Title:
Multiple-dataset Traffic Sign Classification with OneCNN

Authors: Fran Jurišić and Ivan Filković and Zoran Kalafatić
University of Zagreb, Faculty of Electrical Engineering and Computing

Paper link: click here

Group ID: 6
Group Members: Prithvi Raj (2016A7PS0013P)
Aashish Singh (2016A7PS0683P)
Aryan Sawhney (2016A7PS0091P)

Introduction

- Traffic sign classification is an important problem for many real life problems like autonomous driving systems, driver assistance systems etc.
- Traffic sign classification is different from other classification problems because the object to be classified is meant to easily spotted but there are many classes and the classes have a high degree of similarity.
- This paper implements a single CNN classifier for traffic signals and compares it’s performance with other more complex classification systems on different datasets.
- The architecture is based on branched CNN used by Yann LeCun and Pierre Sermanent for similar traffic sign classification.

Architecture

- < show architecture image >
- Traditional CNNs work by detecting low level features in the initial layers and high level ones later.
- Pooling layer is used to decrease the spatial dimensions to focus more on the detected local features. This leads to some loss of spatial information.
- Branched architecture hopes to overcome this problem by feeding the output of earlier layers along with the later layers to the classification stage.
- This allows the classifier to make a decision based on the low level features as well as the high level feature.

Methodology

- Most of the details of the architecture was given in the paper itself and were implemented as given.
- Some of the parameters such as learning rates, momentum, dropout factor etc. we had to determine using trial and error as values were not given in paper, or vlaues given were not giving good results.
- We only used one data set (Belgium TS) because of the high training time.
- Number of epochs were also limited to 50.

Implementation Details

- We used Keras Functional API for implementing our model.
- Other supporting packages used are
  - Numpy
  - Scikit-learn
  - Scikit-image
  - Matplotlib
- We implemented the model in jupyter notebook for ease of visualisation and for seperation of time consuming training step from rest of the code.

Implementation Details

- **Loss function:** Categorical cross-entropy (softmax loss)
- **Optimisation technique:** Stochastic Gradient Descent (SGD)
    - Initial Learning Rate = 0.03
    - Momentum = 0.3
    - Batch size = 32
    - Epochs = 50
- **Dataset augmentation:** ImageDataGenerator() class from keras was used to augment the training data by introducing vertical and horizontal shifts and zooming the training images to create additional training samples.
- **Dropout:** dropout of 0.25 applied to different layers.

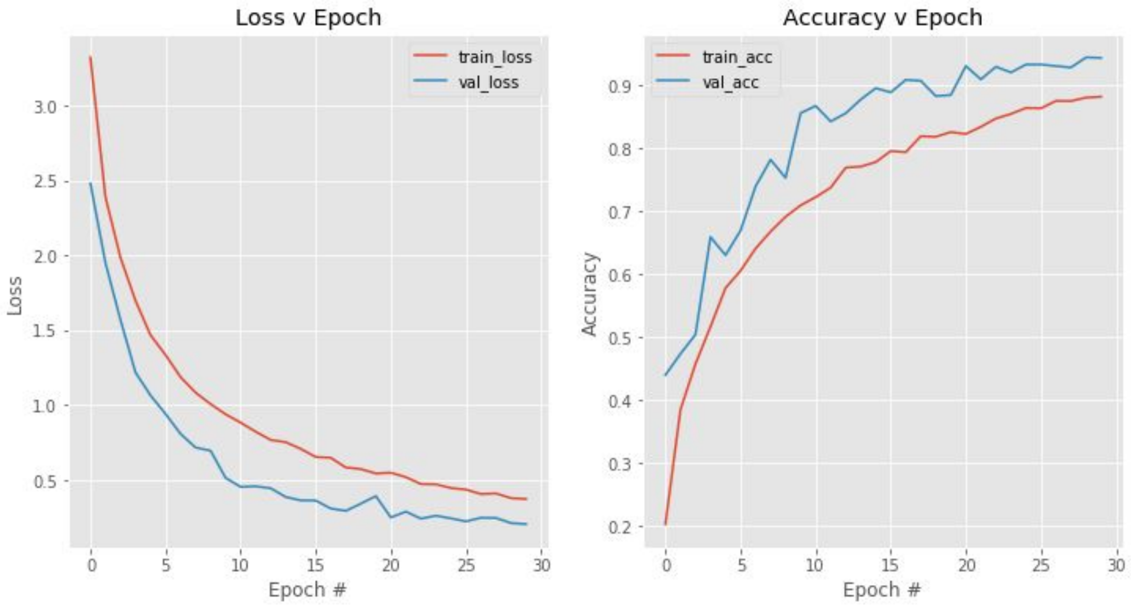
Results

- We achieved an testing set classification accuracy (i.e % of images classified correctly) of roughly **97%** on the Belgium Traffic Sign Classification dataset which is quite close to the accuracy achieved by the authors (**98%±0.2**)
- Following table gives the average values of precision, recall and F-factor

	Precision	Recall	F-factor
Average	0.92	0.90	0.89
Weighted Avg.	0.97	0.97	0.97

Results

- Training curves



Conclusions

- The architecture has been in almost exactly the same manner as given in the research paper.
- The decreased accuracy of our network compared to the authors' implementation can be attributed to the fact that we had to guess some of the parameters and also number of training epochs we used is much less than the authors'.

Dataset

- Belgium Traffic Sign Classification dataset was used. (here)
- The data set consists of 4575 training images and 2554 testing images across 62 classes.
- The images were collected by cameras mounted on a car.
- For classification only purposes, the images have been cropped to include only the traffic sign regions. But the dataset also contains additional uncropped images for detection problems as well.

References

- Keras documentation
- www.towardsdatascience.com
- www.pyimagesearch.com
- https://machinelearningmastery.com