1 Computer Vision: Day 2 - Feature-based Image Classification

Yesterday you have hand-crafted a program to distinguish two classes of traffic signs. Today we take out a bit of the manual tweaking and train a fully-fledged image classifier on a handful of classes from the German Traffic Sign Recognition Benchmark. We use a very well-established method, named Histogram-of-Oriented-Gradients (or HOG for short), in order to map each image to a lower-dimensional vector space. The classification is then performed within this vector space. You will learn new ways to tune your classifiers and (visually) check if your programs run correctly.

1.1 Libraries

For today, the following libraries can and should be used to work on the exercises:

- `six.moves.cPickle` — a library for fast serialization
- `numpy` — a library for efficient matrix operations
- `matplotlib.pyplot` — a library for displaying images and plotting graphs
- `cv2` — a library for basic image processing operations
- `scipy` — a library for scientific computations
- `sklearn` — a library for machine learning

You can use other libraries that are installed on the lab computers, but still please **do not use tensorflow today**.

1.2 Exercises

You can reuse code from the days before, e.g., the functions to load images from GTSRB.

1. Choose three of the 43 classes that you want to work with today. Show a random example image from either of the classes.
2. Convert the chosen images to grayscale.
3. Compute a HOG descriptor for every image.
4. Perform a dimensionality reduction to all the points in the feature space and reduce them to only two dimensions.
5. Show the distribution of the points in the two-dimensional vector space. Use colors to represent each class. What can you learn?
6. Train a classifier from `sklearn` to classify your images (or the vector representations thereof). Split your dataset, identify the hyperparameters of the machine learning algorithm you use and perform a grid-search automatically (!) tune them. Evaluate the performance of the classifier by means of a validation set. What is the error rate of the best-performing classifier? Compute and display a confusion matrix for all the classes you chose.
7. If that works already quite well, how about using all possible 43 classes? What is your error rate?
8. Show a random choice of misclassified images with their correct and their incorrectly assigned class if there are any.

1.3 Hints

1. You can obtain a text label for each traffic sign class. Use them when you display something, do not use the integer label!
2. Always test your code with a toy problem.
3. If you have operations that take a long time, consider memoization in order to not wait for them more than once.

1.4 Functions

Some of the following functions may or may not come in handy when working on the exercises. Have a look at the documentation if they seem worth the while.

`numpy.arange` — Fills an array with a consecutive sequence of integer
`numpy.random.shuffle` — Randomizes the order of the given array
`cv2.HOGDescriptor` — Compute a hog feature for an image
`sklearn.decomposition.PCA` — Compute the principal components of a distribution of multi-dimensional data points
`sklearn.SVM.SVC` — Compute a multi-class support vector machine
`joblib.Memory` — A decorate class for memoization
`math.floor, math.ceil` — Round to the previous or following integer