P2 – Traffic Sign Classifier

Objectives:

1. Create a Convolutional Neural Net to:
   LOAD – PROCESS – TRAIN – VALIDATE – TEST German Traffic Sign Images
2. Ensure the criteria are met for the notebook and the Project Rubric

Project Rubric:

1. Files Submitted:

Attached along with this writeup are the test images and the .html version of the executed ipython notebook

2. Dataset Exploration:

- The 3 pickle files provide the inputs (32x32x3 arrays) and corresponding labels (0-42) for TRAINING(34799), VALIDATION(4410) & TESTING(12630)
- The distribution of labels look similar in all 3 datasets according the histogram plotted in the notebook, meaning even and fair distribution of data.
- Visualization of inputs in the dataset reveals that the images vary greatly in brightness, contrast, accuracy, intensity & noise.
- This might prove to be a negative influence on training our model, hence it highlights the importance of pre-processing the datasets to iron out the abnormalities.

3. Design and Test a Model Architecture:

1. **Pre-Processing**:
   - Initially performed zero centering and normalization, but this did not add significant improvement to the model performance.
   - Similarly tried gray scaling and this also on its own did not add much improvements.
   - Upon further study on the web and the contrast and brightness fluctuations in the sample set, realized the importance of 'HISTOGRAM EQUALIZATION'
   - Using Gray Scaling, Normalization & Histogram Equalizing helped me achieve significant improvements in the accuracy as also evident in the smoothness of pre-processed outputs.
2. **Model Architecture**:
   - As mentioned, started of with the existing LeNet Architecture.
   - Experimented with the following options:
     i. ReLU vs. Dropouts in different combinations
     ii. Max vs. Average Pool in Convolution Layers
     iii. Increase/Decrease Filter Size & Depth of Convolution Layers
     iv. Use of 1x1 Convolutions
     v. Batch Size, Learning Rate and Epoch tuning was also done

- All this lead and further study (after frustrating runs) on the web lead to the following architecture:
    i. **CONV_LAYER_1** (Input : 32x32x1 - Conv : 5x5x64 - COut : 28x28x6 - stride 1x1) - **ReLU** - **Max Pool** : (Filter 2x2 & Stride 2x2) - Pout : 14x14x64
    ii. **CONV_LAYER_2** (Input : 14x14x64 - Conv : 5x5x32 - COut : 10x10x32 - stride 1x1) - **ReLU** - **Max Pool** : (Filter 2x2 & Stride 2x2) - Pout : 5x5x32
    iii. **CONV_LAYER_3** (Input : 5x5x32 - Conv : 1x1x16 - COut : 5x5x16 - stride 1x1)
    iv. **FULLY_CONNECTED_1** (Input:400 Output:300) – Dropout (0.5)
    v. **FULLY_CONNECTED_2** (Input:300 Output:200) – Dropout (0.5)
    vi. **FULLY_CONNECTED_3** (Input:200 Output:100) – Dropout (0.5)
    vii. **FULLY_CONNECTED_4** (Input:100 Output:43)
- Realized the following:
    i. Capture maximum details in the first layer and reduce smoothly along the Network
    ii. Use ReLUs in Convolution Layers to introduce non-linearity & capture maximum data and Dropouts on Fully Connected to make the model more robust and drop unwanted neurons and avoid overfitting
    iii. Use a 1x1 Convolution to connect high dimension Convolution Layer output to Fully Connected layers for better information processing along the network
- **Model Training**: Continued with adamOptimizer, and standard values of Learning Rate, Batch Size and EPOCH. Need to work more on this area to improve efficiency.
- With the above approach reached the desired accuracy but I believe there are more knobs to be tweaked for extracting much more.

Test Model on New Images:

- Acquired 5 new images from the web and tested the model on them, required results are attached in the notebook.

To Do:

- Play around with Data Augmentation (Brightness enhancement, rotation, skews etc.)
- Play around with EPOCH, Batch Size and Learning Rate for more efficient training runs
- Add inception module to the Network and observe its behavior
- Same vs. Valid Padding in Convolutional Layer
- Observe the effect of a wider model vs a longer model
- Try more image samples with different points of view of traffic signs
- Work on Optional Part of Visualizing the Neural Network's stages with images
- Make the implementation more modular so that it can be tweaked around easily, like configuring the shopping list