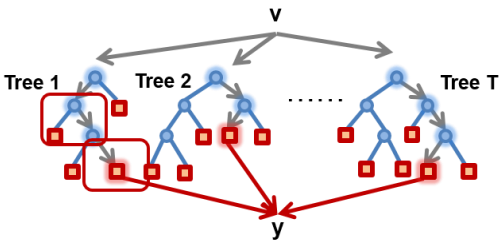
PCA, PCA-LDA Ensemble, SVM for face recognition

# *CS492* ***Machine Learning for Computer Vision***

***Coursework1*** on Manifold Learning, Online Learning, Discriminant Analysis, Ensemble Learning, Bag-of-words, Randomised Forests [45% mark]




**Release on 08 Oct 2021, the report due on 05 Nov 2021 (midnight)**

The coursework requires Python or Matlab programming.

**Submission instructions:**

One joint report by each pair

Page limit: 4-6 A4 pages per report with 10 font size (use the IEEE standard double column paper format, either in MS word or latex).

http://www.pamitc.org/cvpr16/files/egpaper_for_review.pdf

http://www.pamitc.org/cvpr16/files/cvpr2016AuthorKit.zip

Explain physical meanings and discuss with insights behind your answers, within the scope of lectures. ***Quality and completeness of discussions within the page limit*** will be marked.

Source code is not mandatory, unless specified. Optionally, this can go to appendices, which do not count for the page limit.

General principles for writing technical reports are expected to be known and adhered to. Similarly, for practices in conducting experiments, some are as listed below:

- Select/discuss relevant results that support the points you want to make rather than everything that the program code gives.
- The important results should be in the report, not just in the appendix.
- Use clear and tidy presentation style, consistent across the report e.g. figures, tables.
- The experiments should be described such that there is no ambiguity in the settings, protocol and metrics used.
- The main points are made clear, identifying the best and the worst case results or other important observations.

- Do not copy standard formulas from lecture notes, explain algorithms in detail, or copy figures from other sources. References to lecture slides or publications/webpages are enough in such cases, however short explanations of the terms or parameters referred to are needed.
- Find and demonstrate the parameters that lead to optimal performance and validate it by presenting supporting results.
- Include formulas only where appropriate, results presented in figures, and their discussions. Try to visualise any interesting observations to support your answers.

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

Honggyu Choi (honggyuchoi@kaist.ac.kr)

Jihyun Lee (jyun.lee@kaist.ac.kr)

## Experiment with face dataset

Use the provided face data (face.mat), which stores raster-scanned face images (46x56 pixels) in columns. In all questions, you can use any existing toolbox/code, if needed.

### Q1. [10] Computationally Efficient Eigenfaces

Partition the provided face data into your training and testing data, e.g. 8 images and 2 other images of each face identity for training and testing respectively.

Apply PCA to your training data i.e. compute the eigenvectors and eigenvalues of the covariance matrix $S=(1/N)AA^T$ directly, and use the low-dimensional computation of eigenspace i.e. using $(1/N)A^TA$ of your training data.

Show and discuss, in comparison, including: the eigenvectors and eigenvalues, the mean image, how many eigenvectors are with non-zero eigenvalues, if the eigenvectors and eigenvalues obtained by the two methods are identical, what are the pros/cons of each method. Give physical meanings and show respective *measurements* for your answers.

>> Hereinafter, we use the low-dimensional PCA technique, wherever appropriate.

Do face image reconstruction while varying the number of PCA bases learnt. Show and discuss the results quantitatively and qualitatively for different face images.

### Q2. [10] Incremental PCA

Use the same data partition into training and testing as in Q1. Further divide the training data equally into four subsets, each with 104 images (i.e. two images per person). Starting with the first subset, keep adding a more subset into your training.

Perform Incremental PCA, and compare it with the counterpart i.e. batch PCA, and PCA trained only by the first subset, in terms of *training time, reconstruction error, face recognition accuracy (using NN classification).

Show and discuss, including: how accurate your incremental method is, what important parameters in the method are (and how they are set). Provide your own discussions and measurements to support.

*Note: You might ignore all other computations e.g. constructing covariance matrices, orthonormalization, matrix products, (which can be accelerated by proper implementations) than eigen-decompositions.

## Q3. [10] PCA-LDA for Face Recognition

Use the provided face data, and the same data partition into training and testing as in Q1.

### PCA-LDA

Perform the PCA-LDA based face recognition with the NN classifier. Report and discuss, in comparison to the PCA method, including:

- recognition accuracies (success rates) by varying the hyper-parameter values, e.g. M_pca and M_lda
- ranks of the scatter matrices,
- the confusion matrix, example success and failure cases
- time/memory

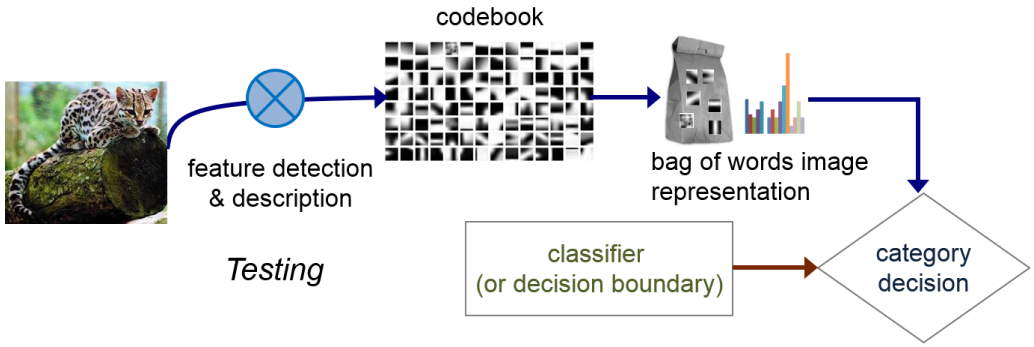
Give insights and reasons behind your observations.

### (optional) PCA-LDA Ensemble

Show, measure and discuss the results, by varying its parameter values. Give insights and reasons behind your answers, including:

- randomisation in feature space
- the number of base models, the randomness parameter,
- the error of the committee machine vs the average error of the individual models
- fusion rules (majority voting and sum)
- recognition accuracy and confusion matrix

## 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

### Q4. [5] K-means codebook

Listing 1. K-means Codebook Construction

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.

Listing 2. Random Forest Embeddings Codebook Construction

### Q5. [10] RF classifier

Train and test Random Forest using the training and testing data set in the form of bag-of-words obtained in **Q4**. 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.

## (optional) RF codebook

In **Q4**, 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 **Q5**. Try different parameters of the RF codebook and RF classifier, and show/discuss the results in comparison with the results of **Q5**, including the vector quantisation complexity.