Traffic Sign Recognition:
A single-image, multi-class classification problem

Alessandro Staffolani
Computer Engineering
At University of Bologna
2019 Bologna, Italy

alessandr.staffolan3@studio.unibo.it

ABSTRACT

This project work aims to realize and compare some Convolutional Neural Networks (CNNs). These CNNs have to classify images belonging to the well-known dataset "*German Traffic Sign Recognition Benchmark*" [1]. It is a comprehensive, lifelike dataset of more than 50,000 traffic sign images. It reflects the strong variations in visual appearance of signs due to distance, illumination, weather conditions, partial occlusions, and rotations. The dataset comprises 43 classes with unbalanced class frequencies. The dataset has been classified using four models; two of them were created specifically for this problem, while the others are the state of art of the models for image classification using CNNs. Each model has been trained using six different configurations and finally the 24 results on test set were taken to allow comparisons.

I. INTRODUCTION

The recognition of traffic signs is a challenging real-world problem of high industrial relevance. In particular, the signs dataset shows a wide range of variations between classes in terms of colour, shape, and the presence of pictograms or text. It is composed by more than 50,000 traffic sign images, divided into 43 classes (see

Fig. 1) with unbalanced frequencies. Each traffic sign image has a size between 15 x 15 to 250 x 250 pixels. The images also contain a margin of 10% (at least 5 pixels) around the traffic sign. The dataset is accompanied by a csv file where are stored both the information about the class of each image and the two coordinates of the ROI (Region Of Interest) of the image, in particular top-left corner and bottom-right corner.

The entire dataset has been divided into 2 subsets, one for training (containing 39,209 traffic sign images) and the other for testing (containing 12,630 traffic sign images). Successively 20% of the training subset has been randomly selected to be used as validation set during the training of the models of the CNNs to help avoiding overfitting on training data.

Fig. 1 The 43 classes of the dataset

To perform this analysis on the dataset the well-known python library Keras [2] on top of TensorFlow [3] framework has been used. Additionally, git [4] and GitHub [5] have been used to maintain history and versioning of the work done. Finally, both TensorBoard and Jupyter Notebook were used to visualize the result, the former to plot training progress over all runs, the latter to perform some interactive analysis on the trained models.

II. MODELS IMPLEMENTED

In this paragraph the models that were trained on the dataset previously described will be discussed. In particular, 4 models were trained: two of them were created from scratch based on the knowledge of the problem, the others two were selected from the list of the state of art of Convolutional Neural Network for image classification. The former two were designed trying to be as simple and small as possible, while the latter two were trained to compare the result obtained from the previous ones.

A. *Simple Model*

Simple Model is the first model that was designed and, as the name suggests, it is the simplest and the smallest one (see Fig. 2).

It is composed of three *Convolutional* layers, with a depth of 32, 64 and 128 nodes respectively. Each layer has a filter size of 3 x 3 and they use *ReLU* [6] as activation function. Between each Convolutional layers, to avoid overfitting as much as possible, there is always a *MaxPooling* layer with size 2 and stride 2 followed by a *Dropout* layer with a rate of 20%.

At the end of the CNN there is a *Flatten* layer to transform the images matrix into vector and then there is a *Dense* layer with 43 nodes just like the classes that the model has to distinguish.

As a whole, the model has 362,411 parameters, it requires a memory space of 2.937 MB and it has a depth of 11 layers.

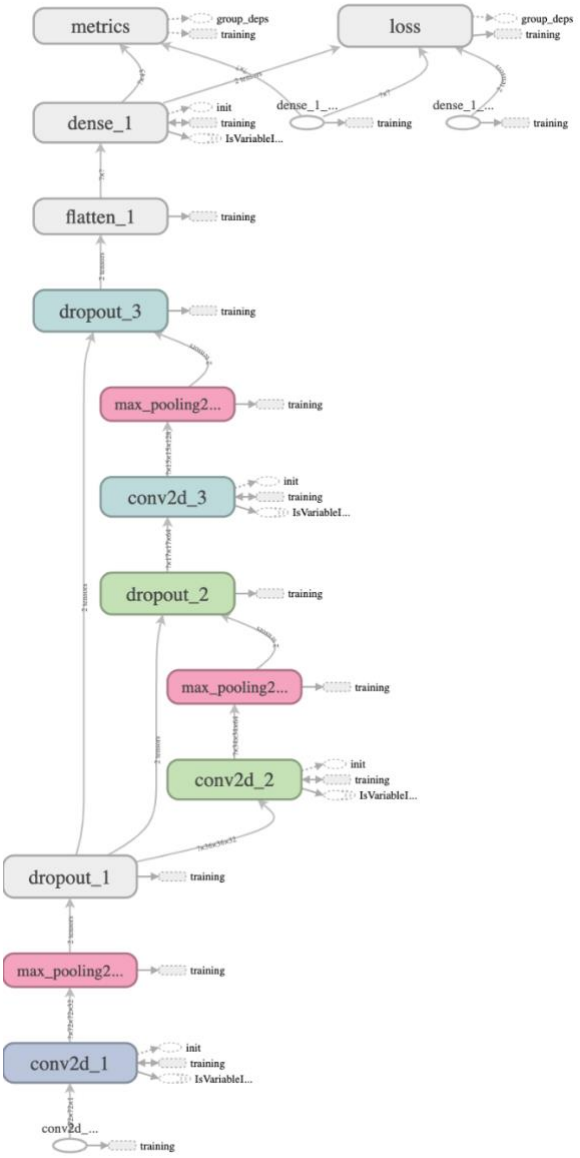

Fig. 2 Simple Model architecture

B. *Advanced Model*

The second model designed is a larger CNN with more *Convolutional* layers and a greater depth in final *Dense* layers (see Fig. 3).

The network is thus composed by three pairs of *Convolutional* layers. In each pair the former layer filters the images with a kernel 3 x 3 using padding to maintain the same image size, then the latter filters with a kernel of 3 x 3 without padding, while, like the activation function, all *Convolutional* layers use *ReLU*. The size of these pairs are 32 at the beginning, 64 in the middle, and 128 at the end. As for the previous model, also in this one between each pair of *Convolutional* layers a *MaxPooling* layer 2 x 2 and a *Dropout* layer of 20% are present. At the end of the network, after the *Flatten* layer, there is an additional *Dense* layer with 512 nodes with *ReLU* activation function followed by a *Dropout* of 50%, finally there is the *Dense* layer with 43 nodes as the classes to be distinguished.

As a whole, the model has 3,520,267 parameters, it requires a memory space of 15.945 MB and it has a depth of 16 layers.

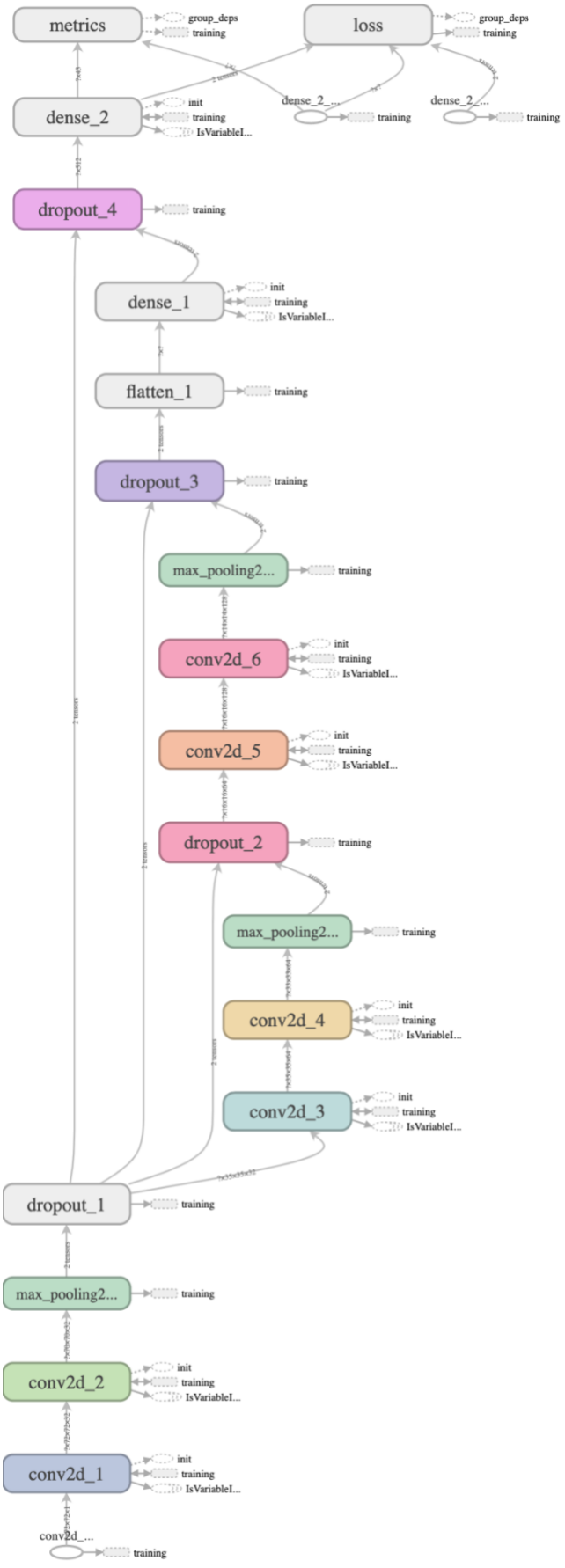

Fig. 3 Advanced Model architecture

C. *State of art Models*

In addition to the model previously described, to allow a better analysis on the developed models, the training was performed also on two famous state of art models: *Xception* [7] and *NASNetMobile* [8]. Both models are preloaded on Keras library and so it was not necessary to create them. The library also allows to load them with a pre-trained set of weights that are trained on the well-known *Imagenet* dataset, but in this case it was decided to train them from the beginning. It is worth pointing out that *Xception* has 20,949,011 parameters, it requires a memory space of 94.298 MB and it has a depth of 134 layers, while *NASNetMobile* has 4,314,591 parameters, it requires a memory space of 29.041 MB and it has a depth of 771 layers.

III. CONFIGURATION USED

The models previously described were all trained with the same configurations. In particular, all the models were trained for 30 epochs with a batch size of 32. The models were also compiled using SGD optimizer (*Stochastic Gradient Descent*), while the training set, as mentioned before, was divided into a validation set (using as split factor 20%) in such a way that the training images were 31,367 and the validation images were 7,842.

Images had been pre-processed before they were added to the training or testing algorithm and the transformations applied were:

1. Resizing: all the images had been resized to a dimension of 72 x 72;
2. Equalizing: the histogram equalization had been executed on all the images, in particular if the image was in RGB mode the HSV equalization had been applied, in which only the colour intensity had been equalized;
3. Normalizing: all the images had been normalized to have intensity value between 0 and 1.

Finally, as it was mentioned before, each model was trained in 6 different configurations.

A. *Grayscale*

The first configuration is the simplest one, in which – in a very simple way - the models were trained using images in grayscale format. That is why an additional pre-processing step has been added to perform the conversion.

B. RGB

The second configuration is the same as the previous one, but instead of using grayscale images, it used coloured images in RGB format.

C. Grayscale using ROI

The third configuration added two additional pre-processing steps: the first one to convert images into their grayscale representation (as for the first one), while the second one to crop the images using the ROI (Region Of Interest) information provided in the dataset information file. Therefore, the images were previously cropped to their ROI and then they were resized to the predefined size (72 x 72).

D. RGB using ROI

The forth configuration is the same as the previous one, but instead of using grayscale images, it used coloured images in RGB format.

E. Grayscale using image augmentation

The fifth configuration added the image augmentation as pre-processing step, in addition to the conversion in grayscale format step. This step consists in increasing the dimension of the dataset, as to increase the number of images used as training examples, by applying some transformations to the images. In particular, the images were shifted vertically and horizontally by a range of 0.1, they were zoomed by a range of 0.2, they were sheared by a range of 0.1, they were rotated by an angle range of 10 degrees and finally their brightness was changed by a range between 0.5 and 1.5.

F. RGB using image augmentation

The sixth configuration is the same as the previous one, but instead of using grayscale images, it used coloured images in RGB format.

IV. RESULTS

In this chapter the results obtained will be analysed. First of all the ranking among the four models will be analysed for each configuration. Secondly, for each model the ranking among the six configurations will be illustrated. Finally, the overall ranking will be shown. It is worth pointing out that the test set used to compare the models on new data, for each model and configuration, has been subjected to the same pre-processing steps to ensure a correct comparison.

A. Grayscale ranking

In this configuration, grayscale images, the model that obtained the best result was *Xception* model with an accuracy of 97.72% followed by the *Advanced* model with 97.69%, while surprisingly the worst result was obtained by *NASNet Mobile* model with an accuracy of 95.42% (see Tab. I).

TABLE I
Ranking among models for the grayscale configuration

Model	Test loss	Test accuracy
Xception	**0.0879**	**0.9772**
Advanced	0.1115	0.9769
Simple	0.1884	0.9602
NASNet Mobile	0.2282	0.9542

B. RGB ranking

In this configuration, RGB images, the model that obtained the best result was *Xception* model with an accuracy of 98.20% followed by the *Advanced* model with 97.86%, while surprisingly the worst result was obtained by *NASNet Mobile* model with an accuracy of 95.53% (see Tab. II).

TABLE II
Ranking among models for the RGB configuration

Model	Test loss	Test accuracy
Xception	**0.0732**	**0.9820**
Advanced	0.1112	0.9786
Simple	0.2097	0.9602
NASNet Mobile	0.2133	0.9553

C. Grayscale using ROI ranking

In this configuration, grayscale images using ROI, the model that obtained the best result was *Xception* model with an accuracy of 97.21% followed by the *Advanced* model with 97.07%, while the worst result was obtained by *Simple* model with an accuracy of 94.35% (see Tab. III).

TABLE III
Ranking among models for the grayscale with ROI configuration

Model	Test loss	Test accuracy
Xception	**0.1382**	**0.9721**
Advanced	0.1608	0.9707
NASNet Mobile	0.2584	0.9546
Simple	0.3323	0.9435

D. RGB using ROI ranking

In this configuration, RGB images using ROI, the model that obtained the best result was *Xception*

model with an accuracy of 97.26% followed by the *Advanced* model with 96.70%, while the worst result was obtained by *Simple* model with an accuracy of 94.98% (see Tab. IV).

TABLE IV
Ranking among models for the RGB with ROI configuration

Model	Test loss	Test accuracy
Xception	**0.1043**	**0.9726**
Advanced	0.1714	0.9670
NASNet Mobile	0.2006	0.9610
Simple	0.2803	0.9498

E. Grayscale using image augmentation ranking

In this configuration, grayscale images using image augmentation, the model that obtained the best result was surprisingly *Advanced* model in test accuracy with a value of 97.84%, while *Xception* model in test loss with a value of 0.0964. The worst result was obtained by *NASNet Mobile* model with an accuracy of 95.54% (see Tab. V).

TABLE V
Ranking among models for the grayscale with augmentation configuration

Model	Test loss	Test accuracy
Advanced	0.0974	**0.9784**
Xception	**0.0964**	0.9772
Simple	0.1888	0.9588
NASNet Mobile	0.1967	0.9554

F. RGB using image augmentation ranking

In this configuration, RGB images using image augmentation, the model that obtained the best result was surprisingly *Advanced* model in test accuracy with a value of 98.02%, while *Xception* model in test loss with a value of 0.0787. The worst result was obtained by *NASNet Mobile* model with an accuracy of 95.08% (see Tab. VI).

TABLE VI
Ranking among models for the RGB with augmentation configuration

Model	Test loss	Test accuracy
Advanced	0.0939	**0.9802**
Xception	**0.0787**	0.9799
Simple	0.2019	0.9607
NASNet Mobile	0.2374	0.9508

G. Simple Model ranking

Simple model performed best when trained using RGB images, in particular with image augmentation where it scores its best accuracy of 96.07%. While, the worst performances where registered using ROI images, both RGB and grayscale with an accuracy less than 95% (see Tab. VII).

TABLE VII
Ranking among configurations for Simple model

Configuration	Test loss	Test accuracy
RGB augmentation	**0.2019**	**0.9607**
RGB	0.2097	0.9602
Grayscale	0.1884	0.9602
Grayscale augmentation	0.1888	0.9588
RGB ROI	0.2803	0.9498
Grayscale ROI	0.3323	0.9435

H. Advanced Model ranking

Advanced model performed best when trained using RGB images, in particular with image augmentation where it scores its best accuracy of 98.02%, while the worst performances where registered using ROI images, both RGB and grayscale with an accuracy of 97.07% and 96.70% respectively (see Tab. VIII).

TABLE VIII
Ranking among configurations for Advanced model

Configuration	Test loss	Test accuracy
RGB augmentation	**0.0939**	**0.9802**
RGB	0.1112	0.9786
Grayscale	0.0974	0.9784
Grayscale augmentation	0.1115	0.9769
RGB ROI	0.1608	0.9707
Grayscale ROI	0.1714	0.9670

I. Xception Model ranking

Xception model performed best when trained using RGB images, in that configuration it scores its best accuracy of 98.20%. On the contrary, the worst performances were registered using ROI images, both RGB and grayscale with an accuracy of 97.26% and 97.21% respectively (see Tab. IX).

TABLE IX
Ranking among configurations for Xception model

Configuration	Test loss	Test accuracy
RGB	**0.0732**	**0.9820**
RGB augmentation	0.0787	0.9799
Grayscale	0.0879	0.9772
Grayscale augmentation	0.0964	0.9772
RGB ROI	0.1043	0.9726
Grayscale ROI	0.1382	0.9721

J. NASNet Mobile Model ranking

NASNet Mobile model performed best when trained using RGB images with ROI, in which scores 96.10%, while the worst results were scored in RGB with image augmentation configuration, in which it recorded 95.08% (see Tab. X).

TABLE X
Ranking among configurations for NASNet Mobile model

Configuration	Test loss	Test accuracy
RGB ROI	**0.2006**	**0.9610**
Grayscale augmentation	0.1967	0.9554
RGB	0.2133	0.9553
Grayscale ROI	0.2584	0.9546
Grayscale	0.2282	0.9542
RGB augmentation	0.2374	0.9508

K. Overall ranking

Overall, the model that performed the best results was *Xception* model, which wins in almost all configurations and scored its best in RGB. The second better model was *Advanced* model, which had similar scores to the *Xception*, while, at the end of the ranking, NASNet Mobile model and Simple model are contending the worst performances (See Tab XI).

V. CONCLUSION

In conclusion, from the analysis that was performed on this dataset, it is possible to point out that, on the whole, good results were obtained; in particular it can be observed that using a relatively small and not heavy model such as *Advanced* model, it is possible to obtain almost identical results to the ones obtained by a state of art model such as *Xception* model.

TABLE XI
Ranking among all models and configurations

Model	Configuration	Test loss	Test accuracy
Xception	**RGB**	**0.0732**	**0.9820**
Advanced	RGB using image augmented	0.0939	0.9802
Xception	RGB using image augmented	0.0787	0.9799
Advanced	RGB	0.1112	0.9786
Advanced	Grayscale using image augmented	0.0974	0.9784
Xception	Grayscale	0.0879	0.9772
Xception	Grayscale using image augmented	0.0964	0.9772
Advanced	Grayscale	0.1115	0.9769
Xception	RGB using ROI	0.1043	0.9726
Xception	Grayscale using ROI	0.1382	0.9721
Advanced	Grayscale using ROI	0.1608	0.9707
Advanced	RGB using ROI	0.1714	0.9670
NASNet Mobile	RGB using ROI	0.2006	0.9610
Simple	RGB using image augmented	0.2019	0.9607
Simple	RGB	0.2097	0.9602
Simple	Grayscale	0.1884	0.9602
Simple	Grayscale using image augmented	0.1888	0.9588
NASNet Mobile	Grayscale using image augmented	0.1967	0.9554
NASNet Mobile	RGB	0.2133	0.9553
NASNet Mobile	Grayscale using ROI	0.2584	0.9546
NASNet Mobile	Grayscale	0.2282	0.9542
NASNet Mobile	RGB using image augmented	0.2374	0.9508
Simple	RGB using ROI	0.2803	0.9498
Simple	Grayscale using ROI	0.3323	0.9435
Xception	RGB	0.0732	0.9820

REFERENCES

[1] J. Stallkamp, M. Schlipsing, J. Salmen e C. Igel, «The German Traffic Sign Recognition Benchmark: A multi-class classification competition,» *Proceedings of the IEEE International Joint Conference on Neural Networks,* p. 1453–1460, 2011.

[2] Team Keras, «Keras,» [Online]. Available: https://keras.io/.

[3] Google Brain, «TensorFlow,» Google Brain, [Online]. Available: https://www.tensorflow.org/.

[4] L. Torvalds, «Git,» [Online]. Available: https://git-scm.com/.

[5] GitHub Inc, «GitHub,» [Online]. Available: https://github.com/.

[6] V. Nair e G. E. Hinton, «Rectified linear units improve restricted boltzmann machines,» 21 06 2010.

[7] F. Chollet, «Xception: Deep Learning with Depthwise Separable Convolutions,» in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, 2017.

[8] V. V. J. S. Q. V. L. Barret Zoph, «Learning Transferable Architectures for Scalable Image Recognition,» in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, 2018.