CNN Output Optimization for More Balanced Classification

Hyukdoo Choi
LG Electronics, Seoul, Korea

Abstract

This paper proposes a convolutional neural networks (CNN) output optimization method to improve accuracies of low-accuracy classes. Since CNN classifiers are trained by datasets whose data distributions of individual classes are not even or similar, they have always suffered from imbalanced classification performances against classes. In this study, CNN output probabilities are optimized by applying weights and biases as if there is an additional layer after the softmax. We formulated the equations to optimize weights and biases for performance improvement. A naïve optimization did not work well, so we devised a more elaborated optimization to focus on the competitive probability range and low accuracy classes. As a result, the classification accuracies of lowest 20% classes in accuracy are improved 1.27% on average while maintaining the total accuracy.

Keywords: Convolutional neural networks, Image classification, Probability optimization

1. Introduction

Convolutional neural networks (CNN) have made a breakthrough in the computer vision field, especially for image classification [1–6] and object detection [7–13] and segmentation [14–16]. Since this paper deals with image classification, especially for improving performances of low-accuracy categories, we start with reviewing a brief history of image classification by CNN. The innovation begun by Lecun et al. [1], where the initial structure of CNN, LeNet-5, was proposed and verified by hand-written digit classification. The real revolution was initiated by the AlexNet [2] which first won the ImageNet competition with neural networks. After the successful debut of AlexNet, development of CNN was accelerated by new structures, optimization techniques, and plenty of large-scale datasets.

The structure of CNN has been getting deeper and more complex. VGG-Net [3] stacked plain $3 \times 3$ conv layers deeply instead of larger kernels in AlexNet. The improved classification and localization performances benefited from the reduced number of parameters and more ReLU layers. VGG-Net stimulated researchers to explore extremely deep networks. GoogLeNet [5] is another most famous structure which introduced the “Inception module”. The inception module has the network-in-network structure [17] which deals with features of various scales. The entire structure has way deep layers more than hundred. ResNet [6] is considered the last innovation in the CNN layout. It has 152 layers, mostly series of plain $3 \times 3$ conv-ReLU layers but they are grouped in the “Residual blocks”. The residual block outputs the input features added by the features from a couple of conv layers. The added features are considered as residual features. Training residual blocks eased the learning because learning
Table 1. The statistics of the three datasets: the numbers of classes, and total, training, validation and test images

| Dataset  | Classes | Total   | Training | Validation | Test  |
|----------|---------|---------|----------|------------|-------|
| CIFAR10  | 10      | 60,000  | 40,000   | 10,000     | 10,000|
| CIFAR100 | 100     | 60,000  | 40,000   | 10,000     | 10,000|
| VOC2012  | 20      | 40,135  | 28,095   | 6,020      | 6,020 |

“residual” features is easier than learning new features and the gradient propagated effectively by the direct path to the input features. In this work, the latest variations were used as pre-trained networks, such as Inception-v4 and Inception-ResNet-v2 from [17], and ResNet-v2-50 and ResNet-v2-101 from [18]. Large-scale datasets supported and led these progresses. New techniques competed over the public datasets such as CIFAR10 [19], CIFAR100 [19], Pascal VOC [20], ImageNet [21], and MS-COCO [22]. We used CIFAR10, CIFAR100, and Pascal VOC2012 datasets to train and test CNN for image classification because they have sufficient number of images and categories with moderate volume. The statistics of the datasets are provided in Table 1.

Though the progress of image classification seems to saturate now, there remain a few problems. First, training the deep networks above with the large-scale datasets takes too much time. Training on the ImageNet dataset took a couple of weeks even with multiple state-of-art GPUs. Second, classification performances are not even for multiple categories. VOC 2012 dataset has 40,136 training annotations and 17,399 of them belong to the ‘person’ category. The smallest category is ‘bus’ and has 683 annotations. This unevenness makes the accuracy of ‘bus’ classification generally lower than that of ‘person’. On the other hand, the both CIFAR datasets contains 60,000 images and they are evenly distributed over 10 and 100 categories. However just even sizes of categories does not guarantees in similar accuracies of individual categories. In our training, per-class accuracies varied from 40% to 91%. Different distributions of categories in feature space also affected the performances of categories differently.

In this paper, we propose a method to quickly optimize CNN outputs to improve image classification performance, especially for low-accuracy categories. We added an optimization layer to the end of CNN’s output layers as shown in Figure 1. The weights and biases of the additional layers were optimized by the validation data. We gave different weights to categories in the optimization in order to improve the accuracies of low-accuracy categories. Rather than probability optimization, it is a more common idea that duplicate training data belonging to low-accuracy classes and retrain the network, but it takes too much time. Training the pre-trained deep networks can take several hours to days and training from the bottom may take a few weeks. However, our optimization is finished within a minute. In other words, the performance can be benefited with a small cost.

This paper is organized as follows. Section 2 introduce how to optimize weights and biases for CNN outputs. However, the naïve approach did not work well in practice. Section 3 presents a more elaborated method to help low-accuracy classes. The experimental results and conclusions are given in Sections 4 and 5.

2. CNN Output Optimization

If a CNN classifier is biased to specific classes, we can correct them by weighting the output probabilities. Our goal is to compute the optimal weights to derive the best performance from the fully trained CNN. With adding biases, it can be seen as attaching an additional layer to the final softmax layer as shown in Figure 1.

Our training process includes three steps. First, the given dataset is split into training, validation and test sets. Second, pre-trained models are fine-tuned by transfer learning with the training set. Third, the optimization layer is added to the CNN and the parameters of the additional layer is optimized for the validation set. Then we can evaluate the performance of the optimized CNN with the test set. The optimization problem is solved and the resulting performance is evaluated in the subsequent subsections.

2.1 Derivation of Optimal Parameters

Assume that there is a CNN that classifies images into $K$ classes. When an input image is fed to the network, it outputs a probability sample $\mathbf{p}_n = \left[ p_{n,1}, p_{n,2}, \ldots, p_{n,K} \right]^T$, where $n$ is
the input image index. When the ground-truth label index is \(c_n\), the target probability, \(p^*_n\), is defined as follows:

\[
p^*_n = \mathbf{p}^*_n (c_n) = \left[ p^*_{n,1} \ p^*_{n,2} \cdots p^*_{n,K} \right]^T, \quad (1)
\]

where \(p^*_{n,k}\) \(\begin{cases} 1 & \text{if } k = c_n, \\ 0 & \text{otherwise}. \end{cases}\)

The raw probabilities are corrected toward the target probabilities via an additional layer which is formulated as

\[
\hat{\mathbf{p}}_n = \mathbf{W} \mathbf{p}_n + \mathbf{b}, \quad (2)
\]

where \(\mathbf{W}\) is the weight matrix and \(\mathbf{b}\) is the bias vector of the additional layer. We aim to compute the parameters such that the distance between \(\hat{\mathbf{p}}_n\) and \(\mathbf{p}^*_n\) is minimized. For optimization convenience, the weight matrix and the bias vector are augmented into the parameter matrix \(\mathbf{H} = \left[ \mathbf{W} \ \mathbf{b} \right]\) and Eq. (2) is rewritten as follows:

\[
\hat{\mathbf{p}}_n = \mathbf{H} \mathbf{p}_n, \quad (3)
\]

where \(\hat{\mathbf{p}}_n = \left[ \mathbf{p}^*_n \ 1 \right]^T\) is the raw probability padded by one. The optimization of the parameter matrix is solved by minimizing the following function:

\[
\hat{\mathbf{H}} = \arg \min_{\mathbf{W}, \mathbf{a}} L (\mathbf{H}) \\
= \arg \min_{\mathbf{W}, \mathbf{a}} \sum_n (\mathbf{H} \hat{\mathbf{p}}_n - \mathbf{p}^*_n)^T (\mathbf{H} \hat{\mathbf{p}}_n - \mathbf{p}^*_n) \\
= \arg \min_{\mathbf{W}, \mathbf{a}} \| (\mathbf{H} \mathbf{p}^*_n - \mathbf{P}^* ) \|^2_F \\
= \arg \min_{\mathbf{W}, \mathbf{a}} tr \left( (\mathbf{H} \mathbf{p}^*_n - \mathbf{P}^* )^T (\mathbf{H} \mathbf{p}^*_n - \mathbf{P}^* ) \right), \quad (4)
\]

where \(\mathbf{P}^* = \left[ \mathbf{p}^*_1 \ \mathbf{p}^*_2 \cdots \mathbf{p}^*_N \right]\), \(\mathbf{P} = \left[ \hat{\mathbf{p}}_1 \ \hat{\mathbf{p}}_2 \cdots \hat{\mathbf{p}}_N \right]\), and \(\mathbf{H} \in \mathbb{R}^{K \times (K+1)}\). To obtain the optimal \(\hat{\mathbf{H}}\), Eq. (4) should be differentiated over \(\mathbf{H}\).

\[
\frac{\partial}{\partial \mathbf{H}} L (\mathbf{H}) = \frac{\partial}{\partial \mathbf{H}} tr \left( (\mathbf{H} \mathbf{p}^*_n - \mathbf{P}^* )^T (\mathbf{H} \mathbf{p}^*_n - \mathbf{P}^* ) \right) \\
= \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{P}^T \mathbf{H}^T \mathbf{H} \mathbf{p}^*_n - \mathbf{P}^T \mathbf{H}^T \mathbf{P}^* \right) \\
- \mathbf{P}^T \mathbf{H} \mathbf{P}^* \\
= \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{P}^T \mathbf{H}^T \mathbf{H} \mathbf{p}^*_n \right) - \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{P}^T \mathbf{H} \mathbf{p}^*_n \right) \\
- \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{P}^T \mathbf{H} \mathbf{P}^* \right) \\
= \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{H} \mathbf{P}^T \mathbf{H} \mathbf{p}^*_n \right) - \frac{\partial}{\partial \mathbf{H}} tr \left( \mathbf{H} \mathbf{P}^T \mathbf{P}^* \right)
\]

For Eq. (5) to be zero, the optimal parameter matrix is solved as follows:

\[
\hat{\mathbf{H}} = \mathbf{P}^* \mathbf{P}^T (\mathbf{P}^* \mathbf{P}^T)^{-1} = \mathbf{P}^* \mathbf{P}^T. \quad (6)
\]

The weights and biases from the parameter matrix are applied to the optimization layer and output probabilities of the original network are corrected by Eq. (2).

### 2.2 The Optimization Results

We evaluated the classification accuracy from the optimized probabilities and compared it with that of the raw probabilities. However, on the contrast to our expectation, the classification performance was almost never improved compared to the original network as shown in Table 2. The error values of \(L (\mathbf{H})\) in Eq. (4) and the classification accuracies are evaluated over combinations of the three datasets and the four CNN models. The details about the datasets and the CNN models will be given in Section 4.1. Though the optimization reduced the error function of \(L (\mathbf{H})\) by 2.9% on average, it did not increase the accuracies. As the result did not make sense to us, we analyzed the intermediate variables to find the reason.

The reason was that \(\hat{\mathbf{H}}\) was not radical enough to change the rankings of probabilities. When we take a look on the values of \(\mathbf{H}\), it was close to an identity matrix. As the majority of the validation set was true positive samples, it tended to preserve the current status. Figure 2 shows the histogram of labeled probabilities which are probabilities corresponding to the ground-truth labels. The majority of labeled probabilities are close to 1. Thus the optimization had little motivity to change the rankings. We concluded that this naïve optimization could not improve the performance and devised a more elaborated method described in the next section.

### 3. Elaborated Optimization

From the results of Section 2, we learned two lessons. If the optimization includes probability samples whose labeled probability is close to 1, high-extreme samples, it will not make a meaningful change. Secondly, we cannot make close-to-zero labeled probabilities to the top place among class probabilities...
Table 2. The classification accuracies and the error function values of \( L (\mathbf{H}) \) from the raw and optimized probabilities

| Dataset     | CNN model  | Accuracy (%) | Error       |
|-------------|------------|--------------|-------------|
|             |            | Raw | Opt. | Raw | Opt. |
| CIFAR10     | IncResNet  | 94.8 | 94.8 | 829.8 | 818.0 |
|             | InceptionV4 | 93.0 | 92.9 | 1,095.0 | 1,075.9 |
|             | ResNet-50  | 91.1 | 91.1 | 1,411.9 | 1,377.6 |
|             | ResNet-101 | 91.8 | 91.7 | 1,292.4 | 1,265.8 |
| CIFAR100    | IncResNet  | 75.3 | 75.3 | 3,704.4 | 3,532.6 |
|             | InceptionV4 | 69.3 | 69.4 | 4,379.9 | 4,196.4 |
|             | ResNet-50  | 65.4 | 65.5 | 4,983.7 | 4,727.8 |
|             | ResNet-101 | 66.2 | 66.4 | 4,818.0 | 4,590.5 |
| VOC2012     | IncResNet  | 92.4 | 92.6 | 720.5 | 706.5 |
|             | InceptionV4 | 91.5 | 91.5 | 780.2 | 767.1 |
|             | ResNet-50  | 88.6 | 88.7 | 1,040.8 | 1,015.5 |
|             | ResNet-101 | 89.3 | 89.3 | 981.8 | 956.9 |

The results are evaluated from combinations of the three datasets and the four CNN models.

Our initial motivation was to balance the classification accuracies of individual classes. We expected that there are many samples of low accuracy classes in the competing range and then low accuracy classes would be naturally benefited in the optimization. However, it was not always true when the number of samples belonging to classes are not even. To improve accuracies of low accuracy classes, we applied different class weights to probability samples. The class weights \( \mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_K \end{bmatrix} \) are defined as follows:

\[
 w_k = (\text{Acc}_k)^{-\beta},
\]

where \( \text{Acc}_k \) is the classification accuracy of the \( k \)-th class from the validation set and \( \beta \) is the exponent parameter to control the effect of accuracies. With the class weights, the error function in Eq. (4) is rewritten as follows:

\[
 L (\mathbf{H}) = \text{tr} \left( \left( (\mathbf{H} \mathbf{P}_R - \mathbf{P}_R^-) \mathbf{S}_R \right)^T \left( (\mathbf{H} \mathbf{P}_R - \mathbf{P}_R^-) \mathbf{S}_R \right) \right),
\]

where

\[
 \mathbf{S}_R = \begin{bmatrix} w_{c_1} & w_{c_2} & \cdots \end{bmatrix} \in \mathbb{R}^{N_R \times N_R},
\]

and subscript \( R \) means that it includes samples collected only from the competitive range and \( N_R \) is the number of the collected samples.

However, Eq. (8) may result in unstable result since there are only a small number of competitive samples. Some classes may not be included in the competitive samples or only a few samples are included. It can result in a biased optimization that almost ignores minor classes. To stabilize the optimization, the probability and weight matrices are padded by identity matrices.
where $I_K$ is an identity matrix with a size of $K$, $1_{1 \times K}$ is a row vector of ones with a size of $K$ and $\alpha$ is the stabilization parameter. To solve the optimal $H$ with sample selection, accuracy weighting and padding, Eq. (3) is differentiated by $H$ as follows:

$$
\begin{align*}
\frac{\partial}{\partial H} L(H) &= \frac{\partial}{\partial H} tr \left( (\hat{H}P_R - P_R^*)S_R \right)^T \left( (\hat{H}P_R - P_R^*)S_R \right) \\
&= \frac{\partial}{\partial H} tr \left( S_R^T (\hat{P}_R^T H^T \hat{H}P_R - \hat{P}_R^T H^T P_R^*) - \hat{P}_R^T H^T P_R^*S_R \right) \\
&\quad - \frac{\partial}{\partial H} tr \left( S_R^T \hat{P}_R^T H^T \hat{P}_R S_R \right) \\
&= \frac{\partial}{\partial H} tr \left( H^T P_R^* S_R \right) - \frac{\partial}{\partial H} tr \left( H^T P_R^* S_R \right) \\
&= - \left( \frac{\partial}{\partial H} tr \left( H^T P_R^* S_R \right) \right) \\
&= 2 \left( H^T P_R^* S_R \right)^T.
\end{align*}
$$

For Eq. (11) to be zero, the optimal $H$ is solved by

$$
H = P_R^* S_R S_R^T P_R^T (P_R S_R S_R^T P_R^T)^{-1} = P_R^* S_R (P_R S_R)^{\dagger},
$$

which is similar to Eq. (5) except for the class weight matrix. The parameter matrix is then applied to the raw probabilities whose maximum probability is less than $R_{\text{high}}$ to improve the classification accuracy in the competitive range.

### 4. Performance Evaluation

Now the elaborated optimization method is supposed to be evaluated and compared with the pure CNN. For reliable evaluation, the proposed method was tested by three different problems with four different CNN models. In our result, the main performance metric is not a total classification accuracy. Instead, we use a mean classification accuracy of lowest 20% classes in accuracy, named low 20% accuracy, as a main metric. The total accuracy does not change much because majority of samples belong to extreme samples. On the other hand, since our optimization reduces false negative or false positive samples in the competitive range, the low 20% accuracy is more likely to be improved, while the total accuracy almost stands still. The preparation process for the evaluation is reviewed first, and then the evaluation results are going to be presented.

#### 4.1 Preparations

The preparation process for the CNN optimization is three-folds: dataset, transfer learning, and parameter settings. Let us describe them one by one.

#### 4.1.1 Dataset

We used three datasets, which means that our method was applied to three classification problems. The datasets were CIFAR10 [19], CIFAR100 [19], and VOC2012 [20]. The CIFAR datasets provide tiny images of $32 \times 32$ pixels belonging to 10 and 100 categories, respectively. VOC2012 dataset is provided by the PASCAL VOC project and contains various sizes of images with object annotations. The annotation consists of category names and bound box information. We trained networks with images extracted from the bound boxes, so the number of training images is larger than the number of images in the dataset. In addition, as the test set in VOC2012 is not fully annotated, we used only the training set and split it into training, validation and test sets for our use. The statistics of the datasets are summarized in Table 1.

#### 4.1.2 Transfer learning

We retrained the pre-trained models to solve the problems given by the datasets. The pre-trained models were Inception-ResNet-V2, Inception-V4, ResNetV2-50 and ResNetV2-101, which are denoted by IncResNet, InceptionV4, ResNet-50, ResNet-101, respectively in Table 2. The models were downloaded from the TF-slim repository and trained by the three datasets respectively. As a result, we had the twelve trained CNN models.

The transfer learning was conducted by the Tensorflow library in Ubuntu 16.04 with Titan X GPU. The training was controlled by the RMSprop method. The initial learning rate was 0.01 and the decay term of the RMSprop was 0.9. The trainings were proceeded over 35,000 steps with the batch size of 32.
4.1.3 Optimization parameters

The elaborated method can make an improvement over the pure CNN and the naïve optimization in Section 2, which will be verified in the next subsection. Instead, we had to manually adjust some parameters, such as the competitive range \((R_{\text{low}}, R_{\text{high}})\), the exponent of accuracy \(\beta\), and the stabilization parameter \(\alpha\). The parameter values selected for evaluation were \((R_{\text{low}}, R_{\text{high}}) = (0.1, 0.8)\), \(\beta = 1.5\), and \(\alpha = 3\). Figure 3 shows the low 20% accuracies with different parameter values. The four subfigures correspond to the four parameters each. The parameter values varied in the vicinity of the selected values. When one parameter is varying, the other parameters are fixed at the selected values. The accuracies in Figure 3 are means of twelve low 20% accuracies from the twelve combinations of datasets and CNN models. The figure shows that our selection of the parameters is at least on the local maximum.

4.2 Performance Comparison

We evaluated total and low 20% accuracies with the twelve CNNs prepared as in the previous section. The results are summarized in Table 3. Overall, there are only little difference between total accuracies of the raw and elaborated methods but clear difference between low 20% accuracies. Specifically, the total accuracy increased 0.04% while the low 20% accuracy increased 1.27% on average. The performance improvement differed mainly with datasets. Low 20% accuracies of CIFAR10 increased only 0.4% because low 20% accuracies of the raw CNNs was already high, which was 85.3%. There was little space to be improved. On the other hand, low 20% accuracies were improved 1.8% and 1.6% on average from the CIFAR100 and VOC2012 datasets respectively. Their accuracies before the optimization were relatively low, 48.3% and 74.1% respectively. As the number of classes increases, classification accuracies are generally degraded.

Figure 4 shows images that are classified as wrong classes by the raw CNN but corrected to right classes by the probability optimization. It shows label names with its probability under images. As we focused on the competitive probability range, \((0.1, 0.8)\), the probabilities of the changed samples are generally low despite they are the highest probabilities among classes. The wrong results of the raw CNN are written on the first lines of annotations but their failures are understandable. Most of them are changed to ‘close’ classes, for example, ‘airplane’ to ‘bird’, ‘man’ to ‘woman’, ‘leopard’ to ‘tiger’ and so on. In addition, the fourth image in the bottom row has both ‘chair’ and ‘person’. As the CIFAR datasets provides low resolution images, it is difficult to classify even for human beings. Our optimization method helped those ambiguous cases, especially for classes with low classification accuracy.

However, since the total accuracy was not improved while low 20% accuracy increased, there are many adversarial examples. Figure 5 shows the wrongly changed examples. Similarly

Table 3. Classification accuracies of total and low 20% classes from the raw and elaborately optimized CNN

| Dataset   | CNN model | Total (%) | Low 20% (%) |
|-----------|-----------|-----------|-------------|
|           | Raw       | Elab.     | Raw         | Elab.     |
| CIFAR10   | IncResNet | 94.8      | 94.7        | 89.0       | 89.1       |
|           | InceptionV4 | 93.0      | 93.0        | 86.3       | 86.7       |
|           | ResNet-50 | 91.1      | 91.2        | 82.5       | 82.9       |
|           | ResNet-50 | 91.8      | 92.0        | 83.5       | 84.1       |
| CIFAR100  | IncResNet | 75.3      | 75.2        | 56.7       | 57.3       |
|           | InceptionV4 | 69.3      | 69.3        | 49.3       | 50.8       |
|           | ResNet-50 | 65.4      | 65.1        | 43.4       | 45.4       |
|           | ResNet-50 | 66.2      | 66.4        | 43.7       | 46.7       |
| VOC2012   | IncResNet | 92.4      | 92.3        | 77.4       | 78.4       |
|           | InceptionV4 | 91.5      | 91.6        | 75.7       | 77.5       |
|           | ResNet-50 | 88.6      | 88.8        | 71.1       | 73.3       |
|           | ResNet-50 | 89.3      | 89.4        | 72.3       | 73.9       |
Figure 4. Examples of correctly changed classification results by the probability optimization. In the annotations below images, first lines represent the result of the raw CNN and the second lines represent the result of the probability optimization which are correct.

to Figure 4, most of the changes make sense to us while abrupt or absurd changes are minor.

5. Conclusions

We proposed the probability optimization method for CNN classifiers. The naïve optimization was tried first but it did not worked. Then we tried a more elaborated method to improve the accuracies of low accuracy classes. The elaborated method focused on the competitive probability range and weighted classes differently with their accuracies from validation sets. As a result, low 20% accuracy was improved by 1.27% while maintaining the total accuracy.

The proposed method can have a meaning when we need a more balanced classifier whose accuracies of individual classes are relatively even. It can be applied to commercial applications where classification failures of specific classes cost large expenses such as autonomous driving systems.
Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

I am really grateful to Professor Euntai Kim in Yonsei University for supporting this research and helping derivation of equations.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. [https://doi.org/10.1109/5.726791]

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

[3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015. Available [https://arxiv.org/pdf/1409.1556/]

[4] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in Proceedings of 2014 European Conference on Computer Vision, Zurich, Switzerland, 2014, pp. 818-833. [https://doi.org/10.1007/978-3-319-10590-1_53]

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, 2015, pp. 1-9. [https://doi.org/10.1109/CVPR.2015.7298594]

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, 2016, pp. 770-778. [https://doi.org/10.1109/CVPR.2016.90]

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 580-587. [https://doi.org/10.1109/CVPR.2014.81]

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in 2014 Proceedings of European Conference on Computer Vision, Zurich, Switzerland, 2014, pp. 346-361. [https://doi.org/10.1007/978-3-319-10578-9_23]

[9] R. Girshick, “Fast R-Cnn,” in Proceedings of 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 2015, pp. 1440-1448. [https://doi.org/10.1109/ICCV.2015.169]

[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” Advances in Neural Information Processing Systems, pp. 91-99, 2015.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: single shot multibox detector,” in Proceedings of 2016 European Conference on Computer Vision, Amsterdam, The Netherlands, 2016, pp. 21-37. [https://doi.org/10.1007/978-3-319-46448-0_2]

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection,” in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, 2016, pp. 779-788. [https://doi.org/10.1109/CVPR.2016.91]

[13] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” 2016. Available [https://arxiv.org/abs/1612.08242].

[14] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected CRFs,” 2014. Available [https://arxiv.org/abs/1412.7062]

[15] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, 2015, pp. 3431-3440. [https://doi.org/10.1109/CVPR.2015.7298965]

[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” 2017. Available [https://arxiv.org/abs/1703.06870]
[17] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” 2016. Available https://arxiv.org/abs/1602.07261

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proceedings of 2016 European Conference on Computer Vision, Amsterdam, The Netherlands, 2016, pp. 630-645. https://doi.org/10.1007/978-3-319-46493-0_38

[19] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009. Available http://www.cs.toronto.edu/\~kriz/learning-features-2009-TR.pdf

[20] M. Everingham, L. van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL visual object classes (VOC) challenge,” International Journal of Computer Vision, vol. 88, no. 2, pp. 303-338, 2010. https://doi.org/10.1007/s11263-009-0275-4

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., “Imagenet large scale visual recognition challenge,” International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, 2015. https://doi.org/10.1007/s11263-015-0816-y

[22] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft coco: common objects in context,” in Proceedings of 2014 European Conference on Computer Vision, Zurich, Switzerland, 2014, pp. 740-755. https://doi.org/10.1007/978-3-319-10602-1_48

Hyukdoo Choi received the B.S. and Ph.D. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2009 and 2014, respectively. He is currently a senior researcher in L&A Research Center of LG Electronics from 2014. His main research interests include machine learning, computer vision, simultaneous localization and mapping (SLAM) and deep learning.

E-mail: goodgodgd@yonsei.ac.kr

www.ijfis.org