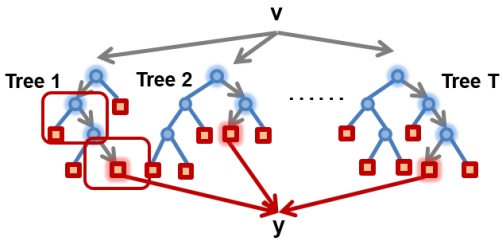
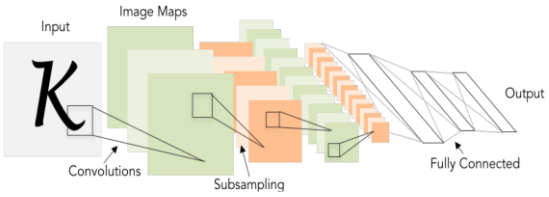
CS485 Machine Learning for Computer Vision

Coursework2: Bag-of-words, Randomised Forests, Convolutional Neural Networks [40% mark]





Release on 07 Nov 2024, the report due on 5 Dec 2024 (midnight).

The coursework requires Python/PyTorch/or Tensorflow and or Matlab programming. In all questions, you can use any existing toolbox/code, unless specified.

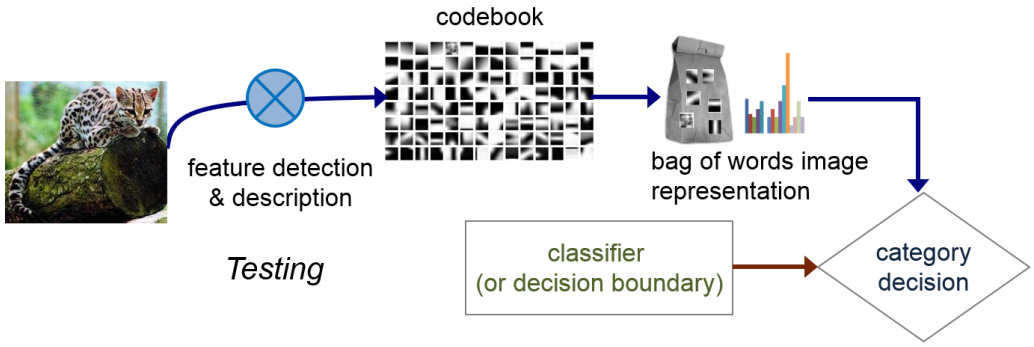
The submission instructions are same as for CW1.

Submit the report **in pdf** through the KLMS system. No hard copy is needed. Write your full names and school ID numbers on the first page.

If you have questions, please contact

Taeyun Woo (taeyun.woo@kaist.ac.kr)
Hangil Park (hangil.park@kaist.ac.kr)
Sanghyeok Nam (sang990701@kaist.ac.kr)

Experiment with Caltech101 dataset for image categorisation



We apply RF to the subset of Caltech101 dataset for image categorisation. Use the provided Caltech101 dataset. We use 10 classes, 15 images per class, randomly selected, for training, and 15 other images per class, for testing. Feature descriptors **d** are given. They are multi-scaled dense SIFT features, and their dimension is 128 (for details of the descriptor, see

http://www.vlfeat.org/matlab/vl_phow.html). In all questions, as you prefer, you can use the provided code set of Random Forest or any existing toolbox/library e.g.

https://github.com/karpathy/Random-Forest-Matlab

http://vision.ucsd.edu/~pdollar/toolbox/doc/

http://code.google.com/p/randomforest-matlab/

http://www.mathworks.co.uk/matlabcentral/fileexchange/31036-random-forest

Q1. [10] K-means codebook

We randomly select 100k descriptors for K-means clustering for building the visual vocabulary (due to memory issue). Open the main_guideline.m and select/load the dataset.

```
>> [data_train, data_test] = getData('Caltech'); % Select dataset
```

Set 'showImg = 0' in getData.m if you want to stop displaying training and testing images. Complete getData.m by writing your own lines of code to obtain the visual vocabulary and the bag-of-words histograms for both training and testing data. You can use any existing code for K-means (note different codes require different memory and computation time). Show, measure and discuss the followings:

- vocabulary size,
- bag-of-words histograms of example training/testing images,
- vector quantisation process.

Q2. [15] RF classifier

Train and test Random Forest using the training and testing data set in the form of bag-of-words obtained in **Q1**. Change the RF parameters (including the number of trees, the depth of trees, the degree of randomness parameter, the type of weak-learners: e.g. axis-aligned or two-pixel test), and show and discuss the results:

- recognition accuracy, confusion matrix,
- example success/failures,
- time-efficiency of training/testing,
- impact of the vocabulary size on classification accuracy.

Q3. [25] RF codebook

In **Q1**, replace the K-means with the random forest codebook, i.e. applying RF to 128 dimensional descriptor vectors with their image category labels, and using the RF leaves as the visual vocabulary. With the bag-of-words representations of images obtained by the RF codebook, train and test Random Forest classifier similar to **Q2**. Try different parameters of the RF codebook and RF classifier, and show/discuss the results in comparison with the results of **Q2**, including the vector quantisation complexity.

Use the above Caltech101 dataset, i.e. 10 classes, 15 images per class for training, and 15 other images per class for testing. The examples are colour images of object categories, each with varying size of pixels. Preprocess* the data so training CNNs becomes easier. (*size-normalise and or zero-center image pixel values).

Q4. [50] Convolutional Neural Networks

Train and test CNNs to classify the visual categorical images.

Design and describe an architecture for CNNs. Train the CNN using Caltech101 training examples and perform evaluation experiments.

You may try different network architectures (e.g. the number of layers, convolution filter sizes, skipping connections) and see how the performances change.

Try different batch normalisation techniques, and analyse their impacts.

When you train the NNs with drop-out, and or a regularisation term, do you observe improved generalisation to the testing data?

You might use the softmax as a default loss function and compare the squared hinge loss.

Try to compress your neural networks e.g. fully connected layers by truncated SVD and observe a trade-off between accuracy and parameter efficiency.

Change other major hyper-parameters (e.g. learning rates and batch sizes) and see how the accuracies change.

Do you train your CNNs from scratch i.e. with random weight initialisation or take initial weight values from a pre-trained CNN and fine-tune them using the Caltech101 training set? In which case, do you obtain higher accuracies?

Report the recognition accuracies on the Caltech101 testing set, in comparison to the success rates in **Q2, Q3**.

Find and demonstrate the parameters that lead to optimal performance and validate it by presenting supporting results. Give insights, discussions, and reasons behind your answers. Quality and completeness of discussions within the page limit will be marked. Include formulas where appropriate, results presented in figures, and their discussions. Try to visualise any interesting observations to support your answers.