Informatics and Software Engineering
p-ISSN/e-ISSN: 2988-2818/2988-2222
Homepage: https://sanscientific.com/journal/index.php/ise
1**(2) 51-58 (2023)**
DOI: 10.58777/ise.v1i2.148

Research Article

Preprocessing Effects in Road Traffic Scene Analysis Using Machine Learning Models

Daehyon Kim[1*]
[1]Department of Culture and Tourism Management, Chonnam National University, South Korea

Received: November 15, 2023; Accepted: November 21, 2023

Abstract
Image-based real-time object detection is a key issue in integrated automatic traffic monitoring and control systems and is one of the main research topics in Intelligent Transportation Systems (ITS). Despite the growing interest in the application of image processing techniques for traffic scene analysis, vehicle detection algorithms are still unreliable in complex real-world backgrounds. Machine learning models are increasingly being applied to automatic video-based object detection and traffic scene analysis due to their excellent performance in real-time pattern recognition. However, the performance of machine learning models is highly dependent on the properties of the input vectors. Studies have shown that the preprocessing of raw image data obtained from digital video can improve the predictive performance of machine learning models. The purpose of this study is to investigate the predictive performance of machine learning models using preprocessing methods such as image size reduction and image filtering. The experiment was performed with the Backpropagation model, which is one of the most popular machine learning models, and the predictive performance was compared to see how it could be affected by different preprocessing techniques.

Keywords: Intelligent Transportation Systems, Machine learning, Backpropagation, Preprocessing.

Abstrak
Deteksi objek real-time berbasis gambar adalah isu kunci dalam sistem pemantauan dan kontrol lalu lintas otomatis terintegrasi dan merupakan salah satu topik penelitian utama dalam Sistem Transportasi Cerdas. Meskipun minat yang meningkat dalam penerapan teknik pemrosesan gambar untuk analisis scene lalu lintas, algoritma deteksi kendaraan masih belum dapat diandalkan dalam latar belakang dunia nyata yang kompleks. Model pembelajaran mesin semakin diterapkan pada deteksi objek berbasis video otomatis dan analisis scene lalu lintas karena kinerja mereka yang sangat baik dalam pengenalan pola real-time. Namun, kinerja model pembelajaran mesin sangat bergantung pada sifat vektor masukan. Studi telah menunjukkan bahwa pra-pemrosesan data gambar mentah yang diperoleh dari video digital dapat meningkatkan kinerja prediktif dari model pembelajaran mesin. Tujuan dari penelitian ini adalah untuk menyelidiki kinerja prediktif dari model pembelajaran mesin menggunakan metode pra-pemrosesan seperti reduksi ukuran gambar dan penyaringan gambar. Percobaan dilakukan dengan model Backpropagation, yang merupakan salah satu model pembelajaran mesin paling populer, dan kinerja prediktifnya dibandingkan untuk melihat bagaimana hal itu dapat dipengaruhi oleh berbagai teknik pra-pemrosesan.

Kata Kunci: Sistem Transportasi Cerdas, Pembelajaran mesin, Backpropagation, Pra-pemrosesan

How to cite: D. Kim (2023). Preprocessing Effects in Road Traffic Scene Analysis Using Machine Learning Models

[*]Corresponding author: Daehyon Kim (daehyon@chonnam.ac.kr)

1. Introduction

Interest in the topic of Intelligent Transportation Systems (ITS) has increased over the past 30 years with the aim of improving safety, maximizing transport capacity, increasing productivity, improving operational efficiency, ensuring sustainable environment and saving energy. In addition, image-based real-time object detection is one of the main research topics of Intelligent Transportation Systems (ITS) as a core task of the integrated traffic control system. In addition, image-based real-time object detection is a core task of the integrated traffic control system and has become a major research field for Intelligent Transportation Systems (ITS).

For several decades, significant research advances have been made in the field of image processing for traffic scene analysis, such as vehicle counting, traffic congestion measurement, vehicle location tracking and incident detection, but the detection accuracy is still low in complex background images (Hoose and Willumsen, 1987; Siyal and Fathy, 1999; Betke et al., 2000; Pang et al., 2007; Kamijo and Fujimura, 2010; Song et al., 2019). In addition, the application of some learning models such as artificial neural networks (ANNs) and support vector machines (SVMs) has increased tremendously and has been successfully applied to traffic scene analysis due to their remarkable ability to achieve high precision.

In machine learning, the properties of the input vector in training data have a great impact on the reliability and predictive performance of the model. Therefore, it is important to choose a feature input vector to achieve high performance in model training. In addition, preprocessing techniques such as image size reduction and image filtering generate new input vector properties for learning and reduce the computer memory and computing time required for training.

The purpose of this study is to investigate the effect of preprocessing methods on the prediction accuracy of a learning machine. In this study, various preprocessing techniques were experimentally verified using a machine learning model of backpropagation, which is widely used as a learning algorithm among neural network models.

2. Machine Learning for Traffic Scene Analysis

Currently, the most widely used neural network model is the backpropagation (Rumelhart, et. al., 1986a) model, which is a basic model of deep learning models that are receiving increasing interest in recent years and is widely used in various problems.

Some advanced backpropagation models, such as the backpropagation with momentum (Rumelhart, et. al., 1986b), Quickprop (Fahlman, 1988) and Backpropagation with Momentum and Prime-offset (Kim, 2002), have been proposed and widely used, mainly to solve some problems of standard backpropagation, such as computation time for training and the possibility of getting stuck in local minima during training.

In order to increase the speed of learning, Rumelhart et al. (1986b) proposed an enhanced backpropagation model, which is a backpropagation with momentum, by adding a momentum term to the delta rule. This additional term, called momentum, tends to maintain the same direction for the weight changes. The momentum parameter provides a type of momentum in weight space that effectively filters out high-frequency variations of the error-surface in the weight space. Fahlman (1988) has developed an algorithm called Quickprop to speed the learning of the Backpropagation model. Quickprop has a number of parameters such as Maximum growth factor, Weight decay, and Prime-offset that are more fine-tuned than the standard Backpropagation. More importantly, Kim (2002) showed that backpropagation using momentum and prime offset (BPMP) was superior to other backpropagation models.

The learning algorithm of the backpropagation can be summarized as follows (Kim, 2002; Kim, 2018; Kim, 2021):

Step 1: Randomize initial weights and set up the input and output vectors
Step 2: Calculate output value for each unit of the network by using

$$Out_{pk}^{l} = f_{pk}^{l}(Net_{pk}^{l}) \quad \text{where} \quad Net_{pk}^{l} = \sum_{j=1}^{K_{l-1}} w_{kj}^{l} In_{pj}^{l} + \theta_{k}^{l} \tag{1}$$

In Equation (1), In_{pj}^{l} are inputs to the k^{th} unit in the layer l, w is the number of synaptic weights in the network, θ_{k}^{l} is a bias term, K_l is the number of l layer units, p is a training pattern and $f(\,)$ is an activation function.
Step 3: For the output layer, $l = L$, calculate the values of weight changes by using

$$\Delta_p w_{kj}^{L} = \eta \delta_{pk}^{L} In_{pj}^{L}, \quad \delta_{pk}^{L} = \left(y_{pk} - Out_{pk}^{L}\right)\left(f_{k}^{L}(Net_{pk}^{L})\right)' + \text{Prime-offset} \tag{2}$$

In Equation (2), y_{pk} is the desired output.

Step 4: For the hidden layers, $l = 1, \ldots, L\text{-}1$, calculate the values of weight changes using

$$\Delta_p w_{ju}^l = \eta \delta_{pj}^l In_{ij}^l \ , \quad \text{where} \quad \delta_{pj}^l = \left(f_j^{\ l}(Net_{pj}^l) \right)' \sum_{k=1}^{K_{l+1}} \delta_{pk}^{l+1} w_{kj}^{l+1} \tag{3}$$

Step 5: Update weights on the output layers by

$$w_{kj}^L(t+1) = w_{kj}^L(t) + \Delta_p w_{kj}^L \tag{4}$$

Step 6: Update synaptic weights on hidden layers by

$$w_{ju}^l(t+1) = w_{ju}^l(t) + \Delta_p w_{ju}^l \quad \text{for } l = 1,\ldots, L\text{-}1$$

(5)

Step 7: Repeat previous steps, until the average squared error computed over the entire training data set is at an acceptably small value. The error for the output units is calculated by

$$Err_p = \sum_{k=1}^{K_L} (y_{pk} - Out_{pk}^L)^2 \tag{6}$$

3. Preprocessing Methods

In the backpropagation neural network mode, data is frequently preprocessed in order to reduce the number of input units and to reduce network complexity and computing time. While preprocessing makes the network architecture simpler, it may lead to loss of some important information for pattern recognition, such as the geometric relationship of pixel gray values. Nonetheless, preprocessing has improved the predictive performance of neural network models (Kim, 2010). In particular, preprocessing may play a role in noise reduction by removing or smearing some noise in images.

In this study, two common types of preprocessing filters, the arithmetic mean filter and the median filter, were applied to mitigate the effects of data preprocessing on the prediction accuracy of a backpropagation model in machine learning.

3.1 Arithmetic mean filter

The arithmetic mean filter can be defined by follows (Gonzalez and Woods, 2008). Let S_{xy} represent the set of coordinates in a rectangular subimage window (neighborhood) of size a × b, centered at point (x, y). The arithmetic mean filter computes the average value of the corrupted image f(x, y) in the area defined by S_{xy}.

The value of the restored image $\hat{h}$ at point (x, y) is simply the arithmetic mean computed using the pixels in the region defined by S_{xy}. In other words,

$$\hat{h}(x,y) = \frac{1}{ab} \sum_{(s,t) \in S_{xy}} f(s,t) \tag{7}$$

This operation can be implemented using a spatial filter of size a × b in which all coefficients have value 1/ab. A mean filter smooths local variations in an image, and noise is reduced as a result of blurring.

3.2 Median filter

Order-statistic filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. The median filter is the best order statistic filter and replaces the value of a pixel by the median of the intensity values in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median). Median filters are popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise (Gonzalez and Woods, 2008). The median filter can be defined mathematically as,

$$\hat{h}(x,y) = \underset{(s,t) \in S_{xy}}{\text{median}} \{ f(s,t) \} \tag{8}$$

The value of the pixel at (x, y) is included in the computation of the median. In order to perform median filtering at a point in an image, we first sort the values of the pixel in the neighborhood, determine their median, and assign that value to the corresponding pixel in the filtered image. For example, in a 3× 3 neighborhood the median is the 5th largest value, in a 5× 5 neighborhood it is the 13th largest value, and so on.

3.3 Image geometric transformation

Image geometric transformations modify spatial relationships between pixels in an image. Geometric transformation in digital image processing consists of two basic operations: spatial transformation of coordinates and intensity interpolation, which assigns intensity values to spatially transformed pixels (Gonzalez and Woods, 2008). The transformation of coordinates may be expressed as $g(u,v) = f(x,y)$, where $f(x,y)$ are pixel coordinates in the original image and $g(u,v)$ are the corresponding pixel coordinates in the transformed image. For example, the transformation $g(u,v) = \mathrm{f}(x,y) = (\mathrm{x}/2, y/2)$ shrinks the original image to half its size in both spatial directions. Image shrinking is achieved by row-column deletion (eg, to reduce an image by half, delete rows and columns one by one).

In machine learning models, image geometric transformation can serve to reduce the number of input units and remove some noise. In particular, it can improve learning performance by simplifying the network architecture and making training more efficient. In this study, we compare the performance of machine learning using the original image size and the reduced image and examine how the machine learning models can vary depending on preprocessing.

4. Experiments and Results

4.1 Experimental data sets

For the experiments in this study, all data sets are the same as those of the previous research (Kim, 2010). The task is to classify three different patterns in traffic scenes: 1st pattern, 2nd pattern, and 3rd pattern. Figure 1 shows the preprocessed image data where the 1st pattern corresponds to the front part of a vehicle, 2nd pattern corresponds to the top of a vehicle, and 3rd pattern is an image of the road with no vehicles. The number of data sets for training and test are 230 and 700, respectively.

As shown in Figure 1, there are three patterns in the preprocessed images collected from the external environment. In this study, four types of preprocessing - Mean filter, Median filter, Geometric transform (30*15), Geometric transform (15*8), - were performed to evaluate the classification accuracy by learning input vectors in neural network models.

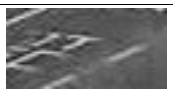
Category	1st Pattern	2nd Pattern	3rd Pattern
Original image (60*30)			
Geometric transform (30*15)			
Geometric transform (15*8)			
Mean filter			
Median filter			

Figure 1. Preprocessed images of three patterns

4.2 Performance Evaluation of Preprocessing

In this study, the prediction rule after training was defined as follows. The output vector should be binarized as

$$\begin{cases} \text{If } Out(i) > 0.5 \text{ then } Out(i) = 1 \\ \quad \text{otherwisw } Out(i) = 0 \end{cases}$$, where $Out(i)$ is the predicted value of output unit i.

A system is defined as unpredictable if its output occurs 1 in more than one pattern. For the experiments, the network topology are 450(input units)-225(hidden neurons)-3(output units) for 1st preprocessing (Mean filter with 30 by 15 pixels) and 2nd preprocessing (Mean filter and geometric transformation (15*8)) and 120(input units)-60(hidden neurons)-3(output units) for Median filter and geometric transformation (30*15) and Median filter and geometric transformation (15*8), respectively.

In the experiments of this study, the network topology utilized 450(input units)-225(hidden neurons)-3(output units) for the first-order preprocessing (mean filter and geometric transformation 30 x 15 pixels) and the second-order preprocessing (mean filter and geometric transformation 15*8 pixels), and the third-order preprocessing (median filter and geometric transformation 30*15 pixels), 4th order preprocessing (and median filter and geometric transformation 15*8 pixels) were applied 120(input units)-60(hidden neurons)-3(output units). The learning model used in this study is BPMP (Backpropagation with momentum and Prime-offset), and the hyperparameter variable values are learning rate of 0.1, momentum of 0.95, and prime offset of 0.1, and the sequential mode was used as the learning method. Ten trials were performed with different initial weights for each data to resolve the effect of initial weights in the neural network models. Table 1, Table2, Table3, Table 4 and Figure 2 show the general performance of the learning machine using the backpropagation model and the effect of preprocessing on image data.

Table 1 shows the prediction error rates over 10 experiments for the mean filter and geometric transformation with image size of 30×15. For pattern 1, the worst-case prediction error was 8.3% for pattern 1, 33.0% for pattern 2, and 0.0% for pattern 3. On the other hand, in the best case, the prediction error was 0.3% for pattern 1, 30.0% for pattern 2, and 22.0% for pattern 3. This means that as the prediction accuracy for pattern1 increases, the prediction accuracy for pattern3 decreases. Also, for pattern 2, the worst-case prediction error was 2.3% for pattern 1, 34.0% for pattern 2, and 0.0% for pattern 3. In the best case, the prediction error was 6.0% for pattern 1, 15.3% for pattern 2, and 2.0% for pattern 3. Also, for pattern 3, the worst-case prediction error was 0.3% for pattern 1, 30.0% for pattern 2, and 22.0% for pattern 3. In the best case, the prediction error was 3.7% for pattern 1, 24.3% for pattern 2, and 0.0% for pattern 3. In this type of preprocessing experiment, the overall performance of the network showed the best performance in terms of prediction rate and prediction accuracy, with patterns 1, 2, and 3 having a total error rate of 9.4% and unpredictability of 6.3%.

Table 1. Prediction error rate (%) for mean filter and geometric transformation (30*15)

Analysis Criteria	Error rate (%)				Prediction failure rate (%)
	pat 1	pat 2	pat 3	total	
Worst case based on pat.1	8.3	33.0	0.0	17.7	1.4
Best case based on pat.1	0.3	30.0	22.0	16.1	2.1
Worst case based on pat.2	2.3	34.0	0.0	15.6	4.6
Best case based on pat.2	6.0	15.3	2.0	9.4	6.3
Worst case based on pat.3	0.3	30.0	22.0	16.1	2.1
Best case based on pat.3	3.7	24.3	0.0	12.0	5.4
Worst case based on Total	8.3	33.0	0.0	17.7	1.4
Best case based on Total	6.0	15.3	2.0	9.4	6.3
Worst case based on prediction failure	4.3	22.7	0.0	11.6	8.4
Best case based on prediction failure	8.3	33.0	0.0	17.7	1.4

Table 2. Prediction error rate (%) for mean filter and geometric transformation (15*8)

Analysis Criteria	Error rate (%)				Prediction failure rate (%)
	pat 1	pat 2	pat 3	total	
Worst case based on pat.1	9.0	31.0	0.0	17.1	1.9
Best case based on pat.1	1.3	21.3	2.0	10.0	4.6
Worst case based on pat.2	9.0	31.0	0.0	17.1	1.9
Best case based on pat.2	6.0	21.3	0.0	11.7	1.6
Worst case based on pat.3	1.3	21.3	2.0	10.0	4.6
Best case based on pat.3	4.3	22.0	0.0	11.3	3.4
Worst case based on Total	9.0	31.0	0.0	17.1	1.9
Best case based on Total	1.3	21.3	2.0	10.0	4.6
Worst case based on prediction failure	4.3	30.7	0.0	15.0	8.1
Best case based on prediction failure	6.0	21.3	0.0	11.7	1.6

Table 2 shows the prediction error rates over 10 experiments for the mean filter and geometric transformation with image size of 15×8. The overall performance of the network in this preprocessing experiment showed the best performance in terms of prediction rate and prediction accuracy, with patterns 1, 2, and 3 having a total error rate of 11.7% - the prediction error was 6.0% for pattern 1, 21.3% for pattern 2, and 0.0% for pattern 3 - and unpredictability of 1.6%. The results of this study show that image shrinking may provide somewhat better performance in terms of prediction accuracy and somewhat poor results in terms of predictive ability.

Table 3 shows the prediction error rates over 10 experiments for the median filter and geometric transformation with image size of 30×15. The overall performance of the network in this preprocessing experiment showed the best performance in terms of prediction rate and prediction accuracy, with patterns 1, 2, and 3 having a total error rate of 9.1% - the prediction error was 4.0% for pattern 1, 17.3% for pattern 2, and 0.0% for pattern 3 - and unpredictability of 11.6%.

Table 3. Prediction error rate (%) in median filter and geometric transformation (30*15)

Analysis Criteria	Error rate (%)				Prediction failure rate (%)
	pat 1	pat 2	pat 3	total	
Worst case based on pat.1	5.7	28.0	0.0	14.4	2.4
Best case based on pat.1	1.3	29.3	0.0	13.1	6.3
Worst case based on pat.2	1.3	33.3	6.0	15.7	2.0
Best case based on pat.2	4.0	17.3	0.0	9.1	11.6
Worst case based on pat.3	1.3	33.3	6.0	15.7	2.0
Best case based on pat.3	4.0	17.3	0.0	9.1	11.6
Worst case based on Total	1.3	33.3	6.0	15.7	2.0
Best case based on Total	4.0	17.3	0.0	9.1	11.6
Worst case based on prediction failure	4.0	17.3	0.0	9.1	11.6
Best case based on prediction failure	4.3	24.3	0.0	12.3	1.3

Table 4 shows the prediction error rates over 10 experiments for the median filter and geometric transformation with image size of 15×8. The overall performance of the network in this preprocessing experiment showed the best performance in terms of prediction rate and prediction accuracy, with patterns 1, 2, and 3 having a total error rate of 11.0% - the prediction error was 5.0% for pattern 1, 20.0% for pattern 2, and 2.0% for pattern 3 - and unpredictability of 4.4%.

Table 4. Prediction error rate (%) in median filter and geometric transformation (15*8)

Analysis Criteria	Error rate (%)				Prediction failure rate (%)
	pat 1	pat 2	pat 3	total	
Worst case based on pat.1	10.3	23.0	0.0	14.3	7.1
Best case based on pat.1	3.3	32.3	0.0	15.3	10.7
Worst case based on pat.2	6.0	35.3	0.0	17.7	3.7
Best case based on pat.2	9.0	22.3	0.0	13.4	2.6
Worst case based on pat.3	4.7	28.3	15.0	16.3	3.7
Best case based on pat.3	9.0	22.3	0.0	13.4	2.6
Worst case based on Total	9.3	34.0	0.0	18.6	1.1
Best case based on Total	5.0	20.0	2.0	11.0	4.4
Worst case based on prediction failure	3.3	32.3	0.0	15.3	10.7
Best case based on prediction failure	9.3	34.0	0.0	18.6	1.1

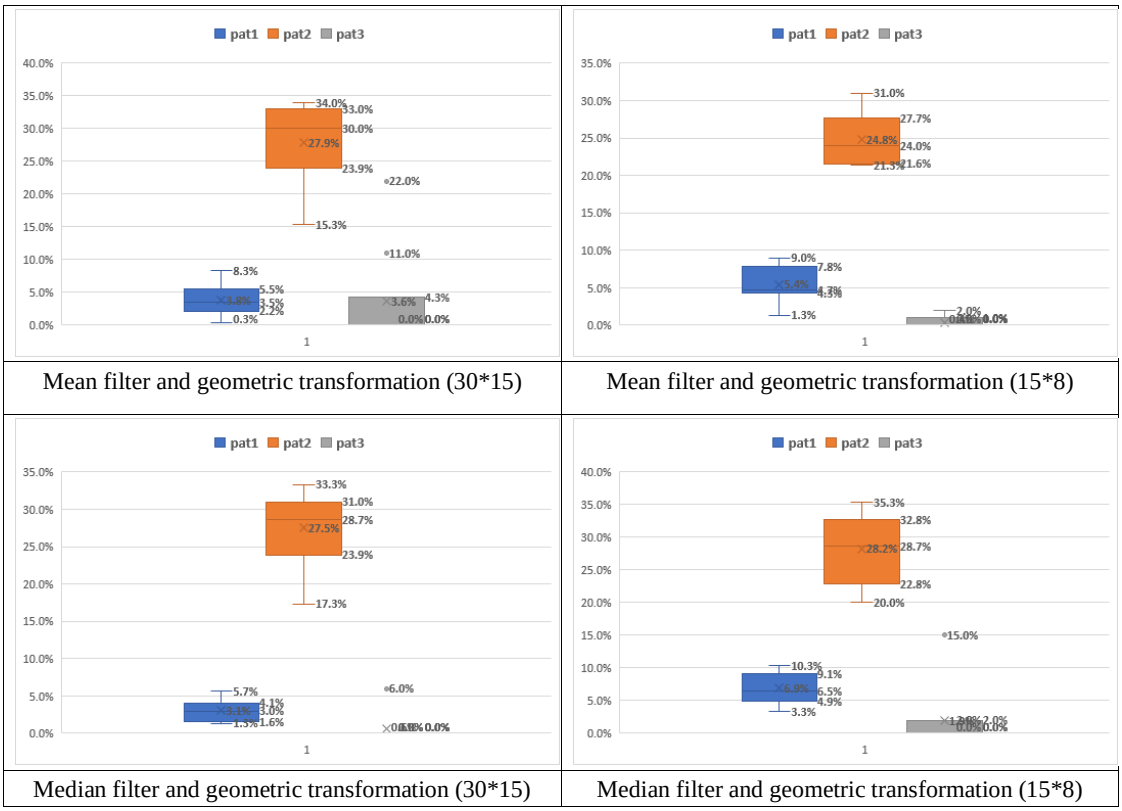


Figure 2. Prediction error rate distribution for three patterns of preprocessed images

As a result of analyzing the experiment with the total error rate of three patterns, good performance was shown in the following order; i.e. the 3rd preprocessing (Median filter and geometric transformation with 30*15, 9.1% error) > 1st preprocessing (Mean filter and geometric transformation with 30*15, 9.4% error) > 2nd preprocessing (Mean filter and geometric transformation with 15*8, 10.0% error) > 4th preprocessing (Median filter and geometric transformation with15*8, 11.0% error) showed good performance in the order. On the other hand, the performance in terms of the prediction rate, showed good performance in the following order; i.e. 4th preprocessing (Median filter and geometric transformation with15*8, 1.1% of unpredictability)> 3rd preprocessing (Median filter and geometric transformation with 30*15, 1.3% of unpredictability)> 1st preprocessing (Mean filter and geometric transformation with 30*15, 1.4% of unpredictability) > 2nd preprocessing (Mean filter and geometric transformation with 15*8, 1.6% of unpredictability).

4. Conclusion

Preprocessing plays an important role in learning machine models with complex input images. Some preprocessing methods, such as image size reduction and image filtering, can improve predictive performance in a machine learning with backpropagation learning algorithm.

In this study, four preprocessing methods were used to develop image processing technology for traffic scene analysis using a robust backpropagation machine learning model. According to the results of the study, there was a difference in the recognition rate for recognizing each pattern according to the preprocessing method. Therefore, it is necessary to develop various preprocessing techniques to improve the predictive performance of machine learning. In particular, in order to increase the accuracy of the prediction system, it is important to analyze the characteristics of the original data as well as the preprocessed data. This means that preprocessing makes the network simpler and reduces or eliminates image noise, but some important information about pattern recognition, such as geometric relationships of object and pixel grayscale, may be lost.

The experimental results of this study show that various preprocessing techniques have different prediction accuracy depending on the learning model, so the results of this study can be used to increase the prediction rate and reduce the prediction error rate in various machine learning models.

References

Betke, M., Haritaoglu, E. and Davis, L.S. (2000). Real-time multiple vehicle detection and tracking from a moving vehicle, *Machine Vision and Applications*, *12*, 69-83.

Fahlman, S. E. (1988). An empirical study of learning speed in backpropagation networks, *Technical Report CMU-CS-88-162*, Canegie Mellon University.

Gonzalez, R. C. and Woods, R. E. (2008). Digital Image Processing, *NJ 07458*, Pearson Prentice Hall.

Hoose, N. and Willumsen, L.G. (1987). Automatically extracting traffic data from video-tape using the CLIP4 parallel image processor, *Pattern Recognition Letters*, *6*(3), 199-213.

Kamijo, S. and Fujimura, K. (2010). Incident Detection in Heavy Traffics in Tunnels by the Interlayer Feedback Algorithm, *Int. J. ITS Res. 8*, 121-130.

Kim, D. and Chang, Y. (2008). Automatic Incident Detection Using Machine Learning, *Seoul Urban Studies*. *6*(1), 71-81.

Kim, D. (2018). Deep Learning Neural Networks for Automatic Vehicle Incident Detection, *Asia-pacific Journal of Convergent Research Interchange*, *4*(3), 107-117.

Kim, D. (2021). Origin-Destination Flow Estimation Using Artificial Neural Networks, *International Journal of Transportation*, *9*(5), 75-82.

Kim, D. (2010). Pre-processing of inputs to a neural network model for better performance in traffic scene analysis, *Civil Engineering and Environmental Systems*, *27*(1), 23-31.

Kim, D. (2002). Standard and Advanced Backpropagation models for image processing application in traffic engineering, *ITS Journal*, *7*(3-4), 199-211.

Pang, C.C.C., Lam, W.W. and Yung, N.H. (2007). A Method for Vehicle Count in the Presence of Multiple-Vehicle Occlusions in Traffic Images, *IEEE Trans. Intell. Transp. Syst.*, *8*(3), 441-459.

Rumelhart, D. E., Hinton, G. E. and McClelland, J. L. (1986a). A general framework for parallel distributed processing, *In D. E. Rumelhart, J. L. McClelland and the PDP Research Group (Eds)*. Parallel Distributed Processing, (vol. 1). Cambridge, Massachusetts: MIT Press, 45-76.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986b). Learning internal representations by error propagation, *In D. E. Rumelhart, J. L. McClelland and the PDP Research Group (Eds)*. Parallel Distributed Processing, (vol. 1). Cambridge, Massachusetts: MIT Press, 318-364.

Siyal, M.Y. and Fathy, M. (1999). A neural-vision based approach to measure traffic queue parameters in real-time, *Pattern Recognition Letters*, *20*, 761-770

Song, H., Liang, H., Li, H., Dai, Z. and Yun, X. (2019). Vision-based vehicle detection and counting system using deep learning in highway scenes, *European Transport Research Review*, *11*(1), 1-16