UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Traffic Sign Classification

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

This was an excellent effort!
Congratulations on successfully completing this project! 🎉

Do read through the review once for some tips and suggestions.
The architecture that you came up with is fit for the given problem.

Feel free to check out the following papers / projects on topics that go beyond the project's minimum requirements -

- Comparing different grayscaling techniques - https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0029740
- A simplified Mathematical presentation of convolution networks and their different layers and operations - https://arxiv.org/pdf/1603.07285.pdf
- A detailed look into the benefits and pitfalls of data augmentation - https://arxiv.org/pdf/1609.08764.pdf
- And here's a really cool project that implements a traffic sign classifier as a mobile app - https://github.com/OluwoleOyetoke/Accelerated-Android-Vision.

Keep up the great work and all the very best for your future projects!

Files Submitted

The project submission includes all required files.

- Ipython notebook with code
- HTML output of the code
- A writeup report (either pdf or markdown)

Dataset Exploration

The submission includes a basic summary of the data set.

Good job completing the TODOs to come up with a basic summary of the dataset -

```
# Number of training examples
n_train = X_train.shape[0]

# Number of validation examples
n_validation = X_valid.shape[0]

# Number of testing examples.
n_test = X_test.shape[0]

# What's the shape of an traffic sign image?
image_shape = X_train.shape[1:]

# How many unique classes/labels there are in the dataset.
signs = pd.read_csv("./signnames.csv")
n_classes = signs['ClassId'].count()
```

The submission includes an exploratory visualization on the dataset.

Well done plotting the traffic signs as well as the bar chart showing the number of images for each of the class labels. This should help you decide if you should generate additional data for some under-represented class of images.

Design and Test a Model Architecture

The submission describes the preprocessing techniques used and why these techniques were chosen.

Conversion of images to grayscale and normalization are both perfectly valid preprocessing techniques to be used here. Well done justifying these choices as well.

Another technique that you may also try out here is histogram equalization (CLAHE). This is a technique for adjusting image intensities to enhance contrast. For details, refer to https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html

You may also want to keep the following link as a reference that explains some other options available for preprocessing - http://cs231n.github.io/neural-networks-2/#datapre

The submission provides details of the characteristics and qualities of the architecture, including the type of model used, the number of layers, and the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.

Awesome!

- Well done using an appropriate architecture.
- Good job using dropout to combat overfitting. For a detailed analysis of dropouts, you may also refer to https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/

A few suggestions:

- Another technique that has been proven to be helpful here is batch normalization. This basically carries out the preprocessing at each of the layers, thereby making the model more robust to the bad initializations. Do check out - http://cs231n.github.io/neural-networks-2/#batchnorm
- You may also try experimenting with the `elu` activation, as it is speed-optimized - https://arxiv.org/abs/1511.07289
- Another problem with the `relu` activation is the issue of "Dying ReLUs". To combat this, we use Leaky ReLUs - https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- For visualizing the model architecture, you may also use Tensor Board - https://www.tensorflow.org/guide/graph_viz

The submission describes how the model was trained by discussing what optimizer was used, batch size, number of epochs and values for hyperparameters.

- Good job using an Adam optimizer. Another fairly recent advancement has been the introduction of cyclical learning rate. This does away with the need to experimentally come up with the best learning rate. Do check out the paper in your spare time - https://arxiv.org/abs/1506.01186
- All the parameters used to train the model are reasonable 👍

You may also want to keep the following resource as a reference that explains some proven strategies to optimize these hyperparameters - http://cs231n.github.io/neural-networks-3/#hyper

The submission describes the approach to finding a solution. Accuracy on the validation set is 0.93 or greater.

Great job discussing your approach concisely and achieving a validation accuracy greater than 93%.

To further improve the robustness of the model (as you correctly noted), you may also use a few data augmentation techniques to generate additional images for balancing the dataset. Feel free to get some ideas from - https://medium.com/@vivek.yadav/dealing-with-unbalanced-data-generating-additional-data-by-jittering-the-original-image-7497fe2119c3

In case you're interested in checking out the winning solution to a deep learning competition involving a similar problem, do check out - https://medium.freecodecamp.org/recognizing-traffic-lights-with-deep-learning-23dae23287cc

Test a Model on New Images

The submission includes five new German Traffic signs found on the web, and the images are visualized. Discussion is made as to particular qualities of the images or traffic signs in the images that are of interest, such as whether they would be difficult for the model to classify.

All the new images chosen are appropriate.
Also note that, in general, such models are likely to face difficulties with images that are occluded or have a lot of background clutter in them.

The submission documents the performance of the model when tested on the captured images. The performance on the new images is compared to the accuracy results of the test set.

Good job documenting the performance of the model when tested on the new images.

The top five softmax probabilities of the predictions on the captured images are outputted. The submission discusses how certain or uncertain the model is of its predictions.

Great job outputting the correct softmax probabilities for each of the new images.

Well done completing the optional part as well! The goodness of the model is evident by it's initial layer's ability to detect the key edges in the images without it being explicitly programmed to do so.

DOWNLOAD PROJECT

RETURN TO PATH