CSE 5524 Final Project: Traffic Sign Classification

By Wenjin Fu, Haidong Tian, Octavian Donca

I. Exploratory Data Analysis

The data set used for the final project is *German Traffic Sign Recognition Benchmark (GTSRB)* from Kaggle (*https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign*). There are 39219 images in training set and 12630 images in test set. The location of the bounding box is given by the coordinates of its upper left and lower right corners within an image. The goal is to classify images into 43 classes of traffic signs. Some sample images from the data set are shown in Fig. 1.

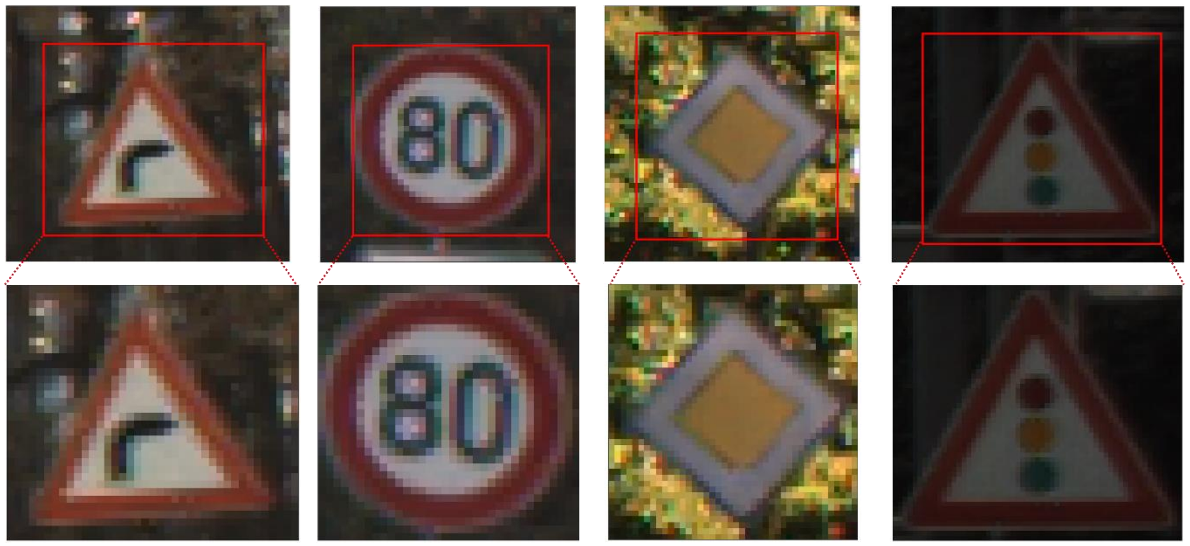


Fig. 1. sample images from the data set

II. Template Matching

In addition to the training images and the test images, templates for each class are as well provided in the data set. Some sample templates are shown in Fig. 2.

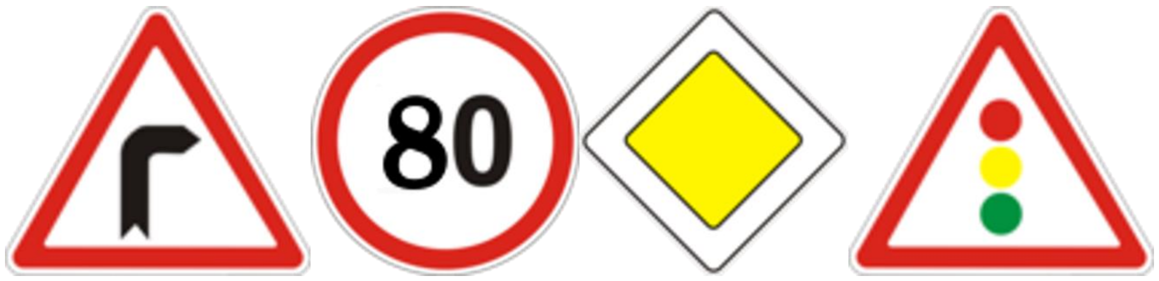


Fig. 2. sample templates from the data set

With the templates provided, Sum-of-Absolute Differences (SAD), Sum-of-Squared Differences (SSD) and Normalized Cross-Correlation (NCC) are used to classify the test images.

The test results for SAD are shown in Fig. 3, along with the classes best and worst classified in terms of accuracy. An overall test accuracy of 0.1652 is achieved. 16 classes are not recognized at all (zero accuracy).

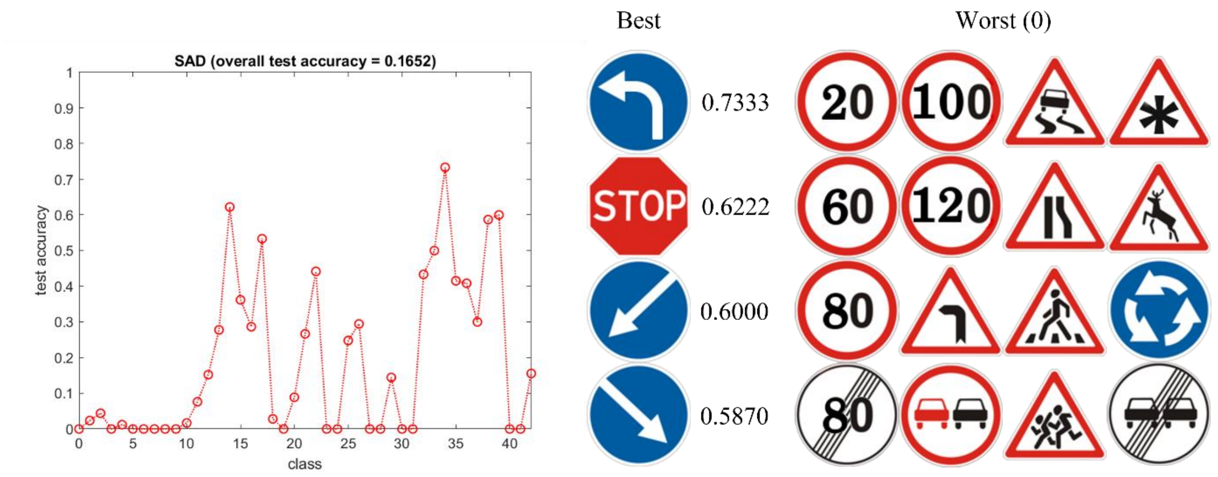


Fig. 3. test results for SAD

The test results for SSD are shown in Fig. 4, along with the classes best and worst classified in terms of accuracy. An overall test accuracy of 0.1804 is achieved. 13 classes are not recognized at all (zero accuracy).

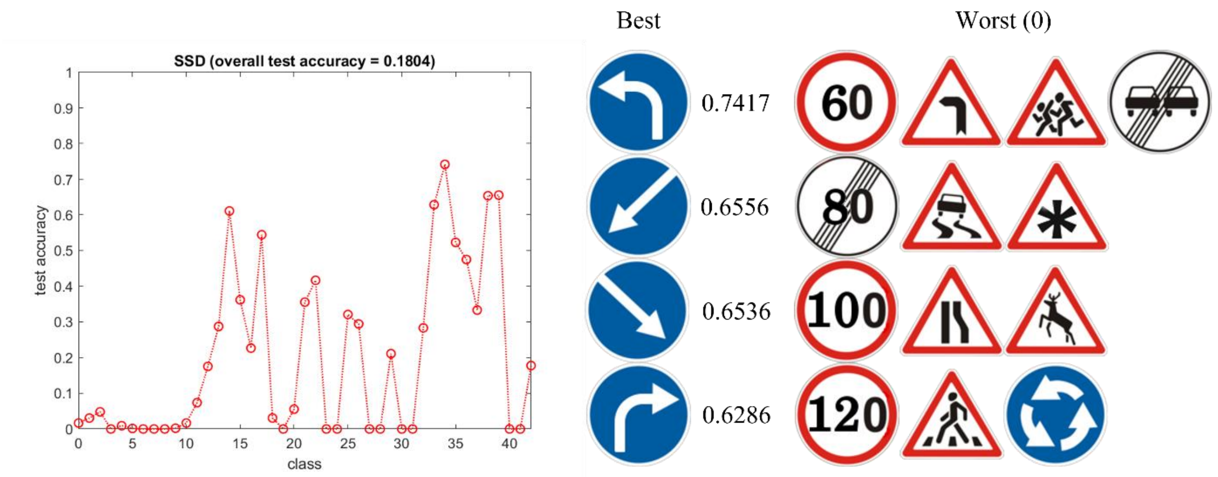


Fig. 4. test results for SSD

The test results for NCC are shown in Fig. 5, along with the classes best and worst classified in terms of accuracy. An overall test accuracy of 0.3116 is achieved. 8 classes are not recognized at all (zero accuracy).

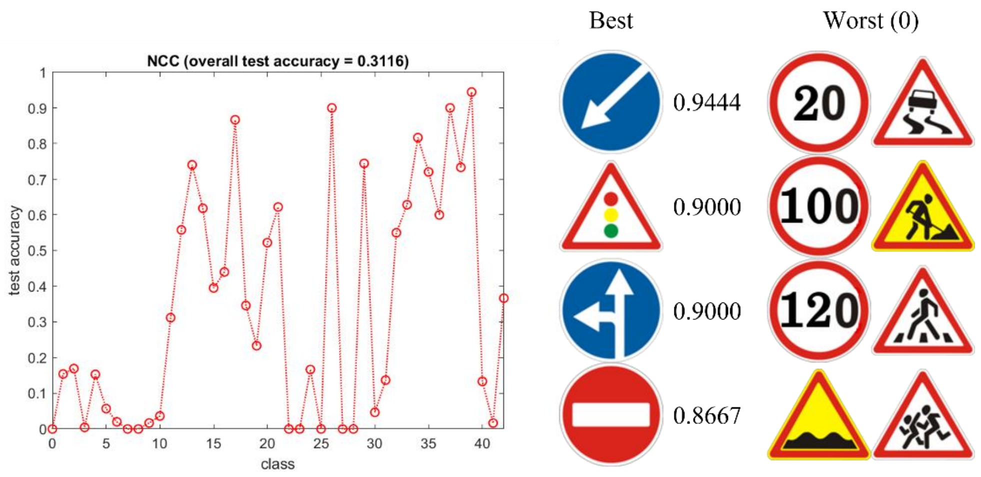


Fig. 5. test results for NCC

III. Histogram of Oriented Gradients (HOG)

Gaussian derivative masks are used to find gradients where the optimal standard deviation *sigma* is determined in the validation process. The whole image is divided into *patchNum* * *patchNum* patches. For each patch, the orientations of the gradients are put into a histogram with *oriNum* bins based on either being signed (two orientations differing by 180 degrees are considered different) or unsigned (two orientations differing by 180 degrees are considered the same). The histogram describing each patch can be considered a vector, from which there are two ways to represent the whole image: (1) a covariance matrix where the minimum Riemannian Manifold with the mean covariance matrices is used to classify images; (2) a rasterized vector from all the patches where the minimum Euclidean distance to the mean vectors is used to classify images. 20% of training images from each class together form a validation set which is used to tune the *parameters (sigma, patchNum, oriNum, signed/unsigned)*.

For preliminary model selection, images are divided into 8*8 patches and there are 9 bins in the histogram. Models are built upon both grayscale images and RGB images. Performance of each model is evaluated upon the validation accuracy with different standard deviations (*sigma = 0.25, 0.5, 0.75 and 1*), where the optimal *sigma* is determined in the validation process.

The validation results from grayscale images are shown in Fig. 6. A best validation accuracy of 0.3059 is achieved in the covariance models and a best validation accuracy of 0.8020 is achieved in the vector models.

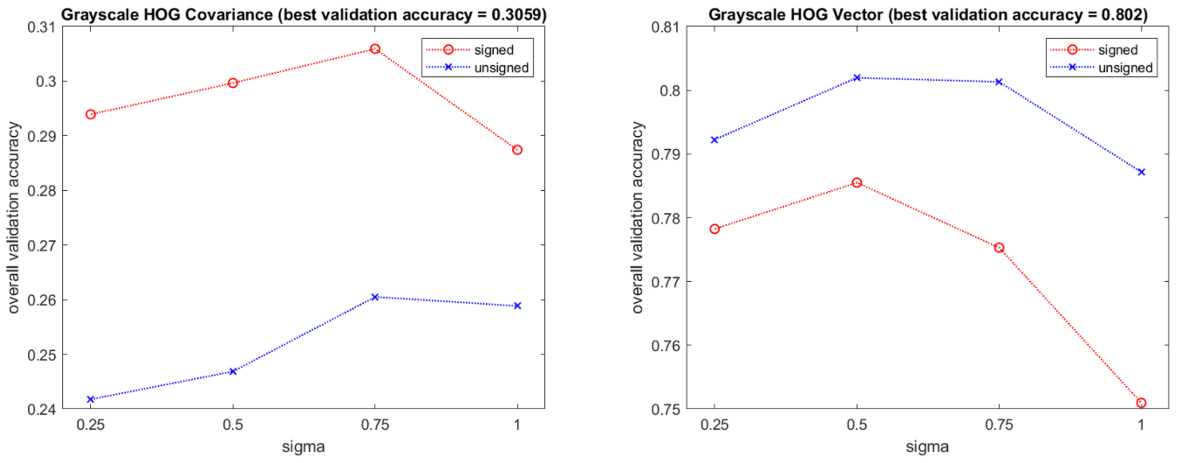


Fig. 6. validation results for grayscale images

The validation results from RGB images using the RGB voting method are shown in Fig. 7, where the classification is determined by the majority of the three channels and, if there is a tie, by the green channel. A best validation accuracy of 0.3063 is achieved in the covariance models and a best validation accuracy of 0.7972 is achieved in the vector models.

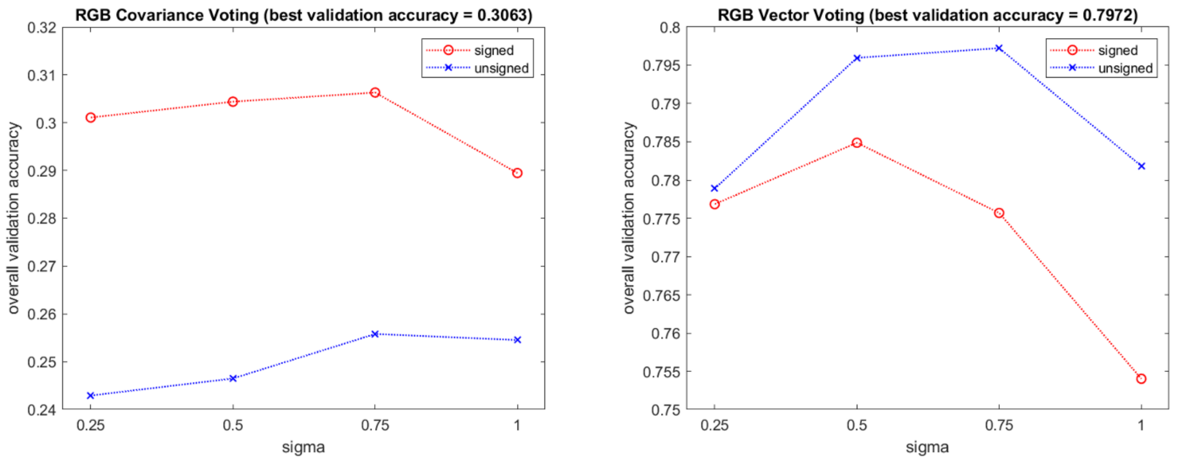
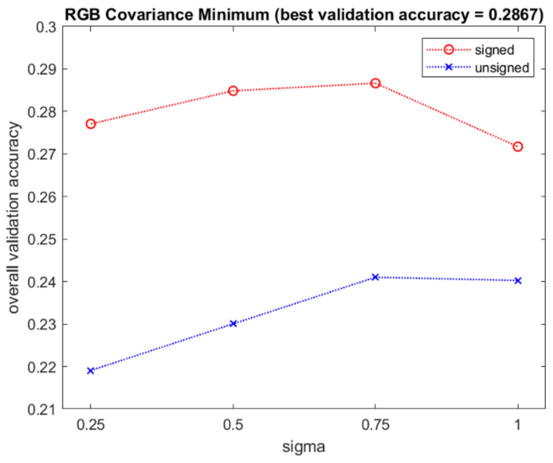


Fig. 7. validation results from RGB images classified using the RGB voting method

The validation results from RGB images using the RGB minimum method are shown in Fig. 8, where the classification is determined by the channel with the minimum measure. A best validation accuracy of 0.2867 is achieved in the covariance models and a best validation accuracy of 0.8011 is achieved in the vector models.



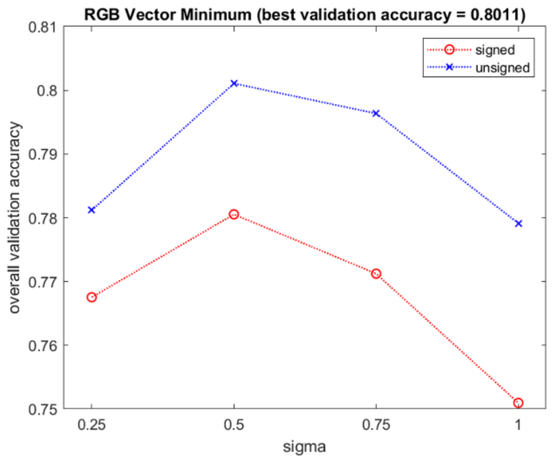


Fig. 8. Validation results from RGB images classified using the RGB minimum method

The validation results from RGB images using the RGB vector total method are shown in Fig. 9, where the classification is determined by the total Euclidean distance in RGB space. A best validation accuracy of 0.8122 is achieved.

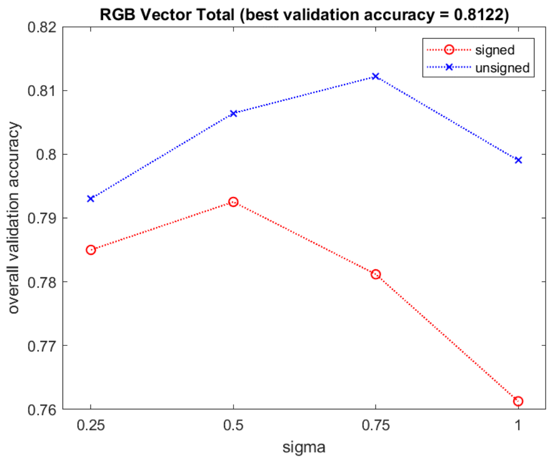


Fig. 9. validation results from RGB images classified using the RGB vector total method

The best validation accuracy achieved from RGB images is 0.8122, which is slightly higher than the best validation accuracy of 0.8020 achieved from grayscale images. The comparison among the RGB images, grayscale images and three channels is shown in Fig. 10. In terms of orientations of gradients, grayscale images have captured most of the information, and, if each channel is treated independently, some information may be lost, which could be the reason why the best validation accuracy achieved from the RGB minimum method is slightly lower than that from the grayscale images. Grayscale images are good enough to build models on.

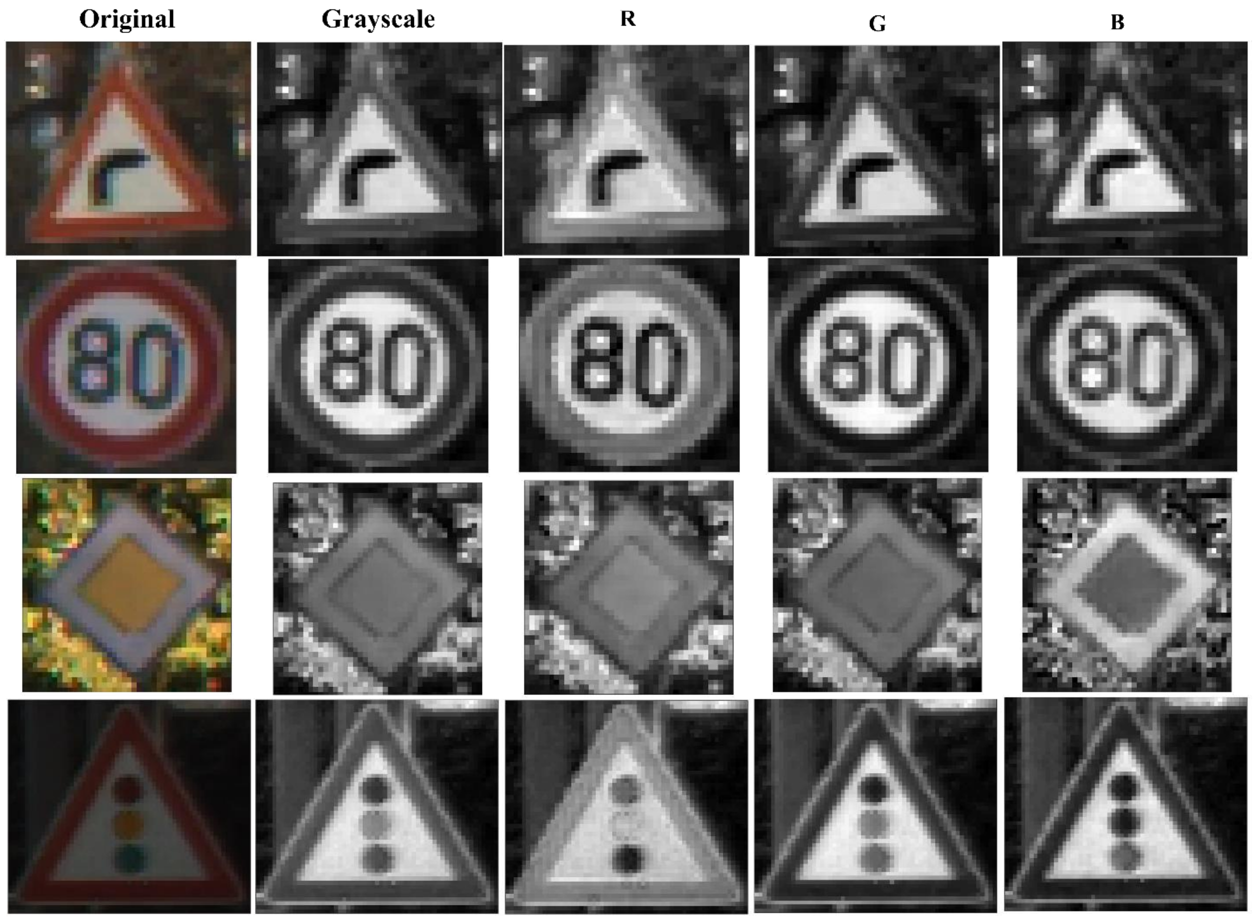


Fig. 10. comparison among the RGB images, grayscale images and three channels

The best performing signed HOG covariance model and unsigned HOG vector model from preliminary model selection are then refined using grayscale images. The optimal number of patches and the optimal number of bins in the histogram are determined in the validation process. The validation results for the signed HOG covariance model and the unsigned HOG vector model are shown in Fig. 11.

A best validation accuracy of 0.3484 is achieved for the signed HOG covariance model and a best validation accuracy of 0.8323 is achieved for the unsigned HOG vector model.

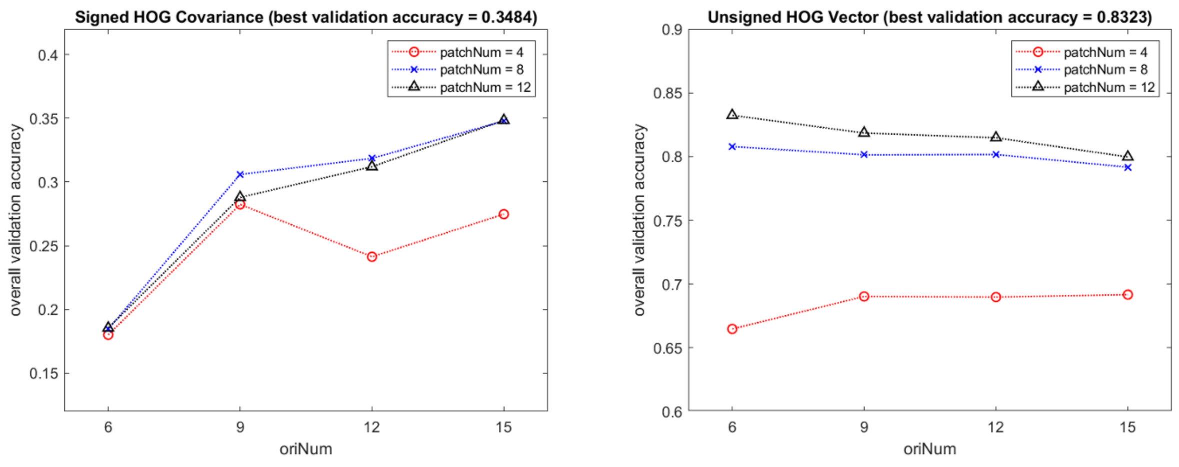


Fig. 11. validation results from the signed HOG covariance model and the unsigned HOG vector model

The test results for the signed HOG covariance model are shown in Fig. 12, along with the classes best and worst classified in terms of accuracy. An overall test accuracy of 0.3146 is achieved. The best test accuracy among classes is 0.8565 and the worst test accuracy among classes is 0.0111.

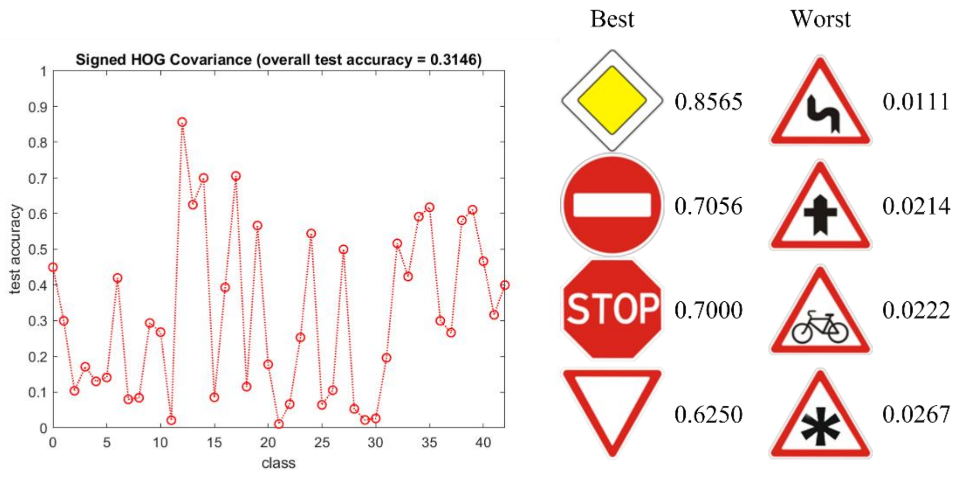


Fig. 12. test results for the signed HOG covariance model

The test results for the unsigned HOG vector model are shown in Fig. 13, along with the classes best and worst classified in terms of accuracy. An overall test accuracy of 0.7788 is achieved. The best test accuracy among classes is 1.000 and the worst test accuracy among classes is 0.0444. The test accuracies for all the classes other than class 29 are above 0.45.

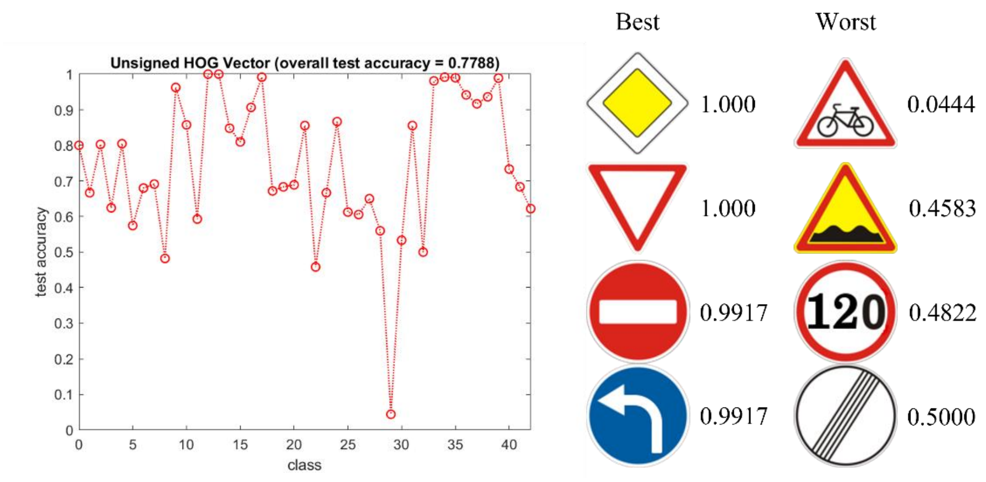


Fig. 13. test results for the unsigned HOG vector model

IV. Prediction using Multivariate Normal Likelihoods (*Octavian Donca*)

1. Data loading

The dataset was split into training, validation, and testing sets. Where a select number of validation images were chosen from the training set to be used for parameter tuning. Training, validation, and test set image counts were chosen manually due to the different number of images per class, where some were as low as 210 and others 500+. Before selection. The original dataset was randomized before each training run.

2. Training process

During the training process, for each image in the training set the region of interest is extracted from the image (provided by the dataset). Then the x and y derivatives of the image are calculated using a gaussian derivative filter. Next, the histogram of gradient (HOG) vector is calculated for the image using the x and y derivatives. Once the histogram of gradient vector is calculated for each image in the training set, a mean vector and covariance matrix is calculated for each class.

3. Histogram of Gradient process

An initial assumption is made for HOG vector calculation, which is that the number of cells to be used is a perfect square so as to ensure the image can be separated into a N x N matrix of cells. First, the image gradient and directions are calculated using the provided x and y derivatives of the image.

Directions are calculated using arctan if the HOG is to be unsigned for a [-pi/2, pi/2] range of directions, or arctan2 if the HOG is to be signed for a [-pi, pi] range of directions. Next, for every cell three histograms are initialized based on the provided number of bins, one for each color channel. Then, for every pixel in every cell, the RGB magnitudes and directions are obtained. The histogram is updated based on the pixel directions, based on parameters the histogram bin value is either simply incremented by one or by the magnitude of the pixel. Finally, the HOG vectors for each cell are concatenated, flattened, and then normalized to be a unit vector.

4. Inference process

The inference process is very similar to the training process. Inference images are selected either from the test set or validation set. For each inference image, the HOG vector is calculated and then used along with each class mean vector and class covariance matrix to describe a multivariate normal distribution. The probability density function of the distribution is then used to calculate the log-likelihood that the image belongs to each class. The final prediction class is then chosen as the class with the highest log-likelihood.

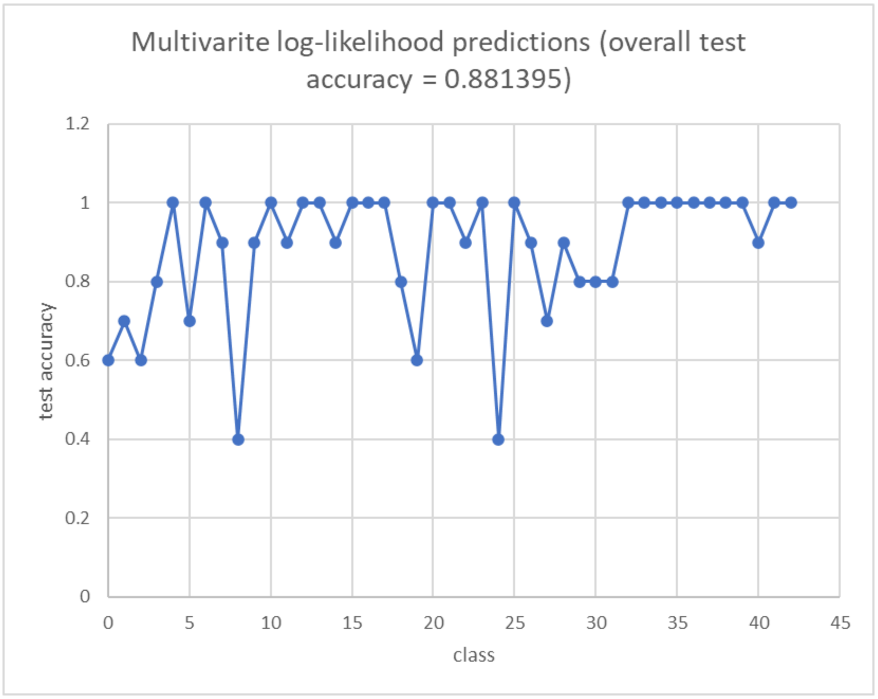


Fig. 14. test results for Multivariate Normal Likelihoods

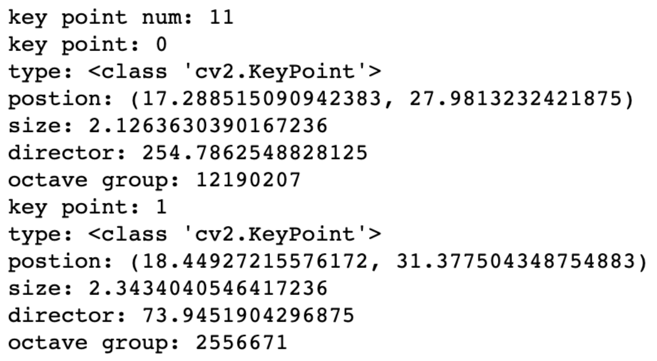
5. Results

Several tests were ran comparing different parameters configurations. Generally, accuracy increased as HOG cell count increased, accuracy increased as HOG bin count increased, accuracy increased with signed HOG compared to unsigned HOG, and accuracy increased using HOG counts instead of magnitudes. The best result achieved was using HOG cell count = 9, HOG bin count = 8, and signed HOG with an overall accuracy of 88.14%, as shown in Fig. 14.

V. Traffic Sign Recognition Using Scale Invariant Feature Transform (SIFT) and K-Nearest Neighborhood (KNN) (*Wenjin Fu*)

The traffic sign recognition has been a challenge because it can be influenced by photographing angle and daylight. Scale-Invariant Feature Transform method is a strong image feature extraction technique created by D.Lowe, University of British Columbia, in 1999. Because it is invariant to image scale, rotation, and changes in image illumination, our group attempted to use it to extract image-level key points feature to develop a feature descriptor for each image. From that the K-Nearest Neighborhood can then be used to cluster 43 groups of images depending on the SIFT feature extracted from each image.

The SIFT feature extraction can be divided into two steps: key points detection and key point description. But each steps can be detailed into 2 sub steps in the following:



```
key point num: 11
key point: 0
type: <class 'cv2.KeyPoint'>
postion: (17.288515090942383, 27.9813232421875)
size: 2.1263630390167236
director: 254.7862548828125
octave group: 12190207
key point: 1
type: <class 'cv2.KeyPoint'>
postion: (18.44927215576172, 31.377504348754883)
size: 2.3434040546417236
director: 73.9451904296875
octave group: 2556671
```

Fig. 15. keypoint location

1. ***Scale Space Interest Points Selection*:** The scale space of images is generated by applying different scale of gaussian blurring to the input image. Based on the size of the image, the scale space is separated

into several number of octaves. Scale Space Interest Points Selection is implemented by using difference of gaussian (DOG) to search all the scale and space of the image in the gaussian pyramid to find the potential feature keypoints in the images.

2. ***Keypoint Localization*****:** For each candidate key point, keypoints are chosen for their stability. A detail example of description of each keypoint is given in Fig. 15.

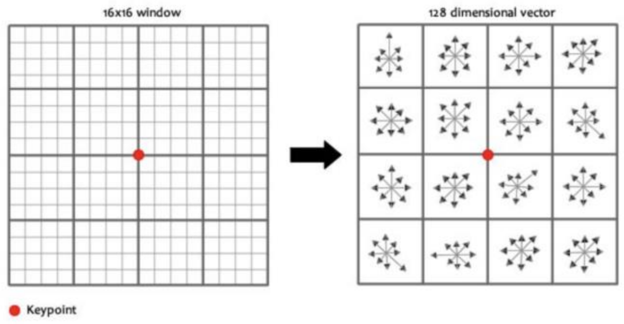


Fig. 16. keypoint descriptor

3. ***Keypoint descriptor*****:** Since the keypoints are detected in different scale of images, so the keypoints detection is invariant to scale of image. Then the keypoints feature can be descripted by computing the 8 bins orientation histogram of for each block of keypoints neighbours. 16 x 16 window are created around the keypoints and it further divided keypoints neighbours into 4 x 4 sub-blocks for computing the 8 bins orientation for each sub-block. Therefore, each keypoints can be descripted as 1 x (4 x 4 x 8) vector feature. A sample keypoint descriptor is shown in Fig. 16.

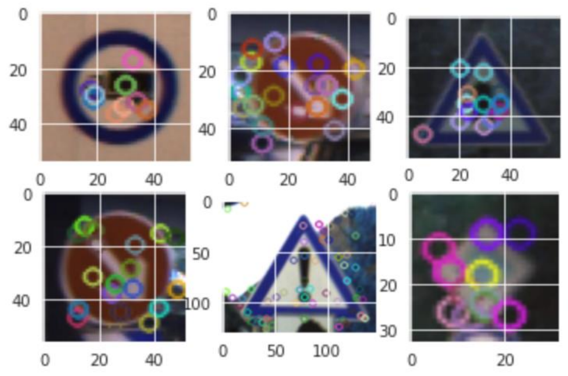


Fig. 17. Drawing keypoints on images

A visualization of drawing keypoints on traffic sign images are tested above in Fig. 17, and A visualization of correspoinding matching points between 2 different images in the same class is shown in Fig. 18.

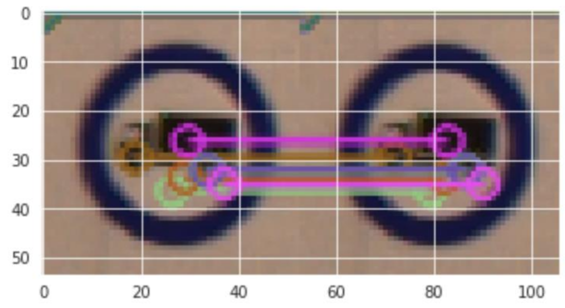
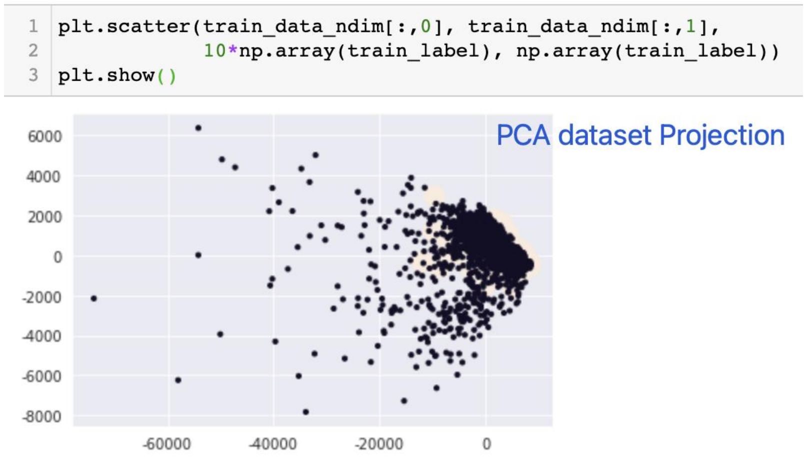


Fig. 18. Corresponding matching points

4. ***Image data transformation*****:** Now the number of keypoints and keypoints descriptor (1x128 vector) are computed on each image. Each image feature can now presented as a matrix of number of keypoints x 128 dimensions. In order to transform the image feature into same dimension (1 x 128) for the future classification, we squeeze the column number by simply summing up the value in vertical axis. The 2 dimension transformed dataset projection through PCA is shown in Fig. 19. (I was trying to classify the data into 43 ID classes by color. But it took me too long to implement it.)



```
plt.scatter(train_data_ndim[:,0], train_data_ndim[:,1],
            10*np.array(train_label), np.array(train_label))
plt.show()
```

Fig. 19. 2 dimension transformed dataset projection through PCA

The K-Nearest Neighborhood can now be used to implement classification on the transformed image data. The KNN model first randomly initialize 43 center clusters. Then the Euclidean distance method is used to compute the distance between each SIFT feature and the cluster center. The new 43 center clusters are selected iteratively respecting to the top smallest distance between each SIFT feature and cluster center until the cluster center stop changing. The overall 80% accuracy on the traffic sign classification are achieved on the test dataset, as shown in Fig. 20.

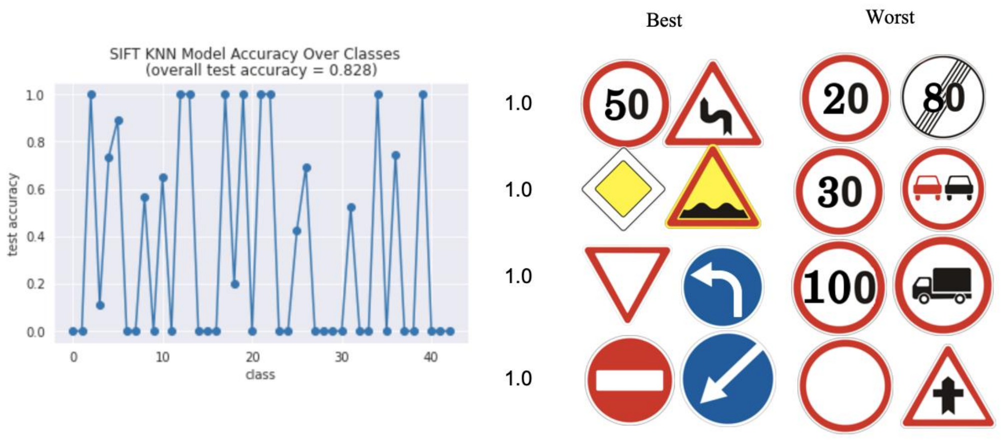


Fig. 20. test results for SIFT KNN model