A Systematic Evaluation of Shallow Convolutional Neural Network on CIFAR Dataset

Reza Fuad Rachmadi, *Member, IAENG,* I Ketut Eddy Purnama, *Member, IAENG,*
Mauridhi Hery Purnomo, *Member, IAENG,* Mochamad Hariadi, *Member, IAENG,*

***Abstract*—Convolutional Neural Network (CNN) classifier is a very popular classifier used to solve many problems, including image classification and object recognition. The CNN classifier usually improved by designing a deeper and bigger classifier which needs more memory and computational power to run the classifier. In this paper, we analyze and optimize the use of small and shallow CNN classifier on CIFAR dataset. Karpathy ConvNetJS CIFAR10 model was used as a base network of our classifier and extended by adding max-min pooling method. The max-min pooling is used to explore the negative and positive response of the convolution process which in theory will be trained the classifier more effectively. We choose several different configurations to analyze the effectiveness of the classifier by combining the training algorithm, batch normalization configuration, weights initialization methods, dropout regularization configuration, and heavy data augmentation. To ensure that the classifier we designed is still small and shallow CNN classifier, we limit the maximum number of layers in our CNN classifier to 15 layers. Experiments on CIFAR10 and CIFAR100 dataset shows that by compacting the kernel on each layer, the classifier can achieve good accuracy and comparable with another state-of-the-art classifier with a relatively same number of layers with an error rate of *6.99%* on the CIFAR10 dataset and *29.41%* on the CIFAR100 dataset.**

***Index Terms*—shallow CNN classifier, max-min pooling, deep convolutional neural network, CIFAR dataset**

I. Introduction

CONVOLUTIONAL Neural Network (CNN) classifier is very famous classifier used for many applications including image classification [1]–[7], video analysis [8]–[13], text analysis [14]–[16], and sound analysis [17]–[20]. The era of CNN was started when Krizhevsky et al. [2] won the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) in 2012 with large margin compared with handcrafted features approaches. After Krizhevsky et al. CNN approach, the researcher starts developing a bigger and deeper CNN classifier to achieve better evaluation score. Some of the CNN architectures approaches, including VGGNet [3], [21], inception network [4], [5], residual network (ResNet) [6], and DenseNet [7]. The disadvantages of all modern CNN architectures are the huge number of parameters of the classifier which leads to huge memory required to perform training and testing of the classifier. Many parameters of modern CNN architecture are inactive due to the use of the regularization method (such as dropout), bad weights initialization, or because of the use of ReLU (Rectified Linear Unit) activation function. The ReLU activation function is widely used in the modern CNN architectures and proved to be very effective, but in the other hand, the ReLU activation function also disrupted the flow of gradient in the backpropagation process which will make many parameters of the classifier inactive. Blot et al. [22] proposed a new pooling method by exploring either negative and positive response of the output of the convolution process called max-min pooling. Blot et al. [22] proved that the max-min pooling method can reduce the problems that appear when using ReLU activation function and produces a better accuracy compared with the classifier that uses ReLU activation function.

In this paper, we investigated several shallow CNN architecture for image classification on CIFAR dataset. The classifier is designed by exploiting either negative or positive output of convolution process by using max-min pooling method [22]. Our contributions can be listed as follows.

- We proposed a CNN architecture that exploiting either negative or positive respond of convolution output. The proposed classifier designed using max-min pooling method with additional normalization layer to reduce the number of parameters of the classifier. By using additional normalization layer, the number of parameters in the classifier is reduced half compared with the classifier using original max-min pooling method.
- We investigated several different CNN configuration for initial experiments and choose one of the configurations as the main CNN architecture for our proposed classifier.
- We investigated the effects of different weights initialization, regularization, and heavy data augmentation method using the main CNN architecture chosen in the initial experiments.

The rest of the paper organized as follows. Section 2 describes related work on CNN classifier development. The design aspect of our proposed CNN architecture which based on Karpathy ConvNetJS CIFAR10 model is discussed in section 3. Experiments on CIFAR10 and CIFAR100 dataset are discussed in section 4, 5, and 6. In the last section, we summarize and conclude the experiments.

Manuscript received May 17, 2018; revised Jan 28, 2019.

This work was partially supported by P3I (Program Percepatan Publikasi Internasional) grants, Institut Teknologi Sepuluh Nopember, Surabaya. Indonesia.

Reza Fuad Rachmadi is with Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia (email: fuad@its.ac.id).

I Ketut Eddy Purnama is with Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia (email: ketut@te.its.ac.id).

Mauridhi Hery Purnomo is with Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia (email: hery@te.its.ac.id).

Mochamad Hariadi is with Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia (email: hariadi@te.its.ac.id).

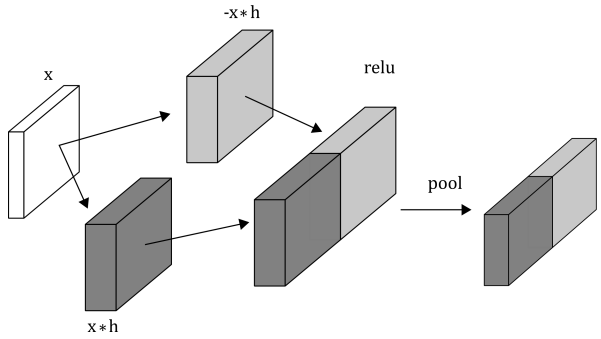


Fig. 1: The original architecture of max-min pooling method for exploiting either positive and negative output of convolution process (adapted from [22]).

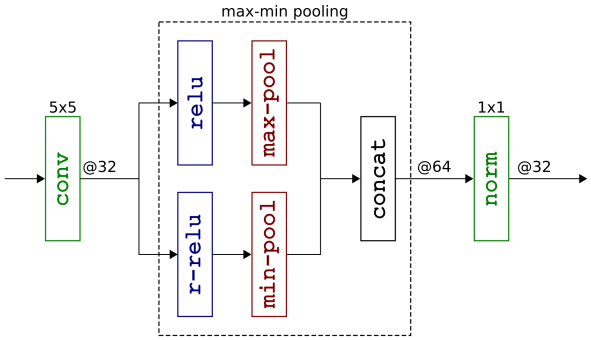


Fig. 2: The proposed extended max-min pooling to reduce the number of parameters by adding normalization layer at the end of the module. The diagram assumed that convolution kernel has 32 channel output.

II. Related Work

Convolutional Neural Network (CNN) classifier has been around for a couple years and widely used to tackle a lot of visual understanding problems, including image classification, image captioning, object detection, semantic scene segmentation, and object re-identification. In the last couple years, the number of parameters of CNN classifier is reduced by designing the network with fewer full-connection layer and added more convolutional layers to the classifier, but on the other hand, the number of layers of the classifier was increased rapidly. Does deeper CNN classifier produce better accuracy? The trend we see in the last couple of years shows that deeper classifier produced better accuracy. CNN classifier with more than 100 layers were designed by taking into consideration the flows of gradients from the end of the network to the first layer of the classifier. The flows of gradients are very important and in the case of the backpropagation, the gradients that arrived at the several first layers of the network may have very low value due to the chain rule used in the backpropagation process.

The first approaches of CNN architecture that considering the flows of gradients in the network was ResNet (Residual Network) which proposed by He et al. [6]. ResNet CNN architecture works by stacking the residual module which will add the features with residual factor computed in each module. Each residual module consists of several convolutional layers and a skip connection with a summing operation at the end of the module. By using the residual module, the ResNet CNN architecture provides a highway flows of gradients from the higher layer to several first layers of the network. The highway flows of the gradients are very efficient to train very deep CNN classifier. Although the solution is working very well, very deep CNN classifier is very difficult to paralleling the process in the cluster or chip due to the dependencies between layers.

The second approaches to optimizing the training process of CNN classifier is max-min pooling strategy which proposed by Blot et al. [22]. The analogy of the current CNN architecture is to search the features that have a connection with the problem definition. For example, if the classifier goal is to classify a person and non-person, the training process will search the optimal features to represent the person/non-person in the images. The ReLU (Rectified Linear Unit) is used to search the optimal features by selecting the positive features and penalize the negative features. Those operations may affect the flows of gradients from higher and create an inactive neuron in the current layer. Blot et al. [22] proposed a max-min pooling which exploiting either negative and positive output of convolution output. The proposed max-min pooling method proved more effective than the plain network with just ReLU and max-pooling layer. One of the disadvantages of max-min pooling method is that the output of the pooling has a double number of features which very difficult to use it for designing a deeper CNN classifier.

III. Proposed Classifier

In this section, we describe how the proposed classifier was designed. For the basis of the classifier, we use Karpathy ConvNetJS CIFAR10 Model [23] composed of three convolutional layers and one fully connected layer. As mentioned on their website [23], the accuracy of Karpathy ConvNetJS CIFAR10 Model is around 80%, which is a very good accuracy consider that the model only has around 20,000 parameters.

A. Max-Min Pooling

One of the important factors for CNN classifier is the choice of the activation function. In recent years, many activation functions were developed for CNN classifier including ReLU (Rectified Linear Unit) [24], Leaky ReLU [25], P-ReLU (Parametric ReLU) [26], and ELU (Exponential Linear Unit) [27]. All of those new activation function explores the positive output of the convolution process. Bolt et al. [22] proposed a method to explore either negative or positive output of the convolution process called max-min pooling method. Figure 1 shows the illustration of max-min pooling method. Let $\mathbf{X}$ an input of max-min pooling method, the output of max-min pooling can be described as follows

$$\mathbf{Y} = [ReLU(\mathbf{X} * \mathbf{h}), ReLU(-\mathbf{X} * \mathbf{h})] \quad (1)$$

where $\mathbf{h}$ is the kernel used in the convolution process and [..] denote the concatenation operation. The disadvantage of max-min pooling method is that the number of output channel is twice the number of the input channel. This behavior increases the number of parameters in the next layer rapidly compared with the standard pooling method.

TABLE I: Several different network configuration we used for the experiments on CIFAR10 and CIFAR100 dataset. The net-A configuration has the same configuration as the Karparthy ConvNetJS baseline but with additional max-min pooling. The "5×5@20" means that the convolution process is conducted with 20 of a 5×5 kernel, while "2×2s2" means that the pooling method is conducted using a 2×2 kernel with stride 2. The "fc-10" means full-connection layer with 10 output.

Group	net-A	net-B	net-C	net-D	net-E	net-F
group1	conv1 (5×5@16)	conv1 (5×5@16)	conv1 (5×5@32)	conv1 (5×5@32)	conv1 (5×5@64)	conv1 (5×5@64)
	mm-pool1 (2×2s2)					
	conv1-n (1×1@16)		conv1-n (1×1@32)		conv1-n (1×1@64)	
group2	conv2 (5×5@20)	conv2 (5×5@32)	conv2-1 (5×5@32)	conv2-1 (5×5@32)	conv2-1 (5×5@64)	conv2-1 (5×5@64)
			mm-pool2-1 (2×2s1)	*mm-pool2-1 (2×2s1)*	*mm-pool2-1 (2×2s1)*	*mm-pool2-1 (2×2s1)*
			conv2-1n (1×1@32)	conv2-1n (1×1@32)	conv2-1n (1×1@64)	conv2-1n (1×1@64)
			conv2-2 (5×5@32)	conv2-2 (5×5@32)	conv2-2 (5×5@64)	conv2-2 (5×5@64)
				mm-pool2-2 (2×2s1)		*mm-pool2-2 (2×2s1)*
				conv2-2n (1×1@32)		conv2-2n (1×1@64)
				conv2-3 (5×5@32)		conv2-3 (5×5@64)
	mm-pool2 (2×2s2)					
	conv2-n (1×1@20)	conv2-n (1×1@32)			conv2-n (1×1@64)	
group3	conv3 (5×5@20)	conv3 (5×5@32)	conv3-1 (5×5@32)	conv3-1 (5×5@32)	conv3-1 (5×5@64)	conv3-1 (5×5@64)
			mm-pool3-1 (2×2s1)	*mm-pool3-1 (2×2s1)*	*mm-pool3-1 (2×2s1)*	*mm-pool3-1 (2×2s1)*
			conv3-1n (1×1@32)	conv3-1n (1×1@32)	conv3-1n (1×1@64)	conv3-1n (1×1@64)
			conv3-2 (5×5@32)	conv3-2 (5×5@32)	conv3-2 (5×5@64)	conv3-2 (5×5@64)
				mm-pool3-2 (2×2s1)		*mm-pool3-2 (2×2s1)*
				conv3-2n (1×1@32)		conv3-2n (1×1@64)
				conv3-3 (5×5@32)		conv3-3 (5×5@64)
	mm-pool3 (2×2s2)					
	conv2-n (1×1@20)	conv3-n (1×1@32)			conv3-n (1×1@64)	
final	fc-10 / fc-100					

B. Design Goals

As mentioned in the previous sub-section, the max-min pooling method has disadvantages that the number of the pooling output has doubled channel compared with standard pooling. To reduce the number of parameters, we added a normalization layer by performing a convolution process using a 1×1 kernel. By setting the output channel of the normalization layer to the same number of the input channel, the max-min pooling process will produce the same number of output channel as the standard pooling method. Figure 2 shows the diagram of our extended max-min pooling module. We introduce r-ReLU (reversed ReLU) which an opposite version ReLU and min-pool which pooling the features using a minimum function instead of maximum function. The r-ReLU can be described as follows

$$f(x) = \min(0, x) \tag{2}$$

The r-ReLU function will support the minimum-pool layer which will explore the negative features of the output of the convolution process. Our version of max-min pooling is quite different, but by using our approach, the convolution process is only conducted one time which will decrease the computational power.

To test the classifier, we designed six different CNN classifier with very simple architecture. All of the configurations can be viewed in Table I. The net-A configuration has the same configuration as the Karpathy ConvNetJS CIFAR10 baseline [23] but with extended max-min pooling module instead of a standard max pooling method. The CNN classifier configuration designed with a maximum of 15 layers. We carefully computed the number of parameters of each configuration to express the correlation between the number of parameters with the accuracy of the classifier.

C. Overfitting

Overfitting is the main problem for modern CNN architecture. There are some methods that designed to reduce the effect of overfitting including dropout [28], batch

normalization [29], and heavy data augmentation process. At this stage, we use two versions of batch normalization configuration, BN and BNv2. In the BN version, we applied batch normalization only after main convolutional layers (the batch normalization does not put after the norm layer). In the BNv2 version, we applied batch normalization on all convolutional layers including the norm layer. For more further experiments, we also use heavy data augmentation process but the experiments are discussed in a separate section.

IV. Initial Experiments

In this section, we describe the results of initial experiments using six different network configuration as shown in Table I on CIFAR dataset [30]. All of the experiments were done using Caffe deep learning framework [31] and use the same parameters for fair comparison. We only use a validation set for evaluation and the winner is chosen for further experiments based on the accuracy on validation set.

A. CIFAR Dataset

CIFAR dataset [30] is a subset of 80 million tiny images introduced by Torralba et al. [32]. CIFAR dataset divided into two different data, CIFAR10 and CIFAR100. CIFAR10 consists of 10 different image classes with a total of 60,000 images while CIFAR100 consists of 100 different image classes with the same number of images. The dataset is one of the main datasets that mainly used for evaluating a new proposed classifier. Around 50,0000 images in the dataset are used for training or validation purposed and the rest of the images (10,000 images) are used for testing purpose. All labels in the dataset are simple objects, like ship, airplane, cat, etc.; and each image only contains one object with 32×32 image resolution.

B. Experiments Setup

Data Augmentation. In the training process, we follow a simple data augmentation process described in [33] by adding 4 pixels of zero padding on each side of the training dataset. The original resolution of training dataset is 32×32 pixel and will be increased to 36×36 after data augmentation process. In the training process, a random crop of 32×32 resolution and random horizontal mirror are used for on-fly data augmentation process. We subtract the training data with a mean value of 128 to create center zero mean data. We take 5,000 images (from 50,000 images) of the CIFAR10/CIFAR100 training data for validation purpose.

Training Process. To perform the training process, we initialize the learning rate at 0.01 and decrease to 0.001 at 200,000 iterations and stops at 600,000 iterations. The momentum of 0.9 and weight decay of 0.0005 is used for the training process. All experiments were performed with same training parameters using Adam stochastic gradient descent algorithm [34]. We use very high maximum iterations to compare the performance of batch normalization method with the classifier that not use batch normalization method. In the training process with CIFAR100 dataset, we decrease the learning rate again to 0,0001 at 400,000 iterations because the number of output class is higher than CIFAR10.

TABLE II: The results of the experiments on CIFAR10 dataset (lower is better).

Classifier	#Params	Validation (Error rate)		
		w/o BN	BN	BNv2
Baseline [23]	0.023M	19.65 %	-	-
net-A	0.025M	17.06 %	17.38 %	17.24 %
net-B	0.050M	15.88 %	15.72 %	15.82 %
net-C	0.120M	12.84 %	12.36 %	12.30 %
net-D	0.175M	13.00 %	11.98 %	11.78 %
net-E	0.466M	11.08 %	10.80 %	11.10 %
net-F	0.687M	10.52 %	11.18 %	10.08 %

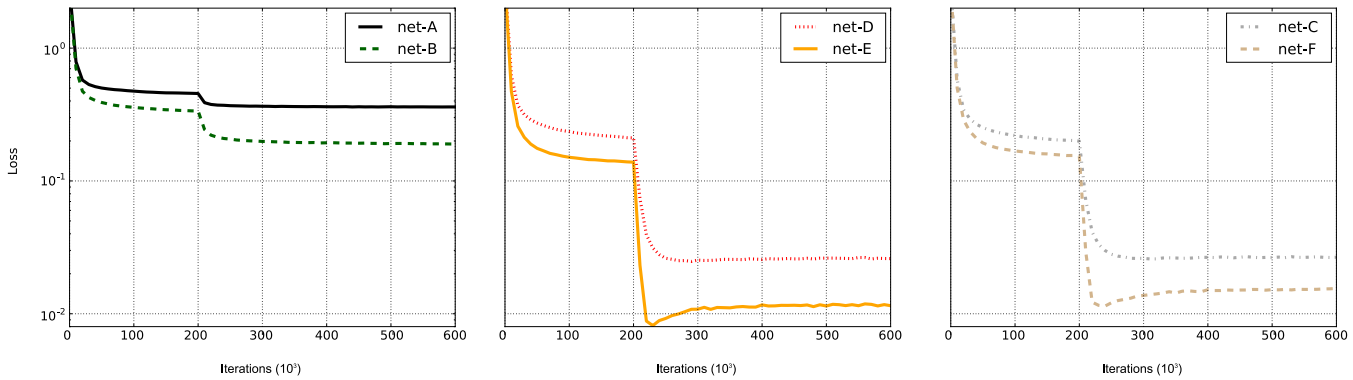
Testing Process. To perform the testing process, we only use one crop of full 32×32 resolution. We also subtract the testing data with a mean value of 128 to create the same zero mean distribution as training data. As mentioned before, we only tested the 5,000 validation images taken from the training data and choose the best network configuration for further experiments.

C. CIFAR10

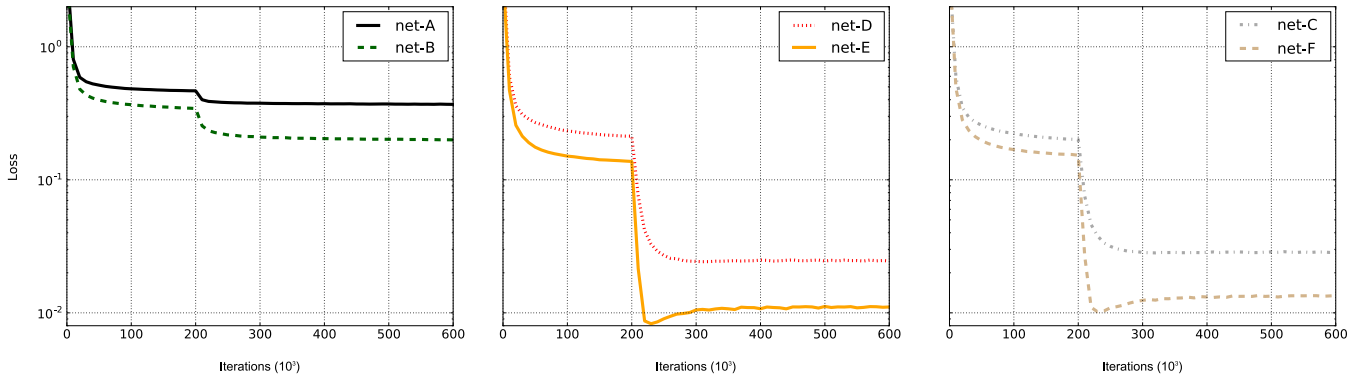
The summary of the experiments on CIFAR10 dataset can be viewed in Table II. As shown in Table II, the best performance is achieved by the classifier with net-F configuration with accuracy around 90%. The batch normalization method seems not significantly improve the performance of the classifier but, when the training loss is plotted, the classifier with batch normalization method has lower loss value compared with classifier without batch normalization method. The plot between training loss and iterations in the training process is shown in Figure 3. As shown in Figure 3, the classifier without batch normalization suffers from overshot loss value which causes by un-normalize convolution output. At this stage, we proved that the classifier without batch normalization can have similar performance as the classifier with batch normalization method but with more iterations in the training process. Overfitting problem also occurs in some classifier configuration. We detect the overfitting problem by comparing the loss value for training data and validation data. Even with batch normalization method, we found that the loss value for training data is very low (less than 0.01) but the loss value for validation data is still high, around 0.4. The problem is discussed and solved in further experiments by adding heavy data augmentation process and apply a dropout regularization method.

D. CIFAR100

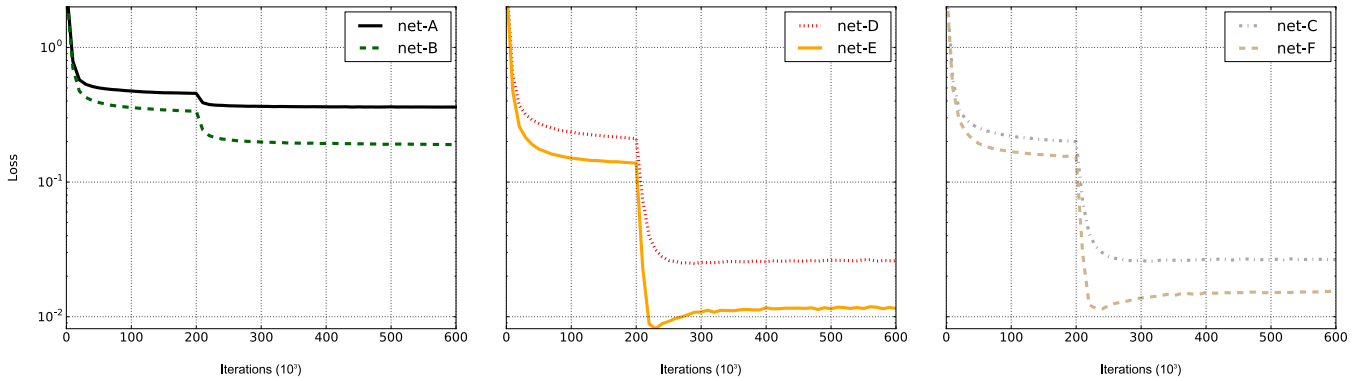
Table III shows the summary of our experiments using CIFAR100 dataset. The baseline [23] produces a quite low accuracy compared with our approaches. For example, the difference between our net-A approach that has a quite similar configuration with the baseline classifier is more than 5%. Unlike the CIFAR10 experiment, the CIFAR100 experiment shows that the best accuracy is achieved by net-E configuration instead of net-F. The plot between loss and iterations in the training process can be viewed in Figure 4. As shown in Figure 4, the training loss of net-F is actually



(a)



(b)



(c)

Fig. 3: The graphic between training loss and iterations on CIFAR10 dataset in the training process, (a) without BN method (b) with BN method (c) with BNv2 method.

lower than net-E but the validation shows that net-E produces more accuracy than net-F. The same phenomena also appear between net-C and net-D configuration. From the machine learning perspective, net-E and net-F suffer for overfitting indicated by very low training loss (around 10^{-2}) with high validation loss (around 2.0). The overfitting problem occurs more strongly in CIFAR100 experiments compared with the CIFAR10 experiments. Even with batch normalization method, the network still suffers from strong overfitting problem. As mentioned in the CIFAR10 experiments, we will try to solve the problem by utilizing heavy data augmentation and dropout in the separate section.

TABLE III: The results of the experiments on CIFAR100 dataset (lower is better).

Classifier	**#Params**	**Validation (Error Rate)**		
		w/o BN	**BN**	**BNv2**
Baseline [23]	0.051M	53.22 %	-	-
net-A	0.054M	47.02 %	48.16 %	47.53 %
net-B	0.096M	45.96 %	45.60 %	46.40 %
net-C	0.167M	45.42 %	44.46 %	43.74 %
net-D	0.222M	47.64 %	46.48 %	47.66 %
net-E	0.558M	42.16 %	37.60 %	37.32 %
net-F	0.779M	42.72 %	41.22 %	41.54 %

V. Deeper Experiments

As described in the previous section, all of the models was suffering from overfitting problem. In this section, we try to reduce the effect of overfitting problem by applying heavy data augmentation method and dropout regularization to the network. We choose net-E network configuration with

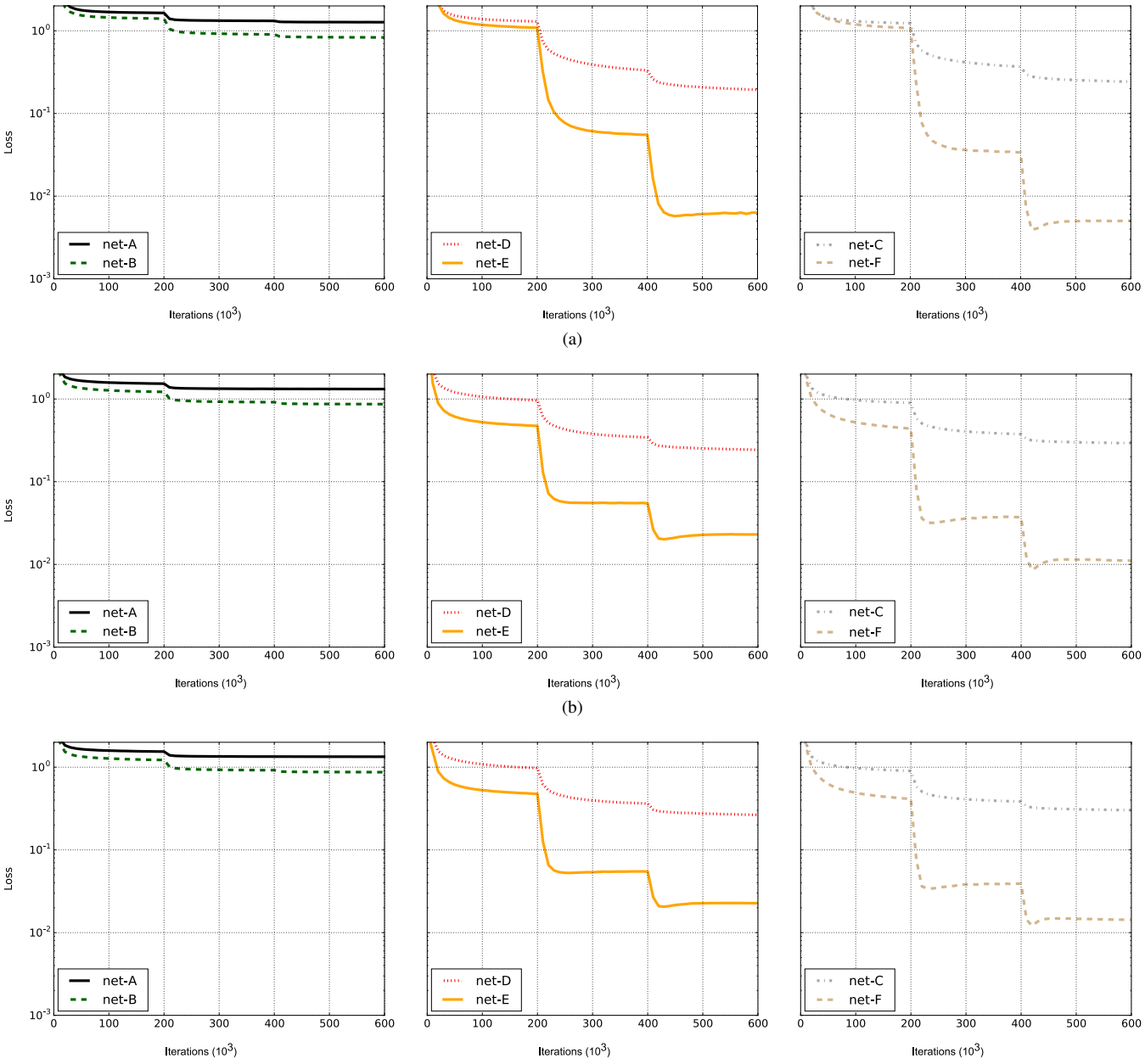


Fig. 4: The graphic between training loss and iterations on CIFAR100 dataset in the training process, (a) without BN method (b) with BN method (c) with BNv2 method.

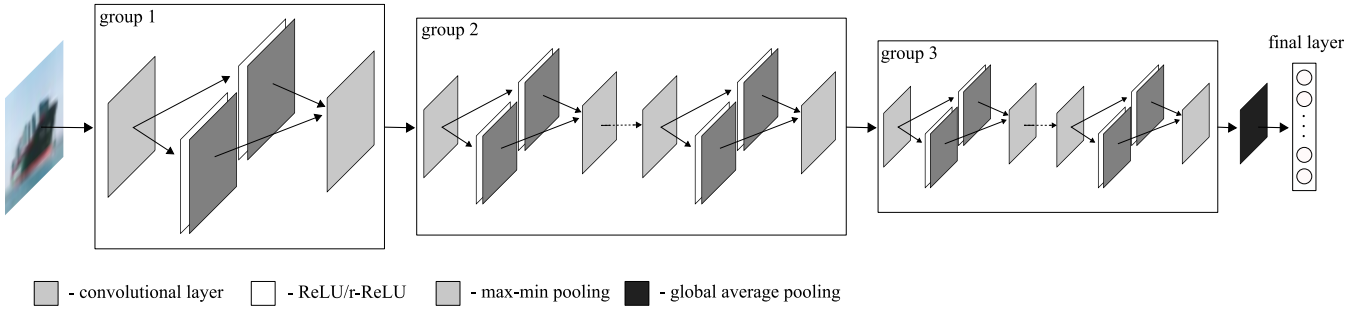


Fig. 5: The CNN structure of the final proposed classifier, which is a modification of net-E network configuration (please see Table I). The detail configuration of each layer can be viewed in Table IV.

TABLE IV: Final Configuration of Proposed Classifier with net-E as the base of the configuration.

Group	Layers	Output	#Params
input	RGB image (32x32@3)	32x32@3	-
group1	conv1 (7x7@64)	32x32@64	9,408
	mm-pool1 (2x2s2)	16x16@128	-
	conv1-n (1x1@64)	16x16@64	8,192
group2	conv2-1 (3x3@128)	16x16@128	73,728
	mm-pool2-1 (2x2s1)	15x15@256	-
	conv2-1n (1x1@128)	15x15@128	32,768
	conv2-2 (3x3@128)	15x15@128	147,456
	mm-pool2-1 (2x2s2)	8x8@256	-
	conv2-1n (1x1@128)	8x8@128	32,768
group3	conv3-1 (3x3@256)	8x8@256	294,912
	mm-pool3-1 (2x2s1)	7x7@512	-
	conv3-1n (1x1@256)	7x7@256	131,072
	conv3-2 (3x3@256)	7x7@256	589,824
	mm-pool3-1 (2x2s2)	4x4@512	-
	conv3-1n (1x1@256)	4x4@256	131,072
final	gobal-pooling	1x1@256	-
	fc-10 / fc-100	10 / 100	2,560/25,600
		Total #Params	1.45M/1.46M

BNv2 batch normalization configuration because the model performs very good validation accuracy on both CIFAR10 and CIFAR100 dataset. Although net-E is not the best model for CIFAR10 experiments, the accuracy gap between net-E with the best model is very narrow while on CIFAR100 experiments, net-E outperforms all other network configurations.

Figure 5 shows the structure of the final CNN classifier used in the experiments. To produce higher accuracy with comparable network parameters, we change the number of filters used in each group with increasing trends and a maximum of 256 filters. We follow the ResNet design strategy [6] by increasing the filter size in the first convolutional layer to 7×7 and reduce the filter size in other convolutional layers to 3×3. We also use average global pooling operation before the final layer, same as ResNet design strategy. Table IV shows the final configuration of the classifier with a total of 1.45 million parameters for CIFAR10 experiments and 1.46 million parameters on CIFAR100 experiments.

A. Choosing Training Algorithm

To analyze the effect of the training algorithm used in the training process, we conducted experiments using several different training algorithms, including SGD (Stochastic Gradient Descent), NAG (Nesterov Accelerated Gradient) [35], RMSProp, AdaGrad [36], and Adam [34] solver. The training process performs using minibatch of 256 examples for 64,000 iterations (or around 350 epochs) with a simple data augmentation method. All weights are initialized using the method described by He et al. [26]. The learning policy used in the training process for each training algorithm described as follows

- **SGD and NAG**. The training process runs for $64{,}000$ iterations with learning rate initialized at 0.1 and decreased to 0.01 and 0.001 at 32,000 and 48,000 iterations respectively.
- **RMSProp, AdaGrad, and Adam**. The training process was run for $64{,}000$ iterations with learning rate initialized at 0.001 and decreased to 10^{-4} and 10^{-5} at 32,000 and 48,000 iterations respectively.

Table V shows the summary of experiments using the proposed classifier trained with several different training algorithms. As shown in Table V, the performance of the proposed classifier is similar when trained using NAG or SGD training algorithm. The final training loss of the proposed classifier for all training algorithm is very low except for AdaGrad algorithm which has bigger training loss value. That phenomena show that Adam and RMSProp algorithm suffers from overfitting problem.

B. Weights Initialization

To provide more detailed analysis, we conducted experiments to analyze the effect of the weights initialization method used in the training process. In this experiment, we use the NAG training algorithm with the same learning rate policy as the one used in the previous experiments and initialize the weights using three different methods, He et al. [26] (MSRA), Xavier & Bengio [37] (Xavier), and Mishkin & Matas [38] (LSUV). Table VI shows the summary of experiments using the proposed classifier with weights initializes using three different weights initialization method. As shown in Table VI, the weights initialization method is not too much effect on the performance of the proposed classifier. We continue our investigation using MSRA weights initialization method with additional dropout and heavy data augmentation.

C. Minibatch Size

One of the parameters that affecting the performance of the classifier is the number of minibatch examples used in the training process. For further analysis, we perform several experiments using different minibatch configuration. To provide a fair comparison, we used epochs to determine the maximum number of iterations in the training process for each minibatch configuration. The learning rate in the training process is initialized at 0.1 and decreased to 0.01 and 0.001 at 50% and 75% of maximum iterations.

Table VII shows the results of experiments using several different minibatch configurations. As shown in Table VII, the best accuracy is achieved using a minibatch size of 512 for CIFAR10 dataset and minibatch size of 256 for CIFAR100 dataset. The minibatch configuration with more than 512 and less than 64 examples per iteration produces lower accuracy and tend to lead the classifier to overfitting condition. For the next experiments, we only use minibatch configuration with 256 and 512 number of examples per iteration.

D. Batch Normalization Configuration

To perform a deeper analysis, we conducted experiments using three different batch normalization (BN) configuration. The first configuration is placing the BN layer before the

TABLE V: Summary of the experiments using the proposed classifier trained using several different training algorithms on CIFAR dataset (reported using error rate, lower is better).

Training Algorithm	CIFAR10			CIFAR100		
	Validation	Testing		Validation	Testing	
		Center Crop	Multi Crop		Center Crop	Multi Crop
SGD	7.70%	8.68%	**7.72%**	32.18%	31.25%	29.65%
NAG	**7.54%**	**8.33%**	7.75%	**31.72%**	**31.01%**	**29.41%**
RMSProp	9.20%	9.57%	8.54%	34.96%	34.70%	32.74%
AdaGrad	19.60%	20.18%	17.85%	49.86%	48.67%	46.72%
Adam	8.82%	9.17%	8.28%	33.84%	33.65%	31.52%

TABLE VI: Summary of the experiments using the proposed classifier trained using several different weights initialization method on CIFAR dataset (reported using error rate, lower is better).

Training Algorithm	CIFAR10			CIFAR100		
	Validation	Testing		Validation	Testing	
		Center Crop	Multi Crop		Center Crop	Multi Crop
MSRA [26]	**7.54%**	8.33%	7.75%	31.72%	**31.01%**	**29.41%**
Xavier [37]	7.94%	**8.29%**	**7.66%**	**30.34%**	31.29%	30.11%
LSUV [38]	7.72%	8.38%	7.85%	32.08%	31.78%	29.92%

TABLE VII: Summary of the experiments using the proposed classifier trained using several different minibatch configurations on CIFAR dataset (reported using error rate, lower is better).

No.	Batch Size	#Epochs	#Iter	CIFAR10			CIFAR100		
				Validation	Testing		Validation	Testing	
					Center Crop	Multi Crop		Center Crop	Multi Crop
1.	16	364	1.024.000	12.16%	12.65%	11.07%	37.84%	38.94%	36.77%
2.	32	364	512.000	8.56%	10.03%	8.75%	35.32%	34.15%	31.83%
3.	64	364	256.000	8.66%	8.84%	7.73%	32.72%	32.29%	30.27%
4.	128	364	128.000	7.56%	8.51%	7.71%	32.42%	31.78%	29.82%
5.	256	364	64.000	**7.54%**	8.33%	7.75%	**31.72%**	**31.01%**	**29.41%**
6.	512	364	32.000	7.80%	**8.15%**	**7.39%**	32.52%	32.02%	30.02%
7.	1024	364	16.000	8.08%	8.55%	7.66%	34.08%	33.43%	31.86%

TABLE VIII: The effect of placing the Batch Normalization layer (reported using error rate, lower is better).

BN-Place	Batch Size	CIFAR10			CIFAR100		
		Validation	Center Crop	Multi Crop	Validation	Center Crop	Multi Crop
Before Activation Layer	256	7.54%	8.33%	7.75%	31.72%	**31.01%**	**29.41%**
	512	7.80%	8.15%	7.39%	32.52%	32.02%	30.02%
After Activation Layer	256	7.90%	**8.13%**	**7.12%**	32.12%	31.43%	29.85%
	512	**7.16%**	8.27%	7.16%	32.14%	31.65%	30.25%
Both	256	7.64%	8.59%	7.69%	**31.16%**	31.34%	29.57%
	512	7.46%	8.34%	7.52%	31.96%	31.75%	30.18%

activation layer which is the default configuration as we used in the previous experiments. The second configuration is placing the BN layer after the activation layer. The third configuration is placing the BN layer in both before and after the activation layer. We use the same training parameters as used in the previous experiments with a minibatch configuration of 256 and 512 number of examples per iteration and MSRA weight initialization method.

Table VIII shows the summary of the experiments using three different BN layer configurations. As shown in Table VIII, placing the BN after the activation layer produces higher accuracy on CIFAR10 dataset comparing with placing the BN before the activation layer. For CIFAR100 dataset, the accuracy for all BN configurations produces similar accuracy which is very different from the results using CIFAR10 dataset.

E. The Effect of Bias Parameter

We conducted further experiments by removing the bias parameters of the classifier except for the last fully-connected layer. The training process is done using the same training parameters as in the previous experiment. We also vary the classifier configuration, including the BN placing variation and two different minibatch configurations.

Table IX shows the results of the effect of bias parameters experiments. As shown in Table IX, the classifier performance is not too affected by the bias parameter. The difference between classifier with bias parameters and without bias parameters is under 1% on CIFAR10 and CIFAR100 dataset. The best performance achieves by the classifier with BN placed after activation layer and with 256 minibatch examples per iterations.

F. Dropout and Heavy Data Augmentation

In this last experiments, we try to reduce the overfitting problem by adding dropout regularizatio and hevay data augmentation method. We vary the classifier configuration using several different dropout probability rate, heavy data augmentation, two BN configuration, and two different minibatch configuration. For heavy data augmentation method, we use five data augmentation process; random image contrast correction, blurring operation, random color shifting operation, random translation operation, and random mirroring operation. The training process is perform using NAG training algorithm with learning rate intialized at 0.1 and decrease to 0.01 and 0.001 at 50% and 75% from maximum iterations respectively. The dropout regularization is attached in the last convolutional layers, just before the global pooling operation (please see Table IV). In total, we conducted experiments using 28 different classifier configuration.

Table X shows the summary of experiments using additional dropout and heavy data augmentation process. As shown in Table X, placing the BN layer after activation layer produces more stabil performance comparing with the classifier that placing the BN layer before the activation layer. The number of examples per iteration or minibatch size is not effected the performance of the classifier and produces similar performance, either on CIFAR10 dataset or CIFAR100 dataset. The best accuracy on CIFAR10 dataset achieves by classifier with additional dropout probability of 25% and no heavy data augmentation method. While the best accuracy on CIFAR100 dataset achieves by classifier with additional dropout probability of 6.25% and no heavy data augmentation method. The experiments conclude that the heavy data augmentation may not too impacted the overall performance of the proposed classifier.

VI. Comparison

For fairness comparison, we only compare the results of the proposed classifier with other classifiers that have shallow CNN architecture, including NiN CNN architecture [39]–[41], All-CNN [42], DSN (Deep Supervised Network) [33], FitNet [43], Highway Network [44], BinaryConnect [45], Gated Pooling CNN [46], and Maxout Network [47]. Table XI shows the error rate comparison of several state-of-the-art methods that have shallow architecture on CIFAR10 and dataset. As shown in Table XI, the proposed classifier outperforms other approaches that have shallow architecture. The detail number of parmeters in the classifier is included except for BinaryConnect and Maxout classifier. In term of the number of parameters, our proposed classifier has around 1.4 million parameters which are similar to several other classifiers, such as All-CNN [42] and Gated Pooling [46]. Unfortunately, several classifiers do not include the total number of parameters in their publication.

VII. Conclusion

We have presented a systematic evaluation of shallow CNN classifier designed using max-min pooling on CIFAR dataset. The max-min pooling method used to either exploring the positive and negative output of the convolution process. There are six different CNN architecture that we evaluated using CIFAR dataset. With the same network configuration, our classifier outperforms the original Karpathy ConvJS baseline by 2% in CIFAR10 experiments and 6% in CIFAR100 experiments. For deeper analysis, we conducted several additional experiments, including the effect of the training algorithm used in the training process, weights initialization, minibatch configuration, BN layer configuration, the effect of bias parameters, dropout regularization, and heavy data augmentation. As shown in the experiments, the classifier designed with max-min pooling method outperforms several other classifiers that have the same number of layers.

This paper only discusses the shallow version of the classifier with max-min pooling method. Additional experiments to analyze the deeper version of the classifier is required to improve the performance of the classifier and comparable with very deep CNN architecture, such as ResNet or DenseNet. Benchmarking the proposed classifier on other datasets, such as SVHN (Street View House Numbers) or STL-10, is also our concern for future work. Furthermore, a combination of the proposed classifier with the highway network or residual concept can be used to optimize the flows of the gradient in the training process and may produces a better accuracy.

References

[1] Q. Zheng, M. Yang, Q. Zhang, and J. Yang, "A bilinear multi-scale convolutional neural network for fine-grained object classification." *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 340–352, 2018.

TABLE IX: The effect of bias parameters (reported using error rate, lower is better).

Bias Param	BN Place	Batch Size	CIFAR10			CIFAR100		
			Validation	Center Crop	Multi Crop	Validation	Center Crop	Multi Crop
With Bias	Before Act. Layer	256	7.54%	8.33%	7.75%	31.72%	31.01%	29.41%
		512	7.80%	8.15%	7.39%	32.52%	32.02%	30.02%
	After Act. Layer	256	7.90%	8.13%	**7.12%**	32.12%	31.43%	29.85%
		512	**7.16%**	8.27%	7.16%	32.14%	31.65%	30.25%
Without	Before Act. Layer	256	7.58%	8.42%	7.49%	32.80%	31.99%	30.37%
		512	7.40%	8.27%	7.30%	32.92%	31.67%	30.95%
	After Act. Layer	256	7.98%	**8.12%**	7.35%	**31.60%**	31.21%	29.80%
		512	7.44%	8.36%	7.23%	31.66%	**30.97%**	**29.39%**

TABLE X: Summary of experiments using the proposed classifier with additional method (dropout regularization and/or data augmentation) on CIFAR dataset (reported using error rate, lower is better).

Additional Method	BN Place	Batch Size	CIFAR10			CIFAR100		
			Validation	Testing		Validation	Testing	
				Center Crop	Multi Crop		Center Crop	Multi Crop
None	Before Act. Layer	256	7.54%	8.33%	7.75%	31.72%	31.01%	29.41%
		512	7.80%	8.15%	7.39%	32.52%	32.02%	30.02%
	After Act. Layer	256	7.90%	8.13%	7.12%	32.12%	31.43%	29.85%
		512	7.16%	8.27%	7.16%	32.14%	31.65%	30.25%
Dropout($p = 6.25\%$)	Before Act. Layer	256	7.58%	8.24%	7.26%	31.78%	30.40%	29.56%
		512	7.36%	7.94%	7.24%	32.16%	31.92%	30.22%
	After Act. Layer	256	7.64%	7.78%	7.09%	31.34%	29.97%	**28.33%**
		512	7.76%	8.21%	7.32%	32.10%	31.24%	29.68%
Dropout($p = 12.5\%$)	Before Act. Layer	256	7.20%	7.52%	6.99%	31.86%	31.73%	30.08%
		512	**7.04%**	7.95%	7.50%	31.90%	31.38%	30.38%
	After Act. Layer	256	7.48%	7.72%	7.08%	30.56%	29.98%	29.05%
		512	7.40%	7.91%	7.06%	31.70%	30.97%	29.85%
Dropout($p = 25\%$)	Before Act. Layer	256	7.58%	7.97%	7.68%	31.06%	30.83%	29.65%
		512	7.38%	7.74%	7.32%	31.64%	31.04%	29.80%
	After Act. Layer	256	7.26%	7.67%	**6.81%**	30.36%	30.48%	29.80%
		512	7.24%	7.38%	7.05%	**30.30%**	31.18%	29.96%
Heavy Data Aug.	Before Act. Layer	256	7.58%	8.65%	7.93%	36.14%	34.83%	32.36%
		512	7.52%	8.12%	7.54%	31.88%	31.25%	29.97%
	After Act. Layer	256	7.46%	8.20%	7.36%	31.42%	30.62%	28.74%
		512	7.44%	7.92%	7.03%	33.22%	31.63%	29.76%
Dropout($p = 6.25\%$) + Heavy Data Aug.	Before Act. Layer	256	7.44%	8.06%	7.43%	31.80%	31.39%	29.51%
		512	7.50%	8.43%	7.50%	31.44%	30.66%	29.37%
	After Act. Layer	256	7.16%	7.97%	7.10%	30.90%	30.68%	28.62%
		512	7.16%	8.17%	7.20%	32.02%	30.58%	29.39%
Dropout($p = 12.5\%$) + Heavy Data Aug.	Before Act. Layer	256	8.22%	8.42%	7.68%	32.36%	31.16%	29.57%
		512	7.36%	8.11%	7.20%	31.94%	30.79%	29.70%
	After Act. Layer	256	7.20%	7.77%	7.25%	30.66%	30.54%	29.04%
		512	7.14%	7.61%	7.02%	32.02%	30.71%	29.30%
Dropout($p = 25\%$) + Heavy Data Aug.	Before Act. Layer	256	7.82%	8.65%	7.60%	31.62%	30.59%	29.96%
		512	7.14%	7.90%	7.21%	30.46%	30.91%	29.83%
	After Act. Layer	256	7.38%	7.68%	7.28%	30.80%	**29.59%**	28.46%
		512	7.44%	**7.35%**	6.95%	30.75%	30.66%	29.90%

TABLE XI: Comparison of error rate (in %) with several state-of-the-art methods on CIFAR10 and CIFAR100 dataset (lower is better).

No	Method	Depth	#Params	CIFAR10	CIFAR100
1	Network in Network (NiN) [39]	11	∼0.9M	8.81	35.68
2	All-CNN [42]	9	∼1.3M	7.25	33.71
3	DSN [33]	10	∼0.9M	7.97	34.57
4	FitNet [43]	19	∼2.5M	8.39	35.04
5	Highway Network [44]	19	∼2.3M	7.60	32.24
6	BinaryConnect [45]	9	-	8.27	-
7	Gated Pooling [46]	9	∼1.8M	7.62	34.21
8	Maxout [47]	5	-	9.38	38.57
9	NiN + APL units [40]	11	∼0.9M	7.51	30.83
10	NiN with RReLU [41]	11	∼0.9M	11.19	40.25
11	**Proposed Classifier (Our Best Approach)**	11	∼1.45M	**6.81**	**28.33**

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, pp. 2261–2269.

[8] R. F. Rachmadi, K. Uchimura, and G. Koutaki, "Video classification using compacted dataset based on selected keyframe," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2017, pp. 873–878.

[9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, vol. 1, 2014, pp. 568–576.

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[11] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[12] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[13] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue, "Evaluating two-stream cnn for video classification," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, 2015, pp. 435–442.

[14] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[15] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[16] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1681–1691.

[17] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[18] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 559–563.

[19] D. Palaz, R. Collobert *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," in *Proceedings of INTERSPEECH*, no. EPFL-CONF-210029, 2015.

[20] Z. Huang, M. Dong, Q. Mao, and Y. Zhan, "Speech emotion recognition using cnn," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 801–804.

[21] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference (BMVC)*, 2014, pp. 1–12.

[22] M. Blot, M. Cord, and N. Thome, "Max-min convolutional neural networks for image classification," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3678–3682.

[23] A. Karpathy, "Convnetjs: deep learning in your browser," http://cs.stanford.edu/people/karpathy/convnetjs/, 2017 (accessed December 1, 2017).

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, no. 1, 2013.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[27] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[28] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14, 2014, pp. 675–678.

[32] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[33] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-

supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[35] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.
[36] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[38] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.
[39] M. Lin, Q. Chen, and S. Yan, "Network in network," *International Conference on Learning Representations*, 2014.
[40] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.
[41] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
[42] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.
[43] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.
[44] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.
[45] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.
[46] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Artificial Intelligence and Statistics*, 2016, pp. 464–472.
[47] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1319–1327.

Reza Fuad Rachmadi received the B.Eng and M.Eng degree from Electrical Engineering Department of Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia in 2008 and 2010 respectively. He received Ph.D degree from Kumamoto University, Japan in 2018. His research interests include computer vision, image understanding, deep convolutional neural network, and image-based intelligent transportation system. He is members of IAENG.

I Ketut Eddy Purnama received the bachelor degree in Electrical Engineering from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia in 1994. He received his Master of Technology from Institut Teknologi Bandung, Bandung, Indonesia in 1999. He received Ph.D degree from the University of Groningen, the Netherlands in 2007. Currently, he is head of Computer Engineering Department of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. His research interest is in Data Mining, Medical Image Processing, and Intelligent System. He is a member of IAENG.

Mauridhi Hery Purnomo earned his bachelor degree from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 1985, then his M.Eng., and Ph.D. degrees from Osaka City University, Osaka, Japan in 1995 and 1997 respectively. He joined ITS in 1985 and has been a Professor since 2004. His current interests include intelligent system applications, electric power systems operation, control, and management. He is an IAENG and IEEE Member.

Mochamad Hariadi received the B.Eng. degree in Electrical Engineering Department of Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 1995. He received both M.Sc. and Ph. D. degrees in Graduate School of Information Science Tohoku University Japan, in 2003 and 2006 respectively. Currently, he is a staff of Computer Engineering Department of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He is the project leader in joint research with PREDICT JICA project Japan and WINDS project Japan. His research interest is in Video and Image Processing, Data Mining and Intelligent System. He is a member of IEEE, IEICE, and IAENG.