Traffic Sign Recognition with Deep Learning

Gavin Parker
gp14958@bristol.ac.uk

John Griffith
jg14857@bristol.ac.uk

Ross Gardiner
rg14820@bristol.ac.uk

***Abstract*—Traffic Sign Recognition is a difficult task with significant importance in automotive research. We present a shallow network for traffic sign recognition that uses comprehensive deep learning techniques to achieve an accuracy of 97.24% on *The German Traffic Sign Recognition Benchmark*[1] test.**

I. Introduction

Traffic Sign Recognition (TSR) is vital to the success of autonomous vehicles, enabling them to recognise and adapt to local traffic laws and conditions without a map of preprogrammed traffic zones. TSR also has applications in driver assistance and GPS software. Drivers can be reminded of driving conditions, and detailed maps can be more quickly generated from collected data. As traffic signs are a format designed to be recognisable by humans, the recognition task is treated as an image classification problem. The difficulty of the task is exacerbated by changes in lighting and background colour. Furthermore any recognition solution must be invariant to motion blur and different projections as traffic sign images are often taken from moving cameras. It is difficult to use traditional image processing and machine learning techniques to create a general solution that handles these variations. Recent progress in Convolutional Neural Networks (CNNs) has been of great benefit to TSR, demonstrating that fast and generalised image classification is possible. In this paper we replicate a CNN designed by Zhang et al.[2], and suggest some possible improvements to the network.

II. Related Work

There is a range of previous work on TSR that utilises CNNs. Some approaches achieve accuracy above 98%. Sermanet and LeCun propose a multi-scale feature–based approach[3] using the same GTSRB dataset (with augmentations). Multi-scale features extend traditional feed-forward CNNs by forwarding the output of multiple layers to the final classification layer. This results in a classifier that depends on features extracted both earlier and later in the neural network, corresponding to high-detail features and low-detail representations. This can help make the result more invariant to image conditions.

TSR is primarily an image classification task, so it follows that many approaches are inspired by image classifier networks from different contexts, such as AlexNet[4]. AlexNet is a CNN based approach to image classification trained on the ImageNet dataset[5], a vast collection of noun-labelled images. AlexNet takes a similar approach to Zhang et al., featuring five convolutional layers; max-pooling and the ReLU activation function are used to model neuron nonlinearity.

Jin et al. describe a hinge loss stochastic gradient descent method applied to TSR[6]. Their shallow CNN achieves an excellent result on the GTSRB dataset of 99.65% accuracy. Hinge loss is a cost function often used to train classifiers with large margins; its definition discards correct classifications by only learning from positive costs. This implementation of hinge loss for CNN architectures revises cross-entropy by introducing a margin threshold for correctly classified data. Data within the margin is discarded, allowing the back-propagation to focus on incorrectly classified inputs. Results show that this led to faster training time and increased accuracy.

III. Dataset

For training we use *The German Traffic Sign Recognition Benchmark*[1], which features 39,209 images categorised into 43 different types of traffic sign. There is also a test set of 12,630 images. The images have varying size, and are supplied with bounding boxes around each traffic sign such that the images can be appropriately cropped to the same size, an approach adopted by Zhang et al. The RGB colour images are stored in the uncompressed PPM format so that no image data is lost to compression. Each image is labelled with a number corresponding to its class, as shown in Figure 3. It is important to note that the number of images provided for each class varies significantly, though this distribution is somewhat reflected in the provided test set. For a generalised solution it would be useful to redistribute the class counts to get a more balanced and accurate representation of the real-world sign class distribution.

IV. Method

TABLE I
Comparison of Training and Recognition Time[2]

Method	Training Time	Recognition Time	Configuration
Our method	0.17h /dataset	0.05 ms/frame	Tesla P100
Zhang et al. replication	0.02h /dataset	0.03 ms/frame	Tesla P100
Zhang et al.	0.9h /dataset	0.64 ms/frame	8 × I7-6700K

The network proposed by Zhang et al. uses a series of three stages to reduce the data to its essential features. Each comprises a convolutional layer (5x5 kernel) and subsampling layer (max- or average-pooling). These are followed by a convolutional layer and a fully-connected layer, where the convolutional layer behaves as a fully-connected layer. The fully connected layers map the convolutions onto single pixels. Finally, classifications are obtained using a softmax

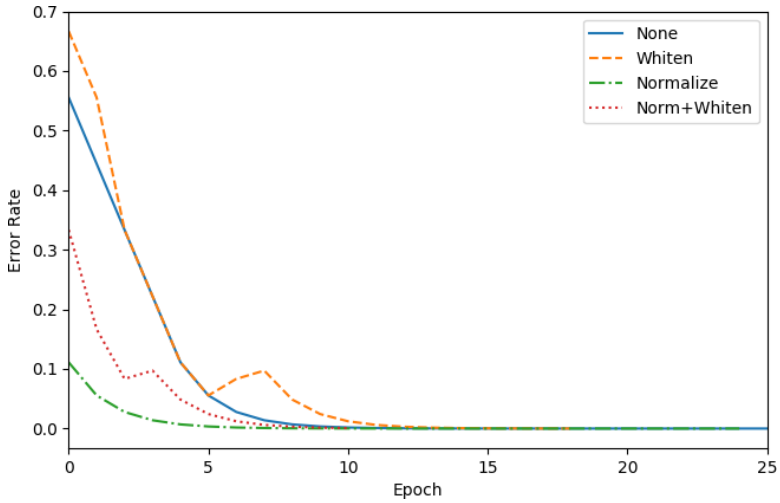

Fig. 1. The effect of different whitening methods. Lines terminate before final epoch due to 'early stopping' technique.

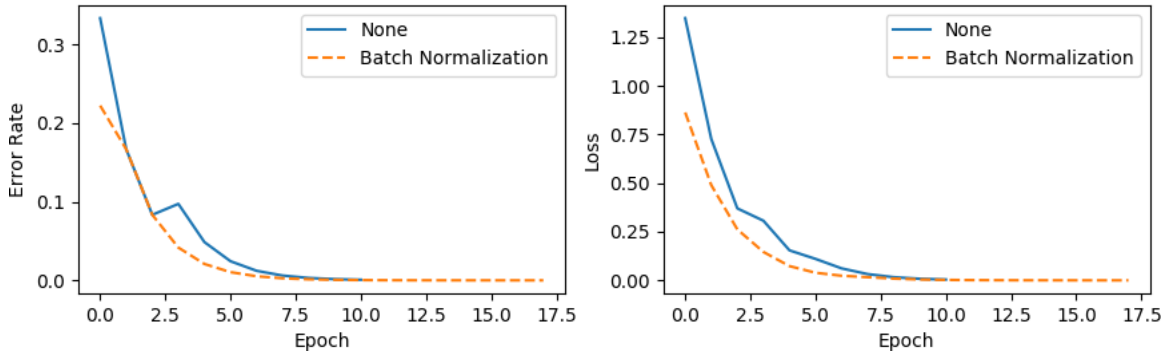

Fig. 2. The effect of batch normalisation.

TABLE II
COMPARISON OF ACCURACY PER SIGN CATEGORY[2]

	Speed Limits	Other Prohibitions	Derestriction	Mandatory	Danger	Unique
Zhang et al. original	99.30	99.80	99.44	100	99.13	99.90
Zhang et al. replication	83.28	96.86	87.81	91.59	77.28	95.52
Our Method	95.75	98.40	97.50	99.26	95.32	99.14

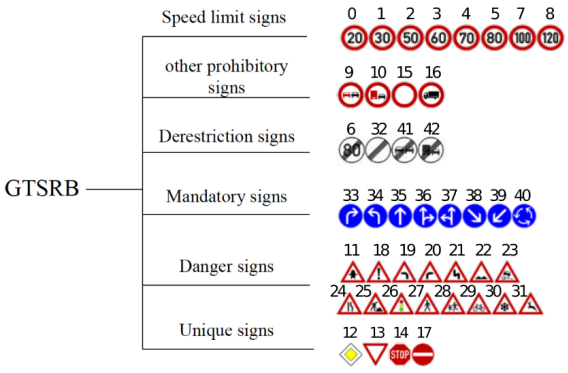

Fig. 3. GTSRB class labels (from [2]).

layer. The network uses ReLU activation functions after each convolutional layer. Before training, Zhang et al. augment their data with small random rotations and translations, and whiten the images. They also experiment with the use of batch normalization but decide not to use it in their network.

V. IMPLEMENTATION DETAILS

Zhang et al. clearly present their network architecture and most of their hyperparameters. We reconstructed the convolution and fully connected layers, replicating the padding, stride length, and pooling methods. The weights and biases for each of the layers use uniform random initialisation between -0.05 and 0.05. Weight decay of 0.0001 was also used for each of the layers. AlexNet is referenced with respect to learning rate, which is 0.01.

Zhang et al. suggest that whitening should lead a much faster reduction in the network's error rate. Because of the

lack of detail in the paper, we were unsure how exactly they whitened images. We experimented by whitening on a per image basis, across the whole dataset and a combination of the two.

The network is trained in batches of 100 images, using the Momentum optimiser (with momentum 0.09) to update the weights. One entire run through the training data is an 'epoch', after each of which we calculate the validation accuracy and loss for the test set. We use TensorFlow's summary operations to record various data at each epoch.

VI. Replicating Figures

After replicating the architecture described by Zhang et al., we reconstructed various figures from the original paper. Figure 1 shows the result of different whitening methods. Normalisation and whitening increase training performance, but all techniques eventually converge on the same error rate.

We experimented with using batch normalisation to replicate the analysis in the original paper. Our results are shown in Figure 2. We found that batch normalisation had minimal impact on our results, though faster convergence does reduce training time.

Figure 7 shows the convolution kernels extracted from our trained replica. The second layer (right), is calculated from a reduced average of the 32 kernel channels. The images differ significantly from the corresponding figures provided by Zhang et al. This suggests that our replica has learned different features during training.

VII. Replicating Quantitative Results

As shown in Table III, we were unable to reach the accuracy claimed by Zhang et al. Our replica achieved 90.08%, not 99.84%.

We saw significantly faster training times than Zhang et al. by using a powerful GPGPU (NVIDIA Tesla P100) for training. Matrix multiplications, a large fraction of calculations in the feed-forward step, can be performed very fast on a GPU, which suits the massively parallel graph processing required for training CNNs. Zhang et al. state that their network was trained on an Intel Core i7-6700K CPU. We train for 50 epochs, and our training time of 83 seconds is an order of magnitude faster than over 3200 seconds they report.

We also found that the relatively shallow network architecture and low image resolution (thus low memory requirement) make it possible to train the network on a consumer-level GPGPU (NVIDIA GTX 1070), rather than a workstation- or datacentre-class device.

In Table II we show that our replica achieved significantly lower accuracy for almost all traffic sign categories. However, danger signs and speed limits exhibit particularly poor performance, suggesting that we should target these for improvement.

VIII. Discussion

We were unable to fully replicate the final results of the architecture described by Zhang et al. Our replica achieved 9.76% lower accuracy.

TABLE III
Comparison of overall accuracy

Methods	Accuracy
Improved Results	**97.24**
Zhang et al. Replication	90.08
Zhang et al.	99.84

Fig. 4. Examples of correctly classified testing images.

Fig. 5. Examples of incorrectly classified testing images.

The lack of exact detail around whitening forced us to experiment with a range of implementations. We only saw a slight improvement from this, as shown in Figure 1, instead of the expected steep, distinct reduction in loss.

Figure 5 shows some incorrectly classified images. In the bottom left image, the triangular sign may have interfered with classification. This suggests the features for classifying mandatory signs may not be as developed as those for danger signs, possibly a result of the imbalance in the training data: there are $1.6\times$ as many danger signs as mandatory signs. This could be fixed by using data augmentation to balance the training set.

It is possible that the top left training image has been misclassified due to the background colour, which is very similar to the edges of the sign, making edge detection more difficult. This might be mitigated by using a different colour

space—for example, HSV might highlight subtle differences in hue.

The bottom right image has low contrast, making its symbols hard to distinguish from the sign background. This might be improved by training on low-contrast images or by enhancing the contrast of input to the network. The top right image exhibits rotation, blur, desaturation and occlusion. Classification could be improved by adding similar images to our augmented data.

IX. Improvements

A. Concatenated ReLU

The activation function of a layer can have a significant impact on the performance of the entire network. In biological neurons the activation function is a complicated non-linear behaviour arising from electrical properties of the cell. This is hard to efficiently emulate in software. Zhang et al. use a popular compute-friendly activation function, the 'Rectified Linear Unit' (ReLU). We achieved better results by using a 'Concatenated Rectified Linear Unit'[7] (CReLU), which aims to extract positive and negative phase information from kernels in convolutional layers. While ReLU extracts positive output using

$$max(0, x)$$

CReLU also extracts the negative output using

$$concat(max(0, x), max(0, -x))$$

resulting in a larger output. In image classification tasks, many of the learned filters will be paired with similar filters of opposite phase. CReLU is able to capture the paired information and therefore avoid redundant filters in the convolutional layer.

B. Multi-scale Features

We implemented multi-scale features (shown in Figure 6) a common CNN enhancement which has previously been applied to TSR by Sermanet and LeCun[3]. This technique aims to capture features at both a high and low level of detail by combining the output from several sequential convolutional layers. The combined output is used as the input for the final layers of the network. Proportional max-pooling ensures that each input stream to the final layers is of the same size. This avoids bias towards the low-level features from earlier in the network. This change resulted in a 3.68% improvement in accuracy with minimal performance cost.

C. Minor Improvements

We made a number of minor additional tweaks. Before training, we used imgaug[8] to augment our training data with random rotations, translations, projections and obstructions, as shown in Figure 8. We also flip classes that are invariant to horizontal flips. These augmentations resulted in an extended training set approximately twice the size of the original, and a 4.87% increase in accuracy. In theory, the more varied data should produce a more general model.

To speed up the training and experimenting process we implemented early stopping. We save the trained network at each epoch; if training fails to improve accuracy for a number of epochs we finish early with the best-known network. In addition to eliminating unnecessary training time, this technique can revert potential overfitting.

Finally, we use drop-out, randomly dropping some connections to avoid overfitting in our fully-connected layer.

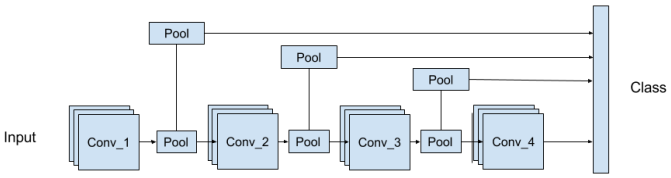

Fig. 6. Our implementation of multi-scale features.

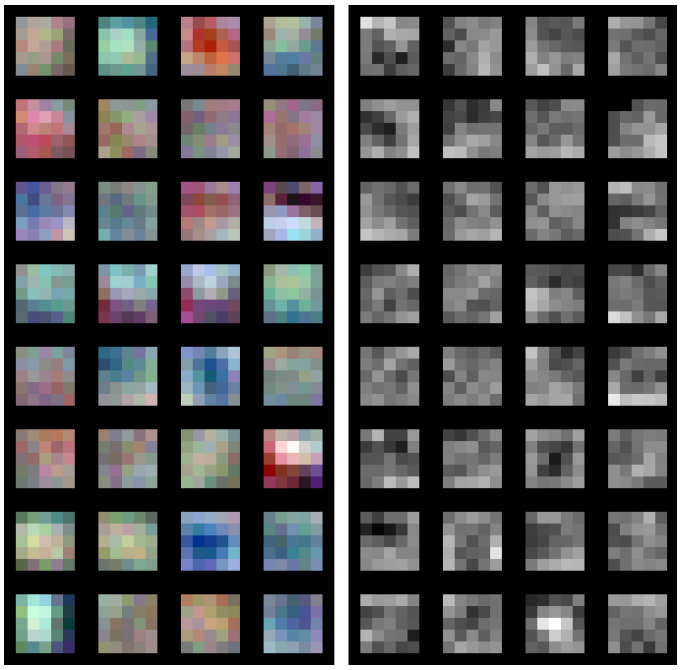

Fig. 7. Kernels learned by convolution layer 1 (left) and layer 2 (right)

X. Conclusion and Future Work

While we achieved good results, we were unable—with the information available—to perfectly reproduce those of original paper. We believe that Zhang et al. may have omitted some key information essential to achieving their claimed accuracy.

Though our test set accuracy is lower, we believe that our improvements should result in a more general model suited to real-world applications. We have attempted to prevent overfitting, and our augmentation helps our data more closely resemble a real-world dataset.

Further work might improve our network's accuracy and training speed. Feature pyramids might improve accuracy, permitting training features at different scales. This makes better use of variably-sized data. Our data pipeline resizes all images to 32x32, resulting in loss of important detail. To learn at multiple scales we could replace our first layer

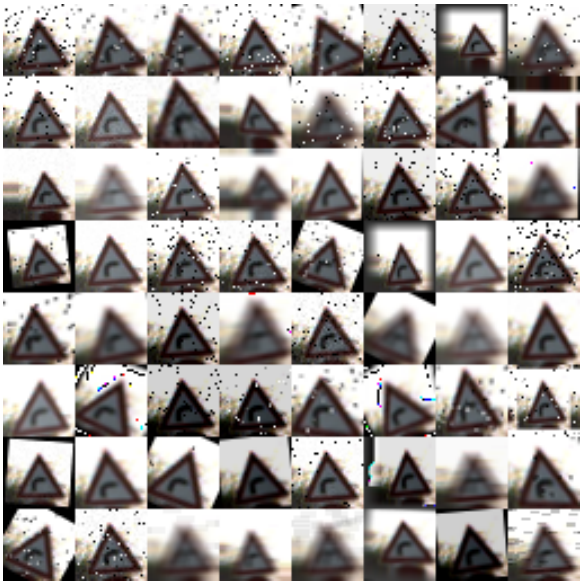

Fig. 8. Examples of augmented images obtained from a single training example.

with an array of convolutional layers, each trained on different image sizes. However, while this may enable learning of more detailed features, it multiplies both the compute and memory requirements of training.

Advances in GPGPU technology have allowed for more exhaustive training and deeper network architectures such as GoogLeNet [9]. By introducing more layers we would be able to learn a more complicated function that captures more abstract, non-linear features of the data. However these networks are slower and more computationally expensive to train and infer from, making them potentially unsuitable for real TSR applications. Furthermore it is unlikely that TSR needs the higher-order features that deep networks make use of for more complicated classification tasks like ImageNet.

We could also further augment our training data. Previous research suggests that the YUV colour space can decrease training time for TSR[3]. Most relevant data is captured in the Y channel, allowing other channels to be reduced in resolution or even eliminated, freeing up GPU memory. Similarly, we could exploit sign colours to reduce training times, for example by enhancing the red channel of the image[10]. To increase generality, we could utilise more 'real' training data in addition to our augmentations. Rather than collecting samples manually, it is common to use simulated data [11]. We could use any of various commercial simulators and games that display traffic signs under different conditions. It is important, however, to ensure that simulated data is of high-enough quality and accurately reflects real world data.

References

[1] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.02.016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457.

[2] J. Zhang, Q. Huang, H. Wu, and Y. Liu, "A Shallow Network with Combined Pooling for Fast Traffic Sign Recognition," *Information*, vol. 8, no. 2, p. 45, 2017.

[3] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2011, pp. 2809–2813.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[6] 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, Oct. 2014, ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2308281.

[7] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units," 2016.

[8] A. Jung, *Imgaug*, version 776d03dc8faa63a037ae1d559ded371ddd33b0b2. [Online]. Available: https://github.com/aleju/imgaug.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CoRR*, vol. abs/1409.4842, 2014. arXiv: 1409.4842. [Online]. Available: http://arxiv.org/abs/1409.4842.

[10] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 16–24, 2014, New Boundaries of Robotics, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2012.07.019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889012001236.

[11] OpenAI. (2016). Universe, [Online]. Available: https://blog.openai.com/universe/.