Master in Computer Vision Barcelona

Module 3: Machine learning for computer vision

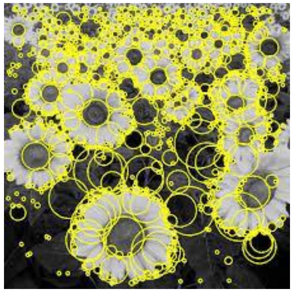
Project: Bag of Visual Words Image Classification

Lecturer: Ramon Baldrich, ramon.baldrich@uab.cat

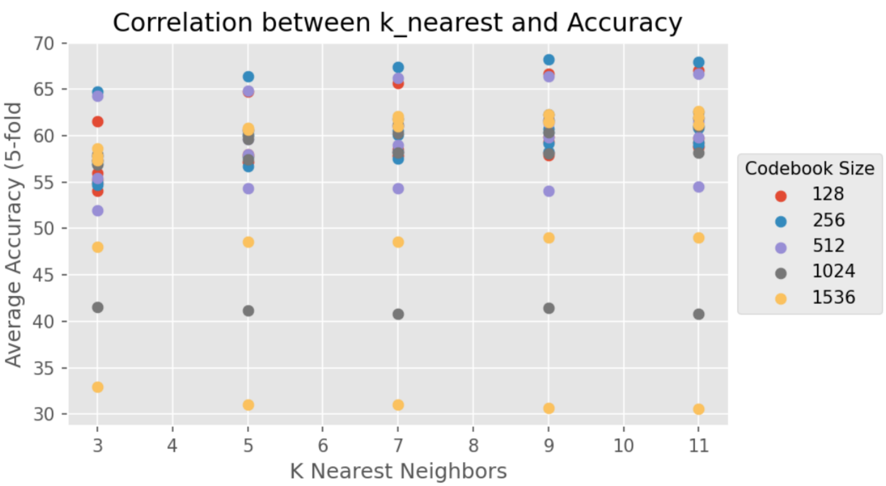
Credit to Marçal Rossinyol

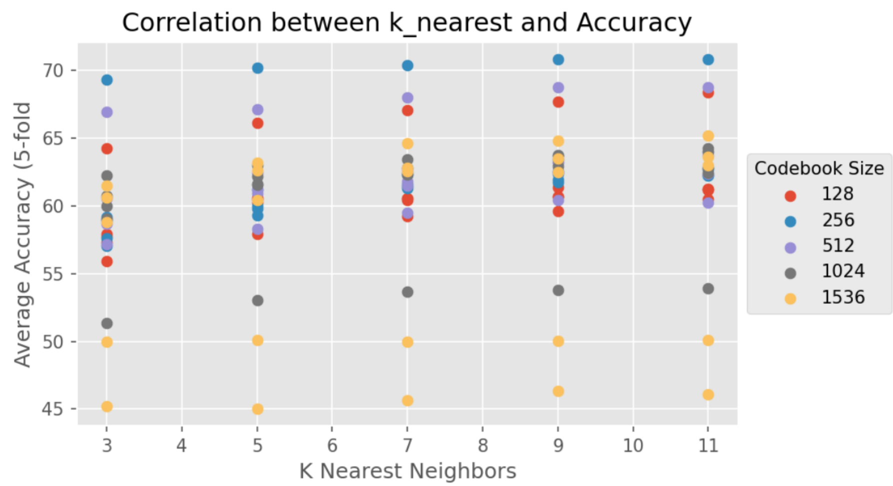
S01 discussion

- Number of keypoints
  - The more the better
- Dense SIFT
  - Nearly nobody tried the role of scales!
- Codebook sizes / k-nn value
- k-nn and distances
  - Just slight differences found between point-wise distances
  - Which distance would work better for HISTOGRAMS?
- Dimensionality reduction
- Precompute stuff, store to disk!



- Amount of points (kaze)

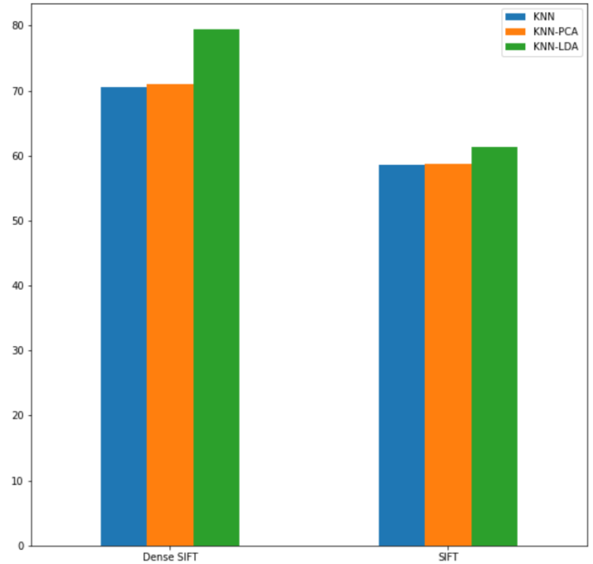


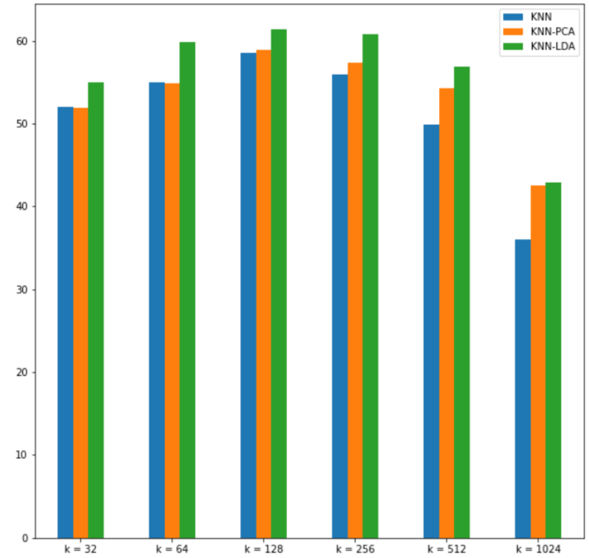


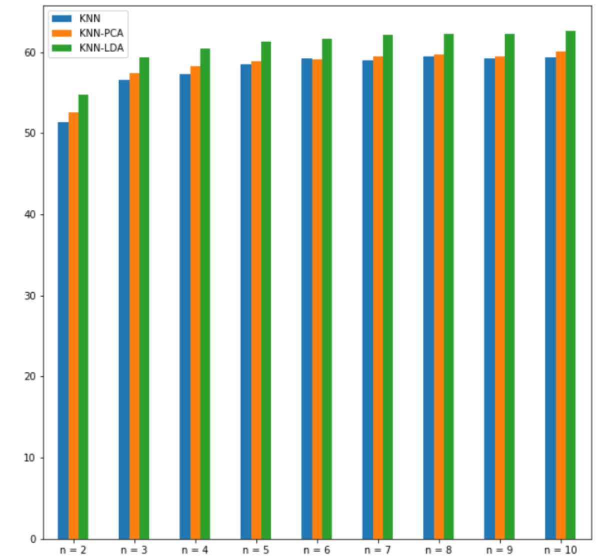
Dense vs Non Dense

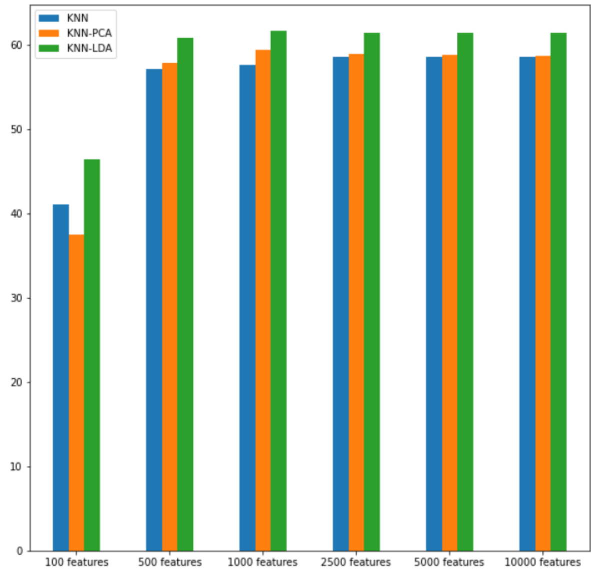
5000 kp

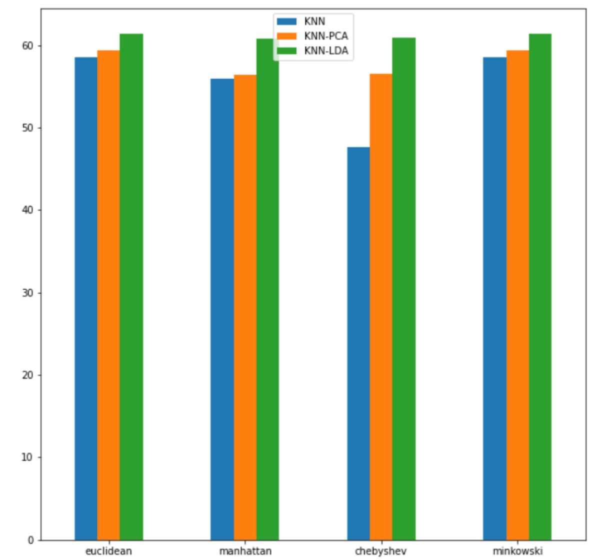
KNN

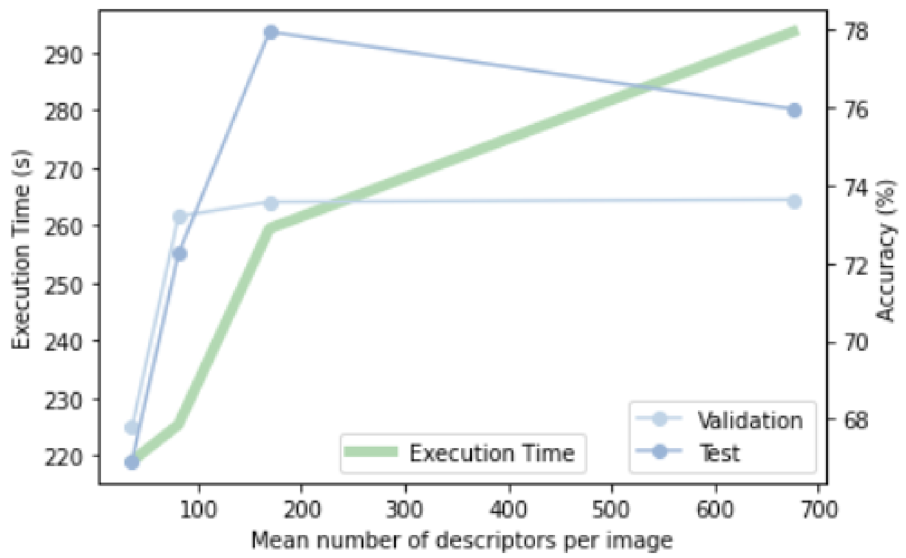


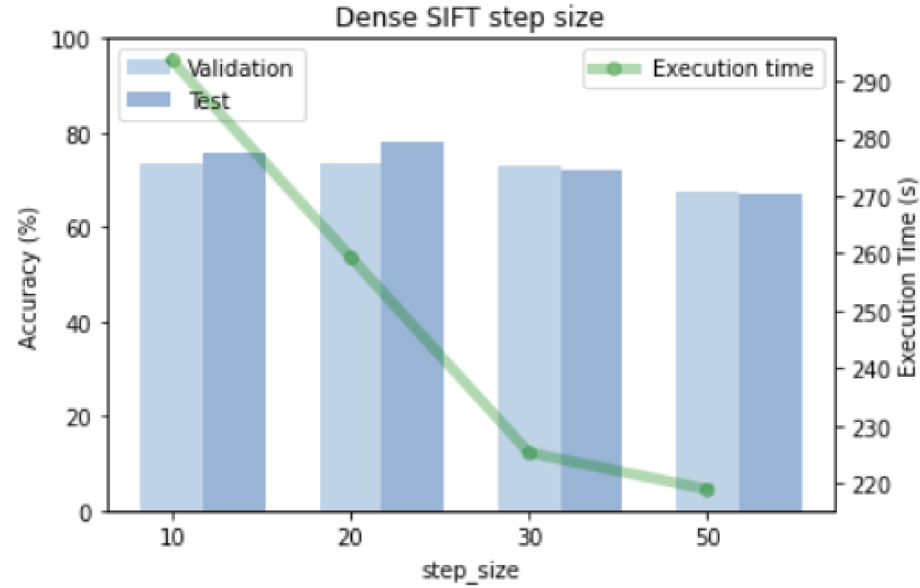


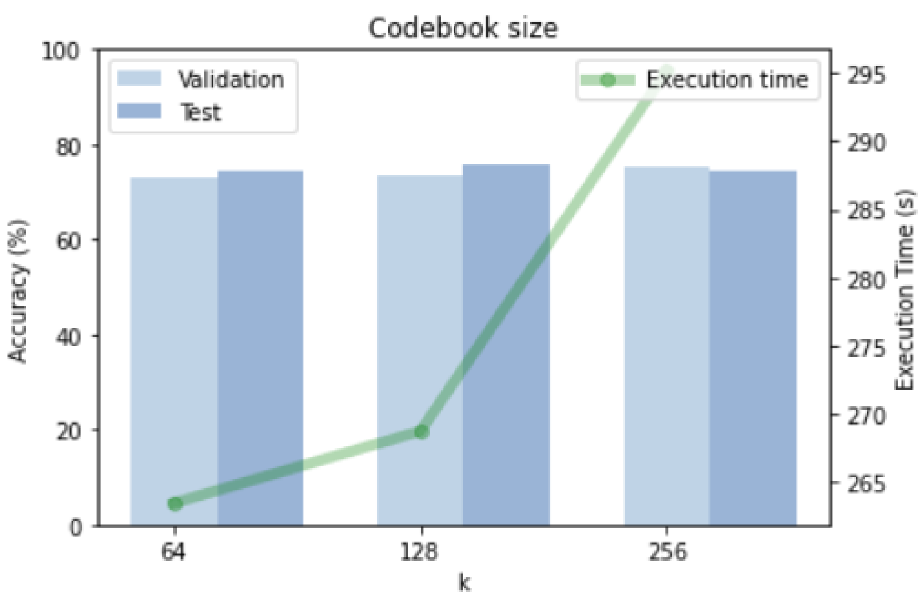


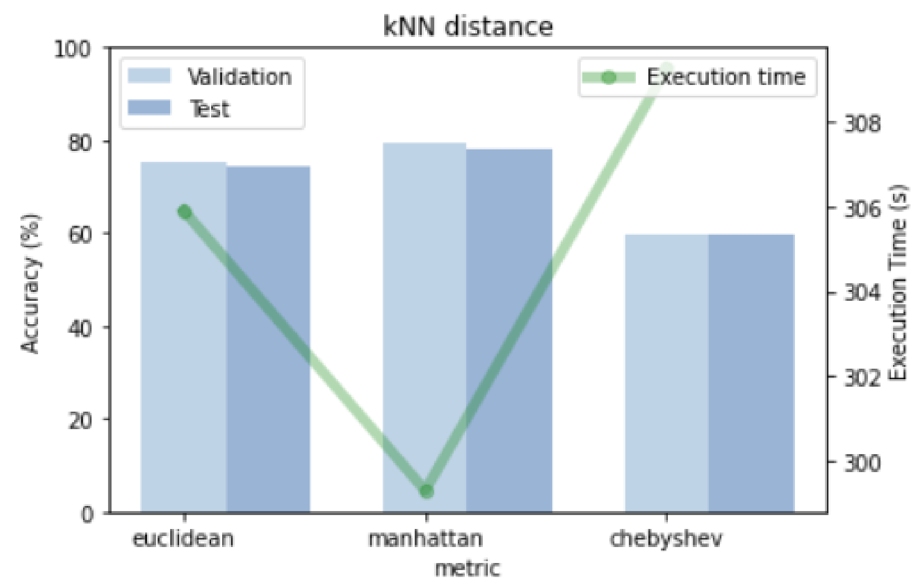


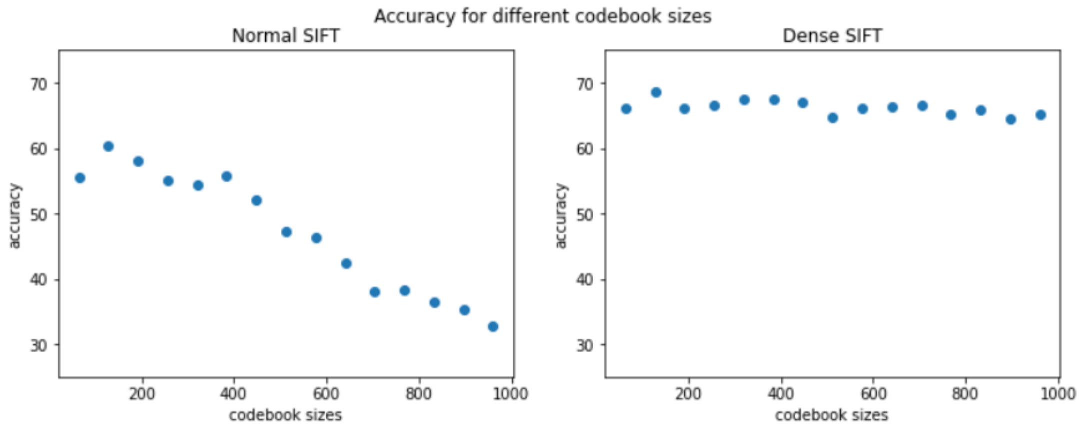


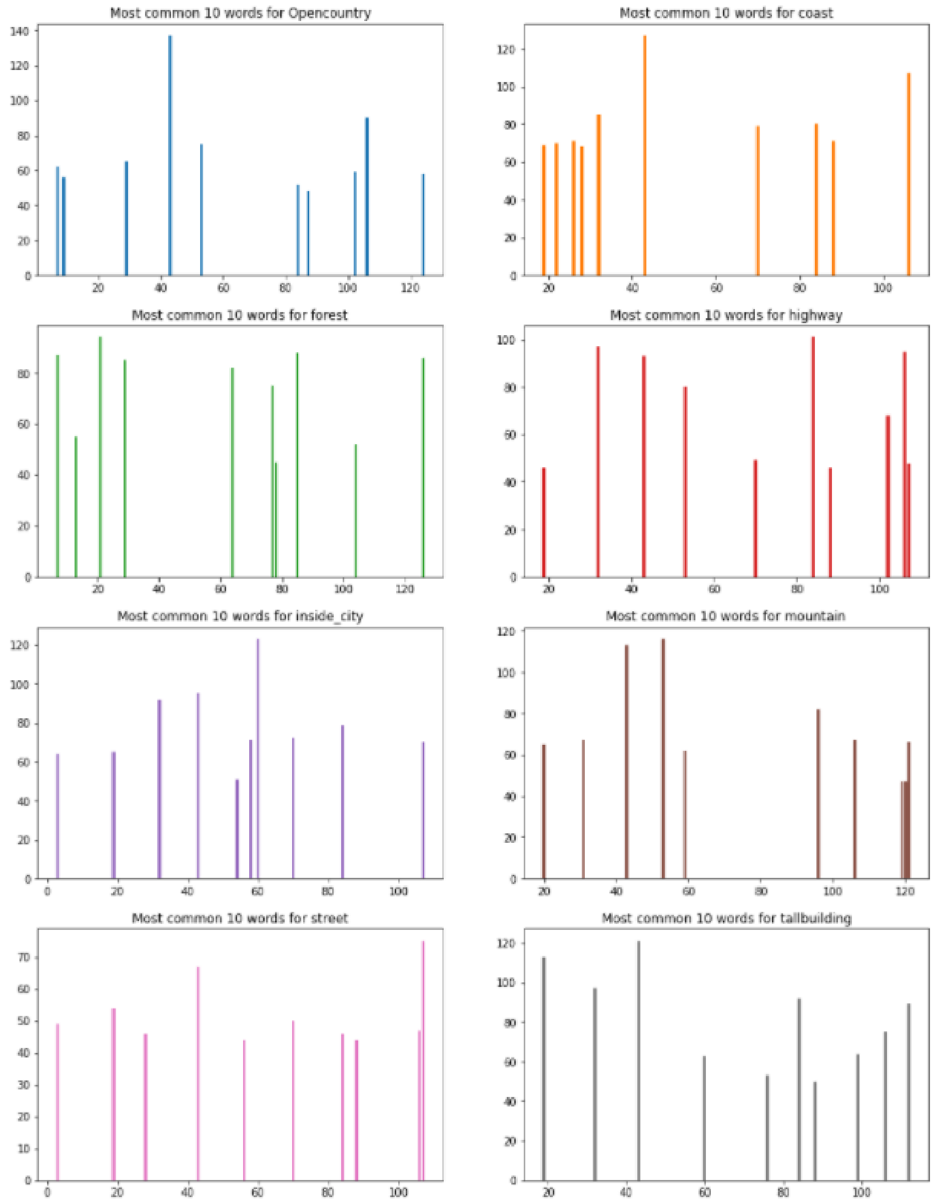

Execution Time (s)
Accuracy (%)
Mean number of descriptors per image
Validation
Test
Execution Time

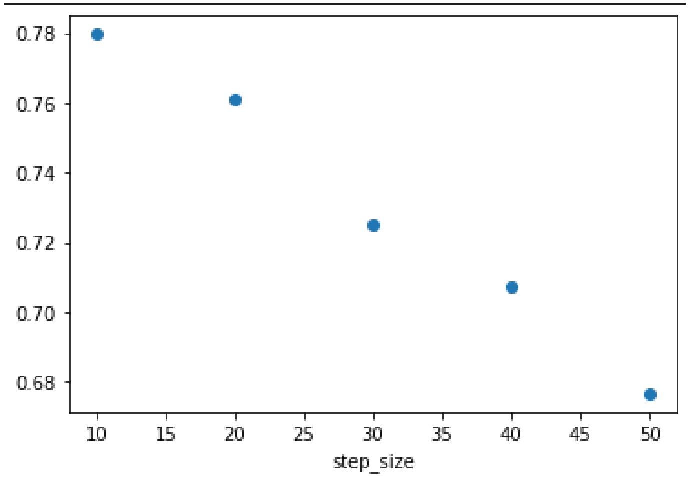

Dense SIFT step size
Validation
Test
Execution time
Accuracy (%)
Execution Time (s)
step_size

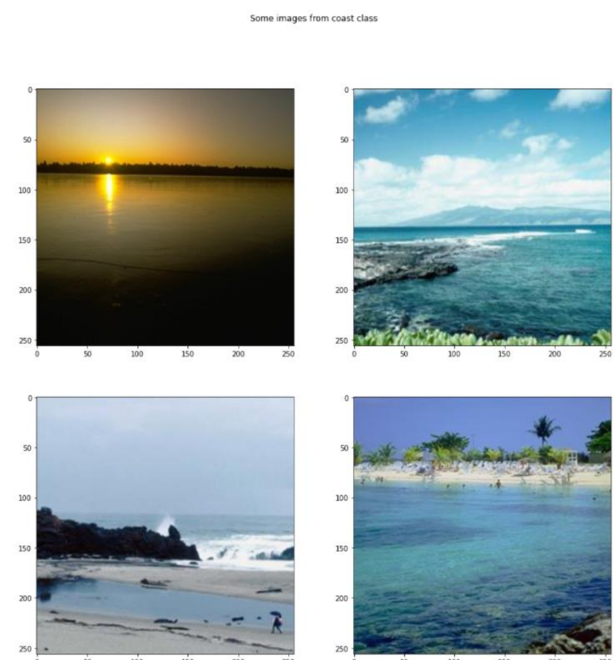

Codebook size
Validation
Test
Execution time
Accuracy (%)
Execution Time (s)
k


kNN distance
Validation
Test
Execution time
Accuracy (%)
Execution Time (s)
euclidean
manhattan
chebyshev
metric


Accuracy for different codebook sizes
Normal SIFT
Dense SIFT
accuracy
codebook sizes
70
60
50
40
30
200
400
600
800
1000


Most common 10 words for Opencountry
Most common 10 words for coast
Most common 10 words for forest
Most common 10 words for highway
Most common 10 words for inside_city
Most common 10 words for mountain
Most common 10 words for street
Most common 10 words for tallbuilding

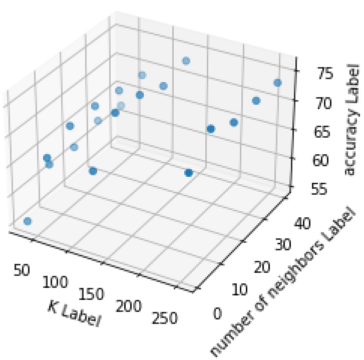


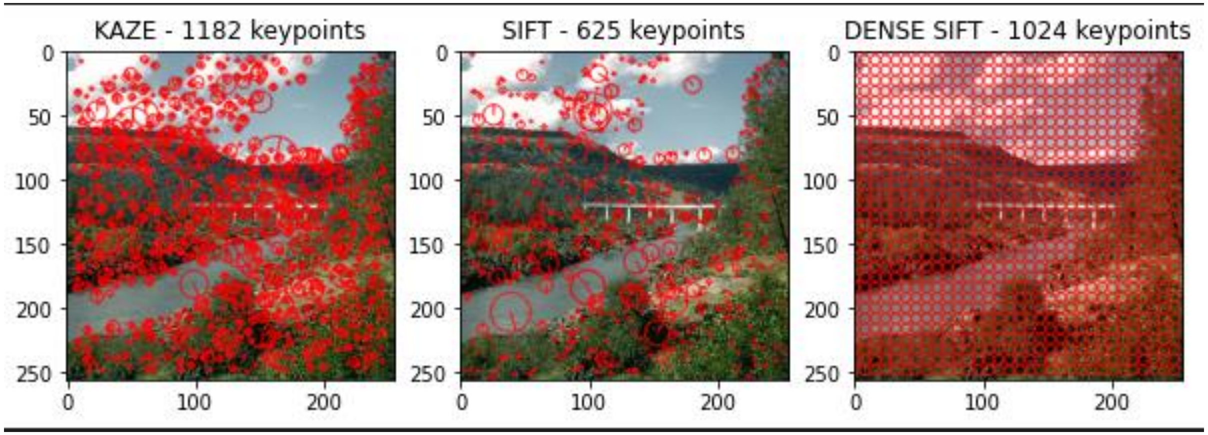


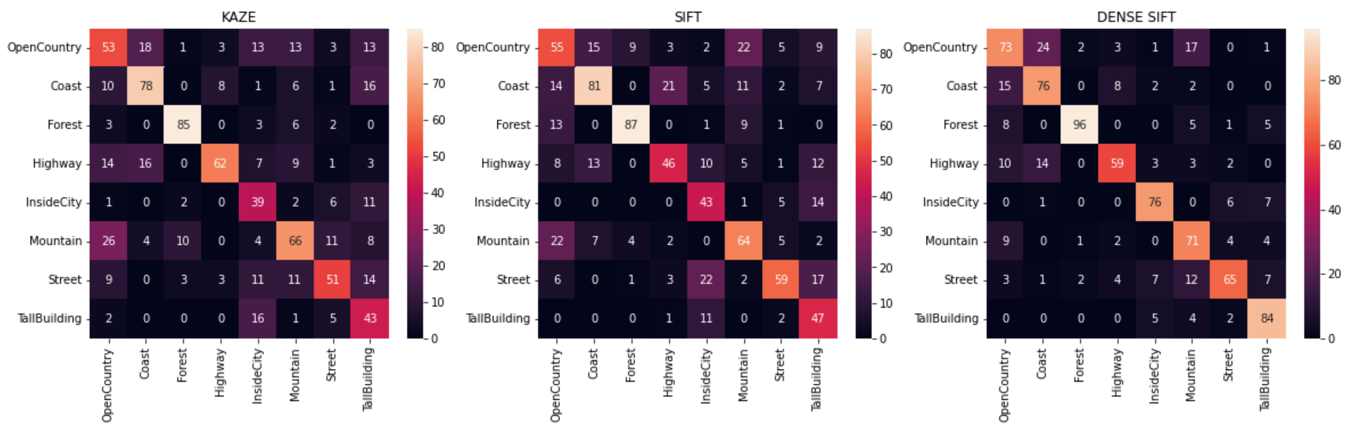

```
Train accuracy for LogReg: 0.9946836788942052
Test accuracy for LogReg: 0.7757125154894672
```

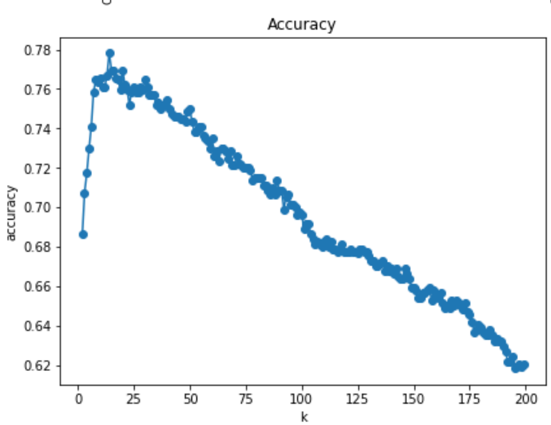
Train Accuracy: KNN = % 84, Logistic Regression = %92
Test Accuracy: KNN = % 79, Logistic Regression = %84

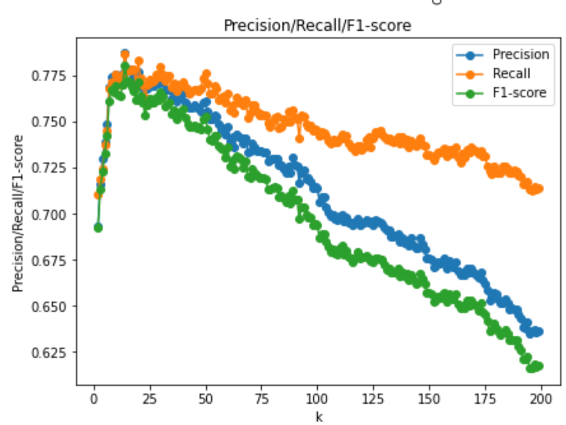

Image
Keypoints:450
Keypoints:710


75
70
65
60
55
accuracy Label
40
30
20
10
0
number of neighbors Label
50
100
150
200
250
K Label

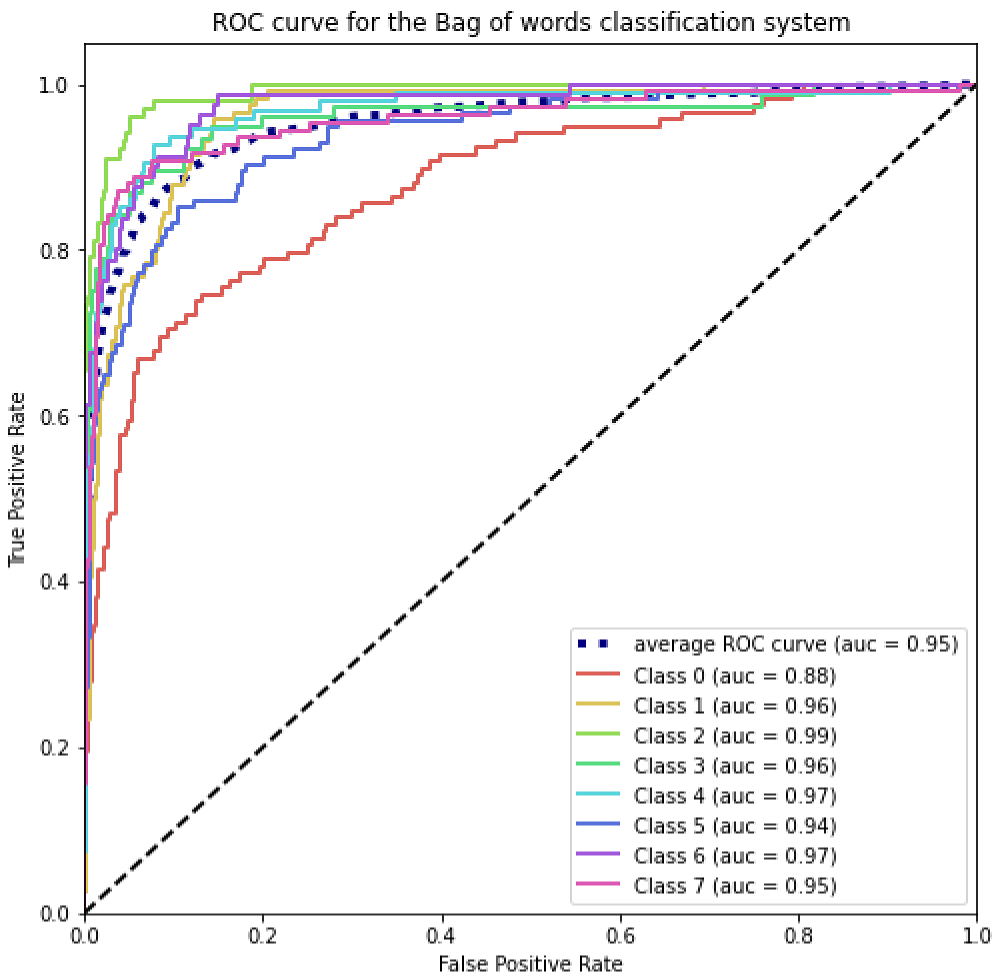

KAZE - 1182 keypoints
SIFT - 625 keypoints
DENSE SIFT - 1024 keypoints


KAZE
OpenCountry 53 18 1 3 13 13 3 13
Coast 10 78 0 8 1 6 1 16
Forest 3 0 85 0 3 6 2 0
Highway 14 16 0 62 7 9 1 3
InsideCity 1 0 2 0 39 2 6 11
Mountain 26 4 10 0 4 66 11 8
Street 9 0 3 3 11 11 51 14
TallBuilding 2 0 0 0 16 1 5 43
SIFT
OpenCountry 55 15 9 3 2 22 5 9
Coast 14 81 0 21 5 11 2 7
Forest 13 0 87 0 1 9 1 0
Highway 8 13 0 46 10 5 1 12
InsideCity 0 0 0 0 43 1 5 14
Mountain 22 7 4 2 0 64 5 2
Street 6 0 1 3 22 2 59 17
TallBuilding 0 0 0 1 11 0 2 47
DENSE SIFT
OpenCountry 73 24 2 3 1 17 0 1
Coast 15 76 0 8 2 2 0 0
Forest 8 0 96 0 0 5 1 5
Highway 10 14 0 59 3 3 2 0
InsideCity 0 1 0 0 76 0 6 7
Mountain 9 0 1 2 0 71 4 4
Street 3 1 2 4 7 12 65 7
TallBuilding 0 0 0 0 5 4 2 84
OpenCountry Coast Forest Highway InsideCity Mountain Street TallBuilding


Accuracy
accuracy
k


Precision/Recall/F1-score
Precision
Recall
F1-score
Precision/Recall/F1-score
k

What I missed

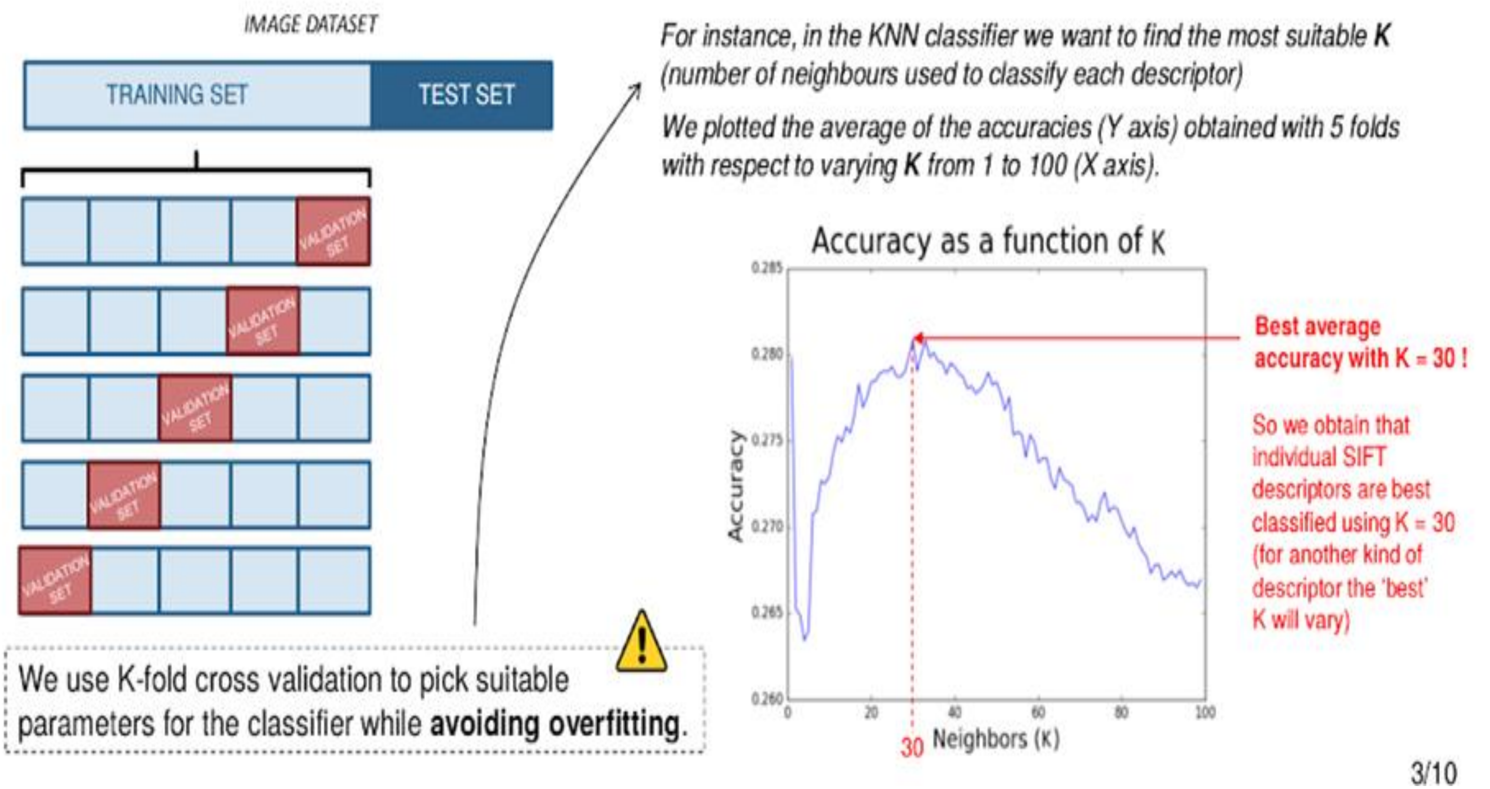


Group	grade
1	8
2	9
3	6
4	10
5	9
6	9
7	8
8	9

S02

- We'll start with BoVW computed with Dense SIFT with a large enough codebook size
- We'll normalize descriptors
  - L2-norm, Power-norm, etc..
- Cross-validation
  - Sklearn functions: StratifiedkFold, GridsearchCV
- Spatial Pyramids
- SVM and kernels
  - Use sklearn standardScaler to project every dimension to [0, 1]!
  - linear kernel
  - RBF kernel
  - our own histogram intersection kernel
- OPTIONAL: Fisher Vectors (http://yael.gforge.inria.fr/tutorial/tuto_imgindexing.html)

Cross Validation



Hyperparamter optimization

Journal of Machine Learning Research 13 (2012) 281-305 Submitted 3/11; Revised 9/11; Published 2/12

Random Search for Hyper-Parameter Optimization

James Bergstra JAMES.BERGSTRA@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Département d'Informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC, H3C 3J7, Canada

Editor: Leon Bottou

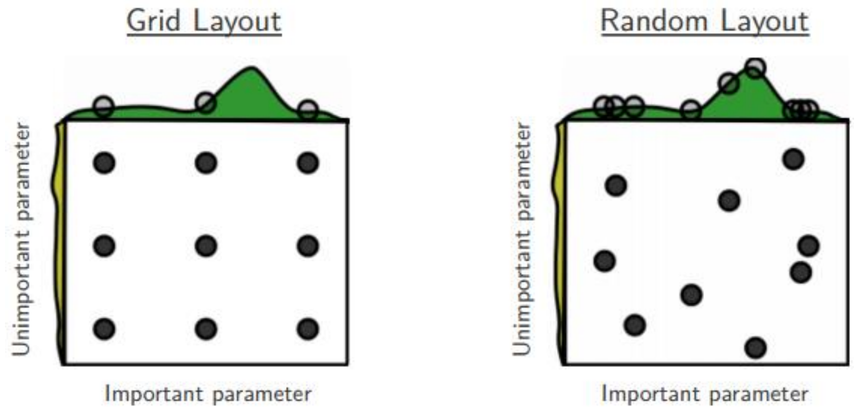
Abstract

Grid search and manual search are the most widely used strategies for hyper-parameter optimization. This paper shows empirically and theoretically that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid. Empirical evidence comes from a comparison with a large previous study that used grid search and manual search to configure neural networks and deep belief networks. Compared with neural networks configured by a pure grid search, we find that random search over the same domain is able to find models that are as good or better within a small fraction of the computation time. Granting random search the same computational budget, random search finds better models by effectively searching a larger, less promising configuration space. Compared with deep belief networks configured by a thoughtful combination of manual search and grid search, purely random search over the same 32-dimensional configuration space found statistically equal performance on four of seven data sets, and superior performance on one of seven. A Gaussian process analysis of the function from hyper-parameters to validation set performance reveals that for most data sets only a few of the hyper-parameters really matter, but that different hyper-parameters are important on different data sets. This phenomenon makes grid search a poor choice for configuring algorithms for new data sets. Our analysis casts some light on why recent "High Throughput" methods achieve surprising success—they appear to search through a large number of hyper-parameters because most hyper-parameters do not matter much. We anticipate that growing interest in large hierarchical models will place an increasing burden on techniques for hyper-parameter optimization; this work shows that random search is a natural baseline against which to judge progress in the development of adaptive (sequential) hyper-parameter optimization algorithms.

Keywords: global optimization, model selection, neural networks, deep learning, response surface modeling

1. Introduction

The ultimate objective of a typical learning algorithm $\mathcal{A}$ is to find a function f that minimizes some expected loss $\mathcal{L}(x; f)$ over i.i.d. samples x from a natural (grand truth) distribution $\mathcal{G}_x$. A *learning algorithm* $\mathcal{A}$ is a functional that maps a data set $\mathcal{X}^{(\text{train})}$ (a finite set of samples from $\mathcal{G}_x$) to a function



Continuous hyperparameter: distribution over possible values

generate random variable

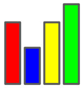
Discrete hyperparameter: list of discrete choices
random selection
(without replacement if all discrete)

Set the number of trials

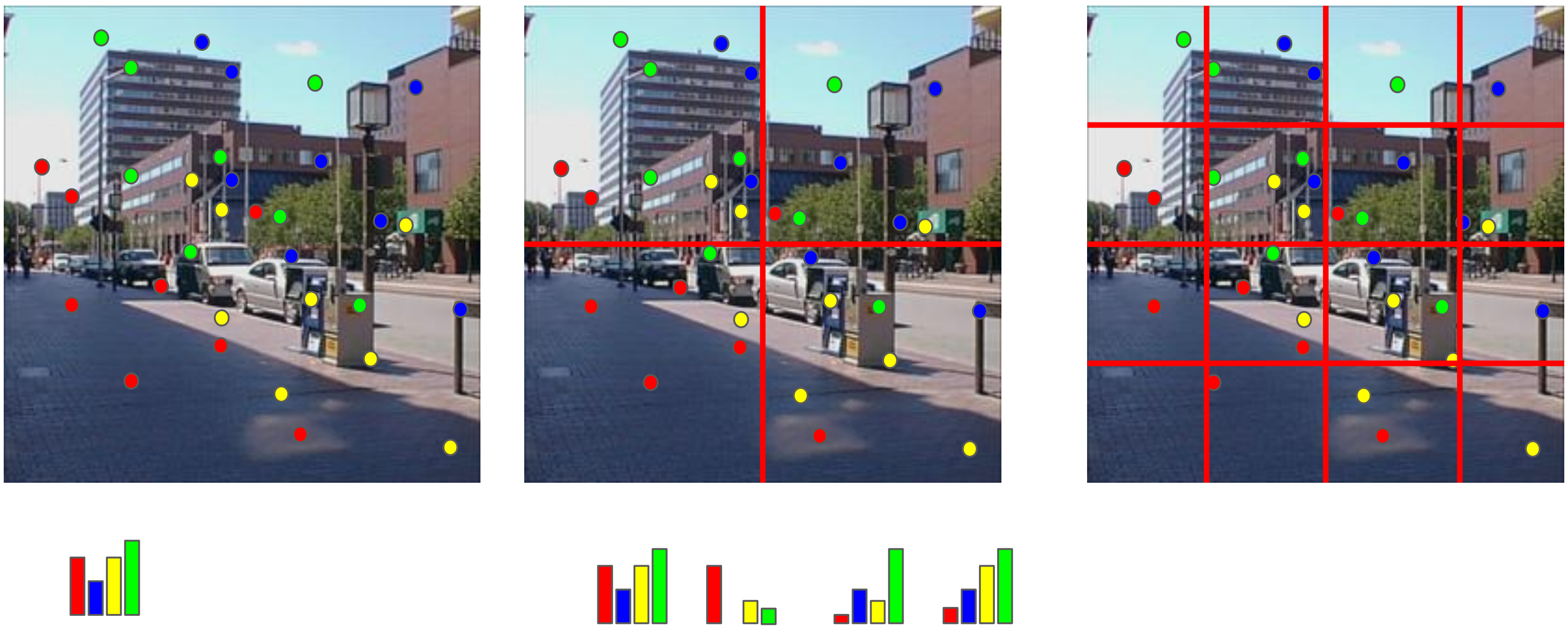
Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

```
params = {
    "C": np.arange(0.001, 1, 0.01),
    "max_iter": np.arange(50, 550, 50)
}

lg = LogisticRegression(solver='liblinear')

lg_grid = GridSearchCV(lg, params, cv=8,
                        scoring=["accuracy", "f1_macro"], refit="accuracy", return_train_score=True)
lg_grid.fit(visual_words, train_labels)
```

Spatial Pyramids



Spatial Pyramids



Histogram Intersection kernel

```
def histogramIntersection(M, N):
```

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

```
    return K_int
```

Tasks to do

Improve the BoVW code with:

- Dense SIFT (with tiny steps and different scales!)
- L2-norm - power norm
- SVM classifier
- StandardScaler
- Cross-validation
- Linear, RBF and histogram intersection kernels
- Spatial Pyramids
- Fisher Vectors (OPTIONAL)

Deliverable

- A **single Python notebook file per group** reporting all the work done,
  - with the different experiments,
  - code,
  - plots,
  - explanations, etc.
  - **EVERYTHING EXECUTED!**

- To deliver by Monday 17th @ 10 A.M.
  - Please, state clearly your group.

Warning: provided code might not work out of the box depending on the used versions (OpenCV, numpy, sklearn…) do not panic, and read the documentation