

# A Practical and Highly Optimized Convolutional Neural Network for Classifying Traffic Signs in Real-Time

Hamed Habibi Aghdam<sup>1</sup>  · Elnaz Jahani Heravi<sup>1</sup> · Domenec Puig<sup>1</sup>

Received: 14 August 2015 / Accepted: 8 September 2016 / Published online: 21 September 2016  
© Springer Science+Business Media New York 2016

**Abstract** Classifying traffic signs is an indispensable part of Advanced Driver Assistant Systems. This strictly requires that the traffic sign classification model accurately classifies the images and consumes as few CPU cycles as possible to immediately release the CPU for other tasks. In this paper, we first propose a new ConvNet architecture. Then, we propose a new method for creating an optimal ensemble of ConvNets with highest possible accuracy and lowest number of ConvNets. Our experiments show that the ensemble of our proposed ConvNets (the ensemble is also constructed using our method) reduces the number of arithmetic operations 88 and 73 % compared with two state-of-art ensemble of ConvNets. In addition, our ensemble is 0.1 % more accurate than one of the state-of-art ensembles and it is only 0.04 % less accurate than the other state-of-art ensemble when tested on the same dataset. Moreover, ensemble of our compact ConvNets reduces the number of the multiplications 95 and 88 %, yet, the classification accuracy drops only 0.2 and 0.4 % compared with these two ensembles. Besides, we also evaluate the cross-dataset performance of our ConvNet and analyze its transferability power in different layers. We show that our network is easily scalable to new datasets with much more number of traffic sign classes and it only needs to fine-

tune the weights starting from the last convolution layer. We also assess our ConvNet through different visualization techniques. Besides, we propose a new method for finding the minimum additive noise which causes the network to incorrectly classify the image by minimum difference compared with the highest score in the loss vector.

**Keywords** Convolutional neural network · Traffic sign classification · Ensemble construction · Visualizing convolutional neural networks

## 1 Introduction

Advanced Driver Assistant System (ADAS) is a crucial component in intelligent transportation systems and it is an indispensable part of today's smart cars. An ADAS performs different functions. Among them is recognizing traffic signs which helps the driver (human or autonomous) to conform with the road rules and drive the car, safely.

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

Even though classification of traffic signs is an easy task for a human, there are some challenges in developing an

Communicated by Hiroshi Ishikawa, Takeshi Masuda, Yasuyo Kita and Katsushi Ikeuchi.

✉ Hamed Habibi Aghdam  
hamed.habibi@urv.cat

Elnaz Jahani Heravi  
elnaz.jahani@urv.cat

Domenec Puig  
domenec.puig@urv.cat

<sup>1</sup> Intelligent Robotic and Computer Vision Group, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Spain



**Fig. 1** Some of the challenges in classification of traffic signs. The signs have been collected in Germany and Belgium

algorithm for this purpose (Fig. 1). First, the image of traffic signs might be captured from different perspectives. Second, weather condition can dramatically affect the appearance of traffic signs. An example is illustrated in Fig. 1 where the “no stopping” sign is covered by snow. Third, traffic signs are being impaired during time and some artifacts may appear on the sign which might have a negative impact on the classification score. Fourth, traffic signs might be partially occluded by another signs or objects. Fifth, the pictograph area might be manipulated by human which in some cases might change the shape of the pictograph. The last and more important issue shown in this figure is the pictograph differences of the same traffic sign from one country to another. More specifically, we observe that the “danger: bicycle crossing” sign poses a few important differences between two countries.

As we mentioned earlier, traffic sign classification is one of the tasks of an ADAS. Consequently, their classification must be done in real-time and it must consume as few CPU cycles as possible in order to release the CPU immediately. Last but not the least, the classification model must be easily scalable so the model can be adjusted to new classes in the future with a few efforts. In sum, any model for classifying traffic signs must be *accurate, fast, scalable* and *fault-tolerant*.

Traffic sign classification is a specific case of object classification where the objects are more rigid and two dimensional. Also, their discriminating part are well-defined. Recently, Convolutional Neural Networks (ConvNets) surpassed human performance on classification of 1000 natural objects (He et al. 2015). Moreover, there are other ConvNets such as Simonyan and Zisserman (2015) and Szegedy et al. (2014a) with close performances compared to He et al. (2015). However, the architecture of the ConvNets are significantly different from each other. This suggests that the

same problem can be solved using ConvNets with different architectures and various complexities. Part of the complexity of ConvNets is determined using activation functions. They play an important role in neural networks since they apply non-linearities on the output of the neurons which enable ConvNets to apply series of non-linear functions on the input and transform the input into a space where classes are linearly separable. As we discuss in Section 3, selecting a highly computational activation function can dramatically increase the number of required arithmetic operations of the network which in sequel increases the response-time of a ConvNet.

From one perspective, a ConvNet can be thought as a black-box to extract a feature vector for a given input. Hence, it is possible to use an already trained ConvNet for extracting feature vectors of traffic signs and train a model for classifying them. This procedure has been previously proposed for generic object recognition problem (Sermanet et al. 2013; Donahue et al. 2014). However, there are two restrictive issues with this approach for classifying traffic signs. First, the networks utilized in Sermanet et al. (2013) and Donahue et al. (2014) are trained on 3D objects which are mostly natural and deformable. For this reason, they need a more complex configuration of layers to effectively model these variations. Consequently, they are not computationally efficient for requirements of an ADAS. In addition, it is not guaranteed that the trained networks generalize well on the traffic sign classification problem.

One of issues with ConvNets is that their behaviour is not well understood. Specifically, it is fairly trivial to answer the question “How might a small variation on an image affect the output of a HOG/LBP/Gabor descriptor?”. Nevertheless, answering this question is not trivial when we use ConvNets to extract features. This is due to the fact ConvNets extract features by applying series of non-linear operations on the input. As we discuss in Sect. 2, there are different studies to analyze behaviour of a ConvNet through visualization but they are not enough to develop an applicable traffic sign classification model.

**Contribution** To our knowledge, this is the first reported work which thoroughly studies different aspects of ConvNets on classification of traffic signs. In particular, our contribution has two folds. First, we investigate the previously proposed ConvNets for classifying traffic signs and analyze their computational complexity in Sect. 3. Then, we propose a new ConvNet which reduces the number of the parameters 27, 22 and 3 % compared with the networks proposed in Ciresan et al. (2012), Sermanet and Lecun (2011) and Jin et al. (2014), respectively. Next, we show how to analyze the proposed architecture in practice and reduce the number of parameters 52 % compared with our original ConvNet. This means our compact ConvNet needs 65, 63 and 54 % fewer parameters compared with Ciresan et al. (2012), Sermanet

and Lecun (2011) and Jin et al. (2014), respectively. It is previously shown (Ciresan et al. 2012; Sermanet et al. 2013; Jin et al. 2014) that an ensemble of ConvNets improves the classification accuracy. However, creating an ensemble is usually done by computing the average score of many ConvNets that is not computationally efficient because of redundancy of ConvNets. We propose a new method for selecting an optimal number of ConvNets with highest classification accuracy in Sect. 3.1. Next, we accurately analyze the number of the arithmetic operations required by previously proposed networks and illustrate that our network requires 88 and 73 % fewer multiplications compared with Ciresan et al. (2012) and Jin et al. (2014). Notwithstanding, it improves the classification accuracy 0.1 % compared with Ciresan et al. (2012) and its accuracy is only 0.04 % less than Jin et al. (2014).

Second, we extensively analyze behaviour of the proposed ConvNet in Sect. 4 by proposing a new method for calculating the *degree of being fault-tolerant* of the network. Our proposed method answers the question “what is the minimum degradation which causes the image to be misclassified with the minimum difference between the winner neuron and actual neuron?”. Then, we investigate “how generic is our network?” by classifying traffic signs of another country which posses appearance differences compared to the traffic signs we have used for training our ConvNet. Next, we answer the question “How scalable is our network?” by transferring the ConvNet to a new dataset with 40 % more traffic sign classes. To be more specific, we investigate the transferability of the ConvNet by decreasingly freezing the layers and fine-tuning the network on the new dataset. Finally, we analyze behaviour of the network by answering the questions “How does the network react when it only perceives a small part of the image?” and “In which locations in the image is the network more sensitive to noise?”.

## 2 Related Work

In general, efforts for classifying traffic signs can be divided into *traditional classification* and *end-to-end learning* methods. In the former approach, researchers have tried to design hand-crafted features and train a classifier on top of these features. In contrast, end-to-end learning approaches learn the representation and classification automatically from data. In this section, we first review the traditional classification approaches and then we explain the previously proposed end-to-end learning methods for classifying traffic signs.

**Template Matching** Early works considered a traffic sign as a rigid object and classified the query image by comparing it with all templates stored in the database (Piccioli et al. 1996). Later, Gao et al. (2006) matched shape features instead of pixel intensity values. In this work, matching features is done using Euclidean distance function. The problem with this

matching function is that they consider every pixel/feature equally important. To cope with this problem, Ruta et al. (2010) learned a similarity measure for matching the query sign with templates.

**Hand-Crafted Features** More accurate and robust results were obtained by learning a classification model over a feature vector. Paclík et al. (2000) produce a binary image depending on the color of the traffic sign. Then, 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-Bascon et al. (2007) addressed this problem by transforming the image into the HSI color space and calculating histogram of Hue and Saturation components. Then, the histogram is classified using a multi-class SVM. In another method, Maldonado Bascón et al. (2010) classified 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. (2010) trained SVM where the input is a  $31 \times 31$  block of pixels in a gray-scale version of pictograph. In a more complicated approach, Baró et al. (2009) proposed an Error Correcting Output Code framework for classification of 31 traffic signs and compared their method with various approaches.

Before 2011, there was not a public and challenging dataset of traffic signs. Timofte and Van Gool (2011), Larsson and Felsberg (2011) and Stallkamp et al. (2012) introduced three challenging datasets including annotations. These databases are called Belgium Traffic Sign Classification (BTSC), Swedish Traffic Sign and German Traffic Sign Recognition Benchmark (GTSRB), respectively. In particular, the GTSRB was used in a competition and, as we will discuss shortly, the winner method classified 99.46 % of test images correctly (Stallkamp et al. 2012). Zaklouta et al. (2011) and Zaklouta and Stanculescu (2012, 2014) extracted Histogram of Oriented Gradient (HOG) descriptors with three different configurations for representing the image and trained a Random Forest and a SVM for classifying traffic signs in the GTSRB dataset. Similarly, Greenhalgh and Mirmehdi (2012), Moiseev et al. (2013), Huang et al. (2013), Mathias et al. (2013) and Sun et al. (2014) utilized the 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 (Huang et al. 2013) use the traditional classification approach. In contrast, Huang et al. (2013) utilize a two level classification. In the first level, the image is classified into one of super-classes. Each super-class contains several traffic signs with similar shape/color. Then, the per-

spective 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. [Timofte et al. \(2011\)](#) proposed a framework for recognition and the traffic signs in the BTSC dataset and achieved 97.04 % accuracy on this dataset.

**Sparse Coding** [Hsu and Huang \(2001\)](#) coded each traffic sign using the Matching Pursuit algorithm. During testing, the input image is projected to different set of filter bases to find the best match. [Lu et al. \(2012\)](#) proposed a graph embedding approach for classifying traffic signs. They preserved the sparse representation in the original space by using  $L_{1,2}$  norm. [Liu et al. \(2014\)](#) constructed the dictionary by applying k-means clustering on the training data. Then, each data is coded using a novel coding input similar to Local Linear Coding approach ([Wang et al. 2010](#)). Recently, we proposed a method based on visual attributes and Bayesian network ([Aghdam et al. 2015](#)). In this method, we describe 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.

**Other** [Fleyeh and Davami \(2011\)](#) projected 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. \(2014\)](#) proposed a novel feature extraction method to effectively combine color, global spatial structure, global direction structure and local shape information. Readers can refer to [Møgelmoose et al. \(2012\)](#) to study traditional approaches of traffic sign classification.

**Discussion** Template matching approaches are not robust against perspective variations, ageing, noise and occlusion. Hand-crafted features has a limited representation power and they might not scale well if the number of classes increases. In addition, they are not robust against irregular artifacts caused by motion blurring and weather condition. This can be observed by the results reported in the GTSRB competition ([Stallkamp et al. 2012](#)) where the best performed solution based on hand-crafted feature was only able to correctly classify 97.88 % of test cases<sup>1</sup>. Later, [Mathias et al. \(2013\)](#) improved the accuracy based on hand-crafted features up to 98.53 % on the GTSRB dataset. Notwithstanding, there are a few problems with this method. Their raw feature vector is a 9000 dimensional vector constructed by applying five different methods. This high dimensional vector is later projected to a lower dimensional space. For this reason, their method is time consuming when they are executed on a multi-core

CPU compared with our single ConvNet in Sect. 4. Note that Table V in [Mathias et al. \(2013\)](#) have only reported the time on classifiers and it has disregarded the time required for computing feature vectors and projecting them into a lower dimension space. Considering that the results in Table V have been computed on the test set of the GTSRB dataset (12630 samples), only classification of a feature vector takes 48 ms. On the other hand, it is not trivial to implement their method on a GPU.

**ConvNets** ConvNets were first utilized by [Sermanet and Lecun \(2011\)](#) and [Ciresan et al. \(2012\)](#) in the field of traffic sign classification during the GTSRB competition where the ConvNet of ([Ciresan et al. 2012](#)) surpassed human performance and won the competition by correctly classifying 99.46 % of test images. Moreover, the ConvNet of ([Sermanet and Lecun 2011](#)) ended up in the second place with a considerable difference compared with the 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 97.88 %, respectively.

Both [Sermanet and Lecun \(2011\)](#) and [Ciresan et al. \(2012\)](#) 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 different. [Ciresan et al. \(2012\)](#) constructs an ensemble of 25 ConvNets each consists of 1, 543, 443 parameters. [Sermanet and Lecun \(2011\)](#) creates a single network defined by 1, 437, 791 parameters. Furthermore, while the winner ConvNet uses the *hyperbolic* activation function, the runner-up ConvNet utilizes the *rectified sigmoid* as the activation function. It is a common practice in ConvNets to make a prediction by calculating the average score of slightly transformed versions of the query image. However, it is not clearly mentioned in [Sermanet and Lecun \(2011\)](#) that how do they make a prediction. In particular, it is not clear that the runner-up ConvNet classifies solely the input image or it classifies different versions of the input and fuses the scores to obtain the final result.

Regardless, both methods suffer from the high number of arithmetic operations. To be more specific, they use highly computational activation functions (we will discuss about this in Sect. 3). To alleviate these problems, [Jin et al. \(2014\)](#) proposed a new architecture including 1, 162, 284 parameters and utilizing the *rectified linear unit* (ReLU) activations ([Krizhevsky et al. 2012](#)). In addition, there is a Local Response Normalization layer after each activation layer. They built an ensemble of 20 ConvNets and classified 99.65 % of test images correctly. Although the number of parameters is reduced using this architecture compared with the two networks, the ensemble is constructed using 20 ConvNets which is not still computationally efficient in real-world applications. It is worth mentioning that a ReLU layer and a Local Response Normalization layer together needs

<sup>1</sup> <http://benchmark.ini.rub.de/>.

approximately the same number of arithmetic operations as a single hyperbolic layer. As the result, the run-time efficiency of the network proposed in Jin et al. (2014) might be close to Ciresan et al. (2012).

Recently, Zeng et al. (2015) trained a ConvNet to extract features of the image and replaced the classification layer of their ConvNet with an Extreme Learning Machine (ELM) and achieved 99.40 % accuracy on the GTSRB dataset. There are two issues with their approach. First, the output of last convolution layer is a 200 dimensional vector which is connected to 12,000 neurons in the ELM layer. This layer is solely defined by  $200 \times 12,000 + 12,000 \times 43 = 2,916,000$  parameters which makes it impractical. Besides, it is not clear why their ConvNet reduces the dimension of the feature vector from  $250 \times 16 = 4000$  in Layer 7 to 200 in Layer 8 and then map their lower dimensional vector to 12000 dimensions in the ELM layer (Zeng et al. 2015, Table 1). One reason might be to cope with calculation of the matrix inverse during training of the ELM layer. Finally, since the input connections of the ELM layer is determined randomly, it is probable that their ConvNet does not generalize well on other datasets.

Despite the impressive results obtained by the four 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. Understanding behaviour of ConvNets has been extensively studied through visualization techniques. Girshick et al. (2014) 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. The mentioned procedure can be formulated by maximizing the inner product of the activation values of a layer and natural basis. However, Szegedy et al. (2014b) argue that the natural basis is not better than a random basis for inspecting the properties of activations.

We can consider  $k$ th layer of a ConvNet as a vector function  $\phi_k(x)$  where  $x$  is the input image and it returns a feature vector calculated at layer  $k$ . Another way to inspect behaviour of a layer is to reconstruct the image  $x$  given the feature vector  $\phi_k(x)$ . Zeiler and Fergus (2014) achieve this goal using Deconvolution Networks. This methods shows which object-parts excite a particular neuron. Mahendran and Vedaldi (2015) minimize the squared Euclidean error between the actual feature vector and the feature vector of the estimated image. Similarly, Dosovitskiy and Brox (2015) minimize the squared Euclidean reconstruction error of the down-sampled version of  $x$ . Both these methods show which parts/details are ignored by different layers.

Contrary to the previous works, Simonyan et al. (2013) maximize the  $L_2$  regularized classification score of the reconstructed image to visualize the class specific model. This helps to find how a ConvNet see the world from the viewpoint of different classes. Another effective way to assess behav-

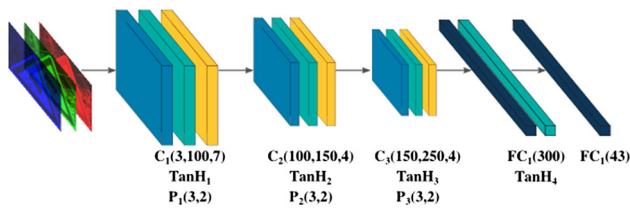
our of each layer is to visualize the generated features of the layers using t-SNE method (Van Der Maaten and Hinton 2008). This is important since it shows how discriminating are different layers and how a layer transforms the output of the previous layer.

**Discussion** The ConvNets for traffic sign classification can be explained from three perspectives including *scalability*, *stability* and *run-time*. From generalization point of view, none of the four ConvNets have assessed the performance on other datasets. It is crucial to study how the networks perform when the signs slightly change from one country to another. More importantly, the transferring power of the network must be estimated by fine-tuning the same architecture on a new dataset with various number of classes. By this way, we are able to estimate the *scalability* of the networks. From *stability* perspective, it is crucial to find out how tolerant is the network against noise and occlusion. This might be done through generating a few noisy images and fetch them to the network. However, this approach does not find the minimum noisy image which is misclassified by the network. Finally, the *run-time efficiency* of the ConvNet must be examined. This is due to the fact that the ConvNet has to consume as few CPU cycles as possible to let other functions of ADAS perform in real-time. In the next section, we will build a ConvNet which possess all these factors and outperforms the state-of-art ConvNets.

### 3 Proposed Network

The basic architecture of our network is inspired by Ciresan et al. (2012) which is shown in Fig. 2. First, the size of the receptive field in the first layer and the subsequent layers are chosen properly. Also, we believe the complexity of the network is enough for modelling a wide range of traffic signs. We aim to reduce the number of the parameters and the number of the arithmetic operations and increase the classification accuracy. To this end, we replace the hyperbolic non-linearities with the Leaky ReLU activation functions (Maas et al. 2013). Beside the favourable properties of ReLU activations, they are also computationally very efficient. The hyperbolic activation function is defined as  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$ . Even with an efficient implementation of exponentiation  $e^x$ , it still requires many multiplications. Note that  $x$  is a floating point number since it is the weighted sum of the input to the neuron.

An efficient way to calculate  $e^x$  is as follows: First, write  $x = x_{int} + r$ , where  $x_{int}$  is the nearest integer to  $x$  and  $r \in [-0.5 \dots 0.5]$  which gives  $e^x = e^{x_{int}} \times e^r$ . Second, multiply  $e$  by itself  $x_{int}$  times. The multiplication can be done quite efficiently. To further increase efficiency, various integer powers of  $e$  can be calculated and stored in a lookup



**Fig. 2** The ConvNet proposed in Ciresan et al. (2012). Light blue, green, yellow and dark blue shapes indicate convolution, activation, pooling and fully-connected layers, respectively.  $C_i(c, k, w)$  shows a convolution layer with  $k$  filters of size  $w \times w$  applied on the input with  $c$  channels.  $P(m, n)$  indicates a MAX pooling layer with kernel size  $m \times m$  and stride  $n$ . Finally,  $FC(x)$  depicts a fully connected layer with  $x$  neurons (Color figure online)

table. Finally,  $e^r$  can be estimated using the polynomial  $e^r = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$  with estimation error  $+3e^{-5}$ . Consequently, calculating  $\tanh(x)$  needs  $[x] + 5$  multiplications and 5 divisions. We assuming that division and multiplication need the same amount of CPU cycles. Therefore,  $\tanh(x)$  can be computed using  $[x] + 10$  multiplications. The simplest scenario is when  $x \in [-0.5 \dots 0.5]$ . Then,  $\tanh(x)$  can be calculated using 10 multiplications. Based on this, the total number of multiplications of the network proposed in Ciresan et al. (2012) is equal to 128, 321, 700. Since they build an ensemble of 25 networks, thus, the total number of the multiplications must be multiplied by 25 which is equal to 3, 208, 042, 500 multiplications for making a prediction using an ensemble of 25 networks shown in Fig. 2. In contrary, a Leaky ReLU function needs only one multiplication in the worst case and if the input of the activation function is positive, it does not need any multiplication.

Furthermore, we also divide the two middle convolution-pooling layers into two separate layers and add another layer after the input layer. Finally, we end up with the architecture shown in Fig. 3. Our network consists of a transformation layer, three convolution-pooling layers and two fully connected layers with a dropout layer (Hinton 2014) in between. Finally, there is a Leaky ReLU layer after each convolution layer and after the first fully-connected layer. The network accepts a  $48 \times 48$  RGB image and classify it into one of the 43 traffic sign classes in the GTSRB dataset. Note that we have reduced the number of the parameters by dividing the two middle convolution-pooling layers into two groups.

More specifically, the transformation layer applies an element-wise linear transformation  $f_c(x) = a_c x + b_c$  on  $c$ th channel of the input image where  $a_c$  and  $b_c$  are trainable parameters and  $x$  is the input value. Note that each channel has a unique transformation function. As we show in Sect. 4.5.2 these functions enhance the illumination of each channel separately. Next, the image is processed using 100 filters of size  $7 \times 7$ . The notation  $C(c, k, w)$  indicates a convolution layer with  $k$  filters of size  $w \times w$  applied on the input with  $c$  channels. Then, the output of the convolution is passed

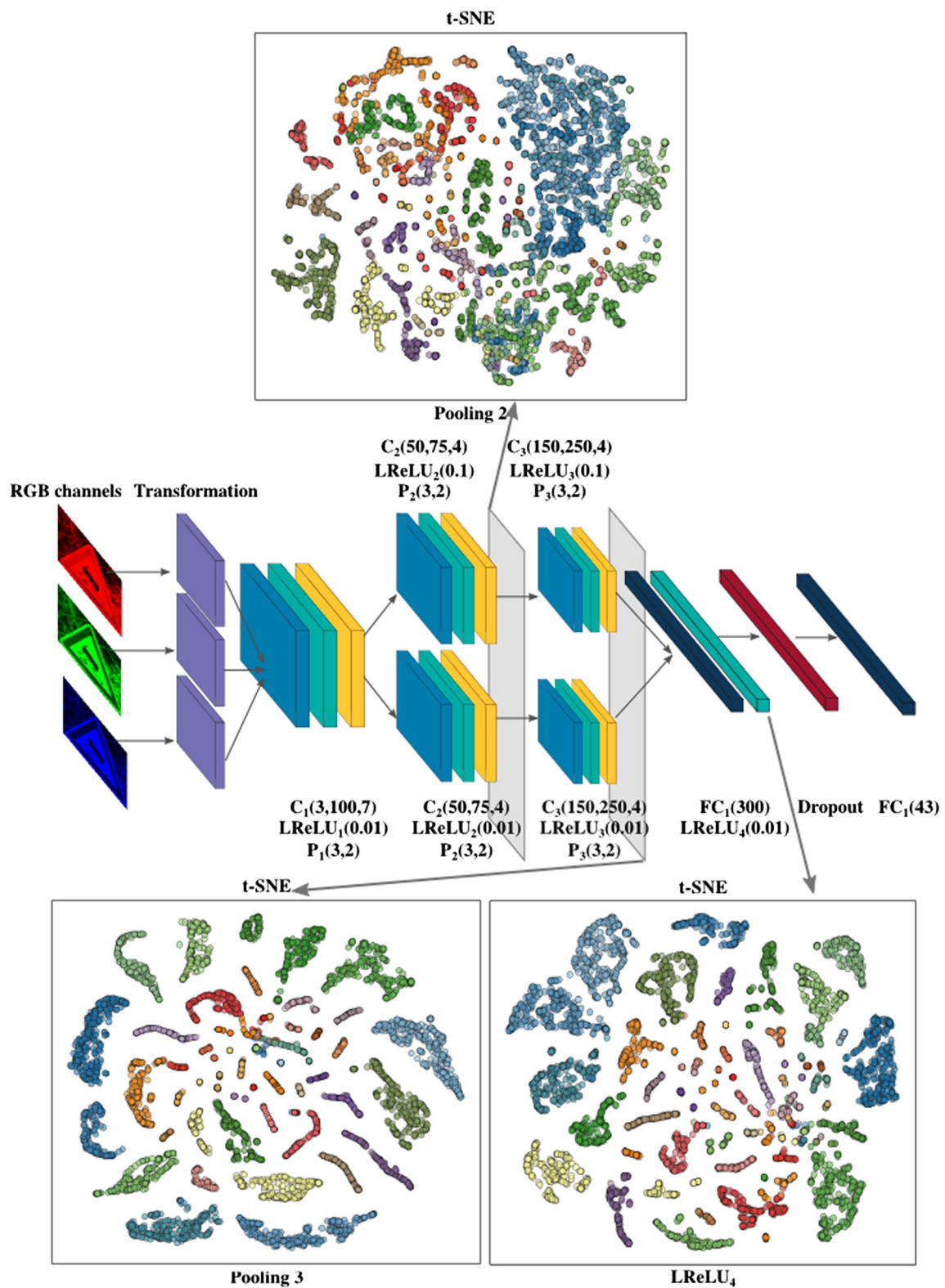
through a Leaky ReLU (Maas et al. 2013) layer and fetched into the pooling layer where a MAX-pooling is applied on the  $3 \times 3$  window with stride 2.

In general, a  $C(c, k, w)$  layer contains  $c \times k \times w \times w$  parameters. In fact, the second convolution layer accepts a 100-channel input and applies 150 filters of size  $4 \times 4$ . Using this configuration, the number of the parameters in the second convolution layer would be 240, 000. The number of the parameters in the second layer is halved by dividing the input channels into two equal parts and fetch each part into a layer including two separate  $C(50, 75, 4)$  convolution-pooling units. Similarly, the third convolution-pooling layer halves the number of the parameters using two  $C(75, 125, 4)$  units instead of one  $C(150, 250, 4)$  unit. Our proposed architecture is collectively parametrized by 1, 123, 449 weights and biases which is 27, 22 and 3 % reduction in the number of the parameters compared with the networks proposed in Ciresan et al. (2012), Sermanet and Lecun (2011) and Jin et al. (2014), respectively. Note that compared with Jin et al. (2014) our proposed network needs much less arithmetic operations since Jin et al. (2014) uses a Local Response Normalization layer after each activation layer which needs a few multiplications per element in the resulting feature map from previous layer. This issue is deeply analyzed in Sect. 4.2.

We trained the proposed network on the GTSRB dataset and applied some preliminary analysis. It is a common practice to inspect extracted features of a ConvNet using a visualization technique. As it is shown in Fig. 3, we visualized the feature maps after the second and third pooling layers as well as the activation layer before the last fully-connected layer using the t-SNE method (Van Der Maaten and Hinton 2008). We observe that the classes are separated properly after the third pooling layer. This implies that the classification layer might be able to accurately discriminate the classes if we omit the first fully-connected layer. According to the t-SNE visualization, the first fully-connected layer does not increase the discrimination of the classes, considerably. Instead, it rearranges the classes in a lower-dimensional space and it might mainly affect the interclass distribution of the samples.

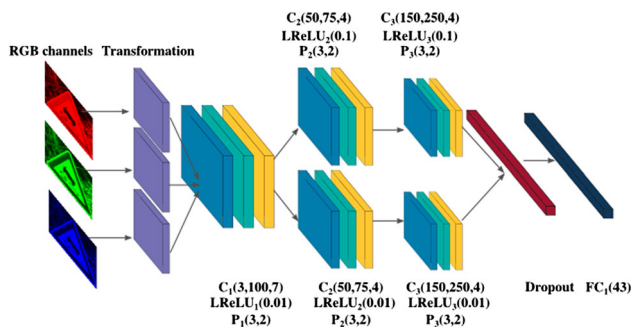
Consequently, it is possible to discard the first fully-connected layer and the subsequent Leaky ReLU layer from the network and connect the third pooling layer directly to the dropout layer. The more compact network is shown in Fig. 4. From optimization perspective, this decreases the number of the parameters from 1, 123, 449 to 531, 999 which is 65, 63 and 54 % reduction compared with Ciresan et al. (2012), Sermanet and Lecun (2011) and Jin et al. (2014), respectively.

As we discuss in Sect. 4, a single ConvNet shown in Fig. 4 is more accurate than a single network in Fig. 3. However, this does not hold when we create an ensemble of ConvNets. In fact, the fully-connected layer shown in Fig.



**Fig. 3** Our proposed network architecture along with visualization of the first fully-connected layer as well as the last two pooling layers using the t-SNE method. The *color codes* are similar to Fig. 2. In addition,

each purple shape shows a linear transformation function. Each class is shown with a *unique color* in the scatter plots (best viewed in *color*) (Color figure online)



**Fig. 4** Compact version of the network illustrated in Fig. 3 after dropping the first fully-connected layer and the subsequent Leaky ReLU layer

3 adds more complexity to our model. Therefore, the diversity of the ensemble increases which in turn improves the classification performance.

### 3.1 Creating Ensemble

Previous studies (Ciresan et al. 2012; Jin et al. 2014; Sermanet et al. 2013) show that creating an ensemble of ConvNets might increase the classification accuracy. These works utilize the model averaging technique in which the average loss of several ConvNets is computed. However, there are two limitations with this approach. First, our experiments revealed that, sometimes, the classification accuracy of an ensemble is not improved substantially compared with a single ConvNet in the ensemble. This is due to the fact that these ConvNets might have ended up in the same local minimum during the training process. As the result, their losses are very similar and their combination does not change the posterior of the classes. The second problem is that there are some cases where adding a new ConvNet to the ensemble reduces the classification performance. One possibility is that the belief about a class posteriors of the new ConvNet is greatly different from the belief of the ensemble. Consequently, the new ConvNet changes the posterior of the ensemble dramatically which in turn reduces the classification performance.

To alleviate this problem, Ciresan et al. (2012) and Jin et al. (2014) create ensembles consisting of many ConvNets. The idea is that the number of the ConvNets which contradicts the belief of the ensemble is less than the number of the ConvNets which increase the classification performance. Therefore, the overall performance of the ensemble increases as we add more ConvNets.

While the idea is generally correct but it poses a serious problem in practical applications. Concretely, an ensemble with many ConvNets needs more time to classify the input image. One solution to this problem is to formulate the ensemble construction as a LASSO regression (Tibshirani 1994) problem. Formally, given the classification score vec-

tor  $\mathcal{L}_i^j$  of  $i$ th ConvNet computed on  $j$ th image, our goal is to find coefficients  $a_i$  by minimizing the following error function:

$$E = \sum_{j=1}^M \left\| y_j - \sum_{i=1}^N a_i \mathcal{L}_i^j \right\| - \lambda \sum_{i=1}^N |a_i| \tag{1}$$

where  $M$  is the total number of the test images,  $N$  is the number of the ConvNets in the ensemble and  $\lambda$  is a user-defined value to determine the amount of sparseness. It is well-studied that  $L_1$  norm regularization produces a sparse vector in which most of the coefficients  $a_i$  are zero. Thus, the ConvNets corresponding to these coefficients can be omitted from the ensemble. The remaining ConvNets are linearly combined according to their corresponding  $a_i$  value. Determining the correct value for  $\lambda$  is an empirical task and it might need many trials. More specifically, small values for  $\lambda$  retains most of the ConvNets in the ensemble. Conversely, increasing the value of  $\lambda$  drops more ConvNets from the ensemble.

In this paper, we formulate the ensemble construction as the *optimal subset selection problem* by solving the following optimization problem:

$$\arg \min_{I \subset \{1, \dots, N\}} \left[ \frac{1}{M} \sum_{j=1}^M \delta \left( y_j - \arg \max_{i \in I} \mathcal{L}_i^j \right) \right] - \lambda |I| \tag{2}$$

where the  $\arg \max$  function returns the index of the maximum value in the classification score vector  $\mathcal{L}_i^j$  and  $y_j$  is the actual class. The first term calculates the classification accuracy of the selected subset of ConvNets over the testing dataset and the second term penalizes the classification accuracy based on the cardinality of set  $I$ . In other words, we are looking for a subset of  $N$  ConvNets where classification accuracy is high and the number of the ConvNets is as few as possible. In contrast to the LASSO formulation, selecting the value of  $\lambda$  is straightforward. For example, assume two subsets  $I_1$  and  $I_2$  including 4 and 5 ConvNets, respectively. Moreover, consider that the classification accuracy of  $I_1$  is 0.9888 and the classification accuracy of  $I_2$  is 0.9890. If we set  $\lambda = 3e^{-4}$  and calculate the score using (2), their score will be equal to 0.9876 and 0.9875. Thus, despite its higher accuracy, the subset  $I_2$  is not better than the subset  $I_1$  because adding an extra ConvNet to the ensemble improves the accuracy 0.02 % which is discarded by the penalizing. However, if we choose  $\lambda = 2e^{-4}$ , the subset  $I_2$  will have a higher score compared with the subset  $I_1$ . In sum,  $\lambda$  shows what is the expected minimum accuracy increase that a single ConvNet must cause in the ensemble.

We utilize an *evolutionary algorithm* for finding the subset  $I$ . Specifically, a population of 50 chromosomes are gener-

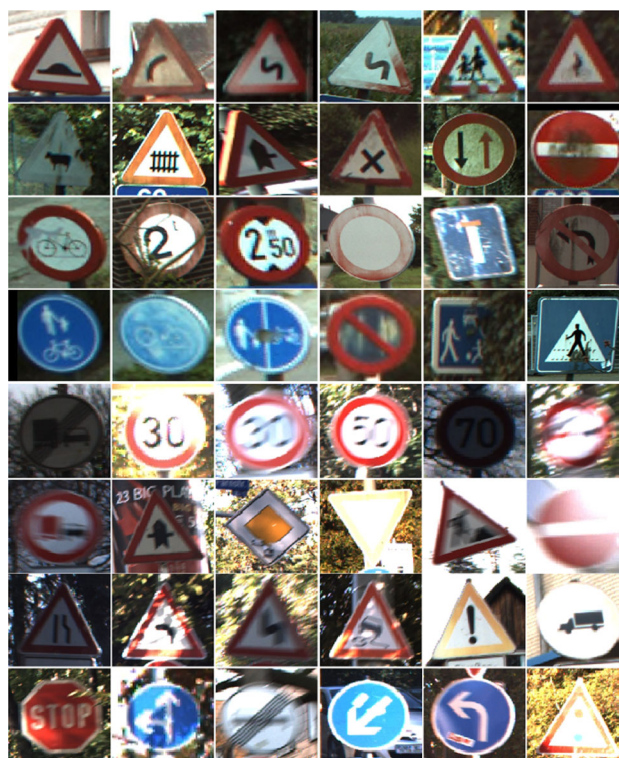


ated in which each chromosome is encoded using the N-bit *binary coding* scheme. A gene with value 1 indicates the selection of the corresponding ConvNet in the ensemble. The *fitness* of each chromosome is computed by applying (2) on the *validation set*. The offspring is selected using the *tournament* selection operator with tournament size 3. The crossover operators are *single-point*, *two-point* and *uniform* in which one of them is randomly applied in each iteration. The mutation operator flips the gene of a chromosome with probability  $p = 0.05$ . Finally, we also apply the *elitism* (with elite count = 1) to guarantee that the algorithm will not forget the best answer. Also, this can contribute for faster convergence by using the best individual so far during the selection process in the next iteration which may generate better answers during. As we show in the next section, using this technique, we are able to construct an optimal ensemble with higher accuracy and much fewer ConvNets compared with the ensembles proposed in Ciresan et al. (2012) and Jin et al. (2014).

## 4 Experiments

We apply our ConvNet on the German Traffic Sign Recognition Benchmark (GTSRB) dataset (Stallkamp et al. 2012). Each color image contains one traffic sign and they vary from  $15 \times 15$  to  $250 \times 250$  pixels. The training set consists of 39, 209 images and the test set contains 12, 630 images. We use 10 % of the training set (after augmenting the in the next section) for validation. We first crop the image within the annotated bounding box resize (downsample or upsample) all the images to  $48 \times 48$  pixels. Finally, the mean image is obtained over the whole dataset and it is subtracted from each image since a previous study (Coates and Ng 2012) suggests that subtracting the mean image increases the performance of the network<sup>2</sup>.

Also, we analyze the scalability of our ConvNet by the Belgium Traffic Sign Classification (BTSC) dataset (Timofte and Van Gool 2011). We inspected the dataset to make it consistent with the GTSRB. For instance, Class 32 in this dataset contains both signs “speed limit 50” and “speed limit 70”. However, these are two distinct classes in the GTSRB dataset. Therefore, we separated the overlapping classes in the BTSC dataset according to the GTSRB dataset. Each image in the BTSC dataset contains one traffic sign and it totally consists of 4672 color images for training and 2550 color images for testing. Finally, we normalize the dataset using the mean image obtained from the GTSRB dataset and resize all the images to  $48 \times 48$  pixels (Fig. 5).



**Fig. 5** Sample images from the BTSC (left) and the GTSRB (right) datasets

### 4.1 Data Augmentation

To reduce the effect of over-fitting, the original dataset is augmented 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  $5 \times 5$  Gaussian filter and median filters. 3) *HSV* The saturation of images are modified by transforming them into the HSV space and scaling the saturation component by factors 0.9 and 1.1. Moreover, the value component is scaled by factor 0.7. 4) *PCA* First, the matrix of principal components  $P \in \mathbb{R}^{3 \times 3}$  of the pixels of the training dataset is computed. Then, pixels of the image are projected into the PCA subspace to obtain the coefficient vector  $b$ . Next, a  $3 \times 1$  vector  $v$  is randomly generated in range  $[0, 0.5]$  and the image is reconstructed by back-projecting the pixels into 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.

### 4.2 Evaluation and Comparison

Each function in the transformation layer is implemented using a convolution layer with  $1 \times 1$  kernel and a bias para-

<sup>2</sup> The ConvNet architecture and its trained models are available at <https://github.com/penn/traffic-sign-recognition>.

**Table 1** Classification accuracy of the single network

Proposed (original)			Proposed (compact)		
Trial	Top 1 acc. (%)	Top 2 acc. (%)	Trial	Top 1 acc. (%)	Top 2 acc. (%)
1	98.87	99.62	1	99.11	99.63
2	98.98	99.64	2	99.06	99.64
3	98.85	99.62	3	98.88	99.62
4	98.98	99.58	4	98.97	99.61
5	98.99	99.63	5	99.08	99.66
6	99.06	99.75	6	98.94	99.68
7	98.99	99.66	7	98.87	99.60
8	99.05	99.70	8	98.98	99.65
9	98.88	99.57	9	98.92	99.61
10	98.77	99.60	10	99.05	99.63
Average	98.94 ± 0.09	99.64 ± 0.05	Average	98.99 ± 0.08	99.63 ± 0.02
Human	98.84	NA	Human	98.84	NA
Ciresan et al. (2012)	98.52 ± 0.15	NA	Ciresan et al. (2012)	98.52 ± 0.15	NA
Jin et al. (2014)	98.96 ± 0.20	NA	Jin et al. (2014)	98.96 ± 0.20	NA

*Left* The proposed network and *Right* its compact version

meter. The network is trained using the mini-batch stochastic gradient descent (batch size=50) with exponential weight annealing using the  $L_2$  regularized softmax loss function. We fixed the learning rate to 0.02, momentum to 0.9,  $L_2$  regularization coefficient to  $1e - 5$ , annealing parameter to 0.9998, dropout ratio to 0.5, the negative slope of the Leaky ReLU to 0.01 and the maximum number of iterations to 60000. The network is trained 10 times and their classification accuracies are calculated. The ConvNet is validated during the training process using the validation set in order to decide when to stop training. It is worth mentioning that [Stallkamp et al. \(2012\)](#) has only reported the classification accuracy and it is the only way to compare our results with [Ciresan et al. \(2012\)](#) and [Sermanet and Lecun \(2011\)](#). Similarly, [Jin et al. \(2014\)](#) evaluated their network using only the classification accuracy. In this paper, we report two different results based on the accuracy of a single network and the accuracy obtained by creating an ensemble using the algorithm mentioned in Sect. 3.1. In addition, the results are further analyzed using class specific *precision* and *recall* measures.

We trained our proposed network and its compact version 10 times and evaluated using the test set provided in the GTSRB dataset. Table 1 shows the results. The average classification accuracy of the 10 trials is 98.94 and 98.99 % for the original ConvNet and its compact version, respectively, which are both above the average human performance reported in [Stallkamp et al. \(2012\)](#). In addition, the standard deviations of the classification accuracy is small which show that the proposed architecture trains the networks with very close accuracies. We argue that this stability is the result of reduction in the number of the parameters and regularizing

the network using a dropout layer. Moreover, we observe that the top-2<sup>3</sup> accuracies are very close in all trials and their standard deviations are 0.05 and 0.02 for the original ConvNet and its compact version, respectively. In other words, although the difference in the top-1 accuracies of the Trial 1 and the Trial 2 in the original network is 0.11 %, notwithstanding, the same difference for top-2 accuracy is 0.02 %. This implies that there are images that are classified correctly in Trial 1 and they are misclassified in Trial 2 (or vice versa) but they are always within the top 2 scores of both networks. As a consequence, if we fuse the scores of the two networks the classification accuracy might increase.

The same argument is applied on the compact network, as well. Compared with the average accuracies of the single ConvNets proposed in [Ciresan et al. \(2012\)](#) and [Jin et al. \(2014\)](#), our proposed architecture and its compact version are more stable since their standard deviations are less than the standard deviations of these two ConvNets. In addition, despite the fact that the compact network has 52 % fewer parameters than the original network, the accuracy of the compact network is more than the original network and the two other networks. This confirms the claim illustrated by t-SNE visualization in Fig. 3 that the fully-connected layer in the original ConvNet does not increase the separability of the traffic signs. But, the fact remains that the compact network has less degree of freedom compared with the original network. Taking into account the Bias-Variance decomposition of the original ConvNet and the compact ConvNet, we

<sup>3</sup> The percent of the samples which are always within the top 2 classification scores.

**Table 2** Comparing the classification performance of the ensembles created by model averaging and our proposed method on the pools of original and compact ConvNets with three state-of-art ConvNets

Name	No. of ConvNets	Accuracy (%)	$F_1$ -score
Ens. of original ConvNets (our)	5	99.61	0.994
Ens. of original ConvNets (avg.)	10	99.56	0.993
Ens. of compact ConvNets (our)	2	99.23	0.989
Ens. of compact ConvNets (avg.)	10	99.16	0.987
Ciresan et al. (2012)	25	99.46	NA
Sermanet and Lecun (2011)	1	98.97	NA
Jin et al. (2014)	20	99.65	NA

**Table 3** Comparing the run-time efficiency of the ensembles created by model averaging and our proposed method on pools of original and compact ConvNets with three state-of-art ConvNets

Name	No. of ConvNets	No. of parameters	No. of multiplications
Ens. of original ConvNets (our)	5	1, 123, 449	382,699,560
Ens. of original ConvNets (avg.)	10	1, 123, 449	765,399,120
Ens. of compact ConvNets (our)	2	531, 999	151,896,924
Ens. of compact ConvNets (avg.)	10	531, 999	759,484,620
Ciresan et al. (2012)	25	1,543,443	3,208,042,500
Sermanet and Lecun (2011)	1	1,437,791	NA
Jin et al. (2014)	20	1,162,284	1,445,265,400

Note that we have calculated the *worst* case in our network by considering that every LReLU unit will perform one multiplications. In contrast, we have computed the *minimum* number of multiplications in Ciresan et al. (2012) by assuming that the input of *tanh* function always falls in range  $[-0.5, 0.5]$ . Similarly, in the case of Jin et al. (2014) we have considered fast but inaccurate implementation of *pow(float, float)*

claim that the compact ConvNet is more biased and its variance is slim compared with the original ConvNet. To prove this, we created two different ensembles using the algorithm mentioned in Sect. 3.1. More specifically, one ensemble was created by selecting the optimal subset from a pool of 10 original ConvNets and the second pool was created in the same way but from a pool of 10 compact ConvNets. Furthermore, two other ensembles were created by utilizing the model averaging approach (Ciresan et al. 2012; Jin et al. 2014), in which each ensemble contains 10 ConvNets. Tables 2 and 3 show the results and compare them with three state-of-art ConvNets.

First, we observe that our proposed method for creating ensemble is more efficient than the model averaging approach. To be more specific, the ensemble created using our algorithm on the pool of original ConvNets needs 50 % less multiplications<sup>4</sup> and its accuracy is 0.05 % higher com-

pared with the ensemble created by averaging all the original ConvNets in the pool (10 ConvNets). Taking into account the fact that there are 12, 630 test images, thus, the ensemble created by model averaging makes 6 more mistakes than our ensemble. Note that the number of ConvNets in the ensemble directly affects the number of the arithmetic operations required for making predictions. This means that the model averaging approach consume double CPU cycles compared with our ensemble.

Moreover, our ensemble reduces the number of the multiplications 88 and 73 % compared with the ensembles proposed in Ciresan et al. (2012) and Jin et al. (2014), respectively. More importantly, the dramatic reduction in the number of the multiplications causes only 5 more misclassification (0.04 % less accuracy) compared with the results obtained by the ensemble in Jin et al. (2014). We also observe that the ensemble in Ciresan et al. (2012) makes 19 more mistakes (0.15 % more misclassification) compared with our ensemble.

Besides, the number of the multiplications of the network proposed in Sermanet and Lecun (2011) is not accurately computable since its architecture is not clearly mentioned. However, the number of the parameters of this ConvNet is more than our original and compact networks. In addition, it

<sup>4</sup> We calculated the number of the multiplications of a ConvNet taking into account the number of the multiplications for convolving the filters of each layer with the N-channel input from the previous layer, number of the multiplications required for computing the activations of each layer and the number of the multiplications imposed by normalization layers. We showed in Sect. 3 that *tanh* function utilized in Ciresan et al. (2012) can be efficiently computed using 10 multiplications. ReLU activation used in Jin et al. (2014) does not need any multiplications and Leaky ReLU units in our ConvNet compute the results using only 1 multiplication. Finally, considering that *pow(float, float)* function needs only 1 multiplication and 64 shift operations (<http://tinyurl.com/yehg932>), the normalization

Footnote 4 continued  
layer in Jin et al. (2014) requires  $k \times k + 3$  multiplications per each element in the feature map.

**Table 4** Benchmarking time-to-completion of our ConvNet along with the compact ConvNet and Jin et al. (2014) obtained by running the forward-pass of each ConvNet 200 times and computing the average time for completing the forward-pass

	Proposed ConvNet	Proposed compact ConvNet	Jin et al. (2014)
CPU	12.96 ms	12.47 ms	14.47 ms
GPU	1.06 ms	1.03 ms	1.45 ms
	Ens. of proposed ConvNet	Ens. of proposed compact ConvNet	Jin et al. (2014) ensemble
CPU	$5 \times 12.96 = 64.8$ ms	$2 \times 12.47 = 24.94$ ms	$20 \times 14.47 = 289.4$ ms
GPU	$5 \times 1.06 = 5.30$ ms	$2 \times 1.03 = 2.06$ ms	$20 \times 1.45 = 29.0$ ms

utilizes rectified sigmoid activation which needs 10 multiplications per each element in the feature maps. In sum, we can roughly conclude that the ConvNet in Sermanet and Lecun (2011) needs more multiplications than our proposed ConvNet. However, we observe that an ensemble of two compact ConvNets performs better than Sermanet and Lecun (2011) and, yet, it needs less multiplications and parameters.

Finally, although the single compact ConvNet performs better than single original ConvNet, nonetheless, the ensemble of compact ConvNets does not perform better. In fact, according to Table 2, an ensemble of two compact ConvNets shows a better performance compared with the ensemble of ten compact ConvNets. This is due to the fact that the compact ConvNet is formulated with much fewer parameters and it is more biased compared with the original ConvNet. Consequently, their representation ability is more restricted. For this reason, adding more ConvNets to the ensemble does not increase the performance and it always vary around 99.20 %. The original network is able to model more complex nonlinearities so it is less biased about the data and its variance is more than the compact network. Hence, the ensemble of the original networks possesses more discriminative representation which increases its classification performance. In sum, if run-time efficiency is more important than the accuracy, then, ensemble of two compact ConvNets is a good choice. However, if we need more accuracy and the computational burden imposed by more multiplications in the original network is negligible, then, the ensemble of the original ConvNets can be utilized.

It is worth mentioning that the time-to-completion (TTC) of ConvNets does not solely depend on the number of multiplications. Number of accesses to memory also affects the TTC of ConvNets. From the ConvNets illustrated in Table 3, a single ConvNet proposed in Jin et al. (2014) seems to have a better TTC since it needs less multiplications compared with our proposed ConvNet and its compact version. However, (Jin et al. 2014, Table IX) shows that this ConvNet needs to pad the feature maps before each convolution layer and there are three local response normalization layers in this ConvNet. For this reason, it needs more accesses to memory which can negatively affect the TTC of this Con-


vNets. To compute the TTC of these ConvNets in practice, we ran the ConvNets on both CPU (Inter Core i7-4960) and GPU (GeForce GTX 980). The hard disk was not involved in any other task and there were no running application or GPU demanding processes. The status of the hardware was fixed during the calculation of the TTC of ConvNets. Then, the average TTC of the forward-pass of every ConvNet was calculated by running each ConvNet 200 times. Table 4 shows the results in the scale of milliseconds for one forward-pass.

The results show that the TTC of single ConvNet proposed in Jin et al. (2014) is 12 and 37 % more than our proposed ConvNet when it runs on CPU and GPU, respectively. This is consistent with our earlier discussion that the TTC of ConvNets does not solely depend on arithmetic operations. But, the number of memory accesses affects the TTC. Also, the TTC of the ensemble of our original ConvNet is 78 and 81 % faster than the ensemble proposed in Jin et al. (2014). Two missing parts in Ciresan et al. (2012), Sermanet and Lecun (2011) and Jin et al. (2014) are that the stability and the generalization power of the ConvNets are not analyzed adequately. In the next sections, we will evaluate our network from different aspects.

### 4.3 Misclassified Images

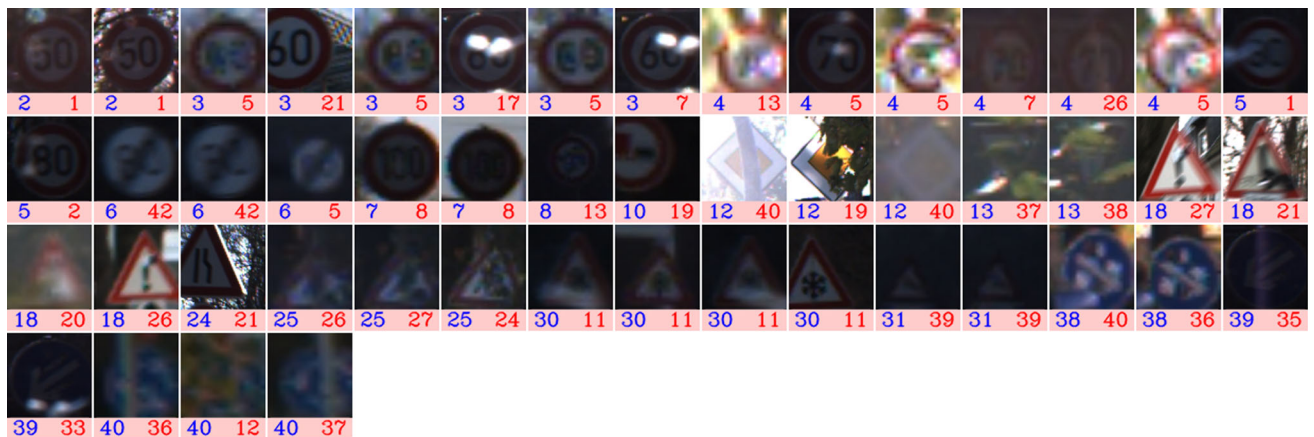
We computed the class-specific *precision* and *recall* (Table 5). Besides, Fig. 6 illustrates the incorrectly classified traffic signs. The blue and red numbers below each image show the actual and predicted class labels, respectively. For presentation purposes, all images were scaled to a fixed size. First, we observe that there are 4 cases where the images are incorrectly classified as class 11 while the true label is 30. Particularly, 3 of these cases are low-resolution images with poor illuminations. Moreover, class 30 is distinguishable from class 11 using the fine differences in the pictograph. However, rescaling a poorly illuminated low resolution image to  $48 \times 48$  pixels causes some artifacts on the image. In addition, two of these images are inappropriately localized and their bounding boxes are inaccurately. As the result, the network is not able to discriminate these two classes on these images. In addition, by inspecting the rest of the misclassi-

**Table 5** Class specific precision and recall obtained by our network



Class	Precision	Recall	Sup	Class	Precision	Recall	Sup	Class	Precision	Recall	Sup
0	1.00	1.00	60	15	1.00	1.00	210	30	1.00	0.97	150
1	1.00	1.00	720	16	1.00	1.00	150	31	1.00	0.99	270
2	1.00	1.00	750	17	1.00	1.00	360	32	1.00	1.00	60
3	1.00	0.99	450	18	1.00	0.99	390	33	1.00	1.00	210
4	1.00	0.99	660	19	0.97	1.00	60	34	1.00	1.00	120
5	0.99	1.00	630	20	0.99	1.00	90	35	1.00	1.00	390
6	1.00	0.98	150	21	0.97	1.00	90	36	0.98	1.00	120
7	1.00	1.00	450	22	1.00	1.00	120	37	0.97	1.00	60
8	1.00	1.00	450	23	1.00	1.00	150	38	1.00	1.00	690
9	1.00	1.00	480	24	0.99	0.99	90	39	0.98	0.98	90
10	1.00	1.00	660	25	1.00	0.99	480	40	0.97	0.97	90
11	0.99	1.00	420	26	0.98	1.00	180	41	1.00	1.00	60
12	1.00	1.00	690	27	0.97	1.00	60	42	0.98	1.00	90
13	1.00	1.00	720	28	1.00	1.00	150				
14	1.00	1.00	270	29	1.00	1.00	90				

Images show corresponding class label of each traffic sign. The column support (sup) shows the number of the test images for each class



**Fig. 6** Incorrectly classified images. The *blue* and *red* numbers below each image show the actual and predicted class labels, respectively. The traffic sign corresponding to each class label is illustrated in Table 5 (Color figure online)

fied images, we realize that the wrong classification is mainly due to *occlusion of pictograph* or *low-quality* of the images. However, there are a few cases where the main reason of the misclassification is due to inaccurate localization of the traffic sign in the detection stage (*i.e.* inaccurate bounding box).

#### 4.4 Cross-Dataset Analysis and Transfer Learning

So far, we trained our ConvNet on the GTSRB dataset and achieved state-of-art results with much fewer arithmetic oper-

ations and memory accesses which led to a considerably faster approach for classification of traffic signs. In this section, we inspect how transferable is our ConvNet across different datasets. To this end, we first evaluate the cross-dataset performance of our network. To be more specific, we use our trained ConvNet to predict the class of the traffic signs in the BTSC dataset. Among 73 classes in the BTSC dataset (after separating the overlapping classes), there are 23 common classes with the GTSRB dataset. We applied our ConvNet trained on the GTSRB dataset to classify these 23 classes inside both the *training* set and the *testing* set in the

**Table 6** Cross dataset evaluation of our trained ConvNet using the BTSC dataset

Class	Precision	Recall	Sup	Class	Precision	Recall	Sup	Class	Precision	Recall	Sup
0	NA	NA	0	15	0.91	0.86	167	30	NA	NA	0
1	NA	NA	0	16	1.00	0.78	45	31	NA	NA	0
2	NA	NA	0	17	1.00	0.93	404	32	NA	NA	0
3	NA	NA	0	18	0.99	0.93	125	33	NA	NA	0
4	1.00	0.93	481	19	1.00	0.90	21	34	NA	NA	0
5	NA	NA	0	20	0.93	0.96	27	35	0.92	1.00	96
6	NA	NA	0	21	0.92	0.92	13	36	1.00	0.83	18
7	NA	NA	0	22	0.72	1.00	21	37	NA	NA	0
8	NA	NA	0	23	1.00	0.95	19	38	NA	NA	0
9	0.94	0.94	141	24	0.66	1.00	21	39	NA	NA	0
10	NA	NA	0	25	0.90	1.00	47	40	0.99	0.87	125
11	0.88	0.91	67	26	0.75	0.86	7	41	NA	NA	0
12	0.97	0.95	382	27	NA	NA	0	42	NA	NA	0
13	0.97	0.99	380	28	0.89	0.91	241				
14	0.87	0.95	86	29	0.19	0.08	39				

Overall accuracy: 92.12 %

Class specific precision and recall obtained by our network are shown. The column support (sup) shows the number of the test images for each class. Classes with support equal to zero do not have any test cases in the BTSC dataset



**Fig. 7** Incorrectly classified images from the BTSC dataset. The blue and red numbers below each image show the actual and predicted class labels, respectively. The traffic sign corresponding to each class label is illustrated in Table 5 (Color figure online)

BTSC dataset. Table 6 shows the class specific precision and recall.

In terms of accuracy, the trained network has correctly classified 92.12 % of samples. However, precisions and recalls reveal that the classification of class 29 is worse than a random guess. To find out the reason, we inspect the misclassified images illustrated in Fig. 7.

Comparing the class 29 in the BTSC dataset with its corresponding class in the GTSRB (Table 5) shows that the pictograph of this class in the GTSRB dataset has significant differences with the pictograph of the same class in the BTSC dataset. In general, the misclassified images are mainly due to pictograph differences, perspective variation, rotation and blurriness of the images. We inspected the GTSRB dataset

**Table 7** Layers which are frozen and adjusted in each trial to evaluate the generality of each layer

ConvNet No.	Trans.	Conv1 layer 1	Conv2 layer 2	Conv3 layer 3	FC1 layer 4	Softmax layer 5
1	Fixed	Fixed	Fixed	Fixed	Fixed	Adjust
2	Fixed	Fixed	Fixed	Fixed	Adjust	Adjust
3	Fixed	Fixed	Fixed	Adjust	Adjust	Adjust
4	Fixed	Fixed	Adjust	Adjust	Adjust	Adjust
5	Fixed	Adjust	Adjust	Adjust	Adjust	Adjust

and found that perspective and rotation is more controlled than the BTSC dataset. As the result, our trained ConvNet has not properly captured the variations caused by different perspectives and rotations on the traffic signs. In other words, we argue that if we present adequate amount of data covering different combinations of perspective and rotation, our ConvNet is able to accurately model the traffic signs in the BTSC dataset.

To prove that, we try to find out how transferable is our ConvNet. We follow the same procedure mentioned in [Yosinski et al. \(2014\)](#) and evaluate the degree of transferability of our ConvNet in different stages. Concretely, the original ConvNet is trained on the GTSRB dataset. The Softmax loss layer of this network consists of 43 neurons since there are only 43 classes in the GTSRB dataset. We can think of the transformation layer up to the  $LReLU_4$  layer as a function which extracts the features of the input image. Thus, if this feature extraction algorithm perform accurately on the GTSRB dataset, it should also be able to model the traffic signs in the BTSC dataset. To evaluate the generalization power of the ConvNet trained only on the GTSRB dataset, we replace the Softmax layer with a new Softmax layer including 73 neurons to classify the traffic signs in the BTSC dataset. Then, we freeze the weights of all the layers except the Softmax layer and run the training algorithm on the BTSC dataset to learn the weights of the Softmax layer. Finally, we evaluate the performance of the network using the testing set in the BTSC dataset. This empirically computes how transferable is our network on other traffic signs datasets.

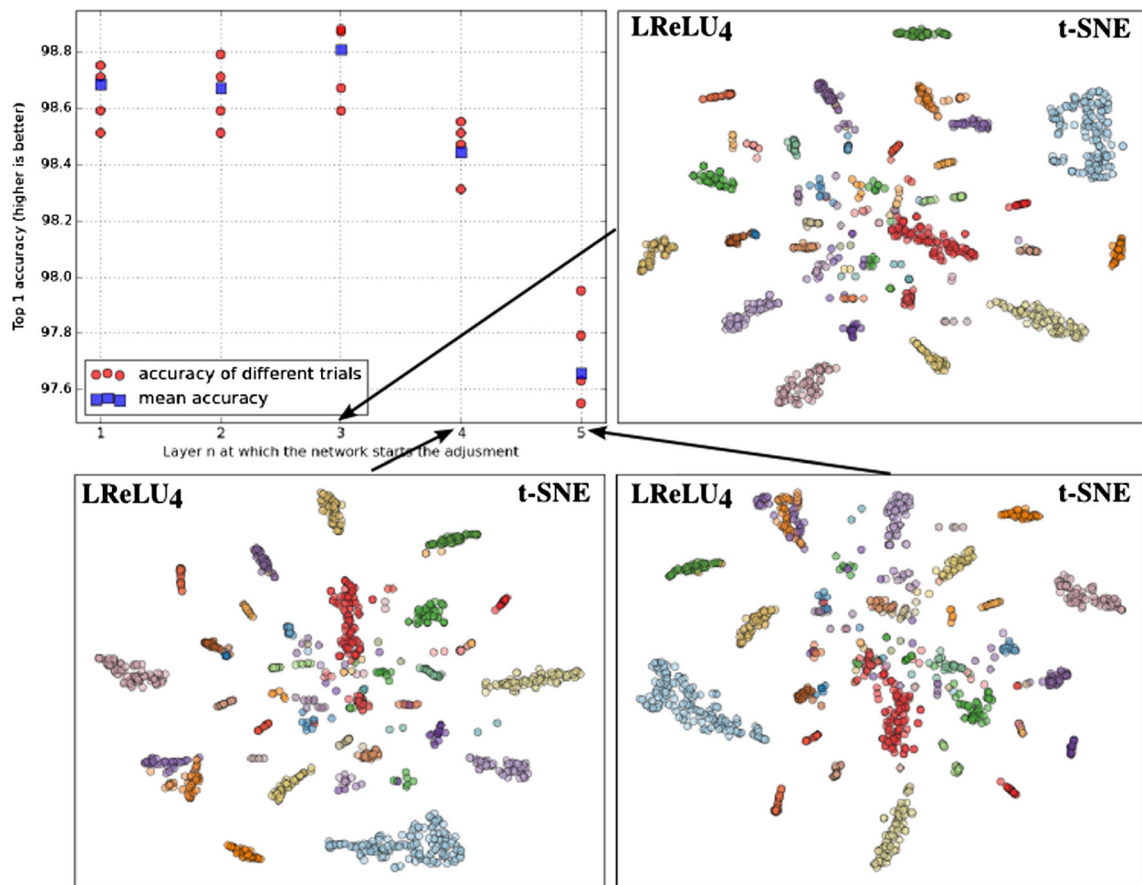
It is well-studied that the first layer of a ConvNet is more general and the last layer is more class specific. This means that the FC1 layer in our network (Fig. 3) is more specific than the C3 layer. In other words, the FC1 layer is adjusted to classify the 43 traffic signs in the GTSRB dataset. As the result, it might not be able to capture every aspects in the BTSC dataset. If this assumption is true, then we can adjust the weights in the FC1 layer beside the Softmax layer so it can model the BTSC dataset more accurately. Then, by evaluating the performance of the ConvNet on the testing set of the BTSC dataset we can find out to what extent the C3 layer is

able to adjust on the BTSC dataset. We increasingly add more layers to be adjusted on the BTSC dataset and evaluate their classification performance. At the end, we have 5 different networks with the same configuration but different weight adjustment procedures on the BTSC dataset. Table 7 shows the weights which are fixed and adjusted in each network. We repeated the training 4 times for each row in this table. Fig. 8 shows the results.

First, we observe that when we only adjust the softmax layer (layer 5) and freeze the previous layers, the accuracy drops dramatically compared with the results in the GTSRB dataset. In the one hand, layer 4 is adjusted such that the traffic signs in the GTSRB dataset become linearly separable and they can be discriminated using the linear classifier in the softmax layer. On the other hand, the number of the traffic signs in the BTSC dataset is increased 70 % compared with GTSRB dataset. Therefore, layer 4 is not able to linearly differentiate fine details of the traffic signs in the BTSC dataset. This is observable from the t-SNE visualization of the  $LReLU_4$  layer corresponding to  $n = 5$  in Fig. 8. Consequently, the classification performance drops because of overlaps between the classes.

If the above argument is true, then, fine-tuning the layer 4 beside the layer 5 must increase the performance. Because, by this way, we let the  $LReLU_4$  layer to be adjusted on the traffic signs included in the BTSC dataset. We see in the figure that adjusting the layer 4 ( $n = 4$ ) and the layer 5 ( $n = 5$ ) increases the classification accuracy from 97.65 to 98.44 %. Moreover, the t-SNE visualization corresponding to  $n = 4$  reveals that the traffic signs classes are more separable compared with the result from  $n = 5$ . Thus, adjusting both  $LReLU_4$  and Softmax layers make the network more accurate for the reason we mentioned above.

Recall from Fig. 3 that  $LReLU_4$  was not mainly responsible for increasing the separability of the classes. Instead, we saw in Sect. 4.2 that this layer mainly increases the variance of the ConvNet and improves the performance of the ensemble. In fact, we showed that traffic signs are chiefly separated using the last convolution layer. To further inspect this hypothesis, we fine-tuned the ConvNet on the BTSC



**Fig. 8** The result of fine-tuning the ConvNet on the BTSC dataset that is trained using GTSRB dataset. *Horizontal axis* shows the layer  $n$  at which the network starts the weight adjustment. In other words, weights of the layers before the layer  $n$  are fixed (frozen). The weights of layer  $n$  and all layers after layer  $n$  are adjusted on the BTSC dataset. We repeated

the fine-tuning procedure 4 times for each  $n \in \{1, \dots, 5\}$ , separately. *Red circles* show the accuracy of each trial and *blue squares* illustrate the mean accuracy. The t-SNE visualizations of the best network for  $n=3,4,5$  are also illustrated. The t-SNE visualization is computed on the  $LReLU_4$  layer (Color figure online)

dataset starting from layer 3 (*i.e* the last convolution layer). Fig. 8 illustrate an increase up to 98.80 % in the classification accuracy. This can be also seen on the t-SNE visualization corresponding to the layer 3 where traffic signs of the BTSC dataset become more separable when the ConvNet is fine-tuned starting from the layer 3.

Interestingly, we observe a performance reduction when the weights adjustment starts from layer 2 or layer 1. Specifically, the mean classification accuracy drops from 98.80 % in layer 3 to 98.67 % and 98.68 % in layers 2 and 1, respectively. This is due to the fact that the first two layers are more general and they do not significantly change from the GTSRB to the BTSC dataset. In fact, these two layers are trained to detect blobs and oriented edges. However, because the number of data is very few in the BTSC dataset compared with the number of the parameters in the ConvNet, hence, it adversely modifies the general filters in the first two layers which consequently affects the weight of the subsequent layers. As the result, the ConvNet overfits on data and does not

generalize well on the test set. For this reason, the accuracy of the network drops when we fine-tune the network starting from layer 1 or layer 2.

Finally, it should be noted that 98.80 accuracy is obtained using only a single network. As we showed earlier, creating an ensemble of these networks could improve the classification performance. In sum, the results obtained from cross dataset analysis and transferability evaluation reveals that our network is able to model a wide range of traffic signs and in the case of new datasets it only needs to be fine-tuned starting from the last convolution layer.

#### 4.5 Analyzing by Visualization

Visualizing ConvNets helps to understand them under different circumstances. In this section, we propose a new method for assessing the stability of the network and, then, conduct various visualization techniques to analyze different aspects of the proposed ConvNet.



#### 4.5.1 Stability of ConvNet

A ConvNet is a non-linear function that transforms a D-dimensional vector into a K-dimensional vector in the layer before the classification layer. Ideally, small changes in the input should produce small changes in the output. In other words, if image  $f \in \mathcal{R}^{M \times N}$  is correctly classified as  $c$  using the ConvNet, then, the image  $g = f + r$  obtained by adding a small degradation  $r \in \mathcal{R}^{M \times N}$  to  $f$  must also be classified as  $c$ .

However,  $f$  is strongly degraded as  $\|r\|$  (norm of  $r$ ) increases. Therefore, at a certain point, the degraded image  $g$  is not longer recognizable. We are interested in finding  $r$  with minimum  $\|r\|$  that causes the  $g$  and  $f$  are classified differently. Szegedy et al. (2014b) investigated this problem and they proposed to minimize the following objective function with respect to  $r$ :

$$\begin{aligned} & \text{minimize } \lambda \|r\| + \text{score}(f + r, l) \\ & \text{s.t. } f + r \in [0, 1]^{M \times N} \end{aligned} \quad (3)$$

where  $l$  is the actual class label,  $\lambda$  is the regularizing weight and  $\text{score}(f + r, l)$  returns the score of the degraded image  $f + r$  given the actual class of image  $f$ . In our ConvNet, the classification score vector is 43 dimensional since there are only 43 classes in the GT SRB dataset. Denoting the classification score vector by  $\mathcal{L} \in [0, 1]^{43}$ ,  $\mathcal{L}[k]$  returns the score of the input image for class  $k$ . The image is classified correctly if  $c = \arg \max \mathcal{L} = l$  where  $c$  is the index of the maximum value in the score vector  $\mathcal{L}$ . If  $\max(\mathcal{L}) = 0.9$ , the ConvNet is 90% confident that the input image belongs to class  $c$ . However, there might be an image where  $\max(\mathcal{L}) = 0.3$ . This means that the image belongs to class  $c$  with probability 0.3. If we manually inspect the scores of other classes we might realize that  $\mathcal{L}[c_2] = 0.2$ ,  $\mathcal{L}[c_3] = 0.2$ ,  $\mathcal{L}[c_4] = 0.2$  and  $\mathcal{L}[c_5] = 0.1$  where  $c_i$  depicts the  $i$ th maximum in the score vector  $\mathcal{L}$ .

Conversely, assume two images that are misclassified by the ConvNet. In the first image,  $\mathcal{L}[l] = 0.1$  and  $\mathcal{L}[c] = 0.9$  meaning that the ConvNet believes the input image belongs to class  $l$  and class  $c$  with probabilities 0.1 and 0.9, respectively. But, in the second image, the beliefs of the ConvNet are  $\mathcal{L}[l] = 0.49$  and  $\mathcal{L}[c] = 0.51$ . Even though in both cases the images are misclassified, the degrees of misclassification are different.

One problem with the objective function (3) is that it finds  $r$  such that  $\text{score}(f + r, l)$  approaches to zero. In other words, it finds  $r$  such that  $\mathcal{L}[l] = \epsilon$  and  $\mathcal{L}[c] = 1 - \epsilon$ . Assume the current state of the optimization function is  $r_t$  where  $\mathcal{L}[l] = 0.3$  and  $\mathcal{L}[c] = 0.7$ . In other words, the input image  $f$  is misclassified using the current degradation  $r_t$ . Yet, the goal of the objective function (3) is to settle in a

point where  $\text{score}(f + r_t, l) = \epsilon$ . As the result, it might change  $r_t$  which results in a greater  $\|r_t\|$ . Consequently, the degradation found by minimizing the objective function (3) might not be optimal. To address this problem, we propose the following objective function to find the degradation  $r$ :

$$\text{minimize } \psi(\mathcal{L}, l) + \lambda \|\mathcal{L}\|_1 \quad (4)$$

$$\psi(\mathcal{L}, l) = \begin{cases} \beta \times \mathcal{L}[l] & \arg \max_c \mathcal{L} = l \\ \max(\mathcal{L}) - \mathcal{L}[l] & \text{otherwise} \end{cases} \quad (5)$$

In this equation,  $\lambda$  is the regularizing weight,  $\beta$  is a multiplier to penalize those values of  $r$  that do not properly degrade the image so it is not misclassified by the ConvNet and  $\|\cdot\|_1$  is the sparsity inducing term that forces  $r$  to be sparse. The above objective function finds the value  $r$  such that degrading the input image  $f$  using  $r$  causes the image to be classified incorrectly and the difference between the highest score in  $\mathcal{L}$  and the true label of  $f$  is minimum. This guarantees that  $f + r$  will be outside the decision boundary of actual class  $l$  but it will be as close as possible to this decision boundary.

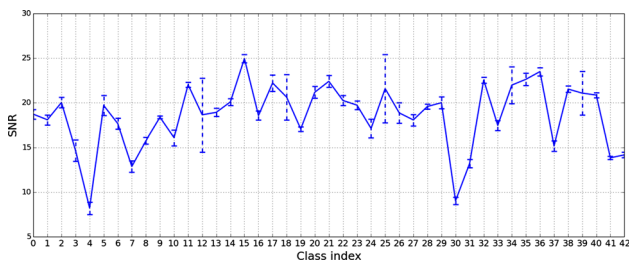
We minimize the objective function (4) using genetic algorithms. To this end, we use real-value encoding scheme for representing the population. The size of each chromosome in the population is equal to the number of the elements in  $r$ . Each chromosome, represents a solution for  $r$ . We use tournament method with tournament size 5 for selecting the offspring. Then, a new offspring is generated using arithmetic, intermediate or uniform crossover operators. Finally, the offspring is mutated by adding a small number in range  $[-10, 10]$  on some of the genes in the population. Finally, we use elitism to always keep the best solution in the population. We applied the optimization procedure for one image from each traffic sign classes. Fig. 9 shows the results.

Inspecting all the images in this figure, we realize that the ConvNet can easily make mistakes even for noises which are not perceivable by human eye. This conclusion is also made by Szegedy et al. (2014b). This suggests that the function presenting by the ConvNet is highly non-linear where small changes in the input may cause a significant change in the output. When the output changes dramatically, it might fall in a wrong class in the feature space. Hence, the image is incorrectly classified. Note that, because of our proposed objective function, the difference between the wrongly predicted class and the true class is positive but it is very close the decision boundary of the two classes. We repeated the above procedure on 15 different images and calculated the mean Signal-to-Noise Ratio (SNR) of each class, separately. Fig. 10 shows the results.

First, we observe that classes 4 and 30 have the lowest SNR values. In other words, the images from these two classes are more tolerant against noise. In addition, class 15 has the highest SNR values which shows it is more prone to be misclassified with small changes. Finally, most of the classes are



**Fig. 9** Minimum additive noise which causes the traffic sign to be misclassified by the minimum different compared with the highest score (best viewed in *color* and electronically) (Color figure online)

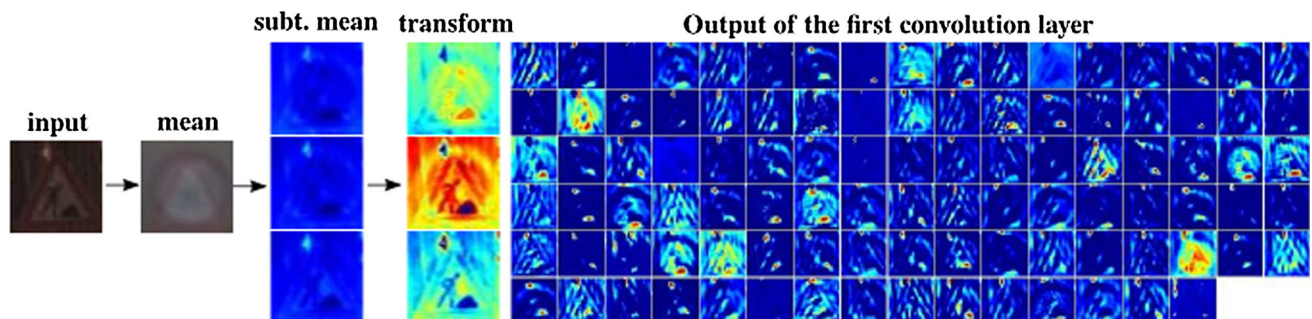


**Fig. 10** Plot of the SNRs of the noisy images found by optimizing (4). The mean SNR and its variance are illustrated

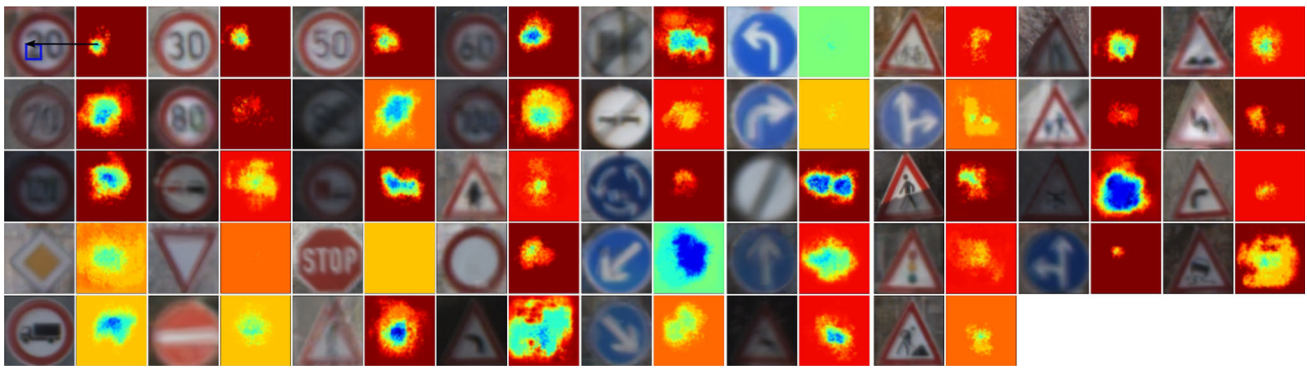
tolerant against noise with approximately the same degree of tolerance since they have close mean SNR values. One simple solution to increase the tolerance of the ConvNet is to augment noisy images with various SNR values so the network can learn how to handle small changes.

#### 4.5.2 Effect of Linear Transformation

We manually inspected the database and realized that there are images with poor illumination. In fact, the transformation layer enhances the illumination of the input image by multiplying the each channel with different constant factors and adding different intercepts to the result. Note that there is a unique transformation function per each channel. This is different from applying the same linear transformation function on all channels in which it does not have any effect on the results of convolution filters in the next layer (unless the transformation causes the intensity of the pixels exceed their limits). In our ConvNet, applying a different transformation function on each channel affects the output of the subsequent convolution layer. By this way, the transformation layer learns the parameters of the linear transformation such that it increases the classification accuracy. Fig. 11 illustrates the output of the transformation and the first convolution layers.



**Fig. 11** Visualization of the transformation and the first convolution layers



**Fig. 12** Classification score of traffic signs averaged over 20 instances per each traffic sign. The warmer color indicates a higher score and the colder color shows a lower score. The corresponding window of element  $(m, n)$  in the score matrix is shown for one instance. It should be

We observe that the input image suffers from a poor illumination. However, applying the linear transformation on the image enhances the illumination of each channel differently and, consequently, the subsequent layers represent the image properly so it is classified correctly.

#### 4.5.3 Visualizing Sensitivity

Assume we are given a pure image which is classified correctly by the ConvNet. We might be interested in localizing those areas on the image where degrading one of these areas by noise causes the image to be misclassified. This helps us to identify the sensitive regions of each traffic sign. To this end, we start from a window size equal to 20% of the image size and slide this window on the image. At each location, the region under the window is degraded by noise and the classification score of the image is computed. By this way, we obtain a score matrix  $H^c$  where element  $H^c(m, n)$  is the score of the image belonging to class  $c$  when a small region of the image starting from  $(m, n)$  is degraded by noise (*i.e.*  $(m, n)$  is the top-left corner of the window not its center). We computed the matrix  $H_i^c, i \in 1 \dots 20$  for 20 different instances of the same class and calculated the average matrix  $\bar{H}^c = \frac{\sum_{i=1}^{20} H_i^c}{20}$  as well as the average image. Fig. 12 illustrates the heat map of  $\bar{H}$ .

First, we observe that the ConvNet is mainly sensitive to small portion of the pictographs in the traffic signs. For example, in the speed limits signs related to speeds less than 100, it is clear that the ConvNet is mainly sensitive to some part of the first digit. Conversely, the score is affected by whole three digits in the “speed limit 100” sign. In addition, the score of the “speed limit 120” sign mainly depends on the second digit. These are all reasonable choices made by the ConvNet since the best way to classify two-digit speed limit signs is to compare their first digit. In addition, the

noted that the  $(m, n)$  is the top-left corner of the window not its center and the size of the window is 20% of the image size in all the results (best viewed in color and electronically) (Color figure online)

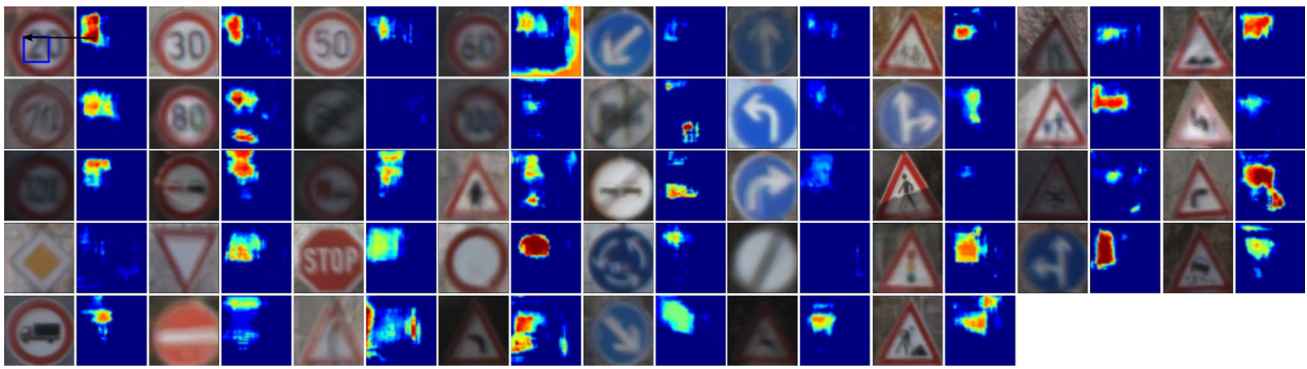
“speed limit 100” is differentiable from “speed limit 120” sign through only the middle digit.

Furthermore, there are traffic signs such as the “give way” and the “no entry” signs in which the ConvNet is sensitive in almost every location on the image. In other words, the score of the ConvNet is affected regardless of the position of the degraded region when the size of the degradation window is 20% of the image. We increased the size of the window to 40% and repeated the above procedure. Since the results are very close to the current experiment, we illustrated the results in the supplementary material.

#### 4.5.4 Visualizing the Minimum Perception

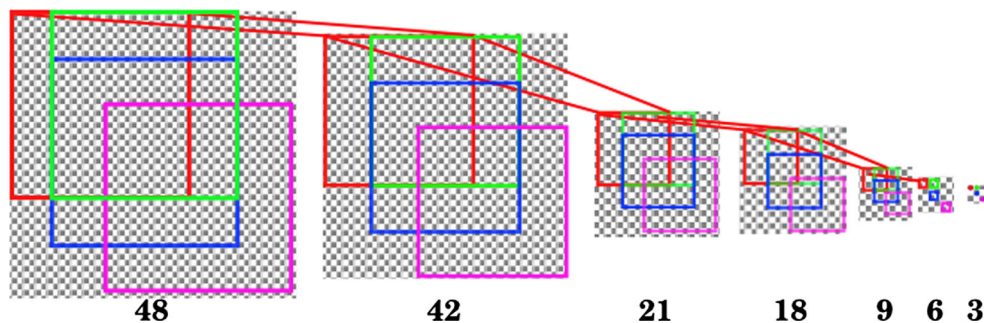
Classifying traffic signs at night is difficult because perception of the traffic signs is very limited. In particular, the situation is much worse in interurban areas at which the only lightening source is the headlights of the car. Unless the car is very close to the signs, it is highly probable that the traffic signs are partially perceived by the camera. In other words, most part of the perceived image might be dark. Hence, this question arises that “what is the minimum area to be perceived by the camera to successfully classify the traffic signs”.

To answer this question, we start from a window size equal to 40% of the image size and slide this window on the image. At each location, we keep the pixels under the window untouched and zero out the rest of the pixels. Then, the image is entered into the ConvNet and the classification score is computed. By this way, we obtain a score matrix  $H$  where element  $H(m, n)$  is the score of the traffic sign when only a small region of the image starting from  $(m, n)$  is perceived by the camera. As before, we computed the average score matrix  $\bar{H}$  using 20 instances for each traffic sign. Fig.



**Fig. 13** Classification score of traffic signs averaged over 20 instances per each traffic sign. The warmer color indicates a higher score. The corresponding window of element  $(m, n)$  in the score matrix is shown for one instance. It should be noted that the  $(m, n)$  is the top-left corner

of the window not its center and the size of the window is 40 % of the image size in all the results (best viewed in color and electronically) (Color figure online)



**Fig. 14** Receptive field of some neurons in the last pooling layer (best viewed in color) (Color figure online)

13 illustrates the heat map plot of  $\bar{H}$  obtained by sliding a window which its size is 40 % of the image size.

Based on this figure, we realize that in most of the traffic signs, the pictograph is the region with highest response. In particular, some parts of the pictograph has the greatest importance to successfully identify the traffic signs. However, there are signs such as the “priority road” sign which are not recognizable using 40 % of the image. It seems instead of pictograph, the ConvNet learns to detect color blobs as well as the shape information of the sign to recognize these traffic signs. We also computed the results obtained by increasing the window size to 60 %. Nonetheless, since the same analysis applies on these results we do not show them in this section to avoid redundancy of figures. But, these results are illustrated in the supplementary material.

#### 4.5.5 Visualizing Activations

We can think of the value of the activation functions as the amount of excitement of a neuron to the input image. Since the output of the neuron is linearly combined using the neuron in the next layer, then, as the level of excitement increases, it also changes the output of the subsequent neurons in the next

layer. So, it is a common practice to inspect which images significantly excite a particular neuron.

To this, we enter all the images in the test set of the GTSRB dataset into the ConvNet and keep record of the activation of neuron  $(k, m, n)$  in the last pooling layer where  $m$  and  $n$  depict the coordinates of the neuron in channel  $k$  of the last pooling result. According to Fig. 3, there are 250 channels in the last pooling layer and each channel is a  $3 \times 3$  matrix. Then, the images are sorted in descending order according to their value in position  $(k, m, n)$  of the last pooling layer and the average of the first 100 images is computed. It should be noted that each location  $(m, n)$  in the pooling layer has a corresponding receptive field in the input image. To compute the receptive field of each position we must back project the results from the last pooling layer to the input layer. Figure 14 shows the receptive field of some neurons in the last pooling layer.

We computed the average image of each neuron in the last pooling layer as we mentioned above. This is shown in Fig. 15 where each image  $im_i$  depicts the receptive field of the neuron  $(0, 0)$  from  $i$ th channel in the last pooling layer. According to these figures, most of the neurons in the last pooling layer are mainly activated by a specific traffic sign.



**Fig. 15** Average image computed over each of 250 channels using the 100 images with highest value in position (0, 0) of the last pooling layer. The corresponding receptive field of this position is shown using a cyan rectangle (Color figure online)

There are also some neurons which are highly activated by more than one traffic signs. To be more specific, these neurons are mainly sensitive to 2–4 traffic signs. By entering an image of a traffic sign to the ConvNet, some of these neurons are highly activated while other neurons are deactivated (they are usually close to zero or negative). The pattern of highly activated neurons are different for each traffic sign and this is the reason that the classes become linearly separable in the last pooling layer.

## 5 Discussion

In this paper, we proposed an efficient and accurate Convolutional Neural Network (ConvNet) for classifying traffic signs and proposed new methods for creating an ensemble and analyzing the stability of the network. Specifically, we reviewed two main approaches including traditional classification and end-to-end learning methods for solving this problem. We also discussed that the best performed method based on the traditional classification approach correctly classifies 98.53 % of test cases on the challenging German Traffic Sign Recognition Benchmark (GTSRB) dataset but it does not generalize well on other datasets. However, the best method based on an ensemble of ConvNets correctly classifies 99.65 % of test cases on the same dataset. Then, we explained four ConvNet based approaches for classifying traffic signs and mentioned their problems. In particular, we discussed that these networks are not computationally efficient to be implemented on an ADAS. In addition, behaviour of the networks on various circumstances are not properly analyzed.

We argued that any ConvNet for classifying traffic signs must be *computationally feasible*, *accurate* and *scalable* to be

utilized in real applications. With these three goals in mind, we proposed a new ConvNet with high accuracy which is computationally very efficient, as well. Then, we analyzed our network using the t-SNE visualization technique and showed that we can omit the fully-connected layer. By this way, we also created a compact version of our original proposed ConvNet. To be more specific, our original network is defined using 1, 123, 449 parameters and it reduces the number of the parameters 27, 22 and 3 % compared with three proposed ConvNets in Ciresan et al. (2012), Sermanet and Lecun (2011) and Jin et al. (2014), respectively. Moreover, our compact ConvNet requires 531, 999 parameters which is 65, 63 and 54 % reduction compared with the same three ConvNets.

It is previously shown that an ensemble of ConvNets can increase the classification performance. Yet, it dramatically increases the number of the arithmetic operations. In particular, the two accurate ConvNets in Ciresan et al. (2012) and Jin et al. (2014) create ensembles of 25 and 20 ConvNet in which these ensembles collectively need 3, 208, 042, 500 and 1, 445, 265, 400 multiplications to compute the classification score, respectively. The huge number of arithmetic operations are the result of redundancy in the ensembles, high number of parameters and choice of activation functions. We already solved the two latter issues in our original ConvNet. To address the redundancy in the ensemble, we proposed a new method and formulated the ensemble creation as an optimal subset selection problem. Using this formulation, we created two different ensembles using the original and compact ConvNets. The first ensemble consists of five original ConvNets and the second ensemble consists of two compact ConvNets which respectively need 382, 699, 560 and 151, 896, 924 multiplications to compute the classification score. Comparing with the two other ensembles, our first

ensemble reduces the number of the multiplications 88 and 73 %.

Our experiments on the GTSRB dataset showed that despite a huge reduction in the number of arithmetic operations, our ConvNet improves the classification accuracy 0.1 % compared with Ciresan et al. (2012) and its accuracy is only 0.04 % less than Jin et al. (2014). In other words, 73 % reduction in the number of the multiplications causes only 5 more misclassification compared with the ensemble in Jin et al. (2014). In addition, we showed that if the run-time has the greatest importance in our system, we can utilize the ensemble of compact networks which reduces the number of the multiplications 95 and 88 %, yet, the classification accuracy drops only 0.2 and 0.4 % compared with Ciresan et al. (2012) and Jin et al. (2014) respectively.

We further analyzed the stability, scalability and properties of the network through different techniques. First, we proposed a new method for finding the minimum degradation that causes the image to be misclassified with the minimum difference compared with the maximum posterior. Our results show that small variations in the input changes the output significantly. For this reason, the ConvNet might make mistakes even on images in which the degradation is not perceivable by human eye. In particular, we analyzed the Signal-to-Noise ratio of the results and found out that the level of tolerance does not change for most of the classes.

Then, we analyzed the cross-dataset generalization of our ConvNet by evaluating its performance on the BTSC dataset. We revealed that the ConvNet is sensitive to changes in pictographs and because of some differences in pictographs of the same class in two datasets, the accuracy of the ConvNet dropped to 92.12 % on BTSC dataset. We analyzed the misclassified images and found that the irregular rotation, perspective variations and pictograph differences have the major impacts on misclassifying the images. Then, we analyzed how transferable is our ConvNet when we adjust its weights to classify the 73 classes in the BTSC dataset.

For this purpose, we started by freezing all the layers of the network except the loss layer and fine-tuned only the weights of the loss layer. Then, weights of the input layer up to the last convolution layer were frozen and the weight of the two fully-connected layers were fine-tuned. We continued decreasing the number of the frozen layer and adjusting more layers. Our results illustrates that fine-tuning the network starting from the last convolution layer produces the best results. This is due to the fact that the ConvNet learns to detect various parts of the image in the last convolution layer. Therefore, when new classes are added to the database the ConvNet must be fine-tuned starting from the last convolution layer so the network can classify the new parts added by the new traffic signs in our dataset. Furthermore, we realized that fine-tuning the first two layers decreases the performance

since these two layers are more general and they increase the number of parameters to be adjusted. Therefore, when they are adjusted using a few number of samples, the network loses its generalization power.

We also analyzed the ConvNet using different visualization techniques. First, we inspected the activations of the neurons in the last convolution layer by keeping record of the activations for every image in the testing set. Then, we sorted the images according to the activation values of each neuron and computed the average image of the first 100 images with highest activations. This visualization shows that each neuron in the last convolution layer learns the parts belonging to only a few traffic signs. This also explains the reason that we need to fine-tune the ConvNet starting from the last convolution layer since when new parts or pictographs are added to the database, the activation pattern of the last pooling layer might overlap with other classes.

Second, we tried to locate the parts of the each traffic sign where they can negatively affect the posterior of the ConvNet if they are degraded by noise. To find out these locations, we start by moving a window over the image of a traffic sign and degrade the pixels under the window and keep clean the pixels outside the window. Then, we enter the image into the ConvNet and compute the posterior of the actual class. Next, we move the window pixel by pixel on the image and repeat the above process. At the end, we obtain a matrix where element  $(m, n)$  reflects the posterior of the ConvNet when we place the window in position  $(m, n)$  on the image and degrade the pixels under this window. We start with window size equal to 20% of the image size and repeat this procedure for several instances of the same traffic signs to compute the average matrix. Then, we show the average matrix using the heat map plot. According to the results, the ConvNet mainly learns to classify the traffic signs using the information coming from pictographs. In addition, there are also some traffic signs in which the network is almost sensitive to all locations. In these cases, the ConvNet also tries to learn the color and shape of the traffic signs. We further analyzed this behaviour by increasing the window size to 40 % of the image size.

Third, we inspected which parts of the image have the greatest importance for the ConvNet. In other words, which parts of the traffic sign are important to be perceived at night so the ConvNet can classify it correctly. To find out the answer, we used the same procedure we mentioned above. Here, we keep clean the pixels under the window and zero out the rest of the pixels. We realize that in most of the traffic signs, pictograph is the region with highest response. In particular, some parts of the pictograph has the greatest importance to successfully identify the traffic signs. However, some traffic signs are not recognizable using 40 % of the information. It seems, instead of pictograph, the ConvNet learns to detect color blobs as well as

the shape information of the sign to classify these traffic signs.

In sum, our proposed ConvNet and its compact version have important advantages over the previously proposed ConvNets in this field. Ensembles created by our ConvNets are computationally more efficient and very accurate. In addition, our ConvNets are easily transferable to new datasets with more classes of traffic signs.

**Acknowledgements** The authors are grateful for the support granted by Generalitat de Catalunya's Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) through FI-DGR 2015 and Martí Franquès 2015 fellowships.

## References

- Aghdam, H. H., Heravi, E. J., & Puig, D. (2015). A unified framework for coarse-to-fine recognition of traffic signs using Bayesian network and visual attributes. In: *10th international conference on computer vision theory and applications (VISAPP)* (pp. 87–96). doi:10.5220/0005303500870096
- Baró, X., Escalera, S., Vitrià, J., Pujol, O., & Radeva, P. (2009). Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification. *IEEE Transactions on Intelligent Transportation Systems*, 10(1), 113–126. doi:10.1109/TITS.2008.2011702.
- Ciresan, D., Meier, U., Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In: *2012 IEEE conference on computer vision and pattern recognition* (pp. 3642–3649). IEEE. doi:10.1109/CVPR.2012.6248110, arXiv:1202.2745v1
- Coates, A., & Ng, A. Y. (2012). Learning feature representations with K-means. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7700 LECTU:561–580, doi:10.1007/978-3-642-35289-8\_30
- Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In: *International conference on machine learning* (pp. 647–655) arXiv:1310.1531.
- Dosovitskiy, A., & Brox, T. (2015). Inverting convolutional networks with convolutional networks (pp. 1–15). arXiv preprint arXiv:1506.02753
- Fleyeh, H., & Davami, E. (2011). Eigen-based traffic sign recognition. *IET Intelligent Transport Systems*, 5(3), 190. doi:10.1049/iet-its.2010.0159.
- Gao, X. W., Podladchikova, L., Shaposhnikov, D., Hong, K., & Shevtsova, N. (2006). Recognition of traffic signs based on their colour and shape features extracted using human vision models. *Journal of Visual Communication and Image Representation*, 17(4), 675–685. doi:10.1016/j.jvcir.2005.10.003.
- Girshick, R., Donahue, J., Darrell, T., Berkeley, U. C., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. doi:10.1109/CVPR.2014.81, arXiv:1311.2524.
- Greenhalgh, J., & Mirmehdi, M. (2012). Real-time detection and recognition of road traffic signs. *IEEE Transactions on Intelligent Transportation Systems*, 13(4), 1498–1506. doi:10.1109/tits.2012.2208909.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852
- Hinton, G. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15, 1929–1958.
- Hsu, S. H., & Huang, C. L. (2001). Road sign detection and recognition using matching pursuit method. *Image and Vision Computing*, 19(3), 119–129. doi:10.1016/S0262-8856(00)00050-0.
- Huang, G., Mao, K. Z., Siew, C., Huang, D. (2013). A hierarchical method for traffic sign classification with support vector machines. In: *The 2013 international joint conference on neural networks (IJCNN)* pp 1–6. IEEE. doi:10.1109/IJCNN.2013.6706803
- Jin, J., Fu, K., & Zhang, C. (2014). Traffic sign recognition with hinge loss trained convolutional neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 15(5), 1991–2000. doi:10.1109/TITS.2014.2308281.
- Krizhevsky, A., Sutskever, I., Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, pp. 1097–1105. Curran Associates, Inc.
- Larsson, F., & Felsberg, M. (2011). Using Fourier descriptors and spatial models for traffic sign recognition. In: *Image analysis lecture notes in computer science* (Vol 6688, pp. 238–249). Springer. doi:10.1007/978-3-642-21227-7\_23
- Liu, H., Liu, Y., & Sun, F. (2014). Traffic sign recognition using group sparse coding. *Information Sciences*, 266, 75–89. doi:10.1016/j.ins.2014.01.010.
- Lu, K., Ding, Z., & Ge, S. (2012). Sparse-representation-based graph embedding for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 13(4), 1515–1524. doi:10.1109/TITS.2012.2220965.
- Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *International conference on machine learning (ICML) workshop on deep learning* (Vol 30)
- Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Computer vision and pattern recognition* (pp. 5188–5196). IEEE, Boston. doi:10.1109/CVPR.2015.7299155, arXiv:1412.0035
- Maldonado-Bascon, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gomez-Moreno, H., & Lopez-Ferreras, F. (2007). Road-sign detection and recognition based on support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2), 264–278. doi:10.1109/TITS.2007.895311.
- Maldonado Bascón, S., Acevedo Rodríguez, J., Lafuente Arroyo, S., Fernández Caballero, A., & López-Ferreras, F. (2010). An optimization on pictogram identification for the road-sign recognition task using SVMs. *Computer Vision and Image Understanding*, 114(3), 373–383. doi:10.1016/j.cviu.2009.12.002.
- Mathias, M., Timofte, R., Benenson, R., & Van Gool, L. (2013). Traffic sign recognition—How far are we from the solution? *International conference on neural networks*,. doi:10.1109/IJCNN.2013.6707049.
- Møgelmoose, A., Trivedi, M. M., & Moeslund, T. B. (2012). Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4), 1484–1497. doi:10.1109/TITS.2012.2209421.
- Moiseev, B., Konev, A., Chigorin, A., & Konushin, A. (2013). Evaluation of traffic sign recognition methods trained on synthetically generated data. In: *15th international conference on advanced concepts for intelligent vision systems (ACIVS)*, Springer, Poznań, pp 576–583, doi:10.1007/978-3-319-02895-8\_52
- Paclík, P., Novovičová, J., Pudil, P., & Somol, P. (2000). Road sign classification using Laplace kernel classifier. *Pattern Recognition Letters*, 21(13–14), 1165–1173. doi:10.1016/S0167-8655(00)00078-7.

- Piccioli, G., De Micheli, E., Parodi, P., & Campani, M. (1996). Robust method for road sign detection and recognition. *Image and Vision Computing*, 14(3), 209–223. doi:[10.1016/0262-8856\(95\)01057-2](https://doi.org/10.1016/0262-8856(95)01057-2).
- Ruta, A., Li, Y., & Liu, X. (2010). Robust class similarity measure for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4), 846–855. doi:[10.1109/TITS.2010.2051427](https://doi.org/10.1109/TITS.2010.2051427).
- Sermanet, P., & Lecun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of the international joint conference on neural networks* (pp. 2809–2813). doi:[10.1109/IJCNN.2011.6033589](https://doi.org/10.1109/IJCNN.2011.6033589)
- Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y. (2013). OverFeat : Integrated recognition , localization and detection using convolutional networks. In arXiv preprint [arXiv:1312.6229](https://arxiv.org/abs/1312.6229), pp. 1–15
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International conference on learning representation (ICLR)* (pp. 1–13), 1409.1556v5.
- Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: visualising image classification models and saliency maps. *arXiv preprint*, 13126034, 1–8.
- Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32, 323–332. doi:[10.1016/j.neunet.2012.02.016](https://doi.org/10.1016/j.neunet.2012.02.016).
- Sun, Z. L., Wang, H., Lau, W. S., Seet, G., & Wang, D. (2014). Application of BW-ELM model on traffic sign recognition. *Neurocomputing*, 128, 153–159. doi:[10.1016/j.neucom.2012.11.057](https://doi.org/10.1016/j.neucom.2012.11.057).
- Szegedy, C., Reed, S., Sermanet, P., Vanhoucke, V., & Rabinovich, A. (2014a). Going deeper with convolutions. In: arXiv preprint [arXiv:1409.4842](https://arxiv.org/abs/1409.4842), pp. 1–12.
- Szegedy, C., Zaremba, W., Sutskever, I. (2014b). Intriguing properties of neural networks. [arXiv:1312.6199v4](https://arxiv.org/abs/1312.6199v4)
- Tibshirani, R. (1994). *Regression Selection and Shrinkage via the Lasso..* doi:[10.2307/2346178](https://doi.org/10.2307/2346178).
- Timofte R, Van Gool, L. (2011). Sparse representation based projections. In: 22nd British Machine Vision Conference (pp. 61.1–61.12). BMVA Press. doi:[10.5244/C.25.61](https://doi.org/10.5244/C.25.61)
- Timofte, R., Zimmermann, K., & Van Gool, L. (2011). Multi-view traffic sign detection, recognition, and 3D localisation. *Machine Vision and Applications* (November):1–15. doi:[10.1007/s00138-011-0391-3](https://doi.org/10.1007/s00138-011-0391-3).
- Van Der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605. doi:[10.1007/s10479-011-0841-3](https://doi.org/10.1007/s10479-011-0841-3).
- Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *IEEE computer vision and pattern recognition (CVPR)* (pp. 3360–3367). doi:[10.1109/CVPR.2010.5540018](https://doi.org/10.1109/CVPR.2010.5540018).
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks ? *Neural Information Processing System (NIPS)*, 27. [arXiv:1411.1792v1](https://arxiv.org/abs/1411.1792v1).
- Yuan, X., Hao, X., Chen, H., & Wei, X. (2014). Robust traffic sign recognition based on color global and local oriented edge magnitude patterns. *IEEE Transactions on Intelligent Transportation Systems*, 15(4), 1466–1474. doi:[10.1109/TITS.2014.2298912](https://doi.org/10.1109/TITS.2014.2298912).
- Zaklouta, F., & Stanculescu, B. (2012). Real-time traffic-sign recognition using tree classifiers. *IEEE Transactions on Intelligent Transportation Systems*, 13(4), 1507–1514. doi:[10.1109/TITS.2012.2225618](https://doi.org/10.1109/TITS.2012.2225618).
- Zaklouta, F., & Stanculescu, B. (2014). Real-time traffic sign recognition in three stages. *Robotics and Autonomous Systems*, 62(1), 16–24. doi:[10.1016/j.robot.2012.07.019](https://doi.org/10.1016/j.robot.2012.07.019).
- Zaklouta, F., Stanculescu, B., & Hamdoun, O. (2011). Traffic sign classification using K-d trees and random forests. In *Proceedings of the international joint conference on neural networks* (pp. 2151–2155). doi:[10.1109/IJCNN.2011.6033494](https://doi.org/10.1109/IJCNN.2011.6033494).
- Zeiler, M., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European Conference on Computer Vision (ECCV)*, 8689, 818–833. doi:[10.1007/978-3-319-10590-1\\_53.1311.2901](https://doi.org/10.1007/978-3-319-10590-1_53.1311.2901).
- Zeng, Y., Xu, X., Fang, Y., & Zhao, K. (2015). Traffic sign recognition using deep convolutional networks and extreme learning machine. In *Intelligence science and big data engineering. image and video data engineering (IScIDE )* (pp. 272–280). Springer. doi:[10.1007/978-3-319-23989-7\\_28](https://doi.org/10.1007/978-3-319-23989-7_28).