Road Signs Classifier

Jakub Nowicki, jakub.nowicki@vodafone.com

Domain background and problem statement

I would like to focus on a computer vision problem of classifying road signs with Convolutional Neuro Networks. We have witnessed a huge progress in CNNs since GPUs became widely available and I am curious if I can apply some of the recent discoveries in my project.

Recognising road signs can have multiple practical use cases. Most obvious use is in self-driving cars. Others are adding functionality to dash-cams to advise the driver on the current speed limit, or creating and maintaining a map of road signs. It can also detect signs covered with graffiti.

A complete solution of a road sign recognition, apart from classification, should also include object detection, extraction and image pre-processing. In this project, I will focus on classification only.

Road signs are good candidates for image classification because they are standardized so there is little variance between signs of the same type and have high contrast features.

Apart from recognising the image, I also have two additional learning objectives:

- I would like to experiment with transfer learning which I find very promising
- I would like to apply some recent discoveries, like Spatial Transformer Network [2]

Dataset and inputs

I plan to use German Traffic Sign Recognition Benchmark (GTSRB) dataset [1] created by Institut für Neuroinformatik. GTSRB was used during a competition in 2011 and has since been used in many ML research projects. The current state-of-the-art solution [4] was created by DeepKnowledge Seville scored 99.71% on test accuracy and uses CNNs with STNs [5].

Sample images from GTSRB dataset:

Train set contains 39209 images and test set has 12630 images.

Solution proposal

I will try to address the problem of road sign classification using CNN with STN module and fully connected classification layer.

I consider using pre-trained model like ResNet18 or ResNet34 and compare it with simpler CNNs defined by me. Although pre-trained models look promising in such applications, they might not be the most efficient choice. They are pre-trained on classes with features that mostly do not appear in traffic signs. Also, traffic signs are extracted from bigger frames and resized to a small images so pre-trained models might be too big for the input. According to [2], six top performing architectures in original a GTSRB competition were all 2-layer CNNs. I would still like to see how they compare.

I would like to apply Spatial Transfomer Network [3], which removes spacial variations from the input. STNs learn to resize, center and affine transform the main object in the image making it easier to classify for subsequent layers.

Existing research [2] and [6] suggests pre-processing the images before using them for training:

- Train and validation split needs to take into account that the dataset contains images of the same physical signs grouped into tracks so a care is needed to avoid leaking data
- Down-sampling to 32x32
- Augmentation by rotation +/- 15%, scaling +/-10% and possibly other transforms
- Augmentation by reusing images [6] of similar signs

Benchmark and evaluation method

The main evaluation metric will be classification accuracy on the test set. It is more relevant metric for end users then Cross Entropy Loss.

My objective is to achieve 97% accuracy on the test set. State-of-the-art models achieve above 99% accuracy but I would like to focus on finishing the project and learning rather than high accuracy. Once I have a ready solution I can later try to push the model to better performance.

For testing, I plan to use a test set available in GTSRB, which will not be used for training neither for validation.

Project design

I would like to train the model using Sage Maker Jupyter notebook running on GPU-enabled instance. I will save a trained model to S3. I will save the logs from model training session into a Jupyter notebook that store the notebook in github for review.

After training the model, I would like to deploy the model to a SageMaker endpoint and expose via API Gateway for use by a web page. End users should be able to select a sign from a test set or upload their own image of a sign and request classification.

As a minimum, I would like to build a static HTML page using API gateway URL as a configuration parameter. If I have time left, I will try to release the flask web server with a page as a docker container.

Sources

[1] http://benchmark.ini.rub.de/?section=gtsrb&subsection=news
[2] http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf
[3] https://arxiv.org/pdf/1506.02025.pdf
[4] http://benchmark.ini.rub.de/?section=gtsrb&subsection=results
[5] https://www.sciencedirect.com/science/article/abs/pii/S0893608018300054?via%3Dihub
(abstract only)
[6] https://medium.com/@wolfapple/traffic-sign-recognition-2b0c3835e104