A COMPARISON OF A CONVOLUTIONAL NEURAL NETWORK AND AN EXTREME LEARNING MACHINE FOR OBSCURED TRAFFIC SIGN RECOGNITION

Folke Drost, s2934833, f.g.drost@student.rug.nl
Supervisors: B.J. Wolf & Dr S.M. van Netten

Abstract: The present paper compares two neural network architectures for traffic sign recognition (TSR). First, a convolutional neural network (CNN) pre-trained for object recognition and retrained for TSR. Second, a single-hidden-layer feed forward neural network (SLFN), trained by an extreme learning machine (ELM) algorithm with a histogram of oriented gradient (HOG) feature extractor. The comparison focusses on recognition accuracy and computational costs regarding normal as well as obscured traffic signs. The models are trained and tested on a combination of traffic signs from the German TSR benchmark dataset, the Belgium traffic sign classification dataset and the revised mapping and assessing the state of traffic infrastructure (revised MASTIF) datasets. Results show an advantage of the ELM in recognition accuracy, computational costs and robustness on obscured traffic signs.

1. Introduction

Data driven traffic sign recognition (TSR) is used increasingly since its first introduction in 2008 in a commercial available product [1]. Current applications include driver assistance systems, advanced navigation systems and autonomous vehicles.

TSR systems operate by detecting the traffic sign through a camera. First, the area that the traffic sign is located in needs to be detected [2]. Second, a cropped image of this area is classified by a trained network. Here, image recognition faces a number of challenges.

An important step in image recognition, and therefor TSR, is how the cropped image is pre-processed by the network. The crop might include surrounding or other information not necessarily needed for a good classification. Traffic signs were designed to be recognisable and easy to read through the use of specific shapes and colours. The symbols are distinctive to the background and the peripheral shape conveys the class of traffic signs. These are the key features to be extracted from the crop. However, using only these standard features for the processing can lead to unforeseen results if the traffic sign is in a (partly) obscured condition. Obscured conditions include for example viewpoint variations, vandalised traffic signs, bad weather or lighting conditions, and bad camera performance. This creates the need for robust feature extraction.

Feature extraction reduces the dimension of the image by extracting useful information and discarding irrelevant information, creating a feature descriptor. Typically, this feature descriptor does not represent the image in a more useful or clear way for humans, but is very useful for tasks such as image recognition and object detection. Which features should be considered useful is one of the challenges of image recognition [3].

A common method used for feature extraction is the histogram of oriented gradients (HOG) [4]. The HOG method uses the distribution (histogram) of oriented gradients as features. The oriented gradients represent the direction of a group of pixels in the image. The magnitude of these gradients will be large around the edges where an abrupt change in intensity is present (i.e. when there is high contrast) and the orientation will be in the direction of the edge. The edges in an image contain much information about the shape of an object, especially concerning traffic signs which consist of highly contrasted shapes. This additionally ensures robustness against changes in illumination. An important choice during the feature extraction is the amount of local detail the HOG represents: A higher number of oriented gradients allows for more correct classifications of detailed traffic signs (e.g. between a cyclist or pedestrian on a warning sign) but can also cause overfitting. Overfitting occurs

when noise in the training data is learned as features of the classes.
Thus, the balance between redundancy and local details is another dilemma in image recognition.

The ability to generalize of the final model is another challenge: International conventions have helped to standardise traffic signs styles across regions, but still existing differences can introduce difficulties for classification (***Figure 1***). The choice between training the model for all variations of a traffic sign (losing accuracy but improving the ability to generalize) or only for a specific region (increasing accuracy but compromising the ability to generalize) is an important decision.

Another main concern for TSR is the speed of recognition. Practical use of TSR concerns mostly high-speed vehicles in which samples from a camera need to be processed and classified quickly. The ecological validity of the TSR method should therefore not only be measured in accuracy but also recognition speed.

Furthermore, the number of different traffic signs in practice is quite high (60+), and the inter-class difference can be low (e.g. different speed limits). The choice of classifier for a multiclass classification problem such as this is subsequently important. Previous studies have used various methods including ensemble classifiers, and more recent, neural networks [5-10].

1.1. CNN

A successful feed-forward, deep neural network (DNN) algorithm is the convolutional neural network (CNN [11]). DNNs combine feature extraction and classification in one network. A DNN consists of many different layers, each layer tasked with extracting or classifying a specific feature of an object (i.e. a picture of a traffic sign). Since training occurs on all layers, one of the previous stated problems is resolved: The network itself finds the most useful features of the images in the training set.

Figure 1: Difference between traffic signs for the Netherlands, the United Kingdom, the United States of America and Australia

DNNs have shown high accuracy (99+%) in TSR [10]. However, the advantage of automatic feature extraction is similarly a disadvantage: Using DNNs includes losing control over the features that get extracted, so they cannot guarantee accurate performance in obscured conditions. This can be resolved by including obscured examples in the training set, which increases the risk of overfitting in edge cases. Additionally, it may be difficult to gather real world obscured examples. Furthermore, due to their many layers, DNNs are computationally expensive and slow for training as well as recognition.

CNNs uses multiple hidden layers which consist of convolutional layers, pooling layers, fully connected layers, concatenation layers and dropout layers.

In the convolutional layer, each neuron only processes information from a small part of the image, determined by the kernel. The kernel is comparable to the receptive field in vertebrates, where specific neurons only receive input from a specific part of the field of view. Mathematically speaking, the network does not use convolutions, but rather cross-correlation. The difference is that cross-correlation loops over the kernel from left to right and from top to bottom, while convolution functions in the opposite direction. The pixels of the image that the kernel overlaps will be multiplied and summed up to calculate the value of the pixel in the middle. The kernel will slide over the complete image to calculate a value for each neuron in the convolutional layer. For a fully connected feed-forward neural network the number of variables will be much bigger with increasing image size, every input pixel has its own node and weight. Convolutions solve this problem by decreasing the number of variables.

The pooling layer combines output from one layer into a single neuron for the next layer. In complicated CNNs, the first part of the network might be trained to classify a specific feature e.g. the shape. After this is determined, the next part of the CNN only needs the classification, which the pooling layer provides. An example of the functioning of a pooling layer would be taking the average of the output or only the maximum value.

A fully connected layer connects every output of one layer to every input of the fully connected

layer. A fully connected layer is identical to a traditional multi-layer perceptron neural network. The full connected layer is useful for classifying high-level features of previous layers.

A concatenation layer combines the output of various layers with various sizes for the next layers.

A dropout layer is used to prevent overfitting by disabling half the neurons in the layer to force the network to learn more general features. Which neurons are disabled, varies between steps. Thus, different neurons will learn the same features.

1.2. ELM

Previous research has also proposed using an extreme learning machine (ELM) [12, 13] algorithm for training single-hidden-layer feedforward neural networks (SLFNs) for TSR [5, 14]. An ELM uses one hidden layer in which the weights connecting the input layer to the hidden layer are randomly chosen. Training occurs on the weights from the hidden layer to the output layer. Since an ELM only has to be trained on these parts, instead of all layers for conventional DNNs, training speed is extremely fast. The trained network itself is also small compared to DNNs which leads to faster recognition speed.

Two methods for TSR are compared: The first method is a CNN, pre-trained for object recognition [15] and retrained for TSR. A CNN is selected because of its popularity and proven performance. The second is an ELM algorithm, with HOG used for feature extraction. An ELM is selected because it tries to solve two of the problems encountered in TSR, namely robustness to obscured conditions and recognition speed while keeping high accuracy.

2. Methods

For the comparison, the two models were trained on the training set and fine-tuned on the validation set. Finally, the accuracy was determined by testing the trained models on the test set. The obscured images were created from the test set.

2.1. Datasets

Three public available datasets are used to train and test the models:

- The German TSR benchmark (GTSRB) dataset.
  The GTSRB dataset contains 43 different classes of traffic signs. The dataset includes 39.253 images. These vary in size from 15x15 to 250 x 250 pixels in PPM format.

- The Belgium traffic sign classification (BTSC) dataset.
  The BTSC dataset contains a total of 7.221 images spread over 62 classes in PPM format.

- The revised mapping and assessing the state of traffic infrastructure (revised MASTIF) dataset.
  The revised MASTIF dataset contains 31 classes spread over 1.992 files in JPG format.

All datasets contain cropped images of labelled traffic signs. The crops leave on average about 5 pixels around the traffic sign. The crops differ drastically in quality: Not only resolution, but also conditions in which the picture was taken. The datasets include sharp, well lit, frontal images, but also those that are dark, blurred and taken from an angle. In addition, the aspect ratios are not always one.

The images from the GTSRB and BTSC dataset are converted from PPM to JPG. The conversion is performed with Python module scikit-image. The images from all three datasets are combined into 60 different groups and are given label names accordingly to the traffic sign.

All image names are randomized to prevent biases in the train/test/validation sets (by dividing the dataset alphabetically) caused by possible structures in the dataset. All images are reviewed and duplicates are removed. Most duplicates are found in the GTSRB dataset in which around 90% of the images are the same, only in a different resolution. Furthermore, all images are rescaled to 64 by 64 pixels using the OpenCV module in Python. Thus, most images are up- or downscaled and the aspect ratio is reshaped to one. The dataset contains a considerable number of dark images in which distinctive features are harder to identify (also for the HOG transform to be discussed later): A Python script is used (using the module PIL) to increase the brightness of all images by 160%.

The final dataset (including the test, train and validation images) contains 19.274 images spread over 60 classes. The lowest class (keepLeft) has 51 images while the highest (priorityRoad) has 1.412, with an average of 321 images per class. The complete distribution of samples per class can be seen in ***Appendix 1***.

Since there is disparity between the amounts of testing images (***Appendix 1***), generalisation is used to limit the number of images used for training of each class. This is done to reduce the bias of classes with a high number of training images against classes with a low number of images. The normalisation limit is set by hyperparameter optimisation.

Both models are also being trained on a traditional non-normalised training set, denoted with the suffix * in the results. Here 80% (15.420) of the dataset is used for training, and a separate 10% (1.927) to validate the intermediate accuracy of the parameters. Finally, 10% (1927) of the dataset is used to test the final model. Using a separate test- and validate set counters overfitting.

2.2. CNN design

The CNN is built in Python3[1] using TensorFlow[2]. The basic architecture used is the Inception v3 architecture model, pre-trained on ImageNet images [15] . A top layer is added and trained to the existing model to recognize the 60 classes of traffic signs. The top layer receives as input a 2048-dimensional vector for each image. A softmax layer is trained on top of this representation. The softmax layer contains 60 labels, corresponding to learning 60 + 2048 × 60 model parameters, which equals the learned biases and weights. The softmax layer calculates a prediction for every output class where all predictions summed are 1.

The pre-trained Inception v3 model is used since this has been trained with millions of images of ImageNet for image recognition; a time-, computational power- and data consuming process. The model is proven to be effective in image classification tasks [15]. The model uses 48 pre-trained layers. ***Figure 2*** shows the layout of the model, with the last red softmax layer as the layer we are adding and training.

Training the last layer required the creation of feature vector representations of the images in the dataset. The feature vectors are created by feeding the images to all layers of the network up to the final layer. The feature vectors represent the prediction of the model up to the final layer. Here the feature vectors are used to configure the classification of the final layer. The feature vectors are calculated before the training and are then stored in cache. The train set is used for training, the validation set to validate the intermediate accuracy of the parameters and the test set is used to test the final model. The final test predicts real-world performance on images that the parameters are not being fitted on.

The learning rate is set at 0.01 and 50.000 training steps are being taken. Each training step includes 128 randomly chosen images. The feature vectors of these images are loaded from cache and are evaluated by the final layer to create predictions. These are then compared with the actual labels to update the weights of the final layer through back-propagation. 500 images of the validation set are used every 20 training steps to test the accuracy of the model so far. When training is finished, the test set is used to measure the real-world performance.

The separate test set is used because the algorithm already uses the results of the validation steps to avoid overfitting during training, therefore also fitting on the validation

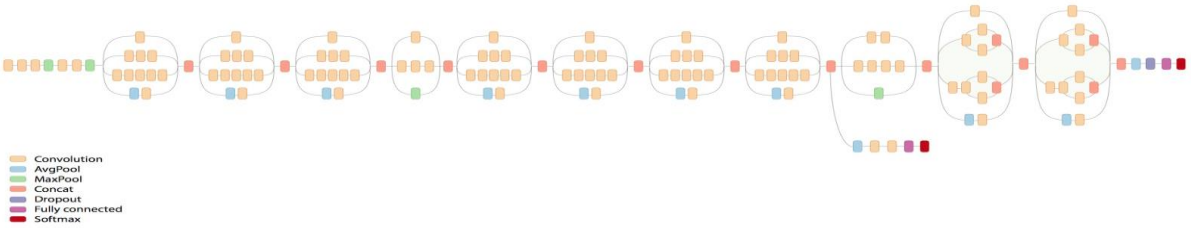


Figure 2: Design of Inception v3

[1] Python Software Foundation. Version 3.6. Available at https://www.python.org

[2] TensorFlow. Version 1.8.0. Available at https://www.tensorflow.org/

set. After training the model is saved to be loaded again for predictions of the obscured datasets.

2.3. ELM design

An ELM is a machine learning algorithm for a single-hidden-layer feed-forward neural network. The input weights W_i and bias b_i to the hidden-layer do not have to be tuned but are randomly assigned by a value between 0 and 1 before training starts. It has been shown that the random assignment of weights combined with a nonlinear piecewise activation function (sigmoid) causes a universal approximation [16]. Thus, through random selection of weights, connections from the input to hidden layer exist which otherwise would have been calculated by training. The weights from the hidden-layer to the output layer are calculated in a single step by feeding the whole training set in one step. A variation on this is the online sequential ELM (OS-ELM) which feeds the training set in batches to the ELM [17] .

The input layer of the ELM is connected to all N input images of feature vector X with dimension N × P, where P is the dimension of the feature vector of one image. The number of hidden nodes is denoted as L. The input layer is connected to the hidden layer by a function of feature mapping the P-dimensional space to the L-dimensional space. The output of the hidden layer, indexed by *i* is denoted as $g(X; W_i; b_i)$ where g is the activation function sigmoid, W_i is the weight vector of all weights from the input to the hidden node with index *i* and b_i is the bias of the hidden node with index *i*, where index $i = 1,\ldots,L$. The hidden layer output matrix H, with dimensions N × L, is then calculated by applying X over all L number of hidden nodes as in the following equation:

$$H = \begin{bmatrix} g(W_1 X_1 + b_1) & \cdots & g(W_L X_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(W_1 X_N + b_1) & \cdots & g(W_L X_N + b_L) \end{bmatrix}$$

The output layer consists of M nodes, in which M is the number of traffic sign classes (60). The output weights between the hidden nodes and the output nodes are denoted as the matrix β. The weight between a single hidden node and a single output node is denoted as β_{ij}, where *i* is the *i*th hidden node and *j* the *j*th output node. Every hidden node is connected to all output nodes. The output matrix Y, with dimension M, is calculated by the following equation:

$$Y = \begin{bmatrix} \sum H_1 \times \beta_{11} + \cdots + H_L \times \beta_{L1} \\ \vdots \\ \sum H_1 \times \beta_{1M} + \cdots + H_L \times \beta_{LM} \end{bmatrix}$$

The training images are fed to the network as N training sample pairs. Each pair consists of an image represented by a feature vector and a matching label, represented by a one-hot-vector. The one-hot-vector is a vector with the same length as the number of classes, with every entry zero except for a one for the right class. All N feature vectors form a matrix X with dimensions $N \times X_k$ where X_k is the feature vector of image k. The dimension P of X_k is determined by the HOG transform. All N one-hot-vectors form a matrix T with dimensions $N \times T_k$ where T_k is the one-hot-vector of image k. The dimension of T_k is 60 since the dataset has 60 different classes.

The training is performed by calculating the output weight matrix β. The output weight matrix β is calculated through a pseudo matrix inverse, depicted by the following equation:

$$\beta = TH^{\dagger},$$

where $H^{\dagger}$ is the MoorePenrose generalised inverse of matrix H.

The goal of the ELM is to minimize the training error as well as the norm of the output weights β. It does this through the following equation:

$$\text{Minimize: } \|T - H\beta\|^2 \text{ and } \|\beta\|$$

The ELM used for this study is built in Python3 using TensorFlow. This is chosen since it allows the CNN and ELM to be made in the same environment, reducing confounds for the comparison. The number of hidden nodes determines the network size and needs to be sufficient large to be able to learn all patterns. If it

is, however, too large, it does not learn specific enough.

2.3.1 HOG descriptor

The HOG descriptor extracts features from the input image by accumulating oriented gradients. A visualisation of the process can be seen in ***Figure 3***. The first step in the extraction is the pre-processing of the image. The JPG image is opened as a 64×64×3 matrix using the PIL and Numpy module. Each of the 64 × 64 = 4096 pixels includes three values for red, green and blue, each from 0 to 256. Next, the image is converted to greyscale using the skimage module.

The gradients are accumulated by dividing the image in cells of 6 × 6 pixels and blocks of 3 × 3 cells. Each pixel in a cell votes for one of nine orientations of bins, with its vote proportional to the gradient magnitude at that pixel. The votes are bilinearly interpolated in both position as orientation to reduce aliasing. This is ensured via also voting for the orientation bin of its neighbours. Similarly, the whole cell also votes for its neighbouring cells, resulting in a local histogram.

The energy of the local histograms over local blocks is used to normalise each cell in the block. The individual cells are shared between several blocks, but the normalisation is block dependent, resulting in cells appearing in the final output multiple times with different normalisations. These normalised blocks form HOG descriptors. Combining all HOG descriptors from all overlapping blocks results in the final HOG feature descriptor of the image. ***Figure 4*** shows the results of the HOG descriptor.

The values of the features vector are normalised between 0 and 1. This feature vector had the dimension 5.184, which results in an input layer of the ELM with 5.184 nodes as well.

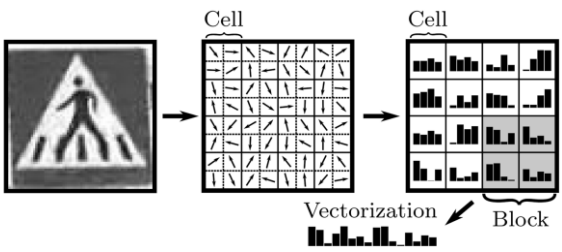


Figure 3: Extraction of HOG descriptor

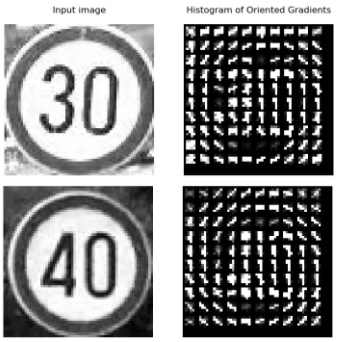


Figure 4: From greyscale to HOG descriptor for speed signs for 30 and 40

2.4. Obscuring traffic signs

The obscured images test set consists of ten randomly chosen images of each class, for a total of 600 images. Two types of obscuring methods are used for two datasets.

The first method, resulting in the test set named *blurred*, has blurred lines through the image. A nine-pixel-wide crop is taken from the image after which the crop is blurred. Two types of crops are taken through the middle of the images, one horizontal and one vertical. The crops are blurred and then pasted back at the same location on the original picture. The radius of the function is set at three pixels, resulting in a smooth blur where distinctive colour features are still slightly visible. Both blurs are saved, resulting in 2 * 600 = 1200 blurred images. An example can be seen in ***Figure 5***.

The second method, resulting in the test set named *black bar*, contains black bars crossing through the image. A five-pixel-wide bar through the middle of the images is made completely black. Four types of images are created, one with the bar horizontal, one with the bar the vertical, and two with the bar across both diagonals. The result are 2.400 images with black bars. An example can be seen in ***Figure 6***.

Figure 5: Blurred stop sign

Figure 6: Stop sign with black bars

2.5. Testing methods

The models' accuracies are measured by calculating the percentage of rightly classified images by comparing the predicted class with the true class. This assessment is done on the test sets. The accuracy is measured for the top one and top five predictions as outputted by the CNN and ELM. For the top one accuracy, only the prediction with the highest probability is compared to the true class, while for the top five, the five predictions with the highest probabilities are compared to the true class. If the true class matches the top one or one of the top five predictions, it is measured as a valid prediction.

The predictions are displayed in a confusion matrix which illustrates the performance per class in an easy to read format (overview in the ***Appendix***). Each row of the matrix represents an instance of the true class while each column represents an instance of the predicted class. The matrix is normalised from 0 to 1, were 1 would be a perfect prediction for that class. The testing methods are used on the test set as well as the obscured images.

The models' training and recognition time are measured on a Windows-based system. The system time is used to report the duration of the different processes. The experiments are run on a PC with the following hardware configuration:

-CPU: I5-3570k (4.5GHz)
-GPU: GTX1070
-Memory: 8 GB DDR3

3. Results

The performance of both networks was measured in recognition accuracy and computational efficiency. In the results ELM* and CNN* are trained on the full train set, while ELM and CNN are trained on a train set normalised to 550. The distribution of the datasets can be seen in ***Table 2***.

3.1. Hyperparameter optimisation ELM

Table 1 and ***2*** show the parameters of the HOG and ELM as found through hyperparameter optimisation. Recognition accuracy was used as the basic performance measure with respect to the parameters. ***Figure 7*** shows the recognition accuracy obtained by the ELM using different numbers of hidden nodes. The number of hidden nodes was set at L = 2000 for the final ELM. Through generalisation to a maximum of 550 images per class for the training set the total number of images trained on was 11414, the total number of images tested on 7860.

Table 1: Parameters of the HOG descriptor

	HOG
Image Size	64 × 64
Block Size	3 × 3
Cell Size	6 ×6
Bins	9
Dimension	5184

Table 2: Distribution of datasets.

	Dataset	***Dataset****
Normalised	Max 550 images per class	No normalisation
Training images	10957	15420
Testing images	6394	1927
Validation images	1923	1927

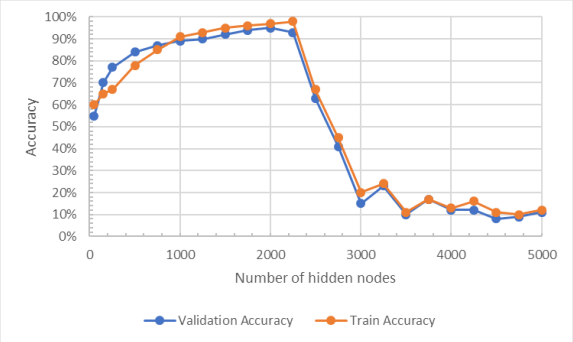


Figure 7: Effect of number of hidden nodes L on recognition accuracy on validation set

3.2. Recognition accuracy

Table 3, ***4*** and ***5*** show the achieved accuracy on the test set and obscured datasets. ***Figure 8*** shows a histogram of the accuracies. The tables show the accuracy when only a correct top prediction is measured as a hit (*Top 1*) and when a correct prediction in the top five predictions is measured as a hit (*Top 5*). ***Figure 9*** shows the progress of the accuracy of the training and validation set of the CNN over the 50000 learning steps.

Table 3: Performance of both networks on the test set.

Network	*Top 1*	*Top 5*
CNN	94.15%	**99.53 %**
ELM	**94.97%**	98.15%
CNN*	79.84%	90.60%
ELM*	**93.31%**	**97.86%**

Table 4: Performance of both networks on the obscured (blurred) test set.

Network	*Top 1*	*Top 5*
CNN	67.00%	92.58%
ELM	**87.92%**	**97.25%**
CNN*	57.25%	84.17%
ELM*	**76.50%**	**96.50%**

Table 5: Performance of both networks on the obscured (black bar) test set.

Network	*Top 1*	*Top 5*
CNN	31.71%	67.67%
ELM	**65.50%**	**88.83%**
CNN*	29.42%	62.08%
ELM*	**53.88%**	**86.92%**

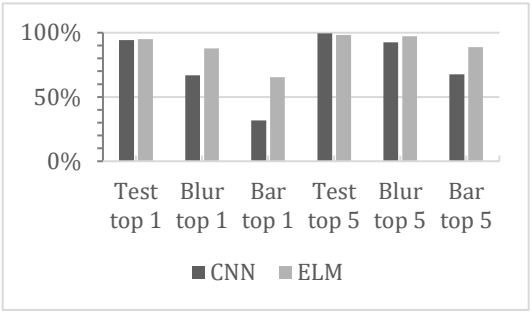


Figure 8: Accuracy of the CNN and ELM on the test and obscured sets

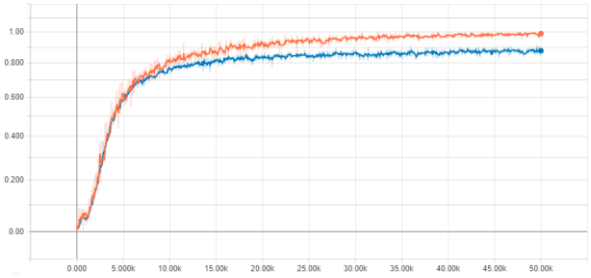


Figure 9: Progress of the accuracy of the CNN across 50k steps Orange line: training accuracy. Blue line: validation accuracy

3.3. Computational efficiency

Table 6 summarises the training and recognition time for both networks on the test set. The training and testing times also include the creation of the feature vectors. The recognition time for one prediction was measured by dividing the total testing time by the number of images.

Table 7 shows the recognition time for both networks on both obscured images sets.

Table 6: Computation time for both networks. $t_{recognition}$ is the time it takes for the prediction of 1 image by the trained network.

Network	*Train set size*	*Test set size*	t_{train}	t_{test}	$t_{recognition}$
CNN	10957	6394	7h 18m	1h 27m	820ms
ELM	10957	6394	**1m 29s**	**41.82s**	**5.32ms**
CNN*	15420	1927	7h 20m	38m 46s	1200ms
ELM*	15420	1927	**2m 5s**	**15.25s**	**8.07ms**

Table 7: Recognition time for both networks on the complete obscured traffic signs test sets.

Network	*blurred*	*black bar*	$t_{recognition}$
CNN	24m 8s	48m 2s	1207ms
ELM	**19.06s**	**34.07s**	**15.9ms**
CNN*	24m 5s	48m 15s	1204ms
ELM*	**21.87s**	**33.27s**	**16.4ms**

3.4. Misclassifications

The confusion matrices in the ***Appendix*** show the recognition performance of the two models on the test set. Some examples of the most pronounced classification errors of the CNN can be seen in ***Figure 10*** and of the ELM can be seen in ***Figure 11***.

Figure 10: Most pronounced misclassifications of the CNN. The upper row shows the correct sign, the lower row the predicted traffic sign (from left to right: junctionWithPriorityLeft, speedLimit70, speedLimit80, speedbump, doubleBendFirstRight, speedLimit120, speedLimit40, junctionWithPriorityRight)

Figure 11: Most pronounced misclassifications of the ELM. The upper row shows the correct sign, the lower row the predicted traffic signs (from left to right: doubleBendFirstLeft, steepHillUp, wildAnimals, doubleBendFirstRight, slipperyRoad, doubleBendFirstRight)

4. Discussion

From the results shown in ***Tables 3-5*** and ***Figure 8*** it can be concluded that the ELM showed similar performance to the CNN predicting the right traffic sign on the test set. On the obscured test sets, however, the ELM retains better performance, while the CNN's performance drops significantly. Comparing CNN and ELM with CNN* and ELM* shows a clear benefit of normalisation of the training set. The effect of normalisation results in better recognition of the test set as well as the obscured test sets, with the result of normalisation most pronounced in the top one prediction. From ***Table 6*** and ***7*** it becomes apparent that in computation time the ELM has a clear advantage over the CNN, with training and recognition times around a factor 200 faster. Similar studies show higher accuracy (+99%) for both a CNN and an ELM in TSR, but those often use a larger dataset and less classes, which hinders a direct comparison [5, 18]. Focussing the comparison on obscured and less than optimal conditions (e.g. classes with sometimes only around 50 examples), it was chosen to train on more classes and a smaller dataset. Taking the difference in datasets into consideration, the ELM and CNN show decent performance on the test set. No studies to compare the performance on the obscured test sets with were available. From the current project we can, however, conclude that here the performance of the ELM has an advantage over the CNN.

4.1. CNN

Taking a closer look at the confusion matrices in ***Appendix 2*** and ***3***, and ***Figure 10*** reveals that most problems of the CNN occurred regarding the speed signs and the warning signs. Similarity between the speed signs, with the only difference being the number, might cause the classification problems. The misclassification between the warning signs, similarly to the speed signs, can be most attributed to the limited training set. The most prominent features are the same in all warning signs (e.g. a black symbol on a white background inside a red triangle), with only the figure inside differing. More time and examples to learn the different figures might improve the ability of the CNN to distinguish the speed signs and warning signs from each other.

The confusion matrices of ***Appendix 6, 7, 10*** and ***11*** illustrate that the misclassified classes in the test set are also the classes with most errors in the obscured sets. For the blurred images classes with distinct enough features are classified correctly, but for the black bar set almost all classes have a large error. ***Figure 13*** shows samples of the classes often predicted wrongly by the CNN for images of the black bar set. For example, the warning signs, in which the symbol is now partially blocked by the bar, are classified as a blank warning sign. The round signs are confused for the no-turning-back-sign, which already has black bars in it. Lastly, we see that priority road at junction is classified very well, probably since the sign itself also has black bars running through it.

Thus, the addition of a black bar seems to cause two possible results. One, the shape of the traffic sign is recognised but the black bar ignored, resulting in the prediction of a blank sign with the same shape (e.g. the blank warning sign). Two, the black bar is interpreted as part of the sign resulting in the prediction of a sign with a bar (e.g. no-turn-left).

We do see, however, in all confusion matrices of the CNN that the right answer is often in the top five predictions. This would indicate that with more training the CNN might be able to get a better top one prediction since the correct prediction is already close to the top prediction.

Improvements to the CNN can still be made in terms of performance and computational time. A custom CNN can be designed, which does not have the complexity of a CNN previously trained for object recognition. This would decrease redundant layers and weights in the network, therefore speeding up the training and

recognition time. Having a CNN specifically designed and trained for TSR also allows the first layers of the CNN to determine the relevant features of the traffic signs, improving the accuracy. The network as used now, is trained on the features of thousands of objects, thus it is only a speculation that these features are similarly useful for TSR. ***Figure 9*** also suggest that the network might benefit from more training steps since the trend still shows a slight increase in accuracy at the end of training. Another point of improvement could be optimisation of the code itself. Updates to the TensorFlow framework might speed up computational times by better use of threads and/or GPUs and the computation might be done faster on specialised neural processing units (NPUs).

Figure 12: Classes the CNN often predicts the black bar set to be (from left to right: giveWay, noTurnLeft, priorityRoadAtJunction and priorityRoadAtJunction)

4.2. ELM

Taking a closer look at the confusion matrices ***Appendix 4, 5, 8, 9, 12*** and ***13***, and ***Figure 11*** reveals decent performance of the ELM for the test and blurred set, but worse performance for the black bar set. ***Figure 13*** shows the classes with the most problems in the black bar set. Comparing ***Figure 13*** with ***Figure 12*** indicates that most problems for the ELM arise for different signs than for the CNN. This provides some insight into the HOG method of feature extraction used by the ELM compared to the feature extraction of inception v3. The roadClosed class, which is a blank round sign, has trouble being correctly classified. This is logical since it is not supposed to contain any symbol (or black bar) in it. It also appears that a lot of input from the black bar set is mistaken for the sharedPath class. This might be due to the edge detection of the HOG extractor which has caused the ELM to have learned the vertical bar in the traffic sign. The vertical bar in some pictures of the black bar set might cause the algorithm to select sharedPath as prediction. When considering the top five predictions on the black bar set, this pattern mostly disappears, leaving only roadClosed as the most problematic class.

Improvements to the ELM could be made by exploring different feature extracting techniques, improvements to the training set and by exploring the OS-ELM variant. The feature extracting method was picked manually (with only hyperparameter optimisation) so it might be interesting to see how the network would perform with different methods (for example SIFT [19]). Alternatively, automatic feature extraction for example through the combination with a CNN could be used [7]. The first layers of a CNN could be trained for feature extraction specific for the traffic signs (instead of the HOG) after which the classification training is done by an ELM. A mayor feature of traffic signs are the colours used, but the HOG method shown uses only greyscale images. The CNN uses tricolour images as input, therefore an improvement could be made by letting the ELM similarly use tricolour input. Improvements to the dataset could be made by removing the background of the images, normalising the dataset to equal number of images per class, increasing the number of samples per class and providing a wider variation of real world obscured traffic signs. Furthermore, it would be interesting to see if statistical approaches for character recognition [20] could benefit TSR.

Figure 13: Problematic classes ELM black bar set (from left to right: sharedPath, roadClosed, roadNarrowBoth)

5. Conclusion

This paper compared a CNN, originally trained for object detection but retrained for TSR, with an ELM using a HOG descriptor for the recognition of obscured traffic signs. A dataset comprised of images of three datasets (GTSRB, BTSC and revised MASTIF) was used to train both models. Two obscured test sets, made by obscuring images of the normal test set, were used to evaluate the networks. Evaluations of the results for normal test set as well as the obscured sets show that the ELM has an advantage in recognition accuracy and in computational time

over the CNN. This results in the following advantages of the ELM over the CNN: The speed and robustness on obscured traffic signs makes it a promising method for high demanding real-time application. Furthermore, the fast training time allows for the training of larger datasets and fast updates to existing networks. Lastly, the small size of the trained ELM network together with the efficient computation allows the use of the network in a much broader range of devices.

Further possible improvements include the combination of a CNN and an ELM, the fine-tuning of a CNN for TSR, experimenting with different feature extractors for the ELM and the expansion of both methods to other applications.

References

[1] Anonymous '"Mobileye Vision Technologies; Combined Technologies From Continental and Mobileye Support the New Speed Limit Information of the New BMW 7 Series," *Entertainment & Travel*, Aug 30, pp. 214. (2008)

[2] J. Greenhalgh and M. Mirmehdi, '"Real-time detection and recognition of road traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1498-1506. (2012)

[3] A. Ihara, H. Fujiyoshi, M. Takaki, H. Kumon and Y. Tamatsu, '"Improvement in the accuracy of matching by different feature subspaces in traffic sign recognition," *IEEJ Transactions on Electronics, Information and Systems*, vol. 129, pp. 893-900. (2009)

[4] N. Dalal and B. Triggs, '"Histograms of Oriented Gradients for Human Detection," *Proceedings* /, vol. 1, pp. 886-893. (2005)

[5] Z. Huang, Y. Yu, J. Gu and H. Liu, '"An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine," *IEEE T.Cybern.*, vol. 47, no. 4, APR, pp. 920-933. (2017)

[6] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, '"Multi-column deep neural network for traffic sign classification." *Neural networks : the official journal of the International Neural Network Society*, vol. 32, pp. 333-8. (2012)

[7] Y. Zeng, X. Xu, Y. Fang and K. Zhao, '"Traffic sign recognition using deep convolutional networks and extreme learning machine,", pp. 272-280. (2015)

[8] Ke Lu, Zhengming Ding and S. Ge, '"Sparse-Representation-Based Graph Embedding for Traffic Sign Recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4. (2012)

[9] Y. Zhu, X. Wang, C. Yao and X. Bai, '"Traffic sign classification using two-layer image representation,", pp. 3755-3759. (2013)

[10] J. Jin, K. Fu and C. Zhang, '"Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1991-2000. (2014)

[11] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, '"Gradient-based learning applied to document recognition," *Proc IEEE*, vol. 86, no. 11, pp. 2278-2324. (1998)

[12] X. Liu, S. Lin, J. Fang and Z. Xu, '"Is Extreme Learning Machine Feasible? A Theoretical Assessment (Part I)," *IEEE Trans.Neural Netw.Learn.Syst.*, vol. 26, no. 1, JAN, pp. 7-20. (2015)

[13] S. Lin, X. Liu, J. Fang and Z. Xu, '"Is Extreme Learning Machine Feasible? A Theoretical Assessment (Part II)," *IEEE Trans.Neural Netw.Learn.Syst.*, vol. 26, no. 1, JAN, pp. 21-34. (2015)

[14] Z. Huang, Y. Yu and J. Gu, '"A novel method for traffic sign recognition based on extreme learning machine,", pp. 1451-1456. (2014)

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, '"Rethinking the inception architecture for computer vision,", pp. 2818-2826. (2016)

[16] G. Huang, L. Chen and C.K. Siew, '"Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans.Neural Networks*, vol. 17, no. 4, pp. 879-892. (2006)

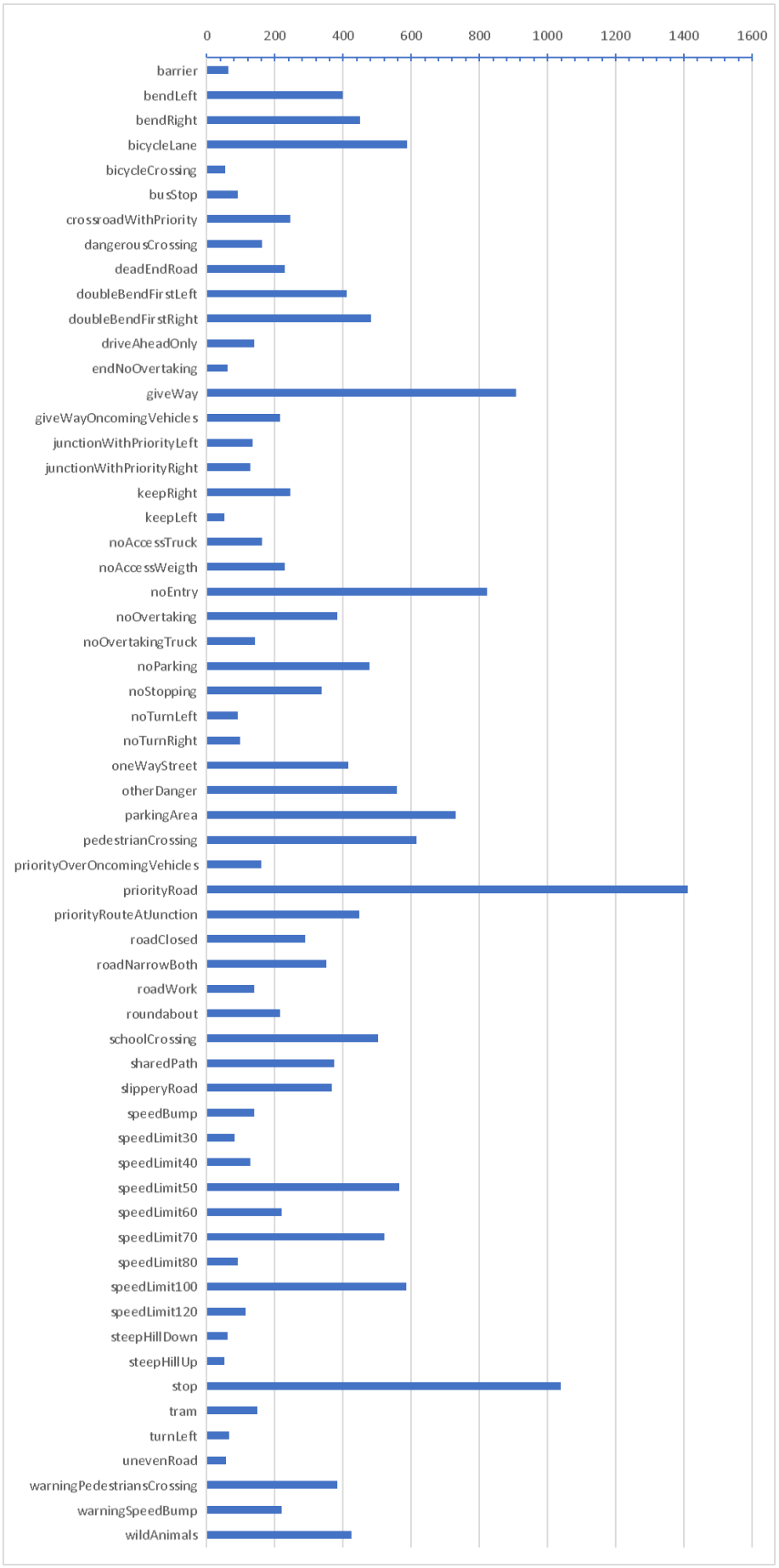
[17] Nan-Ying Liang, Guang-Bin Huang, P. Saratchandran and N. Sundararajan, '"A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE Trans.Neural Networks*, vol. 17, no. 6. (2006)

[18] D. CireşAn, U. Meier, J. Masci and J. Schmidhuber, '"Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333-338. (2012)

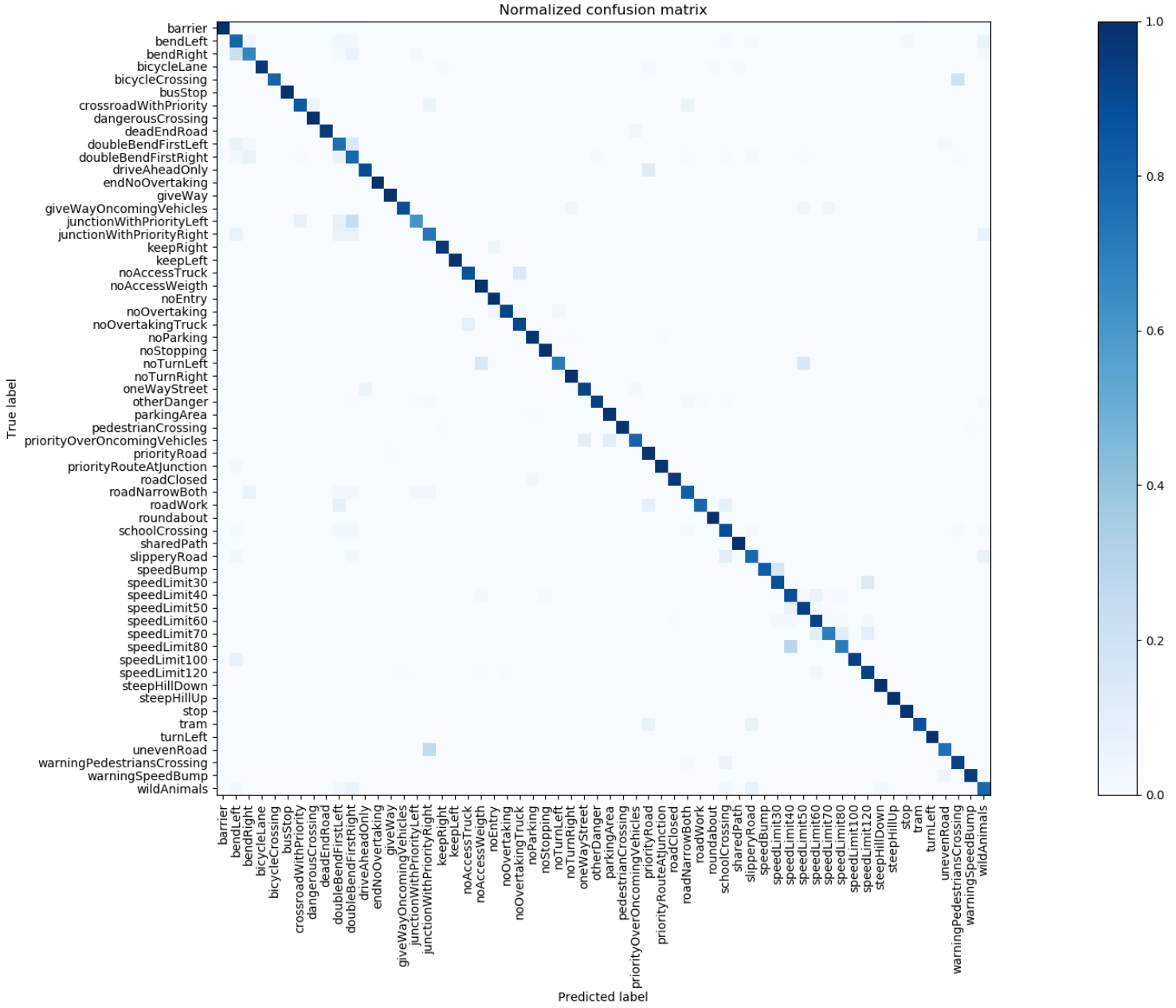
[19] M. Takaki and H. Fujiyoshi, '"Traffic sign recognition using SIFT features," *IEEJ Transactions on Electronics, Information and Systems*, vol. 129, pp. 824-831. (2009)

[20] H. Liu and X. Ding, '"Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes,", pp. 19-23. (2005)

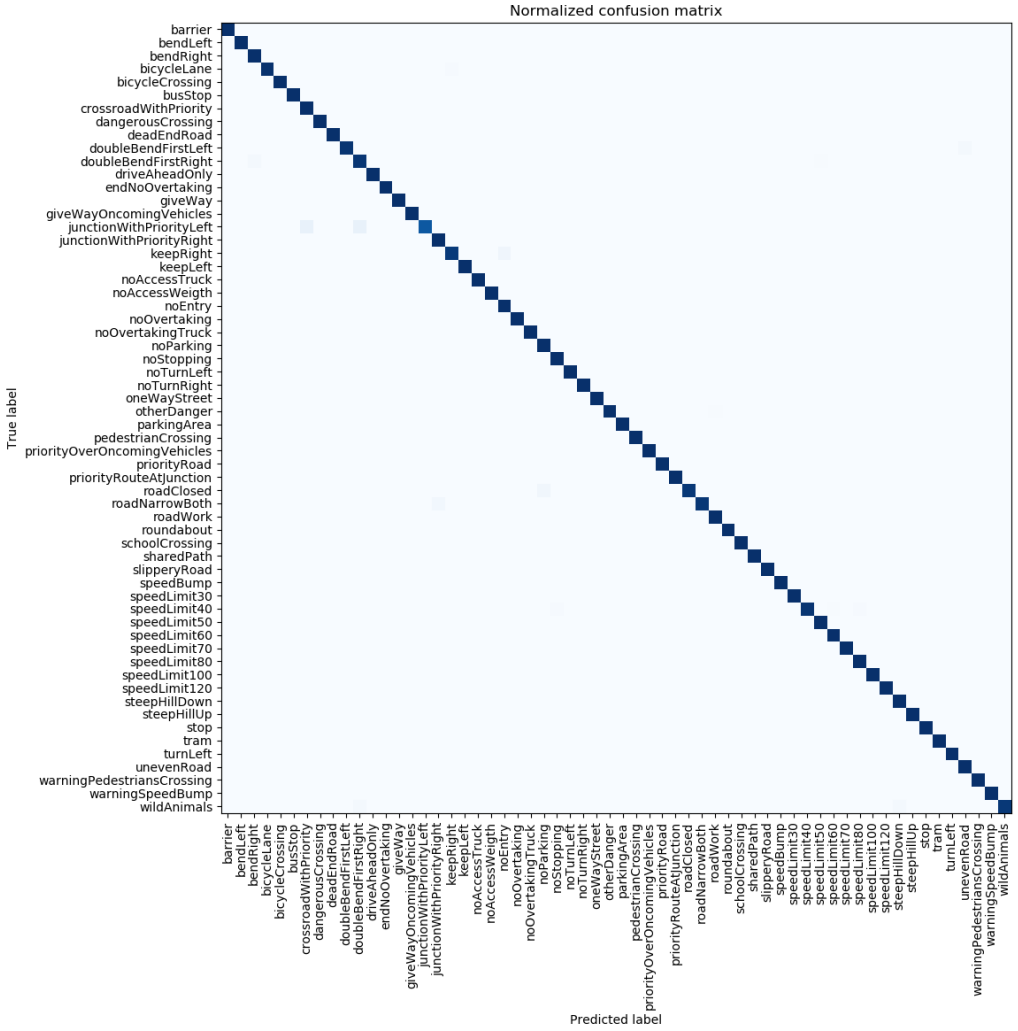
A. Appendix



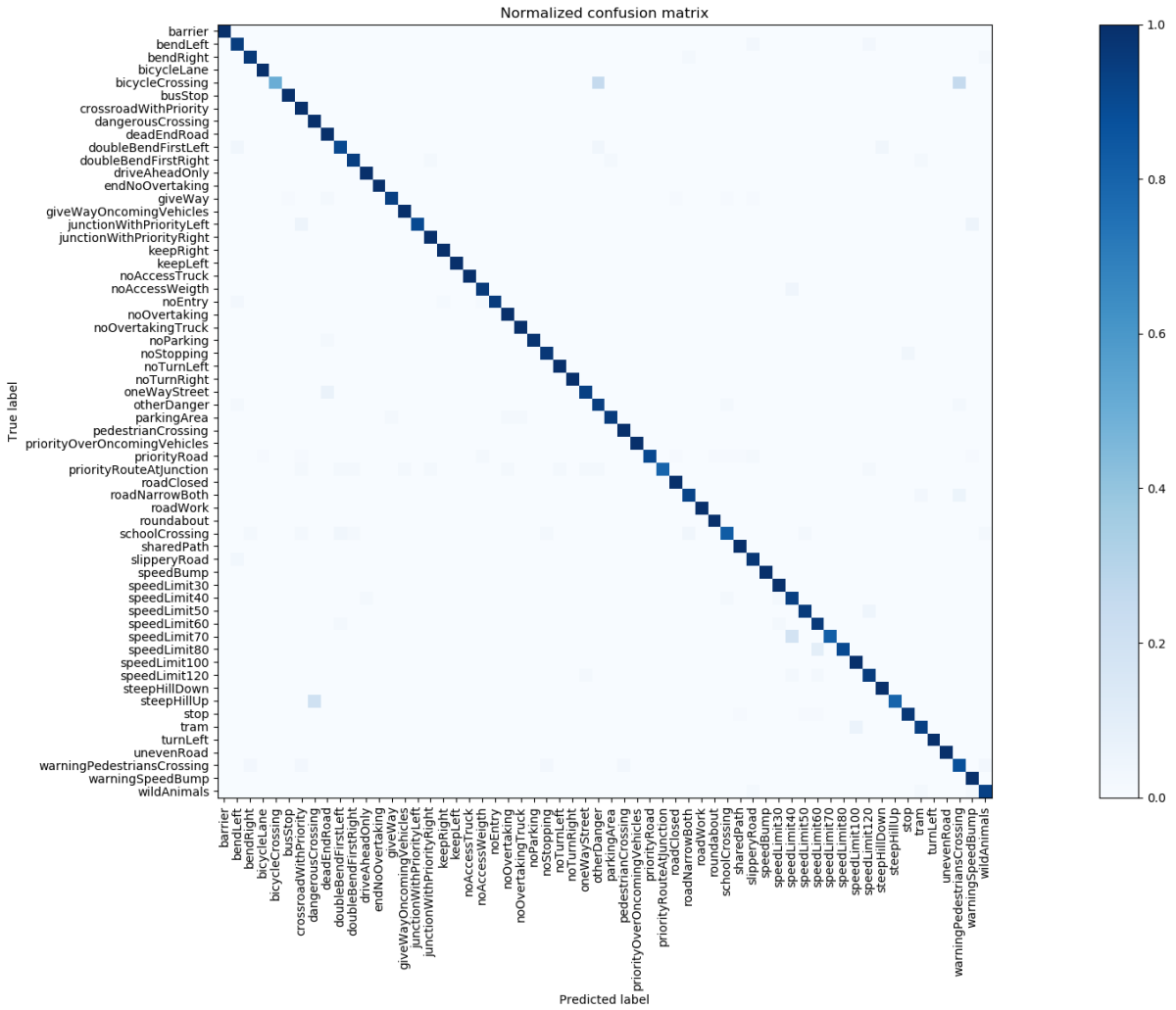
Appendix 1: Distribution of sample per class in the complete dataset



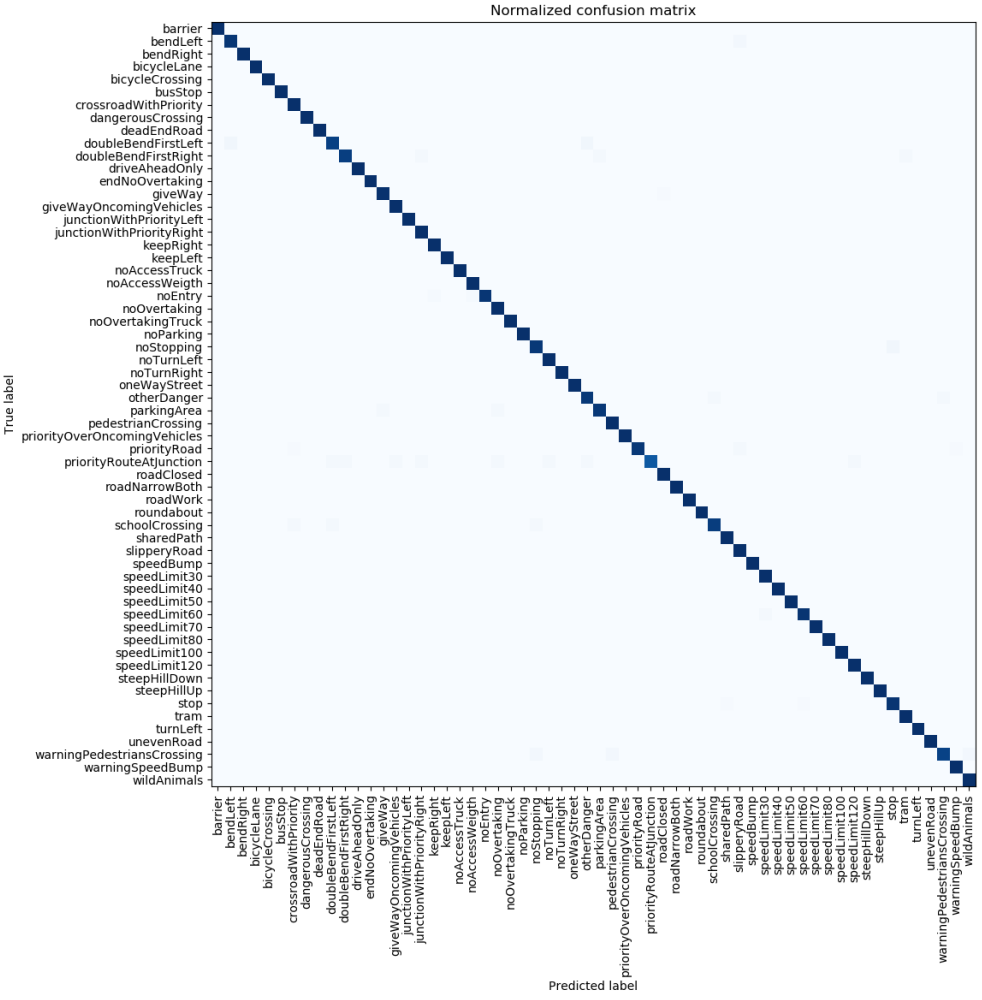
Appendix 2: Confusion matrix CNN for top 1 predictions on the test set (94.2% accuracy)



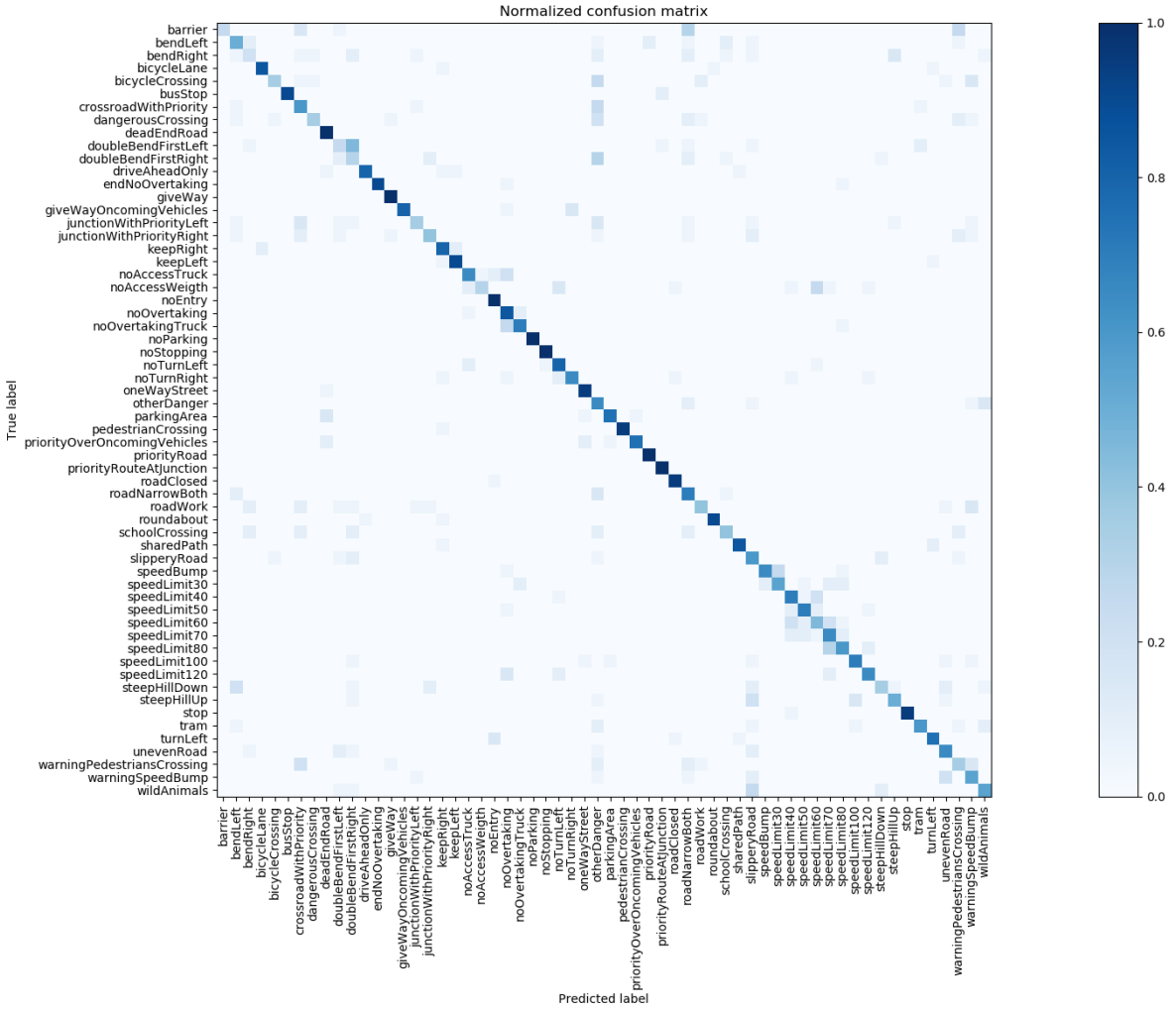
Appendix 3: Confusion matrix CNN for top 5 predictions on the test set (99.5% accuracy)



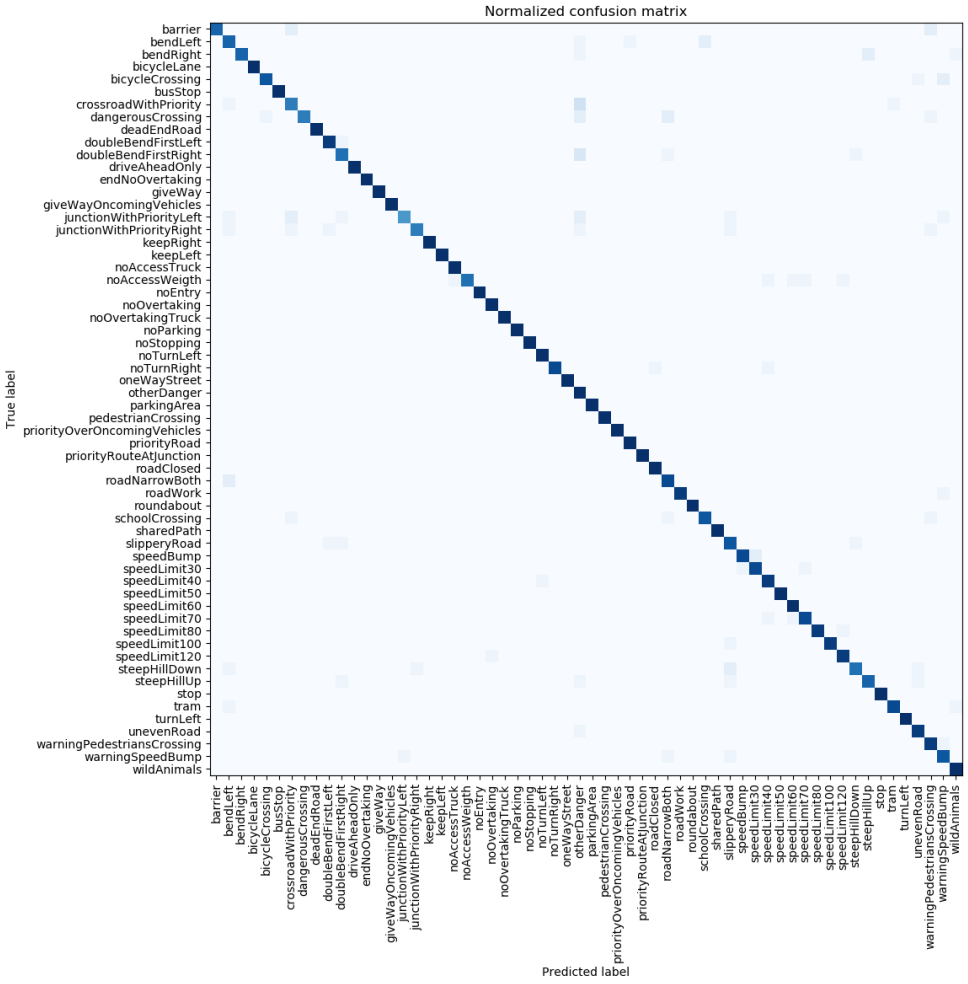
Appendix 4: Confusion matrix ELM for top 1 predictions on the test set (94.9% accuracy)



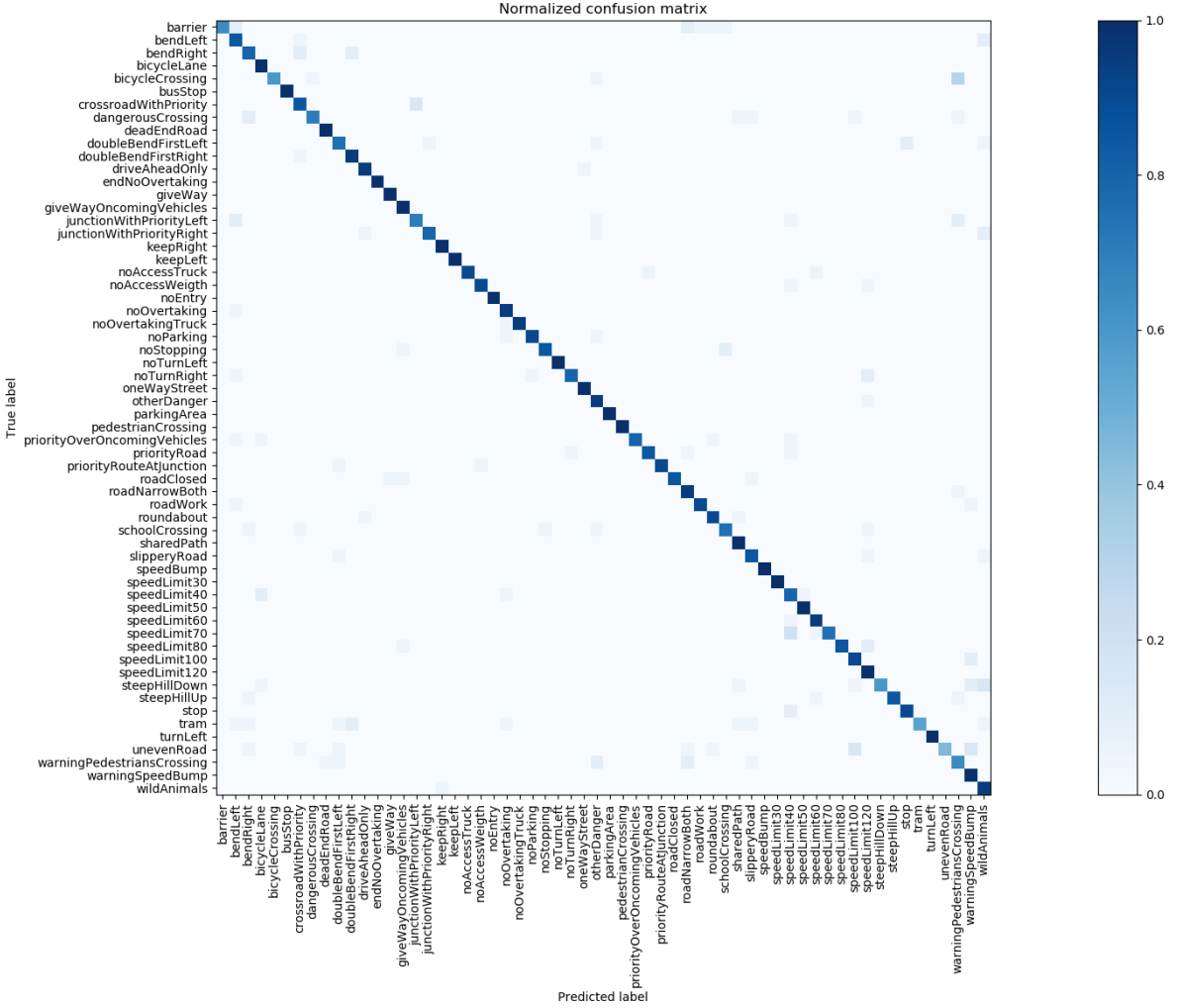
Appendix 5: Confusion matrix ELM for top 5 predictions on the test set (98.1% accuracy)



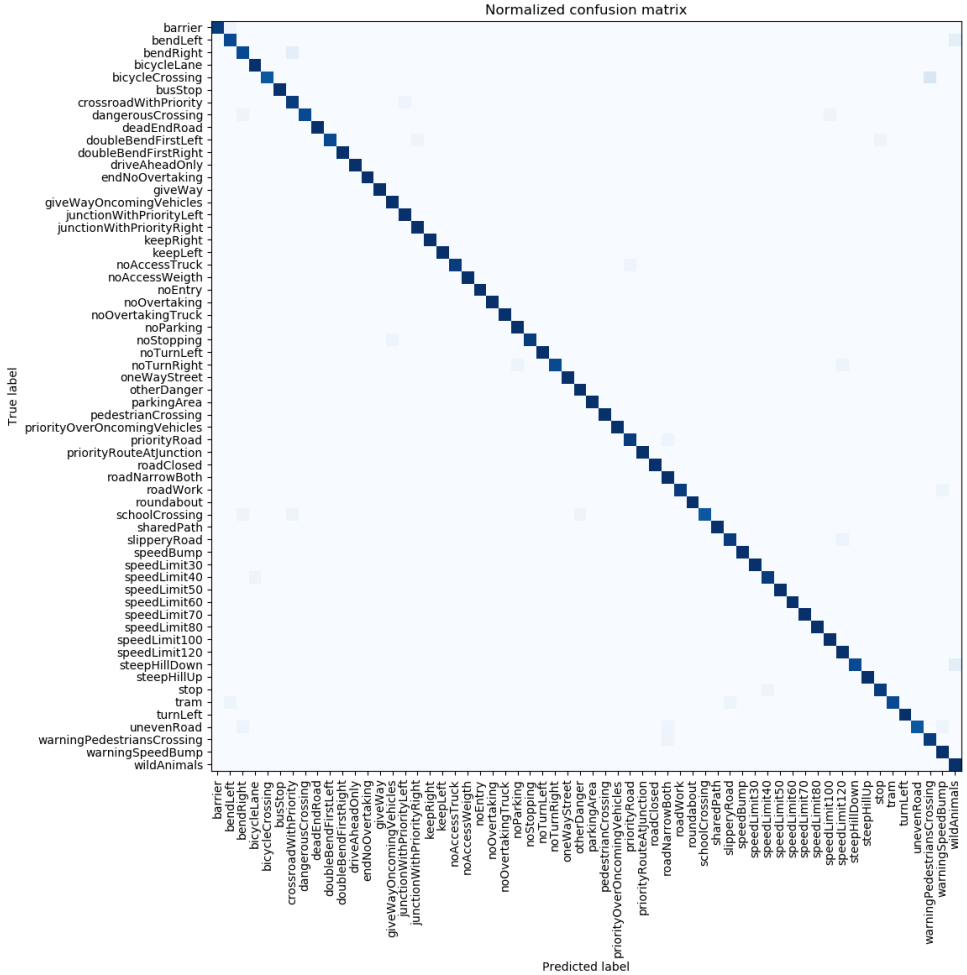
Appendix 6: Confusion matrix CNN for top 1 predictions on the blurred set (67% accuracy)



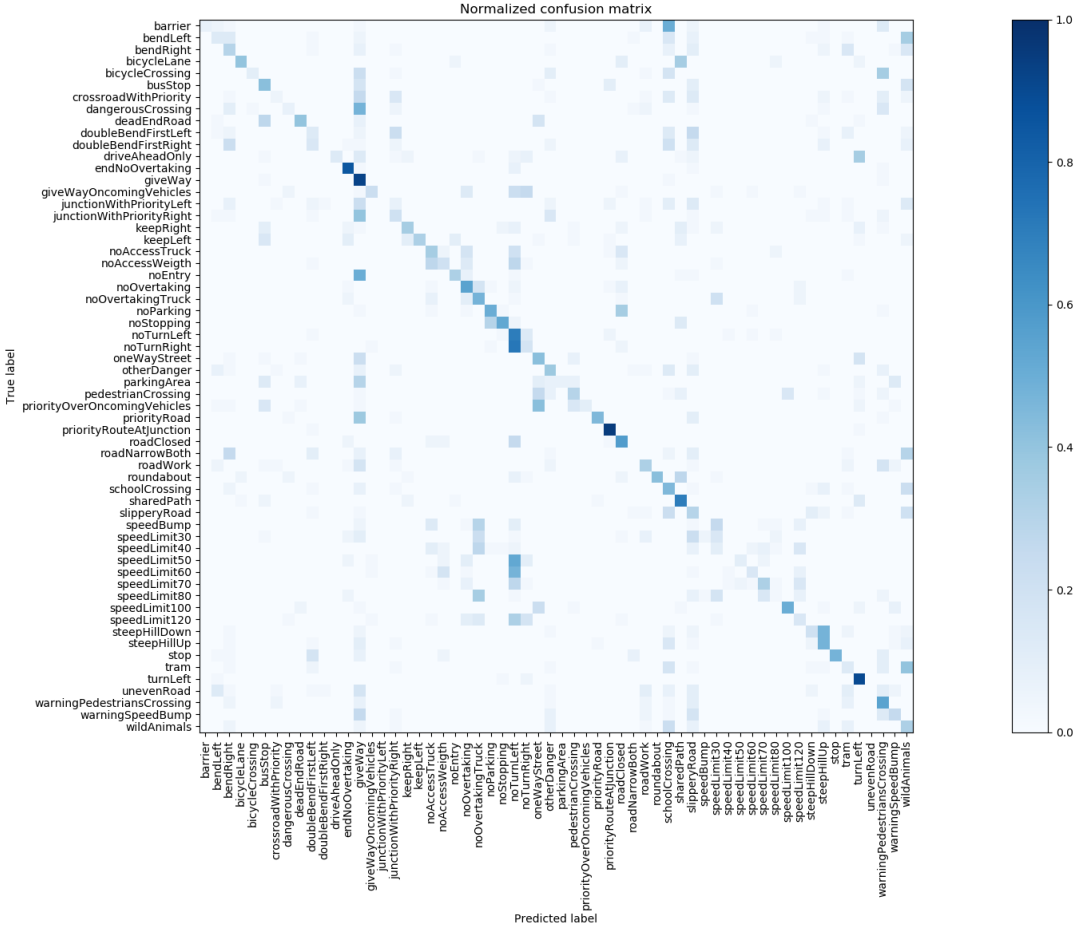
Appendix 7: Confusion matrix CNN for top 5 predictions on the blurred set (92.6% accuracy)



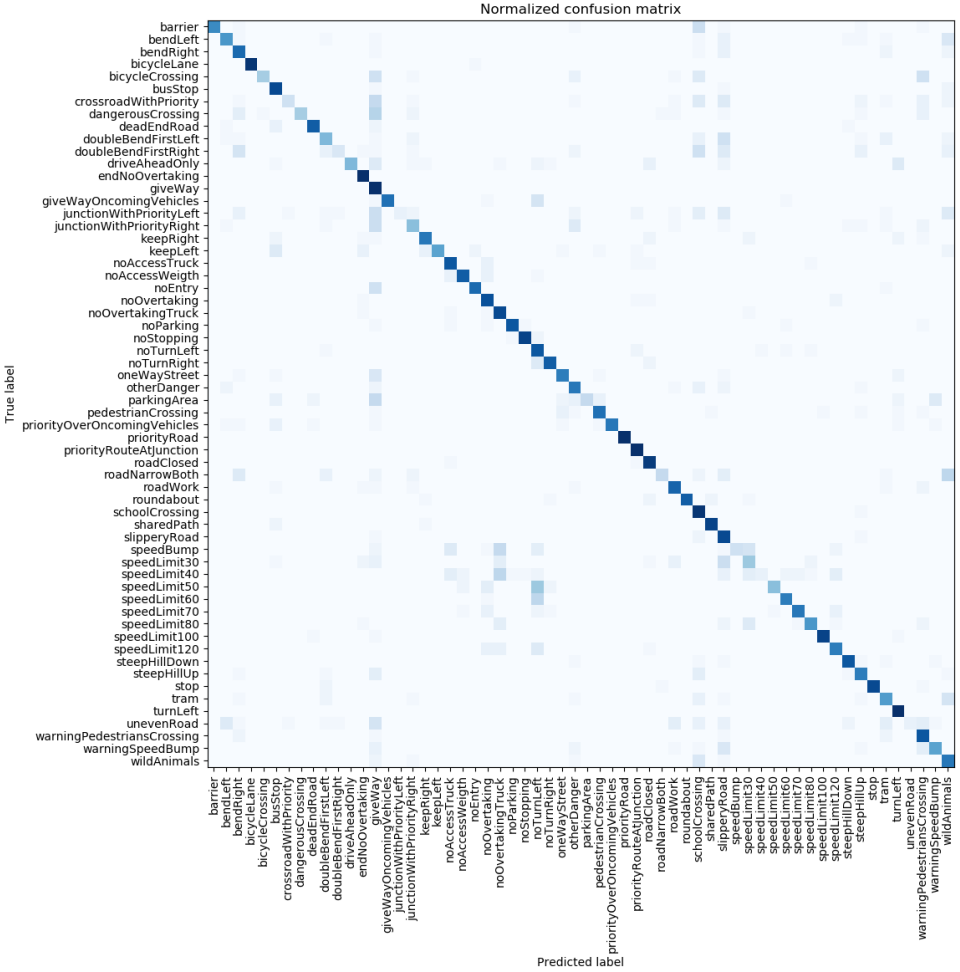
Appendix 8: Confusion matrix ELM for top 1 predictions on the blurred set (87.9% accuracy)



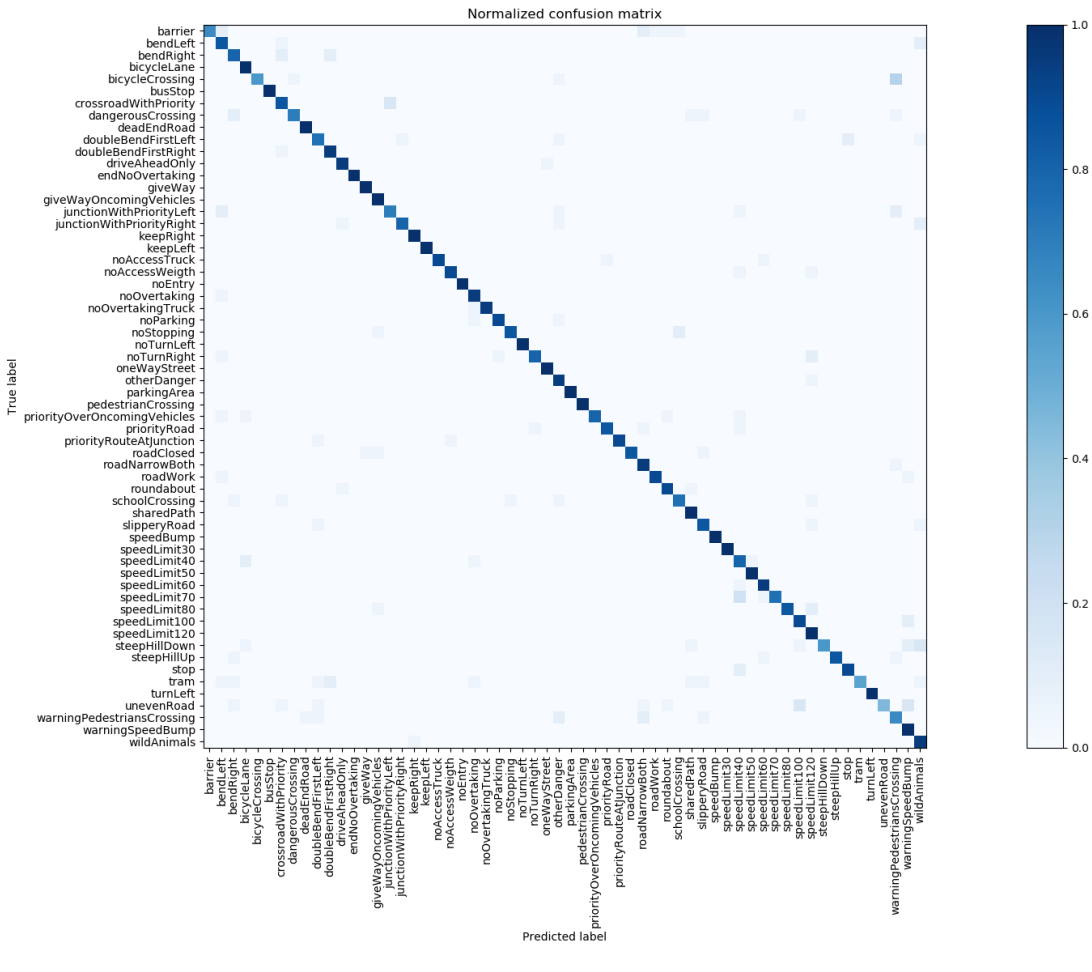
Appendix 9: Confusion matrix ELM for top 5 predictions blurred set (97.2% accuracy)



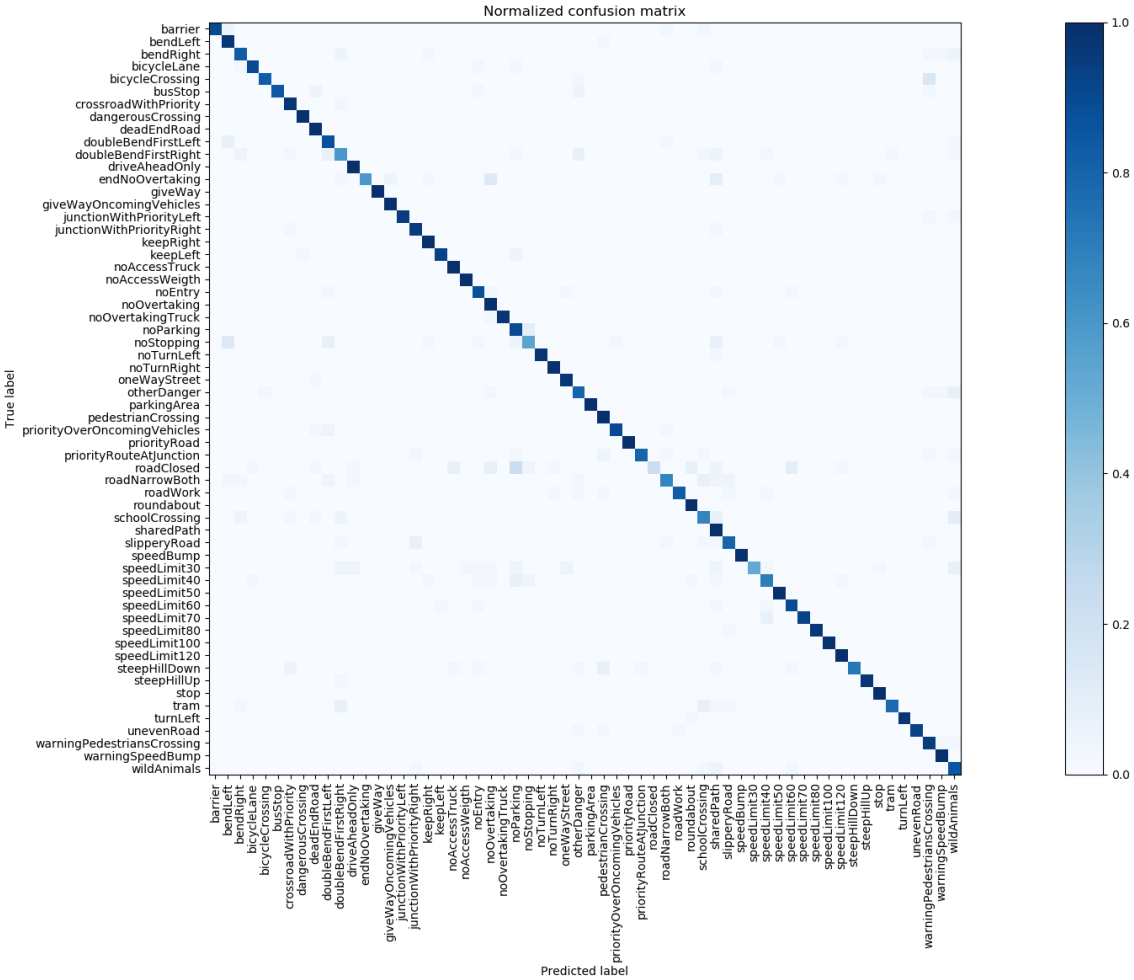
Appendix 10: Confusion matrix CNN for top 1 predictions black bar set (31.7% accuracy)



Appendix 9: Confusion matrix CNN for top 5 predictions black bar set (67.7% accuracy)



Appendix 10: Confusion matrix ELM for top 1 predictions black bar set (65.5% accuracy)



Appendix 11: Confusion matrix ELM for top 5 predictions black bar set (88.8% accuracy)