BOSCH's

TRAFFIC SIGN RECOGNITION

H1_BSC_16

The Benchmark Model

01

MicronNET

Highly compact DeepCNN designed specifically for real-time embedded traffic sign recognition.

Numerically optimized convolutional layer microarchitecture.

As few parameters and computations possible, without compromising on performance.

Based on macroarchitecture design strategies that encourage improved computational efficiency and efficacy in embedded environments.

Why MicronNET?

1.32%

Of the size of MCDNN

Such a small size makes it ideal for deployment in Self-Driving Cars.

98.9%

Accuracy on GTSRB

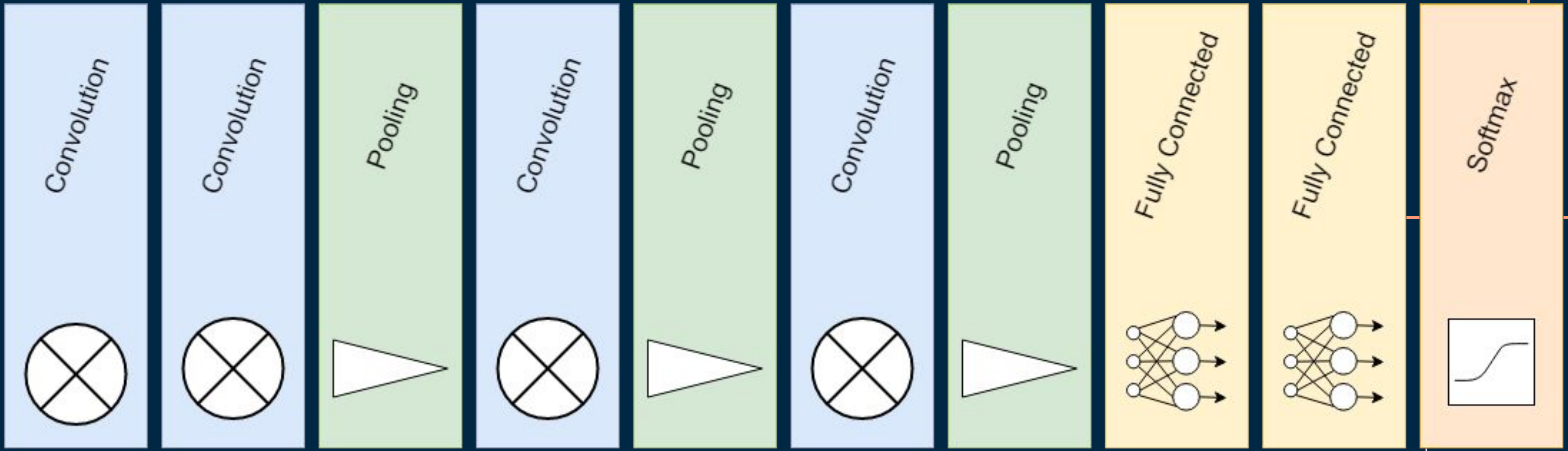
Proven to have Top 1 Accuracy on GTSRB.

102.5

Net Score

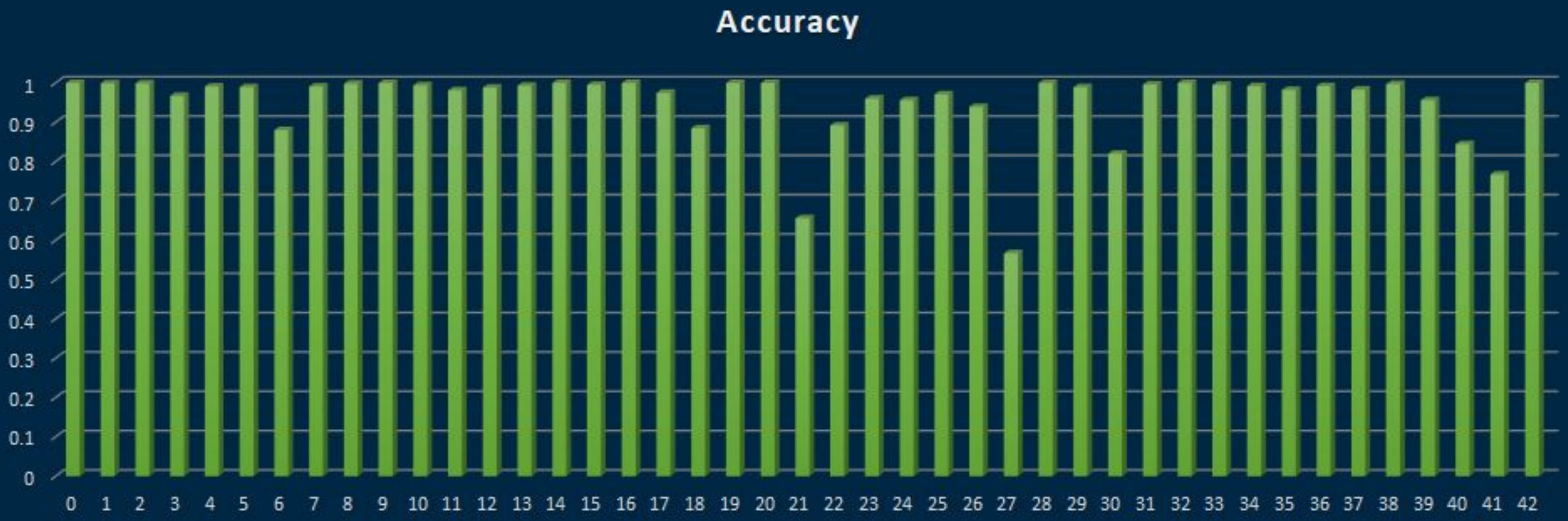
Shows the strong balance between accuracy, architecture complexity, and cost.

MicronNET



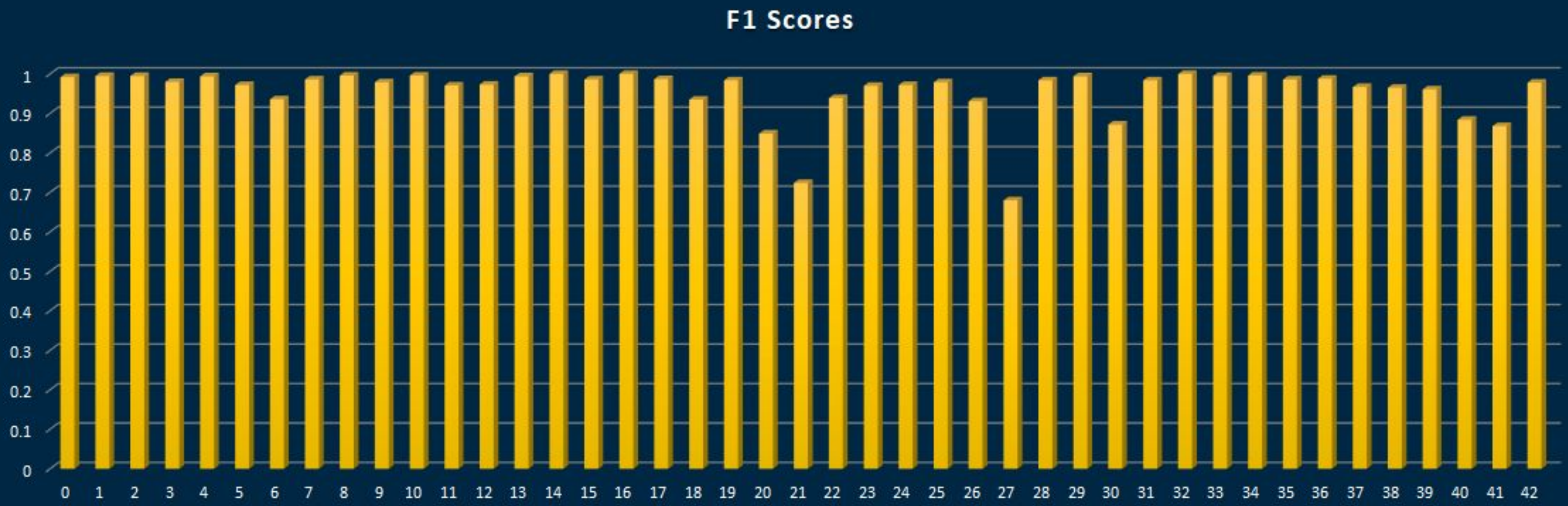
Model Architecture

BENCHMARK METRICS: Accuracy



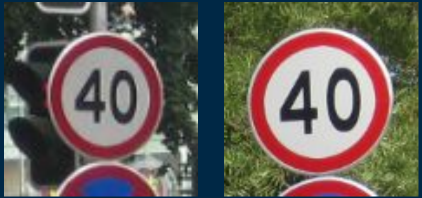
Overall Accuracy: 97.53%

BENCHMARK METRICS: F1 Scores

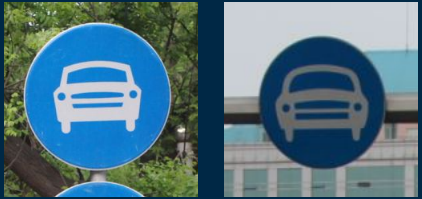


Overall F1 Score: 0.956

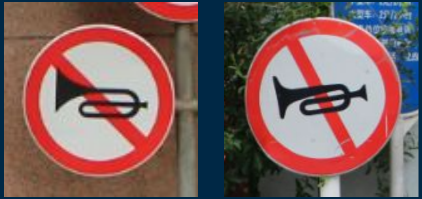
The 5 Classes Introduced



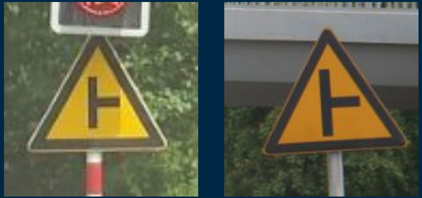
Maximum Speed 40



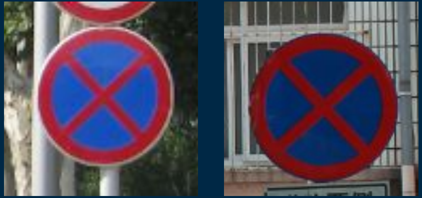
Limited Access Road



No Honking



Side Road Jn. on Right



No Stopping

Source: TSRD Dataset

Increasing Dataset Difficulty

02

Making the dataset more difficult

Dataset

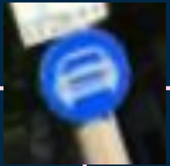
Vanilla GTSRB Dataset

Original Classes, No Changes

DatasetAug

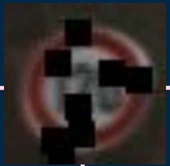
5 classes added from TSRD dataset

Images from the common classes of TSRD and GTSRB were also added



DatasetDiff

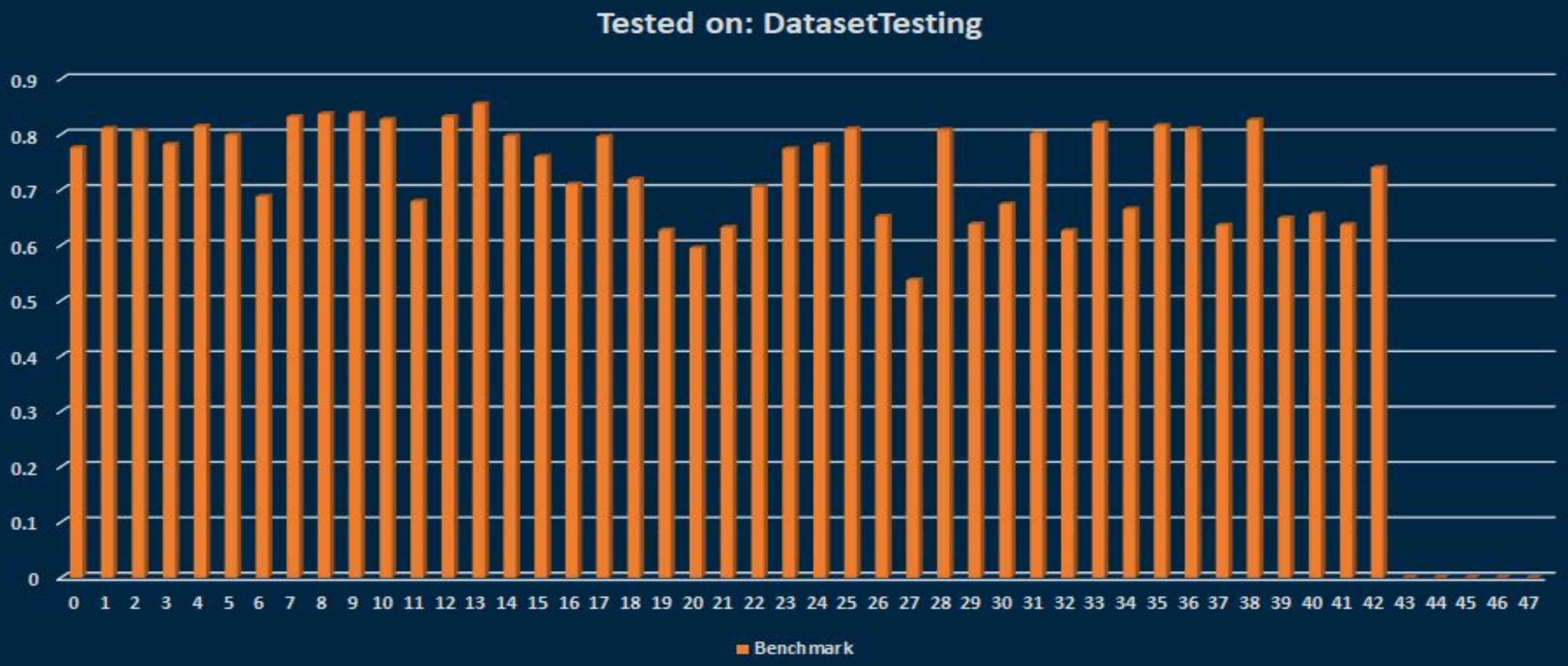
Several transformations applied to all images using several transforms with varying probabilities.



DatasetTesting

Around 4000 images were added with similar transforms of even higher probabilities to increase the number of augmented images.

METRICS: DatasetTesting



Overall F1 Score: 0.685

03

Interpreting Poor Performance

Why is the network failing in particular places?

01 New Classes

02 Augmentations

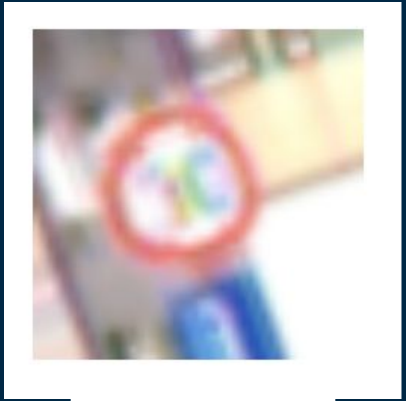
03 Original Misclassifications

Sources of Inference:

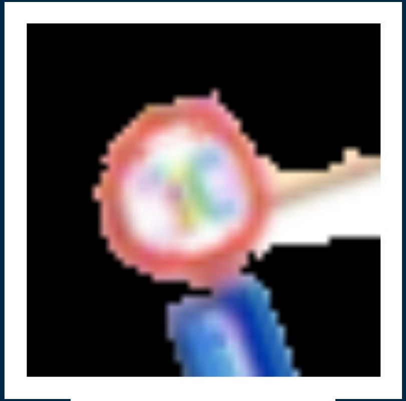
1. F1 scores and Evaluation Metrics
2. Misclassified Images
3. Model Interpretability Features

Anchor Explanations

- Used Anchor method to generate superpixels of the original misclassified image to explain the misclassifications



Original Image



Generated Anchor

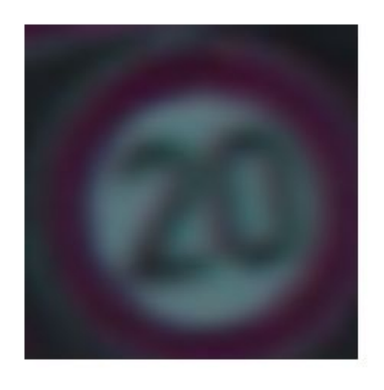
- These superpixels helps in making inferences about the images where the classifier can misclassify and can be used to modify the dataset or the model to improve performance

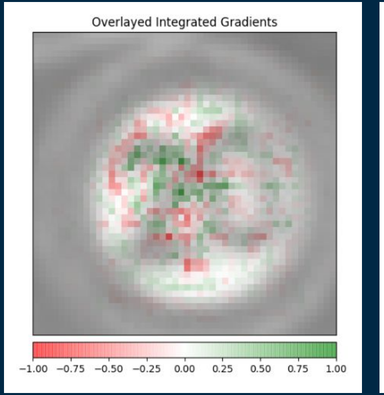
Integrated Gradients

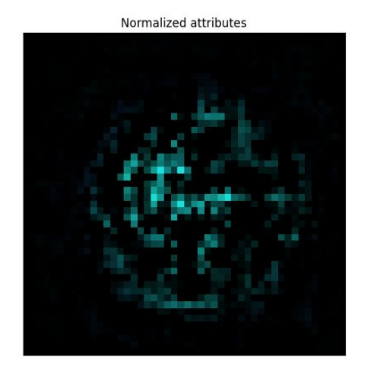
- Integrated gradients were calculated for each feature with respect to the original class as follows:

$$\text{IntegratedGrads}_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i}\, d\alpha$$

- Generated a heat map highlighting the pixels in favour of the class as shown
- Heat maps help in understanding the reason for misclassification
- This information could be used to augment the dataset for better performance







Changing The Network

04

In-place Randomized Augmentations

- Change the training dataset by applying transforms to it and create a new dataset to handle augmentations.

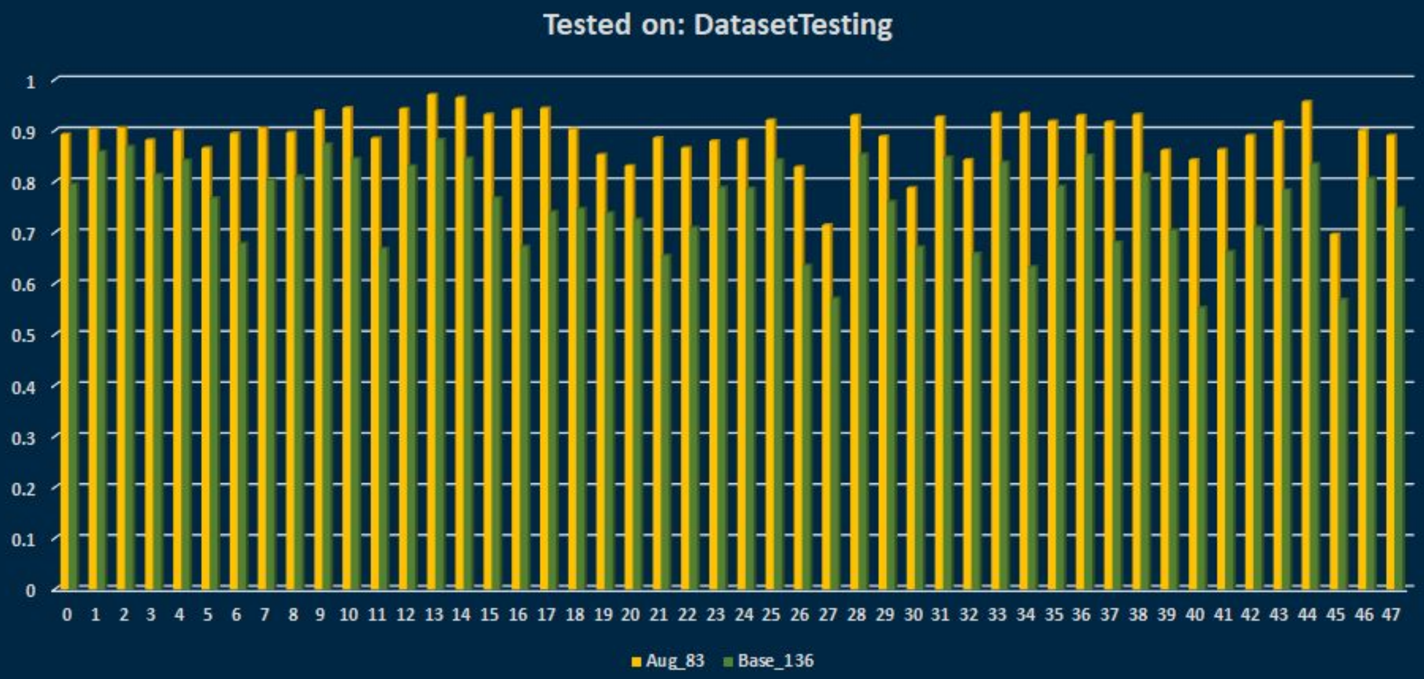
- Storing such a large dataset while new images are added is a problem.

- Integrate the ability of applying transforms to the training images in the model itself.

- Random transformations are applied to the entire training and validation set and as a result, the model almost never sees the same image twice as well as is able to capture a large number of image-transform permutations.

- Also improves performance on non-augmented dataset.

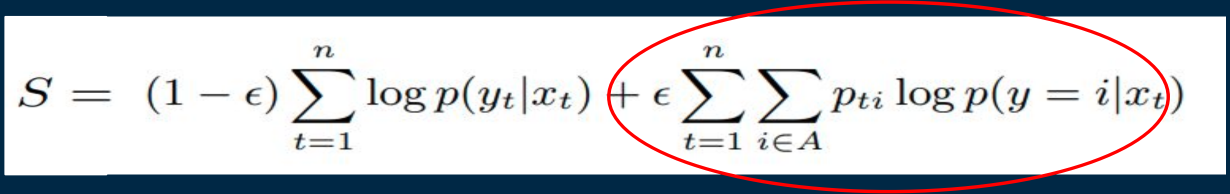
METRICS: DatasetTesting



Overall F1 Score: Aug_83 - 0.893, Base_136 - 0.757

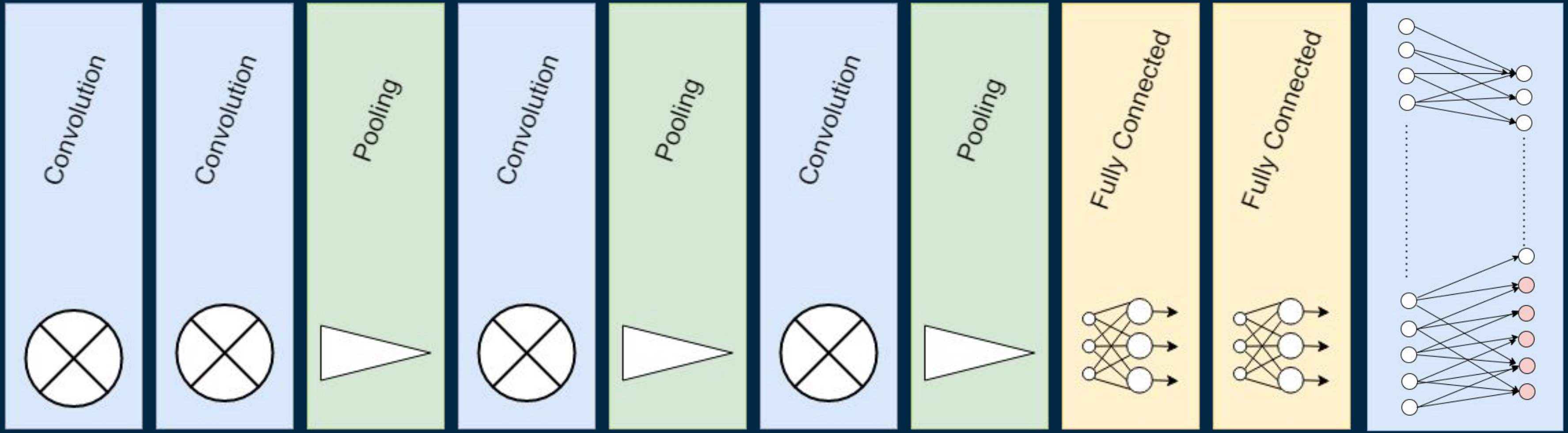
Expanding the Model: Incremental Learning

- Want to train on new classes only without losing information about old classes.
- Need to mitigate “catastrophic forgetting”.
- Add a new weight parameter for each new class in the second FC layer.
- During training, freeze all model parameters except the new weights.
- A modified regularised loss function to dampen forgetting:



$$S = (1-\epsilon)\sum_{t=1}^{n}\log p(y_t|x_t) + \epsilon\sum_{t=1}^{n}\sum_{i\in A}p_{ti}\log p(y=i|x_t))$$

Incremental Learning

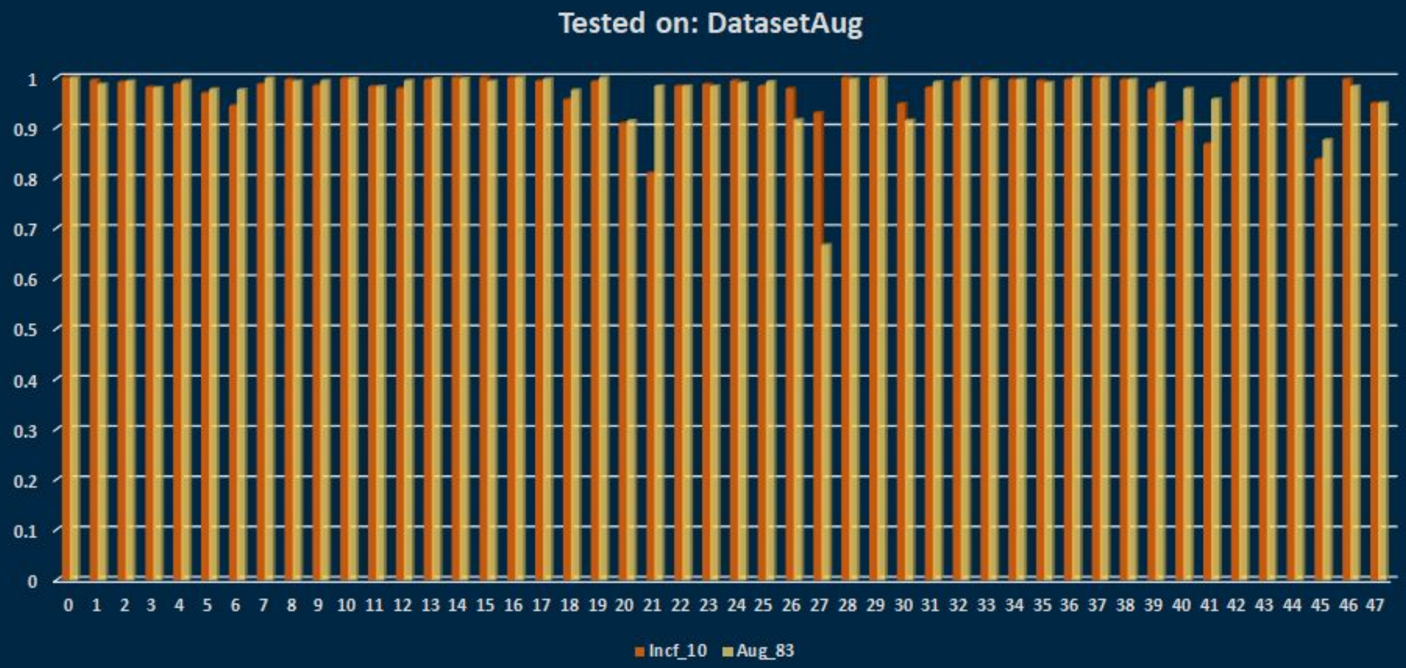


Model Architecture

Why not re-train from scratch?

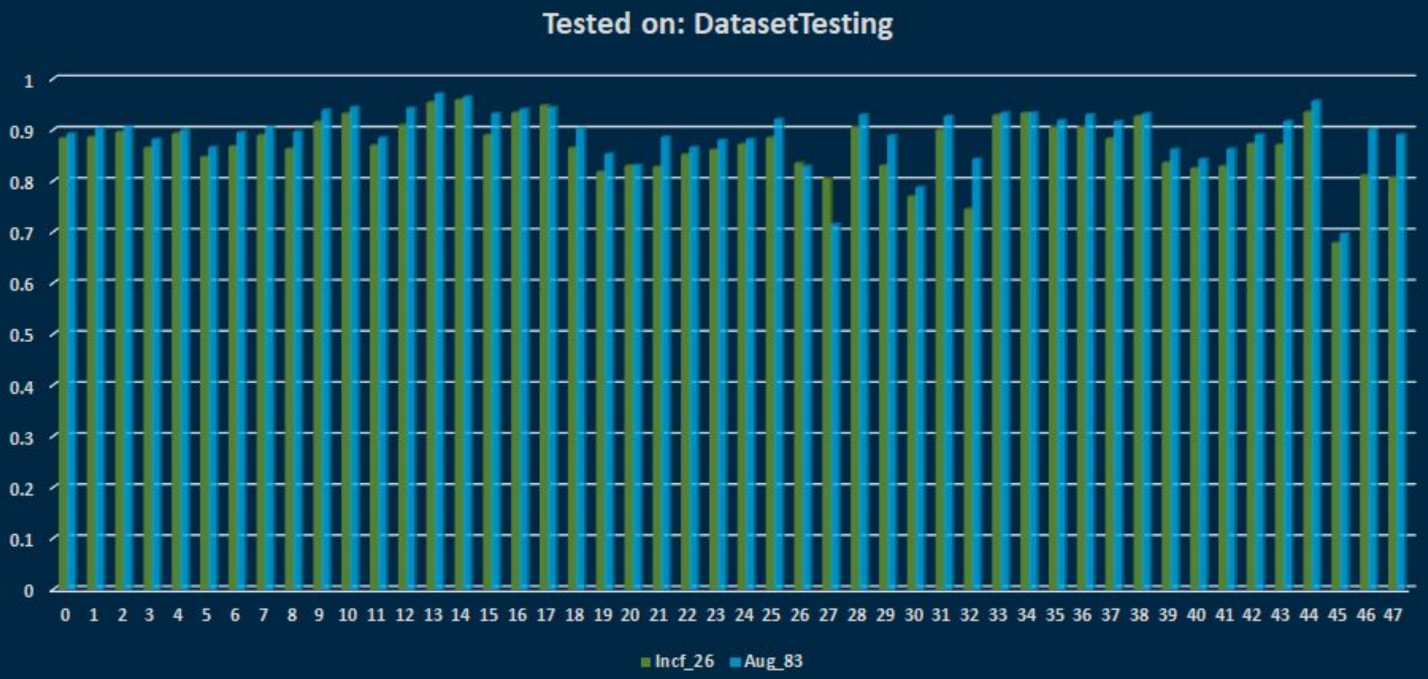
- A very obvious solution - since the model is not performing, retrain it.
- But not viable if the dataset is ever-increasing - does not scale well.
- All the current knowledge gained by the model is also discarded.
- Incremental learning is better - more scalable and faster.
- Also retains current knowledge stored in the model.
- Attains re-train level performance in significantly lesser epochs - more advantageous as dataset size increases.

METRICS: Incremental Learning vs Retraining



Overall F1 Score: Aug_83 - 0.893, Incf_10 - 0.844

METRICS: Incremental Learning vs Retraining



Overall F1 Score: Aug_83 - 0.893, Incf_26 - 0.870

05

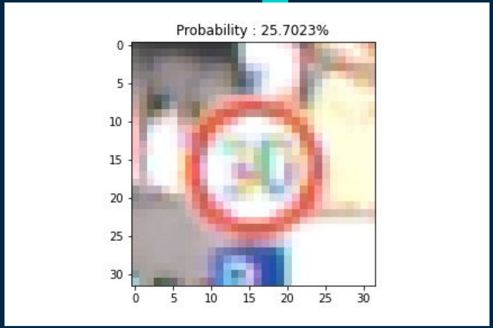
Further Improvements

Activation Maximization

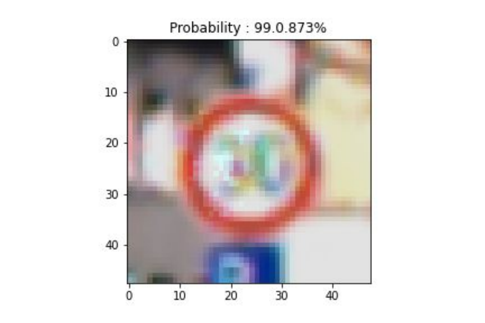
- Used activation maximization method to improve model’s performance.
- Generated images using misclassified images and class representations.
- Searches for an input which produces maximum response from the model by optimizing:

$$\max_{\boldsymbol{x}} \ \log p(\omega_c|\boldsymbol{x}) - \lambda \|\boldsymbol{x}\|^2.$$

- Modifies the pixel values until resulting image has a high probability of belonging to correct class.



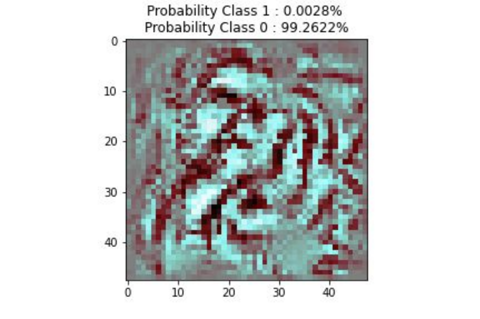
Misclassified Image



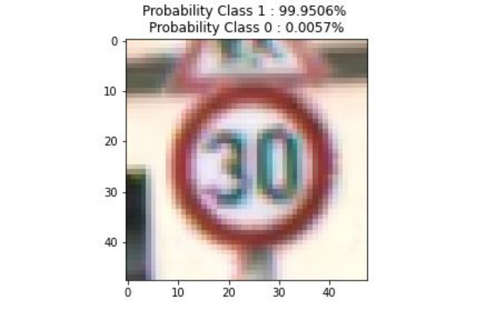
Generated Image

Activation Maximization

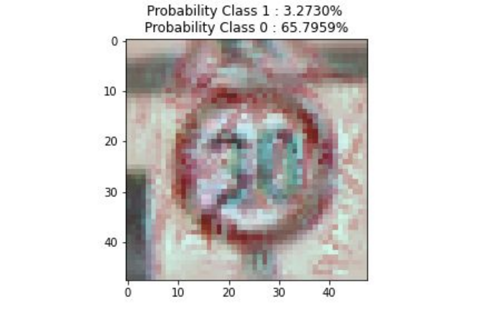
- Generated images did not show much deviation from original image.
- Generated an activation map for class 0 and superimposed it on an image from properly classified image from class 1.



Activation Map



Original Image



Superimposed Image

- Misclassified the superimposed image with high probability.
- This method can be used to improve the dataset instead of generating images around misclassified images.

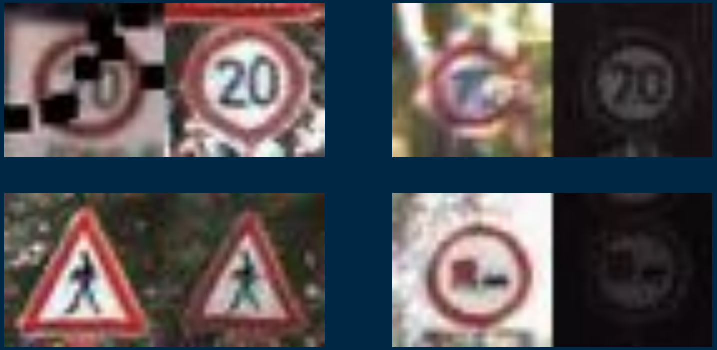
GANs: As a Feedback Loop

- To increase class based model performance, we aimed at generating images similar to misclassified images by utilizing GAN as a generator and retraining on the generated images.

- We used a class conditioned StyleGAN2 with ADA. The motivation for this was the non availability of a very large training corpus which GANs generally require. ADA helps the GAN to generate good quality images with lesser data.

- Even on data points non-classifiable by human intervention, Images generated using StyleGAN2 had considerably higher quality with low FID value.

Target Image
(Present in Dataset)
64px * 64px

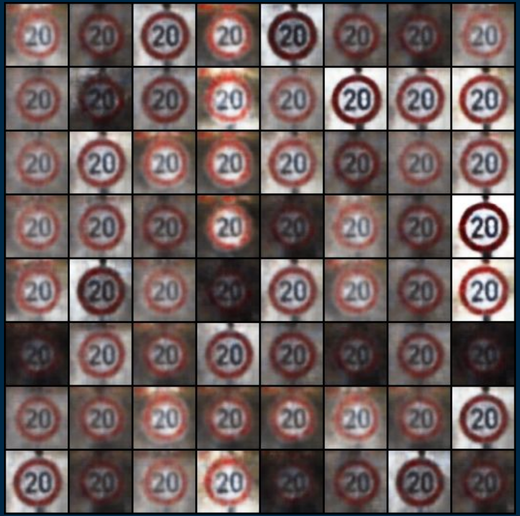


Generated Images
(Video Format)
64px * 64px

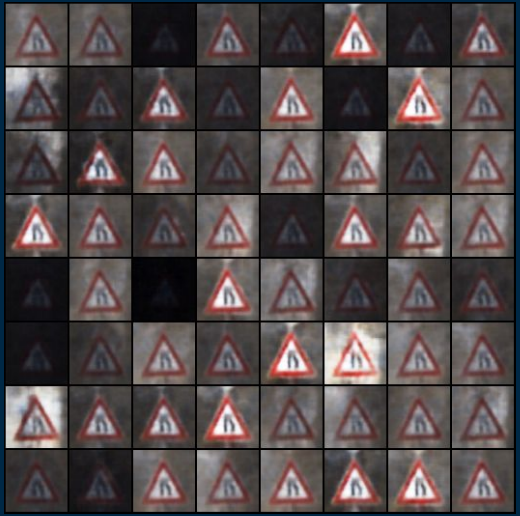
GANs: As a Feedback Loop

- This allows us to get multiple images similar to the original misclassified images on which we can further train the classifier. This led to a individual class based increase in F1 score.

- This approach is independent of the classification model used and can be in a manner of feedback learning on the model.



A Show of Images generated by the full range of vector space.



Out of distribution Detection (OOD)

- OOD or Out of Distribution Detection determines whether the input to the traffic sign classification model belongs to one of the trained classes or not.

- This feature allows us to keep introducing classes beyond the forty-eight that are being used right now.

- We explored three approaches (in upcoming slides):
  - ODIN
  - Outlier Exposure
  - Cluster Entropy

ODIN

- ODIN - Applies pre-processing to input images and temperature scaling to softmax function. This should affect the OOD images more than the in-distribution images.. The temperature scaling is done as follows. T is usually chosen as 1000.

$$S_i(\boldsymbol{x};T) = \frac{\exp\left(f_i(\boldsymbol{x})/T\right)}{\sum_{j=1}^{N}\exp\left(f_j(\boldsymbol{x})/T\right)},$$

- The input pre-processing involves adding small perturbations. The idea is to increase the softmax score of any given input, irrespective of class label. This ideally should affect the in-distribution more but in our implementation it actually affects the out of class traffic signs equally therefore no useful result is obtained.

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \varepsilon \operatorname{sign}(-\nabla_{\boldsymbol{x}} \log S_{\hat{y}}(\boldsymbol{x};T))$$

Outlier Exposure

- Outlier Exposure - Tunes the trained model again by minimizing a cost function so that the model learns to differentiate between in-distribution and OOD images. The custom cost function is designed so that the sign classifying ability is not affected much. The cost function used is:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{in}}}[\mathcal{L}(f(x),y)+\lambda\mathbb{E}_{x'\sim\mathcal{D}_{\text{out}}^{\text{OE}}}[\mathcal{L}_{\text{OE}}(f(x'),f(x),y)]]$$

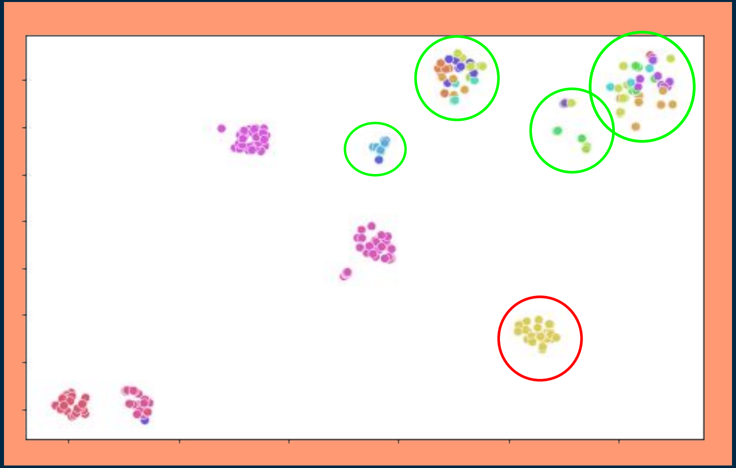
- Unfortunately, this is more suited to differentiating between a traffic sign and an animal, not good enough for determining whether the traffic sign is in distribution.

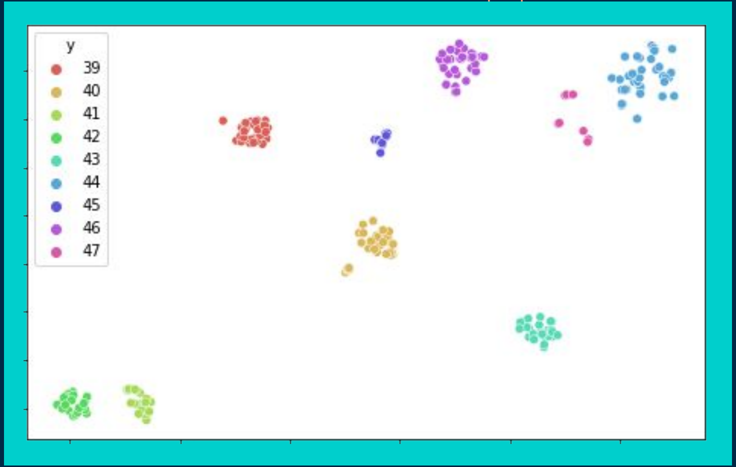
Cluster Entropy

Actual class representations

Can help determine new classes in data.

Feature based clustering puts images having similar attributes in same cluster.





Representation using model predicted labels.

The clusters having high value of entropy indicate new classes.

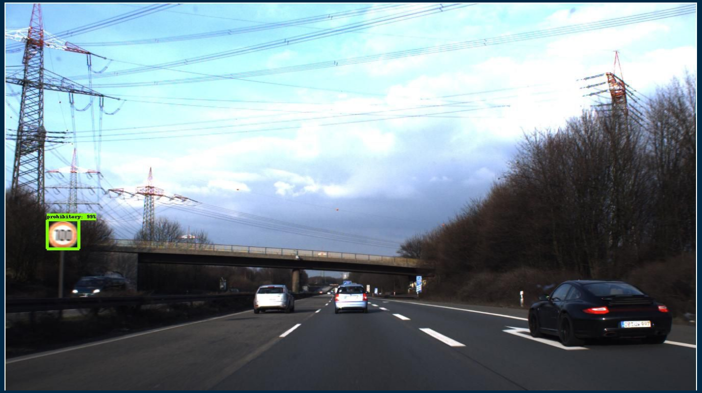
$$H(\omega) = -\sum_{c \in C} P(\omega_c) \log_2 P(\omega_c)$$

SIGN Detection

- We plan to use a pre-trained YOLO model trained on GTSDB images to detect traffic signs from the street view images captured. This will allow us to continuously increase our database as time passes.

- This'll also help us in improve our models by adding images to pre-existing classes and introducing new classes.

- Model in Action:



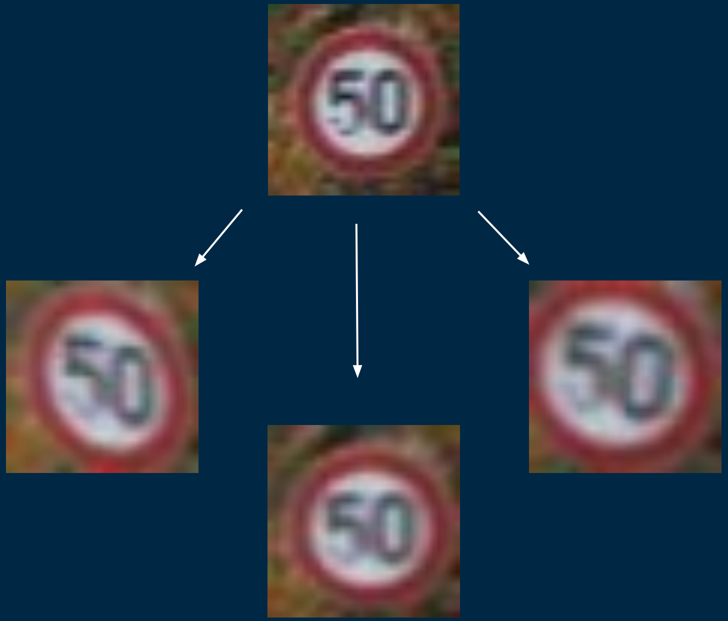
06

UI Demo

Transformations & Augmentations

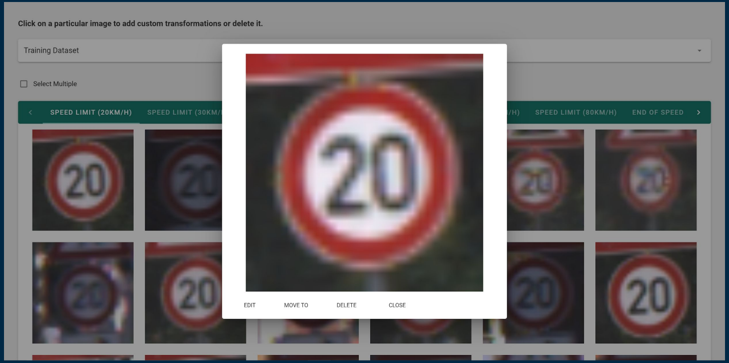
Select one or more (combine and permute) of the following augmentations with full control over parameters, and generate more images:

- Brightness & Contrast
- Shift & Rotate
- Blur (Gaussian, Median and Motion) & Optical Distortion
- Noise (ISO, Gaussian and Multiplicative)
- Hue, Saturation & Color Jitter
- Dropout & Cutout
- Affine & Perspective Transforms



Control Over Additional Images

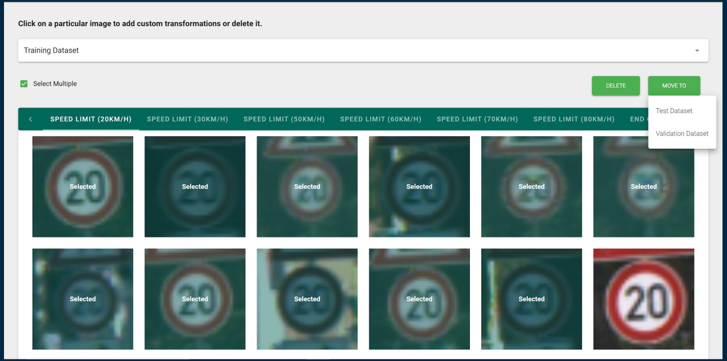
- Visualize each image added through the UI.
- Move images between train, validation and test set.
- Delete unsatisfactory images from the dataset.



Click on a particular image to add custom transformations or delete it.

Training Dataset

□ Select Multiple

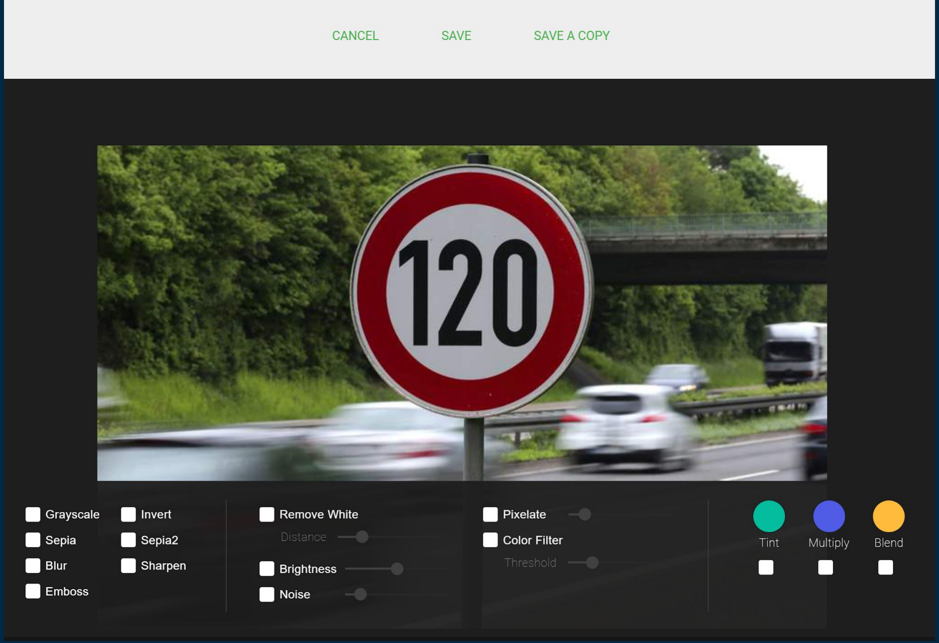


Click on a particular image to add custom transformations or delete it.

Training Dataset

☑ Select Multiple

Inbuilt Image Editor

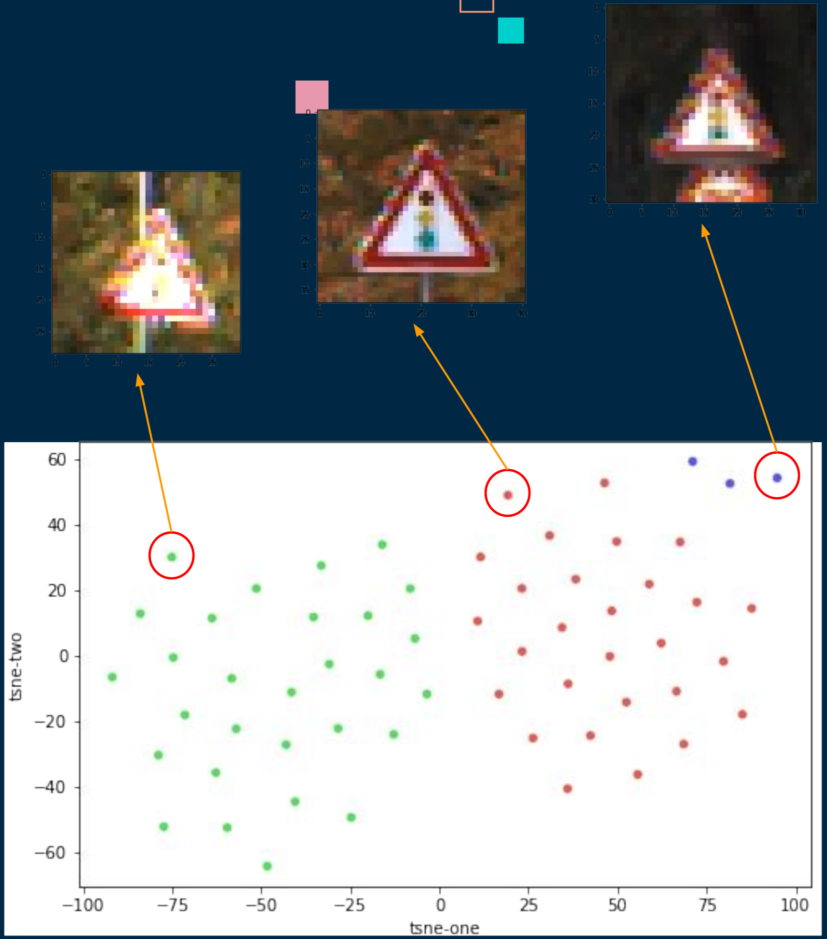


- Manually select and edit images according to user choice.

- The editor has all the basic features like crop, rotate, flip, blur, etc.

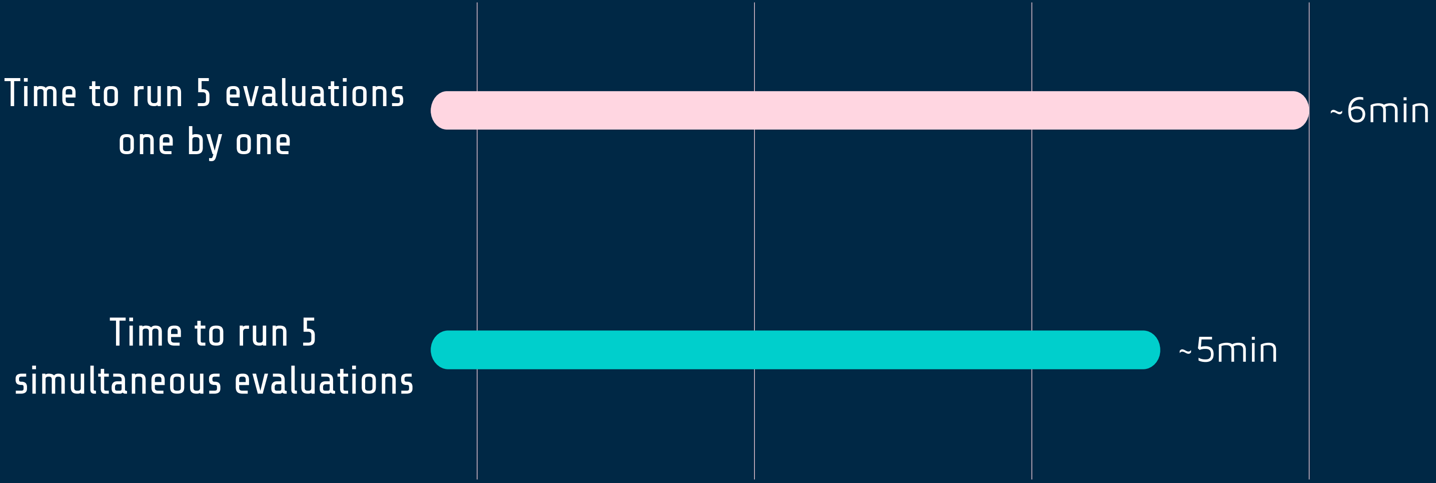
Smart Segregation

- Extract image features and using them to form a suitable number of clusters.
- Each cluster is a set of similar images.
- A uniform split can be achieved by dividing every cluster into training and validation sets in a fixed ratio.



Parallel Evaluations

- User can run upto 5 simultaneous evaluations for models trained using the UI.
- Enables the user to compare the performance of models on different datasets in lesser time.



(Benchmark model evaluated on GTSRB dataset)

Thank You.

Questions?
We will try our best to
answer them!