Robotics and Autonomous Systems 84 (2016) 97–112

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

A practical approach for detection and classification of traffic signs using Convolutional Neural Networks

Hamed Habibi Aghdam *, Elnaz Jahani Heravi, Domenec Puig

Department of Computer Engineering and Mathematics, University Rovira i Virgili, Spain

HIGHLIGHTS

- Implementing the scanning window technique on ConvNets using dilated convolutions.
- Proposing a highly optimized and accurate ConvNet for classification of traffic signs.
- Analyzing stability of the proposed ConvNet on noisy images.

ARTICLE INFO

Article history:
Available online 19 July 2016

Keywords:
Convolutional Neural Networks
Traffic sign detection
Traffic sign classification
Sliding window detection
Dense prediction

ABSTRACT

Automatic detection and classification of traffic signs is an important task in smart and autonomous cars. Convolutional Neural Networks has shown a great success in classification of traffic signs and they have surpassed human performance on a challenging dataset called the German Traffic Sign Benchmark. However, these ConvNets suffer from two important issues. They are not computationally suitable for real-time applications in practice. Moreover, they cannot be used for detecting traffic signs for the same reason. In this paper, we propose a lightweight and accurate ConvNet for detecting traffic signs and explain how to implement the sliding window technique within the ConvNet using dilated convolutions. Then, we further optimize our previously proposed real-time ConvNet for the task of traffic sign classification and make it faster and more accurate. Our experiments on the German Traffic Sign Benchmark datasets show that the detection ConvNet locates the traffic signs with average precision equal to 99.89%. Using our sliding window implementation, it is possible to process 37.72 high-resolution images per second in a multi-scale fashion and locate traffic signs. Moreover, single ConvNet proposed for the task of classification is able to classify 99.55% of the test samples, correctly. Finally, our stability analysis reveals that the ConvNet is tolerant against Gaussian noise when $\sigma < 10$.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Traffic sign recognition is one of the major tasks in Advanced Driver Assistance Systems (ADAS) and it is an indispensable part of autonomous cars. Recognizing traffic signs is mainly done in two stages including *detection* and *classification*. The detection module performs a multi-scale analysis on the image in order to locate the patches consisting only of one traffic sign. Next, the classification module analyzes each patch individually and classifies them into classes of traffic signs.

One of the important characteristics of traffic signs is their design simplicity which facilitates their recognition for a human driver. First, they have a simple shape such as circle, triangle, polygon or rectangle. Second, they are usually composed of basic colors such as red, green, blue, black, white and yellow. Finally, the meaning of traffic sign is basically acquired using the pictograph in middle. Even though the design is clear and discriminative for a human driver, there are challenging problems in real world applications such as shadow, distance, weather condition and age of signs that need to be addressed in the traffic sign recognition systems.

The common pipeline in recognizing objects is to extract some features for each object and train a model to classify them using the extracted features. Conventionally, features are extracted using hand-crafted methods such as HOG, SIFT, BoW, Gabor, LBP and Fisher Vectors. These methods transform an image into a space where classes of objects are properly separable. In large-scale object recognition, objects are likely to be non-linearly separable using these feature extraction methods. As the result, a non-linear

* Corresponding author.
E-mail addresses: hamed.habibi@urv.cat (H. Habibi Aghdam), elnaz.jahani@urv.cat (E. Jahani Heravi), domenec.puig@urv.cat (D. Puig).

http://dx.doi.org/10.1016/j.robot.2016.07.003
0921-8890/© 2016 Elsevier B.V. All rights reserved.

model such as SVM, Random Forest or Gaussian Process is required to learn non-linear decision boundaries in this space.

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.

Another solution is to automatically learn a function to transform the image into a feature space in which classes are linearly separable. A Convolutional Neural Network (ConvNet) is a non-linear function which learns to extract these kinds of features. Krizhevsky et al. [1] developed a ConvNet to classify 1000 classes inside the ImageNet dataset that was more accurate than methods based on hand-crafted features. More recently, He et al. [2] designed a ConvNet which surpassed human performance on classification of objects in the ImageNet dataset.

Moreover, ConvNets classified traffic signs more accurately than a human expert on a practical dataset called German Traffic Sign Benchmark [3]. To be more specific, the winner algorithm proposed by Cireşan et al. [4] computes the average score of 25 ConvNets with the same architecture trained on 5 variations of the original dataset. In addition, the second place was also awarded to another ConvNet proposed by Sermanet and Lecun [5]. In contrast to the winner, it uses only one ConvNet to recognize the traffic signs. In fact, the data-augmentation procedure utilized by both teams is almost identical and the only difference is the architecture of the networks. A single ConvNet of the winner team requires optimizing 1,543,443 parameters. However, the runner-up team trains a network with 1,437,791 parameters. Last not the least, the first network uses the hyperbolic tangent activation function and the second network utilizes the rectified sigmoid activation function. Despite their high accuracies, both networks are not computationally efficient and they require a large amount of multiplications on the hardware. Taking into account the fact that an ADAS has a limited computational power and it must perform various tasks, it is crucial that the traffic sign recognition module consumes as few processing time as possible in order to release the processing unit immediately.

With this goal in mind, we proposed a ConvNet with much less number of arithmetic operations and more accuracy compared with the above ConvNets [6]. The intuition behind our ConvNet is to divide the middle convolution layers into two separate layers in order to halve the third dimension of the convolution kernels and to utilize the Leaky Rectified Linear Units (LReLU) [7] activation functions instead of more computational activations functions such as the hyperbolic function. Visualizing the sensitivity of the classification score suggested that our ConvNet classifies the traffic signs using mainly their pictograph and shape. Based on this results, we can further optimize the architecture of the ConvNet to reduce its *time-to-completion*.

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. We extensively review the state of art methods for classification of traffic signs in Section 2.

Contribution: In this paper, we propose a lightweight and optimized ConvNet for detecting traffic signs (Section 3). Then, the architecture of the ConvNet is modified by utilizing dilated convolutions to implement the sliding window detection approach using only single forward pass of the ConvNet on a high-resolution image. We show that our approach is able to process 37.72 high resolution images (1020×600 pixels in 5 scales) per second in order to detect traffic signs using a GPU (GeForce GTX 980). Moreover, we propose a more optimized and more accurate ConvNet for classifying traffic signs and establish a new record on the GTSRB dataset (Section 3). Finally, Section 4 shows the experimental results on German Traffic Sign Benchmark. Besides, it studies the stability of the proposed classification ConvNet and examines the behavior of the ConvNets by a visualization technique.

2. Related work

In general, works on detection and classification of traffic signs can be divided into the *traditional classification* and the *end-to-end learning* methods. The basic idea in the former approach is to design an algorithm for extracting features of the image and train a classifier on top of them. Features that are extracted in this way are usually called *hand-crafted* features. Indeed, the overall accuracy of the traditional methods mainly depends on the feature extraction algorithm since there are powerful classifiers such as SVM or Random Forest that are able to accurately learn non-linear decision boundaries. However, if classes overlap in the feature space (due to the feature extraction algorithm), the classifiers will not be able to accurately discriminate the classes. In contrast, the *end-to-end* learning methods such as Convolutional Neural Networks (ConvNets) learn a highly non-linear function to project the raw image into a feature space where classes are linearly separable and there is no overlap between classes. In this section, we first review the traditional classification approaches and then we explain the previously proposed end-to-end learning methods for recognizing traffic signs.

Hand-crafted features: Paclík et al. [8] produce a binary image depending on the color of the traffic sign. Then, several moment invariant features are extracted from the binary image and fetched into a one-versus-all Laplacian kernel classifier. One problem with this method is that the query image must be binarized before fetching into the classifier. Maldonado-BascÓn et al. [9] addressed this problem by transforming the image into the HSI color space and calculating the histogram of Hue and Saturation components. Then, the histogram is classified using a multi-class SVM. In another method, Maldonado-BascÓn et al. [10] recognized traffic signs using only the pictograph of each sign. Although the pictograph is a binary image, however, accurate segmentation of a pictogram 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. [10] trained SVM where the input is a 31×31 block of pixels in the gray-scale version of pictograph.

Before 2011, there was not a public and challenging dataset to be used by the research community. Timofte and Van Gool [11], Larsson and Felsberg [12] and Stallkamp et al. [3] introduced three challenging databases including annotations. These database are called Belgium Traffic Sign Classification (BTSC), Swedish Traffic Sign and German Traffic Sign Recognition Benchmark (GTSRB) databases, respectively. In particular, GTSRB was used in a competition and, as we will discuss shortly, the winner method classified 99.46% of test images correctly [3]. Zaklouta and Stanciulescu

[13–15] extracted Histogram of Oriented Gradient (HOG) descriptors with three different configurations for representing the image and trained a Random Forest and a SVM for recognizing traffic sings in GTSRB dataset. Similarly, Greenhalgh and Mirmehdi [16], Moiseev et al. [17], Huang et al. [18], Mathiaset al. [19] and Sunet al. [20] utilized HOG descriptor. The main difference between these works lies in the utilized classification model (*e.g.* SVM, Cascade SVM, Extreme Learning Machine, Nearest Neighbour and LDA). These works except [18] use a traditional classification approach. In contrast, Huang et al. [18] utilize a two level classification. In the first level, the image is classified into one of superclasses. 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.

Fleyeh and Davami [21] projects the image into the principal component space and find the class of the image by computing the Euclidean distance of the projected image with the images in the database. Yuan et al. [22] proposed a novel feature extraction method that was able to effectively combine color, global spatial structure, global direction structure and local shape information. Besides, *sparse coding* is also used as the feature extraction method. Hsu and Huang [23] coded each traffic sign using the Matching Pursuit algorithm. Liu et al. [24] 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 [25]. Recently, we proposed a method based on visual attributes and Bayesian network [26]. In this method, we described each traffic sign in terms of visual attributes. In order to detect visual attributes, we divide the input image into several regions and code each region using the Elastic Net Sparse Coding method. Finally, attributes are detected using a Random Forest classifier. The detected attributes are further refined using a Bayesian network.

Discussion: Hand-crafted features such as HOG has a limited representation power and they might not be accurate on challenging datasets. This can be observed by the results reported in GTSRB competition [3] where the best performed solution based on a hand-crafted feature proposed by Zaklouta et al. [13] correctly classifies 96.14% of test samples whilst the classification accuracy of ensemble of ConvNets in [3] is 99.46% on the same test samples. In other words, we gain 3.32% more accuracy by using a more complex feature extraction model. Notwithstanding, we must taking this fact into account that our framework will be applicable on autonomous car if it performs equal or better than a human driver. If an autonomous car is not able to recognize traffic sign as accurate as a human driver (*e.g* the above hand-crafted feature), it may cause vital consequences. Even though the aforementioned ConvNet is more complex than the hand-crafted features, it surpasses human performance on a challenging dataset. This makes the traffic sign recognition model applicable in real-world applications. Moreover, hand-crafted features might not scale well if the number of traffic sign classes increases. This is due to the fact that they have a limited representation power and traffic sign classes might overlap if their number increases.

ConvNets: ConvNets were first used by Sermanet and Lecun [5] and Cireşan et al. [27,4] in the field of traffic sign recognition during the GTSRB competition where the proposed ConvNet in [4] surpassed human performance and won the competition by correctly classifying 99.46% of test image. Moreover, the ConvNet proposed in [5] ended up in the second place with a considerable difference compared with the 3rd place which was awarded for a method based on the traditional classification approach. The classification accuracies of the runner-up and the 3rd place were 98.97% and 96.14%, respectively.

Both [5,4] augment the data by applying some transformations on the training dataset. The augmentation procedure is relatively similar but the architectures of the two ConvNets are completely different. The winner [4] constructs an ensemble of 25 ConvNets each consists of 1,543,443 parameters. The runner-up [5] 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. However, it is not clearly mentioned in [5] 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. [28] proposed a new architecture including 1,162,284 parameters and utilizing Rectified Linear Unit (ReLU) activations [1]. In addition, there is a Local Response Normalization (LRN) layer after each activation layer. They build an ensemble of 20 ConvNets and established a new record by classifying 99.65% of test images correctly. Although the number of parameters is reduced using this architecture compared with the two networks, notwithstanding, the ensemble is constructed using 20 ConvNets which is still not computationally efficient in real-world applications. It is worth mentioning that the number of memory accesses are very high in [28] due to the LRN layers and zero-padding the feature maps before applying the convolution kernels. Despite the impressive results obtained by the three ConvNets, their response under different circumstances are not clear. In addition, they have only reported the classification accuracy which is not a promising measure for accurately evaluating the results.

We partially addressed these problems by proposing a ConvNet with much less number of arithmetic operations and better accuracy [6]. The basic idea behind our ConvNet was to utilize the LReLU as the activation function which is computationally very efficient and divide the middle convolution layers into two separate layers in order to halve the third dimension of the convolution kernels. We also showed that our ConvNet basically discriminates the traffic signs using their pictographs.

There is still another problem with the all above ConvNets. These ConvNets are designed to identify traffic signs using a 48×48 image and they cannot be directly used for detecting traffic signs on a high-resolution image. Indeed, researchers are actively proposing new approaches for utilizing ConvNet in the task of *object detection*.

Sermanet et al. [29] proposed a method for implementing a multi-scale sliding window approach within a ConvNet. Szegedy et al. [30] formulated the object detection problem as a regression problem to object bounding boxes. Girshick et al. [31] proposed a method so called Region with ConvNets in which they apply ConvNet to bottom-up region proposals for detecting the domain specific objects. Recently, Ouyanget al. [32] developed a new pooling technique called *deformation constrained pooling* to model the deformation of object parts with geometric constraint.

In this paper, we propose two ConvNets for detection and classification of traffic signs. Our ultimate goal is to develop more *accurate* ConvNets that can be executed in *real-time*. More specifically, we build a detection ConvNet and then re-design it using only convolution and pooling layers so they can be applied on images with arbitrary sizes. In addition, we analyze our previously proposed classification ConvNet and propose a more accurate ConvNet with less time-to-completion.

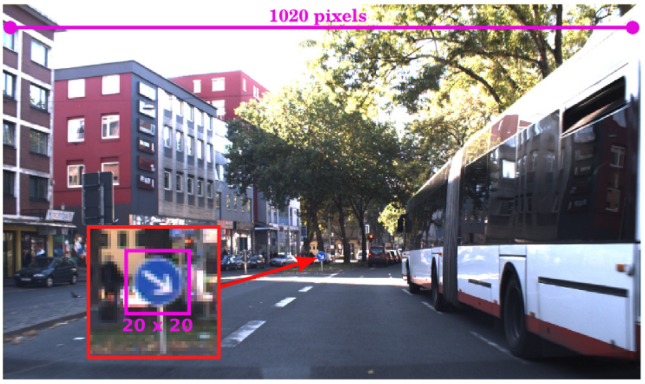


Fig. 1. The detection module must be applied on a high resolution image.

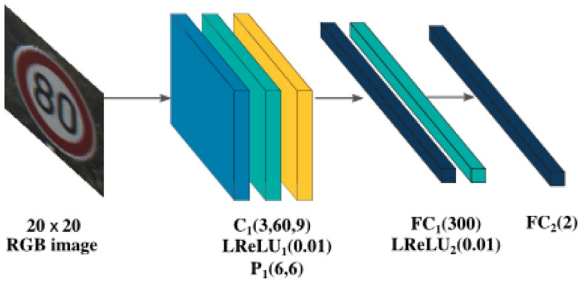


Fig. 2. Proposed ConvNet for detecting traffic signs. The blue, green and yellow color indicate a convolution, LReLU and pooling layer, respectively. $C(c, n, k)$ denotes n convolution kernel of size $k \times k \times c$ and $P(k, s)$ denotes a max-pooling layer with pooling size $k \times k$ and stride s. Finally, the number in the LReLU units indicate the leak coefficient of the activation function. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3. Proposed method

As we mentioned earlier, a traffic sign recognition system consists of a detection and a classification module. In this section, we propose a lightweight ConvNet for detecting traffic signs in an image and explain how to implement a real-time sliding window detector within the ConvNet. Then, we analyze our previously proposed ConvNet [6] for classifying traffic signs and propose a new architecture with higher accuracy and lower time-to-completion.

3.1. The detection module

In contrast to offline applications, an ADAS requires algorithms that are able to perform their task in real-time. On the other hand, the detection module consumes more time compared with the classification module especially when it is applied on a high-resolution image. For this reason, the detection module must be able to locate traffic signs in real-time. However, the main barrier in achieving this speed is that the detection module must analyze high-resolution images in order to be able to locate traffic signs that are in a distance up to 25 m. This is illustrated in Fig. 1 in which the width of the image is 1020 pixels.

Assuming that the length of the bus is approximately 12.5 m, we can estimate that the distance between the traffic sign indicated by the arrow and the camera is approximately 20 m. Although the distance is not large, the bounding box of the sign is nearly 20×20 pixels. Consequently, it is impractical to apply the detection algorithm on low-resolution images since signs that are located in 20 m of distance from the camera might not be recognizable. Moreover, considering that the speed of the car is 80 km/h in an interurban road, it will travel 22 m in one second. For this reason, the detection module must be able to analyze more than one frame per second in order to be able to deal with high speeds motions of a car.

In practice, a car might be equipped with a stereo camera. In that case, the detection module can be applied much faster since most of the non-traffic sign pixels can be discarded using the distance information. In addition, the detection module can be calibrated on a specific car and use the calibration information to ignore non-traffic sign pixels. In this work, we propose a more general approach by considering that there is only one color camera that can be mounted in front of any car. In other words, the detection module must analyze all the patches on the image in order to identify the traffic signs.

We trained separate traffic sign detectors using HOG and LBP features together with a linear classifier and a random forest classifier. However, our experiments showed that the detectors based on these features suffer from low precision and recall values. More importantly, applying these detectors on a high-resolution image using a CPU is impractical since it takes a long time to process the whole image. Besides, it is not trivial to implement the whole scanning window approach using these detectors on a GPU in order to speed up the detection process. For this reason, we developed a lightweight but accurate ConvNet for detecting traffic signs. Fig. 2 illustrates the architecture of the ConvNet.

Our ConvNet is inspired by Gabor feature extraction algorithm. In this method, a bank of convolution kernels is applied on image and the output of each kernel is individually aggregated. Then, the final feature is obtained by concatenating the aggregated values. Instead of hand-crafted Gabor filters, the proposed ConvNet learns a bank of 60 convolution filters each with $9 \times 9 \times 3$ coefficients. The output of the first convolution layer is a $12 \times 12 \times 60$ tensor where each slice is a feature map (*i.e.* 60 feature maps). Then, the aggregation is done by spatially dividing each feature map into 4 equal regions and finding the maximum response in each region. Finally, the extracted feature vector is non-linearly transformed into a 300 dimensional space where the ConvNet tries to linearly separate the two classes (traffic sign vs. non-traffic sign) in this space.

One may argue that we could attach a fully connected network to HOG features (or other hand crafted features) and train an accurate detection model. Nonetheless, there are two important issues with this approach. First, it is not trivial to implement a sliding window detector using these kind of features on a GPU. Second, as we will show in our experiments, their representation power is limited and they produce more false-positive results compared with our ConvNet.

To train the ConvNet, we collect the positive samples and pick some image patches randomly in each image as the negative samples. After training the ConvNet using this dataset, it is applied on each image in the training set using the *multi-scale sliding window* technique in order to detect traffic signs. Fig. 3 illustrates the result of applying our ConvNet on an image.

The red, the blue and the green rectangles show the false-positive, ground-truth and the true-positive patches. We observe that the ConvNet is not very accurate and it produces some false-positive results. Although the false-positive rate can be reduced by increasing the threshold value of the classification score, our aim is to increase the overall accuracy of the ConvNet.

There are mainly two solutions to improve the accuracy. Either to increase the complexity of the ConvNet or refine the current model using more appropriate data. Increasing the complexity of the ConvNet is not practical since it will also increase its time-to-completion. However, it is possible to refine the model using more appropriate data. Here, we utilize the *hard-negative mining* method.

In this method, the current ConvNet is applied on all the training images. Next, all the patches that are classified as positive (the

Fig. 3. Applying the trained ConvNet for hard-negative mining. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

red and the green boxes) are compared with the ground-truth bounding boxes and those which do not align well are selected as the new negative image patches (the red rectangles). They are called *hard-negative* patches. Having the all hard-negative patches collected from all the training images, the ConvNet is fine-tuned on the new dataset. Mining hard-negative data and fine-tuning the ConvNet can be done repeatedly until the accuracy of the ConvNet converges.

3.1.1. Implementing sliding window within the ConvNet

The detection procedure starts with sliding a 20×20 mask over the image and classifying the patch under the mask using the detection ConvNet. After all the pixels are scanned, the image is down-sampled and the procedure repeats on the smaller image. The down-sampling can be done several times to ensure that the closer objects will be also detected. Applying this simple procedure on a high-resolution image may take several minutes (even on GPU because of redundancy in computation and transferring data between the main memory and GPU). For this reason, we need to find an efficient way for running the above procedure in real-time.

Currently, advanced embedded platforms such as NVIDIA Drive Px[1] come with a dedicated GPU module. This makes it possible to execute highly computational models in real-time on these platforms. Therefore, we consider a similar platform for running the tasks of ADAS. Based on this configuration, our detection and classification module are executed on a GPU.

There are two main computational bottlenecks in naive implementation of the sliding window detector. On the one hand, the input image patches are very small and they may use small fraction of GPU-cores to complete a forward pass in the ConvNet. In other words, two or more image patches can be simultaneously processed depending on the number of GPU-cores. However, the aforementioned approach considers the two consecutive patches are independent and applies the convolution kernels on the each patches, separately. On the other hand, transferring overlapping image patches between the main memory and GPU is done thousands of time which adversely affects the time-to-completion. To address these two problems, we propose the following approach for implementing the sliding window method on a GPU. Fig. 4 shows the intuition behind our implementation.

Normally, the input of the ConvNet is a $20 \times 20 \times 3$ image and the output of the pooling layer is a $2 \times 2 \times 60$ tensor. Also, each neuron in the first fully connected layer is connected to $2 \times 2 \times 60$ neurons in the previous layer. In this paper, traffic signs are detected in a 1020×600 image. Basically, sliding window approach scans every pixel in the image to detect the objects.[2] In other words, for each pixel in the image, the first step is to crop a 20×20 patch and, then, to apply the bank of the convolution filters in the ConvNet on the patch. Next, the same procedure is repeated for the pixel next to the current pixel. Note that 82% of the pixels are common between two consecutive 20×20 patches. As the result, transferring the common pixels to GPU memory is redundant. The solution is that the whole high-resolution image is transferred to the GPU memory and the convolution kernels are applied on different patches simultaneously.

The next step in the ConvNet is to aggregate the pixels in the output of the convolution layer using the max-pooling layer. When the ConvNet is provided by a 20×20 image, the convolution layer generates a 12×12 feature map for each kernel. Then, the pooling layer computes the maximum values in 6×6 regions. The distance between each region is 6 pixels (stride=6). Therefore, the output of the pooling layer on single feature map is a 2×2 feature map. Our aim is to implement the sliding window approach within the ConvNet. Assume the two consecutive patches indicated by the red and green rectangles in the convolution layer (Fig. 4). The pooling layer will compute the maximum value of 6×6 regions. Based on our original ConvNet, the output of the pooling layer for the red rectangle must be computed using the 4 small red rectangles illustrated in the middle figure.

In addition, we also want to aggregate the values inside the green region in the next step. Its corresponding 6×6 regions are illustrated using 4 small green rectangles. Since we need to apply the pooling layer consecutively, we must change the stride of the pooling layer to 1. With this formulation, the pooling result of the red and green regions will not be consecutive. Rather, there will be 6 pixels gap between the two consecutive non-overlapping 6×6 regions. The pooling results of the red rectangle are shown using 4 small filled squares in the figure. Recall from the above discussion that each neuron in the first fully connected layer is connected to $2 \times 2 \times 60$ regions in the output of the pooling layer.

Based on the above discussion, we can implement the fully-connected layer using $2 \times 2 \times 60$ *dilated convolution filters* with dilation factor 6. Formally, a $W \times H$ convolution kernel with dilation factor τ is applied using the following equation:

$$(f(m,n) * g)^{\tau} = \sum_{h=-\frac{H}{2}}^{\frac{H}{2}} \sum_{w=-\frac{W}{2}}^{\frac{W}{2}} f(m+\tau h, n+\tau w) g(h,w). \quad (1)$$

Note that the number of the arithmetic operations on a normal convolution and its dilated version is identical. In other words, dilated convolution does not change the computational complexity of the ConvNet. Likewise, we can implement the last fully connected layer using $1 \times 1 \times 2$ filters. Using this formulation, we are able to implement the sliding window method in terms of convolution layers. The architecture of the sliding window ConvNet is shown in Fig. 5.

The output of the fully convolutional ConvNet is a $y' = \mathbb{R}^{1012 \times 592 \times 2}$ where the patch at location (m, n) is a traffic sign if $y'(m, n, 0) > y'(m, n, 1)$. It is a common practice in the sliding window method to process patches every 2 pixels. This is easily implementable by changing the stride of the pooling layer of the fully convolutional ConvNet to 2 and adjusting the dilation factor of convolution kernel in the first fully-connected layer accordingly (*i.e.* $s = 3$). Finally, to implement the multi-scale sliding window, we only need to create different scales of the original image and

[1] www.nvidia.com/object/drive-px.html.

[2] We may set the stride of scanning to 2 or more for computational purposes.

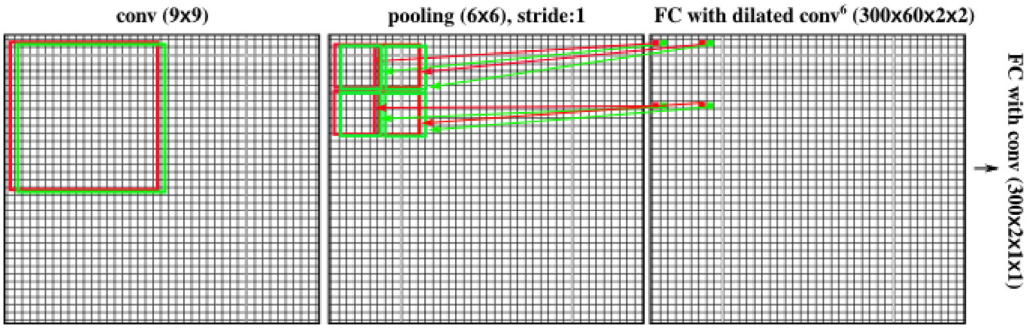


Fig. 4. Implementing the sliding window detector within the ConvNet.

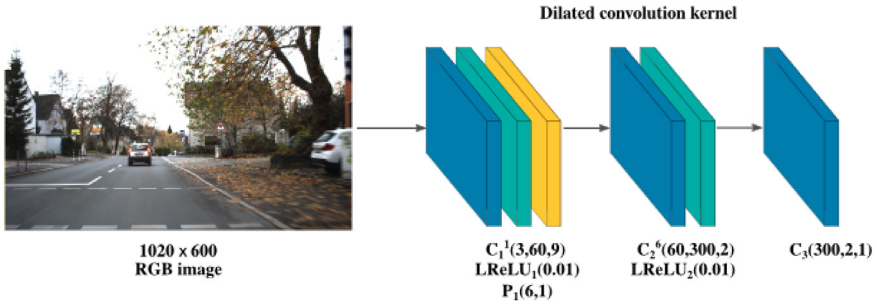


Fig. 5. The architecture of the sliding window ConvNet.

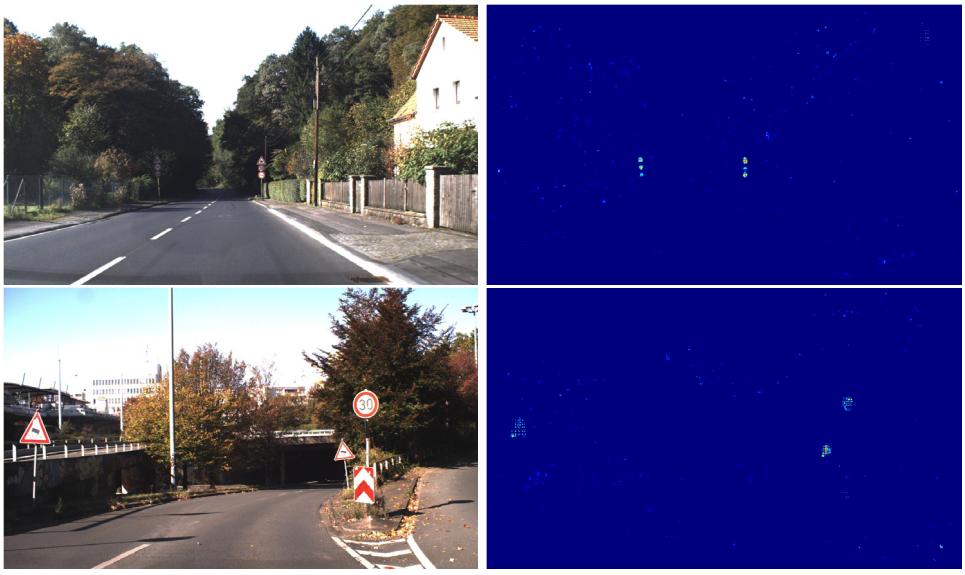


Fig. 6. Detection score computed by applying the fully convolutional sliding network to 5 scales of the high resolution image.

apply the sliding window ConvNet ot it. Fig. 6 shows the detection score computed by our ConvNet on high resolution images.

Detecting traffic signs in high resolution images is the most time consuming part of the processing pipeline. For this reason, we executed the sliding ConvNet on a GeForce GTX 980 card and computed the time-to-completion of each layer, separately. To be more specific, each ConvNet repeats the forward-pass 100 times and the average time-to-completion of each layer is computed. The condition of the system is fixed during all calculations. Table 1 shows the results of the sliding ConvNet on 5 different scales. Recall from our previous discussion that the stride of the pooling layer is set to 2 in the sliding ConvNet.

We observe that applying the sliding ConvNet on 5 different scales of a high resolution image takes 62.266 ms in total which is equal to processing of 19.13 frames per second. We also computed time-to-completion of the sliding ConvNet by changing the stride of the pooling layer from 1 to 4. Fig. 7 illustrates time-to-completion per image resolution (left) as well as the cumulative time-to-completion.

The results reveal that it is not practical to set the stride to 1 since it takes 160 ms to detect traffic sign on an image (6.25 frames per second). In addition, it consumes a considerable amount of GPU memory. However, it is possible to process 19 frames per second by setting the stride to 2. In addition, the reduction in the processing time between stride 3 and stride 4 is negligible. But, stride 3 is preferable compared with stride 4 since it produces a denser detection score. Last but not the least, it is possible to apply a combination of stride 2 and stride 3 in various scales to improve the overall time-to-completion. For instance, we can set the stride to 3 for first scale and set it to 2 for rest of the image scales. By this

Table 1
Time-to-completion (milliseconds) of the sliding ConvNet computed on 5 different image scales.

Layer name	Per layer time to completion in milliseconds for different scales				
	1020 × 600	816 × 480	612 × 360	480 × 240	204 × 120
Data	0.167	0.116	0.093	0.055	0.035
Conv	7.077	4.525	2.772	1.364	0.321
Relu	1.744	1.115	0.621	0.330	0.067
Pooling	6.225	3.877	2.157	1.184	0.244
Fully connected	8.594	5.788	3.041	1.606	0.365
Relu	2.126	1.379	0.746	0.384	0.081
Classify	1.523	0.893	0.525	0.285	0.101
Total	27.656	17.803	10.103	5.365	1.336

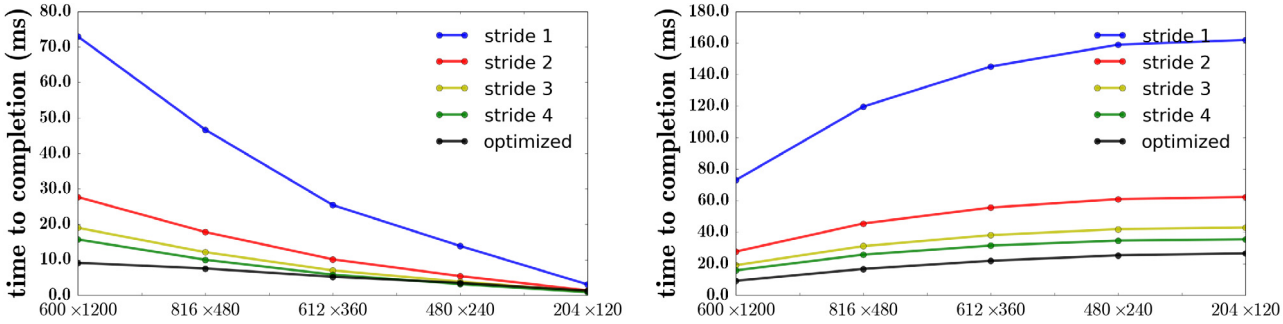


Fig. 7. Time-to-completion of the sliding ConvNet for different strides. (Left) Time-to-completion per resolution and (Right) cumulative time-to-completion.

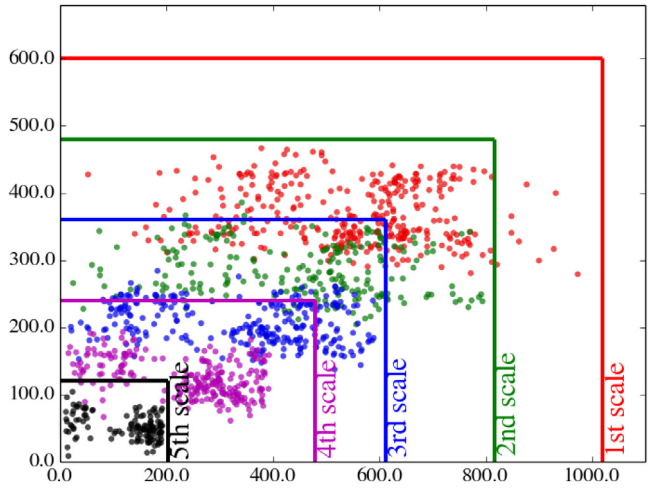


Fig. 8. Distribution of traffic signs in different scales computed using the training data.

way, we can save about 10 ms per image. The execution time can be further improved by analyzing the database statistics.

More specifically, traffic signs bounded in 20 × 20 regions will be detected in the first scale. Similarly, signs bounded in 50 × 50 regions will be detected in the 4th scale. Based on this fact, we divided traffic signs in training set into 5 groups according to the image scale they will be detected. Fig. 8 illustrates the distribution of traffic signs in each scale.

According to this distribution, we must expect to detect 20 × 20 traffic signs inside a small region in the first scale. That said, the region between row 267 and row 476 must be analyzed to detect 20 × 20 signs rather than whole 600 rows in the first scale. Based on the information depicted in the distribution of signs, we process only fetch the 945 × 210, 800 × 205, 600 × 180 and 400 × 190 pixels in the first 4 scales to the sliding ConvNet. As it is illustrated by a black line in Fig. 7, this reduces the time-to-completion of the ConvNet with stride 2–26.506 ms which is equal to processing 37.72 high resolution frames per second.

3.2. The classification module

Having the signs detected on the image, the next step is to classify them into one of traffic sign classes. We have previously proposed an accurate and fast ConvNet for this purpose [6]. We also showed that classification of traffic signs is mainly done using the shape and the pictograph of traffic signs. Therefore, it is possible to discard color information and use only gray-scale pixels to learn a representation by the ConvNet.

Our aim is to build a more accurate and less computational ConvNet for classification of traffic signs. To this end, we computed the layer-wise time-to-completion of our previously proposed ConvNet [6]. Table 2 shows the results in milliseconds. We observe that the two middle layers with 4 × 4 kernels consume most of the GPU time. Moreover, the fully connected layer does not significantly affect the overall time-to-completion. Likewise, the first convolution layer can be optimized by reducing the size of the kernel and number of the input channels.

From accuracy perspective, our aim is to reach a accuracy higher than our previously proposed method. The basic idea behind ConvNets is to learn a representation which makes objects linearly separable in the last layer. Fully connected layers facilitates this by learning a non-linear transformation to project the representation into another space. We can increase the degree of freedom of the ConvNet by adding more fully connected layers to it. This may help to learn a better linearly separable representation. Based these ideas, we proposed the ConvNet illustrated in Fig. 9.

First, we replaced color image with gray-scale image in our ConvNet. In addition, because a gray-scale image is a single channel input, the linear transformation layer in [6] must be also discarded. Second, we have utilized Parametric Rectified Linear Units [2] (PReLU) to learn separate α_i for each feature map in a layer where α_i depicts the value of leaking parameter in LReLU. Third, we have added another fully connected layer to the network to increase its flexibility. Fourth, the size of the first kernel and the middle kernels have been reduced to 5 × 5 and 3 × 3, respectively. Last but not the least, the size of the input image is reduced to 44 × 44 pixels compared with [6] to reduce the dimensionality of the feature

Table 2
Per layer time-to-completion (milliseconds) of the our previous classification ConvNet.

Layer	Data	c1 × 1	c7 × 7	pool1	c4 × 4	pool2	c4 × 4	pool3	fc	Class
Time (ms)	0.032	0.078	0.082	0.025	0.162	0.013	0.230	0.013	0.062	0.032

Table 3
Per layer time-to-completion (milliseconds) of the our proposed classification ConvNet.

Layer	Data	c5 × 5	pool1	c3 × 3	pool2	c3 × 3	pool3	fc1	fc2	Class
Time (ms)	0.036	0.076	0.0166	0.149	0.0180	0.159	0.0128	0.071	0.037	0.032

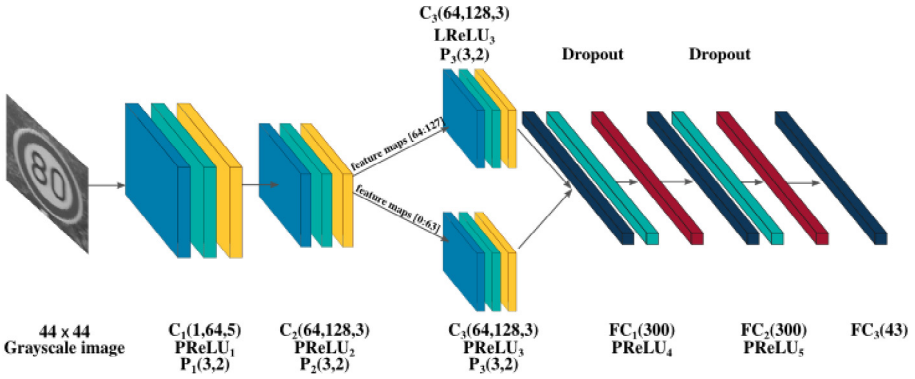


Fig. 9. The slightly modified ConvNet architecture compare with our previous study [6].

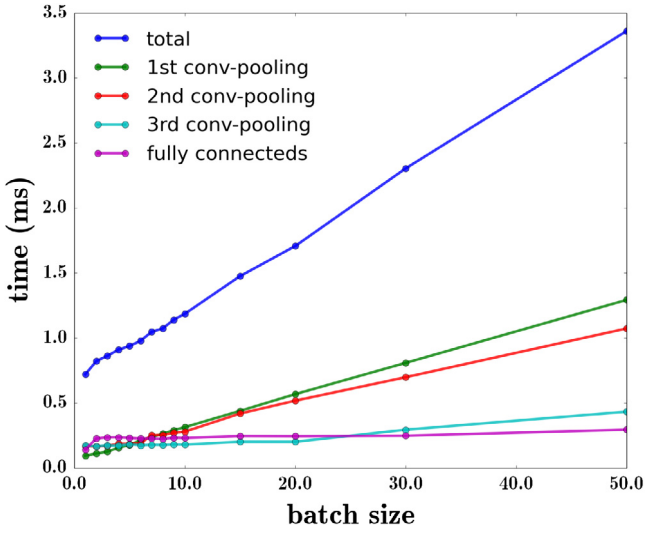


Fig. 10. Relation between the batch size and time-to-completion of the classification ConvNet.

vector in the last convolution layer. Table 3 shows the layer-wise time-to-completion of our proposed ConvNet.

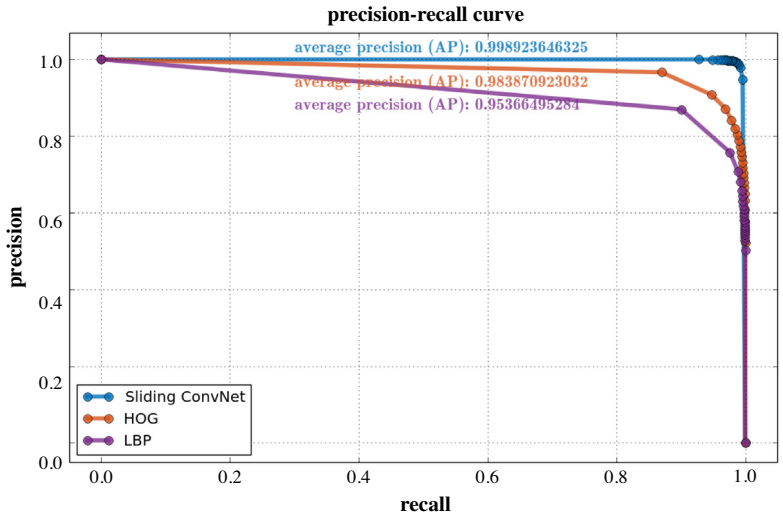
According to this table, time-to-completion of the middle layers has been reduced. Especially, time-to-completion of the last convolution layer has been reduced substantially. In addition, the ConvNet has saved 0.078 ms by removing *c1x1* layer in [6]. It is worth mentioning that the overhead caused by the second fully connected layer is slight. In sum, the overall time-to-completion of the proposed ConvNet is less than [6]. Finally, we investigated the effect of batch size on the time-to-completion of ConvNet. Fig. 10 illustrates the relation between the batch size of the classification ConvNet and its time-to-completion.

According to the figure, while processing 1 image takes approximately 0.7 ms using the classification, processing 50 images takes approximately 3.5 ms using the same ConvNet (due to parallel architecture on the GPU). In other words, if the detection ConvNet generates 10 samples, we do not need to enter the samples one by one to the ConvNet. This will take $0.7 \times 10 = 7$ ms to complete. Instead, we can fetch a batch of 10 samples to the network and process them in approximately 1 ms. By this way, we can save more GPU time.

4. Experiments

We applied our architecture on the German Traffic Sign Recognition Benchmark (GTSRB) and the German Traffic Sign Detection Benchmark (GTSDB) [3]. The original color images in the GTSRB dataset contain one traffic sign each and they vary in size from 15 × 15 to 250 × 250 pixels. The training set consists of 39,209 images and the test set contains 12,630 images. We crop all the images and process only the image within the bounding box (The bounding box is part of the annotation). Then, we resize (downsample or upsample) the all images to 48 × 48 pixels. Finally, the mean image is obtained over the training set and it is subtracted from each image in the dataset since previous studies [33] show that subtracting the mean image increases the performance of the network. Also, the GTSDB dataset consists of 900 images (600 training and 300 test) with annotation of bounding boxes. Size of all the images is 1360 × 800 and they contain at least one traffic sign. All the images are resized to 1020 × 600 before further processing. Finally, both the sliding ConvNet and the classification ConvNet have been implemented using [34].

Data augmentation: To reduce the effect of over-fitting, we augment the GTSRB dataset by applying 12 transformations on the images as follows: (1) *Translation*: We perturb the images in positions [2, 2], [2, −2], [−2, 2] and [−2, −2]. (2) *Smoothing*: The images are smoothed using a 5 × 5 Gaussian filter and a 5 × 5 median filter. (3) *HSV manipulation*: The saturation of images are modified by transforming the images into the HSV color space and scaling the saturation component by factors 0.9 and 1.1. In another transformation, the value component is scaled by factor 0.7. (4) *PCA*: First, the matrix of principal components $P \in \mathbb{R}^{3\times3}$ of the



threshold	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99
precision	94.85	98.78	99.31	99.53	99.65	99.72	99.77	99.84	99.89	99.93	99.98
recall	99.57	98.96	98.61	98.25	97.92	97.62	97.23	96.99	96.49	95.69	92.74

Fig. 11. (Top) Precision–recall curve of the detection ConvNet along with models obtained by HOG and LBP features. (Bottom) Numerical values (%) of precision and recall for the detection ConvNet. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

pixels of the training dataset is computed. Then, pixels of the input image are projected onto the principal components to obtain the coefficient vector b. Next, a 3×1 vector v is randomly generated in range $[0, 0.5]$ and the image is reconstructed by back-projecting the pixels onto RGB color space using the new coefficient vector $b' = b - b \odot v$ where $\odot$ is an element-wise multiplication operator. (5) *Other*: The dataset is further augmented by applying histogram equalization transformation and sharpening the images.

4.1. Detection

The detection ConvNet is trained using the mini-batch stochastic gradient descent (batch size $= 50$) with learning rate annealing. We fix the learning rate to 0.02, momentum to 0.9, L2 regularization to 10^{-5}, step size of annealing to 10^4, annealing rate to 0.8, the negative slope of the LReLU to 0.01 and the maximum number of iterations to 150,000. The ConvNet is first trained using the ground-truth bounding boxes (the blue boxes) and the negative samples collected randomly from image. Then, a hard-negative mining is performed on the training set in order to collect more organized negative samples and the ConvNet is trained again. To compare with hand-crafted features, we also trained detection models using HOG and LBP features and Random Forests by following the same procedure. Fig. 11 illustrates the precision–recall plot of these models.

Average precision (AP) of the sliding ConvNet is 99.89% which indicates a nearly perfect detector. In addition, average precision of the models based on HOG and LBP features are 98.39% and 95.37%, respectively. Besides, precision of the sliding ConvNet is considerably higher than HOG and LBP features. In other words, the number of the false-positive samples in the sliding ConvNet is less than HOG and LBP. It should be noted that false-positive results will be directly fetched into the classification ConvNet where they will be classified into one of traffic sign classes. This may produce dangerous situations in the case of autonomous cars. For example, consider a false-positive result produced by detection module of an autonomous car is classified as "speed limit 100" in an educational zone. Clearly, the autonomous car may increase the speed according to the wrongly detected sign. This may have vital consequences in real world. Even though average precision on the sliding ConvNet and HOG models are *numerically* comparable, using the sliding ConvNet is certainly safer and more applicable than the HOG model.

Post processing bounding boxes: One solution to deal with the *false-positive* results of the detection ConvNet is to post-process the bounding boxes. The idea is that if a bounding box is classified positive, all the bounding boxes in distance of $\{-1, 0, -1\} \times \{-1, 0, -1\}$ must be also classified positive. In other words, if a region of the image consists of a traffic sign, there must be at least 10 bounding boxes over that region in which the detection ConvNet classifies them positive. By only applying this technique, the false-positive rate can be considerably reduces. Fig. 12 illustrates the results of the detection ConvNet on a few images before and after post-processing the bounding boxes.

In general, the detection ConvNet is able to locate traffic signs with high precision and recall. Furthermore, post-processing the bounding boxes is able to effectively discard the false-positive results. However, a few false-positive bounding boxes may still exist in the result. In practice, we can create a second step verification by creating a ConvNet with more complexity and apply it on the results from the detection ConvNet in order to remove all the false-positive results. In fact, this is one of our goals for the future work.

4.2. Classification

The classification ConvNet is also trained using the mini-batch stochastic gradient descent (batch size $=$ 50) with exponential learning rate annealing. We fix the learning rate to 0.02, momentum to 0.9, L2 regularization to 10^{-5}, annealing parameter to 0.99996, dropout ratio to 0.5 and initial value of leaking parameters to 0.01. The network is trained 12 times and their classification accuracies on the test set are calculated. It is worth mentioning that [3] has only reported the classification accuracy and it is the only way to compare our results with [4] and [5].

Table 4 shows the result of training 10 ConvNets with the same architecture and different initializations. The average classification accuracy of the 10 trials is 99.34% which is higher than the average

Fig. 12. Output of the detection ConvNet before and after post-processing the bounding boxes. A darker bounding box indicate that it is detected in a lower-scale image.

Table 4
Classification accuracy of the single network.

Trial	top-1 acc. (%)	top-2 acc. (%)	Trial	Top 1 acc. (%)	Top 2 acc. (%)
1	99.21	99.77	6	99.54	99.78
2	99.38	99.73	7	99.25	99.70
3	99.55	99.83	8	99.21	99.73
4	99.16	99.72	9	99.53	99.82
5	99.35	99.75	10	99.24	99.64
Average top-1		99.34 ± 0.02	Average top-2		99.75±0.002
Human top-1		98.84	Human top-2		NA

human performance reported in [3]. In addition, the standard deviation of the classification accuracy is small which shows that the proposed architecture trains the networks with very close accuracies. We argue that this stability is the result of reduction in number of the parameters and regularizing the network using a dropout layer. Moreover, we observe that the top-2[3] accuracy is very close in all trials and their standard deviation is 0.002. In other words, although the difference in top-1 accuracy of the Trial 5 and the Trial 6 is 0.19%, notwithstanding, the same difference for top-2 accuracy is 0.03%. This implies that some cases are always within the top-2 results. In other words, there are images that have been classified correctly in Trial 5 but they have been misclassified in Trial 6 (or vice versa). As a consequence, if we fuse the scores of two networks the classification accuracy might increase.

Based on this observation, an ensemble was created using the method mentioned in [35]. The created ensemble consists of 3 ConvNets (ConvNets 5, 6 and 9 in Table 4). As it is shown in Table 5, the overall accuracy of the network increases to 99.70% by this way. Furthermore, the proposed method has established

[3] Percent of the samples which are always within the top 2 classification scores.

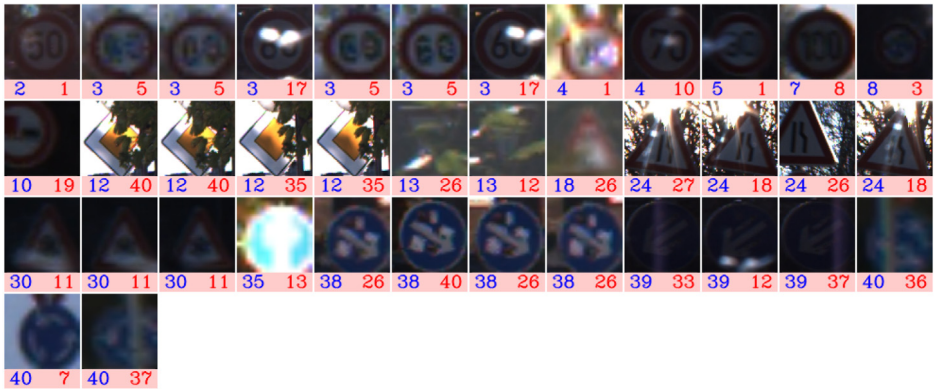


Fig. 13. Misclassified traffic sings. The blue and the red number indicate the actual and predicted class labels, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Comparing the results with ConvNets in [4,3,5].

ConvNet	Accuracy (%)
Single ConvNet (best) [4]	98.80
Single ConvNet (avg.) [4]	98.52
Multi-scale ConvNet (official) [3]	98.31
Multi-scale ConvNet (best) [5]	98.97
Proposed ConvNet (best)	99.55
Proposed ConvNet (avg.)	99.34
Committee of 25 ConvNets [4,3]	99.46
Ensemble of 3 proposed ConvNets	**99.70**

a new record compared with the winner network reported in the competition [3]. Besides, we observe that the results of the single network has outperformed the two other ConvNets. Depending on the application, one can use the single ConvNet instead of ensemble since it already outperforms state-of-art methods as well as human performance with much less time-to-completion.

Misclassified images: We computed the class-specific *precision* and *recall* (Table 6). Besides, Fig. 13 illustrates the incorrectly classified traffic signs. The number below each image shows the predicted class label. For presentation purposes, all images were scaled to a fixed size. First, we observe that there are 4 cases where the images are incorrectly classified as class 5 while the true label is 3. We note that all these cases are degraded. Moreover, class 3 is distinguishable from class 5 using the fine differences in the first digit of the sign. However, because of degradation the ConvNet is not able to recognize the first digit correctly. In addition, by inspecting the rest of the misclassified images, we realize that the wrong classification is mainly due to occlusion of the signs and blurry or degraded images. In addition, the class specific precision and recall show that the ConvNet is very accurate in classifying the traffic signs in all the classes.

4.3. Stability against noise

In real applications, it is necessary to study stability of the ConvNet against image degradations. To empirically study the stability of the classification ConvNet against Gaussian noise, the following procedure is conducted. First, we pick the test images from the original datasets. Then, 100 noisy images are generated for each $\sigma \in \{1, 2, 4, 8, 10, 15, 20, 25, 30, 40\}$. In other words, 1000 noisy images are generated for each test image in the original dataset. Next, each noisy image is entered to the ConvNet and its class label is computed. Table 7 reports the accuracy of the single ConvNet and the ensemble of three ConvNets per each value of σ. It is divided into two sections. In the first section, the accuracies are calculated on the all images. In the second section, we have

Table 6
Class specific precision and recall obtained by our network. Bottom images show corresponding class number of each traffic sign.

Class	Precision	Recall	Class	Precision	Recall	Class	Precision	Recall
0	1.00	1.00	15	1.00	1.00	30	1.00	0.98
1	1.00	1.00	16	1.00	1.00	31	1.00	1.00
2	1.00	1.00	17	0.99	1.00	32	1.00	1.00
3	1.00	0.99	18	0.99	1.00	33	1.00	1.00
4	1.00	1.00	19	0.98	1.00	34	1.00	1.00
5	0.99	1.00	20	1.00	1.00	35	0.99	1.00
6	1.00	1.00	21	1.00	1.00	36	0.99	1.00
7	1.00	1.00	22	1.00	1.00	37	0.97	1.00
8	1.00	1.00	23	1.00	1.00	38	1.00	0.99
9	1.00	1.00	24	1.00	0.96	39	1.00	0.97
10	1.00	1.00	25	1.00	1.00	40	0.97	0.97
11	0.99	1.00	26	0.97	1.00	41	1.00	1.00
12	1.00	0.99	27	0.98	1.00	42	1.00	1.00
13	1.00	1.00	28	1.00	1.00			
14	1.00	1.00	29	1.00	1.00			

Table 7
Accuracy of the ConvNets obtained by degrading the *correctly classified test images* in the original datasets using a Gaussian noise with various values of σ.

	Accuracy (%) for different values of σ									
	1	2	4	8	10	15	20	25	30	40
Single	99.4	99.4	99.3	98.9	98.3	96.3	93.2	89.7	86.0	78.7
Ensemble	99.5	99.5	99.4	99.2	98.8	97.1	94.4	91.4	88.0	81.4
Correctly classified samples										
Single	99.95	99.94	99.9	99.3	98.8	96.9	93.8	90.3	86.6	79.3
Ensemble	99.94	99.93	99.9	99.5	99.1	97.5	95.0	92.0	88.7	82.1

only considered the noisy images whose *clean* version are correctly classified by the single model and the ensemble model. Our aim in the second section is to study how noise may affect a sample which is originally classified correctly by our models.

According to this table, there are cases in which adding a Gaussian noise with $\sigma = 1$ on the images causes the models to incorrectly classify the noisy image. Note that, a Gaussian noise with $\sigma = 1$ is not easily perceivable by human eye. However, it may alter the classification result. Furthermore, there are also a few clean images that have been correctly classified by both models but they are misclassified after adding a Gaussian noise with $\sigma = 1$. Notwithstanding, we observe that both models generate admissible results when $\sigma < 10$.

This phenomena is partially studied by Szegedy et al. [36] and Aghdam et al. [35]. The above behavior is mainly due to two reasons. First, the interclass margins might be very small in some regions in the feature space where a sample may fall into another class using a slight change in the feature space. Second, ConvNets are highly non-linear functions where a small change in the input may cause a significant change in the output (feature vector) where samples may fall into a region representing to another class. To investigate non-linearity of our proposed ConvNet, we computed the Lipschitz constant of the ConvNet *locally*. Denoting the transformation from the input layer up to layer fc_2 by $\mathcal{C}_{fc_2}(x)$ where $m \in \mathbb{R}^{W \times H}$ is a gray-scale image, we compute the Lipschitz constant for every noisy image $x + \mathcal{N}(0, \sigma)$ using the following equation:

$$d(x, x + \mathcal{N}(0, \sigma)) \leq Kd(\mathcal{C}_{fc_2}(x), \mathcal{C}_{fc_2}(x + \mathcal{N}(0, \sigma))) \quad (2)$$

where K is the Lipschitz constant and $d(a, b)$ computes the Euclidean distance between a and b. For each traffic sign category in the GTSRB dataset, we pick 100 correctly classified samples and compute the Lipschitz constant between the clean images and their noisy versions. The top graph in Fig. 14 illustrates the Lipschitz constant for each sample, separately. Besides, the bottom graph shows the $d(x, x + \mathcal{N}(0, \sigma))$ and $d(\mathcal{C}_{fc_2}(x), \mathcal{C}_{fc_2}(x + \mathcal{N}(0, \sigma)))$. The black and blue lines are the linear regression and second order polynomial fitted on the point. The size of circles in the figure is associated with the value of σ in the Gaussian noise. A sample with a bigger σ appears bigger on the plot. In addition, the red circles shows the samples which are incorrectly classified after adding a noise to them.

There are some important founding in this figure. First, $\mathcal{C}_{fc_2}$ is *locally contraction* in some regions since there are instances in which their Lipschitz constant is $0 \leq K \leq 1$ regardless of the value of σ. Also, $K \in [\varepsilon, 2.5)$ which means that the ConvNet is very non-linear in some regions. Besides, we also see that there are some instances which their Lipschitz constants are small but they are incorrectly classified. This could be due to the first reason that we mentioned above. Interestingly, we also observe that there are some cases where the image is degraded using a low magnitude noise (very small dots in the plot) but its Lipschitz constant is very large meaning that in that particular region, the ConvNet is very non-linear along a specific direction. Finally, we also found out that misclassification can happen regardless of value of the Lipschitz constant.

4.4. Visualization

Understanding the behavior of ConvNets has been extensively studied through visualization techniques. Girshick et al. [31] keep the record of excitations of a particular neuron in the layer before the first fully-connected layer and show the images that highly activate this neuron. Another way to study the function of kth layer in a ConvNet is to reconstruct the input image x given the feature vector $\phi_k(x)$ as layer k. Zeiler and Fergus [37] achieve this goal using Deconvolution Networks. Mahendran and Vedaldi [38] minimize the squared Euclidean error between the actual feature vector and the feature vector of the estimated image. Similarly, Dosovitskiy and Brox [39] minimize the squared Euclidean reconstruction error of the down-sampled version of x. Both these methods show which parts/details are ignored by each layer of the network. We also visualized the sensitivity of the classification score for a particular location in the image by covering its surrounding region using a constant valued mask [6].

Another effective way to examine each layer is to non-linearly map the feature vector of a specific layer into a 2-dimensional space using the t-SNE method [40]. This visualization is important since it shows how discriminating are different layers and how a layer changes the behavior of the previous layer. Although there are other techniques such as Local Linear Embedding, Isomap and Laplacian Eigenmaps, the t-SNE method usually provides better results given high dimensional vectors. We applied this method on the fully-connected layer before the classification layer as well as the last pooling layer on the detection and the classification ConvNets individually. Figs. 15 and 16 illustrate the results for the classification and the detection ConvNets, respectively.

Comparing the results of the classification ConvNet show that although the traffic sign classes are fairly separated in the last pooling layer, the fully-connected layers increase the separability of the classes and make them linearly separable. Moreover, we observe that the two classes in the detection ConvNet are not separable in the pooling layer. However, the fully-connected layer makes these two classes to be effectively separable. These results also explain why the accuracy of the above ConvNets are high. This is due to the fact that both ConvNets are able to accurately disperse the classes using the fully-connected layers before the classification layers.

Leaking parameters: We initialize the leaking parameters of all PReLU units in the classification ConvNet to 0.01. In practice, applying PReLU activations takes slightly more time compared with LReLU activations in Caffe framework [34]. It is important to study the distribution of the leaking parameters to see if we can replace them with LReLU parameters. To this end, we computed the histogram of leaking parameters for each layer, separately. Fig. 17 shows the results.

According to the histograms, mean of leaking parameters for each layer is different except for the first and the second layers. In addition, variance of each layer is different. One can replace the PReLU activations with ReLU activations and set the leaking parameter of each layer to the mean of leaking parameter in

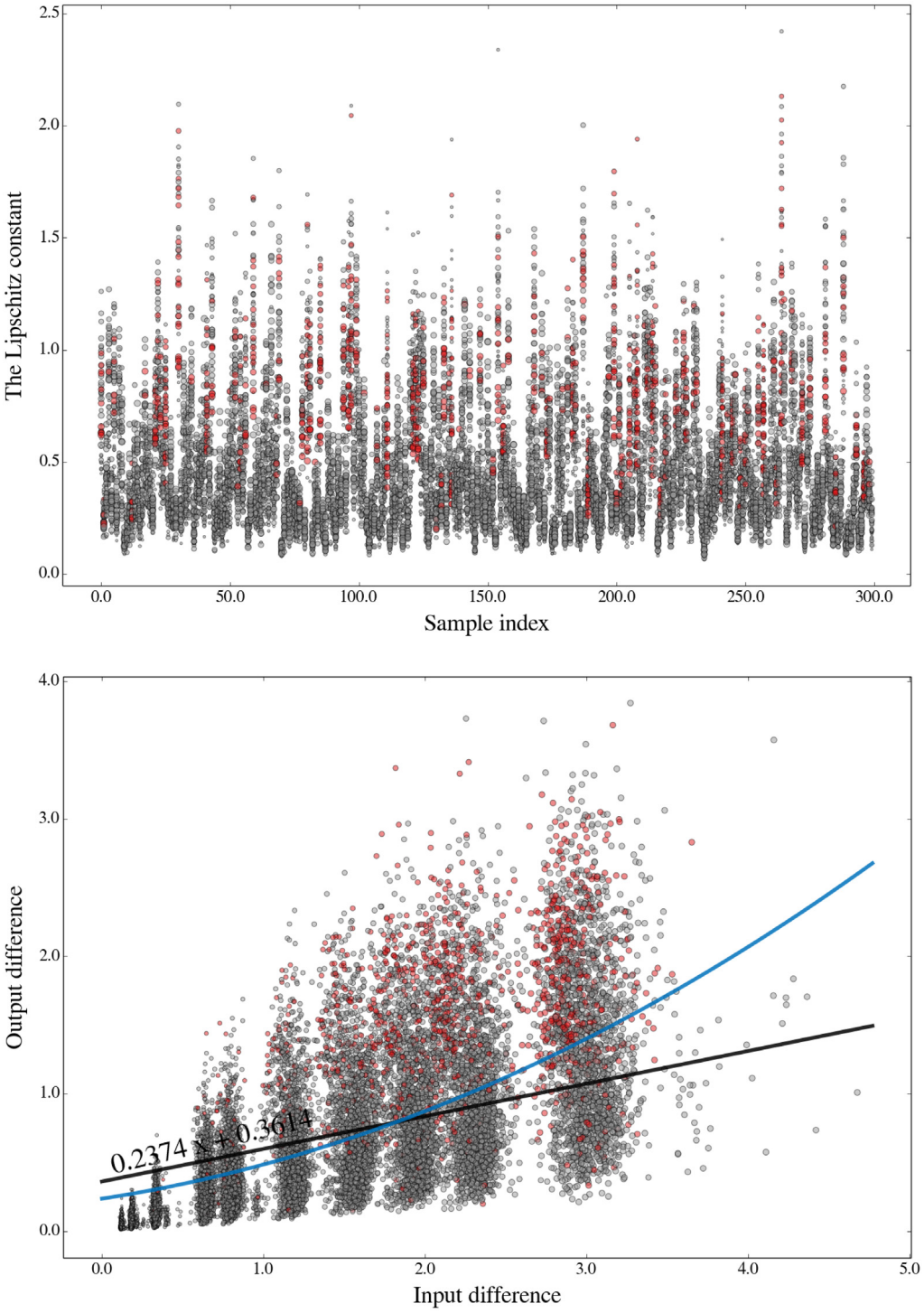


Fig. 14. Lipschitz constant (top) and the correlation between $d(x, x + \mathcal{N}(0, \sigma))$ and $d(\mathcal{C}_{fc_2}(x), \mathcal{C}_{fc_2}(x + \mathcal{N}(0, \sigma)))$ (bottom) computed on 100 samples from every category in the GTSRB dataset. The red circles are the noisy instances that are incorrectly classified. The size of each circle is associated with the values of σ in the Gaussian noise. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

this figure. By this way, time-to-completion of ConvNet will be reduced. However, it is not clear if it will have a negative impact on the accuracy. In the future work, we will investigate this setting.

5. Conclusion

In this paper, we proposed an end-to-end learning framework for detecting and classifying traffic signs. Specifically, a lightweight ConvNet is designed for detecting traffic sings on high-resolution images. This ConvNet is trained in two steps. In the first step, training is done using the negative samples that are randomly selected from the training images. Then, it is applied on the training images in order to mine hard-negative samples. These samples are used in the next step to train the ConvNet using more appropriate data. Using this technique, the average precision of the detection ConvNet increases to 99.89% on the German Traffic Sign Detection Benchmark dataset. We also explained how to implement the sliding window technique within the detection ConvNet using dilated convolutions. We further analyzed

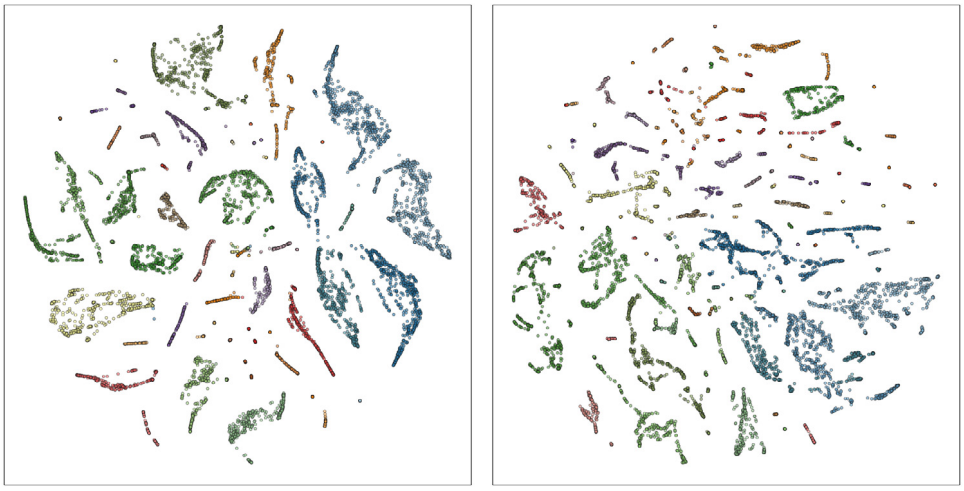


Fig. 15. Visualizing the *relu4* (left) and the *pooling3* (right) layer in the classification ConvNet using the t-SNE method. Each class is shown using a different color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

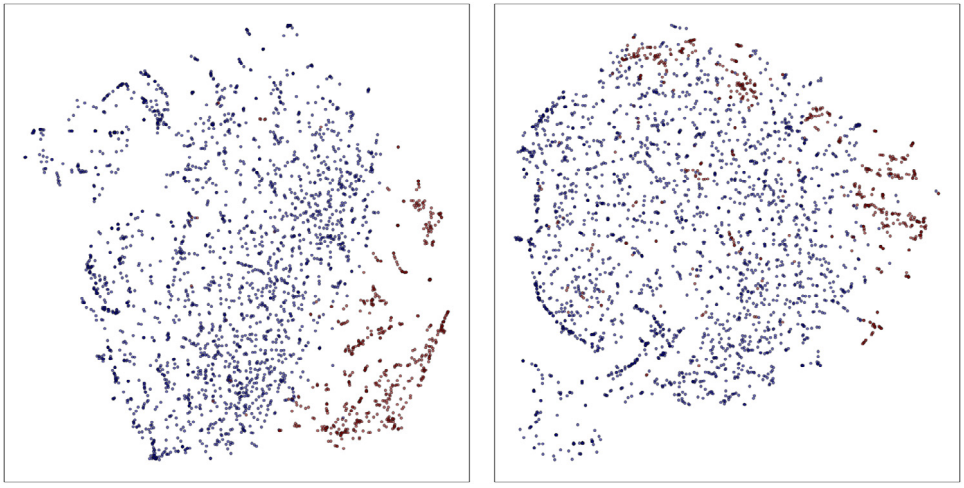


Fig. 16. Visualizing the *relu2* (left) and the *pooling1* (right) layer in the detection ConvNet using the t-SNE method. Each class is shown using a different color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

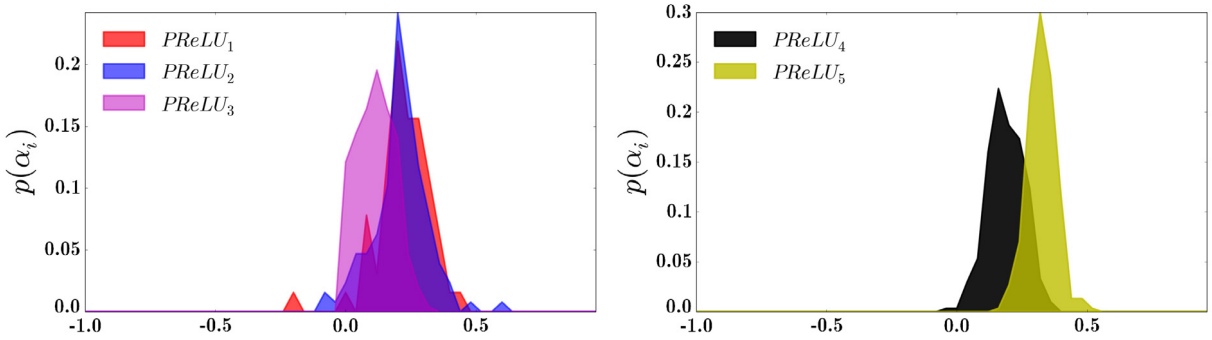


Fig. 17. Histogram of leaking parameters.

time-to-completion of the ConvNet in different settings and showed how to use statistical information from the dataset to speed up the forward pass of the ConvNet. Using this information, 37.72 high-resolution images can be processed per second with high accuracy to locate traffic signs. Next, we modified our previously proposed ConvNet for classification of traffic signs by replacing the color image with a gray-scale image, removing the linear transformation layer, reducing the kernel size in all layers, adding one more fully connected layer and replacing LReLU activation functions with PReLU activation functions. Using these modifications, the time-to-completion of the classification ConvNet was reduced compared with the previous ConvNet. Our experiments on the German Traffic Sign Recognition Benchmark dataset showed that single classification ConvNet is able to classify 99.55% of the test image, correctly. In addition, an ensemble of 3 ConvNets is able to successfully classify 99.70% of the samples. Next, we evaluated the stability of the ConvNet empirically and showed that the classification ConvNet is robust against the Gaussian noise with $\sigma < 10$.

Finally, the ConvNet was further studied using the Lipschitz constant. The results suggest that, in general, ConvNets are not stable against a strong noise since they are highly non-linear functions and the output vector may significantly change even with small perturbations in the inputs.

Acknowledgments

Hamed H. Aghdam and Elnaz J. Heravi are grateful for the supports granted by Generalitat de Catalunya's Agècia de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) through the FI-DGR 2015 fellowship and University Rovira i Virgili through the Marti Franques 2015 fellowship.

References

[1] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2012, pp. 1097–1105.
[2] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in: arXiv preprint arXiv:1502.01852, 2015.
[3] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural Netw. 32 (2012) 323–332. http://dx.doi.org/10.1016/j.neunet.2012.02.016.
[4] D. Cireşan, U. Meier, J. Masci, J. Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural Netw. 32 (2012) 333–338. http://dx.doi.org/10.1016/j.neunet.2012.02.023.
[5] P. Sermanet, Y. Lecun, Traffic sign recognition with multi-scale convolutional networks, in: Proceedings of the International Joint Conference on Neural Networks, 2011, pp. 2809–2813. http://dx.doi.org/10.1109/IJCNN.2011.6033589.
[6] H.H. Aghdam, E.J. Heravi, D. Puig, Recognizing traffic signs using a practical deep neural network, in: Second Iberian Robotics Conference, Springer, Lisbon, 2015.
[7] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: ICML Workshop on Deep Learning, Vol. 30, 2013.
[8] P. Paclík, J. Novovičová, P. Pudil, P. Somol, Road sign classification using Laplace kernel classifier, Pattern Recognit. Lett. 21 (13–14) (2000) 1165–1173. http://dx.doi.org/10.1016/S0167-8655(00)00078-7.
[9] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, F. Lopez-Ferreras, Road-sign detection and recognition based on support vector machines, IEEE Trans. Intell. Transp. Syst. 8 (2) (2007) 264–278. http://dx.doi.org/10.1109/TITS.2007.895311.
[10] S. Maldonado Bascón, J. Acevedo Rodríguez, S. Lafuente Arroyo, A. Fernndez Caballero, F. López-Ferreras, An optimization on pictogram identification for the road-sign recognition task using SVMs, Comput. Vis. Image Underst. 114 (3) (2010) 373–383. http://dx.doi.org/10.1016/j.cviu.2009.12.002.
[11] L.V.G. Radu Timofte, Sparse Representation Based Projections, in: 22nd British Machine Vision Conference, BMVA Press, 2011, pp. 61.1–61.12. http://dx.doi.org/10.5244/C.25.61.
[12] F. Larsson, M. Felsberg, Using fourier descriptors and spatial models for traffic sign recognition, in: Image Analysis, in: Lecture Notes in Computer Science, vol. 6688, Springer, 2011, pp. 238–249. http://dx.doi.org/10.1007/978-3-642-21227-7_23.
[13] F. Zaklouta, B. Stanciulescu, O. Hamdoun, Traffic sign classification using K-d trees and random forests, in: Proceedings of the International Joint Conference on Neural Networks, 2011, pp. 2151–2155. http://dx.doi.org/10.1109/IJCNN.2011.6033494.
[14] F. Zaklouta, B. Stanciulescu, Real-time traffic-sign recognition using tree classifiers, IEEE Trans. Intell. Transp. Syst. 13 (4) (2012) 1507–1514. http://dx.doi.org/10.1109/TITS.2012.2225618.
[15] F. Zaklouta, B. Stanciulescu, Real-time traffic sign recognition in three stages, Robot. Auton. Syst. 62 (1) (2014) 16–24. http://dx.doi.org/10.1016/j.robot.2012.07.019.
[16] J. Greenhalgh, M. Mirmehdi, Real-time detection and recognition of road traffic signs, IEEE Trans. Intell. Transp. Syst. 13 (4) (2012) 1498–1506. http://dx.doi.org/10.1109/tits.2012.2208909.
[17] B. Moiseev, A. Konev, A. Chigorin, A. Konushin, 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, 2013, pp. 576–583. http://dx.doi.org/10.1007/978-3-319-02895-8_52.
[18] G.-b. Huang, K.Z. Mao, C.-k. Siew, D.-s. Huang, A hierarchical method for traffic sign classification with support vector machines, in: The 2013 International Joint Conference on Neural Networks, IJCNN, IEEE, 2013, pp. 1–6. http://dx.doi.org/10.1109/IJCNN.2013.6706803.
[19] M. Mathias, R. Timofte, R. Benenson, L. Van Gool, Traffic sign recognition — how far are we from the solution?, in: Proceedings of the International Joint Conference on Neural Networks http://dx.doi.org/10.1109/IJCNN.2013.6707049.
[20] Z.-L. Sun, H. Wang, W.-S. Lau, G. Seet, D. Wang, Application of BW-ELM model on traffic sign recognition, Neurocomputing 128 (2014) 153–159. http://dx.doi.org/10.1016/j.neucom.2012.11.057.
[21] H. Fleyeh, E. Davami, Eigen-based traffic sign recognition, IET Intell. Transp. Syst. 5 (3) (2011) 190. http://dx.doi.org/10.1049/iet-its.2010.0159.
[22] X. Yuan, X. Hao, H. Chen, X. Wei, Robust traffic sign recognition based on color global and local oriented edge magnitude patterns, IEEE Trans. Intell. Transp. Syst. 15 (4) (2014) 1466–1474. http://dx.doi.org/10.1109/TITS.2014.2298912.
[23] S.-H. Hsu, C.-L. Huang, Road sign detection and recognition using matching pursuit method, Image Vis. Comput. 19 (3) (2001) 119–129. http://dx.doi.org/10.1016/S0262-8856(00)00050-0.
[24] H. Liu, Y. Liu, F. Sun, Traffic sign recognition using group sparse coding, Inform. Sci. 266 (2014) 75–89. http://dx.doi.org/10.1016/j.ins.2014.01.010.
[25] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3360–3367. http://dx.doi.org/10.1109/CVPR.2010.5540018.
[26] H.H. Aghdam, E.J. Heravi, D. Puig, 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, 2015, pp. 87–96. http://dx.doi.org/10.5220/0005303500870096.
[27] D. Cireşan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, no. February, IEEE, 2012, pp. 3642–3649. http://dx.doi.org/10.1109/CVPR.2012.6248110, arXiv:1202.2745v1.
[28] J. Jin, K. Fu, C. Zhang, Traffic sign recognition with hinge loss trained convolutional neural networks, IEEE Trans. Intell. Transp. Syst. 15 (5) (2014) 1991–2000. http://dx.doi.org/10.1109/TITS.2014.2308281.
[29] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, OverFeat: Integrated recognition, localization and detection using convolutional networks, in: arXiv preprint arXiv:1312.6229, 2013, pp. 1–15.
[30] C. Szegedy, A. Toshev, D. Erhan, Deep neural networks for object detection, in: Advances in Neural Information Processing Systems, NIPS, IEEE, 2013, pp. 2553–2561. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6909673.
[31] R. Girshick, J. Donahue, T. Darrell, U.C. Berkeley, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. arXiv:abs/1311.2524, http://dx.doi.org/10.1109/CVPR.2014.81.
[32] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, X. Tang, DeepID-Net: Deformable deep convolutional neural networks for object detection, in: Computer Vision and Pattern Recognition, 2015, arXiv:1412.5661.
[33] A. Coates, A.Y. Ng, Learning Feature Representations with K-means, in: Lecture Notes in Computer Science, 7700 LECTU, 2012, pp. 561–580. http://dx.doi.org/10.1007/978-3-642-35289-8-30, including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics.
[34] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, U.C.B. Eecs, Caffe: Convolutional architecture for fast feature embedding, in: ACM Conference on Multimedia, arXiv:1408.5093v1.
[35] H.H. Aghdam, E.J. Heravi, D. Puig, Analyzing the stability of convolutional neural networks against image degradation, in: Proceedings of the 11th International Conference on Computer Vision Theory and Applications, 2016.
[36] C. Szegedy, W. Zaremba, I. Sutskever, Intriguing properties of neural networks, 2014. arXiv:1312.6199v4.
[37] M. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, ECCV, Vol. 8689, 2014, pp. 818–833. http://dx.doi.org/10.1007/978-3-319-10590-1_53, arXiv:1311.2901.
[38] A. Mahendran, A. Vedaldi, Understanding deep image representations by inverting them, in: Comput. Vis. Pattern Recognit., IEEE, Boston, 2015, pp. 5188–5196. http://dx.doi.org/10.1109/CVPR.2015.7299155, arXiv:abs/1412.0035.
[39] A. Dosovitskiy, T. Brox, Inverting convolutional networks with convolutional networks, in: arXiv preprint arXiv:1506.02753, 2015, pp. 1–15.
[40] L.V.D. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605. http://dx.doi.org/10.1007/s10479-011-0841-3.

Mr. Hamed Habibi Aghdam received his M.Sc. degree in artificial intelligence and robotic in 2012 from Amirkabir University of Technology (Tehran Polytechnic), Iran. He has worked as a software developer and conference organizer in several companies and institutions in the past. He has been also awarded in some robotic competitions and conferences. He is now a Ph.D. candidate in the Intelligent Robotics and Computer Vision Group at the Department of Computer Engineering and Mathematics, University Rovira i Virgili, Tarragona, Spain. His research interests include deep learning, motion understanding, egocentric video analysis and human activity understanding. He has actively

contributed to the IA-BioBreast project TIN2012-37171-C02-02 funded by the Spanish Government. He was also the machine vision researcher and developer for the GAME-ABLING project "EU-FP7-SME-2012-1-315032" funded by the European Union.

Ms. Elnaz Jahani Heravi received her M.Sc. degree in Mechatronic engineering in 2011 from Qazvin Azad University, Iran. She is now a Ph.D. candidate in the Intelligent Robotics and Computer Vision Group at the Department of Computer Engineering and Mathematics, University Rovira i Virgili, Tarragona, Spain. She is mainly interested in computer vision and machine leaning. Her research interests include object detection and classification, fine-grained object recognition, deep learning and attribute based object representation. She has been recently awarded the best paper awards in the acclaimed VISAPP 2015 and ICMV 2015 conferences. She has also actively contributed to the IA-BioBreast project TIN2012-37171-C02-02 funded by the Spanish Government.

Dr. Domenec Puig received the M.S. and Ph.D. degrees in computer science from Polytechnic University of Catalonia, Barcelona, Spain, in 1992 and 2004, respectively. In 1992, he joined the Department of Computer Science and Mathematics, University Rovira i Virgili (URV), Tarragona, Spain, where he is currently Associate Professor and Dean of the School of Engineering. Since July 2006, he is the Head of the Intelligent Robotics and Computer Vision Group at the same university. His research interests include image processing, texture analysis, perceptual models, scene analysis, and mobile robotics. He has participated or managed over 30 funded research projects related to his topics of interest. He has published numerous ISI-JCR journal papers and contributions in international conferences. He has recently supervised 4 Ph.D. theses related to Computer Vision, Image Processing and Pattern Recognition disciplines. He is currently the principal investigator in the URV of the IA-BioBreast project TIN2012-37171-C02-02 funded by the Spanish Government, and the GAME-ABLING project "EU-FP7-SME-2012-1-315032" funded by the EU under the FP7.