Received March 24, 2019, accepted April 15, 2019, date of publication April 24, 2019, date of current version May 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2912311

Real-Time Embedded Traffic Sign Recognition Using Efficient Convolutional Neural Network

XIE BANGQUAN, (Member, IEEE), AND WENG XIAO XIONG
School of Civil Engineering and Transportation, South China University of Technology, Guangzhou 510000, China

Corresponding authors: Xie Bangquan (418153497@qq.com) and Weng Xiao Xiong (ctxxweng@qq.com)

ABSTRACT Traffic sign recognition(TSR) based on deep learning is rapidly developing. Specifically, TSR contains two technologies, namely, traffic sign classification (TSC) and traffic sign detection (TSD). However, the challenge of TSR is to ensure its efficiency, which means adequate accuracy, generalization, and speed in real-time by a computationally limited platform. In this paper, we will introduce a new efficient TSC network called ENet (efficient network) and a TSD network called EmdNet (efficient network using multiscale operation and depthwise separable convolution). We used data mining and multiscale operation to improve the accuracy and generalization ability and used depthwise separable convolution (DSC) to improve the speed. The resulting ENet possesses 0.9 M parameters (1/15 the parameters of the start-of-the-art method) while still achieving an accuracy of 98.6 % on the German Traffic Sign Recognition benchmark (GTSRB). In addition, we design EmdNet' s backbone network according to the principles of ENet. The EmdNet with the SDD Framework possesses only 6.3 M parameters, which is similar to MobileNet's scale.

INDEX TERMS Autonomous driving, convolutional neural network, deep learning, efficient network, perception, traffic sign recognition.

I. INTRODUCTION

Autonomous driving involves broad technologies including perception, localization, path-planning, decision-making, etc. Traffic Sign Recognition (TSR), which is a critical part of the perception technology, includes TSC and TSD. In recent years, TSR has increasingly used deep learning technology. For example, the convolutional neural network is becoming deeper and larger, its accuracy can meet many application scenarios. And many end-to-end technologies which are faster and more convenient, such as one-stage object detection methods, are also used in TSR.

However, there are still many challenges for real-world applications. First, Classical methods rely on shape and color information, which is hard for TSR to work accurately in the real world. Second, The traditional methods are good at identifying traffic signs with similar shapes or colors, which is difficult to maintain a high generalization ability because of the complexity of practical applications. Third, the networks of Object Classification [1]–[4] and TSC [5]–[8] are becoming deeper and larger in order to obtain higher accuracy in recent years, which will reduce the speed and increase the calculation. But the TSR must occur in real time on a computationally limited platform. Thus, the TSR needs to be more efficient, which means having adequate accuracy, generalization, and speed.

In this paper, first, we propose an innovative network construction method, to build a new efficient TSC network called ENet which can achieve an accuracy of 98.6% on the GTSRB. And we build a new efficient TSD network called EmdNet which can solve the practical problems, such as partial occlusion, illumination changes, cluttered background and so on(Just as shown in the fourth part of this paper).

Second, we use the theory of data mining to divide the validation set and perform data augmentation for ENet, and use multiscale operation for EmdNet. And the training samples which include the GTSRB and the LISA US Traffic Sign Dataset (LISA), are from application scenes. All of these can improve the generalization ability of network.

Third, in Traffic Sign Classification, the key to constructing a network is how to extract and integrate features efficiently. Generally, the five networks [9]–[13] all benefitted from DSC. We construct ENet using two novel methods of DSC and Shortcuts, which makes the appropriate trade-off between accuracy and speed. ENet has 0.9M parameters, and spends only 0.62 ms to identify a sample. In Traffic Sign Detection, there are two approaches: the two-stage and one-stage methods. The latter adopted by

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

EmdNet is faster and needs fewer resource. And EmdNet uses the improved multiscale and DSC. Based on modified SDD framework, Deeplab-VGG possesses 33.1M parameters, Inception V2 has 13.7M [14], and EmdNet which has 6.3M is slightly smaller than MobileNet [9].

In addition, there is a close connection between ENet and EmdNet. They both aim to improve the efficiency to build a network. The Shortcut used by ENet and the multiscale convolution adopted by EmdNet are similar in form and function, and the two networks use the DSC. And we design EmdNet' s backbone network according to the principles of ENet.

Our major contributions are as follows.

(1) An innovative network construction method is proposed. We use two famous networks to choose the network size that meets the task, adopt and evaluate various deep learning techniques in four scale samples. And design six networks for comparison to select the best.

(2) We use the theory of data mining to divide the validation set and perform data augmentation. We analyze the datasets in order to perform a more reasonable division of the validation set and effective data augmentation, which will enhance the generalization ability of the network.

(3) We have built a new efficient TSC network. We have provided two versions to meet the different needs of practical applications.

(4) We have built a new efficient TSD network. This network can be adapted to different application scenes in a real-time embedded platform.

This paper is organized as follows: Section 2 describes the related work, Section 3 introduces the methodologies of TSC and TSD, Section 4 is the discussions and evaluations, and section 5 concludes the current work and introduces future work.

II. RELATED WORD

Traffic Sign Recognition (TSR) involves two technologies: TSC and TSD. And TSC is a specific application of Object Classification. The relationship between TSD and Object Detection is similar to this.

A. OBJECT CLASSIFICATION

In recent years, there have been many classic networks related to Object Classification, such as AlexNet [1] (2012), VGG [2] (2014), GoogLeNet [3]–[16] (2015, 2015, 2015, 2016), ResNet [4] (2016), SqueezeNet [17] (2016), Xception [18] (2016), MobileNet [9], [10] (2017)(2018, 1), ShufficNet [11], [12] (2017)(2018, 7), SE-Net [19] (2017), DenseNet [20] (2017), CondenseNet [13] (2017), ResNeXt [21] (2017), and the automatic neutral architecture search [22]–[24]. At first, the convolutional neural network was deepened and enlarged to obtain higher accuracy. However, in recent years, the networks have become smaller and more effective. To achieve accuracy comparable to VGG on ImageNet, ResNets reduces the amount of computations to 1/5 the amount, DenseNets to 1/10 the amount, and MobileNets and ShuffleNets to 1/25 the amount. How to make the network more efficient has become a new research focus.

There are three methods to construct an efficient network: using quantized weights [5], [6], [25], pruning redundant connections [26]–[29], or using new and more efficient networks [4]–[20]. In recent years, there are many new networks with high efficiency. SqueezeNet (2016) created a Fire module that included a squeeze and expand layer to reduce the filter quantity and input channel. Xception (2016) learned from DSC to improve Inception-V3. MobileNet-V1 (2017.04) used DSC instead of traditional convolutions to reduce the parameters and enhance the speed. The DSC factorizes a standard convolution into two convolutions called depthwise and pointwise convolution. About two months later, ShufficNet-V1 (2017.06) used group convolution instead of MobileNet' s depthwise convolution, and used the channel shuffle operation instead of MobileNet' s pointwise convolution which requires more calculations. In that case, ShufficNet-V1 is faster than MobileNet-V1.

SE-Net (2017) introduced the SE block that contained the Squeeze and Excitation operation. CondenseNet(2017) learned from DenseNet' s jump connection and ShuffleNet' s channel shuffle operation, and invented the learned group convolution.

MobileNet-V2 (2018,1) applied DSC to residual blocks, which used Inverted Residuals and Linear Bottlenecks. The Inverted Residuals extracted more features than the residual block. In addition, Linear Bottlenecks are used behind "compression" instead of Relu to decrease the feature losses. ShuffleNet-V2 (2018,7) derives four practical guidelines for efficient network architecture design: using balanced convolutions, considering group convolutions, reducing the degree of fragmentation and elementwise operations.

To sum up, the primary differences between these aforementioned networks lie in how to extract and integrate features efficiently. To extract features, MobileNet-V1 used depth wise convolutions, ShufficNet-V1 used group convolutions, CondenseNet used learned group convolutions, MobileNet-V2 used depthwise convolutions with an added 1×1 "expansion" layer before, and ShuffleNet-V2 used Channel Split. To integrate features, MobileNet-V1 and V2 used pointwise convolution, ShufficNet-V1 (2017) and CondenseNet used Channel Shuffle, and ShuffleNet-V2 [12] (2018.7) used Channel Split. On classification, the accuracy rank of the methods [12] is ShuffleNet-V2 $\geq$ MobileNet-V2>ShufficNet-V1>Xception>MobileNet-V1. Furthermore, we can find that the five networks are all benefitted from depthwise separable convolutions.

Traffic Sign Classification [5]–[8] is a specific application of Object Classification. [6] and [5] used the Spatial Transformer Network (SPN) to achieve state-of-the-art results on the GTSRB.

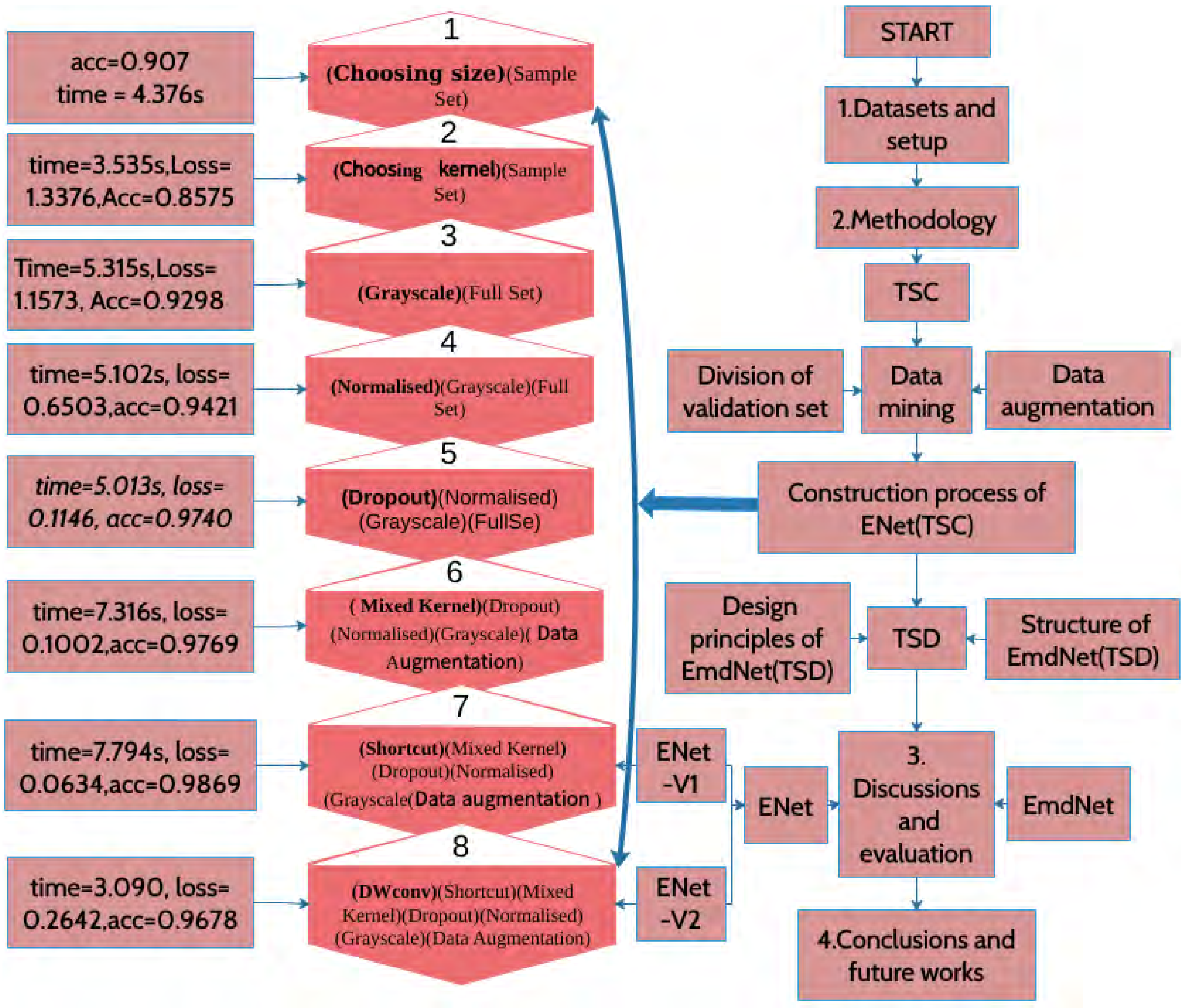
acc=0.907
time = 4.376s
time=3.535s,Loss=1.3376,Acc=0.8575
Time=5.315s,Loss=1.1573, Acc=0.9298
time=5.102s, loss=0.6503,acc=0.9421
time=5.013s, loss=0.1146, acc=0.9740
time=7.316s, loss=0.1002,acc=0.9769
time=7.794s, loss=0.0634,acc=0.9869
time=3.090, loss=0.2642,acc=0.9678
1 (Choosing size)(Sample Set)
2 (Choosing kernel)(Sample Set)
3 (Grayscale)(Full Set)
4 (Normalised)(Grayscale)(Full Set)
5 (Dropout)(Normalised)(Grayscale)(FullSe)
6 (Mixed Kernel)(Dropout)(Normalised)(Grayscale)(Data Augmentation)
7 (Shortcut)(Mixed Kernel)(Dropout)(Normalised)(Grayscale(Data augmentation)
8 (DWconv)(Shortcut)(Mixed Kernel)(Dropout)(Normalised)(Grayscale)(Data Augmentation)
START
1.Datasets and setup
2.Methodology
TSC
Division of validation set
Data mining
Data augmentation
Construction process of ENet(TSC)
Design principles of EmdNet(TSD)
TSD
Structure of EmdNet(TSD)
ENet -V1
ENet
3. Discussions and evaluation
EmdNet
ENet -V2
4.Conclusions and future works

FIGURE 1. The right is the workflow of the full text, the middle is eight steps of building ENet,and the left is the data flow of ENet.

B. OBJECT DETECTION

Object Detection related to deep learning has two approaches: the two-stage and one-stage methods.

The two-stage method is based on region proposal. The R-CNN [25] (2014) obviously improved the average precision for Object Detection. The R-CNN included four steps: A, it generated 2K region proposal using selective search; B, each region proposal was scaled to the same size; C, each region proposal extracted its respective features using a CNN; and D, it used a classifier to predict the classes of objects and used regressors to predict the positions of detection boxes. Later, there have been many better ideas to improve the R-CNN. SPP Net [30] (2014) improved the B and C steps of the R-CNN. B caused data loss or geometric distortion, so SPP Net added the spatial pyramid to CNN to achieve multiscale data input. C was inefficient, so SPP Net only tried to perform convolutions on the original image to get feature map of the whole image. The Fast R-CNN [31] (2015) adopted the SPP Net method to improve the R-CNN. The Faster R-CNN [26] (2015) added the Region Proposal Network (RPN) to extract edges instead of A, which achieved better performance. In addition, the Mask R-CNN [27] (2018) only added a Mask Prediction Branch to the Faster R-CNN and proposed ROI Alignment instead of ROI Pooling.

The one-stage method is based on the regression method, which is an end-to-end technology. YOLO [28] (2016) and YOLO-V2 [29] (2017) integrated object detection and classification into a single convolutional network, thereby increasing the speed but decreasing the accuracy. YOLO V3 [32] (2018.3) used Darknet-53 to replace Darknet-19 as the backbone network and used Multiscale prediction. SSD [33] (2016.5) was faster than YOLO, and had almost the same accuracy as the Faster R-CNN. SSD invented the Multiscale feature map and used convolutional predictors for detection. The Tiny-SSD [34] (2018.2) used Squeeze-Net to replace VGG-16 as the backbone network.

Traffic Sign Detection is a specific application area of Object Detection. Classic traffic sign detection methods rely on shape and color information [35]–[37]. In recent years,

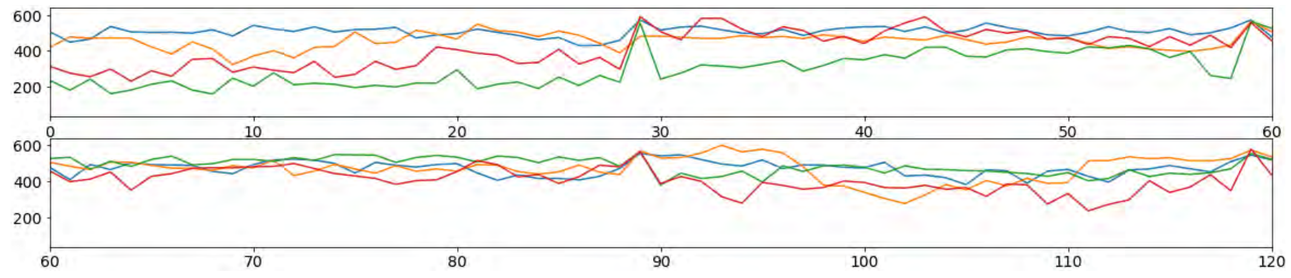

FIGURE 2. The regularity of the train set. The X-axis represents each image and the Y-axis represents the Euclidean distance of the two adjacent graphs of four classes.

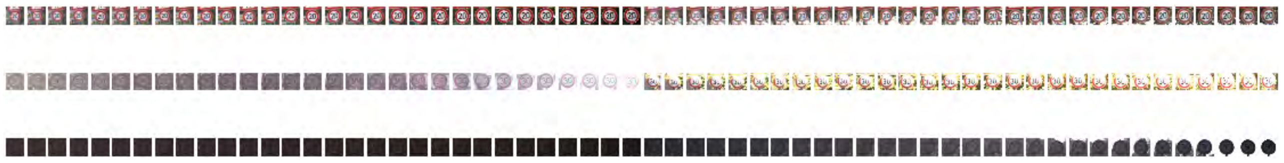

FIGURE 3. The consecutive images of three lasses of the training set. We can see clearly that 30 consecutive images taken in one place are similar.

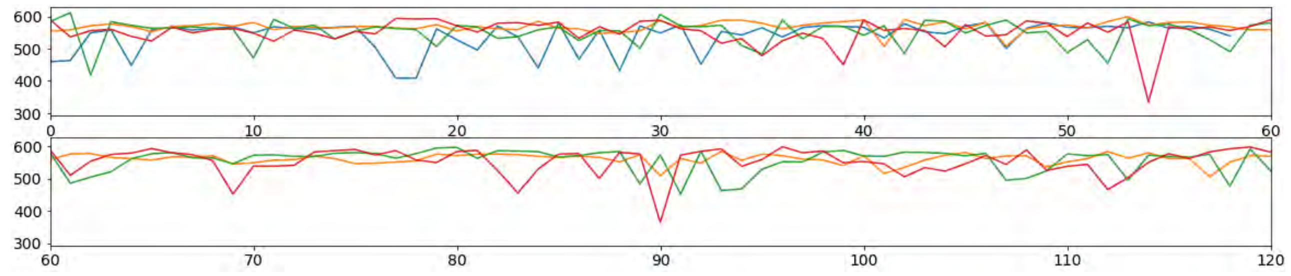

FIGURE 4. The regularity of four classes of the test set.

deep learning-based methods [6], [38]–[44] became prevalent after the emergence of AlexNet(2012). [45]–[47] directly focused on Chinese Traffic Sign Detection based on the CNN.

In general, the challenge of TSR is to ensure efficiency. Increasing the depth and complexity can improve the accuracy, but it also reduces the speed and increases the calculation. Therefore, we have built two new efficient networks called ENet and EmdNet, which balanced the two aspects (accuracy and speed) while maintaining a high generalization ability.

III. DATASETS AND SETUP

A. DATASETS TO TRAIN TSC

The German Traffic Sign Recognition benchmark database (GTSRB) includes real-time scenes captured by video cameras, and it was first provided by the International Federation of Neural Networks in 2011. The cameras continuously shot in each scene and each class included multiple scenes. Because the cameras were gradually closer to the targets, the resolution of objects was in the range of 15×15 to 250×250. The GTSRB includes 43 class and contains 39209 training set samples and 12630 test sets.

The training set distribution is shown in Appendix (FIGURE 12).

B. DATASETS TO TRAIN TSD

The dataset here is the LISA US dataset, which is the acronym for the Laboratory for Intelligent and Safe Automobiles at UCSD. The dataset consists of 7855 annotations on 6610 images. It contains 47 classes, which are listed in appendix(TABLE 6) along with the number of instances for each type.

The original image sizes of LISA vary from 640×480 to 1024×522 pixels. We have converted the input image to 400×260, and used a dynamic scaling factor based on the dimensions of the feature map relative to original image dimensions.

C. SETUP

Two Intel R Xeon(R) CPU X5650 @ 2.67 GHz with the Ubuntu 17.10 operating system is used to run this program to recognize traffic signs. The prototype is developed within the TensorFlow environment. The codes are programmed with Python 3.5, Jupyter notebook, Pickle, and OpenCV-Python. And we also run the program of TSD on a GTX 2080 with i7-8700 as a comparison. And we also run the program of TSD on a GTX 2080 with i7-8700 as a comparison. And we also run the program of TSD on a GTX 2080 with i7-8700 as a comparison.

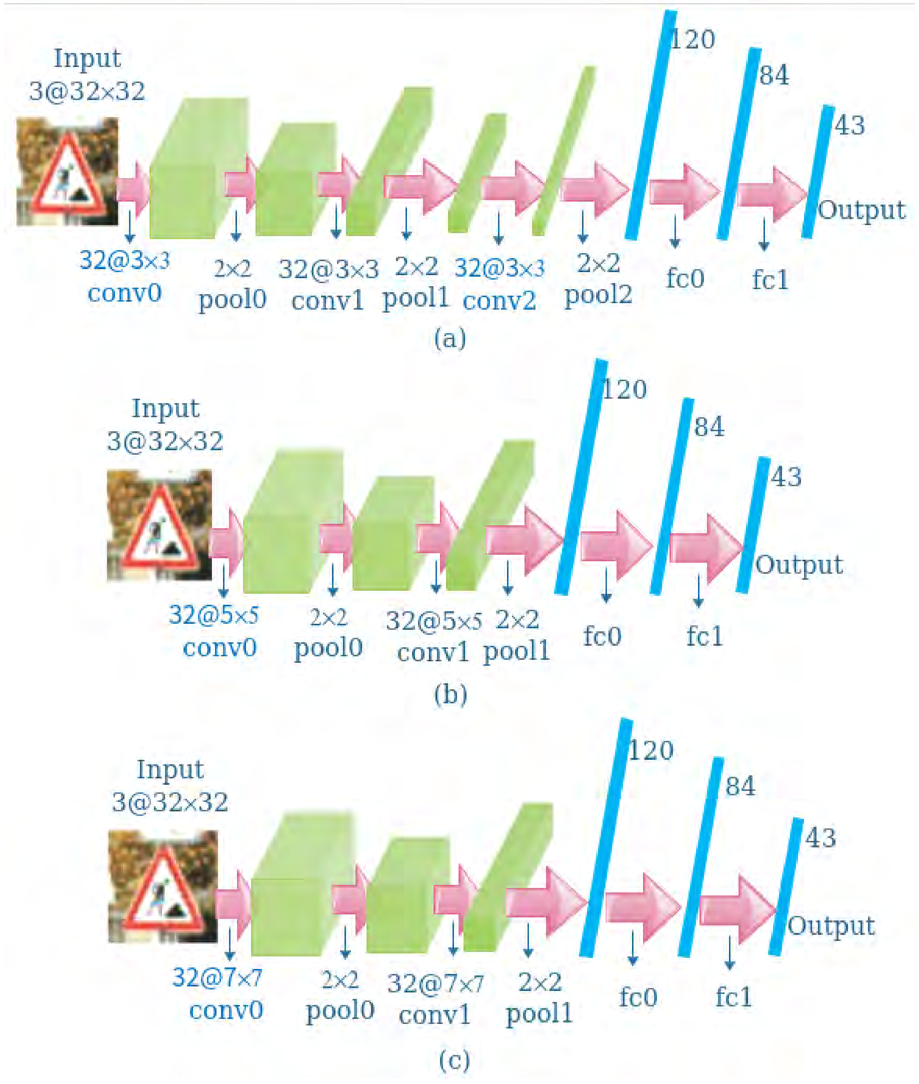

FIGURE 5. (a): 3 × 3 kernel (b): 5 × 5 kernel (c): 7 × 7 kernel.

IV. METHODOLOGY

The workflow and data flow diagram of the full text is shown in Figure 1.

A. METHODOLOGY OF TSC

1) DATA MINING

In order to improve the efficiency of the algorithm, we can analyze the characteristics of the dataset, which is also a kind of data mining. We can divide the validation set and conduct data augmentation more reasonable and effective through data mining.

a: DIVISION OF VALIDATION SET

We can analyze the characteristics of the dataset firstly. The initial 5 pictures of the preceding three classes of the training set are shown in Appendix(FIGURE 13). We found that the adjacent images of each class were similar because the dataset was taken continuously by the camera in each scene, and these adjacent photos should be taken in the same scene. Finally, we use the Euclidean Distance to find out how many photos had been taken in each scene.

We used the Euclidean Distance to represent the similarity the between two images, which is calculated as follows:

$$dist(X_i, X_{i+1}) = \sqrt{\sum\nolimits_{i=1}^{n} (X_i - X_{i+1})^2}$$

X_i is the image which has been flattened. X_{i+1} represents the image adjacent to X_i, dist(X_i, X_{i+1}) is the Euclidean Distance of the two adjacent graphs. The smaller the Euclidean Distance, the more similar the two adjacent images are.

In addition, we can find the regularity of the training set, which is shown in Figure 2, where the X-axis on the graph represents each image and the Y-axis represents the Euclidean

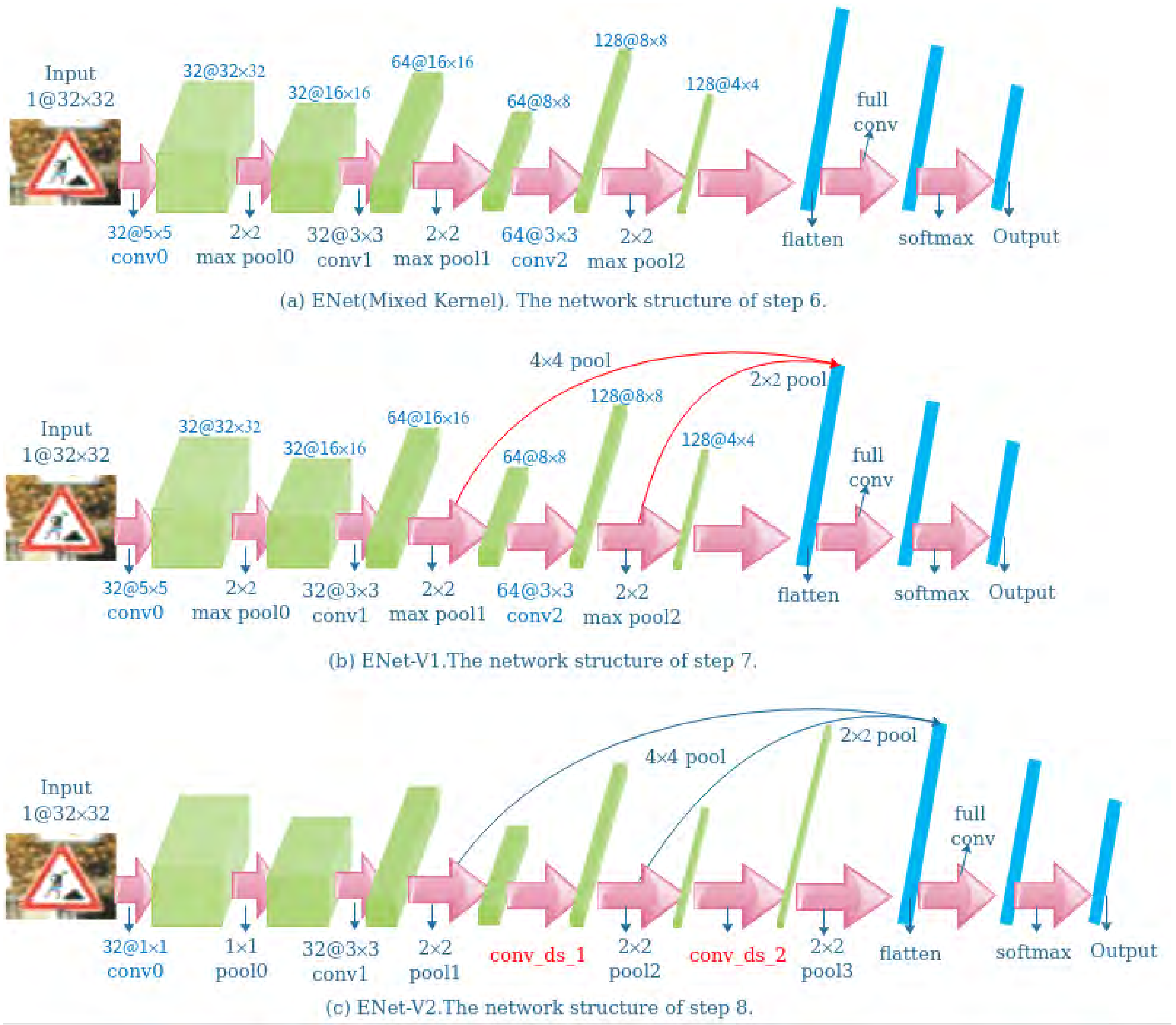

FIGURE 6. (a) ENet (Mixed Kernel). The network structure of step 6. The mixed Kernel reduced the loss and improved the accuracy, although the prediction time increased. (b) ENet-V1.The network structure of step 7. Shortcut reduced the loss and the improved accuracy, although the prediction time increased. This structure is the most accurate model, which is called ENet-V1. (c)ENet-V2.The network structure of step 8. The depthwise separable convolutions obviously reduced the prediction time, although the loss increased and the accuracy was reduced. This structure, which is called ENet-V2, is the most efficient model.

Distance of the two adjacent graphs of four classes. The lower the curve, the more similar the adjacent images are.

We find that the curve changes considerably every 30 images, which means a camera continuously took 30 images in the same scene. And because of the similar lighting and other conditions, the images taken in one scene are similar. Just as is shown in Figure 3, We can see clearly that 30 consecutive images taken in one place are similar.

In this paper, we first shuffled training set. Then the validation set was randomly selected in 0.2 ratio from training set, which called "SHUFFLE" operation.

In contrast, as is shown in Figure 4, we find that Y index of the testing set is much higher than training set' s on average, which means the consecutive images of the testing set are very different.

b: DATA AUGMENTATION

We can also analyze the characteristics of the dataset firstly, to conduct data augmentation more effective.

As is shown in appendix(Figure 12), the distribution of the training set is not uniform and traffic signs of some classes are too few, which will influence the recognition ability of the network. In this paper, we will generate two datasets called the Equaled Set and the Enlarged Set through data augmentation.

Equaled Set: This set is balanced across classes using data augmentation. Each class has 30,000 samples, which contain the original training set.

Enlarged Set: This set which is expanded through data augmentation contains 30× more data than the original one. The data augmentation technologies we used

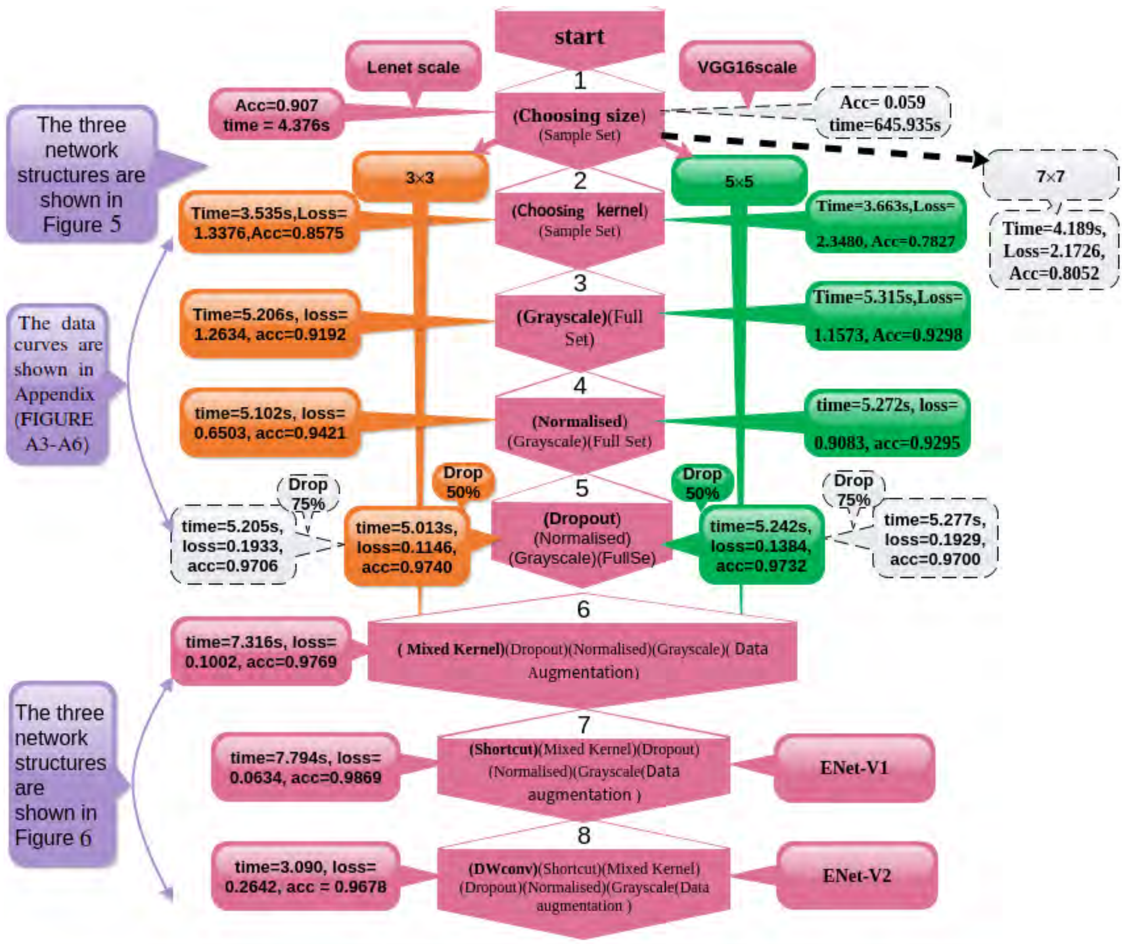
start
Lenet scale
VGG16scale
1
Acc=0.907 time = 4.376s
(Choosing size) (Sample Set)
Acc= 0.059 time=645.935s
The three network structures are shown in Figure 5
3×3
2
5×5
7×7
Time=3.535s,Loss= 1.3376,Acc=0.8575
(Choosing kernel) (Sample Set)
Time=3.663s,Loss= 2.3480, Acc=0.7827
Time=4.189s, Loss=2.1726, Acc=0.8052
3
Time=5.206s, loss= 1.2634, acc=0.9192
(Grayscale)(Full Set)
Time=5.315s,Loss= 1.1573, Acc=0.9298
The data curves are shown in Appendix (FIGURE A3-A6)
4
time=5.102s, loss= 0.6503, acc=0.9421
(Normalised) (Grayscale)(Full Set)
time=5.272s, loss= 0.9083, acc=0.9295
Drop 75%
Drop 50%
5
Drop 50%
Drop 75%
time=5.205s, loss=0.1933, acc=0.9706
time=5.013s, loss=0.1146, acc=0.9740
(Dropout) (Normalised) (Grayscale)(FullSe)
time=5.242s, loss=0.1384, acc=0.9732
time=5.277s, loss=0.1929, acc=0.9700
6
time=7.316s, loss= 0.1002, acc=0.9769
(Mixed Kernel)(Dropout)(Normalised)(Grayscale)(Data Augmentation)
The three network structures are shown in Figure 6
7
time=7.794s, loss= 0.0634, acc=0.9869
(Shortcut)(Mixed Kernel)(Dropout) (Normalised)(Grayscale(Data augmentation)
ENet-V1
8
time=3.090, loss= 0.2642, acc = 0.9678
(DWconv)(Shortcut)(Mixed Kernel) (Dropout)(Normalised)(Grayscale(Data augmentation)
ENet-V2

FIGURE 7. Illustration of the eight steps of building ENet. The dotted box represents an unselected option. The corresponding structures are shown in Figures 5 and 6, and the data curves are shown in appendix(FIGURE 14–17).

TABLE 1. The data of the eight steps of building ENet, The data in Bold Fonts is the best result. We can find that the 3 × 3 kernel works slightly better than 5 × 5 kernel.

Number	Optimization operation	Model 1	Model 2	Model 3
1	Choosing size	Lenet scale:**acc=0.907.time=4.376s**	VGG16 scale:acc=0.059.time=645.935s	
2	Choosing kernel	3 × 3: **time=3.535s,loss=1.3376,acc=0.8575**	5 × 5: time=3.663s,loss=2.3480,acc=0.7827	7 × 7: time=4.189s, loss=2.1726,acc=0.8052
3	Grayscale	3 × 3: time=5.206s,loss=1.2634,acc=0.9192	5 × 5: **time=5.315s,loss=1.1573,acc=0.9298**	
4	Normalised	3 × 3: **time=5.102s,loss=0.6503,acc=0.9421**	5 × 5: time=5.272s,loss=0.9083,acc=0.9295	
5	Dropout	3 × 3(Dropout75%): time=5.205s,loss=0.1933,acc=0.9706 3 × 3(Dropout50%): **time=5.013s,loss=0.1146,acc=0.9740**	5 × 5(Dropout75%): time=5.242s,loss=0.1384,acc=0.9732 5×5(Dropout50%): time=5.277s,loss=0.1929,acc=0.9700	
6	MixedKernel	**time=7.316s,loss=0.1002,acc=0.9769**		
7	Shortcut	**time=7.794s,loss=0.0634,acc=0.9869**		
8	DWconv	**time=3.090,loss=0.2642,acc = 0.9678**		

included the rotation, reflection, and horizontal/vertical flip.

2) THE CONSTRUCTION PROCESS OF ENET

We use the GTSRB database to train ENet, and adopt the softmax cross-entropy loss function using the Adam optimizer. The following gives the specific experimental processes. Figure 7 shows the eight steps to build ENet.

1. Step 1. Experimental Objective: Choosing the size of the network.

a. Method: We chose a sample set from the GTSRB to get results more efficiently. Then, we trained VGG16 and

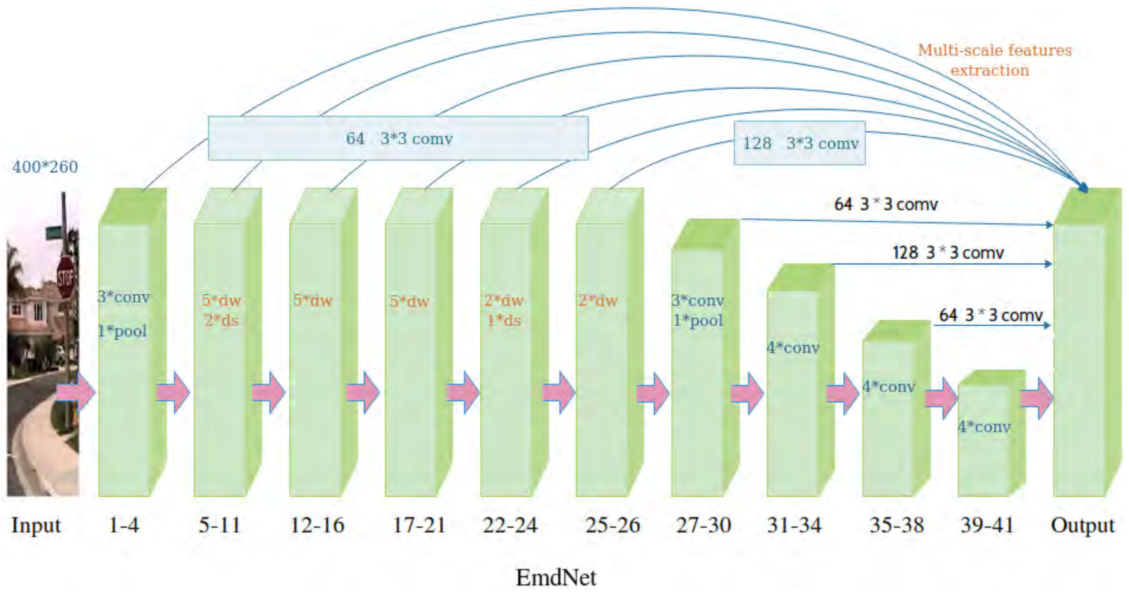

FIGURE 8. EmdNet based on the modified SDD framework, which mainly uses depth wise separable convolutions and multiscale operations.

TABLE 2. The results of the ENet-V1 and ENet-V2.

Model name	time	loss	accuracy	Number of parameters
ENet-V1	7.794s	0.0634	98.69%	0.9M
ENet-V2	3.090s	0.2642	96.78%	0.31M

TABLE 3. Numbers of parameters and top-1 accuracy results of ENet on GTSRB. The results of several state-of-the-art traffic sign classification are provided for comparison purposes.

Model name	Top-1 accuracy (GTSRB)	Numbers of parameters
STDNN[42]	99.7%	14M
EPCNN[43]	99.7%	5.22M
HLSGD[44]	99.6%	23.2M
MCDNN[45]	99.5%	38.5M
CDNN[][45]	98.5%	1.54M
ENet-V1	98.6%	0.9M
ENet-V2	96.8%	0.31M

TABLE 4. The parameters of EmdNet and several famous models.

Framework Resolution	Model	Million Parameters
SSD	Deeplab-VGG	33.1
	Inception V2	13.7
	MobileNet-V1	6.8
	EmdNet	6.3

TABLE 5. The speed of models under different computer configurations.

Model name	Computer configuration	Inference and NMS times
EmdNet	R Xeon(R) CPU X5650 @ 2.67 GHz	5.33 fps (187.7 ms)
EmdNet	GTX 2080 with an Intel Core i7 8700	30.28 fps (33.0 ms)
SSD	Nvidia Titan X	58 fps
Faster RCNN	GPU	5 fps
HyperNet:	GPU	5 fps

LeNet, both of which were classic networks, using the sample set.

b. Data: VGG16: test accuracy = 0.059, and test time = 645.935s. LeNet: test accuracy = 0.907, and test time = 4.376s.

c. Conclusion: We could find that LeNet was faster and more accurate than VGG16 in this case. It was more appropriate to choose a network similar to LeNet.

2. Step 2. Experimental Objective: Choosing the size of the convolution kernel by comparing the 3×3, 5×5, and 7×7 kernels.

a. Method: We used only one of three kernels to build the network, and used the color sample set for training. The corresponding network structures are presented in Figure 5.

Based on the conclusion of the first step, we chose three networks of similar size to LeNet. In order to ensure that the computations of networks are similar, the larger the kernel, the fewer the network layers. Thererfore, the network of 3×3 kernel has 8 layers, the network of 5×5 kernel and 7×7 has 6 layers.

b. Data: The 3×3 kernel: time = 3.535s, loss = 1.3376, and accuracy = 0.8575; 5×5 kernel: time = 3.663s, loss = 2.3480, and accuracy = 0.7827; and the 7×7 kernel: time = 4.189s, loss = 2.1726, and accuracy = 0.8052. The curves are shown in Appendix(FIGURE 14).

c. Conclusion: The 7×7 kernel produced the worst result, and the 3×3 model was a little better than the 5×5 model. We run the 3×3 and 5×5 models side by side in the 3 to 5 steps with the networks shown in Figure 5.

3. Step 3. Experimental Objective: Judging whether Grayscale has the function of optimization.

a. Method: The 3×3 and 5×5 models were respectively trained using the Full Set to compare the results before and

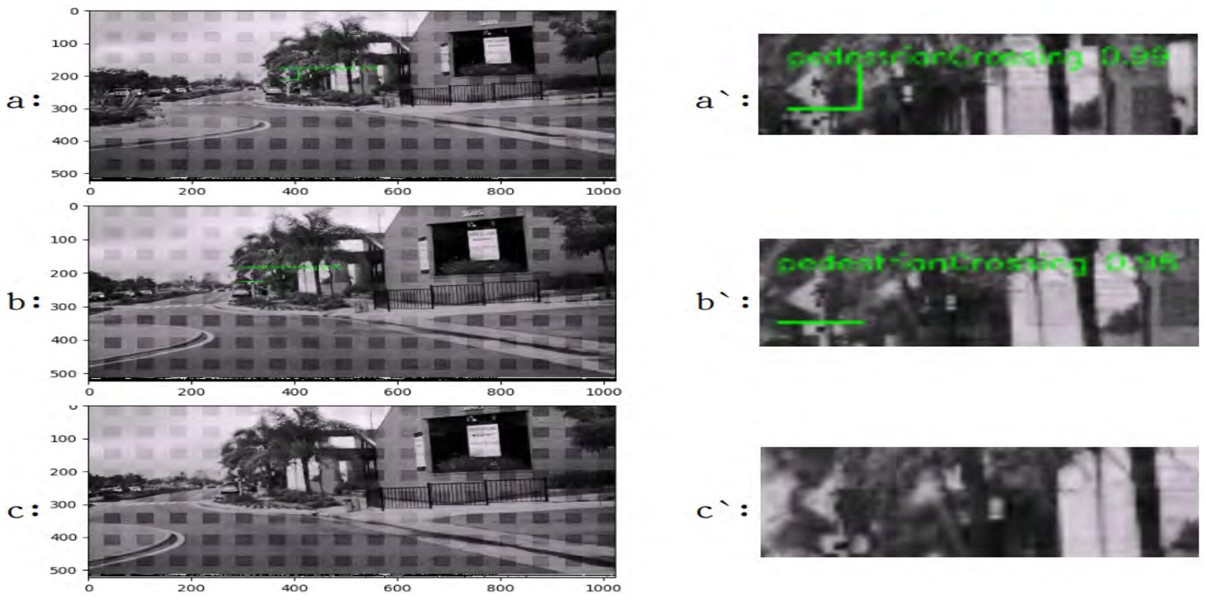

FIGURE 9. a, b, and c taken continuously on the same camera, are the results of detection with partial occlusion and different distances, and a', b', and c' are the amplifications of traffic signs (pedestrian crossing).

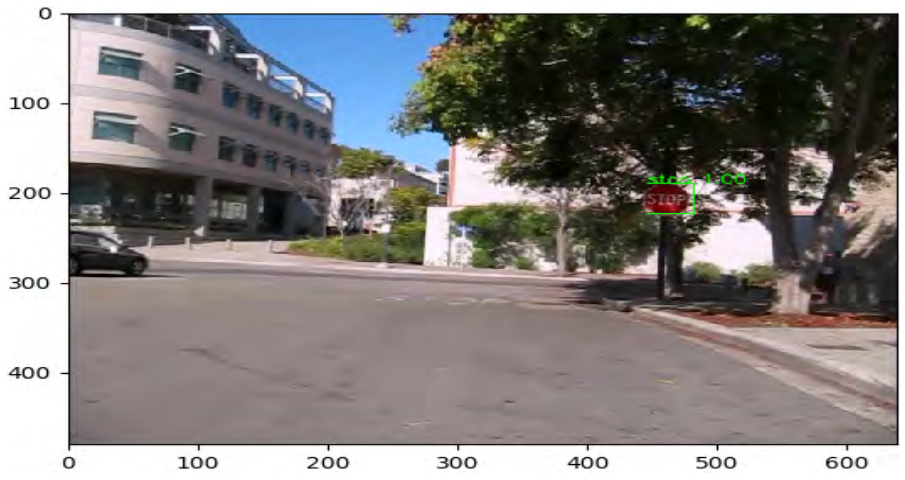

FIGURE 10. This is the result of detection with illumination changes.

after grayscaling. The Full Set includes all the original data of the GTSRB. We used the Full Set instead of the Sample Set to improve the generalization ability.

b. Data: The 3 × 3 model before grayscaling: time = 5.537s, loss = 1.6287 and accuracy = 0.8789; and after grayscaling: time = 5.206s, loss = 1.2634 and accuracy = 0.9192. The 5 × 5 model before grayscaling: time = 6.365s, loss = 2.0055 and accuracy = 0.8849; and after grayscaling: time = 5.315s, loss = 1.1573,and accuracy = 0.9298. The curves are shown in Appendix(FIGURE 15).

c. Conclusion: The two models obtained better results after grayscaling. The time and loss decreased, and the accuracy improved. Once again, the 3 × 3 model was more accurate than the 5 × 5 one. We could see that both models' validation loss increased as the training accuracy improved. This strongly implied that overfitting was occurring.

4. Step 4. Experimental Objective: Judging whether normalization has the function of optimization.

a. Method: The 3 × 3 and 5 × 5 models were respectively trained using the Full Set to compare the results before and after normalization.

b. Data: The 3 × 3 model after normalization: time = 5.102s, loss = 0.6503 and accuracy = 0.9421; and the 5 × 5 model after normalization: time = 5.272s, loss = 0.9083 and accuracy = 0.9295. The curves are shown in appendix(FIGURE 16).

c. Conclusion: The 3 × 3 model had more erratic behavior, which was hard to explain at this stage. Overall, it performed slightly better. Sadly, none of the two models reached 95% test accuracy. Next, we focus on the dropout.

5. Step 5. Experimental Objective: Judging whether Dropout has the function of optimization.

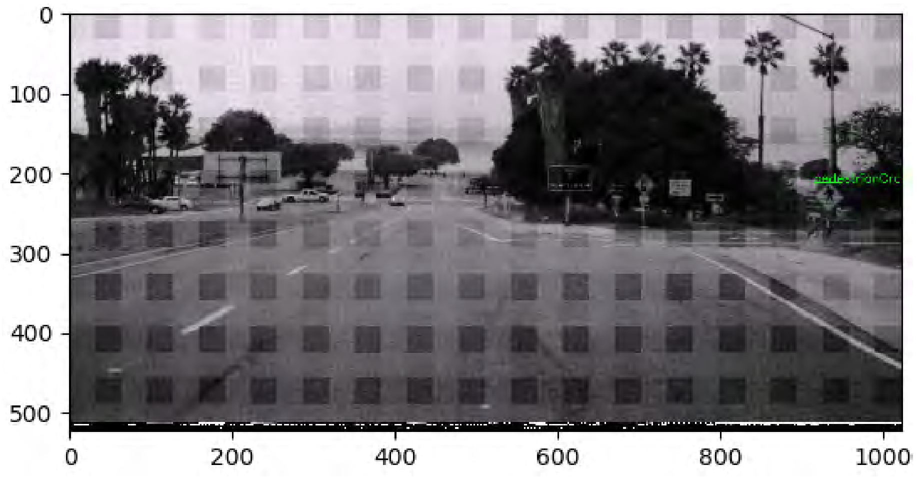

FIGURE 11. This is the result of detection with cluttered background.

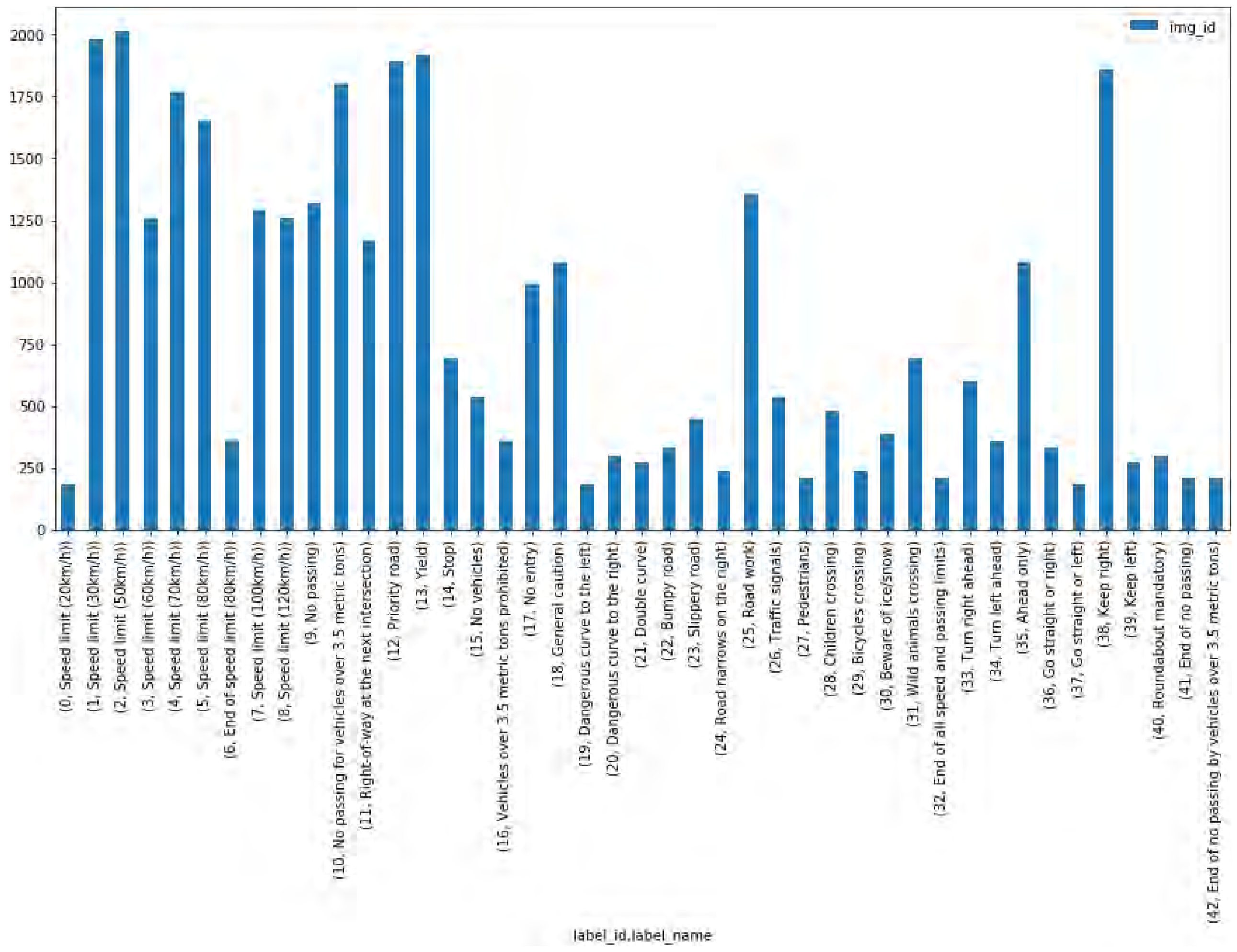

FIGURE 12. The distribution of the training data set.

a. Method: The 3 × 3 and 5 × 5 models were respectively trained using the Full Set to compare the results before and after Dropout.

b. Data(1): The dropout value was 0.9 for the convolution layer and 0.75 for the fully connected layer (p-conv = 0.9, p-fc = 0.75). The 3 × 3 model: time = 5.205s, loss = 0.1933

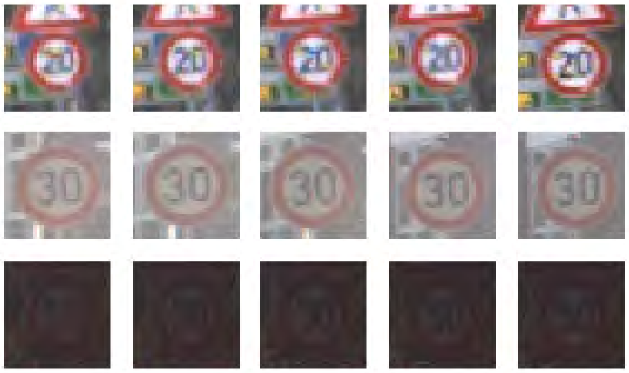

FIGURE 13. The initial 5 pictures of the preceding three classes of the training set.

and accuracy = 0.9706; and the 5 × 5 model: time = 5.277*s*, loss = 0.1929 and accuracy = 0.9700..

b. Data(2): p-conv = 0.9, and p-fc = 0.50. The 3×3 model: time = 5.013*s*, loss = 0.1146, accuracy = 0.9740; and the 5 × 5 model: time = 5.242*s*, loss = 0.1384, accuracy = 0.9732.The curves are shown in appendix(FIGURE 17)

c. Conclusion (1): Once again, both the 3×3 and 5×5 variants were very close. Interestingly, the 3 × 3 model achieved lower loss but also lower accuracy than its counterpart 3 × 3 model on color images. In the future, we should do more runs of those models to determine which one performs better over the long run. Nevertheless, we believe that we did not need to go that far. We could use even more aggressive dropout values to obtain better results in the next step.

c. Conclusion (2): Both models resulted in smooth, satisfactory curves. The 3 × 3 model clearly seemed to perform the best. While we were able to reach above 98% accuracy on the validation set, we had not been able to break through this barrier yet on the test set. Furthermore, we could find that the loss obviously decreased after dropout.

6. Step 6. Experimental Objective: Judging whether Mixed Kernel and Data Augmentation have the function of optimization.

a. Method(1): We built a network containing 3 × 3 and 5 × 5 kernels, as is shown in Figure 6(a). We also used Data Enhancement, as described in section IV-A-1.

b. Data: time = 7.316*s*, loss = 0.1002 and accuracy = 0.9769.

c. Conclusion: The Mixed Kernel reduced the loss and improved the accuracy, although the prediction time increased.

7. Step 7. Experimental Objective: Judging whether Shortcut has the function of optimization.

a. Method: We added three shortcuts from each convolution layer to the fully connected layer, which is illustrated in Figure 6(b).

b. Data: time = 7.794*s*, loss = 0.0634 and accuracy = 0.9869.

c. Conclusion: Obviously, Shortcut reduced the loss and improved the accuracy, although the prediction time was increased. This structure was the most accurate model, which was called ENet-V1.

8. Step 8. Experimental Objective: Judging whether Depthwise Separable Convolution has the function of optimization.

a. Method: We added Depthwise Separable Convolutions to the network structure, which is shown in Figure 6(c). Depthwise Separable Convolution already has had an existing function in TensorFlow.

b. Data: time = 3.090*s*, loss = 0.2642, and accuracy = 0.9678.

c. Conclusion: Obviously, the Depthwise Separable Convolutions reduced the prediction time, although the loss increased and the accuracy decreased. In addition, this structure was the fastest model, which was called ENet-V2. ENet-V1 and ENet-V2, which are efficient networks, can be selected according to our different needs.

Figure 7 shows the eight steps to building ENet. The first and second steps are implemented using the Sample Set, which was part of the raw data, because reducing quantity can save training time. The third to fifth steps were implemented on the Full Set, which was all of the raw data. The sixth to eighth steps are implemented on the Equaled Set and the

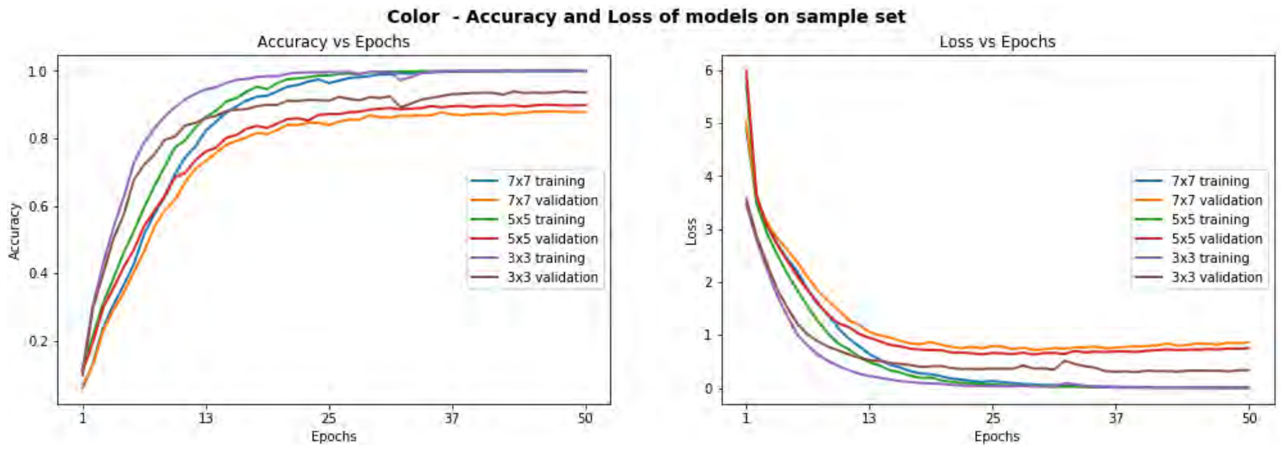

FIGURE 14. Data of step 2.

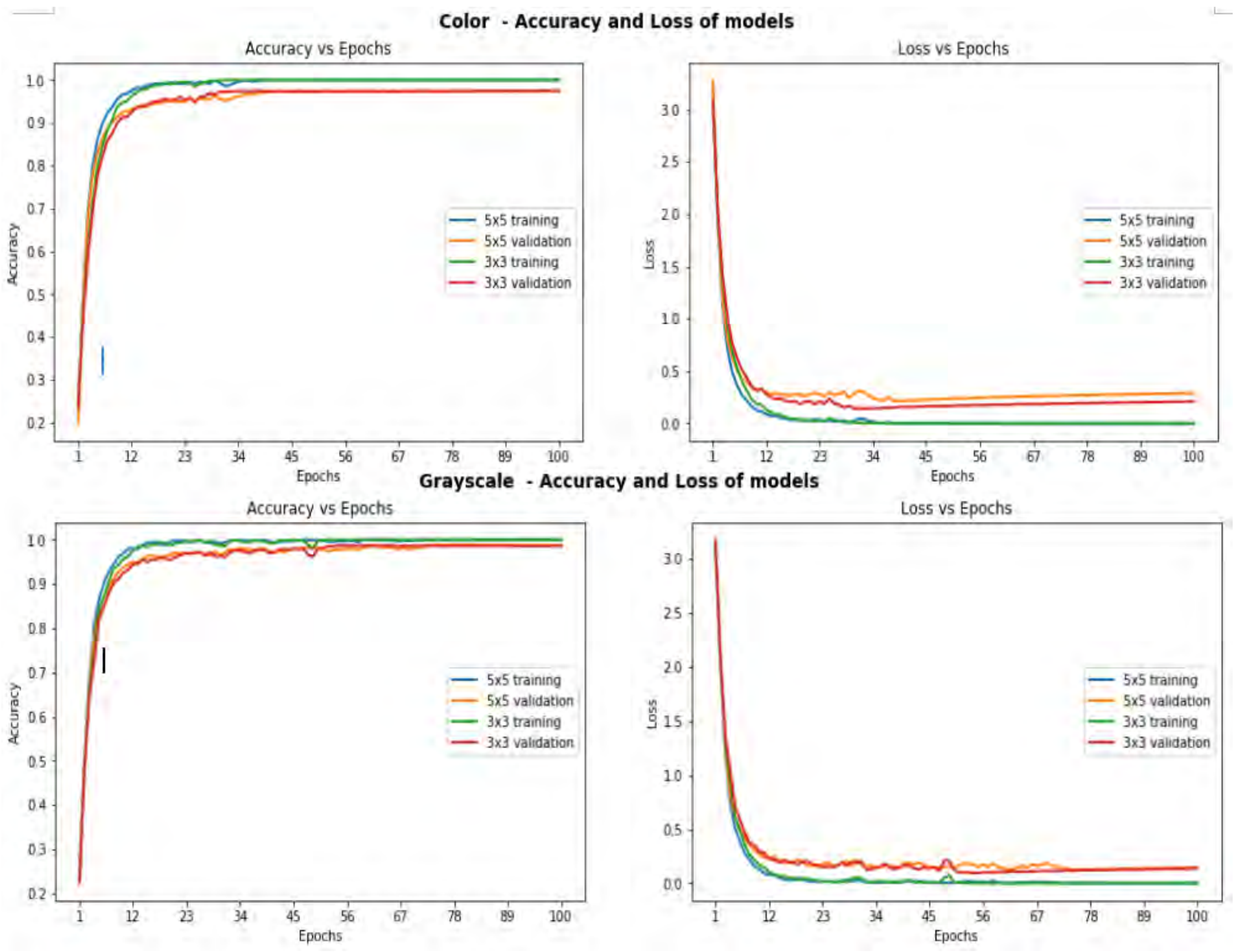

FIGURE 15. Data of step 3.

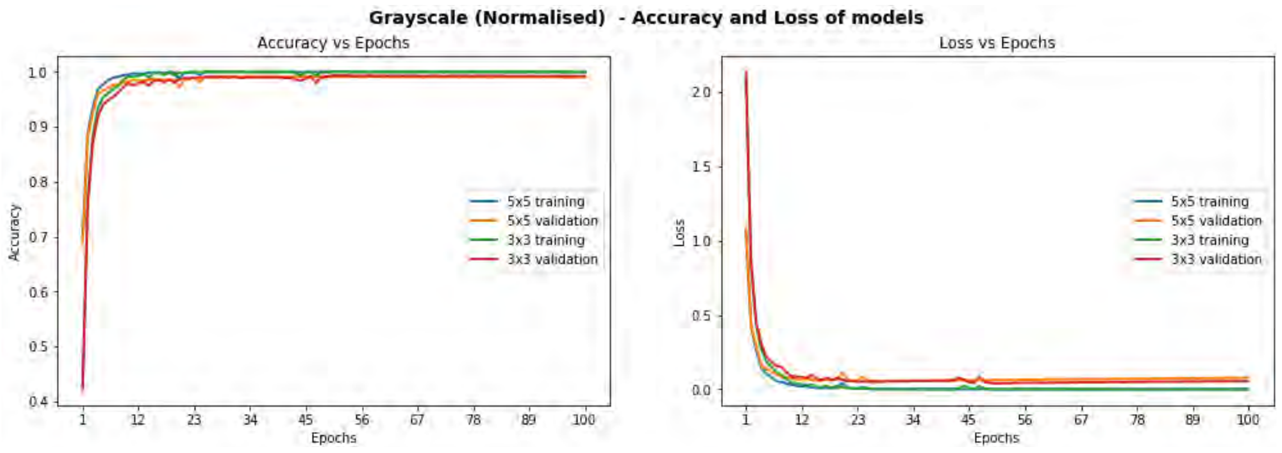

FIGURE 16. Data of step 4.

Enlarged Set, which were the enhancements of the original data(As mentioned in the previous section).

The corresponding network structures are shown in Figures 5 and 6, and the experimental data is shown in Figure 7 and Table 1. In order to ensure that the computations of networks are similar, the networks of step 6 and step 7 have 8 layers, and the network of step 8 has 10 layers.

We designed six networks for comparison, to select the best. At last, we can get two networks called ENet-V1 and ENet-V2. ENet-V1 is more accurate than ENet-V2,

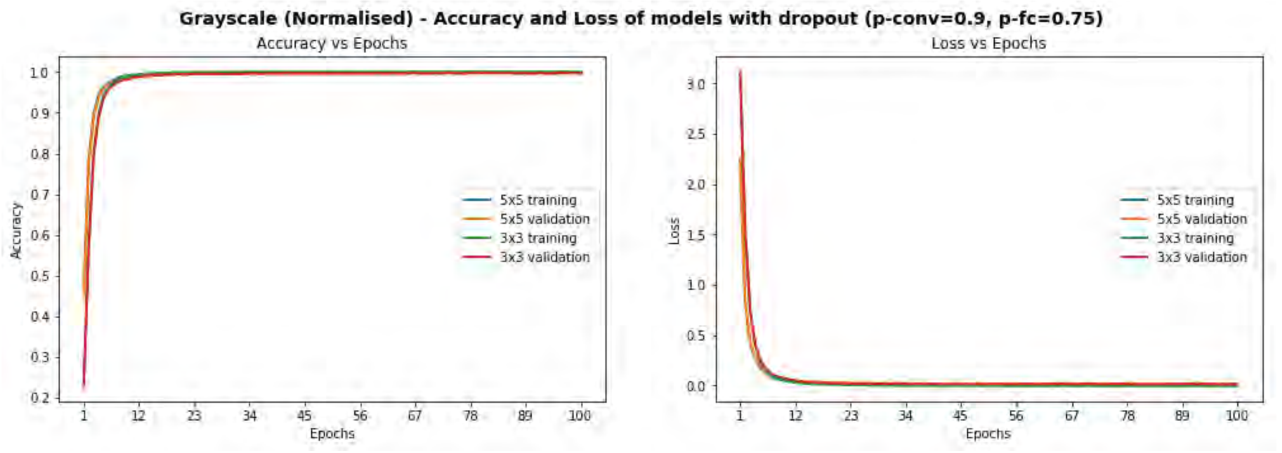

FIGURE 17. Data of step 5.

TABLE 6. The content of the LISA traffic Sign dataset broken down by sign type.

Number	Traffic Sign	Number	Traffic Sign
294	addedLane	34	slow
37	curveLeft	11	speedLimit15
50	curveRight	349	speedLimit25
35	dip	140	speedLimit30
23	doNotEnter	538	speedLimit35
9	doNotPass	73	speedLimit40
2	intersection	141	speedLimit45
331	keepRight	48	speedLimit50
210	laneEnds	2	speedLimit55
266	merge	74	speedLimit65
47	noLeftTurn	132	speedLimitUrdbl
26	noRightTurn	1821	stop
1085	pedestrianCrossing	168	stopAhead
11	rampSpeedAdvisory	20 5	thruMergeLeft
5	rampSpeedAdvisory	35 7	thruMergeRight
3	rampSpeedAdvisory	40 19	thruTrafficMergeLeft
29	rampSpeedAdvisory	45 60	truckSpeedLimit55
16	rampSpeedAdvisory	50 32	turnLeft
3	rampSpeedAdvisoryUrdbl	92	turnRight
77	rightLaneMustTurn	236	yield
53	roundabout	57	yieldAhead
133	school	21	zoneAhead25
105	schoolSpeedLimit	25 20	zoneAhead45
925	signalAhead	In total:	7855 sign annotations

and ENet-V2 is faster than ENet-V1. Both of them are efficient networks for mobile devices, which can be selected according to the different application requirements.

B. METHODOLOGY OF TSD

1) DESIGN PRINCIPLE OF EMDNET

As mentioned in the section IV-A-2(step 2, etc.), we should give priority to the 3×3 kernel instead of the 5×5 and 7×7 kernels, which should be placed in shallow layers to extract the features through a larger window.

As mentioned in section IV-A-2(step 7), Shortcut reduced the loss and the improved accuracy, although the prediction time increased. EmdNet has 9 shortcuts, which is called the multiscale operation. In addition, as mentioned in section IV-A-2 (step 8), the depthwise separable convolutions obviously reduced the prediction time, although the loss increased and the accuracy decreased. Thus, EmdNet mainly combines depthwise separable convolutions and multiscale operations, which has made an appropriate trade-off between the accuracy and speed.

Furthermore, as mentioned in sections I and II, the one-stage method, which is suitable for mobile devices such as self-driving cars, is faster and needs fewer resources. So the EmdNet based on the modified SSD framework is the one-stage method.

2) THE STRUCTURE OF EMDNET

We use the LISA US Traffic Sign Dataset (LISA) to train EmdNet, which uses a revised end-to-end technology for real-time embedded Traffic Sign Detection. EmdNet, which is based on the modified SSD framework, mainly uses depth wise separable convolutions and multiscale operations.

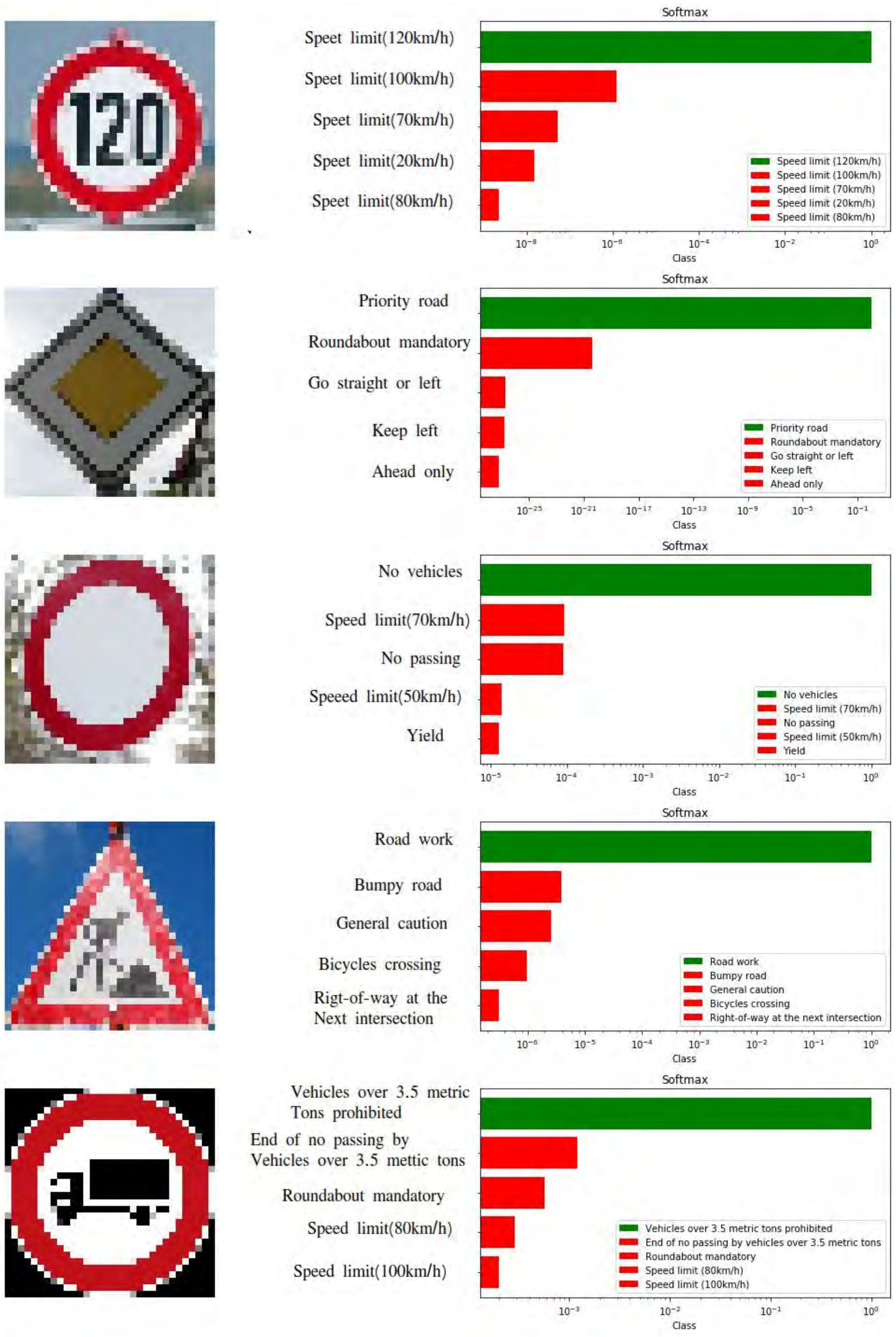

FIGURE 18. Five new german traffic signs taken from the network to test the recognition effect of ENet.

EmdNet has 42 levels including the output layer, which are presented in Figure 8. The parameters and configuration of EmdNet are shown in Appendix(TABLE 7).

The 5th to 26th levels of EmdNet adopt the depth wise separable convolution is backbone network, to extract features. And the depth wise separable convolutions can improve

TABLE 7. The parameters and configurations of EmdNet. EmdNet has 42 layer including the output layer. The 5st to 26th levels of EmdNet mainly adopt the depth wise separable convolution is backbone network, to extract features. And the depth wise separable convolutions can improve the network efficiency. EmdNet has 9 shortcuts, which are called multiscale operations, to produce the high-level feature representations that are amenable to classification early on. This construction principle is similar to ENet.

Layer Name	Operation	Configuration	Weights	Biases	Parameters
Input	Input Image	400x260	0	0	0
1	conv-1	64, [7, 7], 2	3136	64	3,200
2	Pool-1	net, [3, 3], 1	0	0	
3	conv-2	128, [5, 5],2	204,800	128	204,928
4	conv-3	64, [3, 3], 2	73,728	64	73,792
4Hook	Hook(net, ’conv-3’)	2(64, [3, 3])	73,728	128	73,856
5-8	Conv-ds-(1-4)	128,1	70144	512	70,656
9	Conv-ds-5	256, 1	33,920	256	34,176
10	Conv-ds-6	256, 1	33,920	256	34,176
11	Conv-ds-7	64, 1	18,688	64	18,752
11Hook	Conv-ds- Hook(net,’conv-ds-7’)	2(64, [3, 3])	73,728	128	73,856
12-16	Conv-ds- (8-12)	512, 1	1371136	2,624	1,373,760
16Hook	Conv-ds- Hook(net,’conv-ds-12’)	2(64, [3, 3])	73,728	128	73,856
17-21	Conv-ds- (13-17)	512, 1	1,371,136	2,624	1,373,760
21Hook	Conv-ds- Hook(net, ’Conv-ds-17’)	2(64, [3, 3])	73,728	128	73,856
22	Conv-ds- 18	512, 1	266,752	512	267,264
23	Conv-ds- 19	1024, 1	528,896	1024	529,920
24	Conv-ds- 20	64, 1	74,752	64	74,816
24Hook	Conv-ds- Hook(net,’conv-ds-20’)	2(64, [3, 3])	73,728	128	73,856
25	Conv-ds- 21	512, 1	533,504	512	534,016
26	Conv-ds- 22	128, 1	70,144	128	70,272
26Hook	Conv-ds- Hook(net,’conv-ds-22’)	2(128, [3, 3])	147,456	256	147,712
27	conv-4	64, [3, 3], 1	294,912	64	294,976
28	pool-2	[3, 3], 1	0	64	0
29	conv-5	64, [1, 1],1	4,096	64	4,160
30	conv-6	64, [3, 3], 1	36,864	128	36,928
30Hook	Hook(net,’conv-6)	2(64, [3, 3])	73,728	64	73,856
31	conv-7	64, [1, 1], 1	4,096	64	4,160
32	conv-8	64, [3, 3], 1	36,864	64	36,928
33	conv-9	64, [3, 3], 1	36,864	128	36,928
34	conv-10	128, [3, 3], 2	73,728	256	73,856
34Hook	Hook(net,’conv-10’)	2(128, [3, 3])	294,912	64	295,168
35	conv-11	64, [1, 1]	8,192	64	8,256
36	conv-12	64, [3, 3],1	36,864	64	36,928
37	conv-13	64, [3, 3],1	36,864	64	36,928
38	conv-14	64, [3, 3],1	36,864	128	36,928
38Hook	Hook(net, ’conv-14’)	2(64, [3, 3])	73,728	64	73,856
39	conv-15	64, [3, 3],2	36,864	64	36,928
40	conv-16	64, [3, 3],2	36,864	32	36,928
41	conv-17	32, [3, 3],1	18,432	128	18,464
Output					Total: 6,322,656

the network efficiency. EmdNet has 9 shortcuts, which are called multiscale operations, to produce the high-level feature representations that are amenable to classification early on. This construction principle is similar to ENet.

V. DISCUSSIONS AND EVALUATION

A. DISCUSSIONS AND EVALUATION OF TSC

1) DISCUSSION OF ENET

As is shown in Table 2, ENet-V1 is more accurate, although it is slower and larger. The test set has 12630 samples, which means that ENet cannot identify 166 samples. ENet spent 7.794S identifying the test set, which means that every sample took on average 0.62 ms. ENet-V2 is less accurate, but is faster and smaller. ENet-V2’ s size is only 1/3 of ENet-V1. And each sample took on average 0.24 ms.

Table 3 shows the numbers of parameters and Top-1 accuracy results of ENet on the GTSRB. The results of several famous traffic sign classifications are provided for comparison purposes. The resulting ENet-V1 possesses 0.9M parameters, which is 1/15 the parameters of the start-of-the-art method.

2) TESTING USING NEW DATASET

The generalization ability of the optimized ENet is obviously improved. In this section, we will collect five new German traffic signs from the network to test the recognition effect of ENet. The results show that all new data can be identified, which means that ENet has a strong generalization ability. This is shown in Appendix(FIGURE 18).

B. DISCUSSIONS OF TSD

1) THE RESULT OF EMDNET

As is shown in Table 4, EmdNet possesses 6.3M parameters, which is 1/5 the parameters of Deeplab-VGG, 1/22 the parameters of Inception V2, and slightly smaller than MobileNet, all with the SSD framework.

On two Intel R Xeon(R) CPU X5650 @ 2.67 GHz, the neural network inference time of EmdNet is about 130.9ms(7.64 fps), the sum of inference and Non-Maximum Suppression(NMS) times is 187.7 ms (5.33 fps). And on GTX 2080 with an Intel Core i7 8700, the neural network inference time is about 15.4ms(64.82fps), the sum of inference and Non-Maximum Suppression(NMS) times is 33.0 ms (30.28 fps), which is a better performance. As is shown in Table 5, we find that the one-stage method used GPU can improve the speed, and EmdNet can achieve real-time detection.

The detection efficiency of EndNet on LISA is shown in Figure 9 to 11. We can find the following aspects.

1. Our network has enough discrimination ability to detect when under partial occlusion (as is shown in Figures 9(a and b), except for c).

2. The traffic sign in Figure 9(a) is closer than the one in Figure 9(b), but the former' s confidence is higher. So the Object Confidence is not only determined by the distance or size of the target, as is shown in Figures 9(a' and b'). Sometimes a large target is difficult to find, which also proves the importance of multiscale feature extraction.

3. EmdNet can identify objects well, regardless of whether they are colored or not, as is shown in Figure 10.

4. Our network has enough discrimination ability to detect when under illumination changes(as is shown in Figure 10) and cluttered background(as is shown in Figure 11).

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we introduce a TSC network called ENet and a TSD network called EmdNet. We show the experimental process of building ENet's structure. The resulting ENet possesses 0.9M parameters (1/15 the parameters of the start-of-the-art method) while still achieving an accuracy of 98.6% on the GTSRB. Furthermore, we design EmdNet's backbone network according the principle of the ENet. The resulting EmdNet, which is similar to MobileNet with the SDD Framework, possesses 6.3M parameters.These experimental results show that an efficient neural network architecture, which has adequate accuracy, generalization, and speed, can be designed for real-time embedded traffic sign recognition.

In the future, our works will mainly include three objectives. One is to improve the performance, especially for TSD. The second is to use videos instead of images as the input of the network for further practical validation. The third is to research multitask learning and improve the generalization ability and commercialization of the network.

APPENDIX

See FIGURES 12–18 and TABLES 6 and 7.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 120, Jan. 2012, pp. 1097–1105. doi: 10.1145/3065386.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. ED-21, no. 3, pp. 51–56, Jan. 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, vol. 53, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778. doi: 10.1109/cvpr.2016.90.

[5] A. Arcos-García, J. Álvarez-García, and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Netw.*, vol. 99, pp. 158–165, Mar. 2018. doi: 10.1016/j.neunet.2018.01.005.

[6] H. H. Aghdam, E. J. Heravi, and D. Puig, "A practical approach for detection and classification of traffic signs using convolutional neural networks," *Robot. Auton. Syst.*, vol. 84, pp. 97–112, Oct. 2016. doi: 10.1016/j.robot.2016.07.003.

[7] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 1991–2000, Oct. 2014. doi: 10.1109/tits.2014.2308281.

[8] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Netw.*, vol. 32, pp. 333–338, Aug. 2012. doi: 10.1016/j.neunet.2012.02.023.

[9] A. G. Howard *et al.* (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. (2018). "MobileNetV2: Inverted residuals and linear bottlenecks." [Online]. Available: https://arxiv.org/abs/1801.04381

[11] X. Zhang, X. Zhou, M. Lin, and J. Sun. (2017). "ShuffleNet: An extremely efficient convolutional neural network for mobile devices." [Online]. Available: https://arxiv.org/abs/1707.01083

[12] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. (Jul. 2018). "ShuffleNet V2: Practical guidelines for efficient CNN architecture design." [Online]. Available: https://arxiv.org/abs/1807.11164

[13] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. (Nov. 2017). "Condensenet: An efficient densenet using learned group convolutions." [Online]. Available: https://arxiv.org/abs/1711.09224

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. (2015). "Rethinking the inception architecture for computer vision." [Online]. Available: https://arxiv.org/abs/1512.00567. doi: 10.1109/cvpr.2016.308.

[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. (2016). "Inception-v4, inception-ResNet and the impact of residual connections on learning." [Online]. Available: https://arxiv.org/abs/1602.07261

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[18] F. Chollet. (2016). "Xception: Deep learning with depthwise separable convolutions." [Online]. Available: https://arxiv.org/abs/1610.02357

[19] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. (Sep. 2017). "Squeeze-and-excitation networks." [Online]. Available: https://arxiv.org/abs/1709.01507

[20] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, vol. 14, Oct. 2017, pp. 3–7. doi: 10.1109/CVPR.2017.243.

[21] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf.*, vol. 10, Jul. 2017, pp. 5987–5995. doi: 10.1109/cvpr.2017.634.

[22] C. Liu *et al.* (Dec. 2017). "Progressive neural architecture search." [Online]. Available: https://arxiv.org/abs/1712.00559

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. (Jul. 2017). "Learning transferable architectures for scalable image recognition." [Online]. Available: https://arxiv.org/abs/1707.07012

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. (Feb. 2018). "Regularized evolution for image classifier architecture search." [Online]. Available: https://arxiv.org/abs/1802.01548

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, Feb. 2014, pp. 580–587.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, Jan. 2015, pp. 91–99. doi: 10.1109/tpami.2016.2577031.
[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, Oct. 2017, pp. 2980–2988. doi: 10.1109/iccv.2017.322.
[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. CVPR*, Jun. 2016, pp. 779–788. doi: 10.1109/cvpr.2016.91.
[29] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. CVPR*, Jul. 2017, pp. 6517–6525. doi: 10.1109/cvpr.2017.690.
[30] K. He, X. Zhang, S. Ren, and J. Sun. (Jun. 2014). "Spatial pyramid pooling in deep convolutional networks for visual recognition." [Online]. Available: https://arxiv.org/abs/1406.4729. doi: 10.1007/978-3-319-10578-9_23.
[31] R. Girshick, "Fast R-CNN," *Proc. ICCV*, Dec. 2015, pp. 1440–1448. doi: 10.1109/iccv.2015.169.
[32] J. Redmon and A. Farhadi. (Apr. 2018). "YOLOv3: An incremental improvement." [Online]. Available: https://arxiv.org/abs/1804.02767
[33] W. Liu *et al.*, "SSD: Single shot multii box detector," in *Proc. ECCV*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.
[34] A. Wong, M. J. Shafiee, F. Li, and B. Chwyl. (Feb. 2018). "Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection." [Online]. Available: https://arxiv.org/abs/1802.06488
[35] A. de la Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, "Road traffic sign detection and classification," *IEEE Trans. Ind. Electron.*, vol. 44, no. 6, pp. 848–859, Dec. 1997. doi: 10.1109/41.649946.
[36] M. Benallal and J. Meunier, "Real-time color segmentation of road signs," in *Proc. CCECE*, May 2003, pp. 1823–1826. doi: 10.1109/ccece.2003.1226265.
[37] A. Ruta, Y. Li, and X. Liu, "Real-time traffic sign recognition from video by class-specific discriminative features," *Pattern Recognit.*, vol. 43, pp. 416–430, Jan. 2010. doi: 10.1016/j.patcog.2009.05.018.
[38] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 87–92. doi: 10.1109/itsc.2014.6957671.
[39] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. (2017). "Perceptual generative adversarial networks for small object detection." [Online]. Available: https://arxiv.org/abs/1706.05274. doi: 10.1109/cvpr.2017.211.
[40] M. Haloi. (Nov. 2015). "Traffic sign classification using deep inception based convolutional networks." [Online]. Available: https://arxiv.org/abs/1511.02992
[41] A. Wong, M. J. Shafiee, and M. S. Jules, "MicronNet: A highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification," *IEEE Access*, vol. 6, pp. 59803–59810, 2018. doi: 10.1109/ACCESS.2018.2873948.
[42] Z. Gao, D. Y. Wang, S. H. Wan, H. Zhang, and Y. L. Wang, "Cognitive-inspired class-statistic matching with triple-constrain for camera free 3D object retrieval," *Future Gener. Comput. Syst.*, vol. 94, pp. 641–653, May 2019.
[43] S. Ding, S. Qu, Y. Xi, and S. Wan, "A long video caption generation algorithm for big video data retrieval," *Future Gener. Comput. Syst.*, vol. 93, pp. 583–595, Apr. 2019.
[44] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K.-K. R. Choo, "Multi-dimensional data indexing and range query processing via Voronoi diagram for Internet of Things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Feb. 2019.
[45] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2110–2118. doi: 10.1109/cvpr.2016.232.
[46] R. Qian, B. Zhang, Y. Yue, Z. Wang, and F. Coenen, "Robust Chinese traffic sign detection and recognition with deep convolutional neural network," in *Proc. 11th Int. Conf. Natural Comput.*, Zhangjiajie, China, Aug. 2015, pp. 791–796. doi: 10.1109/icnc.2015.7378092.
[47] C. Xiong, C. Wang, W. Ma, and Y. Shan, "A traffic sign detection algorithm based on deep convolutional neural network," in *Proc. IEEE Int. Conf. Signal Image Process.*, Beijing, China, Aug. 2016, pp. 676–679. doi: 10.1109/siprocess.2016.7888348.

XIE BANGQUAN (M'84) received the B.S. degree from the Changsha University of Science and Technology, Changsha, China, in 2007, and the M.S. degree from the South China University of Technology, Guangzhou, China, in 2013, where he is currently pursuing the Ph.D. degree with the School of Civil Engineering and Transportation.

He is also a Senior Lecturer and the author of the book *Detection and Maintenance of Electronic Control System for Automobile Body*. His research interests include perception technology of intelligent driving, machine learning, and intelligent transportation technology.

WENG XIAO XIONG received the B.S. degree from the Dalian University of Technology, Dalian, China, the M.S. degree from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree from the South China University of Technology, Guangzhou, China.

She is currently a Ph.D. Tutor with the South China University of Technology. Her research interests include intelligent transportation and intelligent driving.