Traffic Sign Detection and Recognition

1

1.1 Introduction

Assume you are driving at speed of 90 km/h in a one-way road and you are about to join a new road. Even though there was a "*danger: two way road*" sign in the junction, you have not seen the sign and you keep driving in opposite lane of the new road. This is a hazardous situation which may end up with a fatal accident because the driver assumes he or she is still driving in a two-way road. This was only a simple example in which failing to detect traffic sign may cause irreversible consequences. This danger gets even more serious with inexperienced drivers and senior drivers, specially, in unfamiliar roads.

According to National Safety Council, medically consulted motor-vehicle injuries for the first 6 months of 2015 were estimated to be about 2,254,000.[1] Also, World Health Organization reported that[2] there have been about 1,250,000 fatalities in 2015 due to car accidents. Moreover, another study shows that human error accounts solely for 57% of all accidents and it is a contributing factor in over 90% of accidents. The above example is one of the scenarios which may occur because of failing to identify traffic signs.

Furthermore, self-driving cars are going be commonly used in near future. They must also conform with the road rules in order not to endanger other users of road. Likewise, smart-cars try to assist human drivers and make driving more safe and comfortable. Advanced Driver Assistant System (ADAS) is a crucial component on these cars. One of the main tasks of this module is to recognize traffic signs. This helps a human driver to be aware of all traffic signs and have a more safe driving experience.

[1] www.nsc.org/NewsDocuments/2015/6-month-fatality-increase.pdf.
[2] www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_data/en/.

H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks*, DOI 10.1007/978-3-319-57550-6_1

1.2 Challenges

A Traffic signs recognition module is composed of two main steps including *detection* and *classification*. This is shown in Fig. 1.1. The detection stage scans image of scene in a multi-scale fashion and looks for location of traffic signs on the image. In this stage, the system usually does not distinguish one traffic sign from another. Instead, it decides whether or not a region includes a traffic sign regardless of its type. The output of detection stage is a set of regions in the image containing traffic signs. As it is shown in the figure, detection module might make mistakes and generate a few *false-positive* traffic signs. In other words, there could be a few regions in the output of detection module without any traffic sign. These outputs have been marked using a red (dashed) rectangle in the figure.

Next, classification module analyzes each region separately and determines type of each traffic sign. For example, there is one "no turning to left" sign, one "roundabout" sign, and one "give way" sign in the figure. There are also three "pedestrian crossing"

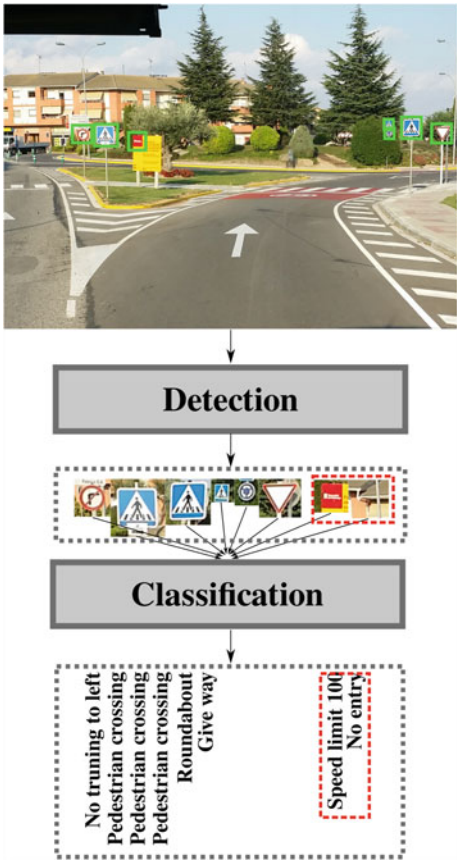

Fig. 1.1 Common pipeline for recognizing traffic signs

signs. Moreover, even though there is no traffic sign inside the false-positive regions, the classification module labels them into one of traffic sign classes. In this example, the false-positive regions have been classified as "speed limit 100" and "no entry" signs.

Dealing with false-positive regions generated by detection module is one of major challenges in developing a practical traffic sign recognition system. For instance, a self-driving car may suddenly brake in the above hypothetical example because it has detected a no entry sign. Consequently, one of the practical challenges in developing a detection module is to have zero false-positive region. Also, it has to detect all traffic signs in the image. Technically, its true-positive rate must be 100%. Satisfying these two criteria is not trivial in practical applications.

There are two major goals in designing traffic signs. First, they must be easily distinguishable from rest of objects in scene and, second, their meaning must be easily perceivable and independent from spoken language. To this end, traffic signs are designed with a simple geometrical shape such as triangle, circle, rectangle, or polygon. To be easily detectable from rest of objects, traffic signs are painted using basic colors such as red, blue, yellow, black, and white. Finally, the meaning of traffic signs is mainly carried out by pictographs in center of traffic signs. It should be noted that some signs heavily depend on text-based information. However, we can still think of the texts in traffic signs as pictographs.

Although classification of traffic signs is an easy task for a human, there are some challenges in developing an algorithm for this purpose. Some of these challenges are illustrated in Fig. 1.2. First, image of traffic signs might be captured from different perspectives. This may nonlinearly deform the shape of traffic signs.

Second, weather condition can dramatically affect the appearance of traffic signs. An example is illustrated in Fig. 1.2 where the "no stopping" sign is covered by snow. Third, traffic signs are being impaired during time and some artifacts may appear on signs which might have a negative impact on their classification. Fourth, traffic signs might be partially occluded by another signs or objects. Fifth, the pictograph area might be manipulated by human which in some cases might change the shape of the pictograph. Another important challenge is illumination variation caused by weather condition or daylight changes. The last and more important issue shown in this figure is pictograph differences of the same traffic sign from one country to another. More specifically, we observe that the "danger: bicycle crossing" sign posses important differences between two countries.

Referring to the Vienna Convention on Road Traffic Signs, we can find roughly 230 pictorial traffic signs. Here, text-based signs and variations on pictorial signs are counted. For example, the speed limit sign can have 24 variations including 12 variation for indicating speed limits and 12 variations for end of speed limit. Likewise, traffic signs such as recommended speed, minimum speed, minimum distance with front car, etc. may have several variations. Hence, traffic sign recognition is a large multi-class classification problem. This makes the problem even more challenging.

Note that some of the signs such as "crossroad" and "side road" signs differ only by very fine details. This is shown in Fig. 1.3 where both signs differ only in small

Fig. 1.2 Some of the challenges in classification of traffic signs. The signs have been collected in Germany and Belgium

Fig. 1.3 Fine differences between two traffic signs

part of pictograph. Looking at Fig. 1.1, we see signs which are only 30 m away from the camera occupy very small region in the image. Sometimes, these regions can be as small as 20×20 pixels. For this reason, identifying fine details become very difficult on these signs.

In sum, traffic sign classification is a specific case of object classification where the objects are more rigid and two dimensional. Also, their discriminating parts are well defined. However, there are many challenges in developing a practical system for detection and classification of traffic signs.

1.3 Previous Work

1.3.1 Template Matching

Arguably, the most trivial way for recognizing objects is *template matching*. In this method, a set of templates is stored on system. Given a new image, the template is matched with every location on the image and a score is computed for each location. The score might be computed using *cross correlation*, *sum of squared differences*, *normalized cross correlation*, or *normalized sum of squared differences*. Piccioli et al. (1996) stored a set of traffic signs as the templates. Then, the above approach was used in order to classify the input image. Note that template-matching approach can be used for both detection and classification.

In practice, there are many problems with this approach. First, it is sensitive to perspective, illumination and deformation. Second, it is not able to deal with low quality signs. Third, it might need a large dataset of templates to cover various kinds of samples for each traffic sign. For this reason, selecting appropriate templates is a tedious task.

On the one hand, template matching compares raw pixel intensities between the template and the source image. On the other hand, pixel intensities greatly depend on perspective, illumination, and deformation. As the result, a slight change in illumination may affect the matching score, significantly. To tackle with this problem, we usually apply some algorithms on the image in order to extract more useful information from it. In other words, in the case of grayscale images, a *feature extraction* algorithm accepts a $W \times H$ image and transforms the $W \times H$ dimensional vector into a D-dimensional vector in which the D-dimensional vector carries more useful information about the image and it is more tolerant to perspective changes, illumination, and deformation. Based on this idea, Gao et al. (2006) extracted shape features from both template and source image and matched these feature vectors instead of raw pixel intensity values. In this work, matching features were done using the Euclidean distance function. This is equivalent to the sum of square differences function. The main problem with this matching function was that every feature was equally important. To cope with this problem, Ruta et al. (2010) learned a similarity measure for matching the query sign with templates.

1.3.2 Hand-Crafted Features

The template matching procedure can be decomposed into two steps. In the first step, a template and an image patch are represented using more informative vectors called feature vectors. In the second step, feature vectors are compared in order to find class of the image patch. This approach is illustrated in Fig. 1.4. Traditionally, the second step is done using techniques of machine learning. We will explain the basics of this step in Sect. 2.1. However, roughly speaking, extracting a feature vector from an image can be done using *hand-crafted* or *automatic* methods.

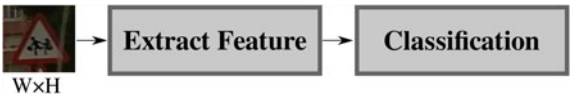

Fig. 1.4 Traditional approach for classification of objects

Hand-crafted methods are commonly designed by a human expert. They may apply series of transformations and computations in order to build a feature vector. For example, Paclík et al. (2000) generated a binary image depending on color of traffic sign. Then, moment invariant features were extracted from the binary image to form the feature vector. This method could be very sensitive to noise since a clean image and its degraded version may have two different binary images. Consequently, the moments of the binary images might vary significantly. Maldonado-Bascon et al. (2007) transformed the image into the HSI color space and calculated histogram of Hue and Saturation components. Although this feature vector can distinguish general category of traffic signs (for example, mandatory vs. danger signs), they might act poorly on modeling traffic signs of the same category. This is due to the fact that traffic signs of the same category have the same color and shape. For instance, all danger signs are triangle with a red margin. Therefore, the only difference would be the pictograph of signs. Since all pictographs are black, they will fall into the same bin on this histogram. As the result, theoretically, this bin will be the main source of information for classifying signs of same category.

In another method, Maldonado Bascón et al. (2010) classified traffic signs using only the pictograph of each sign. To this end, they first segment the pictograph from the image of traffic sign. Although the region of pictograph is binary, accurate segmentation of a pictograph is not a trivial task since automatic thresholding methods such as Otsu might fail taking into account the illumination variation and unexpected noise in real-world applications. For this reason, Maldonado Bascón et al. (2010) trained SVM where the input is a 31×31 block of pixels in a grayscale version of pictograph. In a more complicated approach, Baró et al. (2009) proposed an Error Correcting Output Code framework for classification of 31 traffic signs and compared their method with various approaches.

Zaklouta et al. (2011), Zaklouta and Stanciulescu (2012), and Zaklouta and Stanciulescu (2014) utilized more sophisticated feature extraction algorithm called Histogram of Oriented Gradient (HOG). Broadly speaking, the first step in extracting HOG feature is to compute the gradients of the image in x and y directions. Then, the image is divided into non-overlapping regions called *cells*. A histogram is computed for each cell. Bins of the histogram show the orientation of the gradient vector. Value of each bin is computed by accumulating the gradient magnitudes of the pixels in each cell. Next, blocks are formed using neighbor cells. Blocks may have overlap with each other. Histogram of a block is obtained by concatenating histograms of the cells within the block. Finally, histogram of each block is normalized and final feature vector is obtained by concatenating the histogram of all blocks.

This method is formulated using size of each cell, size of each block, number of bins in histograms of cell, and type of normalization. These parameters are called

hyperparameters. Depending on the value of these parameters we can obtain different feature vectors with different lengths on the same image. HOG is known to be a powerful hand-crafted feature extraction algorithm. However, objects might not be linearly separable in the feature space. For this reason, Zaklouta and Stanciulescu (2014) trained a Random Forest and a SVM for classifying traffic sings using HOG features. Likewise, Greenhalgh and Mirmehdi (2012), Moiseev et al. (2013), Mathias et al. (2013), Huang et al. (2013), and Sun et al. (2014) extracted the HOG features. The difference between these works mainly lies on their classification model (e.g., SVM, Cascade SVM, Extreme Learning Machine, Nearest Neighbor, and LDA). However, in contrast to the other works, Huang et al. (2013) used a two-level classification model. In the first level, the image is classified into one of super-classes. Each super-class contains several traffic signs with similar shape/color. Then, the perspective of the input image is adjusted based on its super-class and another classification model is applied on the adjusted image. The main problem of this method is sensitivity of the final classification to the adjustment procedure.

Mathias et al. (2013) proposed a more complicated procedure for extracting features. Specifically, the first extracted HOG features with several configurations of hyperparameters. In addition, they extracted more feature vectors using different methods. Finally, they concatenated all these vectors and built the final feature vector. Notwithstanding, there are a few problems with this method. Their feature vector is a 9000-dimensional vector constructed by applying five different methods. This high-dimensional vector is later projected to a lower dimensional space using a transformation matrix.

1.3.3 Feature Learning

A hand-crafted feature extraction method is developed by an expert and it applies series of transformations and computations in order to extract the final vector. The choice of these steps completely depends on the expert. One problem with the hand-crafted features is their limited representation power. This causes that some classes of objects overlap with other classes which adversely affect the classification performance. Two common approaches for partially alleviating this problem are to develop a new feature extraction algorithm and to combine various methods. The problems with these approaches are that devising a new hand-crafted feature extraction method is not trivial and combining different methods might not separate the overlapping classes.

The basic idea behind feature learning is *to learn features from data*. To be more specific, given a dataset of traffic signs, we want to learn a mapping $\mathscr{M} : \mathbb{R}^d \rightarrow \mathbb{R}^n$ which accepts $d = W \times H$-dimensional vectors and returns an n-dimensional vector. Here, the input is a flattened image that is obtained by putting the rows of image next to each other and creating a one-dimensional array. The mapping $\mathscr{M}$ can be any arbitrary function that is linear or nonlinear. In the simplest scenario, $\mathscr{M}$ can be

a linear function such as

$$\mathscr{M}(x) = W^{+}(x^{T} - \bar{x}), \tag{1.1}$$

where $W \in \mathbb{R}^{d \times n}$ is a weight matrix, $x \in \mathbb{R}^{d}$ is the flattened image, and $\bar{x} \in \mathbb{R}^{d}$ is the flattened mean image. Moreover, $W^{+} = (W^{T}W)^{-1}W^{T}$ denotes the MoorePenrose pseudoinverse of W. Given the matrix W we can map every image into a n-dimensional space using this linear transformation. Now, the question is how to find the values of W?

In order to obtain W, we must devise an objective and try to get as close as possible to the objective by changing the values of W. For example, assume our objective is to project x into a five-dimensional space where the projection is done arbitrarily. It is clear that any $W \in \mathbb{R}^{3 \times d}$ will serve our purpose. Denoting $\mathscr{M}(x)$ with z, our aim might be to project data on a $n \leq d$ space while maximizing the variance of z. The W that is found using this objective function is called *principal component analysis*. Bishop (2006) has explained that to find W that maximizes this objective function, we must compute the covariance matrix of data and find eigenvectors and eigenvalues of the covariance matrix. Then, the eigenvectors are sorted according to their eigenvalues in descending order and the first n eigenvectors are picked to form W.

Now, given any $W \times H$ image, we plug it in (1.1) to compute z. Then, the n-dimensional vector z is used as the feature vector. This method is previously used by Sirovich and Kirby (1987) for modeling human faces. Fleyeh and Davami (2011) also projected the image into the principal component space and found class of the image by computing the Euclidean distance of the projected image with the images in the database.

If we multiply both sides with W and rearrange (1.1) we will obtain

$$x^{T} = Wz + \bar{x}^{T}. \tag{1.2}$$

Assume that $\bar{x}^{T} = \mathbf{0}$. Technically, we say our data is *zero-centered*. According to this equation, we can *reconstruct* x using W and its mapping z. Each column in W is a d-dimensional vector which can be seen as a template learnt from data. With this intuition, the first row in W shows set of values of first pixel in our dictionary of templates. Likewise, n^{th} row in W is set of values of n^{th} pixel in the templates. Consequently, the vector z shows how to linearly combine these templates in order to reconstruct the original image. As the value of n increases, the reconstruction error decreases.

The value of W depends directly on the data that we have used during the training stage. In other words, using the training data, a system learns to extract features. However, we do not take into account the class of objects in finding W. In general, methods that do not consider the class of object are called *unsupervised* methods.

One limitation of principal component analysis is that $n \leq d$. Also, z is a real vector which is likely to be non-sparse. We can simplify (1.2) by omitting the second term.

Now, our objective is to find W and z by minimizing the constrained reconstruction error:

$$E = \sum_{i=1}^{N} \|x_i^T - Wz_i\|_2^2 \quad s.t. \quad \|z\|_1 < \mu, \tag{1.3}$$

where μ is a user-defined value and N is the number of training images. W and z_i also have the same meaning as we mentioned above. The $\mathscr{L}_1$ constrained in the above equation forces z_i to be *sparse*. A vector is called sparse when most of its elements are zero. Minimizing the above objective function requires an alternative optimization of W and z_i. This method is called *sparse coding*. Interested readers can find more details in Mairal et al. (2014). It should be noted that there are other formulations for objective function and the constraint.

There are two advantages with the sparse coding approach compared with principal component analysis. First, the number of columns in W (i.e., n) is not restricted to be smaller than d. Second, z_i is a sparse vector. Sparse coding has been also used to encode images of traffic signs.

Hsu and Huang (2001) coded each traffic sign using the *Matching Pursuit* algorithm. During testing, the input image is projected to different sets of filter bases to find the best match. Lu et al. (2012) proposed a graph embedding approach for classifying traffic signs. They preserved the sparse representation in the original space using $L_{1,2}$ norm. Liu et al. (2014) constructed the dictionary by applying k-means clustering on the training data. Then, each data is coded using a novel coding input similar to Local Linear Coding approach (Wang et al. 2010). Moreover, Aghdam et al. (2015) proposed a method based on visual attributes and Bayesian network. In this method, each traffic sign is described in terms of visual attributes. In order to detect visual attributes, the input image is divided into several regions and each region is coded using *elastic net* sparse coding method. Finally, attributes are detected using a random forest classifier. The detected attributes are further refined using a Bayesian network. Figure 1.5 illustrates a dictionary learnt by Aghdam et al. (2015) from 43 classes of traffic signs.

There are other unsupervised feature learning techniques. Among them, *autoencoders*, *deep belief networks*, and *independent component analysis* have been extensively studied and used in the computer vision community. One of the major disadvantages of unsupervised feature learning methods is that they do not consider the class of objects during the learning process. More accurate results have been obtained using supervised feature learning methods. As we will discuss in Chap. 3, convolutional neural networks (ConvNet) have shown a great success in classification and detection of objects.

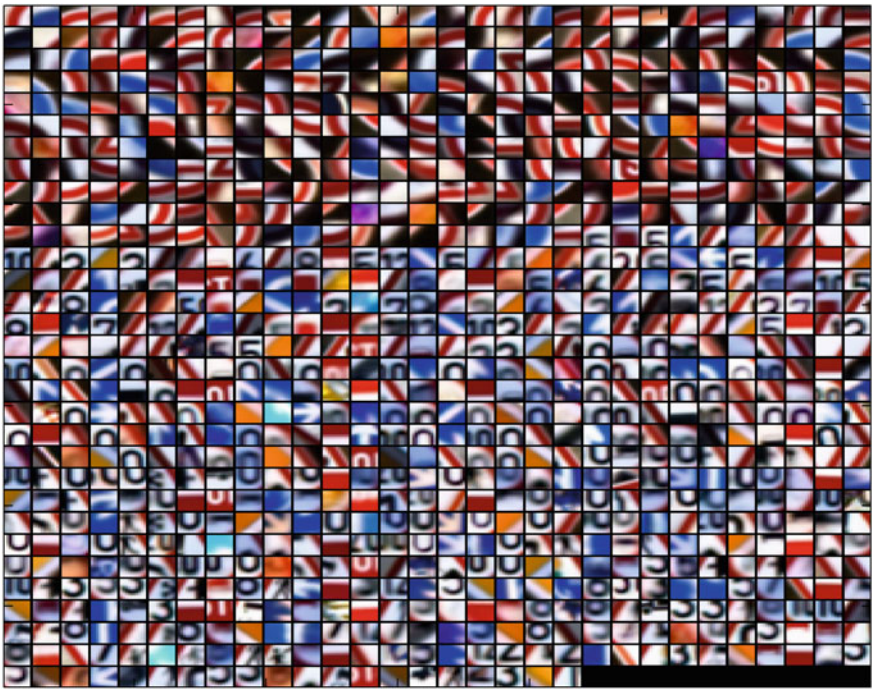

Fig. 1.5 Dictionary learnt by Aghdam et al. (2015) from 43 classes of traffic signs

1.3.4 ConvNets

[3]ConvNets were first utilized by Sermanet and Lecun (2011) and Ciresan et al. (2012) in the field of traffic sign classification during the German Traffic Sign Recognition Benchmark (GTSRB) competition where the ensemble of ConvNets designed by Ciresan et al. (2012) surpassed human performance and won the competition by correctly classifying 99.46% of test images. Moreover, the ConvNet of Sermanet and Lecun (2011) ended up in the second place with a considerable difference compared with the third place which was awarded for a method based on the traditional classification approach. The classification accuracies of the runner-up and the third place were 98.97 and 97.88%, respectively.

Ciresan et al. (2012) constructs an ensemble of 25 ConvNets each consists of 1,543,443 parameters. Sermanet and Lecun (2011) creates a single network defined by 1,437,791 parameters. Furthermore, while the winner ConvNet uses the *hyperbolic* activation function, the runner-up ConvNet utilizes the *rectified sigmoid* as the activation function. It is a common practice in ConvNets to make a prediction by calculating the average score of slightly transformed versions of the query image.

[3]We shall explain all technical details of this section in the rest of this book.

However, it is not clearly mentioned in Sermanet and Lecun (2011) that how do they make a prediction. In particular, it is not clear that the runner-up ConvNet classifies solely the input image or it classifies different versions of the input and fuses the scores to obtain the final result.

Regardless, both methods suffer from the high number of arithmetic operations. To be more specific, they use highly computational activation functions. To alleviate these problems, Jin et al. (2014) proposed a new architecture including 1,162,284 parameters and utilizing the *rectified linear unit* (ReLU) activations (Krizhevsky et al. 2012). In addition, there is a Local Response Normalization layer after each activation layer. They built an ensemble of 20 ConvNets and classified 99.65% of test images correctly. Although the number of parameters is reduced using this architecture compared with the two networks, the ensemble is constructed using 20 ConvNets which is not still computationally efficient in real-world applications. It is worth mentioning that a ReLU layer and a Local Response Normalization layer together needs approximately the same number of arithmetic operations as a single hyperbolic layer. As the result, the run-time efficiency of the network proposed in Jin et al. (2014) might be close to Ciresan et al. (2012).

Recently, Zeng et al. (2015) trained a ConvNet to extract features of the image and replaced the classification layer of their ConvNet with an Extreme Learning Machine (ELM) and achieved 99.40% accuracy on the GTSRB dataset. There are two issues with their approach. First, the output of last convolution layer is a 200-dimensional vector which is connected to 12,000 neurons in the ELM layer. This layer is solely defined by $200 \times 12{,}000 + 12{,}000 \times 43 = 2{,}916{,}000$ parameters which makes it impractical. Besides, it is not clear why their ConvNet reduces the dimension of the feature vector from $250 \times 16 = 4000$ in Layer 7 to 200 in Layer 8 and then map their lower dimensional vector to 12,000 dimensions in the ELM layer (Zeng et al. 2015, Table 1). One reason might be to cope with calculation of the matrix inverse during training of the ELM layer. Finally, since the input connections of the ELM layer are determined randomly, it is probable that their ConvNet does not generalize well on other datasets.

The common point about all the above ConvNets is that they are only suitable for the *classification* module and they cannot be directly used in the task of *detection*. This is due to the fact that applying these ConvNets on high-resolution images is not computationally feasible. On the other hand, accuracy of the *classification* module also depends on the *detection* module. In other words, any false-positive results produced by the detection module will be entered into the classification module and it will be classified as one of traffic signs. Ideally, the false-positive rate of the detection module must be zero and its true-positive rate must be 1. Achieving this goal usually requires more complex image representation and classification models. However, as the complexity of these models increases, the detection module needs more time to complete its task.

The ConvNets proposed for traffic sign classification can be explained from three perspectives including *scalability*, *stability*, and *run-time*. From generalization point of view, none of the four ConvNets have assessed the performance on other datasets. It is crucial to study how the networks perform when the signs slightly change from one

country to another. More importantly, the transferring power of the network must be estimated by fine-tuning the same architecture on a new dataset with various numbers of classes. By this way, we are able to estimate the *scalability* of the networks. From *stability* perspective, it is crucial to find out how tolerant is the network against noise and occlusion. This might be done through generating a few noisy images and fetch them to the network. However, this approach does not find the minimum noisy image which is misclassified by the network. Finally, the *run-time efficiency* of the ConvNet must be examined. This is due to the fact that the ConvNet has to consume as few CPU cycles as possible to let other functions of ADAS perform in real time.

1.4 Summary

In this chapter, we formulated the problem of traffic sign recognition in two stages namely detection and classification. The detection stage is responsible for locating regions of image containing traffic signs and the classification stage is responsible for finding class of traffic signs. Related work in the field of traffic sign detection and classification is also reviewed. We mentioned several methods based on hand-crafted features and then introduced the idea behind feature learning. Then, we explained some of the works based on convolutional neural networks.

References

Aghdam HH, Heravi EJ, Puig D (2015) A unified framework for coarse-to-fine recognition of traffic signs using Bayesian network and visual attributes. In: Proceedings of the 10th international conference on computer vision theory and applications, pp 87–96. doi:10.5220/0005303500870096

Baró X, Escalera S, Vitrià J, Pujol O, Radeva P (2009) Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification. IEEE Trans Intell Transp Syst 10(1):113–126. doi:10.1109/TITS.2008.2011702

Bishop CM (2006) Pattern recognition and machine learning. Information science and statistics. Springer, New York

Ciresan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: 2012 IEEE conference on computer vision and pattern recognition. IEEE, pp 3642–3649. doi:10.1109/CVPR.2012.6248110, arXiv:1202.2745v1

Fleyeh H, Davami E (2011) Eigen-based traffic sign recognition. IET Intell Transp Syst 5(3):190. doi:10.1049/iet-its.2010.0159

Gao XW, Podladchikova L, Shaposhnikov D, Hong K, Shevtsova N (2006) Recognition of traffic signs based on their colour and shape features extracted using human vision models. J Visual Commun Image Represent 17(4):675–685. doi:10.1016/j.jvcir.2005.10.003

Greenhalgh J, Mirmehdi M (2012) Real-Time Detection and Recognition of Road Traffic Signs. Ieee Transactions on Intelligent Transportation Systems 13(4):1498–1506. doi:10.1109/tits.2012.2208909

Hsu SH, Huang CL (2001) Road sign detection and recognition using matching pursuit method. Image Vis Comput 19(3):119–129. doi:10.1016/S0262-8856(00)00050-0

Huang GB, Mao KZ, Siew CK, Huang DS (2013) A hierarchical method for traffic sign classification with support vector machines. In: The 2013 international joint conference on neural networks (IJCNN). IEEE, pp 1–6. doi:10.1109/IJCNN.2013.6706803

Jin J, Fu K, Zhang C (2014) Traffic sign recognition with hinge loss trained convolutional neural networks. IEEE Trans Intell Transp Syst 15(5):1991–2000. doi:10.1109/TITS.2014.2308281

Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. Curran Associates, Inc., pp 1097–1105

Liu H, Liu Y, Sun F (2014) Traffic sign recognition using group sparse coding. Inf Sci 266:75–89. doi:10.1016/j.ins.2014.01.010

Lu K, Ding Z, Ge S (2012) Sparse-representation-based graph embedding for traffic sign recognition. IEEE Trans Intell Transp Syst 13(4):1515–1524. doi:10.1109/TITS.2012.2220965

Mairal J, Bach F, Ponce J (2014) Sparse modeling for image and vision processing. Found Trends Comput Graph Vis 8(2–3):85–283. doi:10.1561/0600000058

Maldonado Bascón S, Acevedo Rodríguez J, Lafuente Arroyo S, Fernndez Caballero A, López-Ferreras F (2010) An optimization on pictogram identification for the road-sign recognition task using SVMs. Comput Vis Image Underst 114(3):373–383. doi:10.1016/j.cviu.2009.12.002

Maldonado-Bascon S, Lafuente-Arroyo S, Gil-Jimenez P, Gomez-Moreno H, Lopez-Ferreras F (2007) Road-sign detection and recognition based on support vector machines. IEEE Trans Intell Transp Syst 8(2):264–278. doi:10.1109/TITS.2007.895311

Mathias M, Timofte R, Benenson R, Van Gool L (2013) Traffic sign recognition - How far are we from the solution? In: Proceedings of the international joint conference on neural networks. doi:10.1109/IJCNN.2013.6707049

Moiseev B, Konev A, Chigorin A, Konushin A (2013) Evaluation of traffic sign recognition methods trained on synthetically generated data. In: 15th international conference on advanced concepts for intelligent vision systems (ACIVS). Springer, Poznań, pp 576–583

Paclík P, Novovičová J, Pudil P, Somol P (2000) Road sign classification using Laplace kernel classifier. Pattern Recognit Lett 21(13–14):1165–1173. doi:10.1016/S0167-8655(00)00078-7

Piccioli G, De Micheli E, Parodi P, Campani M (1996) Robust method for road sign detection and recognition. Image Vis Comput 14(3):209–223. doi:10.1016/0262-8856(95)01057-2

Ruta A, Li Y, Liu X (2010) Robust class similarity measure for traffic sign recognition. IEEE Trans Intell Transp Syst 11(4):846–855. doi:10.1109/TITS.2010.2051427

Sermanet P, Lecun Y (2011) Traffic sign recognition with multi-scale convolutional networks. In: Proceedings of the international joint conference on neural networks, pp 2809–2813. doi:10.1109/IJCNN.2011.6033589

Sirovich L, Kirby M (1987) Low-dimensional procedure for the characterization of human faces. J Opt Soc Am A 4(3):519–524. doi:10.1364/JOSAA.4.000519, http://josaa.osa.org/abstract.cfm?URI=josaa-4-3-519

Sun ZL, Wang H, Lau WS, Seet G, Wang D (2014) Application of BW-ELM model on traffic sign recognition. Neurocomputing 128:153–159. doi:10.1016/j.neucom.2012.11.057

Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, pp 3360–3367. doi:10.1109/CVPR.2010.5540018

Zaklouta F, Stanciulescu B (2012) Real-time traffic-sign recognition using tree classifiers. IEEE Trans Intell Transp Syst 13(4):1507–1514. doi:10.1109/TITS.2012.2225618

Zaklouta F, Stanciulescu B (2014) Real-time traffic sign recognition in three stages. Robot Auton Syst 62(1):16–24. doi:10.1016/j.robot.2012.07.019

Zaklouta F, Stanciulescu B, Hamdoun O (2011) Traffic sign classification using K-d trees and random forests. In: Proceedings of the international joint conference on neural networks, pp 2151–2155. doi:10.1109/IJCNN.2011.6033494

Zeng Y, Xu X, Fang Y, Zhao K (2015) Traffic sign recognition using deep convolutional networks and extreme learning machine. In: Intelligence science and big data engineering. Image and video data engineering (IScIDE). Springer, pp 272–280