Ghent University

Machine Learning

Competition Report

Mathieu De Coster
Bart Middag
Daan Spitael
Group 15
Master of Computer Science Engineering

December 4th, 2015

1 Programming Approach

We chose Python 3.5.0 as our scripting language. We used the `numpy 1.10.1`, `scipy 0.16.0`, `sklearn 0.16.1` and `skimage 0.11.3` modules. For our neural networks, we used `scikit-neuralnetwork 0.3`. This library can be found on Github.

To run our application, execute `python3 main.py`. You can select a classifier and features with the `-c` and `-f` options, or select one of the default models from `default_models.py` using the `-m` option. For detailed information, run `python3 main.py -h`.

Settings such as the output directory for the results of our application and the location of the train and test data can be found and changed in `settings.py`.

2 Problem Analysis and Features

We can immediately distinguish three features when looking at the data set:

- Colour
- Symmetry
- Shape

Colour. Maldonado-Bascon [6] and Eichner [3] mention using colours in the HSV space to *detect* traffic signs. We wish to *recognize* traffic signs. We have tried using percentages of the most occurring colours (red, blue, white and black) in traffic signs as features for our learning algorithm. We obtained suboptimal results. It is easy to see why: a triangular traffic sign indicating an intersection and a traffic sign indicating maximum speed have similar colour percentages, but do not belong to the same class. After finding our best solution yet using HOG, as discussed below, we combined HOG and colour percentages, which improved the results on Kaggle by a small amount.

Symmetry. Symmetry is another interesting feature to consider: while all traffic signs have several axes along which their shapes are symmetric, the symbols on the signs themselves aren't always symmetric. Consider a triangular traffic sign indicating a sharp curve ahead. The sign itself has three symmetrical axes, but the curve is not symmetrical with regard to these axes. We did not use this feature.

Shape. Moutarde *et. al.* [7] note that using shape instead of color as a feature makes the recognition insensitive to color variability and robust to variances in illumination.

It can be difficult to accurately determine the shape of an object. Because of noise and the background part of images, we cannot always accurately determine the shape of an object. We have tried several approaches.

We used canny edge detection [1] to try and detect shapes within traffic signs. Often background parts of the image cause the algorithm to produce less than optimal results.

Finally, we used Histograms of Oriented Gradients [8]. This method allows us to accurately measure the shape of an object. We read many papers where this feature played an important role in the successful classification of traffic signs [4] or hand-written digits [5], where shape is also paramount. We used this method in our best-performing solution, with similar parameters as described by Stallkamp et al. [9]. They used 40x40 images with 5x5 pixel cells, but we found that 48x48 images and 8x8 pixel cells performed slightly better.

3 Preprocessing

As concluded above and briefly mentioned in [4], colour-based features alone are not suited for traffic sign recognition. We converted all images to grayscale as a first preprocessing step.
Before determining the Histogram of Oriented Gradients that we used for our current solution, we rescaled the images to 48x48. We found that it performs best at this size.
We also performed histogram equalization on the resulting image to normalize overly bright or overly dark images.
Finally, after feature detection, we applied feature scaling to every feature, as a lot of estimators in `sklearn` might behave badly if each individual feature does not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.

4 Machine Learning Techniques

We have used and evaluated a variety of classifiers:

- Logistic Regression
- Linear Discriminant Analysis (LDA)
- Support Vector Machines (SVM)
- Artificial Neural Networks
- Convolutional Neural Networks

For the first deadline, we used LDA for dimensionality reduction, as proposed in [4]. For the HOG features, we obtained a reduction in dimensionality from 800 to 80. For classification of the reduced dimensionality features, logistic regression and SVM both far outperform LDA (which we tested first, as it was proposed as a baseline classifier in [9]), but we found SVM to perform slightly

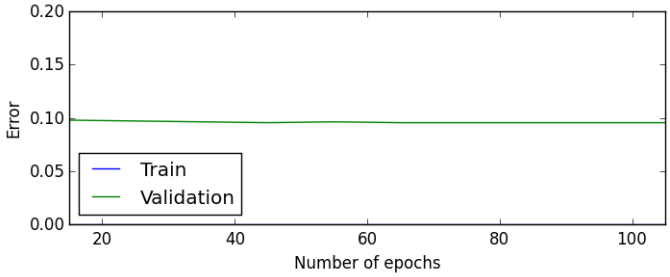

Figure 1: Validation and train error in function of the number of epochs, from 15 to 105. Note that the train error is always 0 and the validation error decreases only slightly. The code used to generate this plot can be found in the error_plot.py file.

better with the current parameters (a linear kernel type and a penalty parameter C of 1).

After the first deadline, we implemented an artificial neural network. Using `GridSearchCV`, we tried to obtain good parameters for our network. We chose to use one hidden layer to reduce complexity of the network, and used the following possible parameters:

- **Layer type** Rectifier, Tanh, Sigmoid.
- **Learning rate** 0.05, 0.01, 0.005, 0.001.
- **Hidden units** $\min(iC, oC) + 1$,$\text{mean}(iC, oC)$, $\max(iC, oC) - 1$, with iC the number of input neurons and oC the number of output neurons.

We obtained the following parameters from `GridSearchCV`: a rectifier activation function, a learning rate of 0.005 and 426 hidden units.

We let the network train for 25 epochs. From tests, we see that the validation error still decreases when increasing the amount of epochs, but only very little. This is shown in Figure 1. To reduce computational complexity, we chose this lower amount of epochs.

We noticed in the results of this neural network that most images were classified correctly with high confidence. Signs that were classified incorrectly or with low probabilities were often traffic signs that are similar in shape to signs belonging to other classes, but with different colours, e.g. classes E9E and X. For this reason, we tried training neural networks with raw RGB values and with hue values as features. Both performed worse than our neural network with HOG as features. To minimize the impact of wrong classifications of our HOG neural network classifier, we chose to train both this network and a network with RGB values and then take the weighted sum of the results of both classifiers, normalized to lay within the range $[0, 1]$. Because the neural network using HOG

outperforms the neural network using RGB, we gave the former a weight of 0.7, and the latter a weight of 0.3. We obtained our best Kaggle score by combining these two classifiers. This was implemented in the `EnsembleClassifier` class, in `ensemble.py`.

Also in `ensemble.py`, we implemented blended learning: a main classifier is trained on the predicted probabilities of its subclassifiers during cross-validation. The subclassifiers are then trained on the full dataset.
In order for the cross-validation probabilities to be similar to the probabilities during the test phase, it is paramount that the cross-validation scheme creates folds that are split realistically. Many of the images in the training dataset have duplicates in different sizes and rotations, but of course it is not likely that an image in the test set would be identical to an image in the training set. We implemented our own cross-validation iterator called `ClassLabelKFold`, a K-fold iterator variant with non-overlapping labels (ensuring that images with the same Pole ID are always in the same fold) but with overlapping classes (ensuring that every class is represented in both training and test set).

We performed blended learning using a neural network with HOG as features and an MLP with RGB as features. While we obtained good results, we did not improve upon our best results using the ensemble method. We also blended an MLP with HOG as features and a convolutional neural network with one hidden layer with 150 channels and a (4,4) kernel shape and a (2,2) pool shape, using max pooling. Using LDA for dimensionality reduction, we reduced the dimensionality from 162 to 80. We obtained good results, but a lower Kaggle score than our ensemble classifier.

Finally, we tried to use a convolutional neural network with two hidden layers with the following configuration:

1. Rectifier, 100 channels, (7,7) kernel shape, (2,2) pool shape, max pooling
2. Rectifier, 150 channels, (4,4) kernel shape, (2,2) pool shape, max pooling

We based this configuration on the findings of Ciresan et al. [2].

We noticed that the network was very confident in its guesses, as most of them had probabilities between 0.98 and 1.0. We presume we scored lower on Kaggle because the network gave some false positives with high confidence. We did not manage to improve on the performance for this classifier before the final deadline.

4.1 Results

We obtained our best score on Kaggle by using two neural networks and taking the normalized weighted sum of their results. The first neural network used HOG as features, and was weighted with 0.7. It had one hidden layer with the following configuration: a rectifier activation function, a learning rate of 0.005 and 426 hidden units. For the second neural network, we used the same parameters, but raw RGB values as features. It was weighted with 0.3.

Because of the low validation error (0.097 with 25 epochs) shown in Figure 1, we conclude that we are not overfitting.

We selected this ensemble classifier and a blended learning classifier with two neural networks using HOG and RGB as our solutions on Kaggle. While the blended learning classifier may not yield the highest score, we think that it is still a robust classifier.

References

[1] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[2] Dan Ciresan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[3] Marcin L. Eichner and Toby P. Breckon. Integrated speed limit detection and recognition from real-time video.

[4] R. Benenson M. Mathias, R. Timofte and L. Van Gool. Traffic sign recognition - how far are we from the solution? In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2013)*, August 2013.

[5] Subhransu Maji and Jitendra Malik. Fast and accurate digit classification. Technical Report UCB/EECS-2009-159, EECS Department, University of California, Berkeley, Nov 2009.

[6] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on support vector machines.

[7] Fabien Moutarde, Alexandre Bargeton, Anne Herbin, and Lowik Chanussot. Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 1122–1126. IEEE, 2007.

[8] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[9] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323 – 332, 2012. Selected Papers from {IJCNN} 2011.