Master in Computer Vision Barcelona

Module 3: Machine learning for computer vision

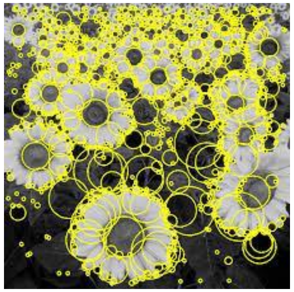
Project: Bag of Visual Words Image Classification

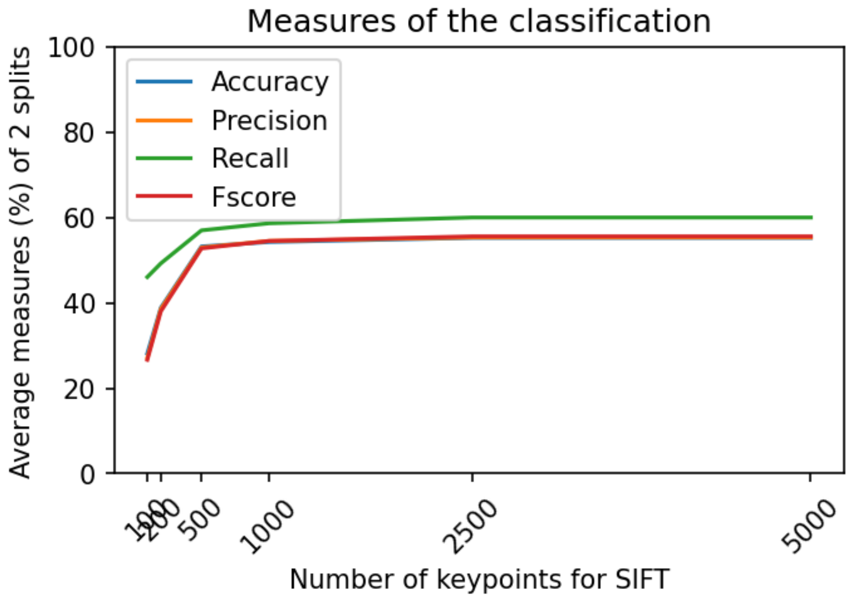
Lecturer: Ramon Baldrich, ramon.baldrich@uab.cat

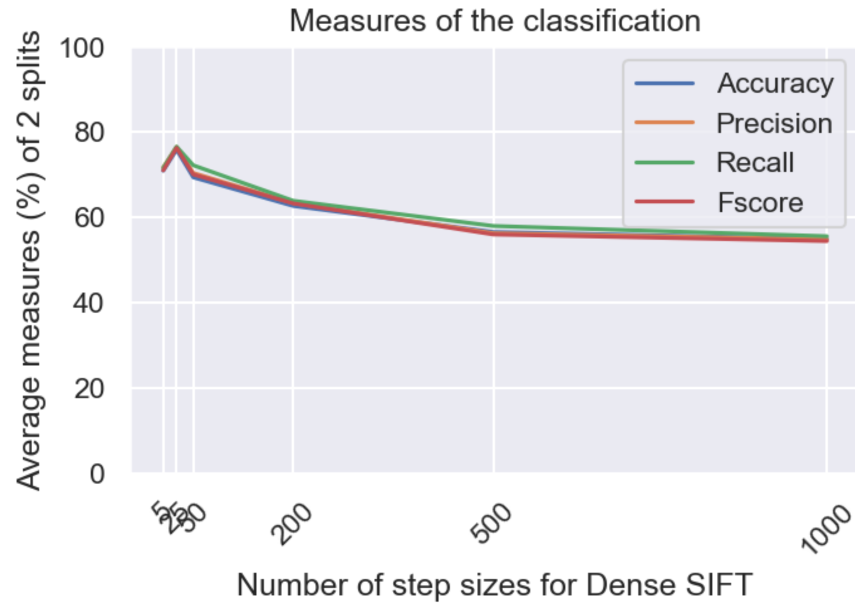
Credit to Marçal Rossinyol

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 not much effect
- Precompute stuff, store to disk!

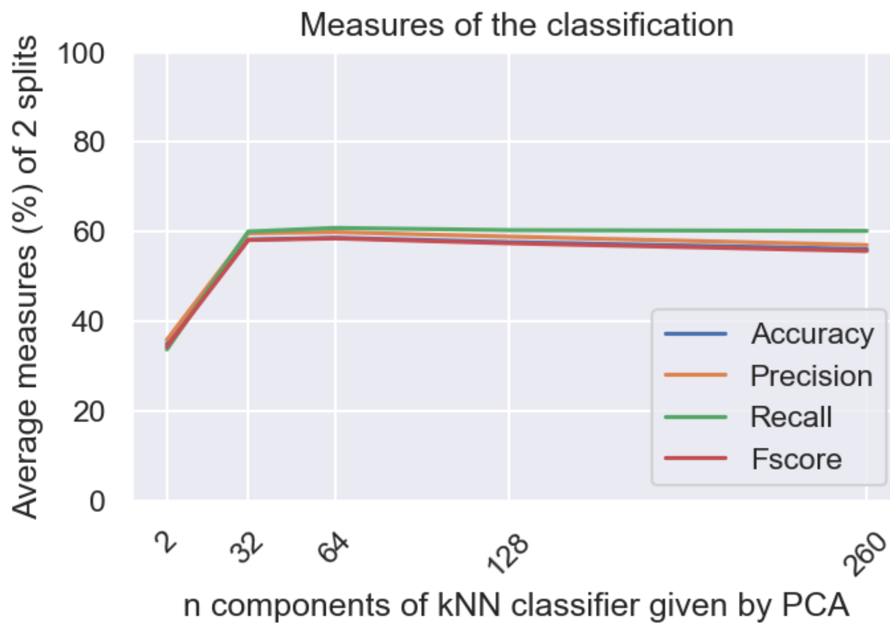



Measures of the classification
Average measures (%) of 2 splits
Accuracy
Precision
Recall
Fscore
100
80
60
40
20
0
100
200
500
1000
2500
5000
Number of keypoints for SIFT

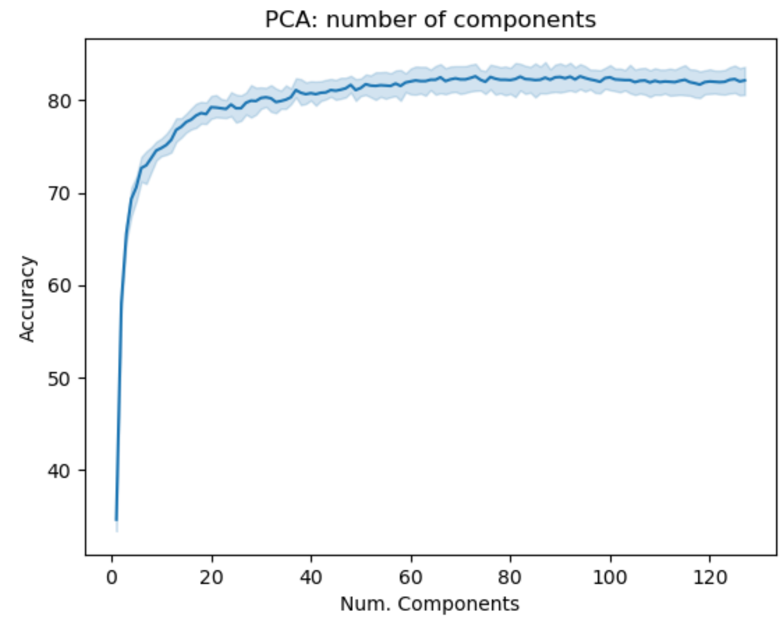

Measures of the classification
Average measures (%) of 2 splits
Accuracy
Precision
Recall
Fscore
5
25
50
200
500
1000
Number of step sizes for Dense SIFT


Dense SIFT: step size
Accuracy
Step size
76
74
72
70
68
66
64
62
60
10
20
30
40
50
60

t-SNE of the visual words used by the k-NN
Opencountry
coast
forest
highway
inside_city
mountain
street
tallbuilding

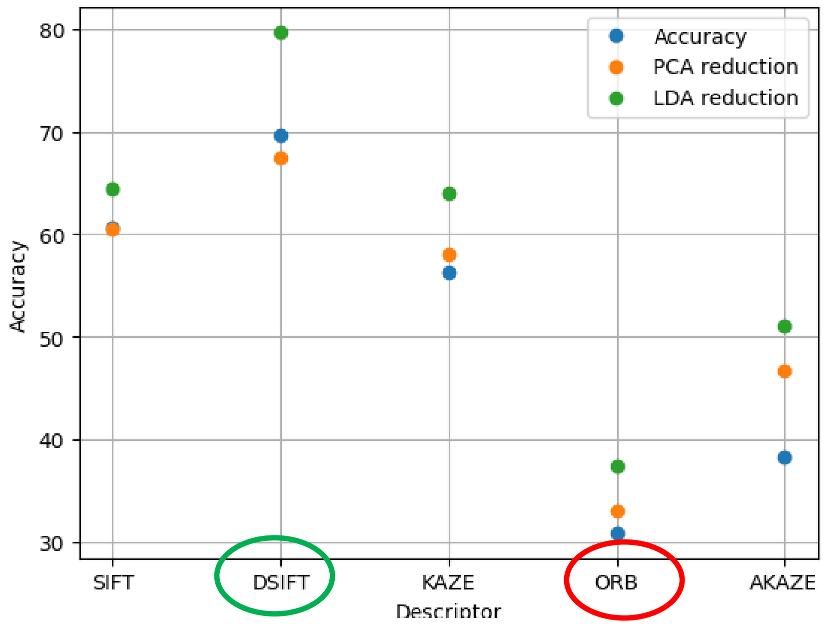


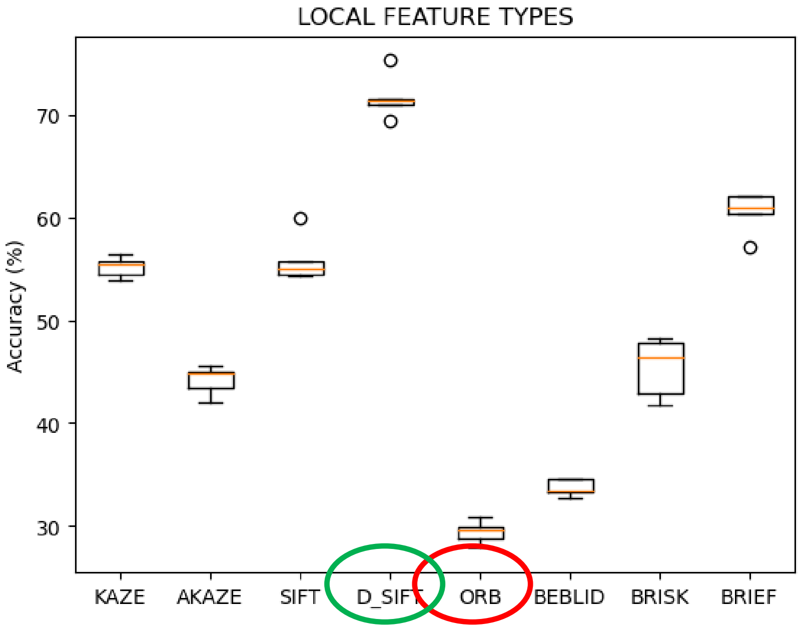
Measures of the classification
Average measures (%) of 2 splits
Accuracy
Precision
Recall
Fscore
n components of kNN classifier given by PCA



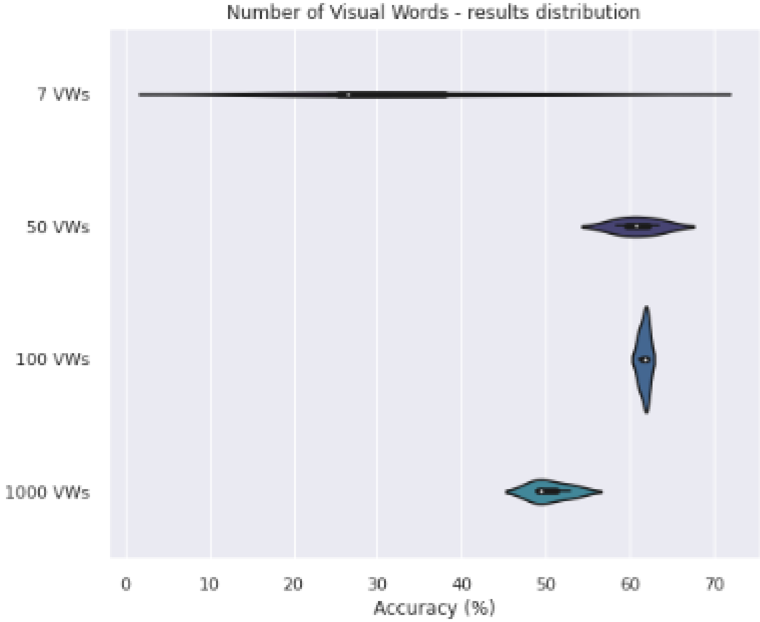
PCA: number of components
Accuracy
Num. Components

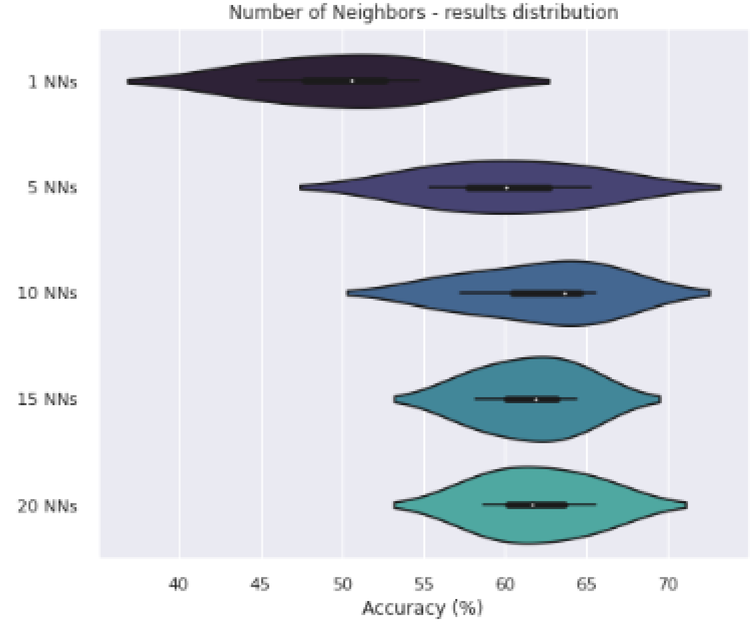
The role of the descriptor

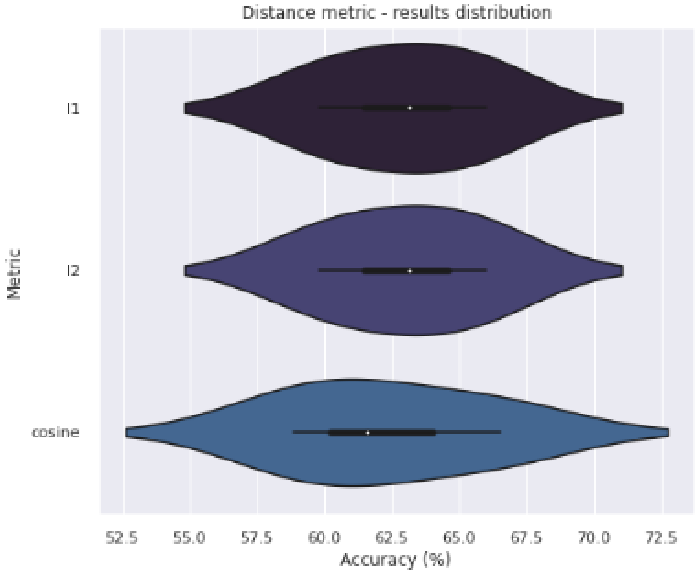


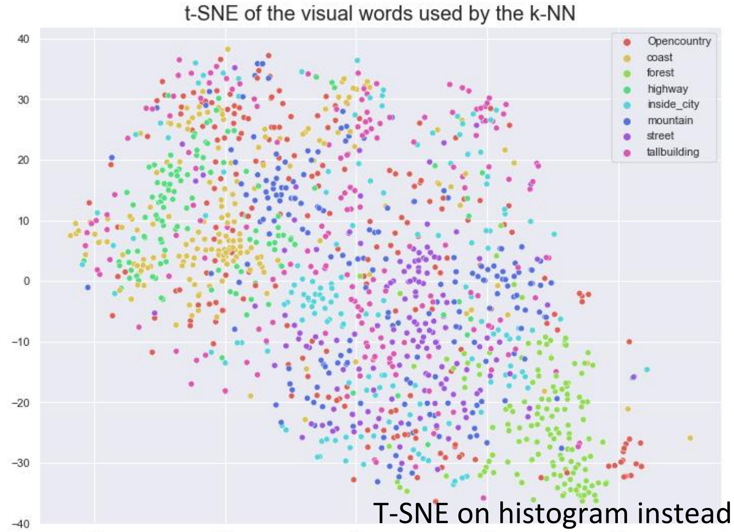


Understanding parameter effects on performance

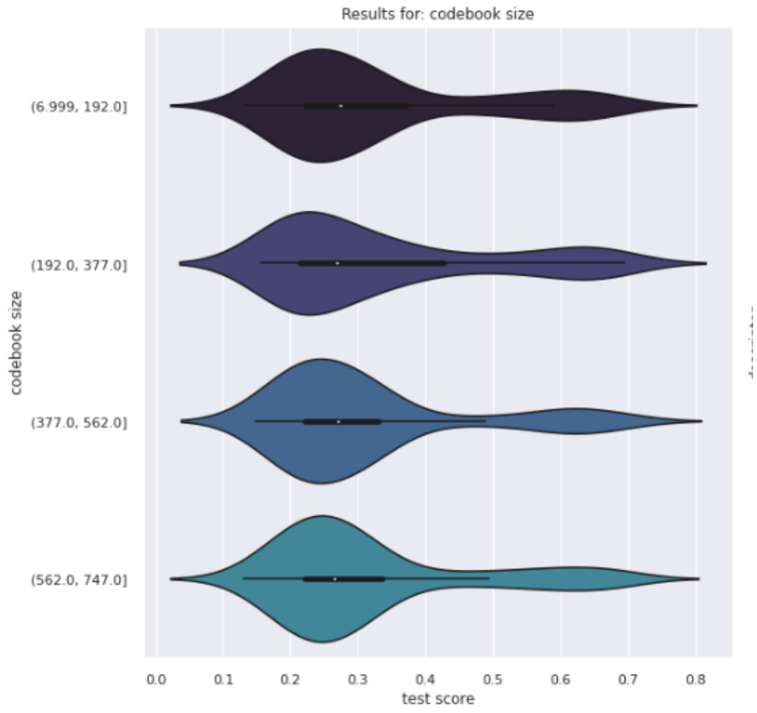






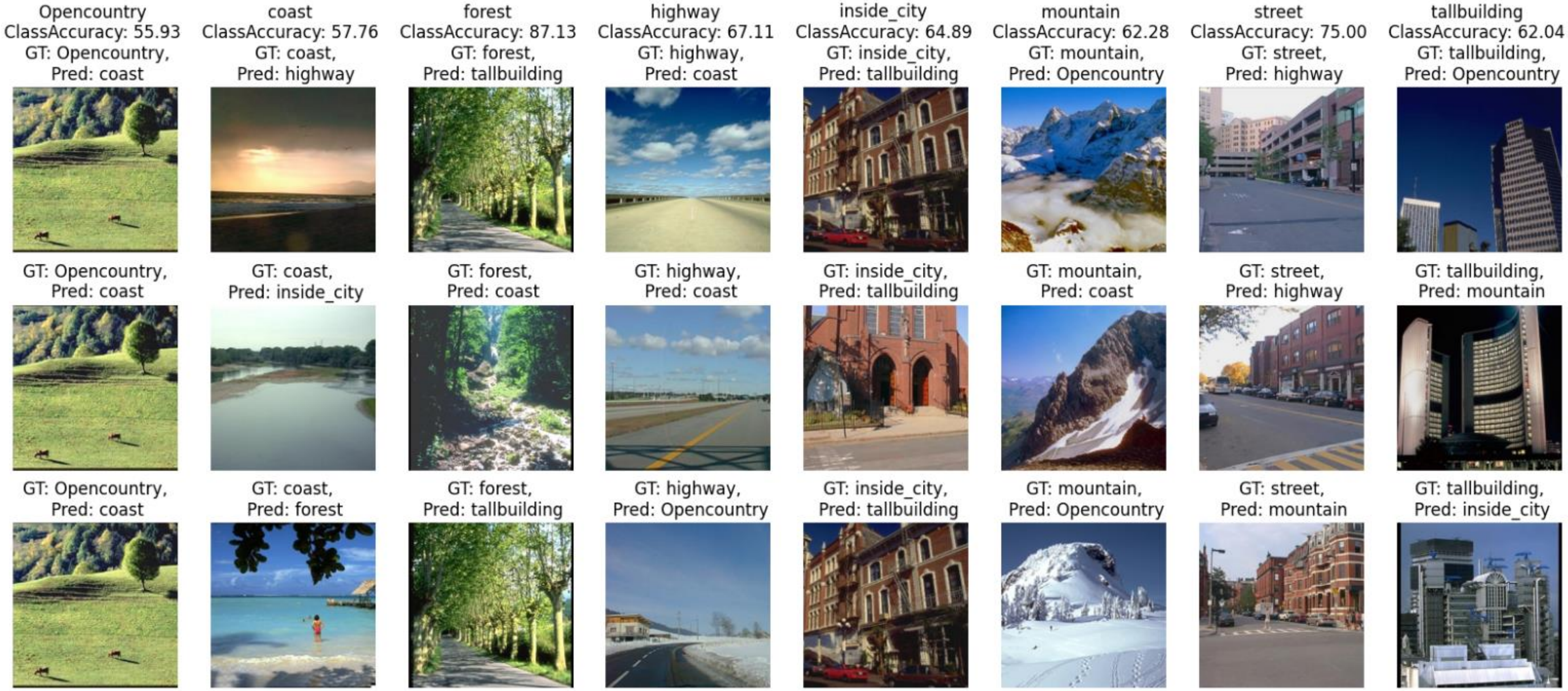


T-SNE on histogram instead?

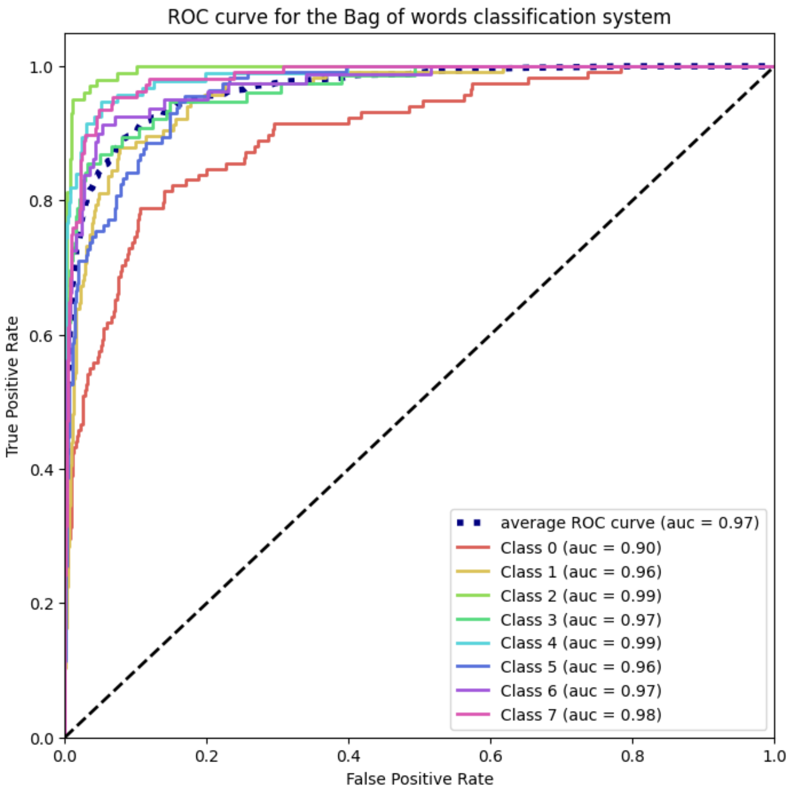


Don't compare parameters where other parameters change

Visualize results (intermediate and final)

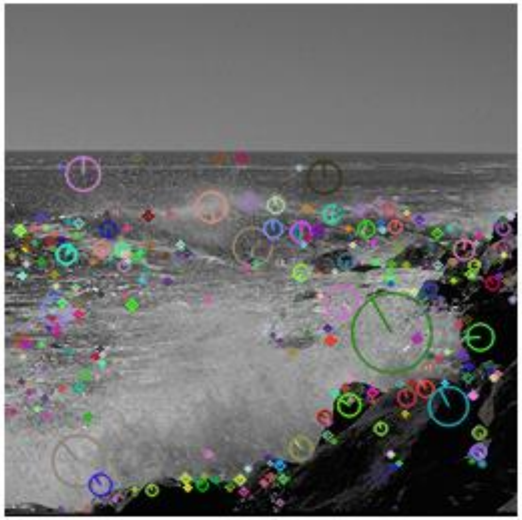


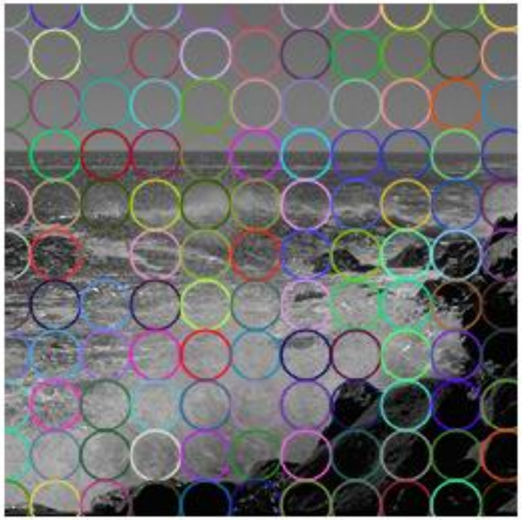
Final performance



or precession/recall curve depending on the task

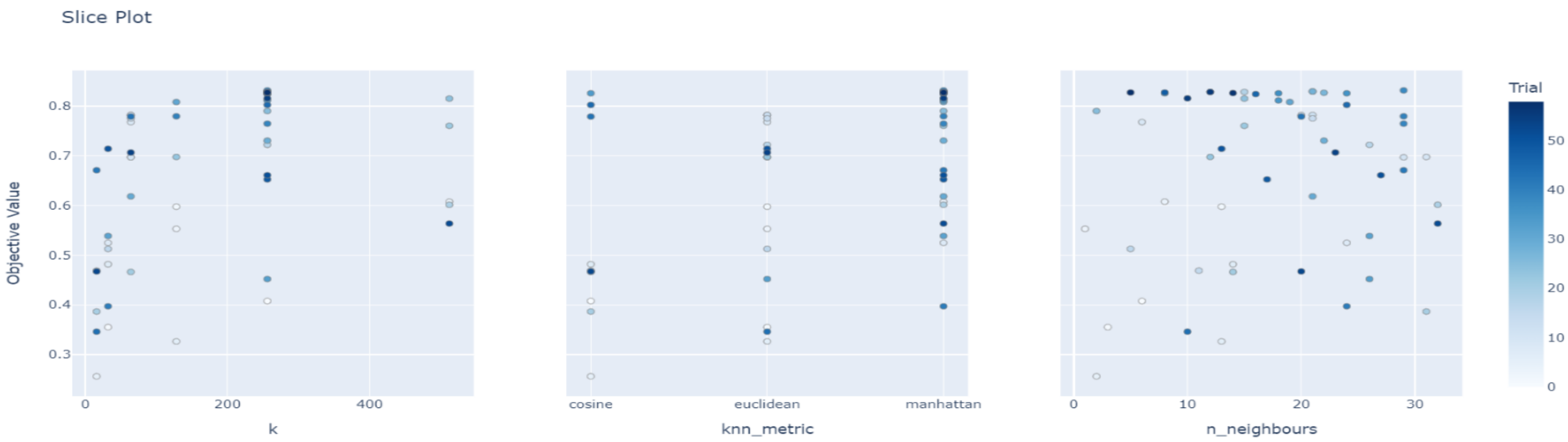
Does scale play any role?





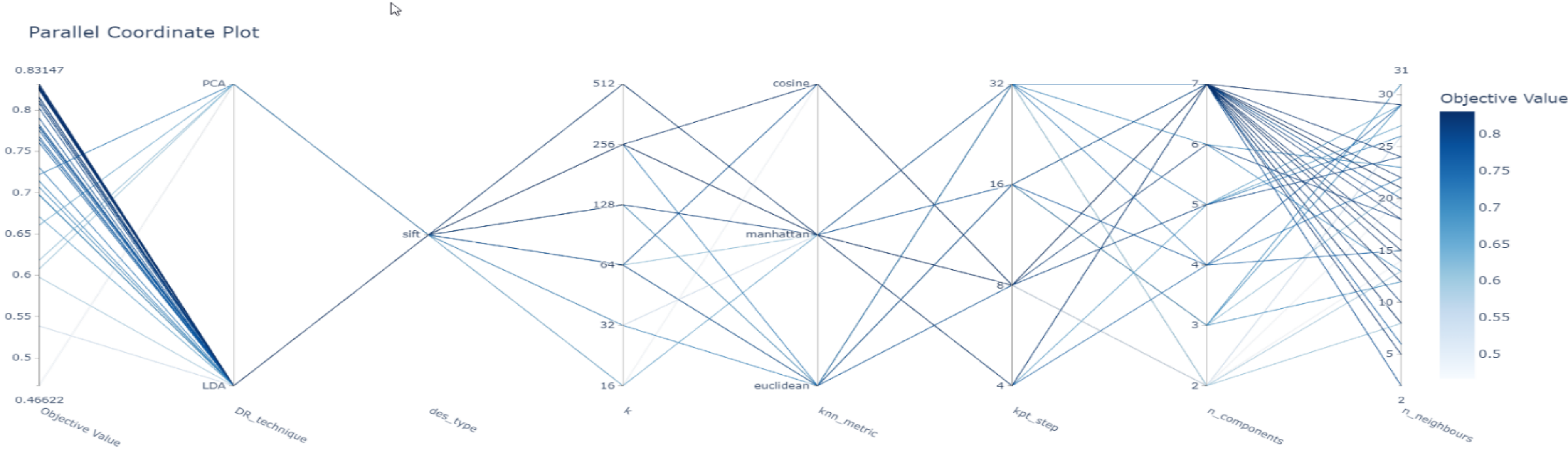
Optuna

```
optuna.visualization.plot_slice(study, params=["k", "n_neighbours", "knn_metric"])
```



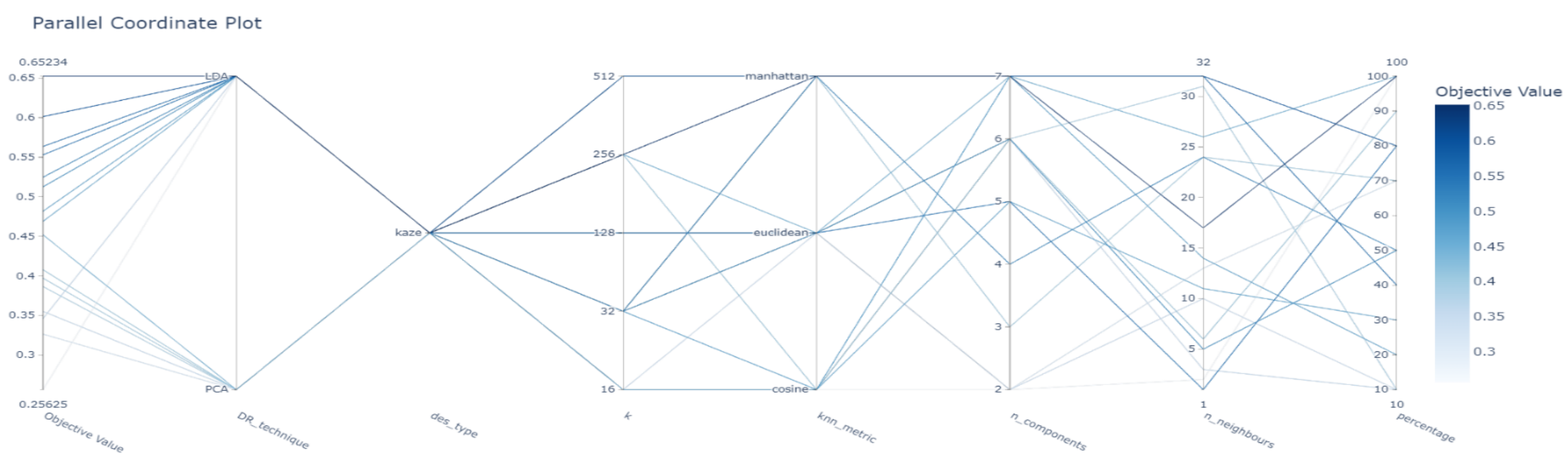
```
fig = optuna.visualization.plot_parallel_coordinate(sift_study, target=lambda t: t.values[0])
fig.data[0].line.reversescale = not fig.data[0].line.reversescale
fig.show()
```

```
2023-01-08 22:16:13,323] A new study created in memory with name: no-name-d00c3c6e-852e-41d8-90b6-a510fc2f6b9a
```




```
fig = optuna.visualization.plot_parallel_coordinate(kaze_study, target=lambda t: t.values[0])
fig.data[0].line.reversescale = not fig.data[0].line.reversescale
fig.show()
```

```
2023-01-08 22:16:09,923] A new study created in memory with name: no-name-271a6e74-960d-4e71-a91a-0a701868f52a
```

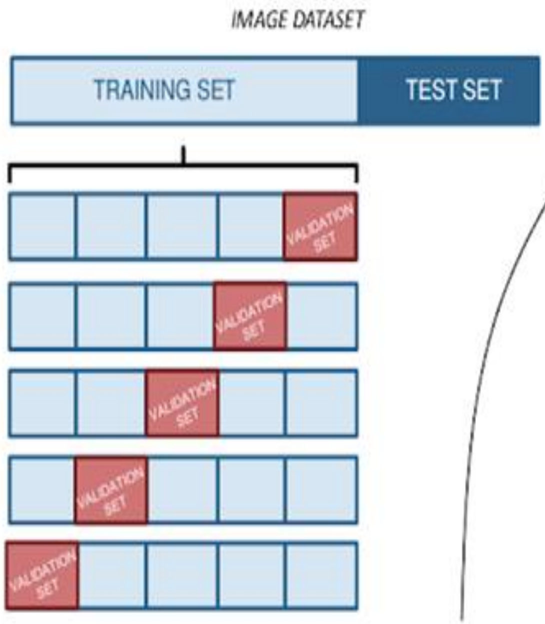


Group	Temptative grade
1	8
2	7
3	8
4	10
5	10
6	7
7	9
8	8
9	9
10	10

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, Optuna
- 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



*For instance, in the KNN classifier we want to find the most suitable **K** (number of neighbours used to classify each descriptor)*

*We plotted the average of the accuracies (Y axis) obtained with 5 folds with respect to varying **K** from 1 to 100 (X axis).*

Best average accuracy with K = 30 !

So we obtain that individual SIFT descriptors are best classified using K = 30 (for another kind of descriptor the 'best' K will vary)

We use K-fold cross validation to pick suitable parameters for the classifier while **avoiding overfitting**.

3/10

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

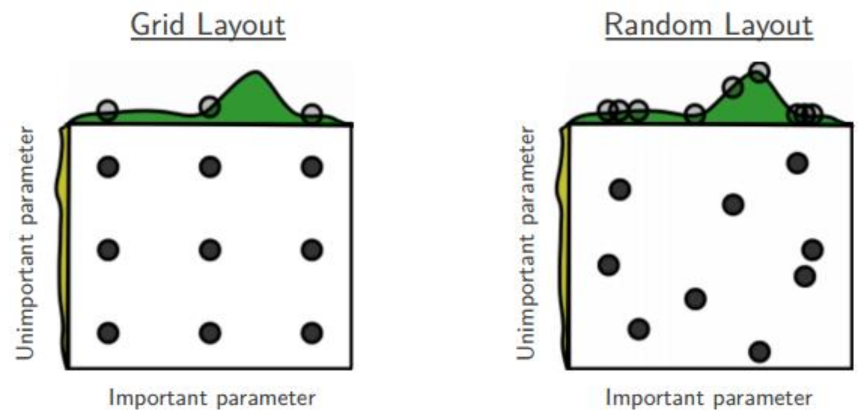
Abstract

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

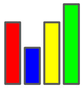
Discrete hyperparameter: list of discrete choices
random selection
(without replacement if all discrete)

Set the number of trials

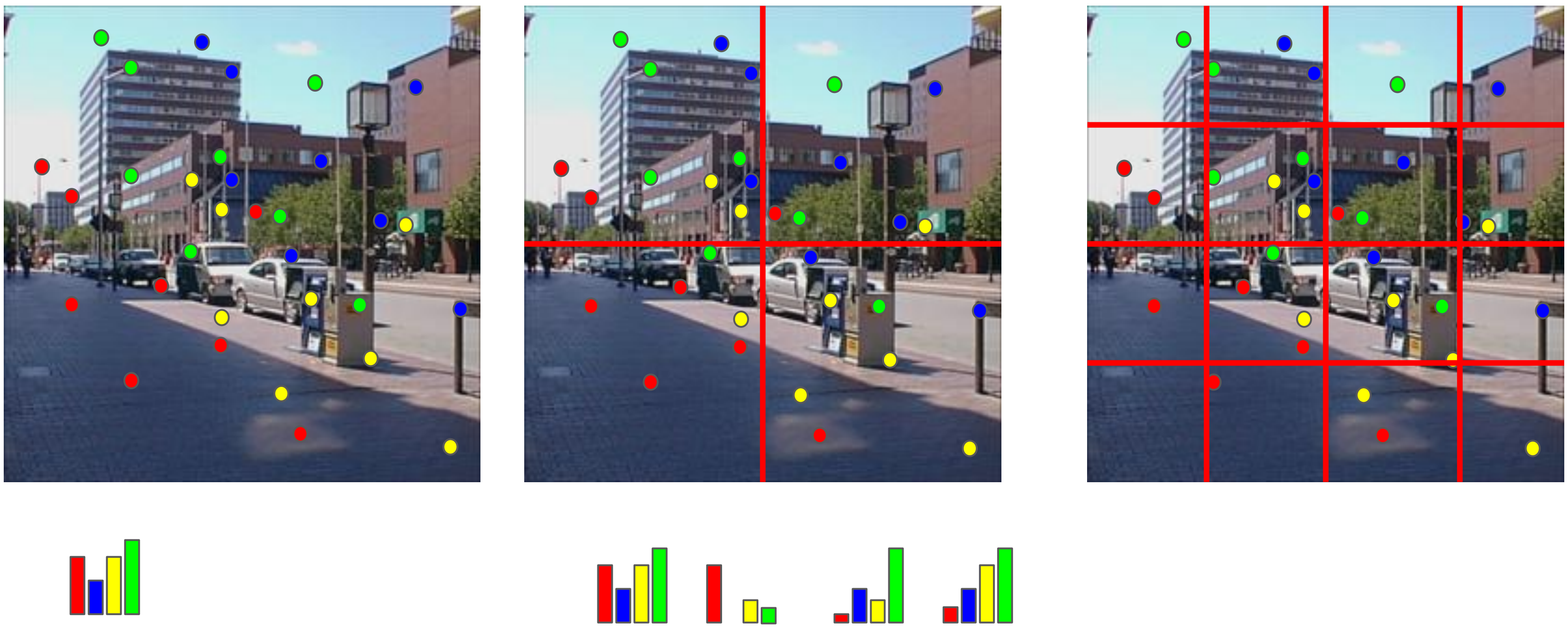
Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

```
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

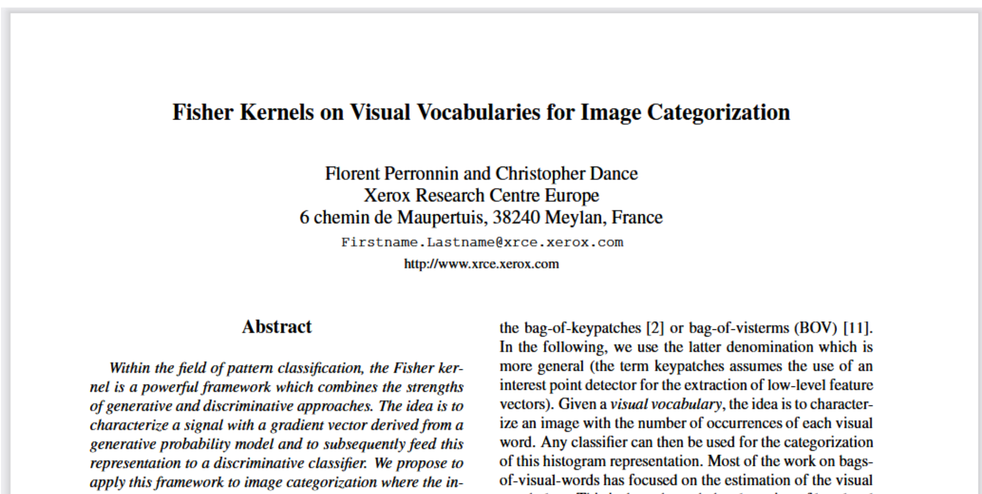
```
def histogramIntersection(M, N):
```

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

```
    return K_int
```

Fisher Vectors

- BoVW is only counting the number of local descriptors assigned to each Voronoi cell
- Why not including higher order statistics?
  - Mean of local descriptors
  - Co-variance of local descriptors
- FV is typically 2 x D x k dimensional



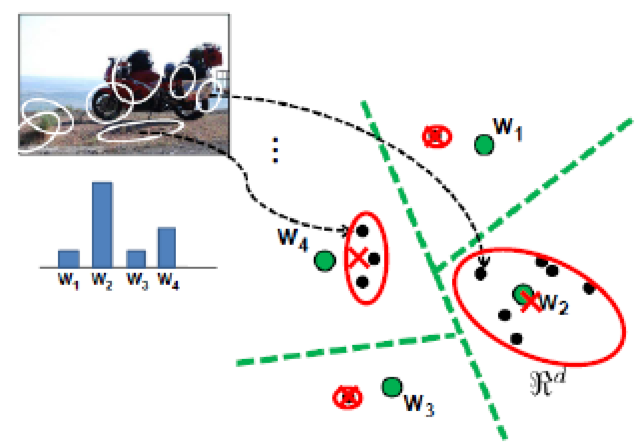
Fisher Kernels on Visual Vocabularies for Image Categorization

Florent Perronnin and Christopher Dance
Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan, France
Firstname.Lastname@xrce.xerox.com
http://www.xrce.xerox.com

Abstract

Within the field of pattern classification, the Fisher kernel is a powerful framework which combines the strengths of generative and discriminative approaches. The idea is to characterize a signal with a gradient vector derived from a generative probability model and to subsequently feed this representation to a discriminative classifier. We propose to apply this framework to image categorization where the in-

the bag-of-keypatches [2] or bag-of-visterms (BOV) [11]. In the following, we use the latter denomination which is more general (the term keypatches assumes the use of an interest point detector for the extraction of low-level feature vectors). Given a *visual vocabulary*, the idea is to characterize an image with the number of occurrences of each visual word. Any classifier can then be used for the categorization of this histogram representation. Most of the work on bags-of-visual-words has focused on the estimation of the visual



Tasks to do

<u>Improve the BoVW code with:</u>

- 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