Project Writeup

Anirudh AV (anirudh.av93@gmail.com)

---

1. Data

1.1. DataSet Summary and Exploration

- Numpy was used to manipulate the data in order to get more insight into the distribution.
- Number of training examples was 34799 and the number of test examples was 12630.
- Plotting the number of examples in each class (the histogram of the dataset) gives us good insight into the characteristics of the dataset.
- This was done using the np.unique function in python to get the different classes and then the examples were put into the respective bins.
- The reason this was done was to see if there was a vast difference in the distribution and additional data needs to be generated.
- Additional data was not generated to preserve the independence of the data.

1.2. Data Pre Processing

- The data was normalized and then converted to gray scale.
- Numpy was used to perform normalization and grayscale as the images were not cv2 Mat images.
- After preprocessing the data was aimed to be zero mean (-0.3 on average of all runs)
- Now the images had only one channel and were zero mean as a result of pre-processing.

2. Model Architecture

Layer	Description
Input	32X32X1
Convolution	valid padding, 1X1 strides
RELU	Activation
Max Pooling	valid padding, 2X2 strides
Convolution	valid padding, 1X1 strides
RELU	Activation
Max Pooling	valid padding, 2X2 strides
Flatten	Output = 400
Fully Connected	Input 400 , Output 120
RELU	Activation
Dropout	To Avoid Overfitting
Fully Connected	Input 120 , Output 84
RELU	Activation
Dropout	To Avoid Overfitting
Fully Connected	Input 84 , Output 43
RELU	Activation
Dropout	To Avoid Overfitting

Table 1: Model Architecture Layers

3. Model Training

- The learning rate of the optimizer was 0.001 to ensure it does not diverge.
- The loss function used was cross entropy which was calculated using the $tf.nn.softmax\_cross\_entropy\_with\_logits$ function
- The Adam optimizer was used which implements the Adam algorithm.
- The number of Epochs chosen was 15
- The batch size chosen was 128
- A dropout rate of 0.5 was chosen.

4. Model Performance

- The train and test split function was used to split the data into training and validation.
- A validation accuracy of 0.97 was obtained.
- The learning rate and droput rate were tuned so that the accuracy does not diverge after fining the optimum.
- The batch size remained at 128
- The LeNet architecture was chosen for the classification, A slight modification was made so that it accepts grayScale images and outputs 43 classes.
- This was the suggestion in the course and since CNNs are known to be best suited for image classification applications, that was finally chosen.
- A test accuracy of 0.901 was obtained

5. Testing model on new images

- The pillow and glob namespaces were used to load and display the images from the directory.
- The os namespace was used to load the images for processing and the images were resized to 32X32X3
- The images were then pre processed just like the training and test data above.
- The same model above was run on these new images and an accuracy of 40% was observed.
- The low accuracy can be attributed to the quality of the images as they had to be severely downsized and the were not centered and rotated.
- The top 5 soft_max probabilities for all the images are displayed using the $tf.nn.top\_k$ function in tensorFlow.