Re-producing A Shallow Network with Combined Pooling for Fast Traffic Sign Recognition

COMSM0018: Applied Deep Learning (2017-18)

Mihail Calin Ionescu
Computer Science Department
University of Bristol
Bristol, UK
mi14828@my.bristol.ac.uk

Mudit Gupta
Computer Science Department
University of Bristol
Bristol, UK
mg14777@my.bristol.ac.uk

I. Introduction

The original paper tries to solve the issue that has arisen from the increasing popularity of the new autonomous driving and driver assistance systems. At the core of these systems are mechanisms based on computer vision and neural networks for Traffic Sign Recognition (TSR) that are used alongside other systems and sensors, sometimes as redundancies, to provide extra information used for navigation and safety measures. When it comes to unmanned driving, even with high accuracy, TSR is usually used to support other techniques like lane graphing and path tracking [1], but it is good enough on its own when providing assistance to a user. Different algorithms have been used in order to solve the problem of correctly identifying traffic signs and, at the moment, the deep learning approach seems to be outperforming the other methods. Deep learning models that accurately identify the traffic signs already existed, but that is not enough if it takes a long time to classify a sign. The paper that we are attempting to re-produce tries to come up with a solution that not only recognises signs with a high accuracy, but it is also quick.

II. Related Work

Previous attempts of using Deep Learning to solve the TSR problem efficiently focused on using histogram of oriented gradient (HOG) features for object detection. The HOG is a feature descriptor used to describe an image by extracting the useful information, it decomposes an image into square cells of a given size and computes a histogram of oriented gradient in each of the cells. After the detection of the sign, an ELM (extreme learning machine, a feedforward neural network where the parameters of the hidden nodes do not have to be tuned) algorithm was used. After some tweaks that reduced the computational time, this approach managed to get up to 98.62% accuracy[2].
Other methods that ended up performing well were based on Convolutional Neural Networks or Auto Encoding Networks that took the raw pixels as input. The problem with these approaches is that the training step requires a lot of computation power and the sign recognition is relatively slow. Also in one of the proposed solutions, in order to get to 99% accuracy, both a CNN and a MLP(Multi Layer Perceptron) were used to decide on the classification of a traffic sign.

Solutions using CNNs on raw pixels, while providing high accuracy, are not robust in the case of rotated/scaled images unless images that cover these scenarios are used in the training phase which makes them highly dependent on the amount of available data.

III. Dataset

The dataset used is the one from the original paper, the German TSR benchmark (GTSRB) with some slight modifications. The set consists of 39.209 training images and 12.630 test images that are meant to be as lifelike as possible and cover a large range of conditions for each sign. The data is split into 43 classes. In the original dataset, the images range anywhere from 15 15 to 250 250 pixels in size and most of them are square, but not all. There is no guarantee that the traffic sign is centered, but there is always border of at least 5 pixels around the sign. The set that has been provided has the images rescaled to 32 x 32 pixels, while still keeping their margins. This was achieved using the same bicubic interpolation method mentioned in the original paper.

IV. Method (Zhang et al)

The original paper proposes a CNN architecture with 10 layers: one input layer, 6 convolutional layers split into three stages each containing two consecutive layers, two fully connected layers and a final softmax-loss layer. Each stage consists of a convolutional layer and a subsampling layer, where the subsampling can be either max pooling or average pooling. To identify which subsamping methods should be used they ran a set of experiments with different combinations of the two previously mentioned methods and found that average-average-max pooling converges the fastest and also achieves the lowest error rate out of the eight

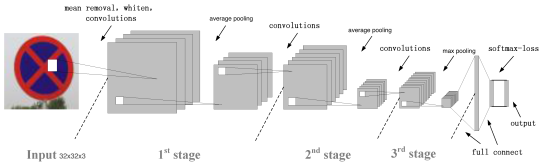

Fig. 1. The original architecture

possible combinations.
ReLU was chosen as the activation function for all of the convolutional layers as it was experimentally discovered that it significantly reduces the computation time while training the network.
In the paper they mention applying local response normalization according to the ImageNet[4] paper, but the results were not favorable. Applying whitening helped them converge faster obtain slightly better accuracy.

The network was trained for a duration of 45 epochs using the following parameters: a batch size of 100 images, momentum of 0.9 and weight decay of 10^{-4} and used the same update rule as AlexNex:

$$v_{i+1} = 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \langle \frac{\partial L}{\partial w}|_{w_i} \rangle_{D_i}$$

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

V. Implementation Details

Replicating the architecture was a challenge since there were ambiguities regarding the implementation techniques that the authors used. For parts which were ambiguous, we tried to experiment with different approaches and choose the one which worked best in our case.

A. Architecture

The architecture used was identical to the one discussed in the Method section. One important thing to note is that although the authors claim to have 2 fully connected layers, the first one is convolutional. It takes 64 kernels of size 4x4 and outputs a 64-way vector which is then fed into the second convolutional layer. Thus, we labelled the first fully connected layer as a convolutional layer in our implementation.

B. Initial parameters

We initialised our weight and bias variables from a uniform random distribution in the range [-0.05, 0.05]. Parameters values for batch size, momentum and weight decay were the same as used by the authors (mentioned in Method section)

C. Preprocessing

Before training the network we perform preprocessing on the entire dataset. We first perform channel wise mean removal on an individual image followed by whitening. Whitening is done by subtracting the mean and dividing by the standard deviation of the entire dataset on a per channel basis.

D. Training

We trained the network for 45 epochs on batches of size 100. For each epoch we calculate the validation accuracy on the entire test set. We used test data for validation instead of having a separate validation set since we already had all the needed parameter values eliminating the need for any tuning. For the update rule we studied the AlexNet[4] paper and realized that MomentumOptimizer function in Tensorflow gives a close formulation of the rule. This seemed to give us the expected results so we didn't write our own implementation for the same.
The learning rate was fixed at 0.01 throughout the training process. We attempted to use learning rate decay by decaying the learning rate thrice during the training process as mentioned in AlexNet paper. However, it decreased our accuracy by 2% so we decided to omit it from our implementation.

E. Regularization

We used weight decay with a factor of 0.0001 as a regularization technique. For the improvement model we also use dropout for regularization which is discussed in the improvements section.

F. Testing

We tested our model on batches of size 100. Overall test accuracy was calculated as the ratio of number of incorrectly classified instances to the total number of instances for the entire test set.

VI. Replicating Figures

All the figures were replicated by calculating the error and loss on the entire test dataset.

A. Local response normalization

Authors tried to incorporate local response normalization (LRN) to see if it aids their results as it did for the AlexNet paper. Local response normalization helps in normalizing unbounded activations of ReLU neurons. We want to detect high frequency features with a larger response. By normalizing around the local neighbourhood of an excited neuron, it becomes even more sensitive as compared to its neighbours.
The paper indicates that LRN doesn't improve results in this particular case and we can confirm that from our results. We applied LRN following each convolutional layer and compared the performance with its presence and absence (error and loss curves given in Fig.2 and Fig.3 respectively). It's clear that we get better values for both error and loss without LRN. In fact, in our case using LRN decreases the convergence rate a lot more than indicated in the paper.

B. Whitening

The paper uses whitening as a preprocessing step. The whitening mechanism has been described in the Implementation Details section. We compared our results with and without whitening as shown in Figure 4. It is important to note that mean removal was still performed on individual images in both

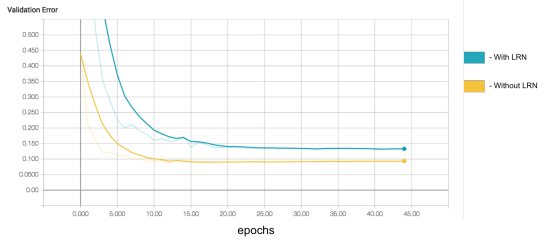

Fig. 2. Error when training with and without local response normalization

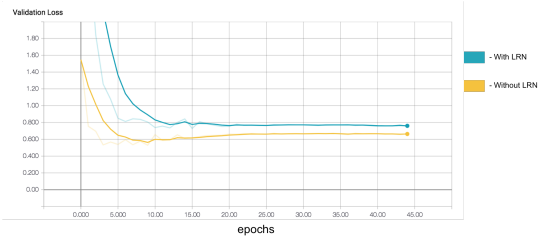

Fig. 3. Loss obtained when training with local response normalization

cases.

We can see that the network convergences faster when applying whitening which is what the authors try to demonstrate in the paper. However, for the authors the difference in convergence rate is much higher. We believe this is due to two reasons given below:

- The authors randomly choose 20% of traffic signs to produce this graph. This is quite ambiguous and would obviously lead to different results depending on the subset we choose. Since we use the entire test set the results are much more robust and indicate the true effect.
- The approach we took to whitening is the same as indicated in the coursework description. This is different from the actual definition of whitening. Besides, it is also unclear how exactly do the authors perform whitening

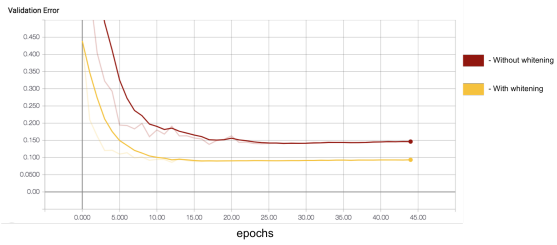

Fig. 4. Error obtained by training with and without whitening

C. Visualization of filters

The filters for the first and second convolutional layer have been visualized in figure 5 and 6 respectively.

D. First convolutional layer filters

For the first convolutional layer we have 32 filters of size 5x5x3. The three layers represent the three different colour channels. In order to produce these filters we take the three layers of of a filter which correspond to different channels and superpose them to represent it as an RGB image.

The filters in general look much more colourful than given in the paper. This is because Tensorflow uses a normalization algorithm in the background to rescale the values so that the largest one is 255. These filters represent preliminary features that the network is trying to extract from the images such as lines, edges, corners and colours in different signs.

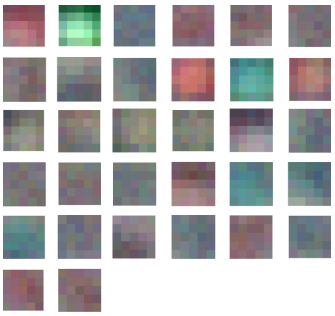

Fig. 5. The weights from the first convolutional layer

E. Second convolutional layer filters

For the second layer we again have 32 filters but these are of size 5x5x32. Based on the paper it wasn't clear how did the authors map these 32 layers to a single layer for grayscale representation. We tried different approaches like taking the maximum, minimum and average of the values along the third dimension. In the end we picked the average partly because it was visually the closest and because it makes sense to incorporate contributions of all 32 layers.

If we observe closely we can see that these filters are picking up higher level features like curved lines at different orientations. As mentioned before our filters are brighter than the ones presented in the paper due to scaling.

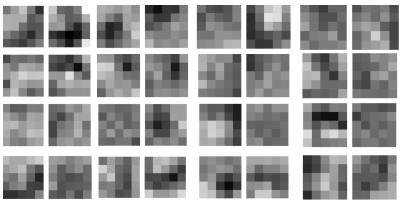

Fig. 6. The weights from the second convolutional layer obtained by averaging over the 32 dimensions

VII. Replicating Quantitative Results

The table below compares the our performance on different subsets of traffic signs with Zhang et al. Derestriciton and mandatory subsets are a bottleneck for us. The authors receive perfect accuracy on Mandatory signs while our accuracy is limited to 93.58%. We have discussed these misclassifications in the Discussion section.

	Speed Limits	Other Prohibitions	Derestriction	Mandatory	Danger	Unique
Zhang et al (2017)	99.93	99.80	99.44	100	99.13	99.90
Our method	90.64	97.02	87.03	93.58	83.73	98.74

The table below lists the overall accuracy achieved by our implementation against the one mentioned in the paper. Given the fact that we have replicated the paper as closely as possible the difference seems to be significant. We have touched upon the possible reasons behind this in our discussion.

Method	Accuracy
Zhang et al (2017)	99.84
Our method	90.6

The table below compares the our training and classification speed with the authors'. The training time refers to the total time taken to train the model for 45 epochs. Recognition time refers to the average time taken to classify a single test image.

Method	Training Time	Recognition Time	Configuration
Zhang et al (2017)	0.9 h/dataset	0.64 ms/frame	CPU: 8 I7-6700K
Our method	0.06 h/dataset	0.04 ms/frame	GPU: 1 x Tesla P100

VIII. Discussion

Although we attempted to follow all the details provided by the authors in order to replicate their results, the accuracy we could achieve on the test set was 90.6% as opposed to 99.84%. We believe this could be due to differences in our interpretation of various parts of the paper. Additionally, since we don't have access to their source code there could be differences in our implementation itself. The different issues we faced during our replication attempt along with possible explanations of difference in results are discussed below:

- **Preprocessing**: Initially we were preprocessing the dataset using only the whitening technique provided in the coursework description. This involved subtracting the mean and dividing by the standard deviation for the entire training dataset on a channel wise basis. However, it didn't lead to any improvements. On inspecting the individual images in the dataset we speculated that the brightness of individual images might be affecting the means calculated for the entire dataset in a negative fashion during whitening. For this reason we decided to perform mean removal (standardization) on all individual images before proceeding with whitening.
- **Learning rate decay**: In the beginning, we were under the impression that the authors were decaying the learning rate based on the details given in the AlexNet paper. However, we soon realized that this probably wasn't true since they nowhere mentioned usage of a validation set and hence no decaying criteria was employed in the first place.

  Out of curiosity, we still tried to examine if using learning rate decay will be beneficial. We experimented with dividing the learning rate by a factor of 10 and 100 when validation accuracy doesn't increase. The results did improve with a division factor of 10 but by an insignificant amount.

A. Misclassified data

By inspecting the instances where the signs were wrongly identified, we concluded that the network is not able to deal well with very bright images. This is most likely because in the case of very a bright light a lot of detail is lost (fig.7), while from the darker images, some information can still be extracted.

We obtained the lowest accuracy on the Derestriction and

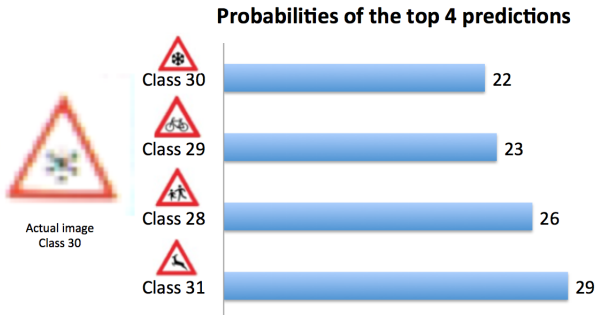

Fig. 7. Misclassified bright image; the probabilities have been normalised with regards to the top 4 picks

Danger examples which we have attributed to a possible lack of data. Furthermore, the diagonal line of the Derestriction signs makes them look quite similar which in turn increases the chance of them being misclassified as another one from the Derestriction set. In addition, we had problems classifying images where the sign was partially covered by shadows (sample 2 and 4, fig.8).

Other instances of misclassified images were the partially obstructed ones, eg. sample 3 (fig.8) where the traffic sign is missing a few pixels from the right side and is also affected by shadows which give it a more triangular look and is probably the reason why it ends up classified as one of the danger signs. Same for sample 5 (fig.8) where the actual image is the "Other danger" sign, but the black spot from the lower right side makes it look like "Road works" sign.

Fig. 8. Example of misclassified signs

IX. Improvements

A. Dropout

In addition to weight decay we used dropout as a regularization technique. The idea behind dropout is to set the output of a random subset of neurons to zero with a certain probability which is called the dropout rate. As a result, the excluded neurons don't contribute to network output and back propagation. Due to this random disabling and enabling, our neurons are able to learn multiple independent representations of the same data thereby reducing co-adaptation and overfitting.
We applied dropout after each pooling layer and first fully connected layer with varying probabilities. On using dropout our accuracy increased by 5.5%. A graph indicating the change the error with and without dropout is shown in figure 7.

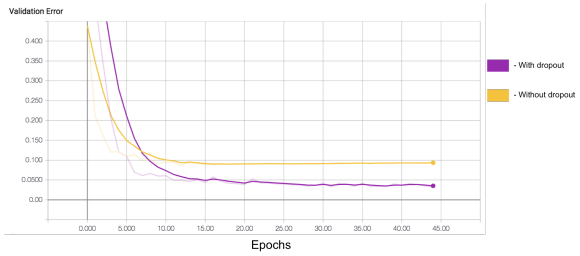

Fig. 9. Error comparison with and without dropout

B. Multi-Scale features

Traditionally, convolutional neural networks are organized in a feed forward structure in which the output of one layer is fed into the next one. When using multi-scale features the output of intermediate pooling layers is also fed into the classifer along with output from final layer. We decided to use this approach because we thought it might help in providing different scales of receptive fields to the classifier. This approach has been inspired from the paper written by Pierre Sermanet and Yann LeCun[5].
It is important to note that branched off layers have to undergo additional pooling to make sure all layers are proportionally subsampled before going into the classifier. We accounted for that by performing additional pooling with different sizes and strides for each layer.
Multi-Scale features improved our accuracy by 2%. The corresponding graph comparing the error with and without applying multiscale features is shown in figure 8.

The table at the end of this section summarizes the results achieved after performing all the improvements. "With" indicates the accuracy achieved when applying only that improvement to the replicated model.

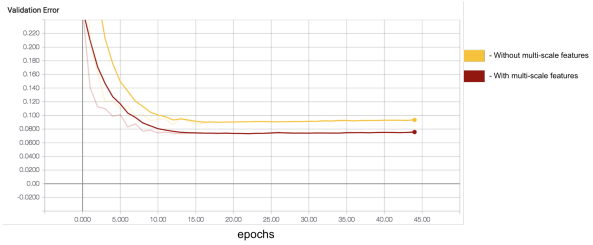

Fig. 10. Error comparison with and without multiscale features

Improvement	Accuracy
Replicated model	90.6
With Dropout	96.2
With Multi-scale features	92.3
With Dropout + Multi-scale features	96.8

X. Conclusion and Future Work

We have attempted to replicate a shallow network for traffic sign recognition that originally managed to achieve 99.84% accuracy on the GTSRB dataset. While implementing the network we found many ambiguities in the original architecture and, after trying different approaches while sticking to what the paper described, we only managed to get up to 90.6% accuracy. We then added some of our own improvements that increased the accuracy up to 96.8% while still keeping the recognition time low at around 0.04 ms/frame.
More work can be done to try to identify more accurately what parameters and architecture were used in the original paper, but we believe that the paper forgot to mention some important details. The accuracy obtained by following the method presented in the paper was significantly lower than the one they stated. Furthermore, data augmentation like rotating and flipping certain sings can be applied to enhance the original dataset which should lead to less overfitting and a possible increase in accuracy.

References

[1] PADEN, Brian, et al. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on Intelligent Vehicles, 2016, 1.1: 33-55. https://arxiv.org/pdf/1604.07446.pdf
[2] HUANG, Zhiyong, et al. An efficient method for traffic sign recognition based on extreme learning machine. IEEE transactions on cybernetics, 2017, 47.4: 920-933. http://ieeexplore.ieee.org/abstract/document/7433451/
[3] CIREAN, Dan, et al. A committee of neural networks for traffic sign classification. In: Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011. p. 1918-1921. http://ieeexplore.ieee.org/document/6033458/citations
[4] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. p. 1097-1105.
[5] SERMANET, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.