020, pp. 1–6, https://doi.org/10.1109/I2MTC43012.2020.9129362.

[16] J. Yamanaka, S. Kuwashima, and T. Kurita, "Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, pp. 217–225, https://doi.org/10.1007/978-3-319-70096-0_23.