Soft Computing (2020) 24:6721–6733
https://doi.org/10.1007/s00500-019-04307-6

METHODOLOGIES AND APPLICATION

# A robust system for road sign detection and classification using LeNet architecture based on convolutional neural network

Amal Bouti[1] · Med Adnane Mahraz[1] · Jamal Riffi[1] · Hamid Tairi[1]

Published online: 7 September 2019


**Abstract**
In this paper, we are reporting a system for detection and classification of road signs. This system consists of two parts. The first part detects the road signs in real time. The second part classifies the German traffic signs (GTSRB) dataset and makes the prediction using the road signs detected in the first part to test the effectiveness. We used HOG and SVM in the detection part to detect the road signs captured by the camera. Then we used a convolutional neural network based on the LeNet model in which some modifications were added in the classification part. Our system obtains an accuracy rate of 96.85% in the detection part and 96.23% in the classification part.

**Keywords** Road signs · TSDR · Detection · Classification · Histogram of oriented gradients · Support-vector machine · Convolutional neural network · LeNet

## 1 Introduction

Detection and recognition of signs is an automobile equipment that reads and interprets permanent and temporary signs located at the edge or over the road, in order to inform the driver in case he could not see them. Speed limitation and over-ride signs are particularly concerned. Signs detection and recognition works through a camera mounted behind the interior rearview mirror. It detects signs located the left, right or over the road and compares it with an internal dataset. Once the sign is recognized, the driver is notified of the situation through a visual on the GPS or instrumentation. It is a useful aid for the driver who, in the ambient traffic, will not necessarily have seen the sign. Depending on the system, the car goes up to compare its speed with the current limitation and alerts the driver if he is overspeeding. In the long term, we can also imagine that the optical reading of the signs can communicate with the adaptive cruise control. It is the vehicle that would automatically decide how fast to drive.

Traffic sign detection and recognition (TSDR) has been very popular in recent years. This is due to the large number of applications that such a system can provide:

- Maintenance of signs.
- Signs inventory.
- Driving assistance.
- Smart autonomous vehicles.

### 1.1 Road sign detection

Methods of detection of road signs are divided into three classes: methods based on color, shape or machine learning. The dominant colors of most road signs are red, blue or yellow. Many authors (Lillo-Castellano et al. 2015; Ellahyani et al. 2016) use this property to detect signs. An associated component segmentation based on a color model is used. The regions of interest are then validated by a recognition algorithm or an appearance model. These methods are usually fast and invariant to translation, rotation, and scaling. Since color can be easily affected by lighting conditions, the main difficulty of color-based

Communicated by V. Loia.

✉ Amal Bouti
amalbouti@gmail.com

Med Adnane Mahraz
adnane_1@yahoo.fr

Jamal Riffi
riffijamal@gmail.com

Hamid Tairi
htairi@yahoo.fr

[1] LIIAN, FSDM, Fez, Morocco

methods is how to be invariant to different lighting conditions. These methods tend to follow a common pattern: the image is transformed into a color space and then thresholded. The two spaces HSV and HSI are often used by researchers because they are based on human perception of colors and encode color information in one channel instead of three (Ardianto et al. 2017).

For shape-based methods, the contours of the image are analyzed by a structural or global approach. These methods are generally more robust than color-based ones because they handle the gradient of the image and can handle grayscale images. These methods are very sensible to occlusions and deformation that affects considerably their performances. To overcome this problem, some researchers proposed to detect circular road signs using the circle detection algorithm EDCircles (Berkaya et al. 2016) and other authors proposed to detect circular and triangular signs using HOG and Linear SVM (Zaklouta and Stanciulescu 2012). Systems that adopt shape-based methods to minimize color change due to lighting and climate change face the problem of detecting occluded and damaged signs that require color-based methods.

Finally, for methods based on machine learning, a classifier (cascade, SVM, neural networks) is trained based on examples. It is applied on a sliding window that traverses the image on several scales. These methods combine geometry and photometry but can be a costly step in computing time. They require the constitution of a learning base by type of signs, tedious step when the number of objects to be detected is large. Many researchers (Ellahyani et al. 2016; Brkić et al. 2010; Chen and Lu 2016; Yi et al. 2016; Bouti et al. 2017) adopt this approach of combination between color-based methods and shape-based methods that can help to minimize the rate of false positives and increase the rate of true positives.

### 1.2 Road sign classification

The methods of classification of road signs are divided into two classes. Learning methods based on hand-crafted features and in-depth learning methods. The basic idea of learning methods based on hand-crafted features is to design an algorithm to extract the characteristics of the image and form a classifier over it. Indeed, the overall accuracy of traditional methods depends primarily on the feature extraction algorithm because there are powerful classifiers, such as SVM or random forest, that can accurately learn nonlinear decision boundaries. However, if classes overlap in the feature space, classifiers will not be able to discriminate classes accurately (Aghdam et al. 2016; Zaklouta et al. 2011; Kedkarn et al. 2015). SVM classifier with HOG represents also one of the most used techniques for the classification of textual information in road signs.

As shown in many researches in the literature (Yang et al. 2018; Kedkarn et al. 2015), in-depth learning methods such as convolutional neural networks learn a highly nonlinear function to project the raw image into a function space where classes are linearly separable and non-overlapping (Aghdam et al. 2016).

Our system consists of two main phases, detection and classification. In the detection phase, we used HOG and linear SVM to detect road signs. Although deep learning approaches have proven their superiority in similar image detection problem, it is interesting to find out how a traditional computer vision approach performs in a situation like this. The representation of the HOG features and SVM greatly improves the results obtained and shows good results in terms of accuracy. The linear SVM not only achieves high accuracy but also costs least compared with another kernel function (Ma and Huang 2015). In the classification phase, we used the convolutional neural networks (CNN) technique which has a formidable capacity to solve this kind of problem (i.e., image classification). We used an already existing CNN architecture LeNet and make some modifications to have the best performance and trained it to recognize signs using a dataset called German Traffic Sign Recognition Benchmark (GTSRB).

To do this, we have structured our article in 3 sections: In the first section, we will present the different methods used in our system among them the convolutional neural networks as well as their interests in the field of classification of images. In the second section, we will describe our system. Finally, in the third section we will show the different results obtained with a small comparison with other systems and in the end, we finish with a general conclusion.

## 2 Materials and methods

### 2.1 Histogram of oriented gradients (HOG)

The HOG feature descriptors were introduced by Dalal and Triggs (2005), researchers at INRIA Grenoble, at the CVPR conference in June 2005 in their pedestrian detection work. The essential idea behind the histogram of oriented gradients is that the local appearance and the object shape in an image can be described by the intensity distribution of the gradients or the direction of the contours. The implementation of these descriptors can be obtained by dividing the image into small connected regions, called cells, and for each cell a histogram of the gradient directions or contour orientations for the pixels in the cell is

calculated. The combination of these histograms then represents the descriptor. For best results, the local histograms are normalized in contrast, calculating a measure of intensity over wider areas than cells, called blocks, and using this value to normalize all cells in the block. This normalization allows a better resistance to the changes of illuminations and shadows.

#### 2.1.1 Gradient computation

The first step of calculation in many feature detectors in image pre-processing is to ensure normalized color and gamma values. This step was finally not necessary; the standardization of the descriptor itself is enough. The first step of the method is the gradient calculation; the most common method for doing this is to apply a $1 - D$ centered derivative filter in the horizontal and vertical directions. The following masks are used for this: $[-1, 0, 1]$ and $[-1, 0, 1]^{\mathrm{T}}$. In the case of color images, the gradient is calculated separately for each component, and for each pixel the gradient of the largest standard is retained.

#### 2.1.2 Histogram construction

The second step is the creation of histograms of gradient orientation. This is done in small square cells ($4 \times 4$ to $12 \times 12$ pixels). Each pixel of the cell then votes for a class of the histogram, depending on the gradient orientation at that point. The vote of the pixel is weighted by the intensity of the gradient at this point. The histograms are uniform from 0° to 180° (unsigned case) or from 0° to 360° (signed case).

#### 2.1.3 Descriptor block

An important step is the standardization of descriptors to avoid disparities due to illumination variations. This step also introduces redundancy into the descriptor. For this purpose, authors group several cells in a block, which is the unit on which the normalization is performed. The blocks overlap, so the same cell participates several times in the final descriptor, as a member of different blocks. Two types of block geometry are available: rectangular (R-HOG) or circular (C-HOG). The experiments done by Dalal and Triggs (2005) showed that the best performance was obtained for rectangular blocks containing $3 \times 3$ cells of $6 \times 6$ pixels each. A minor performance improvement is achieved by weighting the pixels by a Gaussian window on the block, decreasing the contribution of the pixels to the edges.

#### 2.1.4 Block normalization

Dalal and Triggs (2005) explored four different methods for block normalization. Let $v$ be the non-normalized vector containing all histograms in a given block, $\|v\|k$ be its $k$ norm for $k = 1, 2$ and $e$ be some small constant. Then the normalization factor can be one of the following:

- L2-norm: $f = \dfrac{v}{\sqrt{\|v\|_2^2 + e^2}}$ (1)
- L2-hys: L2-norm followed by clipping (limiting the maximum values of $v$ to 0.2) and renormalizing.
- L1-norm: $f = \dfrac{v}{\|v\|_1 + e}$ (2)
- L1-sqrt: $f = \sqrt{\dfrac{v}{\|v\|_1 + e}}$ (3)

In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. In their experiments, Dalal and Triggs (2005) found the L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less reliable performance; however, all four methods showed very significant improvement over the non-normalized data.

#### 2.1.5 Classification

The final step in the object detection process is the use of HOG descriptors to drive a supervised classifier. This step is not part of the definition of the HOG descriptor itself, and different types of classifiers can be used. Dalal and Triggs (2005) voluntarily choose a simple classifier, a linear kernel SVM, to essentially measure HOG input. They specify that it would be interesting to develop a cascade-based method such as the Viola and Jones method, using HOGs. They specify that the use of a Gaussian kernel improves performance by 3%, for a false positive rate per window of $10^{-4}$ but a much higher computational cost.

### 2.2 Support-vector machine (SVM)

Support-vector machine (SVM) is a supervised machine learning algorithm initially defined for discrimination, that is, predicting a binary qualitative variable. They were then generalized to the forecast of a quantitative variable. In the case of discrimination of a dichotomous variable, they are based on finding the optimal margin hyperplane, where possible, classifies or separates the data correctly while being as far as possible from all observations. The principle

is therefore to find a classifier, or a function of discrimination, whose capacity of generalization (quality of forecasting) is the greatest possible. SVM can be used for classification, regression SVR and detection outlets. However, it is mostly used in classification problems. It can solve linear and nonlinear problems and work well for many practical problems.

A version of SVM for regression was proposed in 1996 by Vapnik et al. (Drucker et al. 1996). This method is called support-vector regression (SVR) which has been developed to solve nonlinear forecasting problems. Unlike ANN models, the SVR model can avoid over-learning, local minima, and dimension disaster problems (Hong et al. 2019).

The first part of the process is to create a model from the dataset. This is the learning of the class. The dataset is broken down into a set of positive elements (containing one element of the class) and a set of negative elements (not containing any element of the class). A hyperplane separating the elements of each of the two sets is calculated in order to maximize the margin (Fig. 1), that is, the resistance between the samples and the hyperplane. For this, the study space is transcribed on a larger space where the existence of a linear separator is possible. Finally, the set of positive examples is on one side of the hyperplane while the set of negative examples is on the other side. In the second part of the process, this model allows decision. If the vector e of the hyperplane relative to the positive examples, then it is an element of the class. In the opposite case, it is not an element of the class. It may be noted that the distance of the characteristic vector to the hyperplane gives an evaluation of the reliability of the decision. Indeed, if this distance is very small, the decision will be less sharp because the example is very close to the two classes.

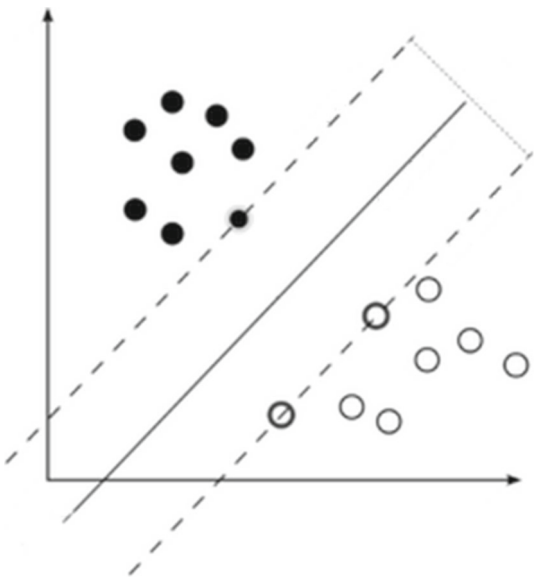

**Fig. 1** Illustration of the principle of operation of SVMs for a simple linear case. The elements of a class (white circles) are separated from elements of another class (black circles) by the separating hyperplane (solid line) which maximizes the margin (dotted lines)

The advantages of support-vector machines are:

- Effective in high-dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

## 2.3 Deep learning

This family of algorithms has made significant progress in the areas of image classification and language processing.

The deep learning models are built on the same model as the MLP multilayer perceptron. However, it should be emphasized that the various intermediate layers are more numerous. Each of the intermediate layers will be subdivided into sub-part, dealing with a sub-problem, simpler and providing the result to the next layer, and so on.

There are different algorithms of deep learning. We can cite:

- *Deep Neural Networks* These networks are like MLP (multilayer perceptron) networks but with more hidden layers. The increase in the number of layers allows a network of neurons to detect slight variations of the learning model, favoring over-learning or over-fitting.
- *Convolutional Neural Networks (CNN)* The problem is divided into subparts, and for each part, a cluster of neurons will be created to study this specific portion. For example, for a color image, it is possible to divide the image over width, height and depth (colors).
- *Deep Belief Network Machine* These algorithms work in a non-supervised first phase, followed by supervised classical training. This unsupervised learning step, furthermore, facilitates supervised learning.

## 2.4 Convolutional neural network (CNN)

Convolutional neural networks are currently the most efficient models for classifying images. Designated by the acronym CNN, they have two distinct parts. In input, an image is provided in the form of a matrix of pixels. It has 2 dimensions for a grayscale image. The color is represented by a third dimension, of depth 3 to represent the fundamental colors (Red, Green, Blue).

The first part of a CNN is the actual convolutive part. It functions as a feature extractor of images. An image is passed through a succession of filters, or convolution kernels, creating new images called convolution maps. Some intermediate filters reduce the resolution of the image by a local maximum operation. Finally, the convolution maps are laid flat and concatenated into a feature vector, called the CNN code. This CNN code at the output of the convolutive portion is then connected to the input of a second portion, consisting of fully connected layers (multilayer perceptron). The role of this part is to combine the characteristics of the CNN code to classify the image.

The output is a final layer with one neuron per category. The numerical values obtained are generally normalized between 0 and 1, of sum 1, to produce a probability distribution on the categories. Creating a new convolutional neural network is expensive in terms of the expertise, material, and amount of annotated data needed.

It is first about fixing the architecture of the network, the number of layers, their sizes and the matrix operations that connect them. The training then consists of optimizing the network coefficients to minimize the output classification error. This training can take several weeks for the best CNNs, with many GPUs working on hundreds of thousands of annotated images.

Research teams specialize in improving CNN. They publish their technical innovations, as well as the details of networks driven on databases of references.

CNN architecture is formed by a stack of independent processing layers:

- Convolutional layer is the key component of convolutional neural networks and is still at least their first layer. Its purpose is to locate the presence of a set of features in the images received as input.
- Pooling layer: This type of layer is often placed between two convolution layers: it receives several feature maps as input and applies the pooling operation to each of them. The pooling operation consists of reducing the size of the images, while preserving their important characteristics.
- Correction layer (ReLU) often referred to as abuse 'ReLU' with reference to the activation function (linear grinding unit). The function ReLU designates the real nonlinear function defined by: $F(x) = \max(0, x)$. This function forces the neurons to return positive values.
- Fully connected layer is a perceptron type layer and is always the last layer of a neural network, convolutional or not—it is not characteristic of a CNN. This type of layer receives an input vector and produces a new output vector. For this, it applies a linear combination then possibly an activation function to the values received at the input. The last fully connected layer classifies the input image of the network
- Loss layer specifies how network training penalizes the difference between the expected and actual signal. It is normally the last layer in the network. Various loss functions suitable for different tasks can be used. The 'Softmax' function is used to calculate the probability distribution on the output classes.

#### 2.4.1 Typical architecture of a convolutional neural network

A CNN is simply a stack of multiple layers of convolution, pooling, ReLU correction and the fully connected layer. Each image received as input will therefore be convolved, reduced and corrected several times, to finally form a vector. In the classification problem, this vector contains the probability of belonging to classes.

#### 2.4.2 Setting the layers

A convolutional neural network is distinguished from another by the way the layers are stacked, but also parameterized. In fact, the convolution and pooling layers have hyperparameters, parameters whose value you must first define. The size of the feature maps at the output of the convolution, and pooling layers depend on the hyperparameters. Each image (or feature map) has dimensions $W \times H \times D$, where $W$ is its width in pixels, H is its height in pixels and D is the number of channels (1 for a black and white image, 3 for an image in colors).

The convolution layer has four hyperparameters:

- The number of filters $K$.
- The size $F$ of the filters: each filter has dimensions $F \times F \times D$ pixels.
- The step $S$ with which we slide the window corresponding to the filter on the image. For example, a step of 1 means that the window is moved one pixel at a time.
- The zero-padding $P$: we add to the input image of the layer a black outline of thickness $P$ pixels. Without this outline, the output dimensions are smaller. Thus, the more one stack of convolution layers with $P = 0$, the more the input image of the network shrinks. So, we lose a lot of information quickly, which makes the task of extracting features difficult.

For each image of size $W \times H \times D$ at the input, the convolution layer returns a matrix of dimensions $W_C \times H_C \times D_C$, where $W_C = \frac{W-F+2P}{S} + 1$, $H_C = \frac{H-F+2P}{S} + 1$ and $D_C = K$.

Choosing $P = \frac{F-1}{2}$ and $S = 1$ thus makes it possible to obtain feature maps of the same width and height as those received as input.

The pooling layer has only two hyperparameters:

- The size $F$ of the cells: the image is divided into square cells of size $F \times F$ pixels.
- Step $S$: the cells are separated from each other by $S$ pixels.

For each image size $W \times H \times D$ at the input, the pooling layer returns a matrix of $W_P \times H_P \times D_P$ dimensions, where $W_P = \frac{W-F}{S} + 1$, $H_P = \frac{H-F}{S} + 1$ and $D_P = D$.

Like stacking, the choice of hyperparameters is done according to a classical scheme:

- For the convolution layer, the filters are small and dragged on the image one pixel at a time. The zero-padding value is chosen so that the width and height of the input volume are not changed at the output. In general, we choose $F = 3, P = 1, S = 1$ or $F = 5, P = 2, S = 1$.
- For the pooling layer, $F = 2$ and $S = 2$ is a wise choice. This eliminates 75% of the input pixels. We can also find $F = 3$ and $S = 2$: in this case, the cells overlap. Choosing larger cells causes too much loss of information and performs poorly in practice.

#### 2.4.3 Learning transfer

To train a convolutional neural network is very expensive: the more layers stack up, the higher the number of convolutions and parameters to be optimized. The computer must be able to store several gigabytes of data and efficiently perform the calculations. That's why hardware manufacturers are stepping up their efforts to provide high-performance graphics processors (GPUs) that can quickly drive a deep neural network by paralleling calculations.

Transfer learning allows you to do deep learning without having to spend a month of calculations. The principle is to use the knowledge acquired by a network of neurons when solving a problem in order to solve another similar. This is a transfer of knowledge, hence the name.

#### 2.4.4 LeNet 5

LeNet-5 is LeCun's latest convolutional network designed for handwritten and printed character recognition. It is a convolutional neural network with enough input to receive multiple objects and multiple outputs called space displacement neural networks (SDNN), capable of recognizing strings in a single pass without segmentation prior. Sparse convolutional layers and max pooling are at the heart of the LeNet family of models. Although the exact details of the model vary considerably (Fig. 2).

The lower layers are composed of convolution layers and max pool. The upper layers are, however, fully connected and correspond to a traditional MLP (hidden layer + logistic regression). The entry of the fully connected first layer is the set of all feature cards in the lower layer.

From the point of view of implementation, this means that the lower layers operate on the 4D tensors. These are then flattened into a 2D matrix of rasterized feature maps, to be compatible with the previous MLP implementation.

## 3 Proposed system

In this paper, we report our built system which is based essentially on two steps: detection and classification. In the detection phase, we used the HOG algorithm to describe the distribution of image gradients in different orientations and to capture shape and aspect characteristics in detecting signs. Then the SVM classifier was used to filter false positives. For the classification phase, we have used convolutional neural networks based on LeCun's model (1998), and by adding some modifications to improve the model, to classify the German traffic signs into predefined classes then we will make a prediction by using the signs obtained in the detection part to check the efficiency of our classifier (Fig. 3).

In the detection step, we have performed a histogram of oriented gradients (HOG) feature extraction on a labeled training set of images. Then, we trained a classifier with the set of sign and non-sign images. Finally, we used the trained classifier to detect signs. In this part linear support-vector machines (SVM) are used to train a model to classify if an image contains a sign or not. HOG is used as the feature representation.

The idea of HOG is instead of using each individual gradient direction of each individual pixel of an image, we group the pixels into small cells. For each cell, we compute all the gradient directions and group them into several orientation bins. We sum up the gradient magnitude in each sample. So stronger gradients contribute more weight to their bins, and effects of small random orientations due to noise are reduced. This histogram gives us a picture of the dominant orientation of that cell. Doing this for all cells gives us a representation of the structure of the image. The HOG features keep the representation of an object distinct but also allow for some variations in shape.

We can specify the number of orientations, pixels per cell, and cells per block for computing the HOG features of a single channel of an image. The number of orientations is the number of orientation bins that the gradients of the

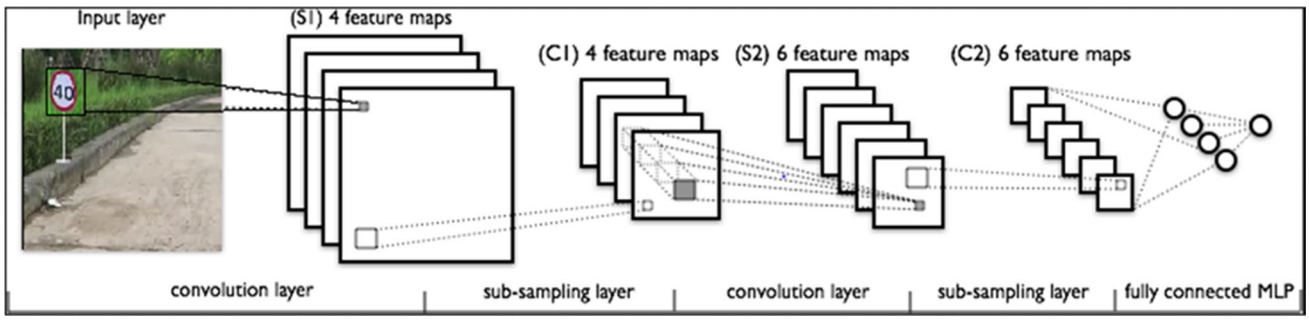


**Fig. 2** A graphical representation of a model LeNet

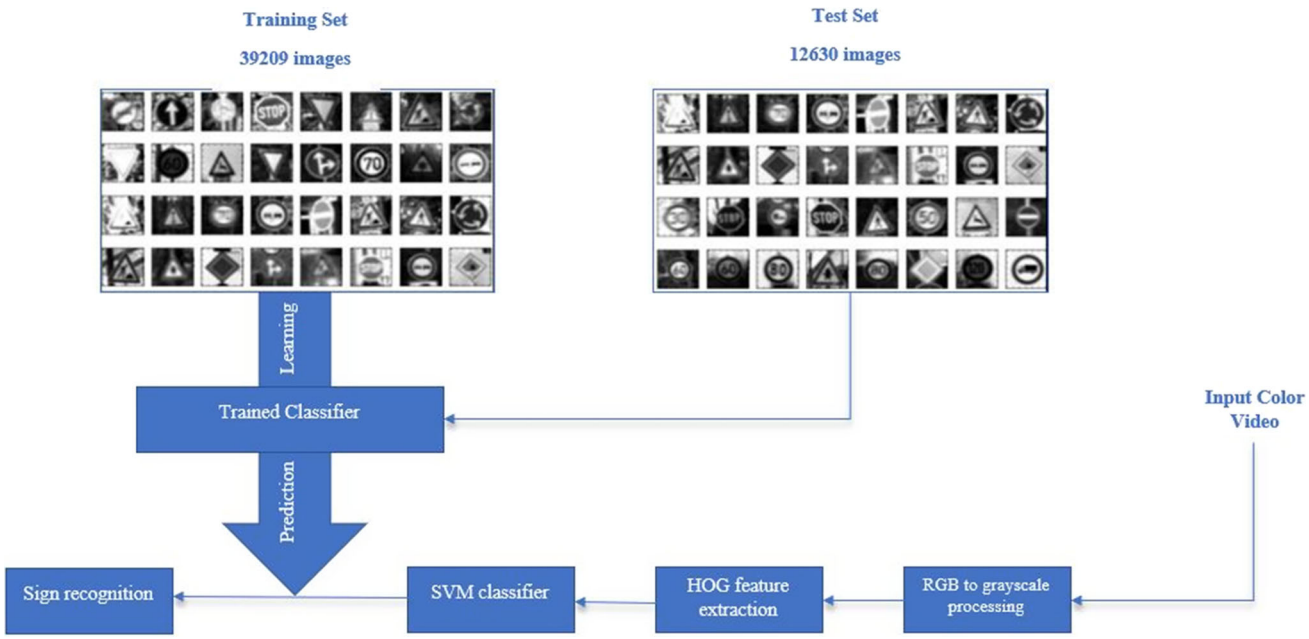


**Fig. 3** The proposed system for the detection and classification of road signs

pixels of each cell will be split up in the histogram. The pixels per cells are the number of pixels of each row, and column per cell over each gradient the histogram is computed. The cells per block specify the local area over which the histogram counts in a given cell will be normalized. Having this parameter is said to generally lead to a more robust feature set.

The classifier algorithm we used in this step is called a linear support-vector machine. As a safety measure, we use a scaler to transform the raw features before feeding them to our classifier for training or predicting, reducing the chance of our classifier to behave badly.

In the classification step, we constructed a deep learning pipeline to classify the German traffic sign dataset (GTSRB). The initial model is a convolutional neural network based on LeCun's LeNet architecture. TensorFlow and the scikit-learn pipeline framework were used with various combinations of transformers and estimators. The scikit-learn pipeline framework is used to manage different pipeline scenarios.

Pipeline can be used to group multiple estimators into one. This is useful because there is often a fixed sequence of steps in data processing, such as feature selection, normalization, and classification. Pipeline serves many purposes here:

- *Convenience and encapsulation* Simply call fit and predict once on your data to fit a complete sequence of estimators.
- *Selecting Common Parameters* You can search parameters of all pipeline estimators at the same time.
- *Safety* Pipelines help prevent statistical leakage from your test data in the model formed during cross-validation, ensuring that the same samples are used to train transformers and predictors.

All but one of the pipeline estimators must be transformers (i.e., they must have a transformation method). The last estimator can be of any type (transformer, classifier, etc.).

### 3.1 Training dataset

The dataset used is the German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al. 2011). This dataset is composed of 39,209 images and 43 classes. The objective is to classify the images of the German dataset signs into the predefined classes (Fig. 4). The images are of different sizes. The brightness of the image is quite random. Images can be rotated slightly. Images may not be exactly centered.

### 3.2 Model architecture

Our model (Fig. 5) is based on LeNet by LeCun et al. (1998). It has 7 layers, including 3 convolutional layers, 2 layers of subsampling (pooling) and 1 fully connected layer, followed by the output layer. The convolutional layers use 5 out of 5 convolutions. The subsampling layers are 2 by 2 average clustering layers. The activation is ReLU except for the output layer that uses Softmax. The output has 43 classes. For this model, the network works well. The performance seems to be pretty good. However, it shows a problem of over-fitting (due to the lower number of epochs).

### 3.3 Optimization of LeNet model parameters

We tried to optimize the basic model (1st model which is LeNet) by modifying some parameters. We tested 3 models presented in Table 1. The modified parameters are convolution layer filters, fully connected layer neurons, epochs, learning rate and adding a dropout layer.

- A convolution layer has a weight, which is its filter, and a bias.
- A fully connected layer is just a regular layer of neurons in a neural network. Each neuron receives the inputs of all the neurons of the previous layer, so closely connected.
- The epoch can be defined as a forward and a backward pass of all learning data.


**Fig. 4** Example of the predefined classes

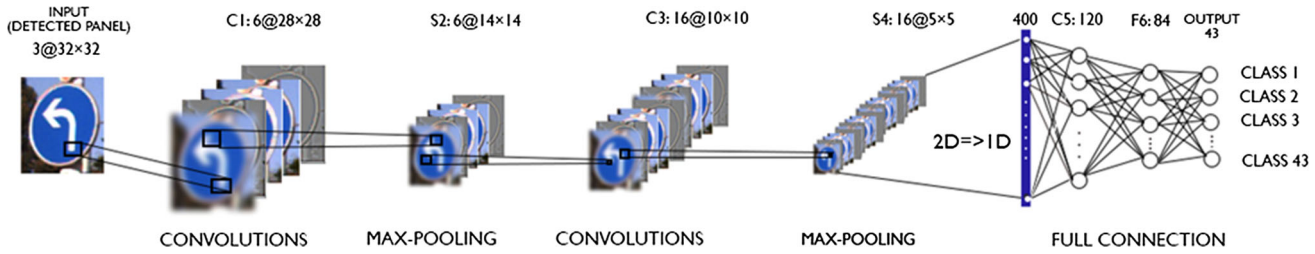


**Fig. 5** Architecture of our model

**Table 1** The different models obtained after the modification of the 1st model

| | Filters in the 1st convolution layer | Filters in the 2nd convolution layer | Neurons in fully connected layer | Epochs | Learning rate | Adding a dropout layer |
|---|---|---|---|---|---|---|
| Model 1 | 5 × 5 × 6 | 5 × 5 × 16 | 120 | 5 | $1.0\ e^{-3}$ | X |
| Model 2 | 5 × 5 × 24 | 5 × 5 × 64 | 480 | 5 | $1.0\ e^{-3}$ | X |
| Model 3 | 5 × 5 × 24 | 5 × 5 × 64 | 480 | 20 | $0.5\ e^{-3}$ | X |
| Model 4 | 5 × 5 × 24 | 5 × 5 × 64 | 480 | 500 | $1.0\ e^{-4}$ | With a dropout layer |

- The learning rate defines the step size during the gradient descent. This is a parameter chosen by the programmer.
- The dropout layer can be considered as a form of regulation to prevent over-fitting. This is a technique of ignoring randomly selected neurons during training.

The following graph (Fig. 6) represents the classification rate of each model.

From this simulation, we can deduce that the model 4 is the most efficient compared to other models.

## 4 Results and discussion

### 4.1 Experimental results

Our system achieved an overall accuracy of 96.85% in the detection part and an overall accuracy of 96.23% in the classification part using a convolutional neural network model. In Fig. 7, we present some results obtained in the detection phase.

In the classification phase, after optimizing our CNN model, we plotted the learning curve and the confusion matrix (Figs. 8, 9). The learning curve is a comparison of the model's performance based on the number of epochs. The convergence of the model is obtained after 500 epochs.

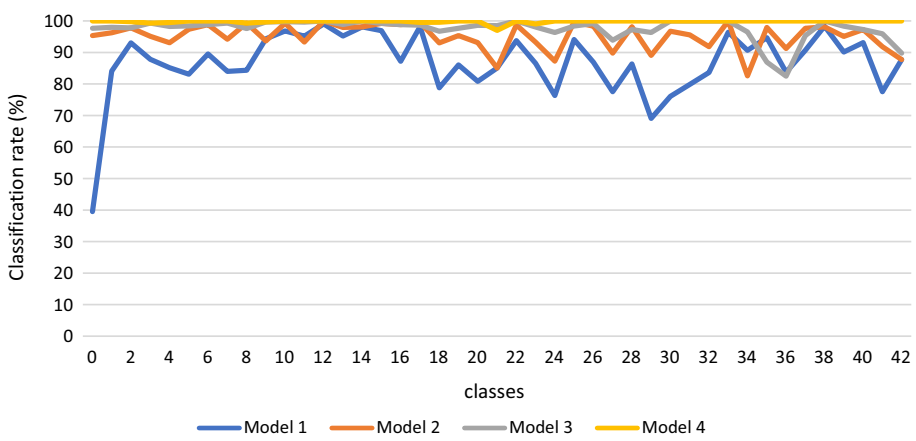


**Fig. 6** The curves of the classification rate of the 4 models

**Fig. 7** Some results obtained in the detection phase

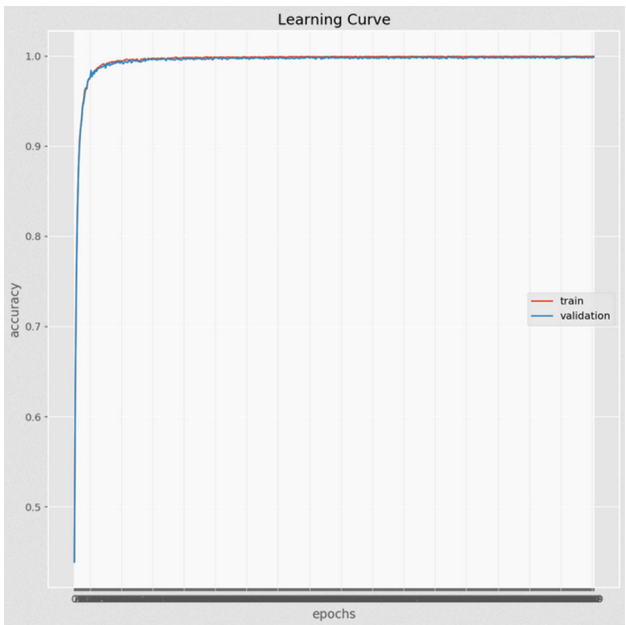


**Fig. 8** The learning curve

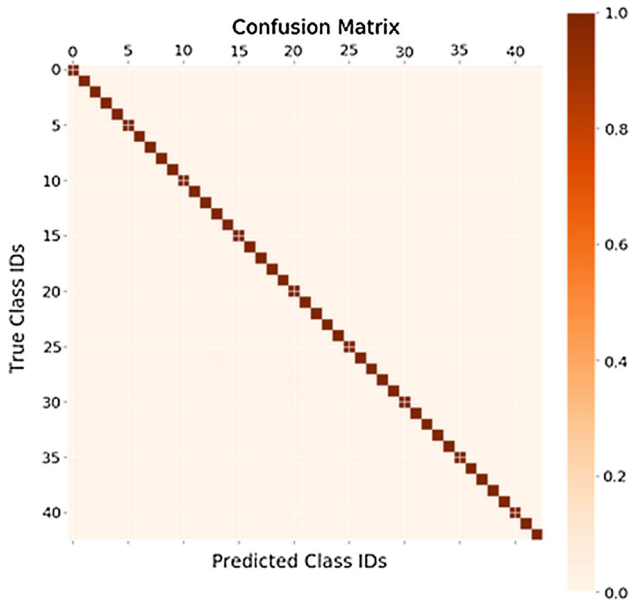


**Fig. 9** The confusion matrix

The confusion matrix allows us to measure the quality of the classification system. We plotted the learning curve and confusion matrix based on the model adopted (see Figs. 8, 9).

In the prediction part, we used the detected signs, in the detection phase, as input to our CNN classifier. As output, we get a percentage of belonging to the five most representative classes. In Fig. 10, we see some results. We can understand why it did not correctly identify the 40 km/h speed limit sign, because we do not have a class of this category (we just have: 20, 30, 50, 60, 70, 80, 100 and 120).

We can already estimate a success rate of up to 100% if the sign is German. This result may deteriorate in the opposite case. It also means that for each country/region, it would be necessary to train the classifier through a learning base of its road signs.

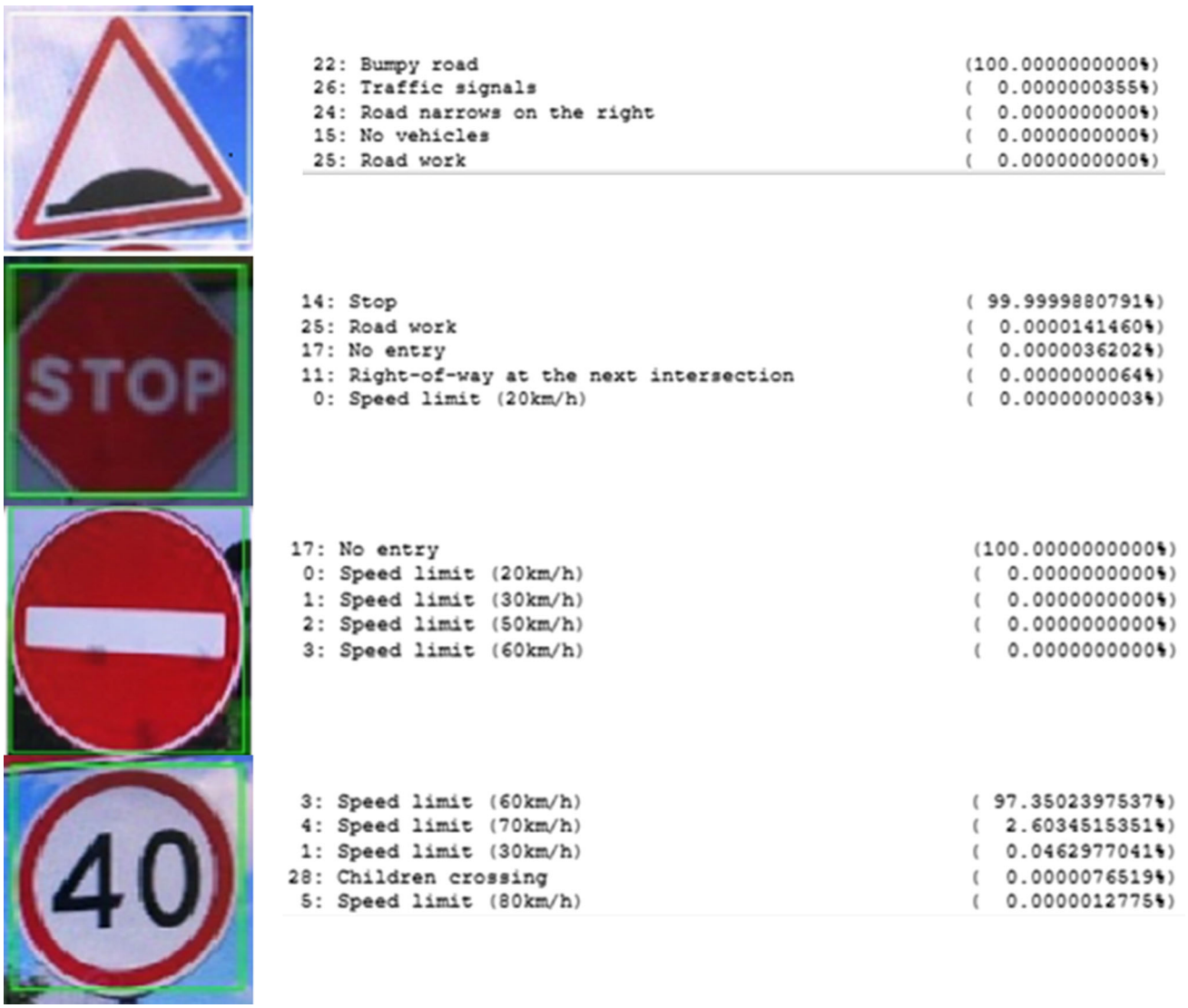


**Fig. 10** The results obtained in the prediction part

## 4.2 Discussion

In this part, we compare our system with other systems based on the same dataset (GTSRB).

In the paper of Huang et al. (2017), first they filter using the color information parts of the images that does not contain panels. Then, they extract the region where image blocks can be, and then extract the candidate region from the image block. Finally, they use deep learning to verify candidate areas of non-road signs and identify the type of road signs. Their system structure focuses on the HSI color space and then divides the candidate area into smaller pieces based on the color, texture, size, and similarity of the regions. The HSI color space is used to reduce the impact of changes in light and shadows in the image. Finally, the convolutional neuron network is used to identify the candidate region. Their system is designed for red traffic signs, so they only record red signs as the truth on the ground. The accuracy of the test depends on the selection of the candidate area by the selective search and its correct extraction with a selection frame. The detection system focuses solely on the correct capture of the location of traffic signs. The image region of the selected road signs is integrated into the identification system. In this system, the part of deep learning is based on Caffe. They use the gradient descent method to perform the training. The correct rates for the detection and recognition systems of Huang et al. (2017) are, respectively, 92.63% and 80.5%, while we obtained 96.85% and 96.23%, which is a very satisfactory result knowing that our system handles all traffic signs, not just red ones.

**Table 2** The accuracy of our system and other systems in the detection and classification steps

| Methods | Huang et al. (2017) | Ours | Yi et al. (2016) |
|---|---|---|---|
| Detection | 92.63 | 96.85 | 97.72 |
| Classification | 80.5 | 96.23 | 97.75 |

In this article (Yi et al. 2016), they present a real-time signal recognition system consisting of detection and classification modules. In the detection module, they first transform the input color image into probability maps of the panels using a color probability model. The probability map is a gray image, in which the pixels of the traffic signs will have a high intensity and the other pixels a low intensity. Secondly, they extract the proposed road signs by looking for extremely stable extreme regions (MSERs) from probability maps. Third, they filter out false positives and classify the remaining road sign proposals into their super classes using support-vector machine (SVM) based on a new HOG color feature. In the detection module, they classified the detected signs into their super classes. However, they still do not know which subclasses they belong to. In addition, there are false alarms in the detected signs. Therefore, they further classify the detected signs in their subclasses or background classes in this module. To this end, they form three CNNs for the three super classes, respectively. In the classification module, they further classify the detected traffic signs in their specific subclasses using a convolutional neural network. Unlike detection, color provides little distinctive information for classification. They only use gray images to reduce the processing time. In addition, they resize all $32 \times 32$ images because the CNN input should be the same size. This document (Yi et al. 2016) reach for the detection phase 97.72% and for the classification 97.75%, which is a performance comparable to that of the state-of-the-art methods.

Table 2 summarizes the success rates found for this works compared to our system.

## 5 Conclusion

In this paper, we tried to propose a system for the detection and classification of road signs. Our system consists of two phases: a detection phase and a classification phase. In the sign detection phase, we used HOG and SVM, while in the classification phase, we adopted an optimized model of a CNN architecture. The learning of our classifier was done using the German dataset. In the experimental part, we tested our system on live video scenes and we proved that our system can achieve a very high detection and recognition rate, which is a very encouraging result.

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References


Aghdam HH, Heravi EJ, Puig D (2016) A practical approach for detection and classification of traffic signs using convolutional neural networks. Robot Auton Syst 84:97–112

Ardianto S, Chen C-J, Hang H-M (2017) Real-time traffic sign recognition using color segmentation and SVM. In: International conference on systems, signals and image processing (IWSSIP)

Berkaya SK, Gunduz H, Ozsen O, Akinlar C, Gunal S (2016) On circular traffic sign detection and recognition. Exp Syst Appl 48:67–75

Bouti A, Mahraz MA, Riffi J, Tairi H (2017) Robust system for road sign detection and recognition using template matching. In: Intelligent systems and computer vision (ISCV), Fez, Morocco

Brkić K, Šegvić S, Kalafatić Z, Sikirić I, Pinz A (2010) Generative modeling of spatio-temporal traffic sign trajectories. In: IEEE computer society conference on computer vision and pattern recognition workshops (CVPRW)

Chen T, Lu S (2016) Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression. IEEE Trans Veh Technol 65:4006–4015

Dalal N, Triggs B (2005) Histogram of oriented gradients for human detection. In: Proceedings of IEEE computer vision and pattern recognition (CVPR)

Drucker H, Burges CJC, Kaufman L, Smola AJ, Vapnik VN (1996) Support vector regression machines. Adv Neural Inf Process Syst 9:155–161

Ellahyani A, El Ansari E, El Jaafari I (2016) Traffic sign detection and recognition based on random forests. Appl Soft Comput 46:805–815

Hong W, Li M, Geng J, Zhang Y (2019) Novel chaotic bat algorithm for forecasting complex motion of floating platforms. Appl Math Model 72:425–443

Huang S-C, Lin H-Y, Chang C-C (2017) An in-car camera system for traffic sign detection and recognition. In: Joint 17th world congress of international fuzzy systems association and 9th international conference on soft computing and intelligent systems (IFSA-SCIS)

Kedkarn C, Anusara H, Ratiporn C, Kittisak K, Nittaya K (2015) Traffic sign classification using support vector machine and image segmentation. In: The 3rd international conference on industrial application engineering

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, vol 86, No. 11

Lillo-Castellano JM, Mora-Jiménez I, Figuera-Pozuelo C (2015) Traffic sign segmentation and classification using statistical learning methods. Neurocomputing 153:286–299

Ma Y, Huang L (2015) Hierarchical traffic sign recognition based on multi-feature and multi-classifier fusion. In: First international conference on information science and electronic technology (ISET)

Stallkamp J, Schlipsing M, Salmen J, Igel C (2011) The German traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks

Yang T, Long X, Sangaiah AK, Zheng Z, Tong C (2018) Deep detection network for real-life traffic sign in vehicular networks. Comput Netw 136:95–104

Yi Y, Hengliang L, Huarong X, Fuchao W (2016) Towards real-time traffic sign detection and classification. IEEE Trans Intell Transp Syst 17:2022–2031

Zaklouta F, Stanciulescu B (2012) Real-time traffic sign recognition in three stages. Robot Auton Syst 62:16–24

Zaklouta F, Stanciulescu B, Hamdoun O (2011) Traffic sign classification using K-d trees and Random Forests. In: The 2011 international joint conference on neural networks