Machine Learning

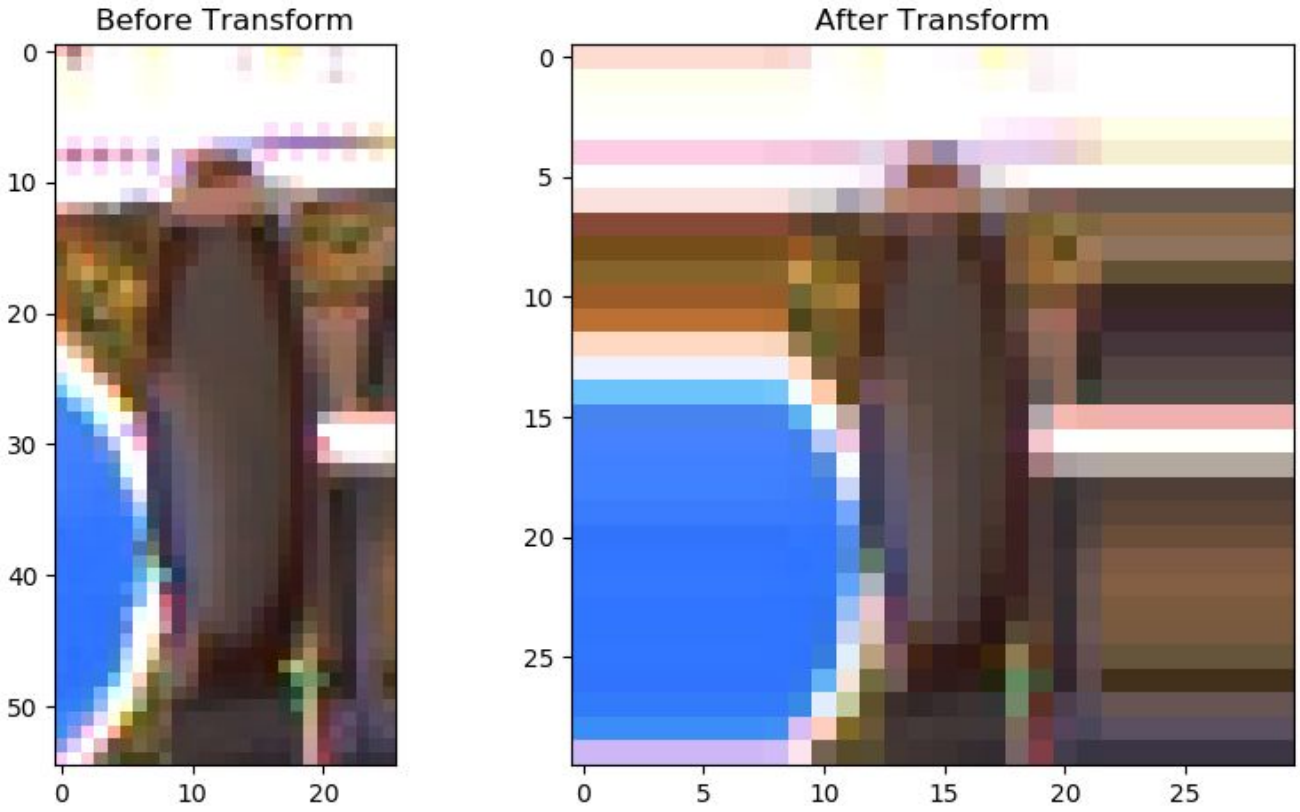
Assignment #2

Andriyanchenko Oleg, DS-02

1. Image Transformation

Used libs: openCV, numpy.

Brief explanation: Firstly, I find the longer side and, based on a difference with shorter one, I calculate the required padding. For padding I replicate the existing border using *cv2.BORDER_REPLICATE*.



2. Data Splitting

Used libs: implemented by myself.

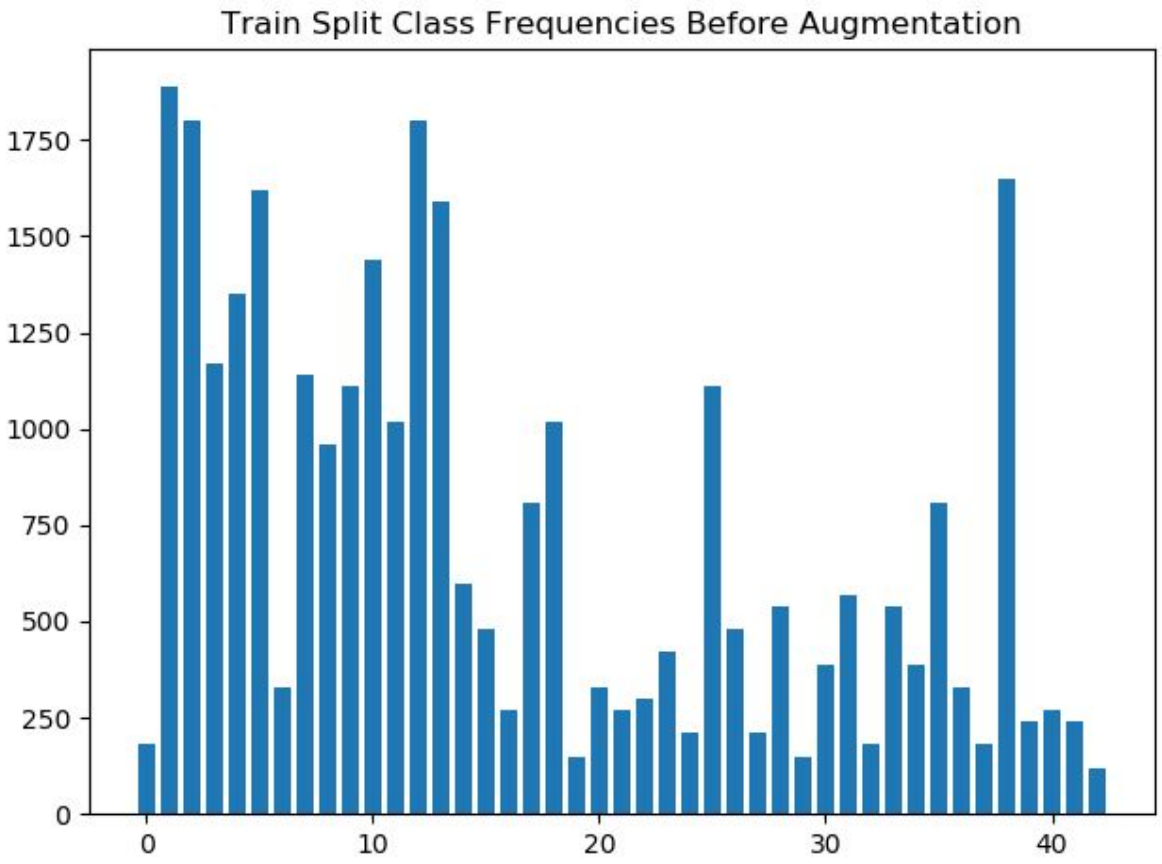
Brief explanation: I use uniformly distributed random variable to split data. Variable refreshes every track, so each track goes fully to either test or validation split.

```
-- Reading data Train Data--
98%|▉▉▉▉▉▉▉▉▉▉| 42/43 [00:27<00:00,  2.25it/s]
-- Splitting the data --
100%|▉▉▉▉▉▉▉▉▉▉| 43/43 [00:27<00:00,  2.63it/s]

  0%|          | 0/39209 [00:00<?, ?it/s] -- Data Splitting --
100%|▉▉▉▉▉▉▉▉▉▉| 39209/39209 [00:00<00:00, 1231555.00it/s]
X_train:  31439
X_validation:  7770
Percentage of train data:  80.18312122216838
```

3. Frequency

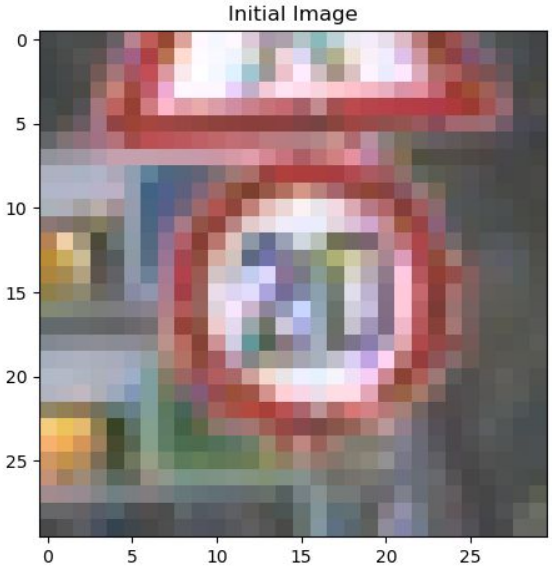
The bar chart representing the frequency after the split:

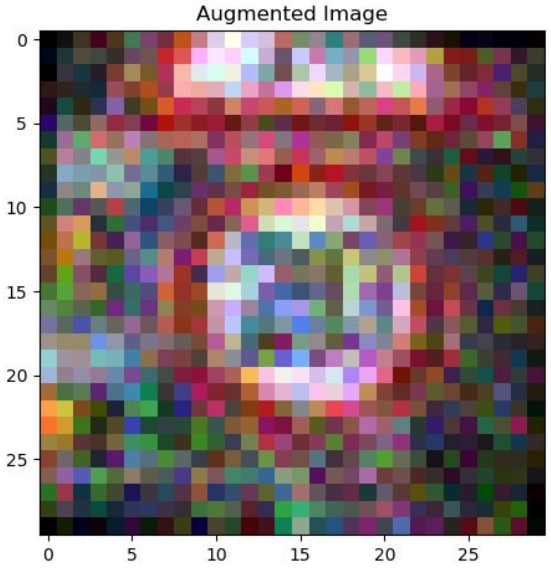


4. Augmentation

Used libs: skimage.

Brief explanation: for augmentation I chose 3 different transformations: rotation from -20 to 20 degrees, random noise and changing brightness. There are other options as well, but in real life these are the best cases. Flipping and mirroring the image will only affect our data in a bad way since we do not see flipped road signs.





5. Evaluation

After testing the trained model I get following results:

Accuracy: 0.73

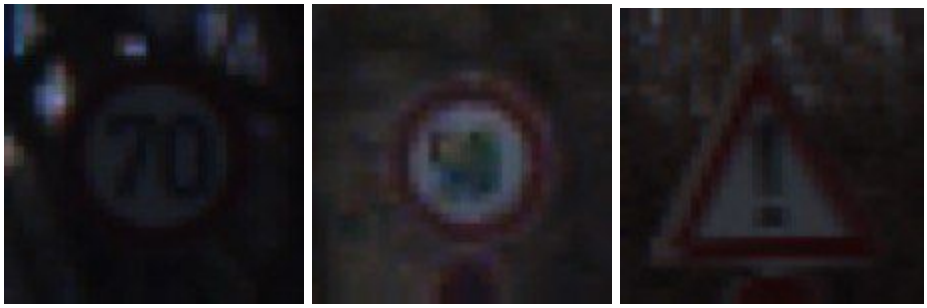
Recall and Precision for each class:

	precision	recall	f1-score	support
0	1.00	0.08	0.15	60
1	0.64	0.79	0.71	720
10	0.90	0.94	0.92	660
11	0.82	0.90	0.86	420
12	0.97	0.91	0.94	690
13	0.93	0.97	0.95	720
14	0.97	0.89	0.92	270
15	0.94	0.70	0.80	210
16	0.99	0.94	0.97	150
17	0.91	0.81	0.85	360
18	0.61	0.59	0.60	390
19	0.68	0.38	0.49	60
2	0.44	0.68	0.54	750
20	0.17	0.21	0.19	90
21	0.94	0.19	0.31	90
22	0.92	0.71	0.80	120
23	0.57	0.45	0.50	150
24	0.88	0.40	0.55	90
25	0.83	0.93	0.88	480
26	0.62	0.62	0.62	180
27	0.40	0.03	0.06	60
28	0.68	0.69	0.69	150
29	0.86	0.47	0.60	90
3	0.52	0.63	0.57	450
30	0.40	0.32	0.36	150
31	0.62	0.74	0.68	270
32	0.00	0.00	0.00	60
33	0.87	0.63	0.73	210
34	1.00	0.93	0.96	120
35	0.83	0.91	0.87	390
36	0.89	0.58	0.70	120
37	1.00	0.87	0.93	60
38	0.87	0.95	0.91	690
39	0.98	0.67	0.79	90
4	0.69	0.68	0.68	660
40	0.95	0.90	0.93	90
41	0.11	0.05	0.07	60
42	0.51	0.29	0.37	90
5	0.51	0.57	0.54	630
6	0.50	0.47	0.48	150
7	0.57	0.64	0.61	450
8	0.69	0.37	0.48	450
9	0.91	0.82	0.86	480
accuracy			0.73	12630
macro avg	0.72	0.61	0.64	12630
weighted avg	0.74	0.73	0.72	12630

0.7288994457640539

As we can see most classes have high Precision and high Recall -> our classifier is good for these classes. The classes that had the least number of images got mostly classified incorrectly.

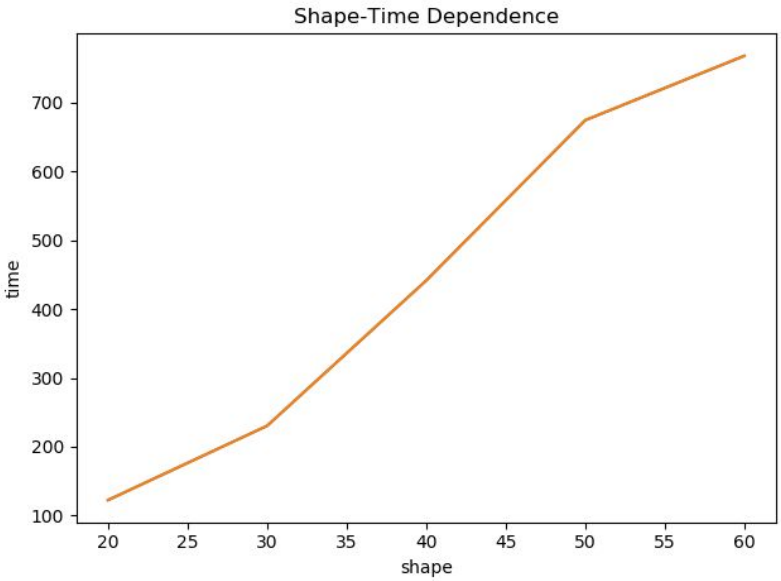
Part of incorrectly classified samples:

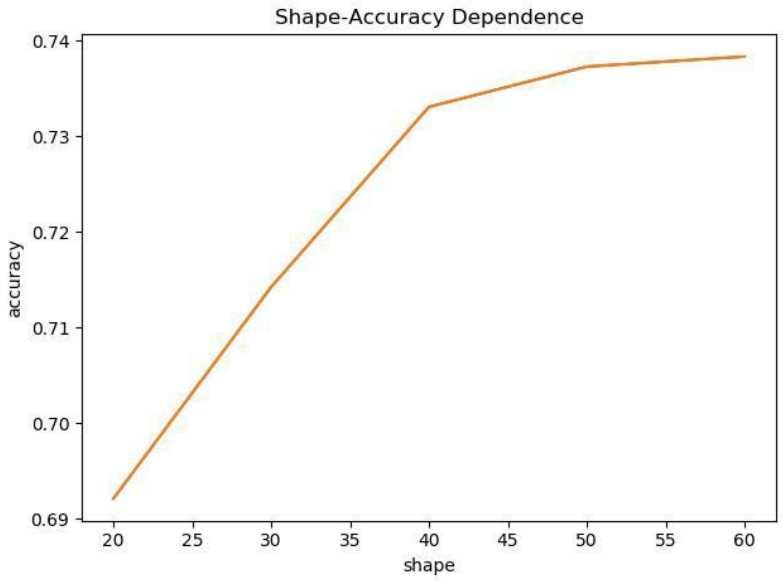


6. Experimenting

Differences in training with and without augmentation:
Accuracy without augmentation: 0.69
Accuracy with augmentation: 0.73
Model that takes augmented data takes significantly more time to finish. Despite that we can clearly see the increase in the accuracy. That is why we prefer to use augmented data as it generally gives us more precise prediction.

After going through 5 different shapes (20x20, 30x30, 40x40, 50x50, 60x60) I discovered the following dependency:





Time and Accuracy increase as the size of the picture increases. At some point it becomes unbeneficial to use bigger size as time increases significantly and accuracy increases only by a bit.

7. Conclusion

Using augmentation really helps out when we have uneven distribution of data or when we have a small amount of data at all. It generates new images without losing the information.
I also did parameter tuning on the validation split. I increased *n_estimators* of the RandomForestClassifier from 50 to 512. From around 126 the time it took to train was much larger than accuracy gain. So I did use 60 estimators in the end as it gave me a decent accuracy and took less time.
My best Accuracy score was 75%. But no matter what I tried I wasn't really able to increase it. As far as my knowledge goes result of RandomForest is considered great if it reaches 85%. Maybe there is not enough data to train on or we have to reconfigure sklearn model to better suit our needs.