Vaskovskaya Daria

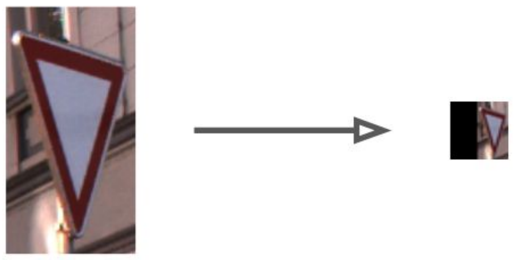
Traffic Sign Recognition

Homework Assignment 2

Transform images to same size

Task: padd rectangular images to square shape and size them to the same dimensionality - 30x30 pixels

Example of transformation (picture “GTSRB/Final_Test/Images/03318.ppm”):



Where original size: (75, 143), new size: (30, 30)

Split data

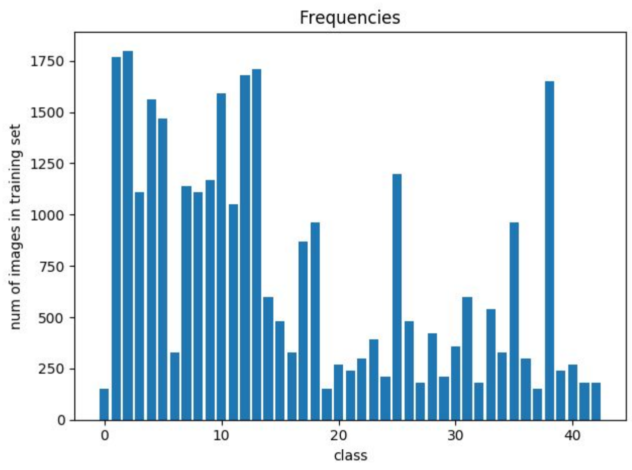
Question: Make sure that images from the particular track end up being in either training or validation split, otherwise validation will be faulty (Think why?)

Answer: Validation will be faulty because pictures from the same track look nearly the same, this would lead to a better accuracy score during validation, because model has already learned nearly the same picture.

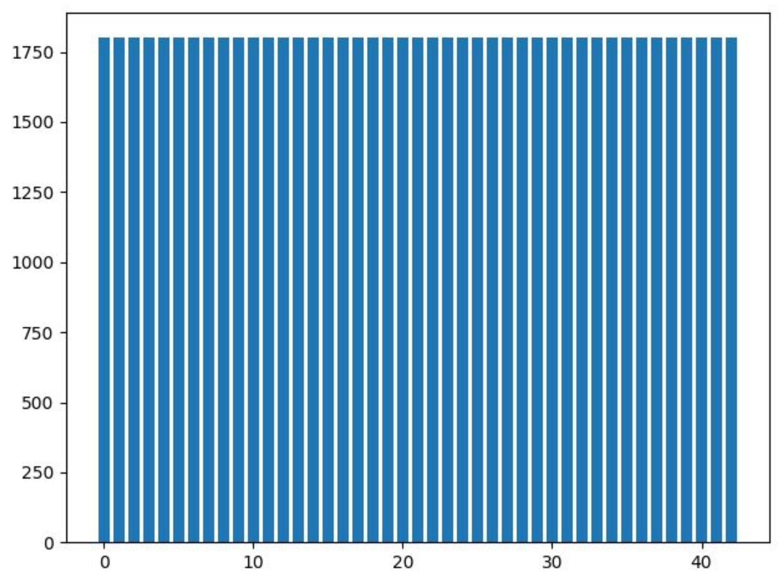
Examples of pictures from the same track:

Frequencies

Task: Find class frequencies in the resulting training set. Include a bar chart in your **Report**



Augmentation



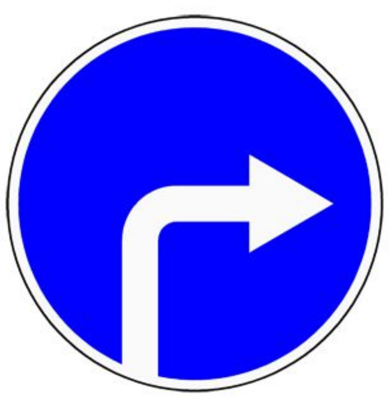
Task: Check some resources to see what are the possible ways to augment your data and choose two-three techniques that are appropriate in our particular case. In the Report you should justify your choice and provide examples of augmentation (original vs augmented images).

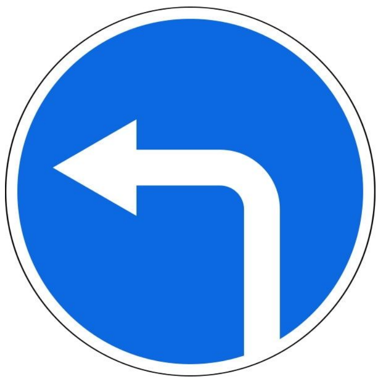
The following techniques for the augmentation were presented:

1) Flip

Not suitable. Direction is crucial to the traffic signs, so flipping them is not a good idea.

Example:

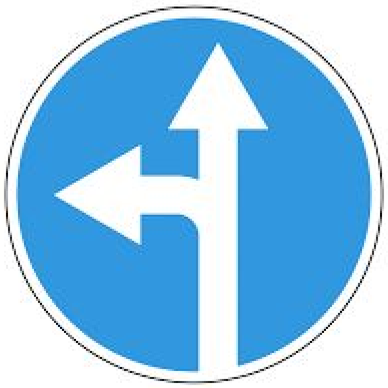




2)Crop

Cropping is possible, however it is pretty hard to make it work well.

The following situation after cropping is possible:



Making traffic sign impossible to guess even for the human being.

3)Randomly Change Brightness, Randomly Change Contrast

It is ok within reasonable limits. Changing brightness and contrast can simulate photos taken during sunny weather,evening etc

4)Randomly Change Hue

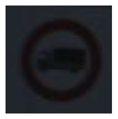
No sense, because color is crucial to the traffic signs

5)Noise

It is ok within reasonable limits. Can simulate photos taken in bad conditions.

I have chosen Randomly Change Brightness, Randomly Change Contrast, Gaussian Blur (it is easier to implement than noize presented in the paper)

Examples:



What are the features?

Image can be represented as a matrix with 3 dimensions ((RGB) * width * height).

After normalization, the vector is made with [numpy.ravel](). Every element of a vector is a feature.

Train model

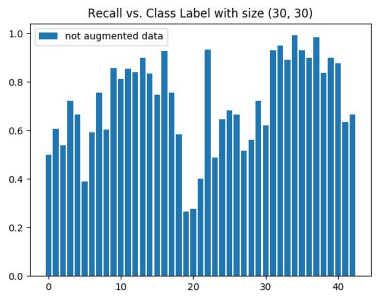
Model:

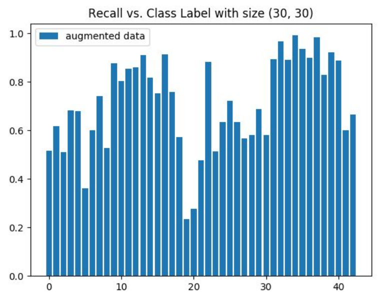
RandomForestClassifier(n_jobs=-1, n_estimators=50)

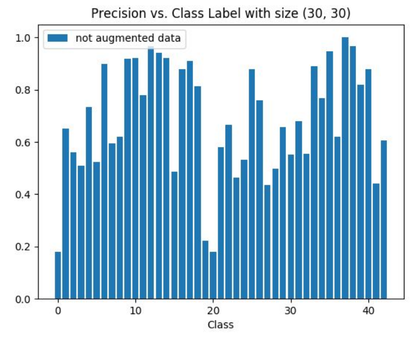
Evaluate

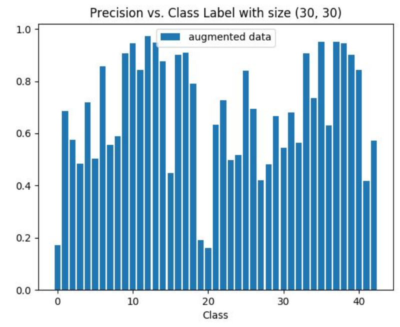
Calculate the resulting overall accuracy on the test set, measure recall and precision for each class separately to analyze problem areas, visualize samples that were classified incorrectly and add them to the **Report**

Overall accuracy on the test set: with image size (30X30) ~72





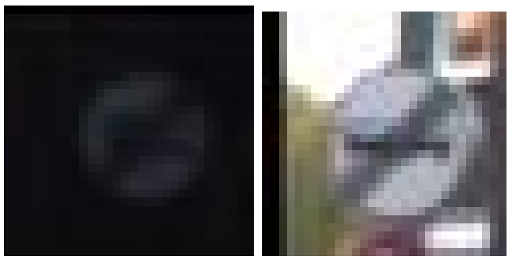




Samples that were classified incorrectly:

Tried to guess: True class : Predicted class:

Some other pictures which were classified incorrectly:



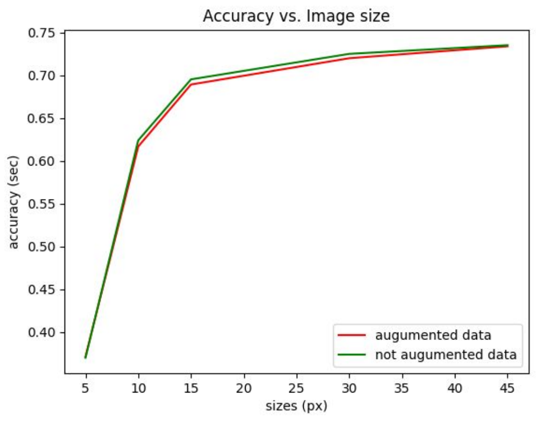
Experiment and analyze!

Task:

Try training the model on non-augmented data. Compare models' performance. Analyze the effect of augmentation. Describe your findings in the Report

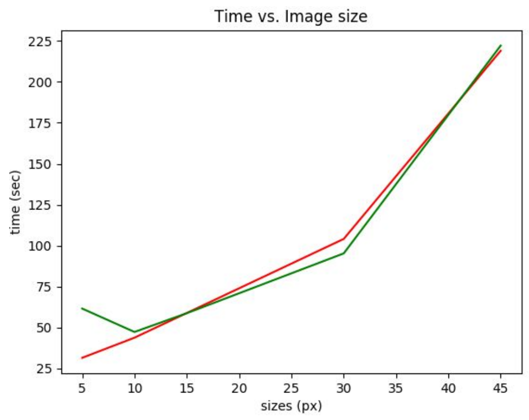
Try changing the size of the images. Does increasing it results in better performance? Choose 5 different size options and include to the Report 2 plots: how accuracy and time depends on image size.

Accuracy:



According to the plot presented above and bars from the “evaluate” part, I can say that my implementation of augmentation nearly does not affect the performance of a model.

Increasing size (sizes = [(5, 5), (10, 10), (15,15), (30, 30), (45, 45)]) helped model to produce better results, however it is time consuming:



Code structure

Part of collected data is written to the console, images and plots are stored there: