Probability Voting-based Ensemble of Convolutional Neural nets Classifiers for Image Classification

Sarwo, Yaya Heryadi, Widodo Budiharto, Edi Abdurachman

Abstract: ***This study explores an ensemble technique for building a composite of pre-trained VGG16, VGG19, and Resnet56 classifiers using probability voting-based technique. The resulted composite classifiers were tested to solve image classification problems using a subset of Cifar10 dataset. The classifier performance was measured using accuracy metric. Some experimentation results show that the ensemble methods of pre-trained VGG19-Resnet56 and VGG16-VGG19-Resnet models outperform the accuracy of its individual model and other composite models made of these three models.***

Keywords :Ensemble Classifiers, VGG16, VGG19, Resnet56, Probability Voting Technique, CIFAR-10.

I. INTRODUCTION

Image classification is a challenging computer vision problem with wide applications ranging from engineering, medical, and bioinformatics. CIFAR-10 dataset consists of 60,000 (32x32) colour images in 10 classes with 6,000 images per class. The dataset is available at www.cs.toronto.edu [1].

The rise of deep learning in the past ten years has attracted the attention of many researchers to solve image classification using various deep neural network models. Surprisingly, many successful models are rooted from convolutional neural networks (CNN). These results have established CNN as a robust class of models to address various pattern recognition problems such as detection and recognition of faces, objects and traffic signs, image segmentation, and image retrieval [1]–[5].

The successful implementation of CNN as a classifier has motivated further research to improve the accuracy of the model, which resulted in various CNN models. These CNN improvements can be categorized broadly into: (1) structure improvement of individual models; and (2) ensemble construction of existing models. The first improvement approach resulted in many models. AlexNet model, for example, was proposed in 2012 AlexNet model, for example, proposed in 2012 [4], ZFNET in 2013 [6], VGG in 2014 [7], and Resnet in 2015. Various techniques which have been reported to improve the model accuracy include: increasing the number of layers [8], increasing layer size [6],[9], introducing dropout layer[10], and combining discriminator and generator models [11].

The second approach for improving the classifier accuracy is adopting ensemble technique which combines various learning algorithms to improve the overall classification accuracy. The premise of this technique is that a set of classifiers give more accurate decisions than a single classifier. However, to the best of our knowledge, there is no guideline on which ensemble technique will give the best model accuracy.

This research aims to explore probability voting technique to build an ensemble technique from the latest best classifiers, namely: VGG16, VGG19, and Resnet56.

II. RELATED REVIEW

A. Image Classification

Image classification has been an important and interesting research problem in computer vision. Research on multi-layered architecture models to build a robust classifier for image classification resulted in various Convolutional Neural Network models such as Deep Residual Learning for Image Recognition [12], Very Deep Convolutional Networks for Large-Scale Image Recognition[7], Densely Connected Convolutional Networks [13] and other research such as [14]–[20].

B. Convolutional Neural Networks

Convolutional Neural Networks are a special type of multi-layer neural networks. In the past decade, various architectures of CNN have been proposed by many researchers to address pattern recognition problems, including classification and forecasting. Among CNN family models, LeNet model proposed by LeCun et al. (1998) perhaps is the first CNN model designed to solve classification problems [1].

The LeNet model is a multi-layer neural network trained with the back-propagation algorithm based on gradient-based learning technique to optimize the respective objective function.

Manuscript published on 30 September 2019

* Correspondence Author

Sarwo*, Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia. Email: sarwo.jowo@gmail.com

Yaya Heryadi, Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480. Email: yayaheryadi@binus.edu

Widodo Budiharto, Computer Science Study Program, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480. Email: wbudiharto@binus.edu

Edi Abdurachman, Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480. Email: ediA@binus.edu

A study by LeCun et al. (1998) further shows that gradient-based learning algorithms can solve optimization problems of complex decision surface. Hence, LeNet can be used to classify high-dimensional patterns e.g. handwritten characters without elaborated input data preprocessing [1].

The backpropagation algorithm to train CNN is an algorithm to approximate a global minimum of the set out objective function. During training process, the weights as CNN parameters were initialized with random values. During iteration steps, the weights were corrected using gradient of the objective function until they were converged or the mean square error between target and actual class was below the predetermined threshold value. The availability of more efficient and faster training algorithm to estimate weights of the CNN models makes it possible to use CNN to address some high-dimensional pattern recognition problems such as character recognition.

In general, architectures CNN consists of three layers. They are Convolutional layers, Pooling layer (the one usually used is Max Pooling), and fully connected layers. We can simplify architecture CNN as follows:

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M ->
[FC -> RELU]*K -> FC.
```

This * sign indicates repetition, which means that the layers convolutions can consists of many layers. Meanwhile, Relu or rectified linear unit (RELU) in general can be defined by the following function:

$$\mathcal{F}(x) = x^{+} = \max(0, x) \quad [1]$$

Where x is input neuron? Relu was firstly introduced by Hahnloser *et al.* in 2000 on a paper published in Nature which became popular in 2015 together with deep learning model. Pooling layer is a function for decreasing the feature and network complexity obtained from convolution process. The model pooling layers mostly used is max pooling. RELU subsists many; some of them are as follows:

1. Noisy ReLUs: Relu can be combined with Gaussian noise algorithm :

$f(x) = \max(0, x + Y)$, *with* $Y \sim N(0, \sigma(x))$ [2]

Noisy ReLUs have been successfully applied to restricted Boltzmann machines for computer vision tasks.

2. Leaky ReLUs

Leaky ReLUs was introduced by Andrew in 2014 (Andrew et al., 2014). Leaky ReLU allows small and non-zero gradient when the unit is not active. So, the equation is as follows:

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0.01x, & otherwise \end{cases} \quad [3]$$

According to He et al. (2015), Leaky ReLUs is Parametric ReLUs take the idea deeper with make leakage the coefficients be parameter learned at the same time as other nerve network parameter. It is shown by the following equation:

$$f(x) = \begin{cases} x, & if\ x > 0 \\ ax, & otherwise \end{cases} \quad [4]$$

Where $a < 1$ equal to function $f(x) = \max(x, ax)$ (He et al. ,2015)

3. ELUs

ELUs or Exponential linear unit is one of the variants of ReLUs where the main difference of ELUs is value ∂^{x-1}. ELUs make the activations become average and also accelerate the learning process, so that ELUs are more accurate than Noisy ReLUs and Leaky ReLUs (Clevert et al., 2015). The equation of ELUs is as follows:

$$f(x) = \begin{cases} x, & if\ x = 0 \\ \partial^{x-1}, & otherwise \end{cases} \quad [5]$$

Another research on RELU is the one conducted by Glorot et al. (2011). The research discussed the rectifying of non-linearities as alternative solutions for the hyperbolic tan x functions whereas tan x wen can be shown by the following function:

$$\tan x = \frac{\sin x}{\cos x} \quad [6]$$

or using the sigmoidal curve using deep artificial neural networks.

Where the ∂ is the hyper parameter and $\partial >= 0$ is a constraint, the last layer is Fully Connected where the Fully Connected is a liaison between the previous layer with the layer after that. Commonly, the Fully connected layers are located one level before classification or detection.

Latest research by Karpathy and Fei-fei (2016) reported the use of CNN-based model to address document recognition problems. The model consists of several main modules working in a pipeline namely: extraction, segmentation, recognition and language modeling. This study combined document recognition and language modeling.

C. VGG Model

VGG model proposed by [7] is a Convolutional Neural Network family model designed based on the study of Simonyan and Zisserman (2015) [7] The authors argue that the representation depth gives significant effects to the classification accuracy to achieve the state-of-the-art performance on the ImageNet challenge dataset.

A study by [3] showed that VGG models can achieve high performance to solve object detection problems. In VGG family models, VGG19 is an important image classifier for large image dataset as it achieved 5-top five image classifier in ILSVRC 2014 with 7.32 percent error.

In a VGG architecture, the input is RGB image with a resolution of 224×224 pixels. Before convolutions, input images were pre-processed to reduce the RGB pixel value. The input images were then convolution using 3×3 filters. The filter size is smaller than the filter used in GoogleNet, AlexNet, and LeNet models. The VGG model was further extended using different number of layers. Two well-known models and its respective number of layers are: VGG16 (16 layers) and VGG19 (19 layers).

In VGG Configuration, VGG16 model has two layers with 16 weight each. The structure of VGG16 is as follows: conv3-64 layers, conv3-64 layers, conv3-128 layers, conv3-128 layers, conv3-256 layers, conv3-256 layers, conv3-256 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers. The stack of layers is followed by 3 layers of fully connected which are: FC-4096, FC-4096 and FC-1000.

A VGG19 has two layers of 19 weight each. The structure of VGG19 is as follows: conv3-64 layers, conv3-64 layers, conv3-128 layers, conv3-128 layers, conv3-256 layers, conv3-256 layers, conv3-256 layers, conv3-256 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers, conv3-512 layers. The stack of layers is followed by 3 layers which are: FC-4096, FC-4096 and FC-1000. Compared to VGG16, VGG19 has two additional layers: conv3-256 layers and conv3-512 layers.

The detailed explanations of VGG are as follows. All of VGG models are convolutional layers that use 3x3 convolutions with stride 1 and padding 1, also pooling layer with 2x2size. The detailed description and calculation on VGG are as follows:

1. INPUT: [224x224x3]
VGG only accept inputs with a size of 224x224 using filter with size 3, so that from this input, the layer was obtained by the memory calculation of 150 Millions achieved from 224*224*3.
2. CONV3-64: [224 x 224 x 64]
On the second layers, these CONV3-64 use size 3 filter and the total output at 64 so that a total memory at 224 * 224 * 64 = 3.2 Millions was obtained and the weights : (3 * 3 * 3) * 64 = 1,728.
3. CONV3-64: [224 x 224 x 64].
The third layers have similarities with the second layer, which are the size of filter 3 and the output 64. The memory calculation obtained is as follows: 224*224*64=3.2 Millions whereas the weights calculation is : (3*3*64)*64 = 36,864.
4. POOL2: [112x112x64]
The fourth layers are pooling layers with size [112x112x64] and the obtained memory: 112*112*64=800 K.
5. CONV3-128: [112x112x128]
The fifth layers are convolutional layers with the size of third filters and the number of output is 126 so that they need a memory at 112*112*128=1.6 Millions and the weights: (3*3*64)*128 = 73,728.
6. CONV3-128: [112x112x128]
The sixth layers of VGG are convolutional layers with the filter size of 3 and the output of 123, so that the obtained memory calculation is : (3*3*128)*128 = 147,456.
7. POOL2: [56x56x128].
The seventh layers are Polling layers with size [56x56x128] so that the obtained memory: 56*56*128=400K.
8. CONV3-256: [56x56x256]
The eighth layers are convolutional layers with the filter size of 3 and the output of 256 so that they need a memory: 56*56*256=800K and the size of weights at (3*3*128)*256 = 294,912.
9. CONV3-256: [56x56x256]
The ninth layers are convolutional layers with the filter size of 3 and the output of 256 so that they need a memory: 56*56*256=800K and the size of weights at (3*3*256)*256 = 589,824.
10. CONV3-256: [56x56x256]
The tenth layers are convolutional layers with the filter size of 3 and the output of 256 so that they need a memory at 56*56*256=800K and the size of weights at (3*3*256)*256 = 589,824.
11. POOL2: [28x28x256]
The next layers are pooling layers. The size of the layers is [28x28x256] so that they need a memory: 28*28*256=200K.
12. CONV3-512: [28x28x512]
The next layers are convolutional layers with the filter size of 3 and the output of 512 so that they need a memory at 28*28*512=400K and the size of weights at (3*3*256)*512 = 1,179,648.
13. CONV3-512: [28x28x512]
The next layers are convolutional layers with the filter size of 3 and the output of 512 so that they need a memory at 28*28*512=400K and the size of weights at (3*3*2512)*512 = 2,359,296.
14. C
ONV3-512: [28x28x512]
The next layers are convolutional layers with the filter size of 3 and the output of 512 so that they need a memory at 28*28*512=400K and the size of weights at (3*3*2512)*512 = 2,359,296.
15. POOL2: [14x14x512]
The next layers are pooling layers with the size of [14x14x512] so that they need a memory: 14*14*512=100K
16. CONV3-512: [14x14x512]
The next layers are convolutional layers with the filter size of 3 and the output of 512 so that they need a memory at 14*14*512=100K and the size of weights at (3*3*512)*512 = 2,359,296.

17. CONV3-512: [14x14x512]
The next layers are convolutional layers with the filter size of 3 and the output of 512 so that they need a memory at 14*14*512=100K and the size of weights at (3*3*512)*512 = 2,359,296.
18. C
ONV3-512: [14x14x512]
The next layers are convolutional layers with the filter size of 4 and the output of 512 so that they need a memory at 14*14*512=100K and the size of weights at (3*3*512)*512 = 2,359,296.
19. FC: [1x1x4096]
The next layers are the fully connected with the size of [1x1x4096] so that they need a memory: 4096 and the weights: 7*7*512*4096 = 102,760,448
20. FC: [1x1x4096]
The next layers are the fully connected with the size of [1x1x4096] so that they need a memory: 4096 and the weights: 4096*4096 = 16,777,216.
21. FC: [1x1x1000]
The latest layers are the fully connected with the size of [1x1x1000] so that need a memory: 1000 and the weights: 4096*1000 = 4,096,000.

It can be summarized that the total layers need a total memory at 24M * 4 bytes ~= 93MB / image and total params: 138M parameters (Karpathy,2016).

D. Resnet Model

Resnet model was proposed by [12] as Convolutional Neural Networks family models with deep layers. In 2015, the model achieved its best performance by becoming one of the top five image classifiers in ILSVRC 2015 with 3.57 percent error.
Different from previous models which only rely on deep layers, the design of Resnet is based on a different perspective. According to [12], the principle underlying the Resnet model is as follows. Let H(x), where x denotes the inputs to the first of these layers, be an underlying mapping to be approximated by several stacked layers of the Resnet model.

It is hypothesized that if multiple nonlinear layers can asymptotically approximate the functions, then the multiple layers can asymptotically approximate the residual functions, i.e., H(x) – x. Consider stacked nonlinear layers fit another mapping of F(x) =H(x)-x. The original mapping is recast into H(x) = F(x)+x. We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping.

In the implementation of Resnet by [12], F(x)+x was implemented using feedforward neural networks with "shortcut connections", a term which refers to the connection that skips one or more layers. The shortcut connections simply perform identity mapping and their outputs are added to the outputs of the stacked layers. Finally, the Resnet can be trained end-to-end by Stochastic Gradient Descent (SGD) algorithm with backpropagation.

E. Ensemble Model

Ensemble models are composite models built from several models which are trained separately to learn patterns from input dataset. Ensemble learning is a term which refers to the learning process of machine learning models aimed at improving the prediction performance. It is achieved by combining predictions of several learning algorithms (Lofstrom, 2015). The main objective of developing an ensemble model is to reduce the risk of selecting a less-perform model by building an ensemble of several candidate models.

Over the past few years, ensemble model has gained research interest and has had significant impacts in various research fields. This new way of developing model has been proven to be very effective in addressing classification problems in many domains and real-life applications. The strength of the ensemble model is its ability to reduce yield variance and improve accuracy for addressing classification problems. A study by [21] shows that ensemble model method can solve several main challenges in classification, such as missing feature, error correction, confidence estimation, incremental learning and imbalanced data classification.

Some advantages identified when we used the ensemble model include: (1) better prediction; (2) more stable model because the ensemble model involves several models together. This is the main reason why the ensemble models are better than an individual model.

In general, techniques to build ensemble models can be categorized broadly into 3(three) categories:

1. Boosting: it is an ensemble technique aimed at creating a strong learning algorithm from a number of weak learning algorithms [27]. One prominent boosting algorithm is AdaBoost proposed by Freund & Schapire, 1997 designed for binary classification.

2. Bagging: it is an ensemble technique proposed by Breiman ,1994) for generating an aggregated classifier from multiple versions of a classifier. Bagging produces replicate training sets by sampling with replacement from the training instances to build some classifiers. The trained multiple classifiers are then combined by voting to form a composite classifier.

3. Stacking: it is an ensemble technique proposed by Ting and Witten (1999). This technique uses a high-level model to combine lower-level models to achieve a model with greater predictive accuracy.

The comparison between Bagging, Boosting and Stacking ensemble techniques is presented in the following table.

TABLE I. Comparison of Bagging, Boosting and Stacking [33]

No	Aspect	Bagging	Boosting	Stacking
1	Partition dataset	Random	Misclassificati on on pre-reference which is higher	Multi dataset
2	Last Results	Optimizing variance	Optimizing the last prediction	Optimizing variance and last prediction
3	Model used	Random subspace	Gradient Descent	Random combinatio n and gradient descent
4	Model used for combinat ion	Average	Weighted majority vote	Logistic Regression

As the table indicates, three common processes of ensemble techniques are as follows: (1) data sampling or selection, (2) training member classifiers, and (3) combining classifiers.

1. **Data sampling/selection**: This step aims to prepare input dataset for testing the models to be ensembled. There are several publicly available datasets for this purpose such as VOC, PASCAL, IMAGE.NET, CIFAR-10, and CIFAR-100.
2. **Training member:** The objective of this step is to train each model separately. These trained models are then combined into an ensemble model.
3. **Combining classifiers:** The last step in the ensemble technique is combining classification to form a composite model with better predictive accuracy than individual classifiers. The three different techniques to combine a composite model are: Voting-base, Bagging or Boosting.

F. Ensemble Vote Classifier

Ensemble Vote Classifier is a composite of trained classifiers in which final prediction is computed by combining majority or plurality voting. The examples of the applications of Ensemble Vote Classifier model are hard-voting and soft-voting which can be described as follows.

1. Hard-voting is a technique where the final prediction is any class which obtains the highest number of votes given by the models. In the process of majority voting or hard voting, there are three conditions as follows:

a) Anonymous voting is the condition where the votes of all classes are the same.

b) Number of classes generated at least 50 percent of all class candidates or the number of abstentions is less than 50 percent.

c) Probability class taken from this model is the highest.

Let N be the number of classifiers. Consider that each classifier vote for a class which it predicts; otherwise, the expert abstains from voting. Given new data as input. Let the total number of votes (K) equals N minus the number of abstentions. Consider votes are given equal weight. If any class C receives $C_v > \frac{K}{2}$ votes, then C is the predicted class; otherwise, the new data are rejected (Miller &Yan, 1999). Using this scheme, the Prediction class C using hard-voting from each classification model C_j given new data x or $C_j(x)$ is computed based on the equation below:

$$C = mode\{C_1(x), C_2(x), \dots, C_m(x)\} \quad [7]$$

Besides the calculated majority vote (hard voting) as described above, alternatively, hard-voting can also be implemented as weighted majority vote by associating weight W_j with classifier C_j.

$$C = argmax_i \sum_{j=1}^{N} w_j X_a\left(C_j(x) = i\right) \quad [8]$$

Where X_a is characteristic function $\left[C_j(x) = i \in A\right]$ and A is unique class label [32].

2. Soft-voting is a technique that predicts the final class of new data as any class which obtains the highest prediction probability given by each model (Cao et al., 2015). Consider $P\left(C_{h_j}|x\right)$ be probability of class C_{h_j} predicted by the j[th] classifier given x as new data. The predicted class is computed by the following equation:

$$C = \text{argmax}_i \sum_{j=1}^{N} W_j P\left(C_{h_j}|x\right) \quad [9]$$

Where W_j is weight of the in ensemble and P_{ij} is probability of the predicted class by the ensemble.

III. RESULT AND DISCUSSION

A. Cifar 10 Training

This experimental using Cifar10 dataset, in this all models will optimized with Stochastic Gradient Descent (SGD) with the following parameter values: learning rate = 0.01, decay = 0.01 / 70, Momentum = 0.9, Nesterov's Acceleration = True, for this training using epoch 70 times.

VGG16 produce train loss: 10%, accuracy train: 96%, vall loss: 98% and vall accuracy: 79%, as indicated by Table II. and Fig.1 below:

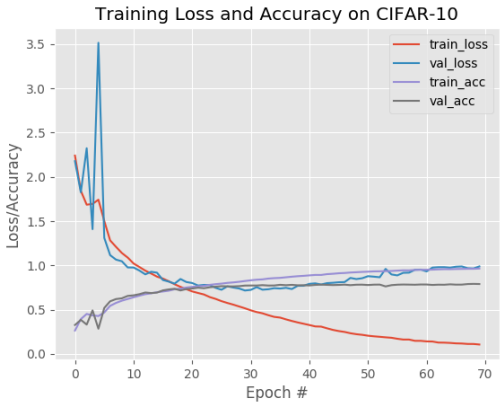

Fig. 1. Training CIFAR-10 using VGG16

TABLE II. TRAINING CIFAR-10 USING VGG16

Datasets	train. loss	train acc.	val-_loss	val_acc
Cifar10	0.1070	0.9650	0.988	0.7901

VGG19 model in this training resulted in loss: 97%, accuracy train: 65%, vall losss: 85% and vall accuracy: 69%, as shown in TABLE III. and FIG. 2 below:

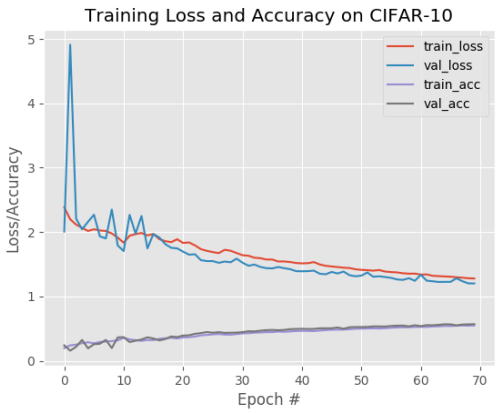

Fig. 2. Training CIFAR-10 using VGG19

Table III. Training CIFAR-10 using VGG19

Datasets	Train loss	Train accuracy	Val_loss	Val_accuray
Cifar10	0.9769	0.6591	0.8538	0.6978

Resnet generated train loss: 30%, accuracy train: 99%, vall losss: 110% and vall accuracy: 80%, as shown in Table IV. and Fig. 3. below:

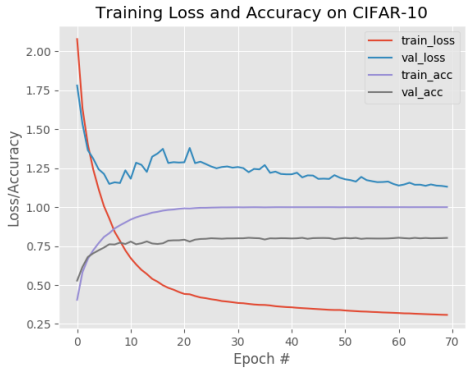

Fig. 3. Training CIFAR-10 on VGG19

Table IV. Training Cifar10 using VGG19

Datasets	Train loss	train accuray	val_loss	Val accuray
Cifar10	0.3082	0.9999	1.1312	0.8024

Testing CIFAR-10 using Individual Model This experiment tested the produced model during the training phase.

The section was divided into two parts namely Individual model and ensemble model. The individual models are shown in Table V to Table VII and Fig. 4 to Fig. 7.

Table V. VGG16 Testing

No	Class/label	Precision	Recall
1	airplane	0.82	0.81
2	automobile	0.91	0.89
3	bird	0.70	0.73
4	cat	0.64	0.56
5	deer	0.74	0.78
6	dog	0.66	0.69
7	Frog	0.83	0.84
8	horse	0.86	0.82
9	ship	0.88	0.90
10	truck	0.87	0.88
Average		**0.79**	**0.79**

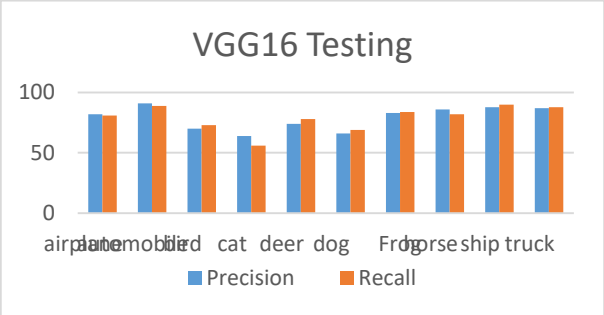

Fig. 4. VGG16 Evaluation

Table VI. VGG19 Testing

No	Class/label	Precision	Recall
1	airplane	0.67	0.69
2	automobile	0.77	0.76
3	bird	0.61	0.36
4	cat	0.42	0.43
5	deer	0.48	0.66
6	dog	0.64	0.38
7	frog	0.56	0.84
8	horse	0.73	0.67
9	ship	0.78	0.75
10	truck	0.72	0.72
Average		**0.64**	**0.63**

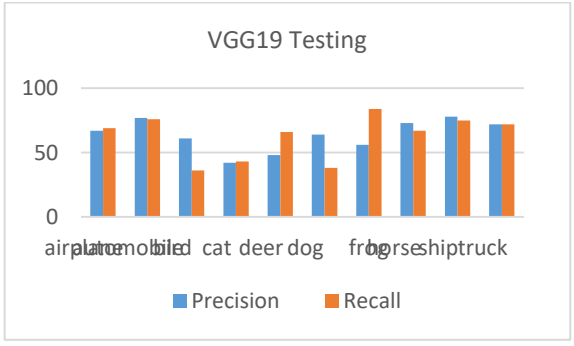

Fig. 5. VGG19 Evaluation

Table VII. Resnet testing

No	Class/label	Precision	Recall
1	airplane	0.82	0.84
2	automobile	0.90	0.90
3	bird	0.76	0.69
4	cat	0.60	0.69
5	deer	0.78	0.79
6	dog	0.74	0.67
7	frog	0.83	0.76
8	horse	0.87	0.81
9	ship	0.89	0.88
10	truck	0.89	0.89
Average		**0.80**	**0.80**

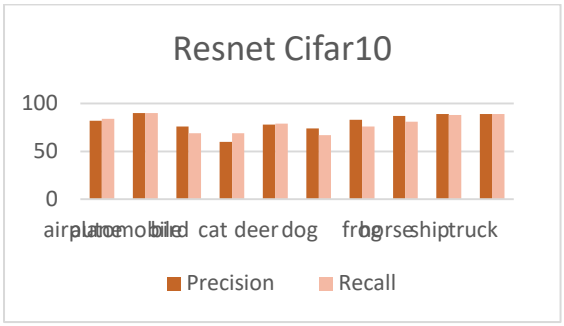

Fig. 6. Resnet Evaluation

G. Ensemble model testing

Hence, the ensemble models of interest are several composite models of three classifiers namely VGG16, VGG19, and Resnet56 Using CIFAR-10 Datasets. The aim of developing composite model is to build a model that achieves better accuracy from each individual classifier. Probability voting of the ensemble classifier of interest can be described as follows :

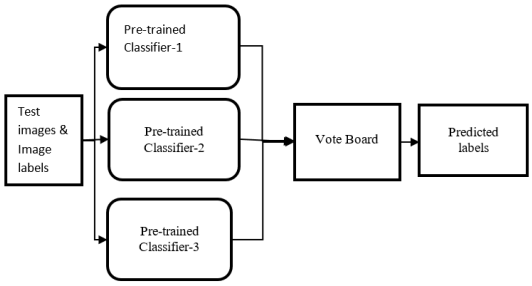

Fig. 7. Probability Voting Ensemble Scheme

1. Hard-voting

Let C be a class of image, x be a test image. The probability voting $P(C|x)$ of the ensemble classifiers aims to maximize the following probability.

$$C = argmax_i \sum_{1=1}^{3} w_j X_a(C_j(x) = i) \quad [4]$$

2. Soft-voting

Let C be a class of image, x be a test image. The probability voting $P(C|x)$ of the ensemble classifiers aims to maximize the following probability:

$$C = \text{argmax}_{\ i} \sum_{j=1}^{3} W_j \, P\left(C_{h_j}|x\right) \quad [5]$$

Where:

$$W_1 = W_2 = W_3 = \frac{1}{3}$$

$P(C_{h_1}|x)$ = Probability of image x has label C_{h_1} using model VGG16.
$P(C_{h_2}|x)$ = Probability of image x has label C_{h_2} using model VGG19.
$P(C_{h_3}|x)$ = Probability of image x has label C_{h_3} using model Resnet56.

Ensemble Testing Using Cifar10 Dataset

Here are the ensemble results from Soft-voting and Hard-voting measurements using Cifar10 datasets.

Table VIII. Precision using ensemble Soft-voting

No	Class	VGG16*VGG19	VGG16 * Resnet56	VGG19 * Resnet	VGG16* VGG19* Resnet56
1	airplane	82	82	82	82
2	automobile	91	76	91	91
3	bird	70	36	76	76
4	cat	64	43	64	64
5	deer	74	66	78	78
6	dog	66	38	74	74
7	frog	83	84	83	84
8	horse	86	67	87	87
9	ship	88	75	89	89
10	truck	87	72	89	89
Average		**79,1**	**79,8**	**81,3**	**81,4**

Table VIII. Presented in graphical form, according to Fig. 8.

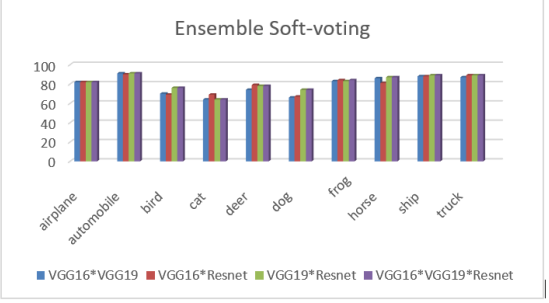

Fig. 8. Precision using Ensemble Soft-voting

The results of Recall measurement using soft-voting ensemble are presented in Table VIII where the highest results are automobile and ship classes with 90% values.

Table VIII. Recall testing using Ensemble Soft-voting

No	Class/label	VGG16* VGG19	VGG16 * Resnet56	VGG19 * Resnet56	VGG16* VGG19* Resnet56
1	airplane	81	84	84	84
2	automobile	89	90	90	90
3	bird	73	69	73	73
4	cat	56	69	69	69
5	deer	78	79	79	79
6	dog	69	67	69	69
7	frog	84	84	84	84
8	horse	82	81	82	82
9	ship	90	88	90	90
10	truck	88	89	89	89
Average		**79**	**80**	**80,9**	**80,9**

The results in TABLE VIII are presented in a graphical form in Fig. 9.

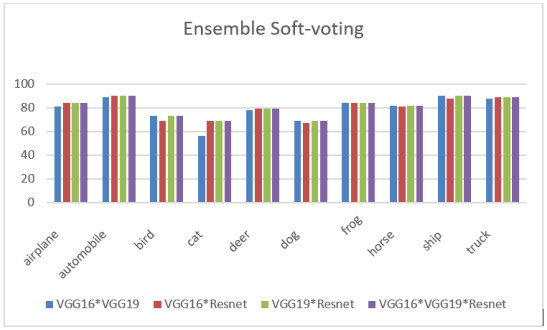

Fig. 9. Recall testing using Ensemble Soft-voting

Ensemble Hard Voting technique results in precision measurement such as TABLE VIII. The highest precision value is 90.5% which is the automobile class .

TABLE VIII. Precision testing using Ensemble Hard-voting

No	Class	VGG16 *VGG19	VGG16 * Resnet56	VGG19 * Resnet56	VGG16 * VGG19 * Resnet56
1	airplane	74.5	74.5	82	77
2	automobile	84.0	83.5	90.5	86.0
3	bird	65.5	68.5	73	69.0
4	cat	53.0	51.0	62	55.3
5	deer	61.0	63.0	76	66.7
6	dog	65.0	69.0	70	68.0
7	frog	69.5	69.5	83	74.0
8	horse	79.5	80.0	86.5	82.0
9	ship	83.0	83.5	88.5	85.0
10	truck	79.5	80.5	88	82.7
Average		**71.5**	**72.3**	**80.0**	**74.6**

The results in the table are represented by the following Fig.

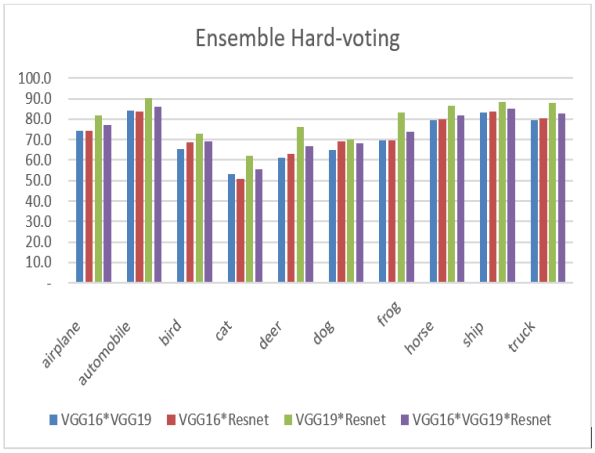

Fig. 10. Precision measurement using Ensemble Hard-voting

Ensemble Hard Voting technique results in Recall measurement such as TABLE IX. The highest precision value is 89.5% which is the automobile class on VGG19 * Resnet model.

Table IX. Recall Measurement using Ensemble Hard-voting

No	Class	VGG16* VGG19	VGG16 * Resnet56	VGG19 * Resnet56	VGG16* VGG19* Resnet56
1	airplane	75.0	76.5	82.5	78.0
2	automo bile	82.5	83.0	89.5	85.0
3	bird	54.5	52.5	71	59.3
4	cat	49.5	56.0	62.5	56.0
5	deer	72.0	72.5	78.5	74.3
6	dog	53.5	52.5	68	58.0
7	frog	84.0	80.0	80	81.3
8	horse	74.5	74.0	81.5	76.7
9	ship	82.5	81.5	89	84.3
10	truck	80.0	80.5	88.5	83.0
Average		**70.8**	**70.9**	**79.1**	**73.6**

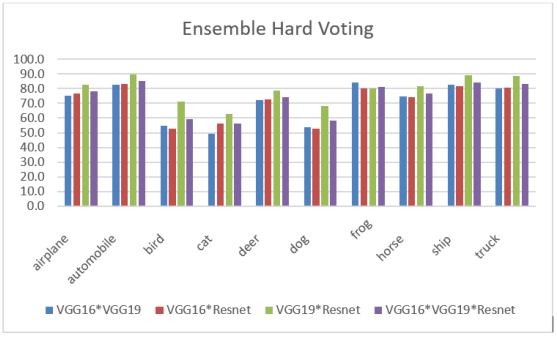

Fig.11. Recall from Ensemble Hard Voting

Table X. comparison of tested models using Cifar10 datasets.

No	Model	Precision (%)	Recall (%)
1	VGG16	79	79
2	VGG19	64	63
3	Resnet56	80	80
4	VGG16*VGG19 (Soft Voting)	79.1	79
5	VGG16*Resnet56 (Soft Voting)	79.8	80
6	VGG16*VGG19 (Soft Voting)	81.3	80.9
7	VGG16*VGG19*Resnet (Soft Voting)	81.4	80.9
8	VGG16*VGG19 (Hard Voting)	71.5	70.8
9	VGG16*Resnet56 (Hard Voting)	72.3	70.9
10	VGG16*VGG19 (Hard Voting)	80.0	79.1

IV. CONCLUSION

Image classification will benefit from classifiers with high accuracy. While a vast number of deep neural nets have been proposed, little has been said about building new models based on ensemble technique of existing classifiers.

Our experiment on Cifar10 shows that ensemble classifier using probability voting technique involving VGG16, VGG19, and Resnet56 models can increase the performance of the respective individual classifier. The future step of this experiment is to explore other techniques to build ensemble classifier using more testing datasets or classifiers for transfer learning.

REFERENCES

1. D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," Nips, pp. 1–9, 2012.
2. Y. L. Clement Farabet, Camille Couprie, Laurent Najman, "Learning hierarchical features for scene labeling," Pattern Anal. Mach. Intell. IEEE Trans., vol. 35, no. 8, pp. 1915–1929, 2013.
3. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 580–587, 2014.
4. A. Krizhevsky and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1–9, 2012.
5. A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., pp. 512–519, 2014.
6. M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013," Comput. Vision–ECCV 2014, vol. 8689, pp. 818–833, 2014.
7. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Int. Conf. Learn. Represent., pp. 1–14, 2015.
8. M. Lin, Q. Chen, and S. Yan, "Network In Network," arXiv Prepr., pp. 1–10, 2013.
9. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat : Integrated Recognition , Localization and Detection using Convolutional Networks," arXiv Prepr., pp. 1–15, 2013.
10. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," ArXiv e-prints, pp. 1–18, 2012.
11. A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," arXiv, pp. 1–14, 2016.
12. Z. He, "Deep Residual Learning for Image Recognition," arXiv.org e-Print Arch., vol. 7, no. 3, pp. 171–180, 2015.
13. G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely Connected Convolutional Networks," 2016.
14. J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," Nips, 2016.
15. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 580–587, 2014.
16. K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, 2015.
17. F. Perronnin et al., "Towards Good Practice in Large-Scale Learning for Image Classification To cite this version : Towards Good Practice in Large-Scale Learning for Image Classification," 2012.
18. Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 248–255, 2009.
19. E. N. Sup, "Understanding Deep Convolutional Networks," axXiv, pp. 1–17, 2016.

20. H. C. Shin et al., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," IEEE Trans. Med. Imaging, vol. 35, no. 5, pp. 1285–1298, 2016.
21. J. Wu and J. M. Rehg, Ensemble Machine Learning. 2012.
22. K. M. Ting and I. H. Witten, "Issues in stacked generalization," J. Artif. Intell. Res., vol. 10, pp. 271–289, 1999.
23. Y. H. Cao, "Study of Ensemble Model for Knowledge Management," Ecbi 2009 Int. Conf. Electron. Commer. Bus. Intell. Proc., pp. 33–36, 2009.
24. R. Cogranne, T. Denemark, and J. Fridrich, "Theoretical model of the FLD ensemble classifier based on hypothesis testing theory," 2014 IEEE Int. Work. Inf. Forensics Secur. WIFS 2014, pp. 167–172, 2015.
25. M. A.Tayebiyan, TA. Mohammad, AH. Ghazali, Syamsiah, "Artificial Neural Network for Modelling Rainfall-Runoff," Pertanika J. Sci. Technol., vol. 24, no. 2, pp. 319–330, 2016.
26. Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. Google Scholar
27. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, August 1997.
28. Miller, David J., and Lian Yan. "Ensemble classification by critic-driven combining." In Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, vol. 2, pp. 1029-1032. IEEE, 1999.
29. Cao, J., Kwong, S., Wang, R., Li, X., Li, K. and Kong, X., 2015. Class-specific soft voting based multiple extreme learning machines ensemble. Neurocomputing, 149, pp.275-284.
30. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.
31. S. Raschka. Python Machine Learning. Packt Publishing Ltd., 2015.
32. A. Galkin, from, https://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning on 25 Agt 2017.

AUTHORS PROFILE

Sarwo, Student Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480

Dr. Ir. Yaya Heryadi, M.Sc, Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta , Indonesia 11480.
https://scholar.binus.ac.id/d3803/dr-ir-yaya-heryadi/

Prof. Dr. Ir. Widodo Budiharto, S.Si., M.Kom., IPM, Computer Science Study Program, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480
https://scholar.binus.ac.id/D2637/widodo-budiharto/

Prof. Dr. Ir. Edi Abdurachman, MS., M.Sc. Computer Science Department, BINUS Graduate Program – Doctor of Computer Science,
Bina Nusantara University, Jakarta, Indonesia 11480
https://scholar.binus.ac.id/D0636/edi-abdurachman/