UDACITY

< Return to Classroom

DISCUSS ON STUDENT HUB

Traffic Sign Classification

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Brilliant Udacity Learner,

Congratulations! 🎉
You have successfully completed this **Traffic Sign Recognition Classifier** project. The implementation was really good and your answers and documentation are well-defined which makes it delightful to review. It's great that you have tried different approaches to make the model perform better. I must admit that I like this hardworking culture since we are able to validate this project in this submission. The accuracy obtained from your model clearly proves how excellent your implementation was. 👍🏽

You should be proud of the work done as you seem to have a good hold on python and Convolutional Neural Network. Please continue with this same spirit of hard work in the projects ahead. It was my pleasure to review this wonderful project.

Remarks

Here are some interesting resources about Convolutional Neural Networks:

- The 4 Convolutional Neural Network Models That Can Classify Your Fashion Images
- How to Develop Convolutional Neural Network Models for Time Series Forecasting
- Recognising Traffic Signs With 98% Accuracy Using Deep Learning

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)

Nice work! All the required files were included in this submission. Well done! 👍

Dataset Exploration

The submission includes a basic summary of the data set.

Excellent job utilizing necessary numpy functions to calculate the basic summary of the dataset. Each image in the dataset is 32 pixels x 32 pixels x 3 Channels (one each for RGB). Here are the following summary obtained:

```
Number of training examples = 34799
Number of testing examples = 12630
Number of validation examples = 4410
Image data shape = (32, 32, 3)
Number of classes = 43
```

The submission includes an exploratory visualization on the dataset.

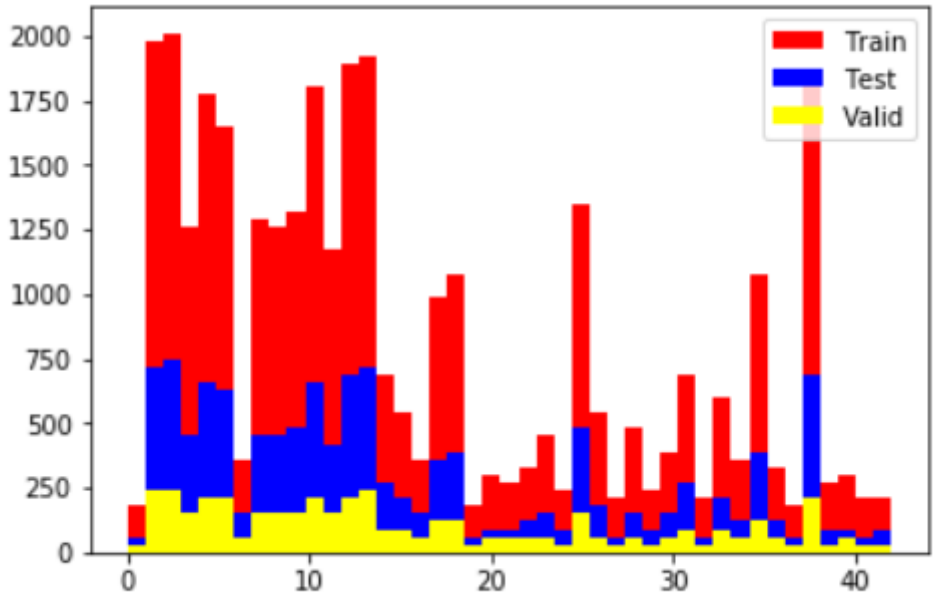
You did very well by producing an exploratory visualization of the dataset. Title and appropriate labels (y and x-axes labels) are also encouraged for a better understanding of the distribution of the training, testing and validation data. It was indeed interesting to evaluate the dataset particularly because of its distribution imbalance in the number of examples per class.

Suggestions and Comments

You could produce a visualization using histogram, scatter plot or bar chart showing the number of images in each class. This facilitates the representation of some features of your dataset that would lead to a better observation on how is the distribution of training, test and validation data.

- Here you'll find a host of example plots in Matplotlib with the code that generated them.
- You can also check this Basic Data Plotting with Matplotlib by Bespoke which discusses how useful are Histograms in any case where you need to examine the statistical distribution over a variable in some sample.
- I would also like to recommend this discussion together with this useful resources on How to Plot Multiple bars in matplotlib which could be very useful in this case; to display multiple numbers of

examples per class in a single figure like this example below:



Design and Test a Model Architecture

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

Well done providing an explicit description on the preprocessing techniques used and a discussion as to why these techniques were chosen. 👍

- Normalization
- Grayscale conversion

Suggestions and Comments

The images differ significantly in terms of features such as contrast, brightness and distribution, that is why preprocessing techniques are necessary to improve feature extraction and it helps the model converge faster. Here are some preprocessing techniques that I would like to recommend:

- Minimally, the image data should be normalized so that the data has mean zero and equal variance. For image data, (pixel - 128)/ 128 is a quick way to approximately normalize the data and can be used in this project.

- Here is a good resource about Image Data Pre-Processing for Neural Networks
- Since there was an uneven distribution on the number of images for each class as shown in the exploratory visualization of the training set images, Image augmentation techniques such as randomly rotates, translates, and shears could be applied on the input image to avoid replicates of the input dataset. There are a lot of good Python libraries for image transformation like OpenCVor Pillow. But I would personally prefer scikit-image which is the easiest library to use from my point of view

- You can also enhance the contrast in an image, in order to stretch out the intensity range using a contrast enhancement technique like openCV's Histogram equalization. Or you could also apply another technique called Contrast-Limited Adaptive Histogram Equalization (CLAHE).
- Noise reduction techniques like Gaussian Blur and Grayscale can also be used to improve the accuracy of a model.

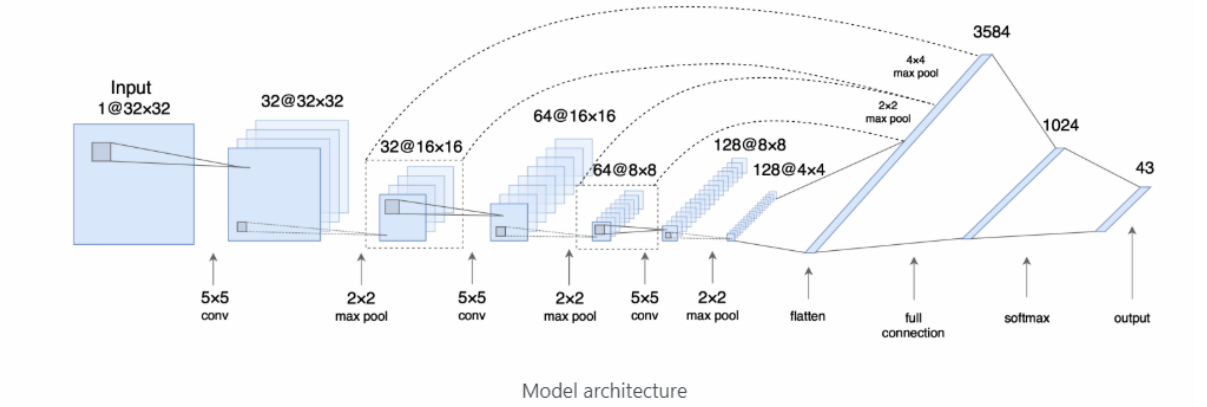
The submission provides details of the characteristics and qualities of the architecture, including the type of model used, the number of layers, and the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.

Great discussion in this section. You clearly outlined all the characteristics and qualities of your model architecture and these includes:

- The type of model used.
- The number of layers in the model.
- The input and output sizes of each layer.

Suggestions and Comments

You may check Daniel Nouri's tutorial and Pierre Sermanet / Yann LeCun paper to know more about Traffic Sign Recognition with Multi-Scale Convolutional Networks. It is fairly simple and has 4 layers: 3 convolutional layers for feature extraction and one fully connected layer as a classifier.



Model architecture

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

An elaborate description has been provided on how the model was trained by specifying the optimizer, batch size, number of epochs and values for hyperparameters used in this implementation. Awesome! 👏

- Learning rate = 0.001
- EPOCHS = 60
- BATCH_SIZE = 128

Suggestions and Comments

- Finding the right hyperparameters for your deep learning model can be a tedious process. Here is a good article about Hyperparameter Optimization with Keras which states that it should not be difficult to find state-of-the-art hyperparameter configuration for a given prediction task.
- Here is a link that contains A Comprehensive List of Hyperparameter Optimization & Tuning Solutions.

- A Quick Notes on How to choose Optimizer In Keras
- Check out this nice discussion on How big should batch size and number of epochs be when fitting a model.
- How does one choose optimal number of epochs.
- Visualize the training parameters, metrics, hyperparameters or any statistics of your neural network with TensorBoard. You may check this Tensorboard tutorial to learn how to start TensorBoard followed by an overview of the different views offered.

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

You have provided a very good discussion on the approach you took in finding the best solution which gives you an Accuracy on the validation set greater than 0.93. 👏

> validation set accuracy of 95.2%

Suggestions and Comments

The following regularization techniques can be used to minimize overfitting to training data:

- Dropout is an amazing technique which could drastically improve the generalization of your model. Normally you may only want to apply dropout to fully connected layers, as shared weights in convolutional layers are good regularizers themselves. You may check this helpful article on Why dropouts prevent overfitting in Deep Neural Networks.
- L2 Regularization. Using lambda = 0.0001 which seemed to perform best. The important point here is that L2 loss should only include weights of the fully connected layers, and normally it doesn't include bias term. The intuition behind it is that bias term is not contributing to overfitting, as it is not adding any new degree of freedom to a model. Here is An Overview of Regularization Techniques in Deep Learning.
- Early stopping.You can apply stopping with the patience of 100 epochs to capture the last best-performing weights and roll back when the model starts overfitting training data. A validation set cross entropy loss as an early stopping metric, the intuition behind using it instead of accuracy is that if your model is confident about its predictions it should generalize better. Please check this article to know How to Stop Training Deep Neural Networks At the Right Time Using Early Stopping.

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.

Excellent work providing a visualization of new German Traffic signs found on the web. A nice and detailed discussion on the particular qualities of the images that might make classification difficult was also well discussed in this section. 👏

> The images are of relatively good quality. Difficulties in classification might arise from shadows (image 1), over-exposure (3), blue fringing (3) and under-exposure. I did normalize the images so that the exposure problems are lessened to a manageable level.

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.

You did very well here by providing sufficient discussion on the accuracy/performance of the model when tested on the captured images. Well done comparing the performance of the new images to the accuracy results of the test set. 👍

Suggestions and Comments

You may use a Confusion Matrix to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

- Here is a Simple guide to confusion matrix terminology
- And a good resource to Understand Confusion Matrix.

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.

Well done using the model's softmax probabilities to visualize the certainty of its predictions. You've utilized tf.nn.top_k correctly to print the statement of the probabilities and support your explanation for which captured image example the model is certain or uncertain of its predictions.

Suggestions and Comments

Using the softmax activation function in the output layer of a deep neural net to represent a categorical distribution over class labels, and obtaining the probabilities of each input element belonging to a label. Please check some resources below to know more about softmax probabilities:

- Make sure to take the softmax of the logits before getting the top five probabilities.
- Use tf.nn.softmax on the logits before using `tf.nn.top_k`.
- The Softmax Function, Neural Net Outputs as Probabilities, and Ensemble Classifiers
- Understanding Softmax/Probabilities Output on a multi-class classification problem
- Extracting probabilities from a softmax layer in [tensorflow 1.00]
- Softmax Regression using TensorFlow

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START