# Tooploox Data Scientist Exercise – Solution

The solution was implemented in Python. It consists of 3 files:

- utils.py – set of util methods operating on CIFAR-10 or helping in peforming classification
- process_cifar.py – this module downloads and extracts dataset, then visualize 10 images per each class
- shallow_learning.py – performs SVM classification on HOG features (with hyperparameter optimization)
- transfer_learning.py – extracts features using transfer learning approach with the use Inception v3 model, visualizes them using two dimension reduction techniques – PCA and t-SNE, performs SVM classification based on these features (with hyperparameter optimization). Moreover, to increase the accuracy additional approaches were performed: using different pretrained model (in this case InceptionResNet v2), combining features from two models (used features obtained by Inception v3 and InceptionResNet v2) as well as data augmentation.

The order of executing files is arbitrary, but suggested one is as follows: process_cifar.py, then shallow_learning.py and, finally, transfer_learning.py. Due to the fact that some of the operation are long-running, these results are saved on disk during the first execution of such operation (and during next executions the results are read from disk). I don't have access to CUDA-enabled GPU, so I decided to perform all computation on 1/10 of the dataset, as it was suggested in the task.

Steps of solution (referring to steps from the task description):

1. CIFAR-10 is downloading from the following url: http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz. It has 5 training batches and a test batch. Due to lack of access to CUDA-enabled GPU, I decided to use 1/10 of the data from each batch. Each batch contains images in random order (as it was stated on CIFAR-10 website), so, to simplify, first 1/10 of the data was read from each batch.

2. Visualization of 10 random images for each class:
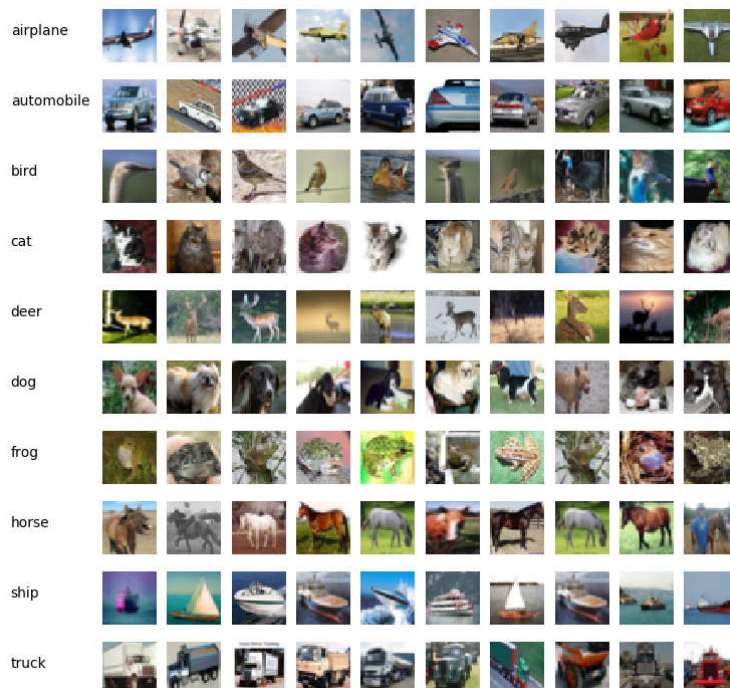
3. As a shallow classifier, I extracted HOG features and then, SVM was utilized for classification (with hyperparameter optimization)

Accuracy on test set: **0.477** (for 'C'=0.1 parameter)

Evaluation metrics per each class (precision, recall, f1 and support):

|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.41 | 0.47 | 0.43 | 103 |
| 1 | 0.50 | 0.60 | 0.54 | 89 |
| 2 | 0.48 | 0.33 | 0.39 | 100 |
| 3 | 0.40 | 0.27 | 0.32 | 103 |
| 4 | 0.42 | 0.34 | 0.38 | 90 |
| 5 | 0.35 | 0.26 | 0.30 | 86 |
| 6 | 0.48 | 0.60 | 0.53 | 112 |
| 7 | 0.49 | 0.70 | 0.57 | 102 |
| 8 | 0.53 | 0.59 | 0.56 | 106 |
| 9 | 0.62 | 0.56 | 0.59 | 109 |
| avg / total | 0.47 | 0.48 | 0.47 | 1000 |

4. For transfer learning, Inception v3 pretrained model was used. The penultimate layer from such network was used to extract features (as it is a standard way).

5. Visualization of visual features extracted from pretrained model was carried out by 2 dimensionality reduction techniques:

- PCA



PCA
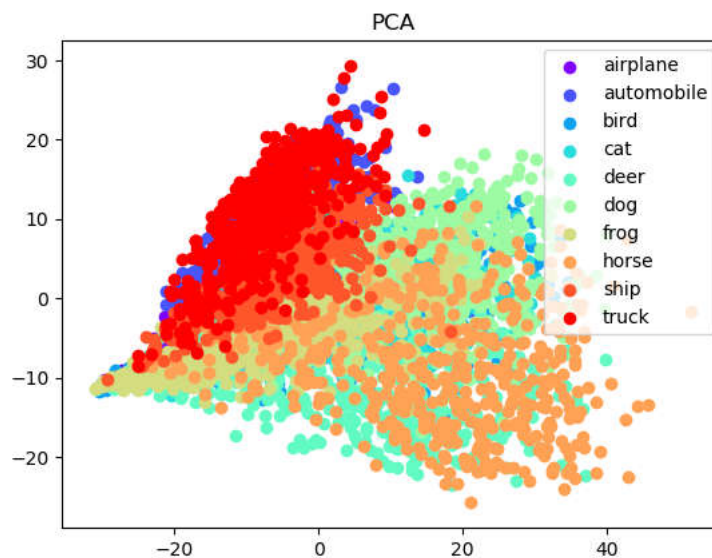
- t-SNE – this method is much more computationally intensive than PCA, so it is recommended to initially use another dimensionality reduction technique, and then apply t-SNE. In this case, firstly PCA with 50 components was executed, and then t-SNE on the reduced dataset.
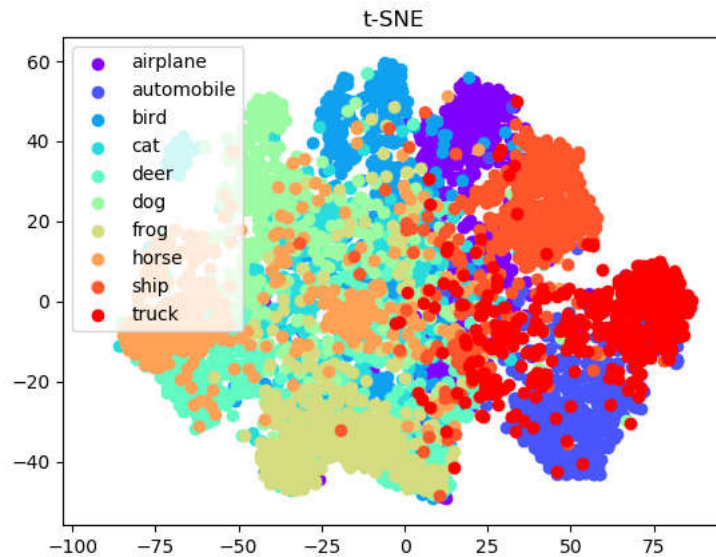
t-SNE

Comparing these two results obtained by 2 different methods, we can notice that t-SNE better visualize separation between classes (but some overlappings are also visible).

6. On the top of features extracted from Inception v3 model, a SVM classifier was trained with hyperparameter optimization. The best results were provided using 'C'=0.0001.

7. Obtained accuracy on the test set: **0.812**. Evaluation metrics per each class:

|   | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.75 | 0.76 | 0.75 | 103 |
| 1 | 0.88 | 0.83 | 0.86 | 89 |
| 2 | 0.89 | 0.76 | 0.82 | 100 |
| 3 | 0.72 | 0.74 | 0.73 | 103 |
| 4 | 0.75 | 0.78 | 0.77 | 90 |
| 5 | 0.75 | 0.74 | 0.75 | 86 |
| 6 | 0.82 | 0.86 | 0.84 | 112 |
| 7 | 0.83 | 0.83 | 0.83 | 102 |
| 8 | 0.86 | 0.95 | 0.90 | 106 |
| 9 | 0.87 | 0.84 | 0.86 | 109 |
| avg / total | 0.81 | 0.81 | 0.81 | 1000 |

**Bonus tasks:**

3 additional approaches will be presented (each makes an improvement over the results from 7th step)

8. Trying a different pretrained network which achieves better results on ImageNet dataset. Inception v3 has 159 layers and scores accuracy 0.788 on ImageNet. I decided to use InceptionResNet v3 network which has 572 layers and achieves accuracy 0.804 on ImageNet. The process of extracting features was similar as in the previous steps. Also in this case, features were fed to a SVM classifier with hyperparameter optimization. The best results were provided using 'C'=0.001. Obtained accuracy on the test set: **0.851**.
Detailed results for other evaluation metrics per each class:

|            | precision | recall | f1-score | support |
|------------|-----------|--------|----------|---------|
| 0          | 0.83      | 0.83   | 0.83     | 103     |
| 1          | 0.89      | 0.82   | 0.85     | 89      |
| 2          | 0.87      | 0.78   | 0.82     | 100     |
| 3          | 0.79      | 0.77   | 0.78     | 103     |
| 4          | 0.84      | 0.83   | 0.84     | 90      |
| 5          | 0.84      | 0.83   | 0.83     | 86      |
| 6          | 0.83      | 0.90   | 0.86     | 112     |
| 7          | 0.86      | 0.91   | 0.89     | 102     |
| 8          | 0.88      | 0.92   | 0.90     | 106     |
| 9          | 0.89      | 0.90   | 0.89     | 109     |
| avg / total| 0.85      | 0.85   | 0.85     | 1000    |

9. Combining features extracted from two pretrained models: Inception v3 and InceptionResNet v3. Also in this case, a SVM classifier with hyperparameter optimization was applied. The best results were achieved using 'C'= 0.0001. Obtained accuracy on the test set: **0.87**. Details for other evaluation metrics:

|            | precision | recall | f1-score | support |
|------------|-----------|--------|----------|---------|
| 0          | 0.78      | 0.83   | 0.81     | 103     |
| 1          | 0.92      | 0.89   | 0.90     | 89      |
| 2          | 0.89      | 0.79   | 0.84     | 100     |
| 3          | 0.81      | 0.81   | 0.81     | 103     |
| 4          | 0.88      | 0.84   | 0.86     | 90      |
| 5          | 0.86      | 0.85   | 0.85     | 86      |
| 6          | 0.85      | 0.92   | 0.88     | 112     |
| 7          | 0.90      | 0.92   | 0.91     | 102     |
| 8          | 0.89      | 0.96   | 0.92     | 106     |
| 9          | 0.95      | 0.87   | 0.91     | 109     |
| avg / total| 0.87      | 0.87   | 0.87     | 1000    |

The results are better than using only features from one of the pretrained models.

10. Data augmentation. This approach was used to the initial solution with Inception v3 model. For each training batch, additional 250 images were generated (only horizontal flip was applied) resulting 1250 images per batch. I decided to use only horizontal flip, because this transformation seemed the most promising (looking at classes and their object e.g. in figure with 10 examples per each class, horizontal flipping seems very natural, contrary to vertical flipping) and the number of generated augmented images is not so big. Again, a SVM classifier with hyperparameter optimization was utilized. The best results were obtained using 'C'=0.0001. The achieved accuracy was **0.823** on the test set (increase from value 0.812 without data augmentation). Other evaluation metrics have the following values:

|           | precision | recall | f1-score | support |
|-----------|-----------|--------|----------|---------|
| 0         | 0.75      | 0.77   | 0.76     | 103     |
| 1         | 0.90      | 0.82   | 0.86     | 89      |
| 2         | 0.92      | 0.78   | 0.84     | 100     |
| 3         | 0.76      | 0.75   | 0.75     | 103     |
| 4         | 0.78      | 0.81   | 0.80     | 90      |
| 5         | 0.78      | 0.77   | 0.77     | 86      |
| 6         | 0.82      | 0.88   | 0.85     | 112     |
| 7         | 0.83      | 0.84   | 0.84     | 102     |
| 8         | 0.82      | 0.94   | 0.88     | 106     |
| 9         | 0.88      | 0.84   | 0.86     | 109     |
| avg / total | 0.83    | 0.82   | 0.82     | 1000    |