Neural Networks 87 (2017) 109–121

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection

Jihun Kim, Jonghong Kim, Gil-Jin Jang, Minho Lee *

School of Electronics Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, Republic of Korea

ARTICLE INFO

Article history:
Received 10 April 2016
Received in revised form 11 November 2016
Accepted 2 December 2016
Available online 10 December 2016

Keywords:
Convolutional neural network
Extreme learning machine
Advanced driver assistance system
Lane detection

ABSTRACT

Deep learning has received significant attention recently as a promising solution to many problems in the area of artificial intelligence. Among several deep learning architectures, convolutional neural networks (CNNs) demonstrate superior performance when compared to other machine learning methods in the applications of object detection and recognition. We use a CNN for image enhancement and the detection of driving lanes on motorways. In general, the process of lane detection consists of edge extraction and line detection. A CNN can be used to enhance the input images before lane detection by excluding noise and obstacles that are irrelevant to the edge detection result. However, training conventional CNNs requires considerable computation and a big dataset. Therefore, we suggest a new learning algorithm for CNNs using an extreme learning machine (ELM). The ELM is a fast learning method used to calculate network weights between output and hidden layers in a single iteration and thus, can dramatically reduce learning time while producing accurate results with minimal training data. A conventional ELM can be applied to networks with a single hidden layer; as such, we propose a stacked ELM architecture in the CNN framework. Further, we modify the backpropagation algorithm to find the targets of hidden layers and effectively learn network weights while maintaining performance. Experimental results confirm that the proposed method is effective in reducing learning time and improving performance.

1. Introduction

According to the "European Accident Research and Safety Report 2013", more than 90% of driving accidents are caused by human error (Truck, 2013). Recently, more cars have been equipped with advanced driver assistance systems (ADASs) to assist drivers in recognizing dangerous situations. Among the functions of the ADAS, lane departure warning and lane change assistance are most relevant to these situations (Tapia-Espinoza & Torres-Torriti, 2013).

In general, lane detection consists of preprocessing and detection. A simple and well-known method for detecting lanes is random sample consensus (RANSAC) (Kim, 2008), which can identify straight lines by combining scattered and neighboring points in image scenes. However, RANSAC frequently becomes unreliable as complexity and illumination in road scenes for instance in cases with shadows, occlusions, and curves. To address these difficult cases, convolutional neural networks (CNNs) (LeCun, Bottou, Bengio, & Haffner, 1998) have been implemented to enhance input images and extract regions of interest (ROIs) before performing RANSAC (Kim & Lee, 2014).

A CNN is an effective solution in classification and recognition problems for large datasets, such as ImageNet. In contrast with other learning algorithms, a CNN has characteristics such as a local receptive fields and shared weights. A receptive field exploits the sparse connectivity of neurons to only a local region of an adjacent layer. In shared weights, replicated units create a feature map using shared parameters and increase robustness, which is efficient in lane detection problems that include different road environments.

The integration of CNN and RANSAC leads to an acceptable result with complex road scenes. However, several problems remain in lane detection tasks. The first problem is the limited training data available for complex road scenes. Complex road environments are quite different from normal and highway road cases. Further, substantial time is necessary to create ground truth for the training data. Moreover, a CNN requires additional learning time owing to the large number of parameters it requires. In general, a backpropagation method (Rumelhart, Hinton, & Williams, 1988) is used to optimize the weights between the layers of a CNN, which requires excessive iteration steps to get a desired accuracy for training. In particular, when the number of layer increases, training time increases dramatically. However, most recent CNN

* Corresponding author.
E-mail address: mholee@gmail.com (M. Lee).

http://dx.doi.org/10.1016/j.neunet.2016.12.002
0893-6080/

studies are concerned with improving performance, not training speed. It is common to perform offline training and to speed up training with the support of GPUs. When faced with large datasets including tremendous input dimensions, for example the ILSVRC (Russakovsky et al., 2015) dataset, training can take several days or weeks to achieve the best performance (Simonyan & Zisserman, 2014). Therefore, we consider training speed to be a valuable property in cost effective computing environment without GPU. Thus, this paper applies an extreme learning machine (ELM), because training all the weights in a single layer is accomplished in a single operation, significantly reducing training times. Furthermore, the ELM is not only suitable for large dataset but also it is effective for small training dataset. When we encounter a lack of data, using a CNN is very difficult, because of the amount of data a conventional CNN requires for training. Therefore, this paper modifies the structure of the CNN to exploit the advantages of the ELM. Through this modification, the CNN can be trained with a smaller dataset while maintaining accurate performance in extracting features from complex scenes and reducing training time significantly. Since we focus entirely on such non-trivial situations, we end up applying our method over a small dataset. ELM offers faster convergence speed and better optimization performance for small data and efficiently optimizes a network by using pseudo inverse calculation irrespective of training size. Though not demonstrated in this work, ELM can easily scale well to larger datasets and can produce faster convergence.

ELM (Huang, Zhu, & Siew, 2006) is a new learning method for single-hidden-layer neural networks. It uses random mapping from the input to the hidden layer and a single matrix inversion to determine the optimal weights from the hidden to the output layer. Unlike with backpropagation, there is no iterative adaptation of weight learning; hence, training time can be reduced dramatically. However, it cannot be applied to deep neural networks because it requires a fixed input and target; however, in deep neural networks, the majority of hidden layers are composed of unknown latent variables.

In this paper, we suggest a new learning framework that combines the CNN and ELM. In the proposed method, an additional layer is inserted between the convolutional and subsampling layers. The newly added layer in the last fully connected nodes of the CNN is optimized by conventional ELM algorithms and other additional layers are updated by a modified backpropagation learning rule derived from the original ELM learning algorithm. Exploiting the extremely fast learning speed of the ELM, we can reduce computation time dramatically and obtain superior performance.

In Section 2, we describe the related work in lane detection and how a CNN is applied to an ADAS. We present the details of the proposed combined architecture and the learning algorithm derivations in Section 3. In Section 4, we provide experimental results evaluating the performance of the proposed method on a lane detection task. Finally, we detail our conclusions in Section 5.

2. Related works

2.1. Lane detection

For practical applications, the common approach to lane detection follows these steps: (1) preprocessing, (2) detection, and (3) tracking. Preprocessing typically consists of image smoothing, edge detection, ROI setting, and perspective matching. Image smoothing is used to reduce camera noise; methods include Gaussian filtering (Shihavuddin, Ahmed, Munir, & Ahmed, 2008) and median smoothing (Wen, Yang, Song, & Jia, 2008). Edge detection is required because lanes are represented in different colors including white, yellow, and blue. Sobel (He, Rong, Gong, & Huang, 2010) and Canny (Zhou et al., 2010) edge detectors are used primarily in lane detection. Many papers set ROIs to reduce computation time because real-time application is required for an ADAS. Some papers use perspective analysis to improve detection. Perspective analysis can be carried out using lane slopes and vanishing point (Kreucher, Lakshmanan, & Kluge, 1998); IPM (Inverse Perspective Mapping) is one such method (Bertozzi & Broggi, 1996). Furthermore, IPM is sometimes used to improve curve fitting. In lane detection, two methods are generally used for this purpose. The first is the Hough transform (Assidiq, Khalifa, Islam, & Khan, 2008), which can detect the line with the greatest value in an edge image that includes many lines. Thus, the selected line is assumed to be the lane. RANSAC (Aly, 2008) is a model-fitting algorithm that reduces the effect of outliers in model fitting. For tracking, Kalman (Voisin, Avila, Emile, Begot, & Bardet, 2005) and particle filters (Apostoloff & Zelinsky, 2003) are typically used.

The latest lane detection methods focus on special cases or combine techniques from various fields, instead of increasing performance in simple cases such as a highway with clear lane markers on a bright day. For example, lane detection using Canny edge detection and the Hough transform is popular and has achieved high performance in such simple cases. However, its performance is limited by changing illumination, noise, and curves. Thus, the latest lane detection algorithms have focused on cases of reduced illumination (Son, Yoo, Kim, & Sohn, 2015) and improved curve fitting (Niu, Lu, Xu, Lv, & Zhao, 2015). Moreover, the integration of lane detection and other methods, such as lane detection using GPS signals (Lee, Im, Heo, & Jee, 2015), the combination of camera and Light Detection and Ranging (LiDAR) data (Shin, Shim, & Kweon, 2015), and cooperative map-based data (Kim et al., 2015) are widely used in ADAS technology. However, most of these approaches are computationally extensive or have other limitations. LiDAR requires much more expensive sensors than a single camera, so it is generally used in very complicated applications such as autonomous vehicles. Also, GPS signals are not accurate enough to find lanes, so various error correction techniques with extensive computation are required. In map-based approaches, maps change frequently as a result of construction, weather, and a variety of other factors. Thus, it is necessary to update the map periodically in order to be useful in real-world situations.

2.2. ADAS implementation using CNNs

Deep learning has been adopted in most recent machine learning and artificial intelligence problems to improve performance. There have been attempts to apply deep learning to ADAS applications because many ADAS implementations require detecting objects in the road, such as vehicles, lanes, traffic signs, lane marks, and traffic signals from camera image inputs. Among many deep neural network architectures, the CNN is more commonly used than any other deep learning methods such as deep belief networks (DBN) and deep auto-encoder. In ADAS applications, the CNN has been applied to detecting traffic signs (Sermanet & LeCun, 2011) and vehicles (Zheng, Wang, & Zeng, 2015), in road mark recognition (Chen, Chen, Shi, & Huang, 2015), and in road estimation (Brust, Sickert, Simon, Rodner, & Denzler, 2015). The aforementioned cases are related to detection and recognition tasks. However, the goal of the proposed method is input image enhancement, which can be applied to many other applications. In the resulting images, unnecessary, impeditive information in road images (fences, other vehicles, shadows, and myriad other obstacles) is suppressed. In addition, the lane shapes become clearer for subsequent detection. In our framework, we use the RANSAC algorithm to detect lanes in the enhanced images.

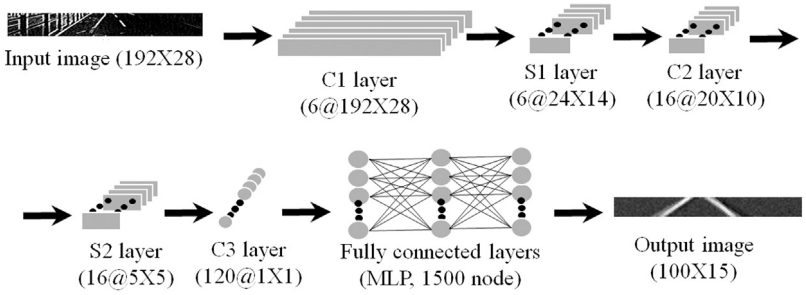

Fig. 1. CNN structure.

2.3. CNN applications with ELM

There are several previous works that attempt to apply an ELM to CNN structure. One such study focuses on traffic sign recognition (Zeng, Xu, Fang, & Zhao, 2015), using a CNN as a feature extractor with average image subtraction as preprocessing and then applying an ELM as the classifier after training the CNN. Additionally, CNNs have been used for feature extraction and applied to the public datasets, MNIST (Guo & Ding, 2015), CIFAR-10, and NORB (Lee & Park, 2015). One study tried to apply a CNN extractor and ELM classifier to age prediction from face images (Gurpinar, Kaya, Dibeklioglu, & Salah, 2016). All of these papers applied ELM as the classifier of the CNN structure and achieved better results compared to fully connected network classifiers. From this investigation, we can easily determine that ELMs are useful when applied to CNN structure. However, these approaches only applied ELMs in the final classification layer because ELMs always require target information for training, and the only target we generally have is the desired output. Our proposed method addresses this point. Unlike previous works, we include ELM layers not only as output, but also as hidden layers. To solve the target problem of hidden layers, we used backpropagation and estimated target values from hidden error. There have also been ELM-related approaches that develop an ELM with autoencoder structure. One such example is an autoencoder stacking approach with label pixels (Tissera & McDonnell, 2016). The authors applied an ELM to an autoencoder with the same input and output except for the label pixels included in the output image. Another approach uses ELM autoencoder structure for multi-modal feature extraction (Wei, Liu, Yan, & Sun, 2016). The ELM autoencoder is used for pre-training multi-modal features; by using a shared layer, classification is carried out with an ELM. There is another ELM approach that changes CNN pre-training with an ELM (Yoo & Oh, 2016). The input, output, and kernel shape are transformed in order to train with an ELM algorithm and pre-train the CNN kernel with autoencoder structure. These ELM approaches are very much related to autoencoder structure. However, our proposed model is based on CNN structure without pre-training.

3. Proposed method

3.1. Extreme learning convolutional neural networks (ELCNNs)

A CNN is constructed by alternately stacking convolutional layers (simple cells) and subsampling layers (complex cells). The convolutional layers of the CNN act as a local receptive field; the subsampling layers mimic complex cells' functions in data abstraction by extracting maximum or average responses from the lower convolutional layers. After passing through a number of convolutional and subsampling layer pairs, outputs are passed to a fully connected multilayer perceptron (MLP) to finally classify the patterns or infer the desired values. In classification problems, the target signals at the output layer are usually binary-encoded class labels or real values. The errors to the target are propagated backward to the lower layers of the network to minimize the error in the CNN. In this paper, our network consists of two subsampling layers, three convolutional layers, and two fully connected layers, as illustrated in Fig. 1.

In the case of the conventional backpropagation method, weights are optimized to minimize the error between the network output and the given targets using optimization methods such as stochastic gradient descent learning (Robbins & Monro, 1951), Quasi-Newton methods (Shanno, 1970), and the weight conjugate gradient method (Møller, 1993). Most optimization methods are based on gradient descent learning, which gradually adjusts the weights in the opposite direction of the gradient; these may easily fall into local minima because the learning is done locally.

ELM is a fast learning method for artificial neural networks with single hidden layer; it replaces iterative backpropagation learning with a single-step matrix inversion. The weight matrix from the input nodes to the hidden nodes is obtained by random initialization, and the weight matrix between the hidden and output layers is updated by a closed-form equation.

Fig. 2 presents the structure of the network architecture where ELM learning can be applied. The conventional ELM can only be applied to a single hidden layer perceptron structure. The input is assumed to be a vector, expressed $x = [x_1, x_2, \ldots, x_N]$ and the target is another vector, $t = [t_1, t_2, \ldots, t_N]$ whose elements are binary or real for the classification or the inference problems, respectively. The weight vector and scalar bias from the input nodes to the hidden node, denoted by $\mathbf{w}_i$ and b_i, are set as random values. The output from the hidden layer is computed by

$$h_{ij} = g\left(\mathbf{w}_i \cdot \mathbf{x}_j + b_i\right) = g\left(\sum_{k=1}^{K} w_{ik} x_k + b_i\right), \tag{1}$$

where w_i and x_k are the kth elements of weight vector $\mathbf{w}_i$ and input vector $\mathbf{x}$, respectively; and K is the dimension of the input vector. For multiple training samples, it is more precise to use matrix notation. We denote a set of N input samples as an $N \times K$ matrix,

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}. \tag{2}$$

Then, the output at the hidden nodes is written as the following matrix equation:

$$\begin{aligned} \mathbf{H}(\mathbf{W}, \mathbf{b}, \mathbf{X}) &= \begin{bmatrix} h_{11} & \cdots & h_{M1} \\ \vdots & \ddots & \vdots \\ h_{1N} & \cdots & h_{MN} \end{bmatrix} \\ &= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{bmatrix}, \end{aligned} \tag{3}$$

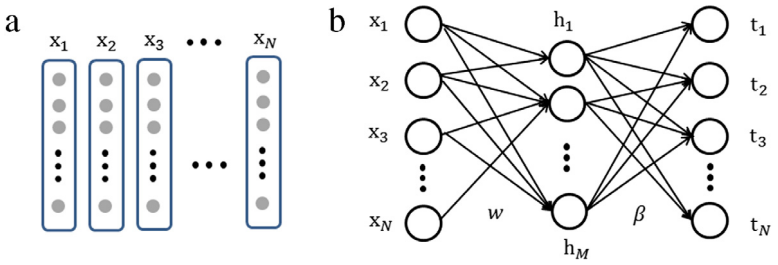

Fig. 2. ELM structure: (a) Input in vector form, (b) an ELM in a three-layer network.

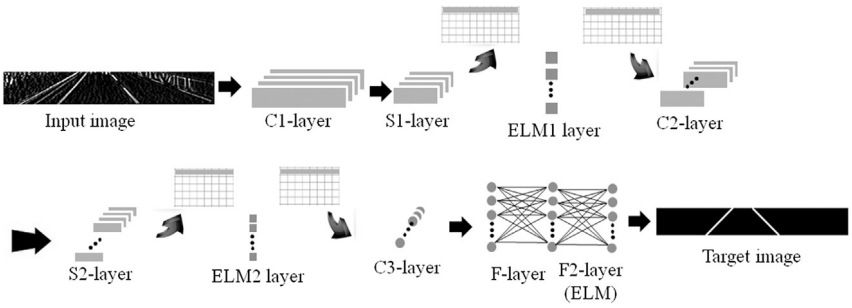

Fig. 3. Structure of the ELCNN.

where M is the dimension of the output vector at the hidden layer. The individual output at the topmost output layer can also be represented by multiplying the weight vector and is assumed to approximate the target as follows:

$$\sum_{i=1}^{M} \beta_{ji} g\left(\mathbf{w}_i \cdot \mathbf{x}_j + b_i\right) = t_{jn}, \quad j = 1, \ldots, N, \tag{4}$$

where β_{ji} is a weight from the hidden node i to the output node j.

The above equation can be expressed by the following simple matrix formula:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{5}$$

where $\mathbf{T}$ is a matrix for a set of N target vectors. If $\mathbf{H}$ is nonsingular, the optimal weight matrix $\boldsymbol{\beta}$ can be found with the following pseudo-inversion operation:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}. \tag{6}$$

The computation time required for training CNNs increases proportionally with the number of layers it contains, because the standard backpropagation algorithm is an iterative adaptation (He & Sun, 2015). Hence, significantly more computation is needed when the network weights are not properly initialized. The advantage of the ELM is that learning the weights between connected layer pairs is mathematically well-defined by a deterministic pseudo-inversion; hence, learning is achieved with considerably fewer computations compared to standard backpropagation learning. The overall procedure of the proposed method is illustrated in Fig. 3. In the proposed CNN structure, there are two S-layers and one fully connected layer. The network connection between one S-layer and the higher level C-layer is replaced with an ELM with a single hidden layer, resulting in an increase in the number of layers. The fully connected layer has the same structure as a conventional CNN; however, we use the ELM algorithm for training.

Although ELM learning is fast and efficient, the output target signal must exist in order to obtain the weight matrix from the target values using a pseudoinverse operation. The output of the fully connected layer is the target signal; hence, we can use Eq. (6) without ambiguity. However, because in backpropagation learning the output error is propagated through chained multiplication of multi-level derivatives, no exact target value is available in the mid-level layers. Similar to backpropagation, we borrow values of the next C-layer as target signals. However, because we are not directly addressing the actual target, the target should be found in iterative manner and the ELM should be updated for each iteration. To begin, the ELM weights are randomly initialized. The input to $ELMi$ $(i = 1, 2)$ is $\boldsymbol{K_i}$ which is a vector of the outputs of the current S-layer, denoted by $y_{s(i)}$. Thus, the input to the next, $c(i+1)$-layer is defined as

$$x_{c(i+1)} = g_{mat}\left(g_{vec}\left(y_{s(i)}\right) \cdot w_{ELM(i)}\right), \tag{7}$$

where g_{vec} is a function that converts a two-dimensional (2D) image to a one-dimensional vector and g_{mat} is a reshaping function from a vector to a 2D matrix. To define the target value, we use backpropagation error, as illustrated in Fig. 4.

For the generated output y given the input vector x, we can derive a cost function by

$$J = \frac{1}{2}\left|\left|T - y\right|\right|^2. \tag{8}$$

Thus, the F-layer error signal is set by

$$\delta_f = \frac{\partial J}{\partial w_f} = \left(W^T \delta_{f+1}\right) f'\left(w_f\right), \tag{9}$$

where W^T is a weight vector at the corresponding layer and $f'(\cdot)$ is the derivative of the activation function. The backpropagation formula for the fully connected F-layer is the same as that of the conventional MLP and δ_{f1} is used to calculate backpropagation error at the $C3$-layer. In C-layer cases, we must consider a convolution operation and weights of kernels. Thus, the error of the C-layer can be defined by

$$\begin{aligned}\delta_{c(i)} &= \frac{\partial J}{\partial x_{c(i)}} = \left(\sum_{j=1} \delta_{y_{c(i)}} w_j\right) f'(w) \\ &= \left(\delta_{y_{c(i)}} * filp\left(w\right)\right) f'(w),\end{aligned} \tag{10}$$

where x_{ci} is the input map of the C_i^{th} layer and y_{ci} is the result of the convolution $(f\left(x_{c(i)} * w\right) = y_{c(i)})$ with kernel weight w which is randomly initialized fixed value. The backpropagation procedure is

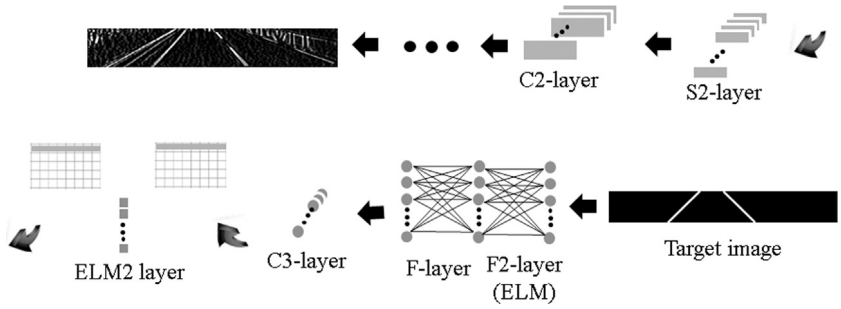

Fig. 4. ELCNN structure.

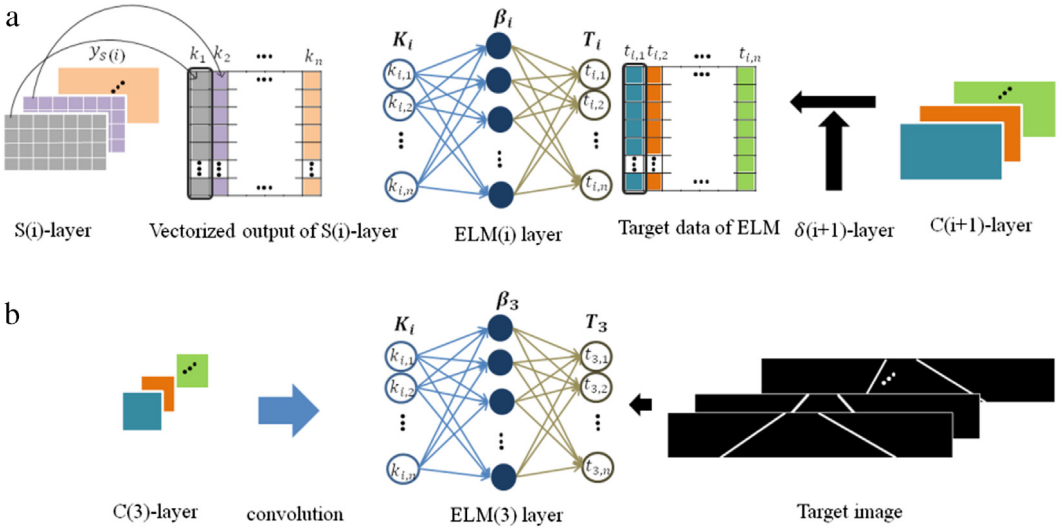

Fig. 5. Operation of ELM: (a) ELM1 and ELM2 cases, (b) ELM3 case.

derived by a convolution with flipped weights. Thus, the derivative of kernels is given by

$$\frac{\partial J}{\partial w_c} = \delta_{y_{c(i)}} * x_{c(i)}. \tag{11}$$

Backpropagation error calculation in the C and F layers is the same as that of conventional CNNs. However, the backpropagation at the S-layer is different because of the ELM-layer located between the S-layer and C-layer. The weights in the ELM layers are not updated with backpropagation learning; rather, other operations such as inner product and vectorization are added to the training procedure in the S-layers:

$$\delta_{s(i)} = \text{upsample}\left(g_{mat}\left(g_{vec}\left(\delta_{c(i)}\right) \cdot w^{-1}_{ELM(i)}\right)\right), \tag{12}$$

where the function *upsample* is an inversion of pooling.

After backpropagation, we use the error of the convolutional layers as target values for the ELM. Fig. 5(a) illustrates the ELM update. The ELM layers are updated by combining forward and backward information. To apply the ELM learning algorithm, the target at the output must be defined. However, because a target does not exist in the middle layers, we approximate the target at layer i as

$$\mathbf{T}_{ELMi} = x_{c(i+1)} - a\delta_{c(i+1)} + b, \tag{13}$$

where $x_{c(i+1)}$ is the value of the next C-layer in the forward process, $\delta_{c(i+1)}$ is the error of the next C-layer in backpropagation, a is the weight factor for the desired output, and b is a bias. According to the delta rule of backpropagation, the delta is usually proportional to the amount of the error. In conventional backpropagation, the delta is used to find the hidden target of each layer based on the stochastic gradient descent method. Training conventional CNN through backpropagation requires many iterations to achieve the desired performance. However, ELM can be trained with pseudoinverse calculation in only a few epochs. For the training of ELM, a target value is required, so in our proposed model, we need to estimate the hidden targets of each ELM. Our proposed method estimates the hidden target by Eq. (13) based on the backpropagation error. Because the backpropagation error is proportional to the error between the output of ELM and the hidden target of ELM, we can estimate the hidden target. After hidden target estimation, ELM can be trained with pseudoinverse calculation. In our proposed model, because of the local minima problem, backpropagation error may not be enough to find the hidden target. Therefore, we search the hidden target in an iterative manner. For convergence, unlike conventional CNN, our proposed model requires only a few iterations by virtue of the advantages of ELM learning. The forward equation of ELM at the ith layer is thus given by

$$\boldsymbol{K}_i\boldsymbol{\beta}_i = \boldsymbol{T}_i. \tag{14}$$

If an inverse operation exists in Eq. (14), an optimal $\boldsymbol{\beta}_i$ can be calculated. Thus, we use a pseudoinverse matrix whose weight is defined as

$$\boldsymbol{\beta}_i = \boldsymbol{K}_i^{\dagger}\boldsymbol{T}_i, \tag{15}$$

where $\dagger$ is the pseudoinverse operator. Fig. 5(b) illustrates ELM3, which is identical to the conventional ELM because ELM3 is applied to a fully connected output layer. The input of ELM3 is $\boldsymbol{x_f}$, obtained by the output of the $C3$-layer. T is a vectorized target image that is used as the target of ELM3 and H is the output of the $F2$-layer. $\boldsymbol{\beta}_3$

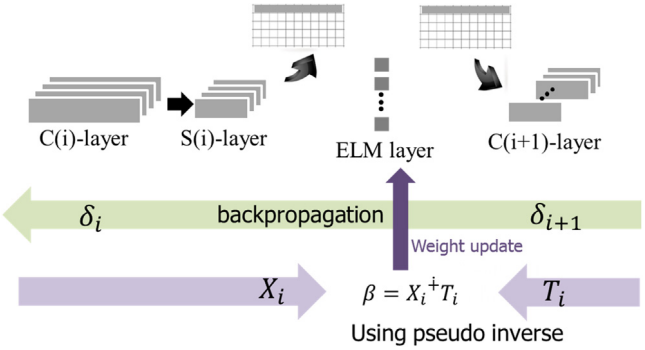

Fig. 6. Structure of the ELM weight updates.

can be further calculated by

$$\boldsymbol{\beta}_i = \boldsymbol{K}_i^{+}\boldsymbol{T}_i. \tag{16}$$

Thus, we can summarize the entire ELCNN process, which is illustrated in Fig. 6:

1. Compute forward pass.
2. Compute error of convolutional layers using backpropagation.
3. Calculate target of ELM1 with respect to the backpropagated error.
4. Calculate ELM1 weights to minimize the forward pass error.
5. Calculate target of ELM2 with respect to the backpropagated error
6. Calculate ELM2 weights to minimize the forward pass error.
7. Repeat 2–6.
8. Calculate ELM3.
9. Repeat 1–8 if the error is smaller than a given threshold.

3.2. Lane detection using a CNN with ELM

3.2.1. Lane detection using RANSAC

Before detecting a lane, several preprocessing methods are applied to obtain clearer lane images. In this paper, smoothing and edge detection are used. Smoothing can reduce noise in the camera image; we use a smoothed image that is obtained from a 5×5 Gaussian smoothing function before edge detection. This step reduces environmental noise and produces more reliable scene information.

A lane is typically marked as a white line on a gray road. Hence, an easy method for detecting lane is to use edge information. For robust lane detection, the area surrounding the lane must be suppressed. Hence, we use a hat-shaped kernel to strengthen lane information while suppressing the surrounding area during edge detection. Fig. 7(a) shows the hat-shaped kernel. The edge image is calculated via a convolution of the lane image and this kernel; the performance of this method is superior to several other preprocessing methods. The result of edge detection in several situations is displayed in Fig. 7(b).

After calculating the edges, lane detection is performed using RANSAC. RANSAC is an estimation technique based on the principle of hypothesis generation and verification. Given a model with randomly selected edge points and assuming a line model, the RANSAC algorithm determines the most likely candidate lines. We set ROIs to reduce computational load; we set the candidate arrival and departure points on the upper and lower ROI lines. Then, low-value lines are set as outliers. Thus, an expectation model can be calculated with the remaining high-value lines, and they define the lane. When a road scene is complex, the selected point is not likely to create a lane and the accuracy of the lane detection may decrease.

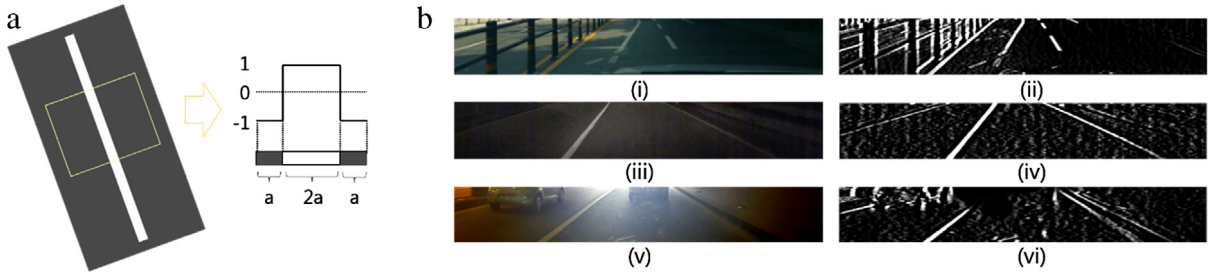

Fig. 7. Kernel for edge detection and examples. (a) The kernel for edge detection. (b) (i) Fence and shadow image, (ii) fence and shadow image using edge detection, (iii) night image, (iv) night image using edge detection, (v) tunnel image with changing light intensity, (vi) tunnel image with changing light intensity using edge detection.

Fig. 8. Example of lane detection method. The blue lines are the candidate arrival and departure lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

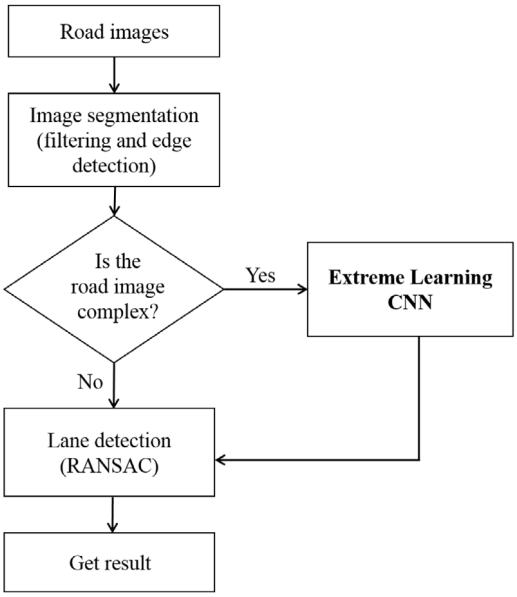

Fig. 9. Flow chart of lane detection system.

We also set the candidate arrival and departure points on the upper and lower ROI line because a lane typically resembles a vertical shape from the inside of the car. Fig. 8 presents an example ROI, candidate arrival and departure points, and the detected lane.

3.2.2. Enhanced lane detection using an ELCNN

A flow chart outlining the proposed method is presented in Fig. 9. First, road images are captured by a camera while driving. Image segmentation is performed to reduce environmental noise and sensor distortion. In general, lanes are painted in different colors compared with the road. If the road image changes dynamically owing to weather conditions, brightness, or obstacles, the captured video recordings may contain considerable noise, which decreases lane detection accuracy. To alleviate the effects of environmental changes before using RANSAC to detect lanes, the CNN is applied to enhance road images by emphasizing lines and suppressing noisy fragments. ELM algorithms are used to efficiently learn the network weights. Lanes can be detected in the enhanced images output by the ELCNN using RANSAC.

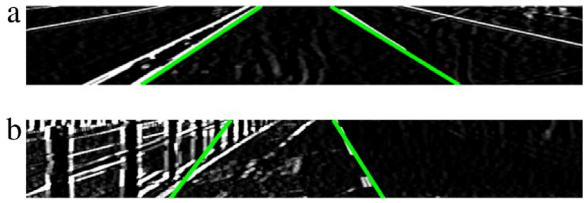

Fig. 11. Results of lane detection using RANSAC: (a) well-detected lane in a simple image, (b) incorrect lane in a complex image.

Table 1
Number of training and test images of KNU dataset and Caltech dataset for different complex lane images extracted from video clips.

Dataset	Case	Number of images	
		Train	Test
KNU dataset	Case 1	200	115
	Case 2	105	41
	Case 3	136	34
Caltech dataset	Case 1	68	16
	Case 2	42	10
	Case 3	81	20

Lane detection accuracy using RANSAC is highly dependent on road conditions. There are three cases in which RANSAC may fail to detect lanes: (1) when more than two lines are detected, (2) when the difference in lane locations between frames is excessively large, and (3) when the difference in the positions of a vanishing points detected using the left and right lines is too distant. Fig. 10 displays examples of the aforementioned three cases. In all three cases, we use the proposed ELCNN before the RANSAC algorithm to emphasize lane shapes and suppress the surrounding noise.

4. Experimental results

4.1. Experimental setup

Our proposed model extends previous work in lane detection (Kim & Lee, 2014) and is based on LeNet (LeCun et al., 1998).

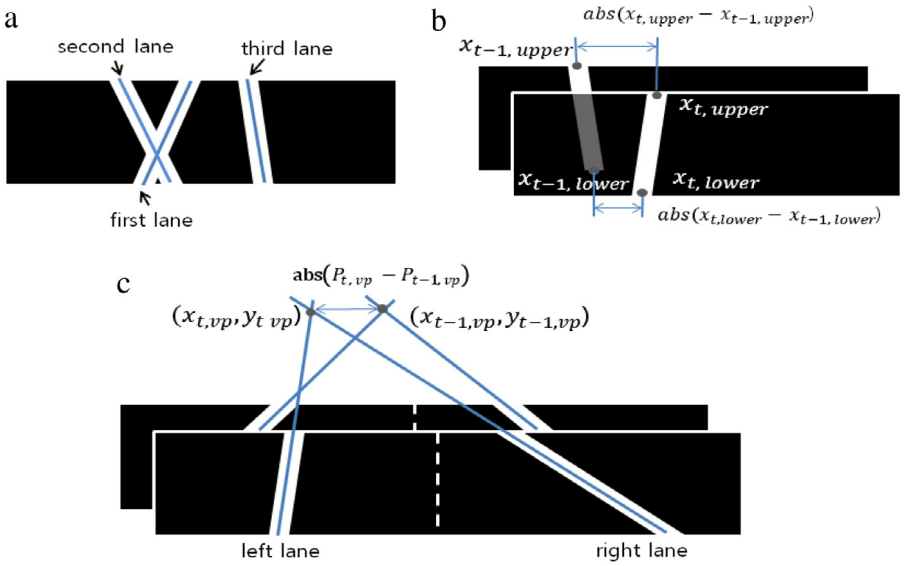

Fig. 10. Example of three cases: (a) there are three lane lines, (b) position of the lane, where $x_{t,upper}, x_{t-1}$, *upper*, and $x_{t,lower}, x_{t-1}$, *lower* in the x-axis direction are represented upper and lower points of lane in t and $t-1$ frame and T_{upper} and T_{lower} are threshold values at upper and lower positions of lane, respectively, (c) the vanishing point of the lane, where $P_{t,vp}$ is the vanishing point at $(x_{t,vp}, y_{t,vp})$ in t frame and $P_{t-1,vp}$ is the vanishing point at $(x_{t-1,vp}, y_{t-1,vp})$ in $t-1$ frame. T_{vp} is the threshold value of vanishing point.

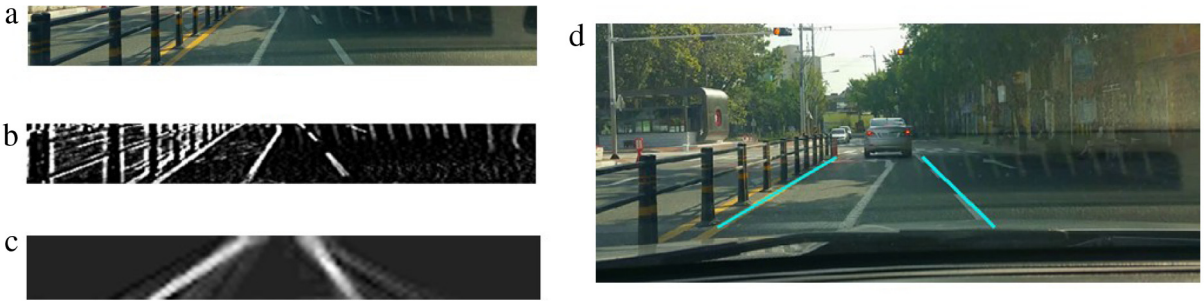

Fig. 12. Simulation results: (a) original image, (b) edge image, (c) ELCNN output image, (d) RANSAC result after ELCNN.

Table 2
Performance evaluation in three conditions in two datasets comparing an ELM, a CNN with an ELM classifier, a conventional CNN, and the ELCNN.

Dataset	Case	Method (%)			
		ELM	CNN	CNN with ELM classifier	ELCNN
KNU database	Case 1	59.6	94.8 (±0.39)	96.1 (±0.30)	96.2 (±0.44)
	Case 2	85.4	93.3 (±0.63)	95.0 (±1.20)	95.9 (±1.44)
	Case 3	52.9	92.3 (±1.16)	94.8 (±1.40)	93.0 (±2.78)
Caltech dataset	Case 1	93.8	97.5 (±2.12)	97.5 (±2.32)	98.1 (±1.53)
	Case 2	70.0	95.0 (±0.00)	100.0 (±0.00)	100.0 (±0.00)
	Case 3	72.5	90.0 (±1.44)	95.5 (±2.45)	96.5 (±1.22)

Creating several kinds of lane data with corresponding ground truth takes substantial time and it is difficult to find varied situations because load state is mostly straightforward. Therefore, we suffer from lack of training data. With a few data, recent large CNN models are unsuitable, since they must be trained on large datasets, such as the ILSVRC dataset (Russakovsky et al., 2015), which consists of 1.2 million images. Unless there is a lack of training data for the scale of the model, we may suffer from overfitting, so we have chosen the baseline CNN model as a smaller model that provides the best performance, as shown in Table 3. As shown in Fig. 1, the baseline CNN model consists of 3 convolutional layers and 2 subsampling layers; our proposed ELCNN model includes the addition of 2 ELM layers and one ELM classification output layer. The size of each convolutional kernel is 5×5 and the convolution performs with a stride of 1. The number of kernels is 6 for first layer, 20 for second, and 50 for third. Since we are considering lane images, the shape of the input images is a wide rectangle. Therefore, for subsampling, we also use a wide subregion with size 8×2 for the first subsampling layer and 4×2 for the second. The ELM layers are square matrices, in order to make the shape of the ELM output the same as its input. Our model resizes all input images 196×32. After the first convolution and subsampling, the size of the input is 24×14. To maintain the same size of ELM output and input, ELM1 is 336×336 for a 24×14 input. Similarly, ELM2 is 25×25 for a 5×5 input. The number of ELM weight matrices is the same as the number of input feature maps. The size of output layer ELM3 is 200×1500, which is the same as a fully connected network. Finally, the reconstructed output image is 100×15. The final output target image is the ground truth lane image, which includes only two ground truth lines. The target approximation parameter a is first set to 0.5. After each iteration, a is reduced by a factor of 0.9 until the end of training. The parameter b is set to zero.

4.2. Evaluation on the KNU database

The result of lane detection using the RANSAC algorithm is presented in Fig. 11. The results are highly dependent on the degree of complexity of the input road scenes. If there is considerable noise, such as fences, walls, or lights reflecting lights on windshield, RANSAC-based lane detection can easily follow false lanes, as indicated in Fig. 11(b).

To improve the accuracy of lane detection in complex road scenes, we apply RANSAC to detect candidate lanes on the output of the ELCNN. The results are displayed in Fig. 12. Because the learning algorithm provides a candidate region for the lanes, noise reduction results are obtained as indicated in Fig. 12(b). We apply RANSAC again to the output of the ELCNN. Finally, we obtain the detected actual lane. Note that the proposed method is robust to changes in road environment, as indicated in Fig. 12(c).

We tested a conventional CNN on complex video clips and compared the results with those of the proposed method. We evaluated the performance of each method using lane detection rate from these video clips. Table 1 shows the number of training and test images for KNU and Caltech datasets. For the given dataset, we randomly select the train and test data. We perform each experiment 10 times with random weight initializations. The performance evaluation method is inspired from Borkar Amol's paper (Borkar, Hayes, & Smith, 2012). Firstly, ground truth on actual lane is defined manually from lane images and the RANSAC algorithm is employed for lane detection. For performance assessment, RANSAC-detected lanes are overlapped with the ground truth and lane detection is taken successful if the overlapping area is more than 50%. This criterion is applied to all of our experiments. The results are presented in Table 2 and Fig. 13. The ELM column is the result of a single hidden layer; CNN is two hidden S-layers, two hidden C-layers, and a fully connected network with a single hidden layer. The CNN was better than the ELM in all cases except for KNU database case 3, owing to the deeper layers; the ELCNN was superior to the conventional CNN. The reason that the ELCNN was better than the conventional CNN is because ELM learning guarantees a global minimum, whereas gradient descent learning may easily fall into local minima; ELCNN results are closer to the global minimum. Also, applying ELM classifier to trained CNN features was better than conventional CNN. Comparing CNN with ELM classifier with ELCNN, ELCNN outperformed the CNN with ELM classifier only except on Case 3 in the KNU database. Furthermore, the training time for the CNN with ELM classifier is much more than that of the proposed ELCNN because of many iterative steps for kernel optimization based on the conventional gradient approach.

Fig. 13. Results from the KNU dataset: (a) road images, (b) lane detection results.

4.3. Evaluation on the Caltech database

To provide a fair comparison of the proposed method with other existing methods, we performed an evaluation on the Caltech database (Aly, 2008), one of the popular databases for lane detection and other vehicular vision applications. The Caltech database was collected by the California Institute of Technology for lane detection. This dataset includes four clips captured on the streets of Pasadena: cordova1 with 250 frames, cordova2 with 406 frames, washington1 with 337 frames, and washington2 with 232 frames. In this experiment, we replaced the lane with non-trivial situations with enhanced lane images. These enhanced lane images were obtained as the output of ELCNN. We performed 5-fold cross validation in this experiment for validation. The final accuracy is calculated after combining the results from RANSAC algorithm over non-trivial and trivial situations for lane detection. Table 3 presents the lane detection accuracy of several existing methods compared with the proposed method. Unfortunately, the frames used in the evaluation are different depending on the condition of individual methods and the selected training data are also different. Selected lane detection results using the proposed method from the selected images are provided in Figs. 13 and 15.

Fig. 14 analyzes the image enhancement results of the CNN and ELCNN with several complex road scenes. The images in the top row (rectangular) are the edge detection results; the images in the second row are grayscale histograms illustrating the distributions of grayscale intensities. The greater the variance of a histogram, the more difficult it is to distinguish lane edges from road regions. When variance is small, the boundary between lane edge distribution and road distribution is clear.

Table 3
Performance comparison with the Caltech dataset.

Video	Caltech dataset (%)			
	cordova1	cordova2	washington1	washington2
CEDHT[a] in YCbCr (Son et al., 2015)	93.6 (569 frame, using illumination data)			
RODT[b] with particle filter (Ruyi et al., 2011)	97.4	91.1	97.8	97.3
IPM with particle filter (Sehestedt, Kodagoda, Alempijevic, & Dissanayake, 2007)	97.8	89.4	92.2	96.2
Caltech lane detection (Aly, 2008)	97.2	96.2	96.7	97.3
Line segment detection based on vanishing point (Benligiray et al., 2012)	98.8	98.3	91.4	95.3
Proposed method	98.7	98.9	98.0	98.6

[a] Canny edge detection and Hough transform.
[b] Row orientation distance transform.

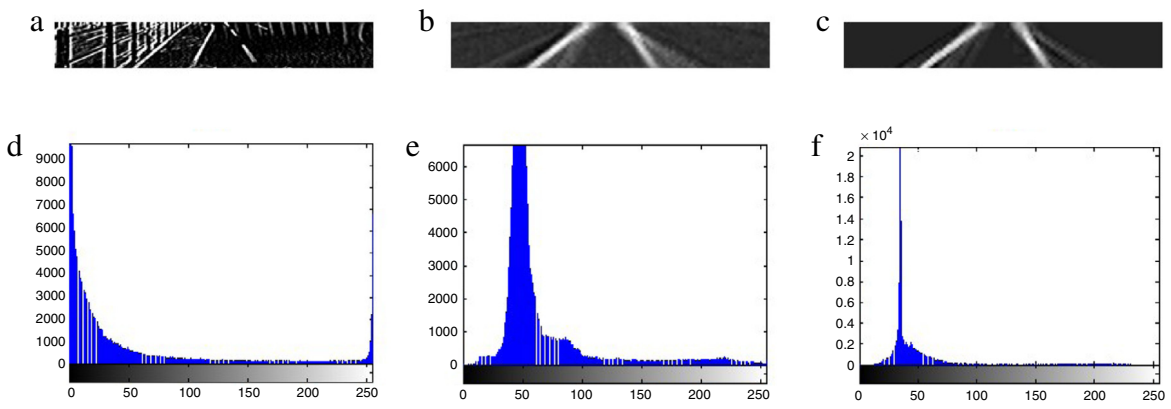

Fig. 14. Results of lane detection, grayscale histogram, and 3D graph with the CNN and ELCNN: (a) edge image, (b) result of CNN, (c) result of ELCNN, (d) histogram of edge image, (e) histogram of CNN result, (f) histogram of ELCNN result.

4.4. Computation time

Another benefit of using ELM is that it can learn weights extremely quickly; hence, applying ELM learning to a CNN can dramatically reduce the required computation time. Table 4 shows a comparison of training computation time. For a fair comparison, we included additional fully connected layers to the CNN model so that it would have the same number of parameters as the ELCNN. The training speed of the ELCNN is at least 800 times faster, suggesting that the proposed ELCNN is particularly effective in applications where rapid training or model adaptation are required. Unlike the conventional CNN, the ELCNN does not require training of convolutional kernels to extract the best feature representations. Rather, the ELCNN optimizes the model by training ELM layers with less effective randomly weighted convolutional kernel features. The effectiveness of randomly selected features has been evaluated in previous work (Saxe et al., 2011). As the authors pointed out, randomly selected features also can generate useful features. Even if those features are not the best features, they can be sufficient for classification.

Fig. 16 indicates convergence curves for each task. Since the ELCNN converges within only 25 epochs whereas the conventional CNN converges after 100,000 epochs, we used a logarithmic scale for the epoch axis. This training graph shows how the ELCNN converges faster than the conventional CNN. As shown in Table 4, even though the ELCNN takes a little bit more computational load per training epoch because of the pseudoinverse calculation, the required number of epochs is significantly less than that needed for conventional backpropagation. The experiment for comparing computation times was performed using an Intel i7-4790 3.6 GHz CPU.

5. Conclusion

In this paper, we propose a novel method, called an extreme learning convolutional neural network, and demonstrated its effectiveness in a lane detection task. To detect lanes efficiently while ignoring noise and other obstacles, we applied the RANSAC algorithm for simple road scenes and employed a machine-learning-based method to more complex road scenes. In previous work, conventional CNNs have been used to enhance the lines defining driving lanes in input images. Although CNNs provide acceptable lane detection performance, learning the complex convolution and deep network architecture requires significant computation. We propose a new learning algorithm that significantly reduces training time compared with conventional CNNs. Using the same dataset, training time was reduced by 1/50–1/200; hence, the applicability to real-time applications is significantly increased. Moreover, because ELM learning ensures a global maximum given the target output, we obtained superior accuracy to that of the conventional CNN on several different databases.

To apply lane detection in ADAS systems, detection should work in real time in embedded platforms, so it is necessary to reduce computational overhead. As future work, we will consider a fully connected ELCNN without convolution layers. Generally, a C-layer can connect local input in a feedforward process and thus can reduce local minima problems. However, the proposed method can use a C-layer to calculate middle-layer targets and not reveal the advantage of the C-layer. Thus, we expect better performance in a fully connected network. Finally, we will also attempt to apply this method to a large-scale dataset. ELM requires matrix inversion, which can increase computation time dramatically for high-dimensional data. Various iterative methods are available for matrix inversion without the high-dimensionality problem, so the proposed method can be improved to deal with large-scale datasets.

Acknowledgments

This research was supported by ICT R&D program of MSIP/IITP. [R7124-16-0004, Development of Intelligent Interaction Technology Based on Context Awareness and Human Intention Understanding] and was supported by the National Research Foundation

Fig. 15. Results from the Caltech dataset: (a) road images, (b) results.

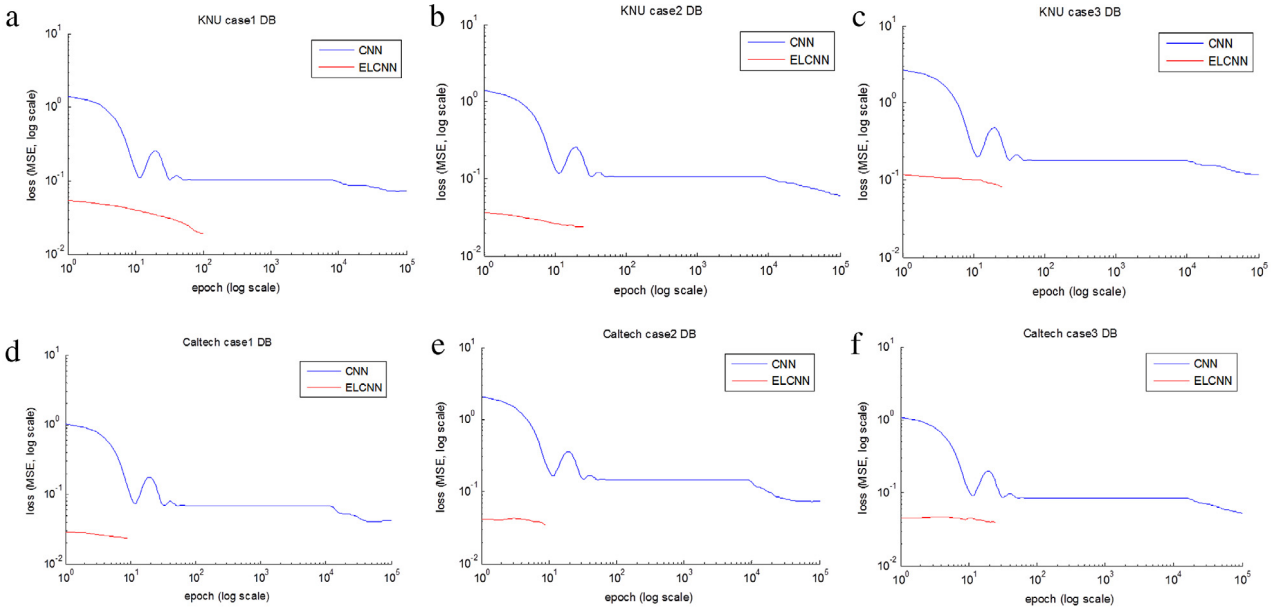

Fig. 16. Loss convergence curve comparison between CNN and ELCNN: (a) KNU DB Case 1 convergence curve. (b) KNU DB Case 2 convergence curve. (c) KNU DB Case 3 convergence curve. (d) Caltech DB Case 1 convergence curve. (e) Caltech DB Case 2 convergence curve. (f) Caltech DB Case 3 convergence curve.

Table 4
Comparison of computation time for training.

		CNN		ELCNN	
		KNU DB	Caltech DB	KNU DB	Caltech DB
Total training CPU time ((CPU time per epoch) × (number of epochs for desired accuracy))	Case 1	>1000 h (36.01 s/epoch × 100,000 epochs)	>300 h (11.82 s/epoch × 100,000 epochs)	74–75 min (44.44 s/epoch × 100 epochs)	2–3 min (14.66 s/epoch × 9 epochs)
	Case 2	>600 h (22.40 s/epoch × 100,000 epochs)	>200 h (7.27 s/epoch × 100,000 epochs)	12–13 min (28.89 s/epoch × 25 epochs)	1–2 min (9.44 s/epoch × 9 epochs)
	Case 3	>700 h (25.96 s/epoch × 100,000 epochs)	>300 h (13.85 s/epoch × 100,000 epochs)	13–14 min (33.37 s/epoch × 25 epochs)	7–8 min (17.86 s/epoch × 25 epochs)
	Total	>2300 h	>800 h	99–102 min	10–13 min

of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2016M3C1B6929647).

References

Aly, M. (2008). Real time detection of lane markers in urban streets. In *Intelligent vehicles symposium, 2008 IEEE* (pp. 7–12). IEEE.

Apostoloff, N., & Zelinsky, A. (2003). Robust vision based lane tracking using multiple cues and particle filtering. In *Intelligent vehicles symposium, 2003. Proceedings. IEEE* (pp. 558–563). IEEE.

Assidiq, A. A., Khalifa, O. O., Islam, R., & Khan, S. (2008). Real time lane detection for autonomous vehicles. In *International conference on computer and communication engineer-ing, 2008*, ICCCE 2008. (pp. 82–88). IEEE.

Benligiray, B., Topal, C., & Akinlar, C. (2012). Video-based lane de-tection using a fast vanishing point estimation method. In *Multimedia, ISM, 2012 IEEE international symposium on IEEE, December* (pp. 348–351).

Bertozzi, M., & Broggi, A. (1996). Realtime lane and obstacle detection on the system. *Proceedings of the IEEE Intelligent Vehicles*, 213–218.

Borkar, A., Hayes, M., & Smith, M. T. (2012). A novel lane detection system with efficient ground truth generation. *IEEE Transactions on Intelligent Transportation Systems*, *13*(1), 365–374.

Brust, C.A., Sickert, S., Simon, M., Rodner, E., & Denzler, J. (2015). Convolutional patch networks with spatial prior for road detection and urban scene under-standing. ArXiv Preprint arXiv:1502.06344.

Chen, T., Chen, Z., Shi, Q., & Huang, X. (2015). Road marking detection and classification using machine learning algorithms. In *Intelligent vehicles symposium, IV, 2015 IEEE, June* (pp. 617–621).

Guo, L., & Ding, S. (2015). A hybrid deep learning CNN-ELM model and its application in handwritten numeral recognition. *Journal of Computational Information Systems*, *11*(7), 2673–2680.

Gurpinar, F., Kaya, H., Dibeklioglu, H., & Salah, A. (2016). Kernel ELM and CNN based facial age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 80–86).

He, J., Rong, H., Gong, J., & Huang, W. (2010). A lane detection method for lane departure warning system. In *2010 International conference on optoelectronics and image pro-cessing, vol. 1*, (ICOIP). (pp. 28–31). IEEE.

He, K., & Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5353–5360).

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*(1), 489–501.

Kim, Z. (2008). Robust lane detection and tracking in challenging scenari-os. *IEEE Transactions on Intelligent Transportation Systems*, *9*(1), 16–26.

Kim, J., & Lee, M. (2014). Robust lane detection based on convolu-tional neural network and random sample consensus. In *Neural information processing* (pp. 454–461). Springer International Publishing.

Kim, S. W., Qin, B., Chong, Z. J., Shen, X., Liu, W., Ang, M. H., et al. (2015). Multivehicle cooperative driving using cooperative perception: design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 663–680.

Kreucher, C., Lakshmanan, S., & Kluge, K. (1998). A driver warning sys-tem based on the LOIS lane detection algorithm. In *Proceedings of IEEE international conference on intelligent vehicles, Stuttgart, Germany, Vol. 1, October* (pp. 17–22).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, B. H., Im, S. H., Heo, M. B., & Jee, G. I. (2015). Curve modeled lane and stop line detection based GPS error estimation filter. In *Intelligent vehicles symposium (IV), 2015 IEEE* (pp. 406–411). IEEE.

Lee, K., & Park, D. C. (2015). Image classification using fast learning convolutional neural networks. *Advanced Science and Technology Letters*, *113*, 50–55. (Art, Culture, Game, Graphics, Broadcasting and Digital Contents 2015).

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, *6*(4), 525–533.

Niu, J., Lu, J., Xu, M., Lv, P., & Zhao, X. (2015). Robust lane detection using two-stage feature extraction with curve fitting. *Pattern Recognition*,.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*, 3.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Ruyi, J., Reinhard, K., Tobi, V., & Shigang, W. (2011). Lane detection and tracking using a new lane model and distance transform. *Machine Vision and Applications*, *22*(4), 721–737.

Saxe, A., Koh, P.W., Chen, Z., Bhand, M., Suresh, B., & Ng, A.Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning, ICML-11* (pp. 1089–1096).

Sehestedt, S., Kodagoda, S., Alempijevic, A., & Dissanayake, G. (2007). Efficient Lane Detection and Tracking in Urban Environments. In *ECMR*.

Sermanet, P., & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Neural networks, IJCNN, The 2011 international joint conference on IEEE, July* (pp. 2809–2813).

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, *24*(111), 647–656.

Shihavuddin, A. S. M., Ahmed, K., Munir, M. S., & Ahmed, K. R. (2008). Road boundary detection by a remote vehicle using radon transform for path map generation of an unknown area. *International Journal of Computer Science and Network Security*, *8*(8), 64–69.

Shin, S., Shim, I., & Kweon, I. S. (2015). Combinatorial approach for lane detection using image and LIDAR reflectance. In *2015 12th International conference on ubiquitous robots and ambient intelligence*, (URAI). (pp. 485–487). IEEE.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. ArXiv Preprint arXiv:1409.1556.

Son, J., Yoo, H., Kim, S., & Sohn, K. (2015). Real-time illumination invariant lane detection for lane departure warning system. *Expert Systems with Applica-tions*, *42*(4), 1816–1824.

Tapia-Espinoza, R., & Torres-Torriti, M. (2013). Robust lane sensing and departure warning under shadows and occlusions. *Sensors*, *13*(3), 3270–3298.

Tissera, M. D., & McDonnell, M. D. (2016). Deep extreme learning machines: supervised autoencoding architecture for classification. *Neurocomputing*, *174*, 42–49.

Truck, V. (2013). European Accident Research and Safety Report 2013.Gothenburg, January.

Voisin, V., Avila, M., Emile, B., Begot, S., & Bardet, J. C. (2005). Road markings detection and tracking using hough transform and kalman filter. In *Advanced concepts for intelligent vision systems* (pp. 76–83). Berlin, Heidelberg: Springer.

Wei, J., Liu, H., Yan, G., & Sun, F. (2016). Multi-modal deep extreme learning machine for robotic grasping recognition. In *Proceedings of ELM-2015 volume 2* (pp. 223–233). Springer International Publishing.

Wen, Q., Yang, Z., Song, Y., & Jia, P. (2008). Road boundary detection in complex urban environment based on low-resolution vision. In *Proceeding of the 11th joint conference on information science, State Key Laboratory on Intelliget Technology and Systems, Tsinghua National Laboratory for Information Sicence and Technology, Department of Computer Science and Technology, Tsinghua University Beijieng, Vol. 100084, December* (pp. 1–7).

Yoo, Y., & Oh, S.Y. (2016). Fast training of convolutional neural network classifiers through extreme learning machines. In *The bi-annual IEEE world congress on computational intelligence, IEEE WCCI, July*.

Zeng, Y., Xu, X., Fang, Y., & Zhao, K. (2015). Traffic sign recognition using extreme learning classifier with deep convolutional features. In *The 2015 international conference on intelligence science and big data engineering, IScIDE 2015, Suzhou, China, June*.

Zheng, J., Wang, Y., & Zeng, W. (2015). CNN based vehicle counting with virtual coil in traffic surveillance video. In *Multimedia big data, BigMM, 2015 IEEE international conference on IEEE, April* (pp. 280–281).

Zhou, S., Jiang, Y., Xi, J., Gong, J., Xiong, G., & Chen, H. (2010). A novel lane detection based on geometrical model and gabor filter. In *Intelligent vehicles symposium (IV), 2010 IEEE* (pp. 59–64). IEEE.