J Electr Eng Technol.2018; 13(6): 2468-2478
http://doi.org/10.5370 /JEET.2018.13.6.2468

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

Sparse Feature Convolutional Neural Network with Cluster Max Extraction for Fast Object Classification

Sung Hee Kim*, Dong Sung Pae*, Tae-Koo Kang, Dong W. Kim*** and Myo Taeg Lim†**

Abstract – We propose the Sparse Feature Convolutional Neural Network (SFCNN) to reduce the volume of convolutional neural networks (CNNs). Despite the superior classification performance of CNNs, their enormous network volume requires high computational cost and long processing time, making real-time applications such as online-training difficult. We propose an advanced network that reduces the volume of conventional CNNs by producing a region-based sparse feature map. To produce the sparse feature map, two complementary region-based value extraction methods, cluster max extraction and local value extraction, are proposed. Cluster max is selected as the main function based on experimental results. To evaluate SFCNN, we conduct an experiment with two conventional CNNs. The network trains 59 times faster and tests 81 times faster than the VGG network, with a 1.2% loss of accuracy in multi-class classification using the Caltech101 dataset. In vehicle classification using the GTI Vehicle Image Database, the network trains 88 times faster and tests 94 times faster than the conventional CNNs, with a 0.1% loss of accuracy.

Keywords: Deep learning, Online-training control, Object recognition, Classification.

1. Introduction

Object classification is used in various research fields, such as computer vision [1] and pattern recognition [2]. In vision processing fields, the classifier changes the image information into a computation-available form as [19] and [20], since the computer cannot recognize the raw image as vision information. The transformed data are used for various applications, e.g. object recognition for autonomous driving [3], visual tracking [4], visual navigation [5], detection system of self-driving cars [6], etc.

Visual object classifiers are composed of two parts, a feature extractor and an object classifier. The feature extractor represents the input image using a descriptor while the object classifier sorts the extracted feature with an output label.

Recently, big data with image dataset such as Imagenet, [21] which has large scale and multi-class data is hard to handle with conventional visual classifiers. As the data keeps getting bigger, deep learning is proposed as the solution tool for big data analysis and prediction [22] and deep learning visual object classifiers showed overwhelming recognition accuracy. In 2012, the network proposed by Alex et al. [7], which works by applying convolution and pooling methods with the Rectified Linear Unit (ReLU) activation function in an image to extract the object feature, showed great results in large scale multi-class visual classification. Since the performance of Convolutional Neural Network (CNN) was demonstrated in [7], the application of convolution and pooling methods in deep neural networks has been researched extensively as in [8]-[10]. In contrast to prior feature extractors that extracted fixed handmade features, deep-learning classifiers learn the object features by back-propagating the learning of neural networks. Unlike previous visual classifiers, autonomy in the classification process to determine an optimal network through training dictates improvement in the performance of CNN.

Although deep neural networks show high classification accuracy, the large volume of the network leads to inefficiencies in time and data management. Due to the enormous network, the processing time and calculation cost are extensive, making real-time applications such as on-line training or online machine learning difficult. We propose the Sparse Feature Convolutional Neural Network (SFCNN), which reduces the volume of the network by producing a sparse feature map through the extraction of meaningful value exclusively from the full feature map of the CNN. Conventional CNNs are composed of two parts, a feature extraction through convolution and pooling, and an object classification using the fully connected neural network. Recent studies introduced methods to improve the performance of the feature extractor [17] and object classifiers [18], but most studies were focused on improving the accuracy. We add a feature reduction process to the

† Corresponding Author: Dept. of Electrical Engineering, Korea University, Korea. (mlim@korea.ac.kr, tkkang@smu.ac.kr)
* Dept. of Electrical Engineering, Korea University, Korea. ({haha217 paeds915}@korea.ac.kr)
** Dept. of Digital Electronics, Inha Technical College, Korea. (dongwon.kim25@gmail.com)
*** Dept. of Information and Telecommunication Engineering, Sangmyung University, Korea. (tkkang@smu.ac.kr)
Received: March 27, 2018; Accepted: May 28, 2018

conventional CNN to develop SFCNN. Feature reduction is performed between the output of feature extraction part and the input of object classification part. During the feature reduction process, the network selects the appropriate values to reduce the network volume with minimum performance loss. Through visualization of the feature map, we find that the extracted features converge on several regions, indicating that meaningful values exist in specific domains. To exclude meaningless values such as wide-range zero values and retain only feature-representative values, we produce a sparse feature map using region-based feature reduction methods, specifically cluster-max extraction and local-max extraction.

For evaluation, we compare the classification accuracy and processing time reduction for these two region-based feature reduction methods using the Caltech101 dataset. We also compare the multi-object classification performance of the proposed network to the performance of two conventional CNNs, Alexnet [7] and the VGG16 Network [11], using the Caltech-101 dataset. From the experimental results, we show that the proposed network has similar accuracy to the conventional CNNs, but the smaller network requires less processing time for both training and testing. Application to vehicle detection is conducted to measure the accuracy and processing time of the proposed network using the GTI Vehicle Image Database. The experimental result is also compared with those of the Alexnet and VGG16 networks, demonstrating the potential for application to real-time vehicle detection systems.

The rest of this paper is composed as follows. In Section 2, we review the conventional CNN. In Section 3, we discuss the proposed network, SFCNN. In Section 4, we present the experimental results to demonstrate the performance of our proposed network. We conclude the paper in Section 5.

2. Review of conventional CNN

The conventional CNN model is an autonomous and effective network for object recognition. The network mimics the human neuron recognition process with two object recognition stages. First, the network extracts local range feature using convolution and pooling layers. Afterward, it sorts the data into learned labels with a global range using a fully-connected classifier. Fig. 1 shows an overview of the standard CNN model.

In the convolutional stage, the 2D-convolution of input image I with the filter W_C is calculated to print the local response of I into W_C. Since I is fixed data and W_C is a network-trained variable, the characteristics of W_C determine the output of the convolutional process. In deep learning networks, the value of the filter W_C is modified to a conditional-optimized value through a back-propagation method. We will discuss the training method implemented afterward. Additionally, to implement the neuron model of CNN, a nonlinear function such as ReLU or a sigmoid function is processed to a convolutional response. The resulting layer for the convolutional network $\mathbf{L}_{conv}$ can be calculated using the following equation:

$$\mathbf{L}_{conv} = f\left(I * W_C\right), \tag{1}$$

where $f(x)$ is nonlinear function.

To summarize and concentrate $\mathbf{L}_{conv}$ with nearby output, the max-pooling layer is followed by the convolution layer. In the max pooling layer, a maximum value of 2×2 kernel is extracted as the output. Defining the single $m\times n$ array of $\mathbf{L}_{conv}$ as L_{conv}, the $i\text{-}th, j\text{-}th$ value for the $\frac{m}{2}\times\frac{n}{2}$ array feature map F, $F(i,j)$, is calculated as follows:

$$\begin{aligned} F(i,j) = \max(&L_{conv}(2i,2j), L_{conv}(2i,2j+1),\\ &L_{conv}(2i+1,2j), L_{conv}(2i+1,2j+1)), \end{aligned} \tag{2}$$

in the range of $1 \le i \le \frac{m}{2}$, $1 \le j \le \frac{n}{2}$. After the process is completed for every value of layer $\mathbf{L}_{conv}$, the full feature map $\mathbf{F}$ is calculated as a result of the feature extraction process.

To classify the output feature map layer $\mathbf{F}$, regional information $\mathbf{F}$ should be connected with the full data range as a plain neural network. Consequently, the multi-dimensional feature map is altered to 1-dimensional data in this process. Therefore, a 3-dimensional layer $\mathbf{F}$ with size $w\times h\times d$ is reshaped as a 1-dimensional array $\boldsymbol{F}$ with size $N = w\cdot h\cdot d$. The output value of the fully-connected layer l is calculated with fully-connected network weights W_f as follows :

$$l(j) = \sum_{i=1}^{N} W_f(j,i)\cdot \boldsymbol{F}(i), \tag{3}$$

and the final output label y is calculated as:

$$y = \arg\max_{l}(softmax(l)). \tag{4}$$

As in the convolutional stage, the network weight W_f determines the classification result, and the network optimizes the weight to create a powerful network. For convenience, we define the full network weight including W_C and W_f as weight W. Furthermore, the weight of the $i\text{-}th$ training step is defined as W_i. Like standard neural networks, CNN trains its weight using the back-

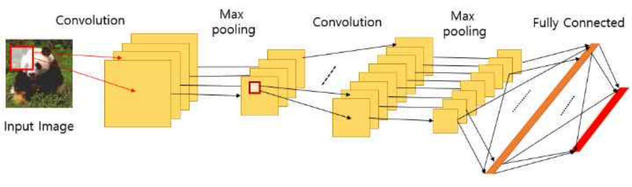


Fig. 1. Structure of the standard CNN

propagation method. Since the purpose of the network is accurate classification, the training progresses by finding an optimized W that has a stochastic gradient descent with momentum [12] and optimizing the multinomial logistic regression using mini-batch. This method is also used in [7] to find an optimized weight *w*, and the updated rule is stated in [7] as follows:

$$V_{i+1} := 0.9 \cdot V_i - 0.0005 \cdot \varepsilon \cdot W_i - \varepsilon \cdot \left\langle \frac{\partial L}{\partial W}\Big|_{W_i} \right\rangle_{D_i}, \tag{5}$$

$$W_{i+1} := W_i + V_{i+1}, \tag{6}$$

where i = iteration index, V = momentum variable, ε = learning rate.

3. Sparse Feature Convolutional Neural Network

3.1 Overview of the SFCNN

The proposed SFCNN is composed of three parts: Pre-trained CNN for feature extraction, Sparse feature map extraction for feature reduction and Fully-connected neural network for classification. The full structure of the network is shown in Fig. 2. Since earlier studies have demonstrated good feature extraction accuracy and feature training requires a significant amount of time, substituting the training process with a pre-trained system is a reasonable choice compared to training a novel network. We use a pre-trained CNN weight to produce a feature map instead of training the convolutional feature weight W_C . In the feature reduction process, we produce a sparse feature map layer $\mathbf{F}_{spr}$ from the full feature map of the previous process with region-based maximum extraction. In the training process, we train the fully-connected network with stochastic gradient descent using momentum to assign the object to the correct label.

3.2 Feature Extraction with Pre-trained CNN

From Section 2, the feature map layer F for the conventional CNN is extracted by passing input image I through the convolution and pooling layers. To extract a

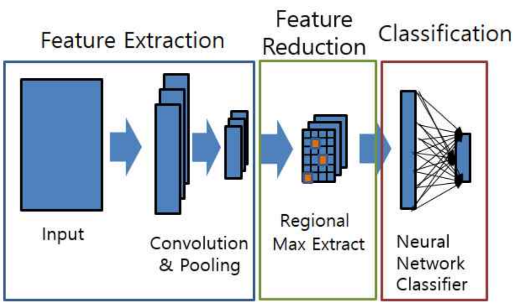


Fig. 2. Overview of the SFCNN

classification-optimized F, the network trains the convolutional weight W_C of the convolutional layer using back-propagation methods in the training step. Although training W_C for the corresponding network results in the best performance, the training step is a time-consuming task in deep learning. Alternatively, using a pre-trained network weight to generate a feature map instead of training the full network weight has been proposed in recent studies, and the training process is simplified with good experimental results as shown in [14-16]. We chose to use the pre-trained filter Wp for feature extraction instead of training the optimal convolutional weight W_C. In this paper, Wp is chosen as the pre-trained weight from the VGG16 network [11]. The convolutional response also requires a nonlinear function to mimic the neural network as stated in Section 2, and we chose ReLU as the activation function as in [11]. Therefore, the feature extraction part of the network goes through the same network structure as [11], and the feature map layer of the proposed network $\mathbf{F}$ is approximately calculated as :

$$\begin{aligned} \mathbf{F}_1 &= \mathrm{ReLU}\left(I * Wp_1\right), \\ \mathbf{F}_m &= \mathrm{ReLU}\left(\mathbf{F}_{m-1} * Wp_{m-1}\right), \\ \mathbf{F}_{n+1} &= \max pool\left(\mathrm{ReLU}\left(\mathbf{F}_n * Wp_n\right)\right), \end{aligned} \tag{7}$$

where the function $\max pool()$ is a simplified statement from Eq. (2), $\mathbf{F}_t$ is the t-th output layer of the feature extraction process, and Wp_t represents the t-th pre-trained weights for the VGG16 network, with a range of $n = 2,4,7,10$ and $m = 3,5,8,9,11,12,13$.

3.3 Feature reduction using region-based features

In CNN feature extraction, the feature map is extracted as the output of the convolution and pooling processes. For example, as stated in Section 3.1, each $\mathbf{F}_n$ is drawn as the convolutional calculation of the input, input image I or prior feature map $\mathbf{F}_{n-1}$, and the pre-trained filter Wp_n . Since the 2D-convolution process operates as a vision filter, a larger output value for $\mathbf{F}_n$ means a stronger response of the input data to the network filter Wp. In the CNN classifier, the object classifier section determines the output label with the final response for the convolutional feature weights. However, the extracted feature map often includes meaningless information, such as a zero response to the convolutional result with network filters. As shown in Fig. 3, during the visualization of the final layer in the feature extraction process in the VGG16 network, strong responses converge in only a few domains while the zero-value domain is widely represented. Instead of substituting a full large feature map into the object classifier, using only meaningful information on significant domains as the input for the posterior part of the network enables reduction of the network volume and the computational power without a large performance loss. In this paper, the generation of a sparse feature map using values from the strong-response domain is proposed for feature reduction. To implement a region-based sparse feature map from the full feature map $\mathbf{F}_n$, we choose two methods for extraction of the domain representation: local max extraction and cluster max extraction.

3.4 Sparse feature map produced using cluster max

In Fig. 3, the visualization result shows strong outputs from the feature map converge in several regions, and each response domain includes specific features that indicate similar characteristics. We define the concatenated region without the zero response of a feature map as a feature cluster, and each feature cluster represents a single feature. To implement the sparse feature map stated in Section 3.3, the most straightforward and simplest method is achieved by retaining only the maximum value of the feature cluster. The extracted values represent each feature cluster, which means that single values contain regional information. Fig. 4 shows the cluster maximum values extracted from a single full feature map. The actual representative values are the maximum values as shown in the 3-dimensional feature map in Fig. 4(a), and the locations of the values are illustrated on the original feature map in Fig. 4(b).

To obtain the cluster max value and a sparse feature map,

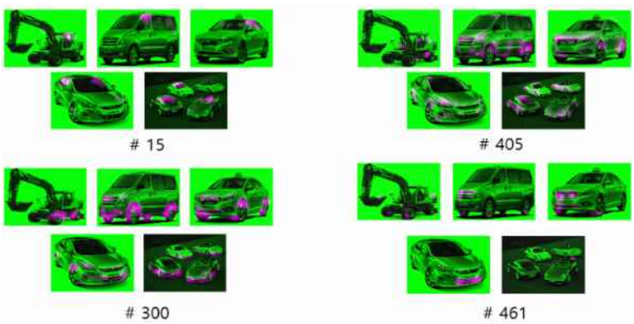


Fig. 3. Examples of Visualization for several feature responses over original image

we first define the input for the full feature map. The final feature map from the feature extraction process $\mathbf{F}_{final}$ can be stated as follows:

$$\mathbf{F}_{final} = \left(F_1, F_2, \cdots, F_d\right), \tag{8}$$

where F_i is the i-th feature map for the $x \times y \times d$ dimensional $\mathbf{F}_n$. F_i can also be denoted as:

$$F_i = \left\{C_1, C_2, \cdots, C_{N_i}\right\}, \tag{9}$$

where N_i is the number of feature clusters for F_i and C_t is the set of values included in the t-th feature cluster. From Eq. (8), a sparse feature map layer with cluster max $\mathbf{F}_{spr\_cls}$ is given by :

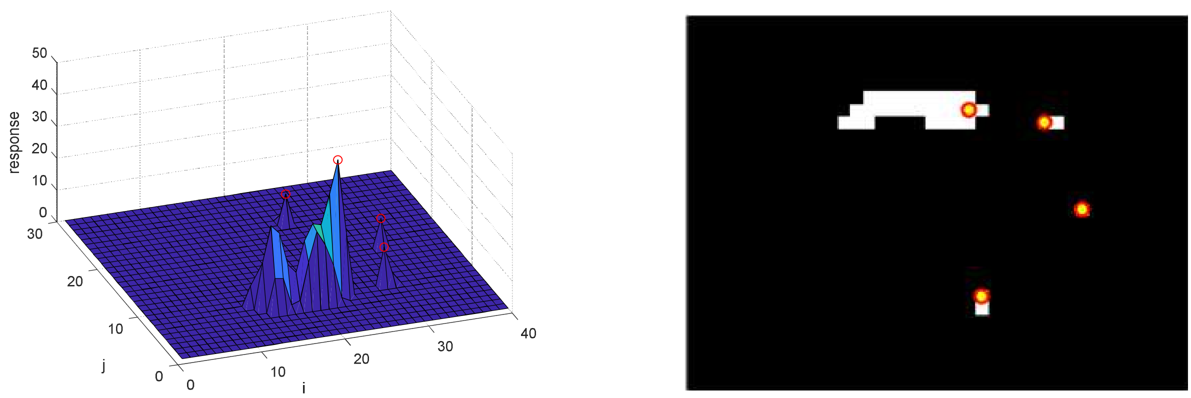
$$\begin{aligned} \mathbf{F}_{spr\_cls} &= \left(R_1, R_2, \cdots, R_d\right), \\ R_i &= \max_k \left(\max C_1, \max C_2, \cdots, \max C_{N_i}\right), \end{aligned} \tag{10}$$

where R_i represents the regional maximum features extracted from F_i and k is the number of extracted features. Since k determines the size of the sparse feature map, the network volume is altered by the value of k. Thus, an experiment to determine the network value is conducted in Section 4

3.5 Sparse feature map produced using local max

Although cluster max extraction provides the simplest representation of the feature map, situations in which a single cluster contains more than a single important piece of information can occur. In this case, the cluster maximum data is insufficient to represent the full set of region-based features. To overcome the possibility of poor performance, our second proposal for feature reduction is local max extraction. As shown in Fig. 5, local max values contain more maximum data than cluster max values.

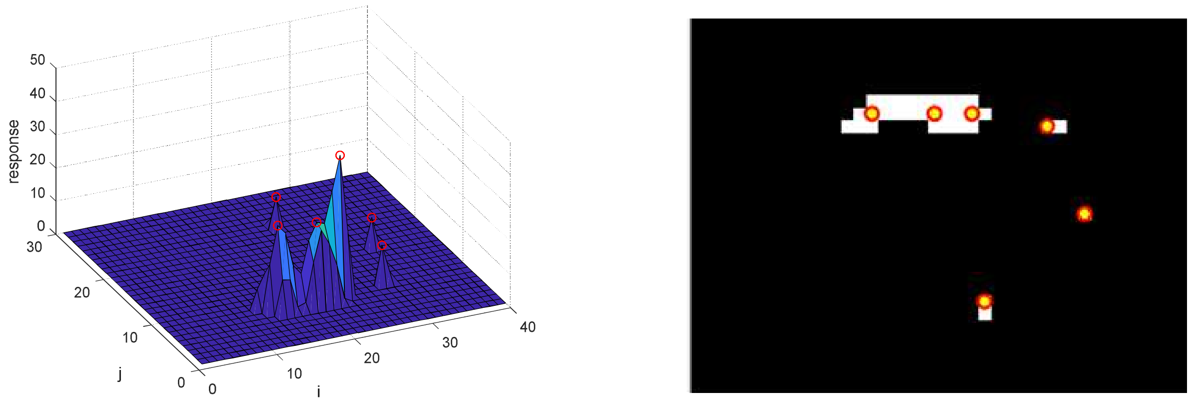
Consulting Eqs. (8) and (10), a sparse feature map based on local max extraction $\mathbf{F}_{spr\_lmx}$ can be calculated as follows:



(a) Cluster max values on 3D feature map

(b) location of values on 2D feature map

Fig. 4. Visualization of cluster max value over feature map



(a) Local max values on 3D feature map (b) Location of values on 2D feature map

Fig. 5. Visualization of local max value over feature map

$$\mathbf{F}_{spr\_lmx} = \left(R_1, R_2, \cdots, R_d\right),\quad R_i = \max_k\left(L_i\right), \tag{11}$$

where L_i is a set including the values of $F_i(m,n)$ for all m,n that satisfy the local maximum point condition, $\nabla F(m,n) = 0, D(m,n) > 0$ *and* $F_{xx}(m,n) < 0$.

Local max extraction collects all peak values for regions with complex calculations, including the gradient computation process for the feature map. More data is also computed for this process than for simple binary clustering and maximum extraction. Therefore, the maximum extraction method proposed in this section takes longer than the cluster max extraction described in Section 3.4, but has better classification performance. Since the two proposed methods have complementary processing time and classification accuracy, we need to choose a feature reduction method that considers both elements to implement the optimal SFCNN. An experiment is conducted to compare the performance of these methods in Section 4.

3.6 Classification with neural network

After $\mathbf{F}_{spr}$ is extracted from Eqs. (10) and (11), the classification to print the label output progresses in the classifier. We train the fully-connected weight W_f of the classifier using stochastic gradient descent with the momentum given in Eqs. (4) and (5). As shown in the fully-connected calculation in Section 2, the final layer $\mathbf{F}_{spr}$ is reshaped to a 1-dimensional array $\boldsymbol{F}_{spr}$ with size $N = k \cdot d$ to compute the output of the object classifier. Consequently, the trained network predicts the output label y as follows:

$$y = \arg\max_j(softmax(\sum_{i=1}^{N} W_f(\mathrm{j}, i) \cdot \boldsymbol{F}_{spr}(i))) \tag{12}$$

4. Experimental Results

In this section, we conduct experiments to evaluate the proposed network. As stated in Section 3, the methods used for producing the sparse feature map are complementary in their classification ability and extraction speed. We compare both elements in this section to show how much loss occurs in both metrics. After drawing a comparison of the performance, we choose a more reasonable method to produce the sparse feature map and implement SFCNN. To validate the performance of SFCNN, we measure the network classification accuracy and processing time for training and testing with a multi class dataset from Caltech 101. Two conventional CNNs are used for the comparison with SFCNN: Alexnet [7] and the VGG16 Network [11]. For application, we conduct vehicle detection using a binary car dataset from the GTI Vehicle Image Database in comparison with conventional CNN networks.

Since our main proposal is the reduction of network volume, the evaluation of feature extraction is not required. Instead of training the full network, we use pre-trained features included in the MATLAB Neural Network Toolbox and train and evaluate the network with the same toolbox. We used the initial learning rate with 0.01 and reduced the learning rate by a factor of 0.1 every 5 epochs for accurate training. We trained the network for roughly 30 epochs through whole training/test dataset.

4.1 Metrics

To complete the experiment on the proposed network, we need to define the valuation basis of the evaluation elements. We define the valuation standards as follows:

$$Accuracy = \frac{\#\, images\ correctly\ classified}{\#\, whole\ images}, \tag{13}$$

where '# images correctly classified' indicates the number of predicted labels y from Eq. (12), which is the same as the target labels from the dataset, and '#whole images' indicates the number of full dataset images.

$$\text{Feature reduction time} = \text{seconds to produce } \mathbf{F}_{spr}, \tag{14}$$

$$\text{Training time} = \text{seconds for object classifier training}, \quad (15)$$

$$\text{Test time} = \text{seconds for classifying test set images}. \quad (16)$$

Eqs. (13)-(16) are used as metrics for subsequent experiments. Defined accuracy metric (13) means ratio of correctly labeled data to the full dataset with evaluating network, and implies how accurately the network classifies the data. Feature reduction time is a metric used to evaluate and choose the proper method for the proposed network. Temporal metrics (14)-(16) are measured with Intel® Core™ i5-6600 CPU 8GB.

4.2 Comparison of experimental results for region-based feature reduction methods

To implement the SFCNN, we devise two region-based value extraction methods proposed in Section 3 to produce a sparse feature map from the full feature map. Although an estimation of the complementary characteristics in the proposed methods is possible, assuming the actual results

Table 1. Experimental results for comparison of feature reduction methods

(a) Classification accuracy

method \ k	1	2	3	4	5	6	7	8	9	10
cluster max extraction	0.9152	0.9137	0.9145	0.9138	0.9139	0.9134	0.9132	0.9143	0.9145	0.9144
local max extraction	0.9228	0.9245	0.9239	0.9252	0.9233	0.9249	0.9242	0.9241	0.9252	0.9235

(b) Processing time

Method \ Data size	30	50	Perimage (average)
cluster max extraction	78	129	2.59
local max extraction	1196	1984	39.773

Table 2. Experimental results for different k values

(a) Accuracy

k \ #	1	2	3	4	5	6	7	8	9	10	Average
1	0.918824	0.91902	0.917451	0.921373	0.917647	0.914706	0.918431	0.919608	0.917843	0.915098	0.918
2	0.919412	0.917451	0.91902	0.91902	0.915882	0.912745	0.920392	0.918039	0.922353	0.913137	0.9177451
3	0.916275	0.917647	0.914314	0.919412	0.913137	0.917843	0.915686	0.917843	0.918824	0.917255	0.9168235
4	0.913922	0.917255	0.92	0.916275	0.916471	0.91902	0.914706	0.917059	0.912549	0.919608	0.9166863
5	0.913333	0.91902	0.916471	0.918431	0.917451	0.916078	0.920196	0.916863	0.916667	0.911765	0.9166275
6	0.916078	0.916667	0.915294	0.917059	0.912353	0.913922	0.916078	0.916863	0.916275	0.917451	0.9158039
7	0.915294	0.910392	0.918235	0.913529	0.915294	0.914314	0.920784	0.916275	0.918235	0.918039	0.9160392
8	0.922549	0.918235	0.921765	0.913922	0.914902	0.916275	0.918039	0.911569	0.920784	0.915686	0.9173725
9	0.922353	0.920392	0.92	0.914902	0.910588	0.918039	0.918235	0.913333	0.916667	0.914314	0.9168824
10	0.918627	0.914706	0.916275	0.918627	0.916863	0.918824	0.918235	0.916471	0.918039	0.917843	0.917451

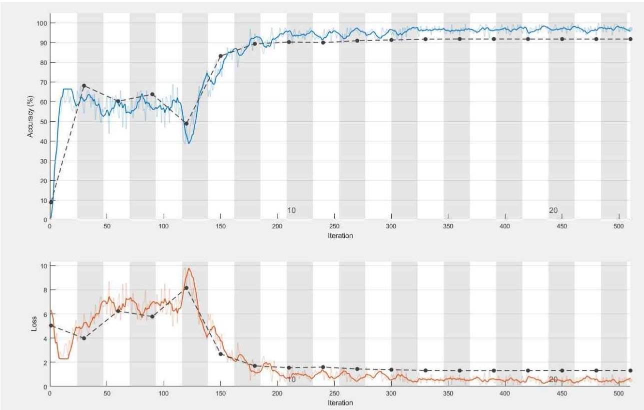
(b) Training time

k \ #	1	2	3	4	5	6	7	8	9	10	Average
1	9.112112	8.780905	9.203149	8.793945	9.299647	8.766104	8.538836	8.805861	8.943518	8.435374	8.8679451
2	9.408318	9.195352	9.933948	10.06871	9.725592	11.09524	9.803016	9.185193	9.346909	9.243009	9.7005287
3	10.08126	10.20141	10.28818	9.583059	11.24963	10.53073	9.666482	10.03054	9.777324	10.69334	10.210196
4	10.67842	10.33728	10.42727	10.11791	11.10267	10.4377	10.74463	9.966559	10.26459	10.43198	10.4509
5	10.21453	10.74307	10.44612	11.07926	10.61474	10.79631	9.849193	10.07195	10.06806	10.50908	10.439231
6	10.59755	10.62607	10.48982	10.91005	10.15259	10.53713	10.38536	10.46862	10.87574	10.11982	10.516274
7	11.5008	10.56858	11.38086	10.87813	11.59874	10.72806	11.49977	10.8405	11.58501	11.61597	11.219643
8	11.89303	11.87131	12.05618	11.25029	10.68471	10.90669	11.1413	10.80616	10.85951	10.85006	11.231924
9	11.42857	11.26142	11.56518	11.74766	11.92416	11.68333	11.2173	11.244	11.28108	11.86943	11.522215
10	11.91704	11.65416	11.7561	12.079	12.42428	12.20164	12.06104	12.10413	11.94613	12.38452	12.052806

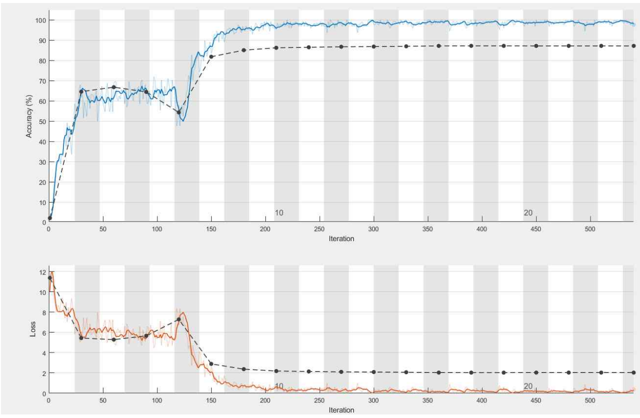
(c) Test time

k \ #	1	2	3	4	5	6	7	8	9	10	Average
1	0.208085	0.158223	0.194821	0.156955	0.14279	0.143225	0.136025	0.13322	0.172064	0.137443	0.1582852
2	0.191423	0.217647	0.267479	0.277543	0.223952	0.227364	0.221728	0.198737	0.298119	0.200733	0.2324725
3	0.332046	0.261039	0.376069	0.26119	0.258344	0.34088	0.354954	0.383225	0.209196	0.208984	0.2985924
4	0.254691	0.26086	0.613878	0.259271	0.255166	0.26148	0.268422	0.263093	0.253042	0.260907	0.2950809
5	0.338577	0.327721	0.288348	0.294929	0.288862	0.286086	0.27303	0.355802	0.284779	0.287293	0.3025428
6	0.305512	0.308272	0.317823	0.314494	0.317997	0.319497	0.317364	0.316755	0.31953	0.317288	0.3154532
7	0.373383	0.343199	0.347526	0.351437	0.340855	0.35217	0.339532	0.342879	0.348464	0.346069	0.3485513
8	0.447192	0.3718	0.371874	0.37965	0.378065	0.371697	0.370566	0.380605	0.372871	0.383328	0.3827648
9	0.413125	0.412776	0.417107	0.417829	0.41182	0.415139	0.41512	0.41828	0.427408	0.424211	0.4172816
10	0.434065	0.437885	0.435506	0.435838	0.433948	0.437413	0.429472	0.438744	0.433724	0.435548	0.4352142

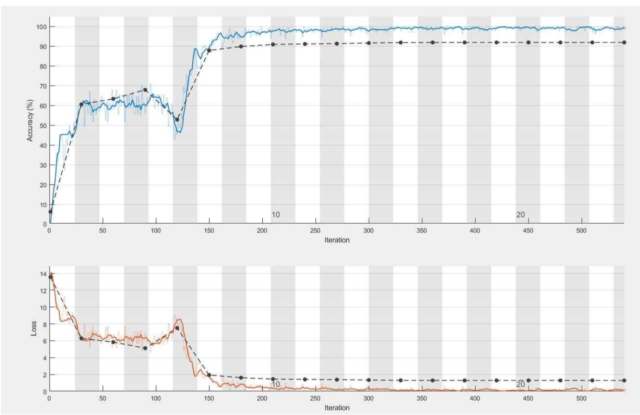
by comparing the efficiency is difficult without execution. We compare the feature reduction methods by implementing SFCNN with domain value extractions and conducting multi-class classification on the Caltech101 dataset. The performance of the networks is evaluated with two metrics: network classification accuracy with Eq. (13) and processing time with Eq. (14). The experimental result is shown in Table 1. From the experimental result, we conclude that the number of extracted features k barely affects the network classification performance. As stated and expected in Section 3, the accuracy of the network using local max extraction is higher than that using the cluster max method and the feature reduction time of the network using local max extraction is longer than that



(a) Accuracy/loss graph of SFCNN

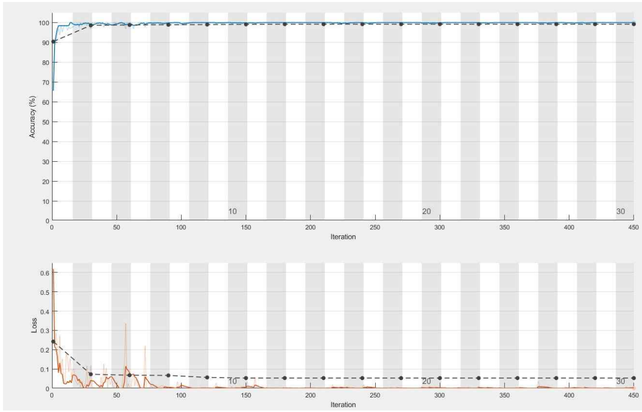


(b) Accuracy/loss graph of Alexnet

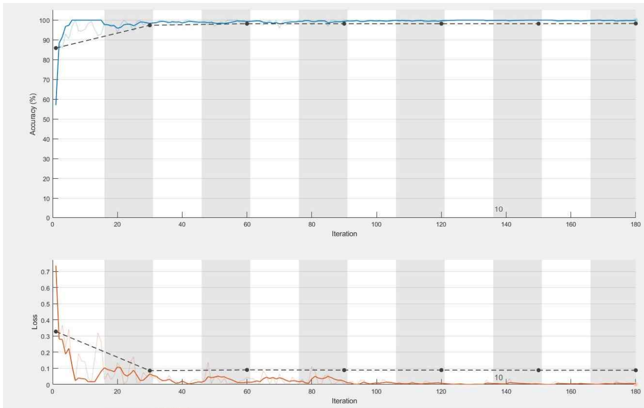


(c) Accuracy/loss graph of VGG16

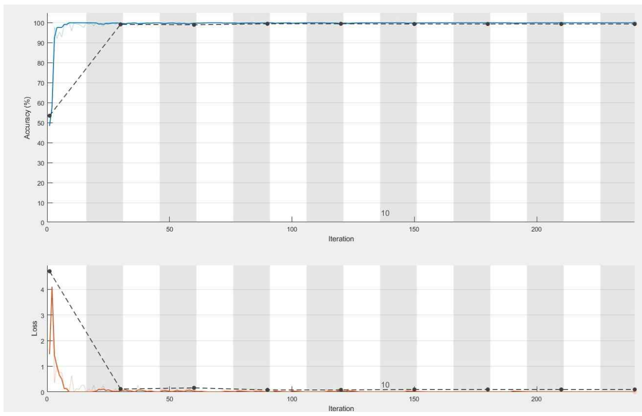
Fig. 6. Accuracy and Loss graph of the training and test of Multi-class Classification. The solid line represents the training accuracy/loss and the dotted line represents the test accuracy/loss respectively



(a) Accuracy/loss graph of SFCNN



(b) Accuracy/loss graph of Alexnet



(c) Accuracy/loss graph of VGG16

Fig. 7. Accuracy and Loss graph of the training and test of Vehicle Detection. The solid line represents the training accuracy/loss and the dotted line represents the test accuracy/loss respectively

using the cluster max. In detail, the average accuracy of the network using local max extraction is 0.9242, which is 1.01 times better than the value using the cluster max method. The feature reduction time for the network using cluster max extraction is 15.3 times faster than that using the local max. Although classification accuracy is a powerful metric for object classifiers, we chose the cluster max method for SFCNN since this paper is aimed at implementing a fast and less-computational network rather than an accurate network. Additionally, the speedy results shown as the experimental results in this section fit the goal of the paper. From this point forward, our proposed SFCNN uses cluster max extraction to produce a sparse feature map for feature reduction.

4.3 Decision of parameter k

In Section 4.4, our proposed network is evaluated with the Caltech101 dataset [18], which is a set of image data containing 102 classes including the background as shown in Fig. 8, to demonstrate the performance in general multi-class classification. For demonstration, we conduct a classification experiment on SFCNN and conventional CNNs and compare the experimental results with three metrics. However, to produce a sparse feature map as shown in Eq. (10), k, the size of the sparse feature map, should be defined before the network structure is declared. Before we conduct a comparative experiment with conventional CNNs, we need to determine the value of k to create a powerful SFCNN. To find the optimal k, we train and test the SFCNN with a multi-class classification dataset in the range of $k = 1, \cdots, 10$. The network performance is evaluated with three metrics from Eqs. (13), (15), and (16). All experiments are conducted 10 times iteratively on each parameter value. The full experimental results are presented in Table 2.

As shown in the result, k =1 draws the best result among the parameters, both in accuracy and proceeding time. We set the parameter k=1 to implement the optimal SFCNN and the proposed SFCNN is evaluated in Section 4.4.

Fig. 8. Caltech 101 dataset

4.4 Result for general classification

Now that we have discussed the network structure, we will proceed with a discussion of the substantive utility for fast networks. We demonstrate the significant performance of SFCNN against the conventional CNNs in general classification with a multi-class dataset, the Caltech 101 dataset. For evaluation, we constitute the training set with 30 random images per category from the training dataset and the test set with 50 random images per category from the test dataset. The number of data in each category of the Caltech 101 dataset is imbalanced and standardization of the dataset is necessary to prevent overfitting. We compare SFCNN to two popular and standard CNN models, Alexnet and the VGG16 Network. As in Chapter 4.3, experimental metrics are chosen as Eqs. (13), (15), and (16).

Each experiment is conducted 10 times iteratively and the results are presented in Fig. 6 with accuracy/loss graph for each training/test process and Table 3 with the average value for the whole experiment. As shown in Table 3, the numerical accuracy value of SFCNN is 1% lower than that of the VGG16 network, and 7% higher than that of Alexnet. Fig. 6 also shows the accuracy flow of proposed SFCNN during the training and test step is similar to conventional CNNs. Additionally, the proposed network is 24 times faster for training and 29 times faster for testing than Alexnet, and 59 times faster for training and 81 times faster for testing than the VGG16 Network. This significant processing time reduction comes from the reduced dimension of $\mathbf{F}_{spr}$. While the size of the full feature map layer $\mathbf{F}_{final}$ is defined as $14 \times 14 \times 512$ in [11], $\mathbf{F}_{spr}$ reduces the layer size with $k \times 512$ using a sparse feature map. When the layer size of $\mathbf{F}_{spr}$ is reduced, the size of W_f is reduced simultaneously since the fully-connected weight has an identical layer size to the connected layers. We can conclude that reduction of the input feature map layer causes a decrease in the network volume. Training a smaller network requires less computation, thus a decreased network volume leads to a shorter processing time.

Consequently, although the accuracy decreases with SFCNN, the experimental result shows powerful performance with regard to processing time with a small loss in classification ability due to the reduced layer size of the sparse feature map, meaning that SFCNN reflects the

Table 3. Results for multi-class classification

	Accuracy	Training Time(s)	Test Time(s)
Alexnet	0.8613	308.989	6.7923
VGG16	0.9254	750.0722	19.3117
SFCNN	0.9184	19.31166	0.2364

Table 4. Results for vehicle detection

	Accuracy	Training Time (s)	Test Time (s)
Alexnet	0.9917	176.8507	0.8018
VGG16	0.9934	423.7619	1.8256
SFCNN	0.9859	4.7564	0.0193

Fig. 9. GTI vehicle image database

network for classification in terms of speed.

4.5 Application to Vehicle detection

In this part, our algorithm is applied to distinguish vehicle from non-vehicle data using the GTI Vehicle Image Database [13], which includes positive and negative vehicle data as shown in Fig. 9. Training of the binary classification network with SFCNN is conducted for application to the vehicle detection model. Fig. 7 shows the training accuracy/loss graph for each training/test process of proposed SFCNN and conventional CNNs.

Classification performance is evaluated with Accuracy, Training time and Test time from Eqs. (13), (15), and (16). We use 1000 images per category as the training dataset and 500 images per category as the test dataset. We compare the experimental results of SFCNN to two CNN models, Alexnet and VGG16. Each experiment is conducted 10 times iteratively and the results are presented in Fig. 7. with accuracy/loss graph and in Table 4 with the average value of the experiments. The accuracy of SFCNN is 0.1% lower than that of the VGG16 network and Alexnet, while accuracy/loss graph shows similar shape in its training/test process with VGG16 network and Alexnet. However, the proposed network is 37 times faster for training and 42 times faster for testing than Alexnet and 88 times faster for training and 94 times faster for testing than the VGG16 Network. The reason for better performance compared to the conventional CNNs is that the faster object classifier causes a reduction in the volume of the network. SFCNN also shows better performance in vehicle classification than general multi-class classification, along with more accurate classification and faster processing. Our proposed SFCNN has the potential to be applied as a vehicle detector in future studies.

5. Conclusion

To employ deep learning networks in real-time applications, the conventional extensive deep networks require calculation improvements. SFCNN is proposed to reduce the network volume and processing time, while maintaining classification performance. In the feature map for conventional CNNs, meaningful values are focused in several regions and zero-value regions occupy a considerable amount of the network. Substituting with a few representative features can reduce the network volume. We proposed two complementary methods to generate a sparse feature map by extracting the maximum value of regions using local maximum extraction and cluster maximum extraction. We compared both proposals through experiments and chose cluster max extraction as the main activation function due to better time performance. From the experimental results, the classification accuracy of SFCNN found no significant differences between that of conventional CNNs with similar training process but the processing time is overwhelmingly better. Application to vehicle classification also results in fast object detection.

Our method still has limits in accuracy. Our future works will attempt to converge the accuracy into conventional CNN by supplementing the maximum extraction function and reducing the processing time simultaneously. Further, to develop a real-time system for vehicle detection, we will implement an on-line training system with the proposed network.

Acknowledgements

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B01016071), and (NRF-2016R1D1A1B 03936281), and also (No. 2017R1D1A1B03031467).

References

[1] B. Zhao, J. Feng, X. Wu and S. Yan, “A survey on deep learning-based fine-grained object classification and semantic segmentation,” *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119-135, April 2017.

[2] C. Stauer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 22, no. 8, pp. 747-757, August 2000.

[3] A. Uçar, Y. Demir and C. Güzeliş, “Object recognition and detection with deep learning for autonomous driving applications,” *Simulation*, vol. 93, no. 9, pp. 759-769, June 2017.

[4] B. Babenko, M. H. Yang, and S. Belongie, “Robust object tracking with online multiple instance learning,” *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 33, no. 8, pp. 1619-1632, August 2011.

[5] A. Giusti, J. Guzzi, D. C. Ciresan, F. L. He, J. P.

Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett*, vol. 1, no. 2, pp. 661-667, July 2016.

[6] D. Geronimo, A. M. Lopez, A. D. Sappa and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 7, pp. 1239-1258, May 2009.

[7] 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.

[8] S. Herath, M. Harandi and F. Porikl, "Going deeper into action recognition: A survey," *Image and Vision Computing*, vol. 60, pp. 4-21, April 2017.

[9] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision* (ICCV), pp. 2980-2988, 2017.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," P*roc. International Conference on Learning Representations*, http://arxiv. org/abs/1409. 1556, 2014.

[12] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145-151, January 1999.

[13] [Online] Available: https://www.gti.ssr.upm.es/data/Vehicle_database. html.

[14] S. Hinterstoisser, V. Lepetit, P. Wohlhart and K. Konolige, "On Pre-Trained Image Features and Synthetic Images for Deep Learning," *arXiv preprint arXiv:1710. 10710*, 2017.

[15] J. Wang, C. Luo, H. Huang, H. Zhao and S. Wang, "Transferring pre-trained deep CNNs for remote scene classification with general features learned from linear PCA network," *Remote Sensing*, vol. 9, no. 3, p. 225, March 2017.

[16] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao and S. Yan, "Hcp: A flexible CNN framework for multi-label image classification," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 38, no. 9, pp. 1901-1907, September 2016.

[17] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 38, no. 1, pp. 142-158, January 2016.

[18] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 39, no. 7, pp. 1476-1481, July 2017.

[19] Y. Ke and R. Sukthanka, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[20] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust Object Recognition with Cortex-Like Mechanisms," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 29, no. 3, pp. 411-426, March 2007

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, December 2015.

[22] X, Chen, X. Lin, "Big Data Deep Learning: Challenges and Perspectives," *IEEE Access*, vol. 2, pp. 514-525, May 2014.

Sung Hee Kim received B. S and M. S. degree in electrical engineering from Korea University, Seoul, Korea in 2016 and 2018 respectively. Her research interests are robust control, mechatronics, machine learning, artificial intelligence and robotics.

Dong Sung Pae received the B. S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2013, where he has been working toward the Ph. D. degree with the School of Electrical Engineering since 2013. His current research interests include computer vision, feature extractor, video stabilization, artificial intelligence, and their application to intelligence vehicle systems.

Tae-Koo Kang received his B. S. in Applied Electrical Engineering, M. S. in visual image processing, and Ph. D. in Electrical Engineering from Korea University, Seoul, Republic of Korea, in 2001, 2004, and 2012 respectively. He was a research professor at Korea University, Seoul, Republic of Korea from 2012 to 2014 and an assistant professor in Information and Telecommunication Engineering, Cheonan, Republic of Korea in 2015 and 2016. He is now an Assistant Professor in Department of Human Intelligence and Robot Engineering, Sangmyung University, Cheonan, Republic of Korea. His research interests include computer vision, robotics, artificial intelligence, and machine learning.

Dong W. Kim received the PhD degree in Electrical Engineering from Korea University, Seoul, Korea in 2007. He was a post-doctoral research scholar at BISC (Berkeley Initiative in Soft Computing), University of California Berkeley in 2008 and AHMCT (Advanced Highway Maintenance and Construction Technology Research Center), University of California Davis in 2009. He has been a professor in the Department of Digital Electronics, INHA Technical College. His research focuses on the intelligent humanoid and vision system, autonomous multi-mobile robot system, robot intelligence based on machine learning. He is associate editor of Mechatronic Systems and Control, and Frontiers in Robotics and AI(Humanoid Robotics specialty section). He received the best paper award at International Conference on Knowledge-based and Intelligent Information and Engineering Systems, and Best Student Paper Competition Finalist Award at IEEE International Conference on SMC in 2003, the Prize from Seoam Scholarship Foundation in 2002.

Myo-Taeg Lim received his B. S. and M. S. degrees in Electrical Engineering from Korea University, Korea in 1985 and 1987, respectively. He also received M. S. and Ph. D. degrees in Electrical Engineering from Rutgers University, NJ, USA in 1990 and 1994, respectively. He was a Senior Research Engineer, Samsung Advanced Institute of Technology and a Professor in the Department of Control and Instrumentation, National Changwon University, Korea. Since 1996, he has been a Professor in the School of Electrical Engineering at Korea University. His research interests include optimal and robust control, vision based motion control, and autonomous mobile robots. He is the author or coauthor of more than 80 journal papers and two books (Optimal Control of Singularly Perturbed Linear Systems and Application: High-Accuracy Techniques, Control Engineering Series, Marcel Dekker, New York, 2001; Optimal Control: Weakly Coupled Systems and Applications, Automation and Control Engineering Series, CRC Press, New York, 2009). Dr. Lim currently serves as an Editor for International Journal of Control, Automation, and Systems. He is a Fellow of the Institute of Control, Robotics and Systems, and a member of the IEEE and Korea Institute of Electrical Engineers.