Improved Shallow Network for Fast Traffic Sign Recognition

Daniel Keitley
University of Bristol
Bristol, United Kingdom
dk14988@bristol.ac.uk

Ngoc Khanh Nguyen
University of Bristol
Bristol, United Kingdom
nn14160@bristol.ac.uk

***Abstract*—Zhang et al. (2017) present a model of traffic sign recognition which achieves 99.84% accuracy on the German Traffic Sign Recognition Benchmark (GTRSB) data-set. To our knowledge, this outperforms all previous traffic sign recognition models. In an attempt to reproduce the results of Zhang et al. we implement and extend the deep convolutional neural network model used. We obtain an accuracy of 90.8% on the GTSRB data-set and find improvements which reach 96.68%.**

I. Introduction

Driverless cars promise to revolutionise the way we travel [10], [12]. Not only do they remove human interaction from the driving experience but also offer a way of making driving safer. Autonomous systems are able to detect, analyse and respond to the environment far more quickly and consistently compared to human drivers. As a result, large technology companies and car manufacturers such as Uber and BMW are racing to develop driverless car technologies [8]. As part of this ambitious goal, firms are turning towards machine learning research to seek inspiration for learning systems which are able to understand events on the road. One particular problem for autonomous vehicles is to be able to detect traffic signs [3], which inform drivers about potential hazards, rules of the road and other contextual information.

The difficulty of recognising traffic signs is that they are often viewed from different perspectives and are situated in different surroundings and lighting conditions. Consequently, deep neural networks are deployed to recognise signs in spite of such variance. Zhang et al. [17] present a deep neural network model for traffic sign recognition that achieves a 99.84% accuracy on traffic sign images. This report summarises the approach taken by Zhang et al. and describes an attempt to replicate the results achieved.

II. Related Work

In combination with the development of driverless car technology, research in traffic sign recognition has been catalysed by the release of publicly available data-sets. The German Traffic Sign Recognition Benchmark [14] and Belgian traffic sign data-sets [15], have enabled researchers in computer vision to develop machine learning models and evaluate their performance on real world images. The German Traffic Sign Recognition Benchmark (GTSRB) for instance, was released as a machine learning competition, which awarded submissions that achieved the highest test accuracy.

The most successful of these models, submitted in 2011 by the IDSIA AI research lab in Switzerland [1], utilised a series of deep convolutional neural network (CNNs) arranged into columns. The authors produced CNNs with 8 layers intertwined with max pooling layers, that was able to achieve an accuracy of 99.46%, surpassing human level accuracy [14]. Convolutional neural networks are a standard choice for image classification. Inspired by the human visual system, they apply local filters to parts of an image so as to extract spatial information and relevant features. Other studies have also had success applying CNNs to traffic sign recognition [11], [6].

Other traffic sign recognition models have applied standard machine learning techniques on pre-computed image features, such as those derived from histograms of oriented gradients (HOGs). HOGs are calculated by counting the occurrences of image gradients, directional changes in image intensity, in localised regions of an image. They have proven to be effective features for a range of computer vision tasks such as pedestrian detection [2], since they are able to express shape information on a very fine scale, making them insensitive to geometric perturbations such as rotations, scaling or translations. In the context of traffic signs, where geometric patterns are crucial for recognition, HOGs provide a powerful tool for classification. HOGs have been used in combination with a wide range of models such as support vector machines (SVMs) [5], linear discriminant analysis (LDA) [14] and random forests [16].

In their paper, Zhang et al. use the former method of applying a deep convolutional neural network, to improve traffic sign recognition. The authors modify the typical CNN structure by combining different pooling operations to achieve state of the art performance.

III. Dataset

The data-set used to train our neural network model is the German Traffic Sign Recognition Benchmark (GTSRB) data-set, which contains 51,840 images of 43 different classes of traffic signs [14]. The GTSRB data-set was collected using a video camera mounted onto a driving vehicle. The traffic signs were then manually extracted and annotated.

The data was partitioned into splits of 75% and 25% for training and test data respectively. Each image has size 32x32 and pixels are encoded as RGB values. The labels for each training and test image are represented using one-hot encoding where only one component of a 43x1 vector has value 1, which represents the true class.

IV. Method by Zhang et. al.

The model developed by Zhang et al. can be summarised by outlining the 4 components below.

1) *Pre-Processing*: Zhang et al. first normalise the data-set, to encourage illumination invariance.
2) *CNN:* Pre-processed data is fed into a convolutional neural network consisting of 3 convolutional layers, followed by either average or max pooling layers, and 3 fully connected layers.
3) *Cost function:* Network performance is evaluated using a softmax-loss function which takes the softmax of the CNN output and computes the negative log likelihood.
4) *Optimisation:* The model parameters are optimised using a Nesterov momentum update rule, taken from [7].

V. Implementation Details

This section will describe how the components outlined above were implemented to match the Zhang et al. model.

A. Pre-Processing

In their paper, Zhang et al. present a pre-processing method whereby the mean value of each image channel across the entire data-set is subtracted from each pixel. This reduces the effect of illumination differences between images. The authors then apply a whitening step, a standard method of decorrelating pixel values to remove redundancy.

It was reported that implementing whitening reduces model performance and so in our replica model, standard normalisation has been applied, where the mean is subtracted from each pixel (as above) and divided by the standard deviation. The mean and standard deviation was calculated across the training set, while normalisation was applied to all images.

B. CNN

The CNN architecture was recreated from Table 1 in [17].

With each training step, a batch size of 100 pre-processed traffic sign images are fed into the network. The data is fed through the first convolutional layer, which applies a 5x5 filter, with stride 1, across the images. The output is zero-padded to produce 32, 32x32 feature maps. The output to the convolutional layer is then passed through a ReLu activation function which adds non-linearity to the network. The data is down-sampled using an average-pooling layer, which extracts the average responses within a 3x3 sliding window. The resulting data consists of 32, 16x16 feature maps. The data is passed through a second convolutional layer, ReLu activation function and average pooling layer with the same kernel sizes, strides and padding, to produce 32, 8x8 feature maps. Once again the network convolves the feature maps and passes the output through a ReLu activation. This time however, the data is down-sampled using a max-pooling layer. The max-pooling layer extracts the maximum response within a 3x3 sliding window. After this layer, 64, 4x4 feature maps are obtained. At this stage, the data is passed through the first fully connected layer. This layer could equally be implemented as a convolutional layer with a 4x4 kernel, of equal size to the 64, 4x4 feature maps. The result of the fully connected layer is once again passed through a ReLu activation function to produce 64, 1x1 feature maps. Next, the data is passed through another fully connected layer in which the input and output dimensions are equal thus maintaining the 64, 1x1 feature maps. The ReLu activation function is not applied at this stage. Finally, the data is passed through a final fully connected layer which downsizes the 64x1x1 input to a vector of length 43, representing each of the possible classes. The softmax function is applied which converts these output values to probabilities.

The network is initialised by sampling weights and biases from a $(-0.05, 0.05)$ uniform random distribution.

C. Cost function

In the final stage of the CNN pipeline, the output predictions are fed through a softmax function which enforces that each value of the output vector is between 0 and 1 and that all values sum to 1. As a consequence, each component i of the modified output vector can be interpreted as the probability that the input image belongs to class i. The softmax-loss function described in [17], takes the component j corresponding to the correct class, and computes the negative log-likelihood.

To implement this, the component corresponding to the true class label j, is obtained by multiplying the CNN output vector, v', element-wise with the true class label vector y. This produces a one-hot encoded vector v'' containing the network estimate of the probability of belonging to the true class j. To extract this value, the vector is summed. Now the negative log likelihood is computed and averaged across all values in the batch.

$$v = (v_1, \cdots, v_{43})$$

$$v' = softmax(v) = \left(\frac{e^{v_1}}{\sum_{i=1}^{43} v_i}, \cdots, \frac{e^{v_{43}}}{\sum_{i=1}^{43} v_i} \right)$$

$$v'' = \sum_{k=1}^{43} v' * y = \frac{e^{v_j}}{\sum_{i=1}^{43} v_i}$$

$$v''' = -\log \left(\frac{e^{v_j}}{\sum_{i=1}^{43} v_i} \right)$$

$$L_{SL} = \frac{1}{N} \sum_{k=1}^{N} v'''$$

Clearly the closer $softmax(v_j)$ is to 1, the closer the negative log, v''' is to 0. Thus to minimise the cost function, L_{SL} the network must learn to correctly estimate a high probability of the input image belonging to the correct class j.

While also learning to correctly identify input images, the network is also encouraged to maintain small parameter values so as to reduce overfitting. In Zhang et al. the authors incorporate this through a weight decay term in the update rule (see subsection V-D). In our implementation we achieve the same effect using L2 regularisation. All weights of the convolutional neural network are squared, summed and multiplied by a weight decay constant (or regularisation factor) with

value 0.0001. The regularisation term is subsequently added to the softmax-loss (L_{SL}). As a result of the L2 regularisation, higher penalties are given to larger parameter values.

$$L = L_{SL} + \frac{1}{2}\sum_{w \in W} \beta w^2$$

D. Optimisation

The update rule used in Zhang et al. is taken from the AlexNet model [7] which uses Nesterov momentum [9]. The Nesterov momentum update rule for weight parameter w evaluated at w_i, is given by:

$$v_{i+1} = \alpha v_i - \epsilon \left\langle \frac{\partial L}{\partial w}\bigg|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} = w_i + v_{i+1}$$

for momentum parameter α and learning rate ϵ where D_i is the current batch.

In our implementation the weight decay is incorporated into the cost function. Substituting the derivative with added L2 regularisation achieves the desired AlexNet update rule:

$$\left\langle \frac{\partial L}{\partial w}\bigg|_{w_i} \right\rangle = \left\langle \frac{\partial L_{SL}}{\partial w}\bigg|_{w_i} \right\rangle + \beta w_i$$

$$v_{i+1} = \alpha v_i - \epsilon\beta w_i - \epsilon \left\langle \frac{\partial L_{SL}}{\partial w}\bigg|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} = w_i + v_{i+1}$$

Parameter values for the momentum, learning rate and batch size are set to 0.9 and 0.01 and 100 respectively.

VI. Replicating Figures

In this section we reproduce Figures 3, 4 and 7 from [17] and analyse similarities and differences between our replica with Zhang et. al.

As can be seen from Figure 1, our implemented model maintains some similarities with the original paper. The normalisation performed by Krizhevsky et al.[7] as predicted in [17], performs worse. With the *local response normalisation* (LRN) applied, both the objective and error rate converges more slowly and tends to less optimal values. Here, the error rate is defined as the number of incorrectly classified samples, divided by the total number of samples.

One noticeable difference between models was the effect of whitening. In our implementation, which replaces the whitening from [17] with standard normalisation (referred to as whitening from here onward), whitening has a less drastic effect than was reported (Figure 3). This could be due to the differences in implementation, although it should be noted that whitening made a considerable improvement during early stages of training. After 10 epochs of training, however, the differences are less clear.

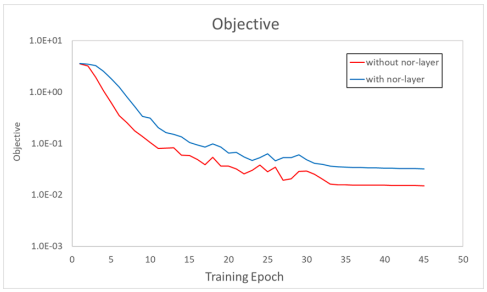

Fig. 1. Softmax-loss values recorded over the training period with and without the normalisation applied in [7]. Loss values are averaged across batches in each epoch and are presented on a logarithmic scale.

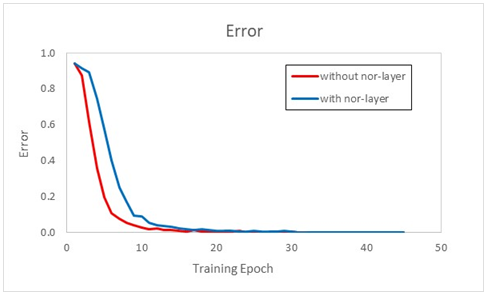

Fig. 2. The model error with and without the normalisation method presented in [7].

Fig. 3. The model training error with and without whitening over the training epochs.

In Figures 4 and 5 we present the learned filters of the first and second convolutional layers respectively. The 5x5 kernels of the first convolutional layer appear to extract fundamental image properties, such as the red banner and inner fill of typical traffic signs. Other kernels in the first and second convolutional layers are more abstract.

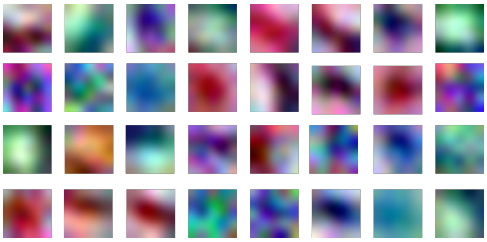

Fig. 4. Learned filters of the first convolutional layer.

On the other hand, the kernels start to represent the input images much more abstractly in the second convolutional layer (see Fig. 5). In particular, colour information is lost at this stage

since colorful kernels from Fig. 4 convert input images into grey mappings.

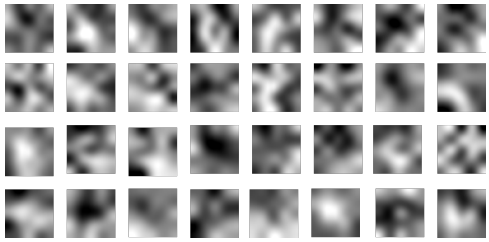

Fig. 5. Learned filters of the second convolutional layer.

VII. Replicating Quantitative Results

To replicate the results of Zhang et al. our Tensorflow implementation was trained and tested using the BlueCrystal high performance computer at the University of Bristol [18]. The model achieved an accuracy of 90.82% (see Fig. 6), a result significantly less than the 99.84% achieved in the original model [17]. Nevertheless, the result is comparable with other replication attempts conducted by researchers at the University of Bristol.

Model	Accuracy
Zhang et. al. [17]	99.84
Keitley & Nguyen	90.82

Fig. 6. Comparison of overall accuracy of our model with Zhang et. al.

Despite differences in performance, both models perform similarly (relative to overall accuracy) on the same traffic sign classes (see Fig. 8). For instance, the highest accuracy in both models is achieved on *other prohibition* signs, while also performing poorly on *derestrictions* signs.

Model	SPD	PHB	DTR	MDT	DGR	UQE
Zhang et. al. [17]	99.54	100	98.33	99.94	98.96	99.95
Keitley & Nguyen	89.68	96.51	82.22	94.67	79.19	95.83

Fig. 7. Comparison of the accuracies obtained by Zhang et. al. and our replication for the following subsets of traffic signs. Here, SPD - speed limits, PHB - other prohibitions, DTR - derestrictions, MDT - mandatory, DGR - danger and UQE - unique.

As mentioned above, our model implementation was trained using one GPU with Tesla P100 on the BlueCrystal high performance computer [18]. The training and recognition times are 0.03h/dataset and 0.02ms/frame respectively. Differences in values between models is expected as the authors in the original paper used CPUs, as opposed to our GPU configuration.

Model	Training Time	Recognition Time	Configuration
Zhang et. al. [17]	0.9h/dataset	0.64ms/frame	CPU: 8 × I7-6700K
Keitley & Nguyen	0.03h/dataset	0.02ms/frame	GPU: 1 × Tesla P100

Fig. 8. Comparison of training time and recognition time.

VIII. Discussion

The figures in the previous section demonstrate that our implemented model does not replicate the state of the art performance obtained by Zhang et. al. Concretely, our model achieves an accuracy 8% lower than reported. Moreover, our model without pre-processing applied, obtained significantly higher results than was stated in [17]. Despite these differences, our model agrees that the LRN method does not improve accuracy as significantly as whitening.

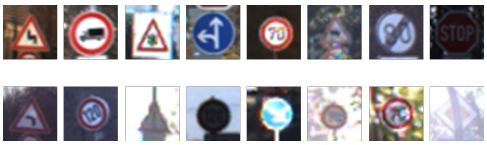

Fig. 9. Samples of images correctly (above) and incorrectly (below) classified by the model.

A sample of images correctly and incorrectly classified by our implementation can be seen in Figure 9. Incorrectly classified images tend to have particularly high or low brightness, which could indicate a weakness in the model. However, as demonstrated in the next section, brightness augmentation does not improve accuracy.

While not all correct classifications are shown, it is clear that the network has learned to correctly identify traffic signs even in cases with occlusion, different vantage points and lighting conditions.

IX. Improvements

As can be seen in Figure 3, the model learns very quickly and then maintains a loss of $0.01 - 0.02$ after the first 10 epochs. During this large period of consistent loss, it is possible that the model is trying to overfit the training data.

In this section we present two significant improvements to the original model which contributed increase of accuracy. These are are (i) *data augmentation*, and (ii) *dropout*.

Model	Accuracy
Whitening Only	90.82
Whitening + DAUG	94.06
Whitening + DROP	95.31
Whitening + DROP + DAUG	96.68

Fig. 10. Comparison of our proposed improvements with the original model including whitening. Here, DAUG - data augmentation, DROP - dropout.

A. Data Augmentation

Data augmentation provides a way to expand the training set by slightly modifying training images. Classical examples of this technique include adding random brightness/saturation or applying linear transformations to an image. In this work, we use the augmentation technique similar to the one by Jin et. al. [6] and apply the following: (i) we first choose a number k uniformly at random from the interval [0.9,1.1] and resize by a factor of k, (ii) we then take a random vector $\mathbf{v} = (a, b)$, where $a, b \in \{-2, -1, 0, 1, 2\}$ and translate the image by $\mathbf{v}$ pixels. (iii) Finally we rotate images by n degrees for n chosen uniformly at random from $[-5, 5]$. In order to keep the original size of images, we zero pad where necessary. Examples of such modifications are shown in Fig. 11.

Implementing data augmentation on our model improves the accuracy by about 3% (see Fig. 10). We believe that

Fig. 11. Augmented training data. Here, each image is rotate by a small degree between -5° and 0°.

applying these types of image transformations make our neural network learn better. Indeed, we have checked that adding random brightness, contrast or saturation returns an accuracy of around 92%.

B. Dropout

Dropout is a state-of-the-art technique introduced by Srivastava et. al. [13] which forces each neuron in some layers of our choice to 'turn off' i.e. output 0 with probability p. This method prevents from overfitting and is particularly effective when used in fully-connected layers. We implement such dropout for the first two fully-connected layers of our deep neural network with $p = 0.5$. There is a discussion in deep learning if applying the dropout in late convolutional layers would also increase the overall accuracy (e.g. [13], [4]). We tried this method on our third convolutional layer with probabilities $p \in \{0.1, 0.2\}$ and obtained around 90-91% accuracy, which is comparable to not using dropout at all.

After applying the technique by Srivastava et. al. we obtain an accuracy of 95.31% (see Fig. 11). Combined with data augmentation, we obtain a final result of 96.68%. This shows that implementing both data augmentation and dropout improves our model and prevents overfitting.

X. Conclusion and Future Work

The traffic sign recognition model implemented in this report achieves 96.68% accuracy on the German Traffic Sign Recognition Benchmark data-set, comparable to other top performing models. However, future work has the potential to improve our results further. Firstly, we see that the data augmentation technique increased the accuracy only by around 2%. One suggestion would be to find a better method to augment the training data which combined with dropout could achieve more than 98%. Also, we have not considered an effective way to detect whether our network is stuck in local minima. We have tried decaying the learning rate in a similar manner to AlexNet and use Adam optimisation, however worse results are obtained than applying data augmentation. Changing the weight decay from 0.0001 to 0.0005 (like in AlexNet) or even 0.001 also did not improve the model.

Furthermore, previous methods of traffic sign recognition have typically applied deep convolutional networks in conjunction with HOG features [1], [11]. These features directly provide the network with shape information, which may provide a better source of information for convolutional layers.

Acknowledgment

The authors would like to thank Dr Dima Aldamen, and teaching assistants: Vangelis Kazakos and Will Price for useful suggestions and help with debugging the code.

References

[1] CireAn, Dan, et al. "Multi-column deep neural network for traffic sign classification." Neural Networks 32 (2012): 333-338.

[2] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[3] Fu, Meng-Yin, and Yuan-Shui Huang. "A survey of traffic sign recognition." Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on. IEEE, 2010.

[4] Gal, Yarin, and Zoubin Ghahramani. "Bayesian convolutional neural networks with Bernoulli approximate variational inference." arXiv preprint arXiv:1506.02158 (2015).

[5] Greenhalgh, Jack, and Majid Mirmehdi. "Real-time detection and recognition of road traffic signs." IEEE Transactions on Intelligent Transportation Systems 13.4 (2012): 1498-1506.

[6] Jin, Junqi, Kun Fu, and Changshui Zhang. "Traffic sign recognition with hinge loss trained convolutional neural networks." IEEE Transactions on Intelligent Transportation Systems 15.5 (2014): 1991-2000.

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[8] Mercer, Christina. "Which companies are making driverless cars?" Tech World (2018). Retrieved from https://www.bloomberg.com/news/articles/2017-11-20/uber-steps-up-driverless-cars-push-with-deal-for-24-000-volvos

[9] Nesterov, Yurii. "Gradient methods for minimizing composite objective function." (2007).

[10] Rodel, Christina, et al. "Towards autonomous cars: the effect of autonomy levels on acceptance and user experience." Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. ACM, 2014.

[11] Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[12] Shanker, Ravi, et al. "Autonomous cars: Self-driving the new auto industry paradigm." Morgan Stanley Blue Paper, November (2013).

[13] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research 15.1 (2014): 1929-1958.

[14] Stallkamp, Johannes, et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." Neural networks 32 (2012): 323-332.

[15] Timofte, Radu, Karel Zimmermann, and Luc Van Gool. "Multi-view traffic sign detection, recognition, and 3d localisation." Machine vision and applications 25.3 (2014): 633-647.

[16] Zaklouta, Fatin, Bogdan Stanciulescu, and Omar Hamdoun. "Traffic sign classification using kd trees and random forests." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[17] Zhang, Jianming, et al. "A Shallow Network with Combined Pooling for Fast Traffic Sign Recognition." Information 8.2 (2017): 45.

[18] Advanced Computing Research Centre, University of Bristol. https://www.acrc.bris.ac.uk/