Traffic Sign Classifier Using LeNet Architecture

RUBRIC:

1. **Dataset Exploration**
   - The submission includes a basic summary of the data set.
   - The submission includes an exploratory visualization on the dataset.
2. **Design and Test a Model Architecture**
   - The submission describes the preprocessing techniques used and why these techniques were chosen.
   - 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.
   - The submission describes how the model was trained by discussing what optimizer was used, batch size, number of epochs and values for hyperparameters.
   - The submission describes the approach to finding a solution. Accuracy on the validation set is 0.93 or greater.
3. **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.
   - 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.
   - 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.

PROJECT DESCRIPTION:

1. **Dataset Exploration:** German traffic sign dataset consists of 32x32 color images and 43 classes. There are 34799 training images,12630 test images and 4410 validation images. Figure 1 shows the distribution of classes for the training, test, and validation sets. There is clearly an imbalance in the amount of data for each class. Figure 2 shows 10 example images for 10 random classes. The first column shows the mean image for the class.

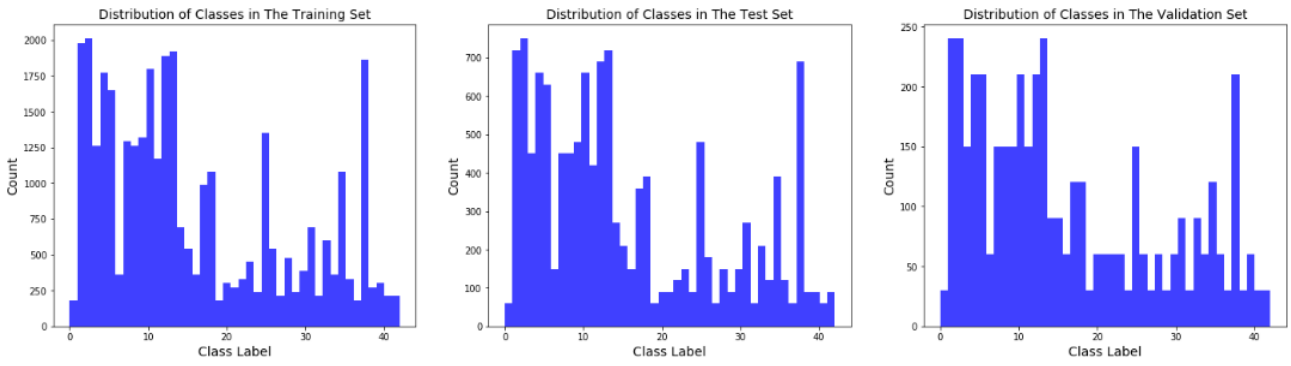

Figure 1:Distribution of classes in the training, test and validation sets

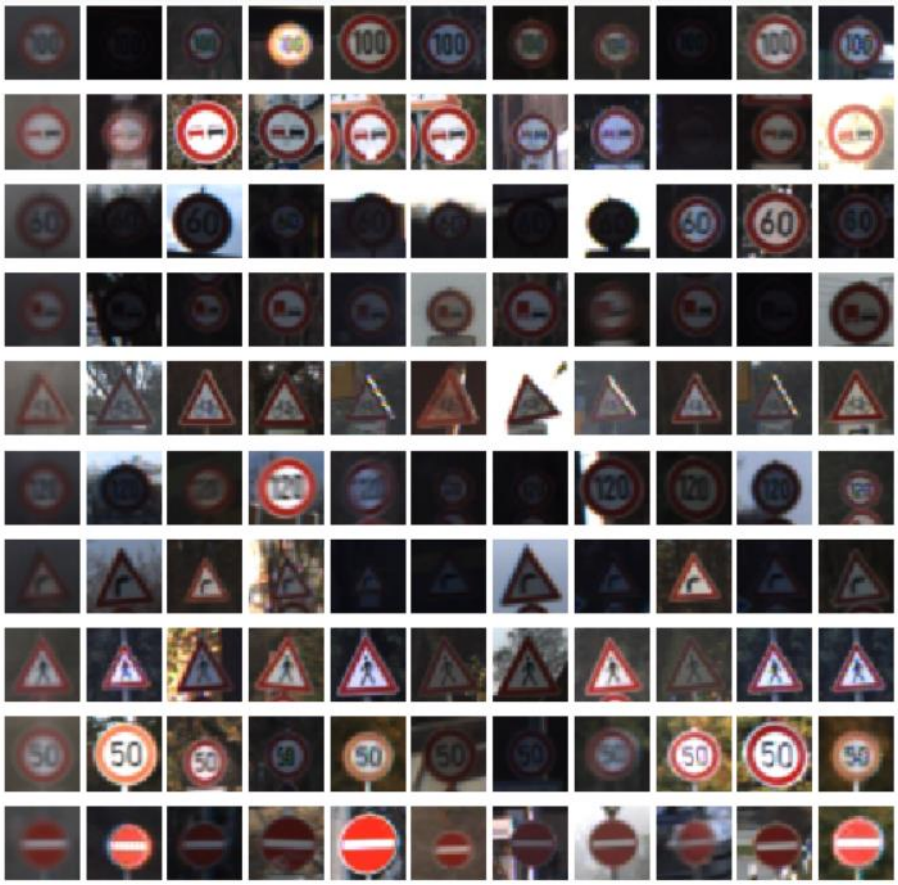

Figure 2: Sample images from the dataset

2. **Model Architecture:** This project uses LeNet architecture, which has historically been a popular network for the MNIST data set.

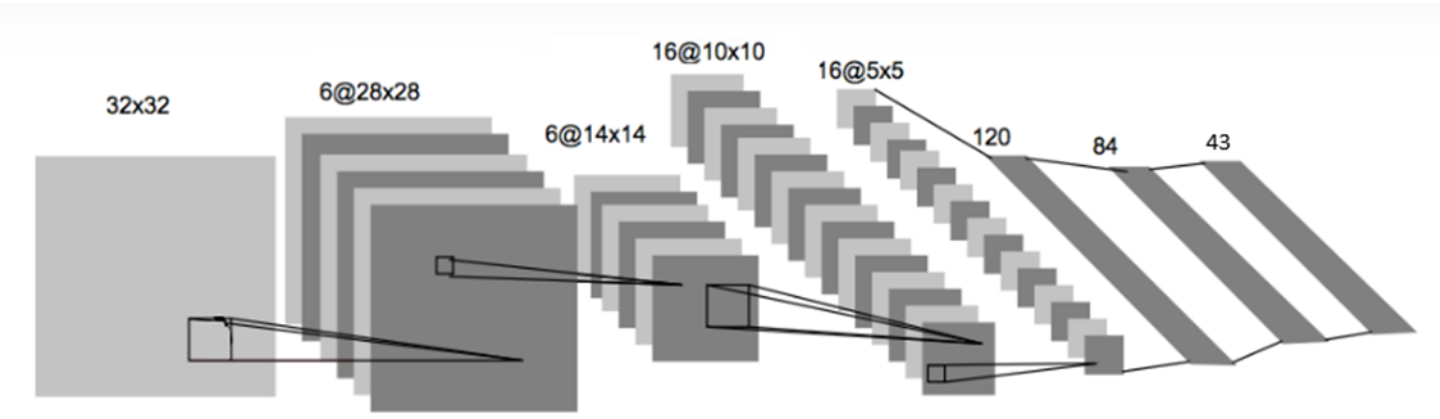

Figure 3: LeNet Architecture

Data normalization: I originally simply normalized the dataset using *normalize_dataset(X_train)* in Cell 6, and used learning rate = 0.001, EPOCHS = 30, BATCH_SIZE = 128. I also normalized the validation set. The network didn't do so well -maximum validation accuracy was around 80%.

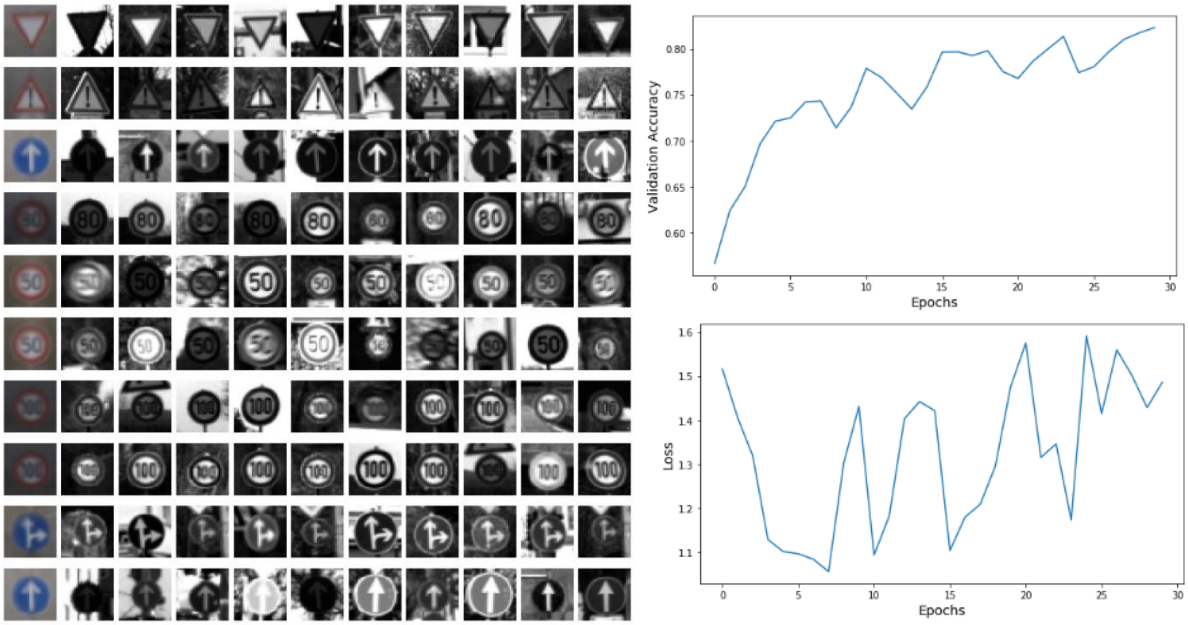

Figure 4: Normalized dataset

Histogram Equalization: In the normalized dataset there is still substantial variation in brightness of the images. I applied histogram equalization to each channel before normalizing the image using: *normalize_dataset(preprocess_dataset(X_train, option='color'))* . The network did better but still not close to 90%.

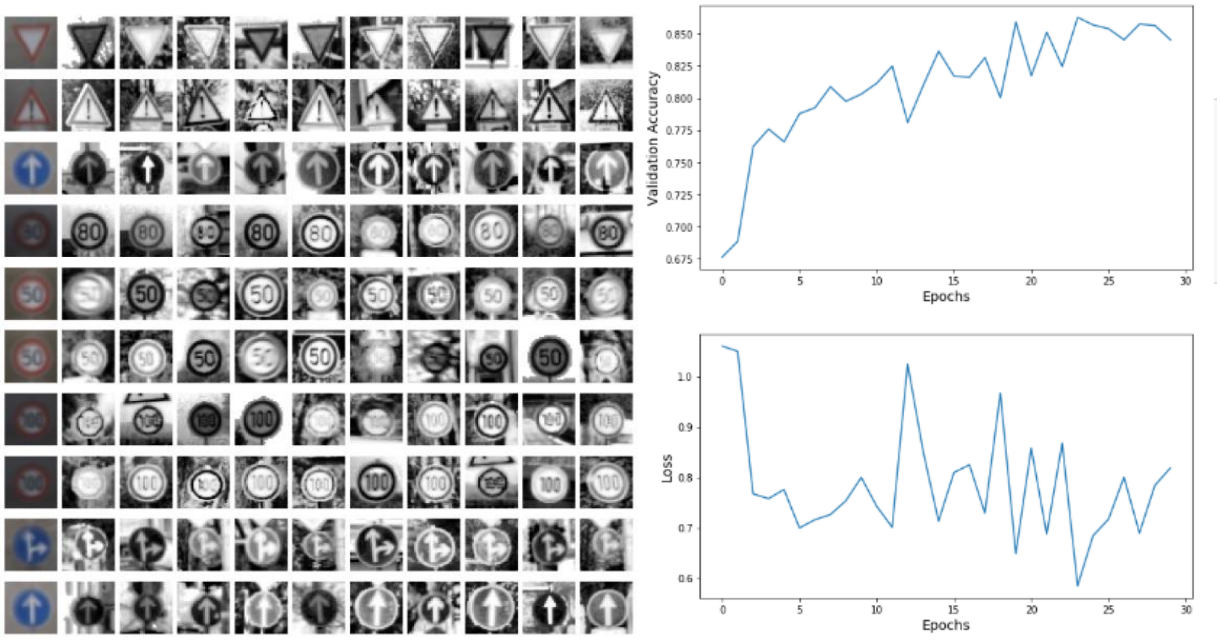

Figure 5: Histogram equalization preprocessing step

Data Augmentation: I created new images for classes which have less than the average number of images per class using scaling, translation and warping operations implemented in [https://github.com/jeremy-shannon/CarND-Traffic-Sign-Classifier-Project]. The size of the augmented training set is 46714.

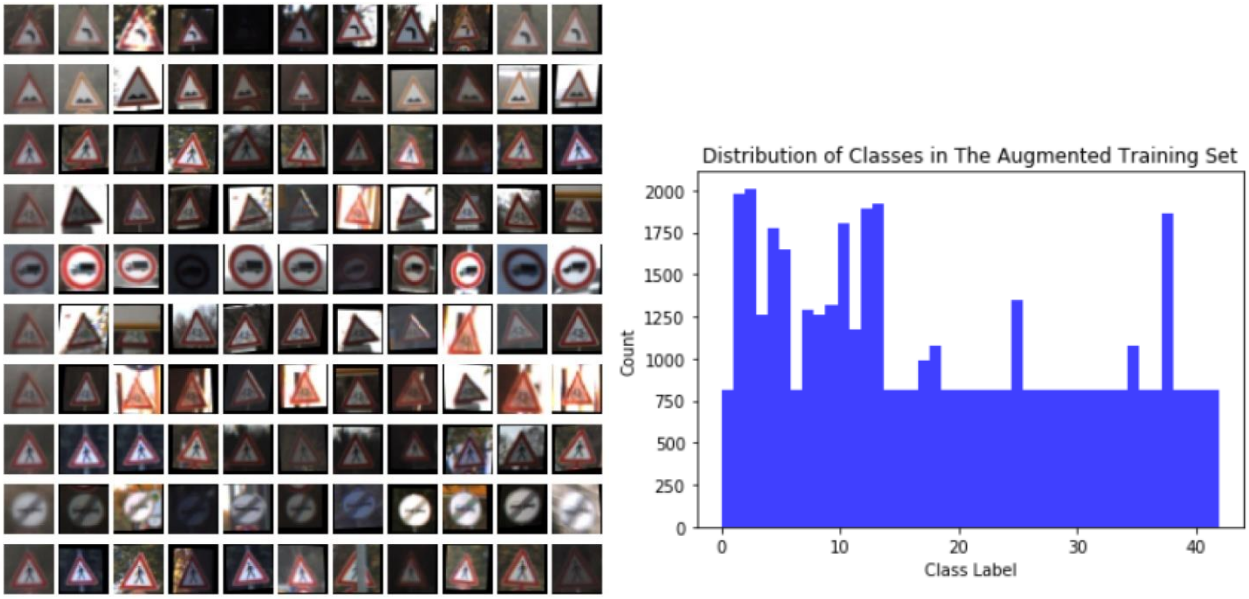

Figure 6: Augmented data and the new class distribution for the training set

I also converted images to a single-Y channel before applying histogram equalization as suggested in [https://mc.ai/traffic-sign-recognition/]. The final set of preprocessing steps are in Cell 7 and can be applied using the function *preprocess_dataset_new(X_train).* Note that, test and validation sets also need to be preprocessed.

Figure 7: Histogram equalization of Y-channel.

The validation accuracy increased to around 95%.

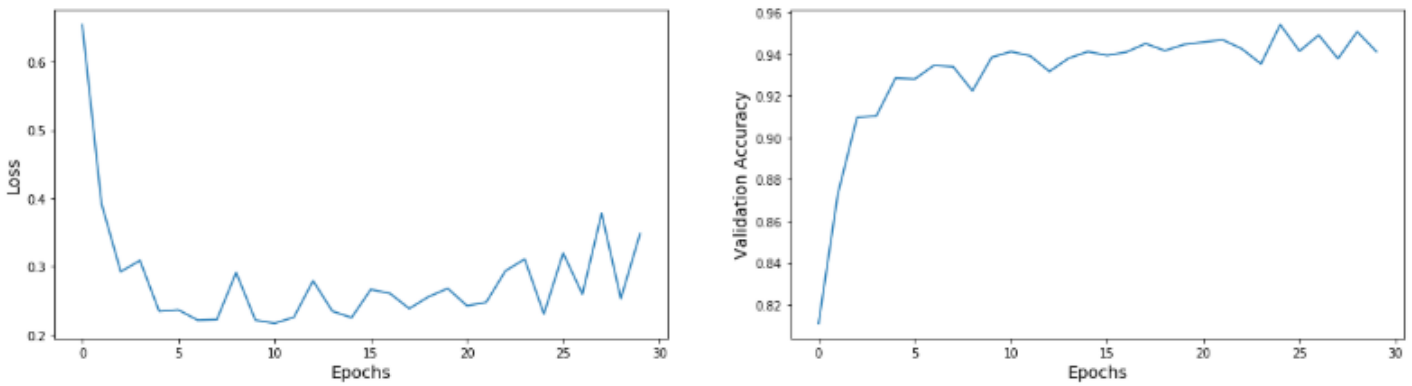

Figure 8: Validation accuracy after augmenting the training set

Finally, I added dropouts after the fully connecting layers with a 0.8 probability of keeping the weights. The final architecture looks like this: Conv → RELU → Max-Pool→ Conv→ RELU → Max-Pool → Flatten→ Fully Connected → RELU → Dropout→ Fully Connected → RELU→ Dropout→ Logits. After training, the validation accuracy increased close to **98%.** Voila!

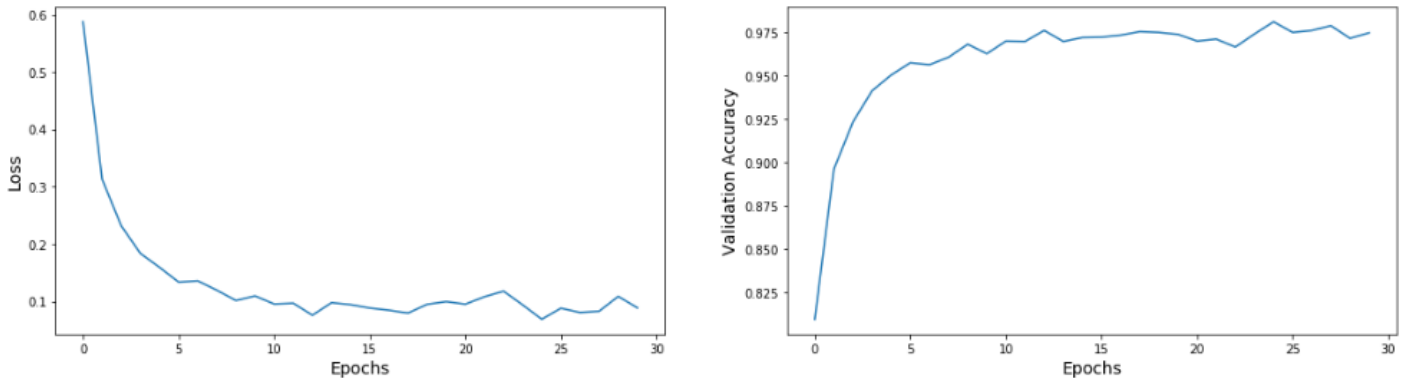

Figure 9 : Validation accuracy after adding dropouts.

Increasing the validation accuracy further will require a larger or deeper network architecture. When I tested the final network on the test images, I got a test accuracy of **95.7%.** I'll take it.

3. ***Test Model on New Images:*** I found the following 5 images on the web to test the accuracy of the network. The second image on the left does not belong to any class in the training set. I was curious to see what the network would predict.

Figure 10: Test images found online

```
Test Accuracy = 80.000%
Class:9 - Prediction: 9
Class:44 - Prediction: 23
Class:12 - Prediction: 12
Class:11 - Prediction: 11
Class:36 - Prediction: 36
```

The test accuracy is lower than the accuracy of the test set as expected -- the second test image is not part of the original dataset and was predicted as “slippery road” sign.

When I tried different sets of new images, I got varying results between 80-100%. The figure below shows the top 5 softmax probabilities for each image found on the web. It’s interesting to note that the prediction for the second image is somewhat close, in the sense that, there is a -45-degree object at the center of the image.