MCV-M3: Image Classification (week 2)

Jonatan Poveda, Martí Cobos

Image Classification Pipeline

Train Database

Input Images and Labels

Feature Extraction

SIFT descriptor

Vocabulary Learning

k-means clustering

Global image representation

BoVW framework

Classifier Training

SVM classifier

Classification model

Test Database

Input Images and Labels

Feature Extraction

SIFT descriptor

Global image representation

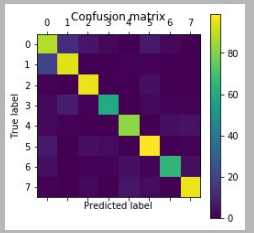
BoVW framework

Classify Image

SVM classifier

Evaluate Code Performance

Compute metrics using ground truth information



Task 1: Modularize the code

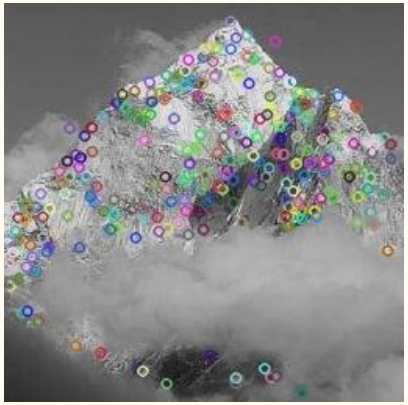
- Implemented various classes to separate code by domain (reusing some code from week 1)
  - Database: Loading and saving features from memory
  - FeatureExtractor: SIFT feature extraction.
  - Evaluator: Evaluate and plot code performance evaluation
  - BoVW: implementation of bag of visual words framework.
- Generalize code by implementing inheritance and polymorphism
  - Allows to switch descriptor or classifier with minimal changes in the code
- Database improve speed for testing multiple classificators
  - Allows to load already computed descriptors instead of recomputing each test

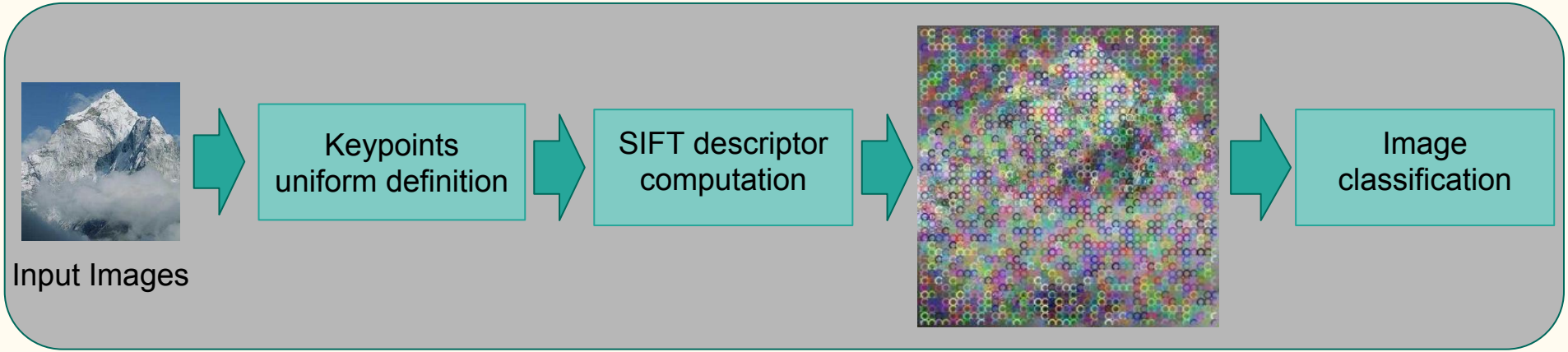
Task 2: Implement dense SIFT

Dense SIFT:

- Normal SIFT implementation computes features on corner detections. (See right image).
- Dense SIFT implemented by selecting keypoints uniformly spaced on the image. Python implementation using cv2.FeatureDetector_create("Dense") function
- Use standard SIFT descriptor to obtain the features on the selected keypoints.
- Dense SIFT improves accuracy from 0.66 to 0.79 (cross-validation average).

Standard SIFT:

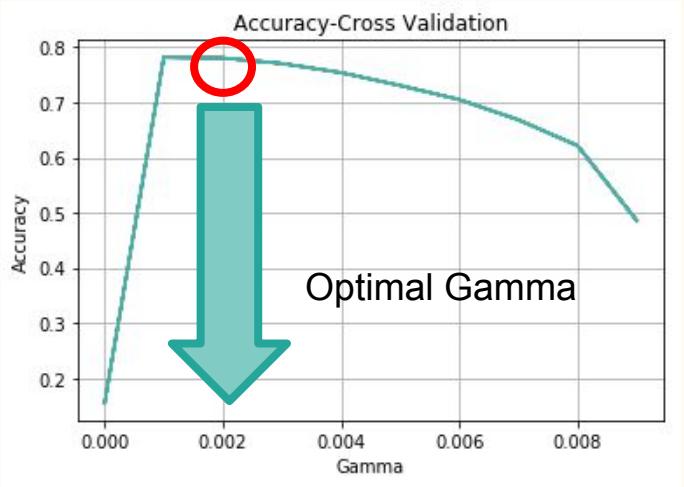




Task 3: Parameters validation with Cross-Validation

Cross-Validation

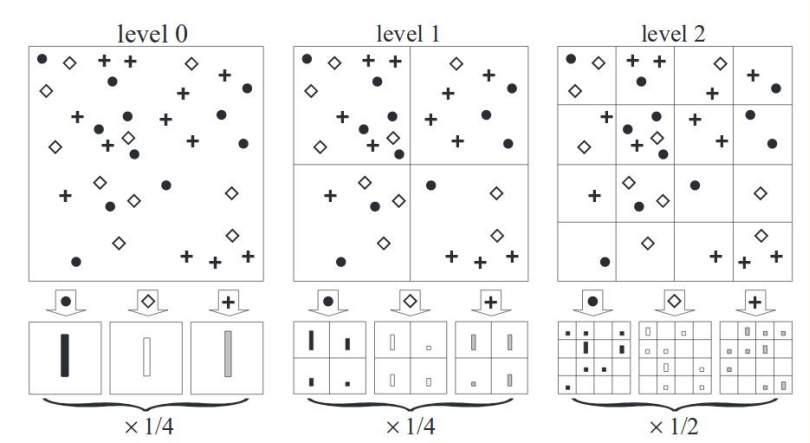
- In order to determine best parameters analyzing the train image set and validate them, cross validation has been implemented using Sklearn GridSearchCV function with 5 stratified folds.
- Several SVM kernels (lineal and radial “basis function”) and parameters are tested to achieve best results.
- For Dense SIFT feature extractor, the optimal kernel is “radial basis function” with the following parameters: $C = 10$; $\gamma = 0.002$ achieving an accuracy of 0.79.
- Right figure shows average test accuracy appling cross-validation while sweeping gamma value.



Task 4: Spatial Pyramid Implementation

Spatial pyramids

- In order to add spatial information to the BoVW frameworks, spatial pyramids are implemented. At first level, the image is divided in 4 four equals regions and histograms of words are computed on each region. Then a 2nd level of histograms is computed dividing the original image into 16 regions.
- Finally the resulting histograms are concatenated with specific weights.
- The image on the right summarizes the procedure:



Task 5: Histogram intersection kernel for SVM

Custom histogram intersection kernel implementation

- A custom implementation of the histogram intersection kernel has been implemented in order to improve the computing time of the SVM classifier.
- Histogram intersection implemented using following equation:

$$K_{int}(A, B) = \sum_{i=1}^{m} \min\{a_i, b_i\}.$$

- Cross-validation is applied to the train set in order to validate performance and compare with other SVM kernels. Results are shown on following table:

Pixel based	Accuracy	Precision	Recall	F1-score	time[s]
Method 1	0.793	0.799	0.793	0.796	579.2
Method 2	0.844	0.853	0.844	0.849	1104.3

- Method 1: Dense SIFT + SVM (RBF kernel)
- Method 2: Dense SIFT + Spatial Pyramids + SVM (histogram intersection kernel)

Task 5: Fisher Vector

- Implementing GMMs with all the dataset is not possible due to memory limitations.
  - An smaller dataset is built with 50 images per class (400 images in total instead of 1889), therefore the results are not comparable to the former ones.
- We run out of time to re-implement with optimizations and to implement fisher vectors as well. Even though, the partial results we got are in the following slide as Method 4 and 5.

Classification performance evaluation (on test images)

Pixel based	Accuracy	Precision	Recall	F1-score	time[s]
Method 1	0.705	0.714	0.708	0.711	547.41
Method 2	0.793	0.799	0.793	0.796	579.2
Method 3	**0.844**	**0.853**	**0.844**	**0.849**	**1104.3**
Method 4	0.663	0.671	0.667	0.669	9.46*
Method 5	0.711	0.716	0.718	0.717	15.45*

* tests computed using different hardware

Algorithm performance evaluated on test images using the following pipelines:

- Method 1: SIFT + k-means + SVM (RBF kernel-parameters obtained through cross-validation)
- Method 2: dense SIFT + k-means + SVM (RBF kernel-parameters obtained through cross-validation)
- **Method 3: dense SIFT + k-means + spatial pyramids + SVM (histogram intersection kernel)**
- Method 4: dense SIFT + GMM 32 clusters with soft assignment
- Method 5: dense SIFT + GMM 64 clusters with soft assignment

Conclusions

- Optimal SVM parameters and best combination of detector and descriptor determined using cross-validation on train images.
- Dense SIFT improves results of normal SIFT implementation due to have information from all image regions (not just corners).
- Spatial pyramids also improves results because the spatial information is taken into account when classifying an image.
- Best results achieved using method 3: dense SIFT + k-means + spatial pyramids + SVM (histogram intersection kernel)
- Accuracy results are expected to be further improved with the use of deep learning techniques.

Pixel based	Accuracy	Precision	Recall	F1-score	time[s]
Method 3	0.844	0.853	0.844	0.849	1104.3

Method 3: dense SIFT + k-means + spatial pyramids + SVM (histogram intersection kernel)

Conclusions

Best method Confusion Matrix plot

- Almost all classes are classified with high accuracy.
- The confusion matrix shows (see red rectangle) that OpenCountry and Coast image class are the most difficult to differentiate with the proposed implementation of BoVW (dense SIFT, spatial pyramids and histogram intersection)