# Bosch's Traffic Sign Recognition System

**25th March 2021**

## OVERVIEW

Bosch Traffic Sign Recognition System (BTSRS) provides a tool that provides a toolset for the entire ML Model Lifecycle from dataset curation to model explainability, integrated into a seamless UI.

## GOALS

1. Create and benchmark a classification system against the German Traffic Sign Recognition Benchmark Dataset.
2. Design and develop a UI that assists with pre- and post-training tasks.

## DIFFERENTIATORS

### Weather Based Augmentation

Real world applications of Traffic Sign Recognition Systems involve handling inputs with interference from weather phenomenon. To make the recognition system more resilient, we have augmented the images with weather effects such as fog, rain and snow.

### Workspaces with Pre-curated Datasets

The tool borrows off years of UX research and introduces the concept of Workspaces which bring together models and datasets. What is easier for dataset curation than having pre-curated datasets available? We provide pre-curated datasets that can be imported into your Workspace directly.

## COMPLETE SPECIFICATION

### Augmentation

- Standards Augmentations applied are:
    - Shearing, Translating, Image Jittering in terms of Brightness, Hue, Saturation, Random Contrast, Random Flipping, Random Brightness, Correct Exposure, Lightness and Speeding, to the training and test dataset.
- Apart from these weather augmentations are also applied which include :
    - Rain, Snow, Fog, Autumn.
- Images were Rescaled to (45,45) and Normalized after each augmentation.

### Model

- The Network Architecture has 3 Convolutional Layers,2 Fully connected Layers , 1 spatial transformer network layer.On each output of Convolutional Layer, Leaky ReLU activation function, Max Pooling Layers and Batch Normalization is applied.
- Dropout is used to reduce the density of the Fully connected Layers.
- The CNN feature maps [3 -> 100 -> 150 -> 250 -> 350],
- The filter size used is [5,3,3],
- Spatial transformer network feature maps: 3 -> 8 -> 10 -> 32,
- Spatial transformer network filter size: [7,5]

### Test train split

- For test train split we have the function “train_test_split” this function accepts the following  arguments
- dataset_dir: which is the absolute path of our dataset
- Test_dir: which is absolute path of the folder which we want our test set to be
- test_set_file_path:  this is a python list containing all the images the user want to be in the testing set
- train_set_file_names:  this is a python list containing all the images the user want to be in the training set
- num_class: which is (integer) the number of classes present in the dataset
- split_ratio: which is (a float) the ratio in which we want to split the dataset
- For example if we want 80 percentage of data in training dataset and 20 percentage in test dataset give split_ratio as 0.20

The function first goes through the dataset and counts the total number of images in the dataset. Then calculate the number of training and test images according to the split ratio. Then it moves the files in the *test_set_file_path* to the test folder. Then the function calculates the number of more files to be moved to the test set. Then it moves the required number of files. Before moving the function just ensures  that the files in the list  *train_set_file_names* will remain in the dataset.

## Dataset addition:

- The dataset to be added will have to be in a certain format, each class has unique id corresponding to it , if in case user adds a new unique class not common to the classes already present a separate unique id will be created for it
- Here's the link to classes and their ids: classid info
- The data from classes common to GTSRB will simply be appended separate folders with unique ids will be added for uncommon dataset
- One can also add multiple datasets on the same dataset

## Interpretability:

- **Guided Grad CAM**: Guided Grad CAM combines the best of Grad CAM, which is class-discriminative and localizes relevant image regions, and Guided Backpropagation, which visualizes gradients with respect to the image where negative gradients are set to zero to highlight important pixels in the image when backpropagating through ReLU layers.
  Thus we use Guided Grad CAM to give the user an idea of which parts of the image contributed most to the prediction it made, whether right or wrong. In the case of a wrong prediction, the user has the option to compare the Guided Grad CAM results for both the correct and wrong classes and see what parts of the images caused the current model to classify it wrongly.
- **Class-wise Inaccuracy plots**: The user is provided with plots showcasing the number of misclassified images per class thereby informing him about where the model's weaknesses lie. Accordingly, more images can be uploaded for those classes on which the current model performs weakly.
- **Shapley Gradients:** Gradient SHAP is a gradient method to compute SHAP values, which are based on Shapley values proposed in co-operative game theory. Gradient SHAP adds Gaussian noise to each input sample multiple times, selects a random point along the path between baseline and input, and computes the gradient of outputs with respect to those selected random points. The final SHAP values represent the expected value of gradients * (inputs - baselines). (This method approximates Shapley values of a given prediction by examining the effect of removing a feature under all possible combinations of presence or absence of the other features.) This is a method to interpret the model predictions on a given image from the perspective of optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions.