Bag-of-Words based Image Classification (with OpenCV-Python)

Heeseung Yun, Jaewoo Ahn

SEOUL NATIONAL UNIV.
VISION & LEARNING

(Content credit: Byeongchang Kim)

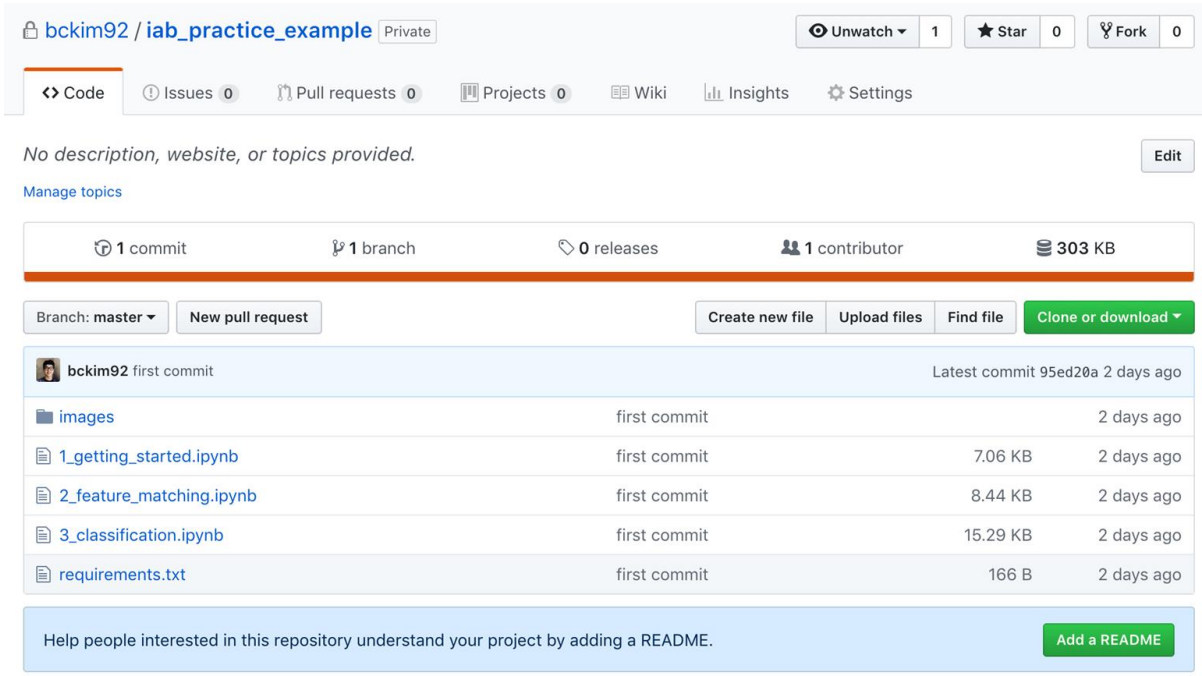
공지사항

- 질문은 비밀글로 올리시면 답변해드리지 않습니다
- 컴퓨터비전 1,2 실습은 아래의 3개로 구성되어 있습니다.
  숙제는 3번파일에 들어있는 미완성되어 있는 부분을 완성하여 **9/24 23:59**까지 **ETL**에 제출하면 됩니다.
  - 1_getting_started.ipynb (수업시간에 같이)
  - 2_feature_matching.ipynb (문제 없음)
  - 3_classifictaion.ipynb (**ETL에 제출**)

실습코드 받는법

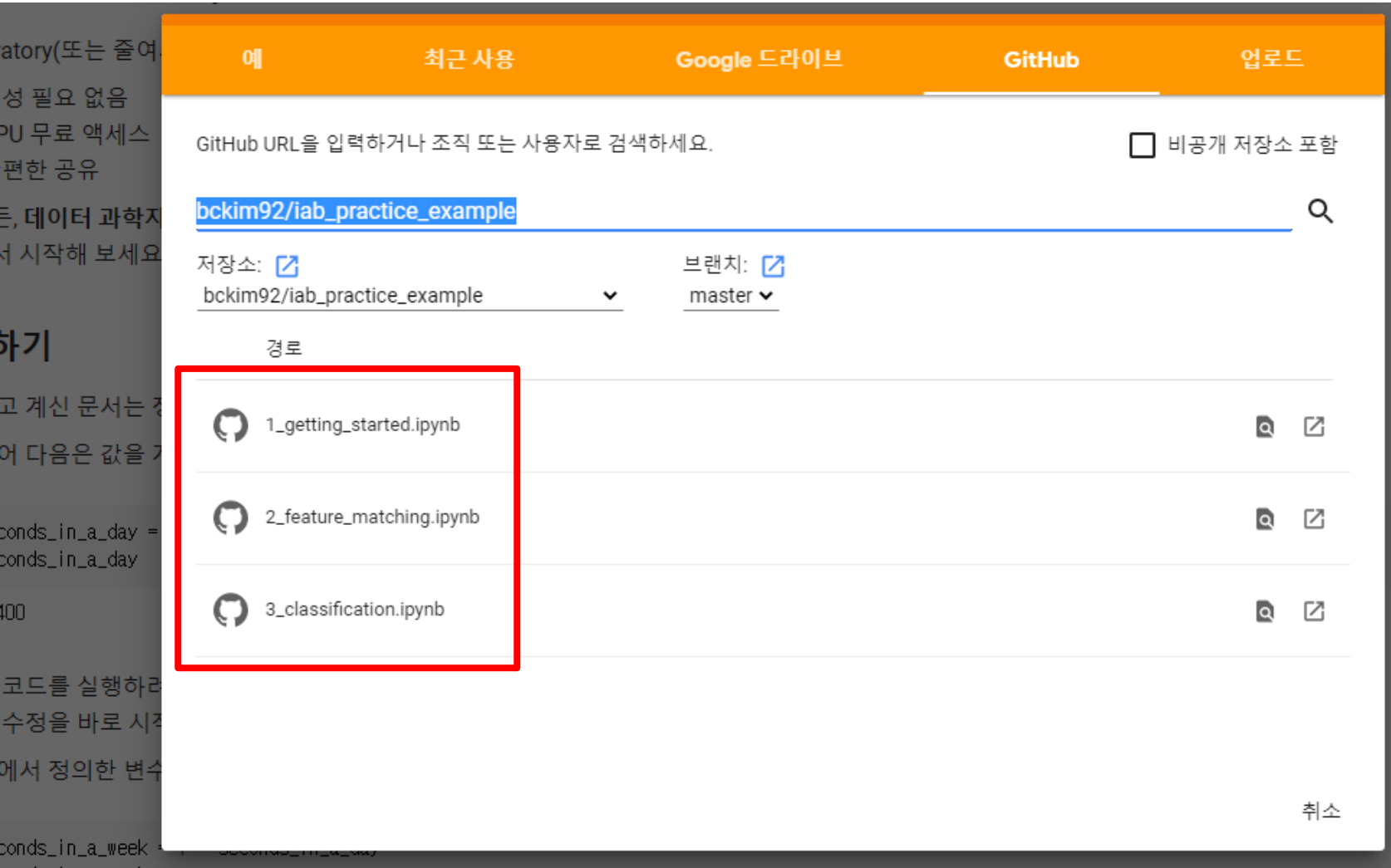
https://github.com/bckim92/iab_practice_example

```
$ git clone https://github.com/bckim92/iab_practice_example.git
$ cd iab_practice_example/
$ pip install -r requirements.txt
$ jupyter notebook
```



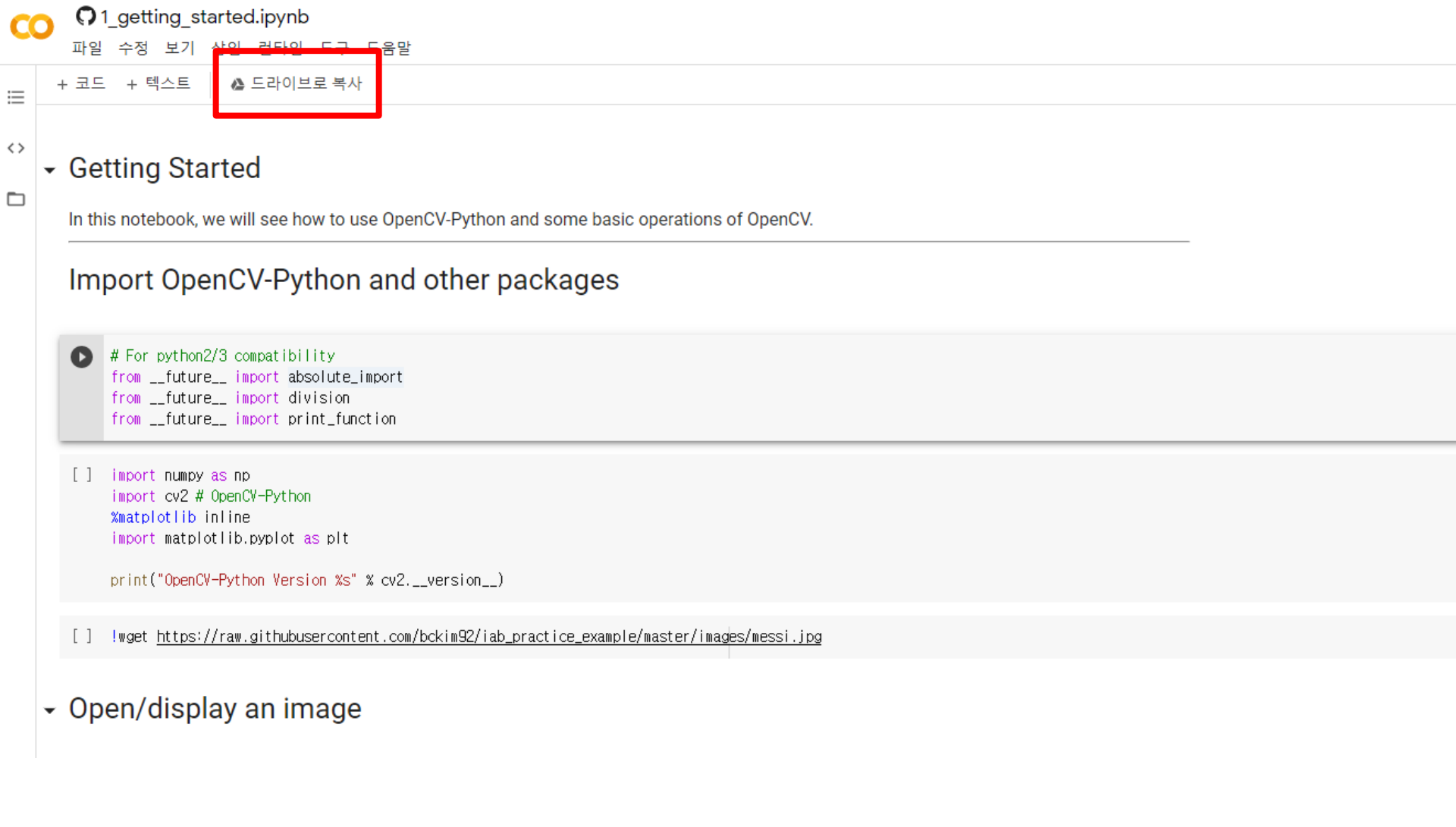
실습환경 세팅 google colab

파일 -> 노트열기 -> github 탭 ->
bckim92/iab_practice_example 입력 -> 해당 ipynb file 클릭



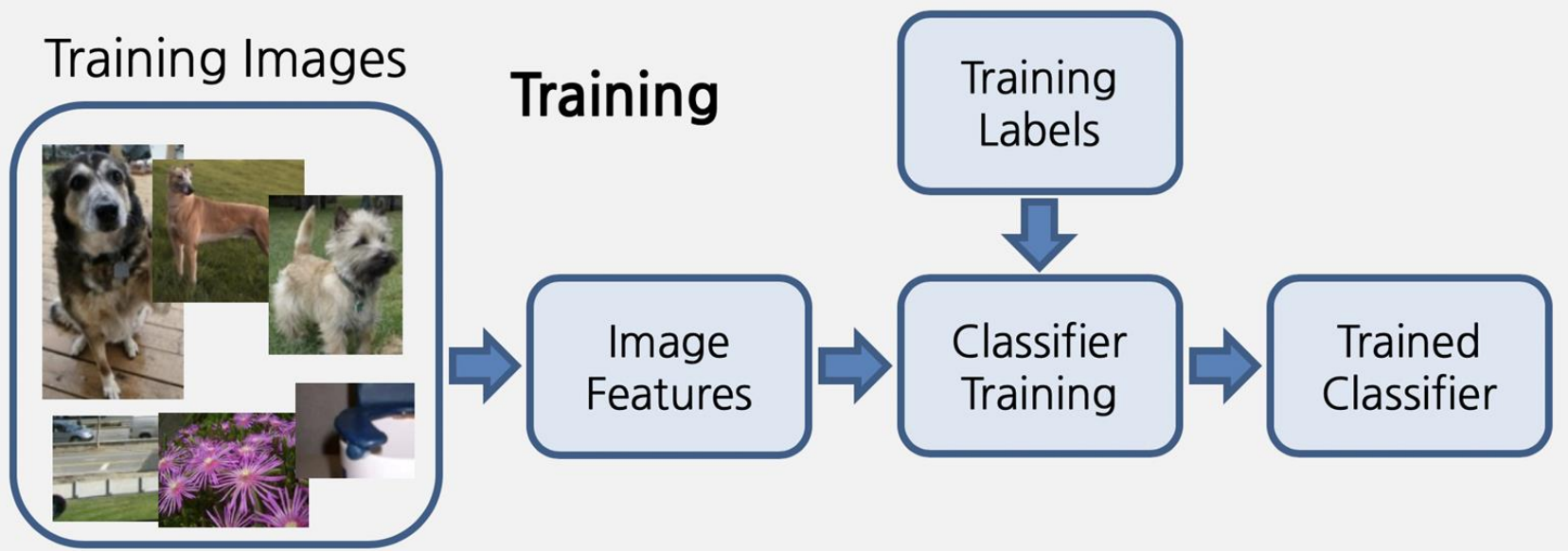
실습환경 세팅 google colab

드라이브로 복사



Contents

- Backgrounds
  - OpenCV-Python
  - Image features (e.g. SIFT)
  - Classifier (e.g. SVM)
- Bag-of-words based image classification



(Image credit: Gunhee Kim's lecture slide)

OpenCV-Python

Introduction
Image manipulation
Draw objects

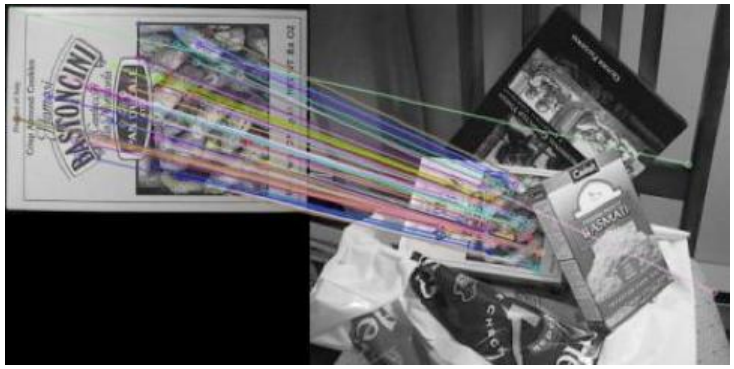
OpenCV-Python

- OpenCV
  - **Computer vision library** started from 1995(Intel)
  - Now supports a multitude of algorithms related to CV and ML (a little of)
- OpenCV-Python
  - OpenCV is basically written in C++
  - OpenCV-Python is a **Python wrapper** of OpenCV
- Prior knowledge of **Python** and **Numpy** is needed
  - A Quick guide to Python - A Byte of Python
  - Numpy Quickstart Tutorial / Justin Johnson's Numpy Tutorial

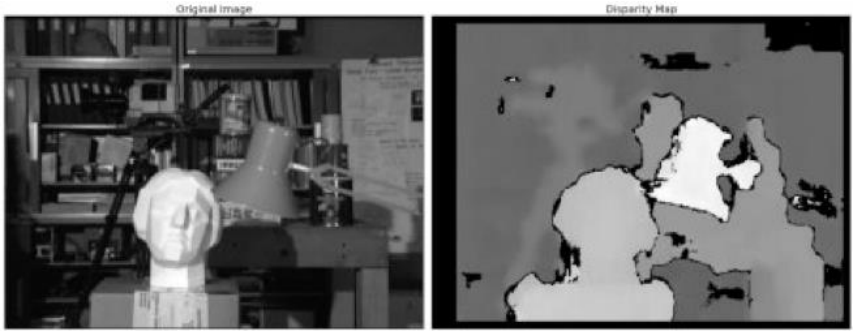
OpenCV-Python

- Examples of algorithms with OpenCV
  - Face Detection

  - Feature extraction/matching



  - Depth map for stereo images



  - Image inpainting

Using OpenCV-Python

- Import OpenCV-Python package “cv2”

```
import numpy as np
import cv2 # OpenCV-Python
%matplotlib inline
import matplotlib.pyplot as plt
```

- **Numpy** arrays are data structure used in cv2
  - Converted from/to CvMat(OpenCV in C++) by OpenCV-Python
  - Also used in many python packages

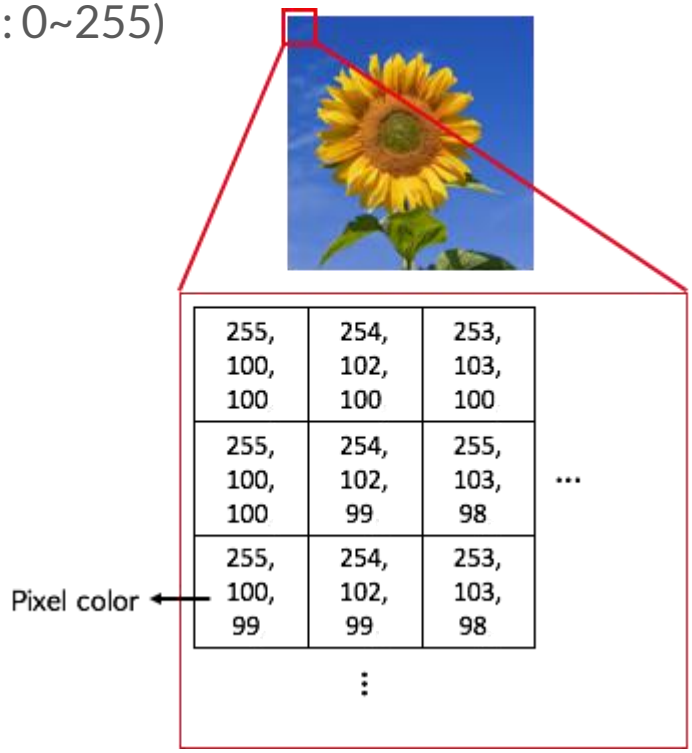
Open/Display an Image

- Open an image

```
img = cv2.imread('image.jpg', cv2.IMREAD_COLOR)
```

- The output is a Numpy array
  - 3D (H x W x C for color) / 2D (H x W for grayscale)
  - Top-left to bottom-right
  - Data type (dtype): `np.uint8` (1-byte unsigned: 0~255)

- Flag specifies the way image should be read
  - cv2.IMREAD_COLOR
  - cv2.IMREAD_GRAYSCALE
  - cv2.IMREAD_UNCHANGED

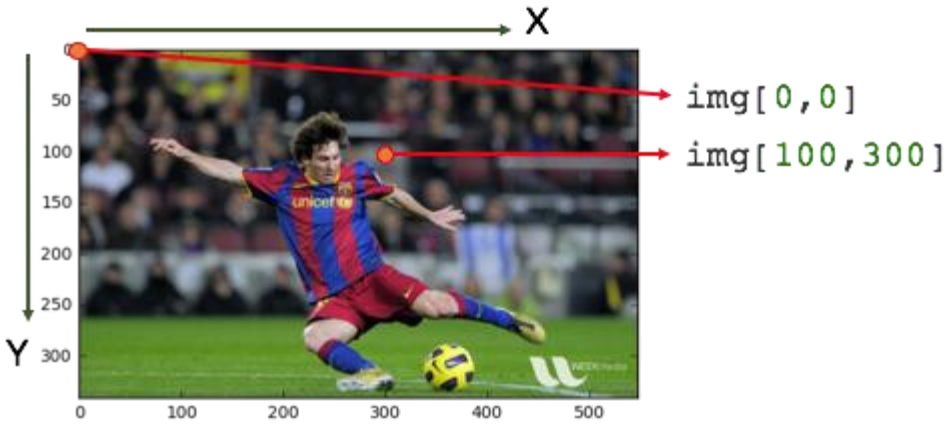


Open/Display an Image

- Display an image using Matplotlib

```
# display an image using matplotlib
plt.imshow(img)  # => The output in wrong color!!
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

  - `plt.imshow(img)` displays an (RGB, RGBA, grayscale) image
  - OpenCV represents RGB images as Numpy arrays in **REVERSE** order (**BGR** not RGB)
  - `cv2.cvtColor(img, conversion)` provides conversion among many colortypes



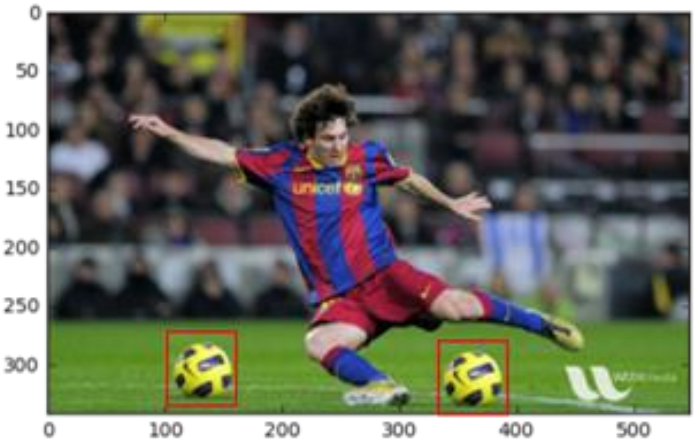
Modify Pixels & ROI

- Pixel and ROI(Region of Interest) can be accessed by Numpy indexing
  - [row, column] ordering - same as matrix indexing

```
# Access a pixel value (BGR order)
img[50, 235]
```

```
=> array([27, 25, 24], dtype=uint8)
```

```
# ROI is obtained using Numpy indexing
ball = img[280:340, 330:390]
img[273:333, 100:160] = ball
```



Draw Objects

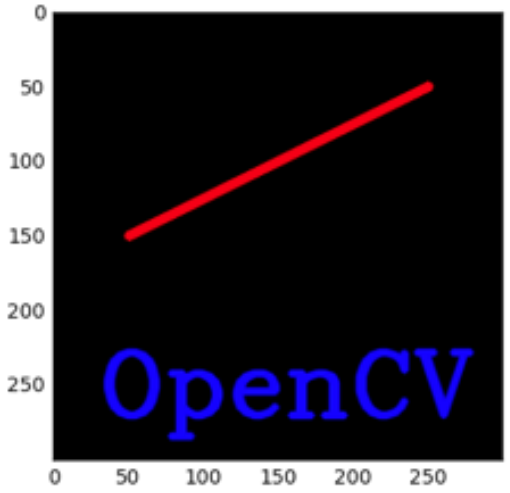
- Draw object (line, rectangle, circle, ellipse, polygon)
  - `cv2.line(), cv2.rectangle(), cv2.circle(), cv2.ellipse(), cv2.polyline()`
- Put some text
  - `cv2.putText()`
- Arguments
  - `cv2.function(image, {properties of object})`
  - RGB order in color (not BGR)
  - X, Y order in position (not row, column)

Draw Objects

- Example

```
# cv2.line(image, startPoint, endPoint, rgb, thickness)
cv2.line(img, (50,150), (250,50), (255,0,0), 5)

# cv2.putText(image, text, bottomLeft, fontType, fontScale,
rgb, thickness, lineType)
font = cv2.FONT_HERSHEY_COMPLEX
cv2.putText(img, 'OpenCV', (30,270), font, 2, (0,0,255), 3,
cv2.LINE_AA)
```



Let’s Check the Code

1_getting_started.ipynb

Image Features

- Understanding Features
  - Feature detection / description
  - SIFT
- Feature Extraction in OpenCV-Python

Understanding Features

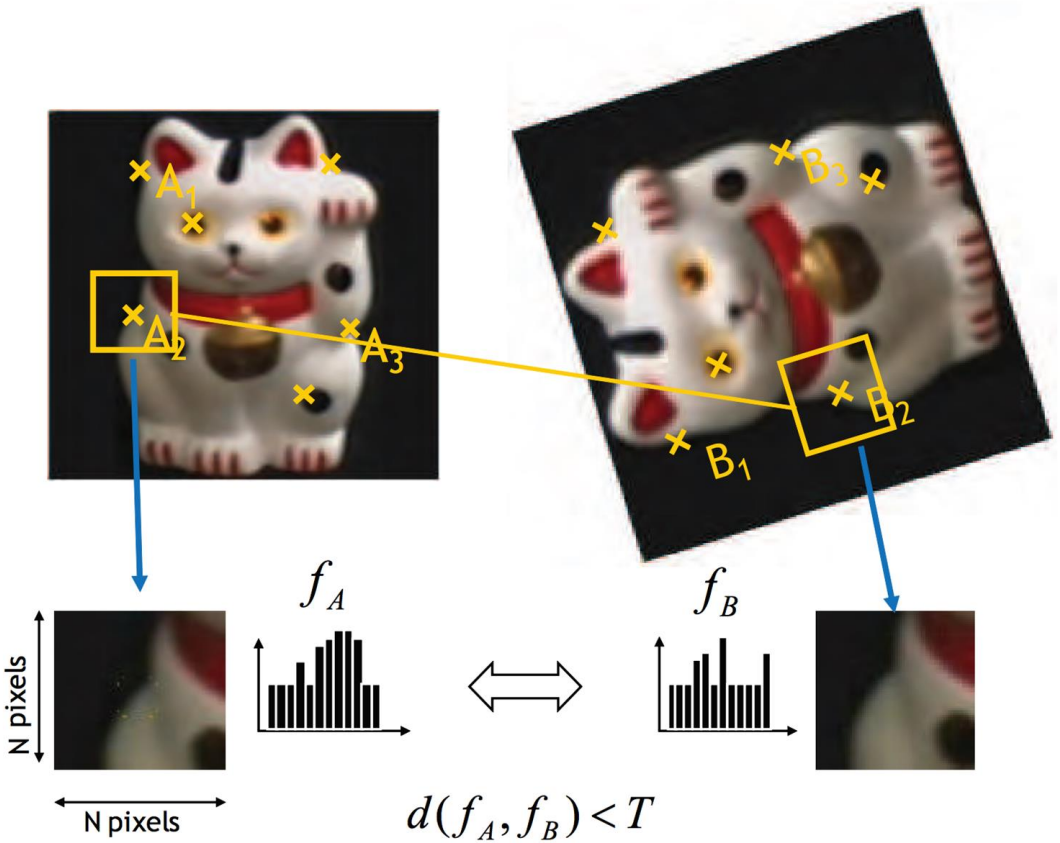
- Feature : A piece of information relevant for solving task.
  - Specific patterns which are unique, which can be easily tracked, which can be easily compared
  - Local visual feature: **Salient point** and its **representation**

- An example: Find patches A~F in the picture

Understanding Features

- Local Visual Features
  - Feature detection / Feature description

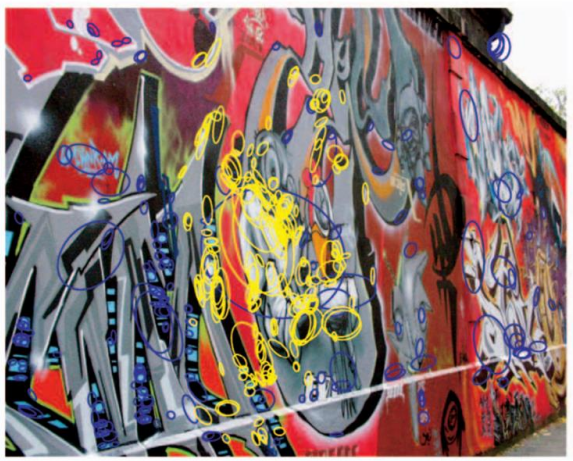
- Keypoint Matching
  - Find distinctive points
  - Define local regions around the points
  - Compute local descriptors from the region
  - Match local descriptors of two images



Feature Detection / Description

- Keypoint(Local Feature) Detection
  - Finding keypoints / interest points
  - Usually corners and blob centers
  - Harris corner detector, DoG, MSER

- Good Feature Detection
  - Repeatable: Robust to scaling / rotation / viewpoint change
  - Distinctive: Different features should look differently

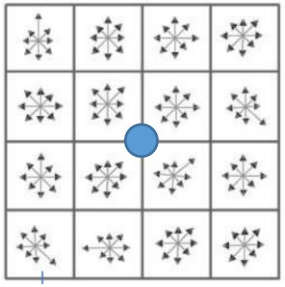


Wikipedia, *Scale-invariant feature transform* [https://en.wikipedia.org/wiki/Scale-invariant_feature_transform]
Krystian Mikolajczyk and Cordelia Schmid, "*A Performance Evaluation of Local Descriptors*"

Feature Detection / Description

- Feature Description
  - Represent keypoints or local regions around them as a vector
  - SIFT, SURF

- ex) SIFT descriptor – 128-D vector



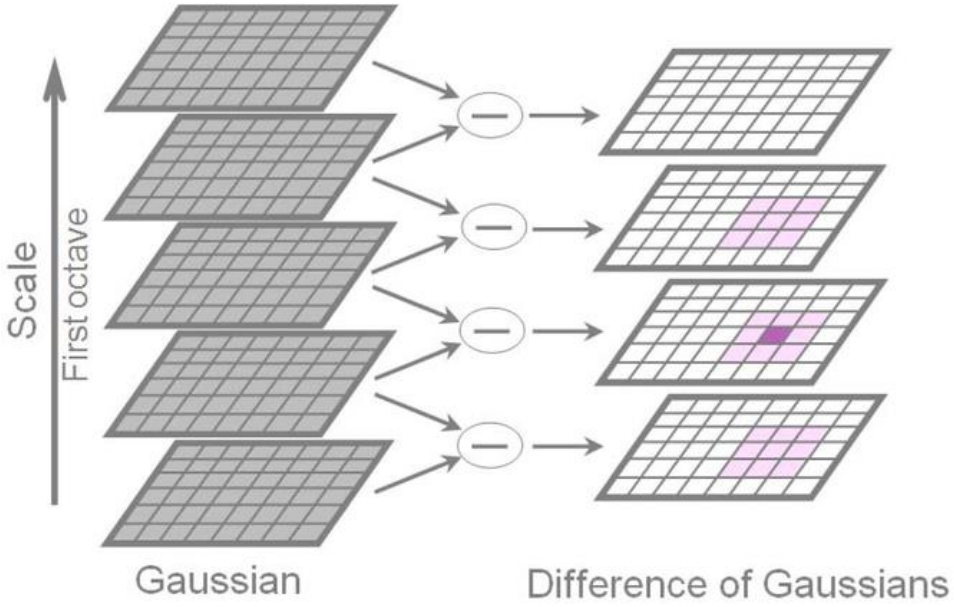
128 Dimension

(7.0, 0.0, 0.0, 0.0, 128.0, 86.0, …, 4.0 2.0 0.0)

SIFT (Scale-Invariant Feature Transform)

- An algorithm to **detect** and **describe** local features in images
  - Feature detector + Feature descriptor
  - Published by David Lowe in 1999

- We will skip the details of SIFT in this class

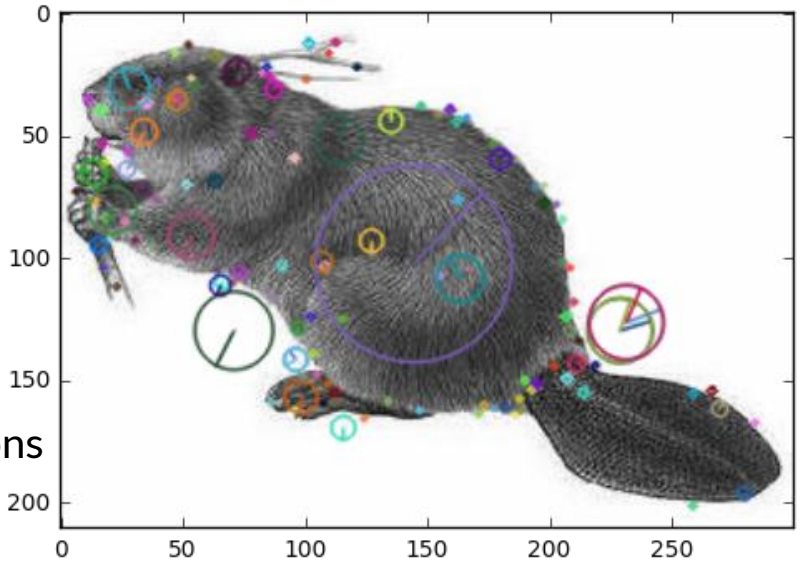


Feature Extraction in OpenCV

- `Cv2.xfeatures2d` module
  - Sub-module for feature detectors / descriptors (opencv_contrib package)
  - SIFT, SURF, BRIEF …
- Module for extracting and computing SIFT

```
# SIFT feature detector/descriptor
sift = cv2.xfeatures2d.SIFT_create()
kp, des = sift.detectAndCompute(gray, kp)
```

- Output
  - **`kp`**: A list of N **`'cv2.KeyPoint'`**
    - position, scale, orientation
  - **`des`**: Descriptors($N \times 128$ Numpy array)
- Caveat
  - SIFT is excluded in recent OpenCV versions
  - Solution: use opencv-python==**3.4.2.17**



Feature Matching

- Brute Force Matching

i

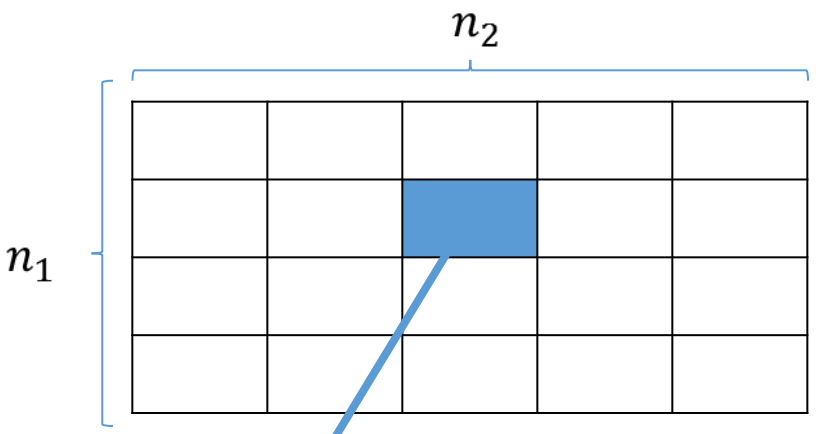
j

128-D vector v

n_1

n_2

Distance Matrix M



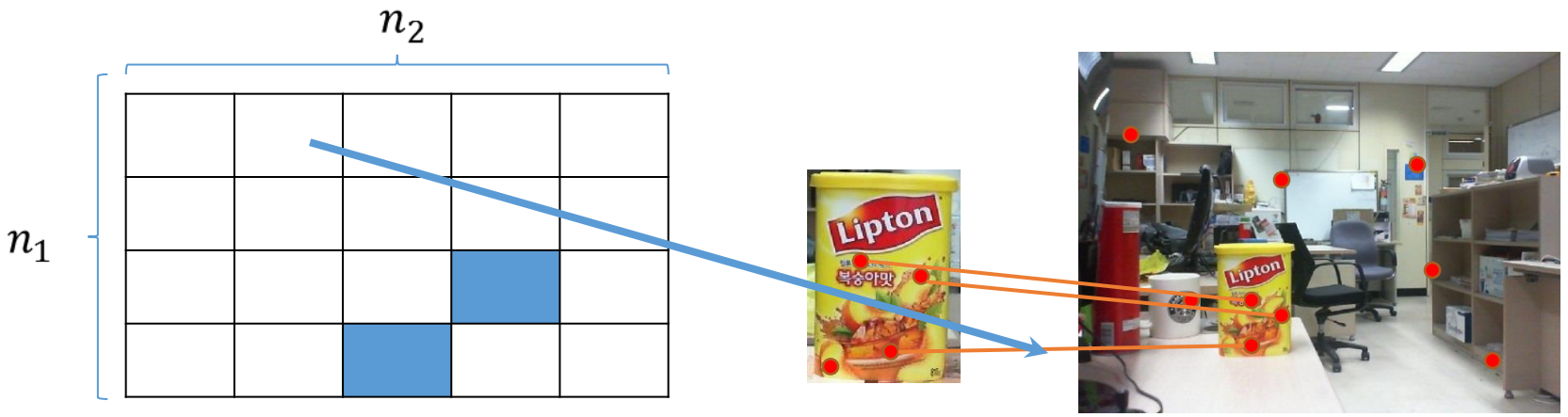
$$M_{i,j} = d(u_i, v_j)$$

Feature Distance

- L2-norm: $\|u_i - v_j\|$
- Hellinger distance: $\sqrt{1 - \sqrt{\overline{u_i}\,\overline{v_j}}}$

Feature Matching

- Brute Force Matching



- Brute Force Matching in OpenCV-Python

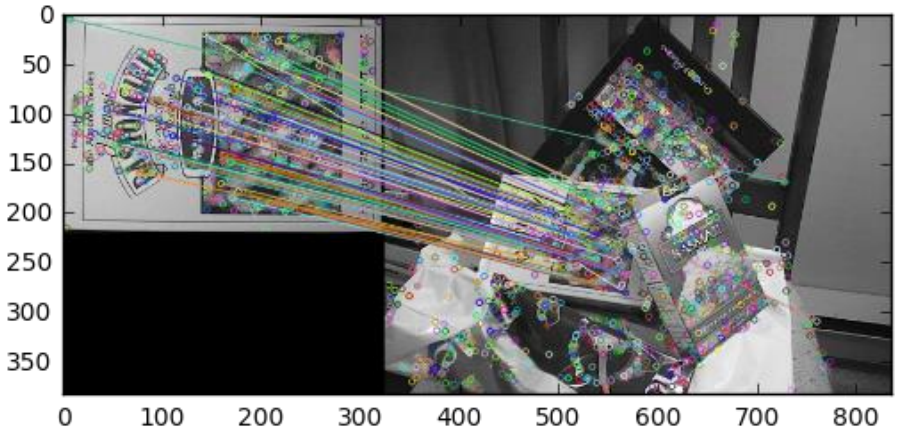
```
# BFMatcher(Brute Force Matcher) with default setting(L2 distance)
bf = cv2.BFMatcher(cv2.NORM_L2)
# Find closest 2 des2 points for each point in des1
matches = bf.knnMatch(des1, des2, k=2)
```

Feature Matching

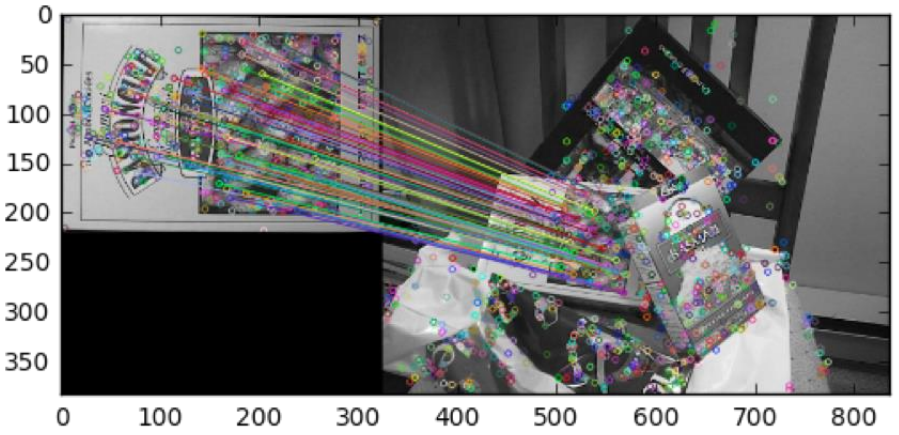
- Display Matches

```
# Display matches
img_draw = cv2.drawMatches(img1, kp1, img2, kp2, matches, None)
plt.imshow(cv2.cvtColor(img_draw, cv2.COLOR_BGR2RGB))
```

- Euclidean distance



- Hellinger distance



Let’s Check the Code

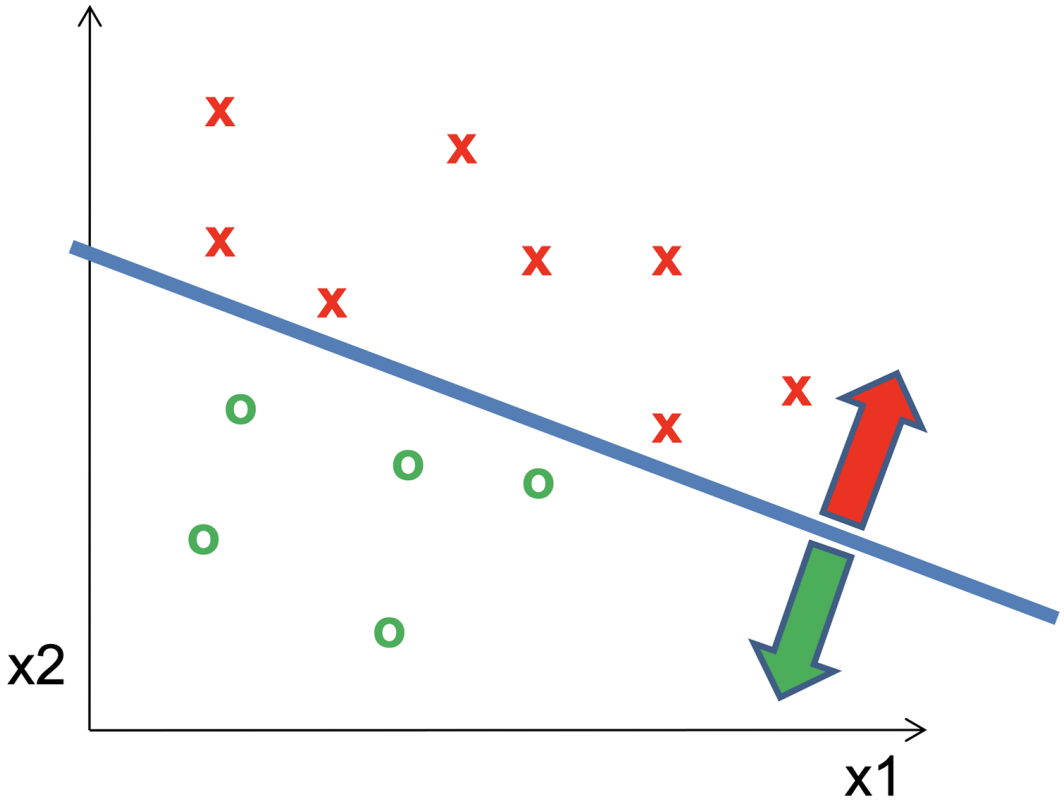
`2_feature_matching.ipynb`

Classifier

Support Vector Machine (SVM)

Classifier

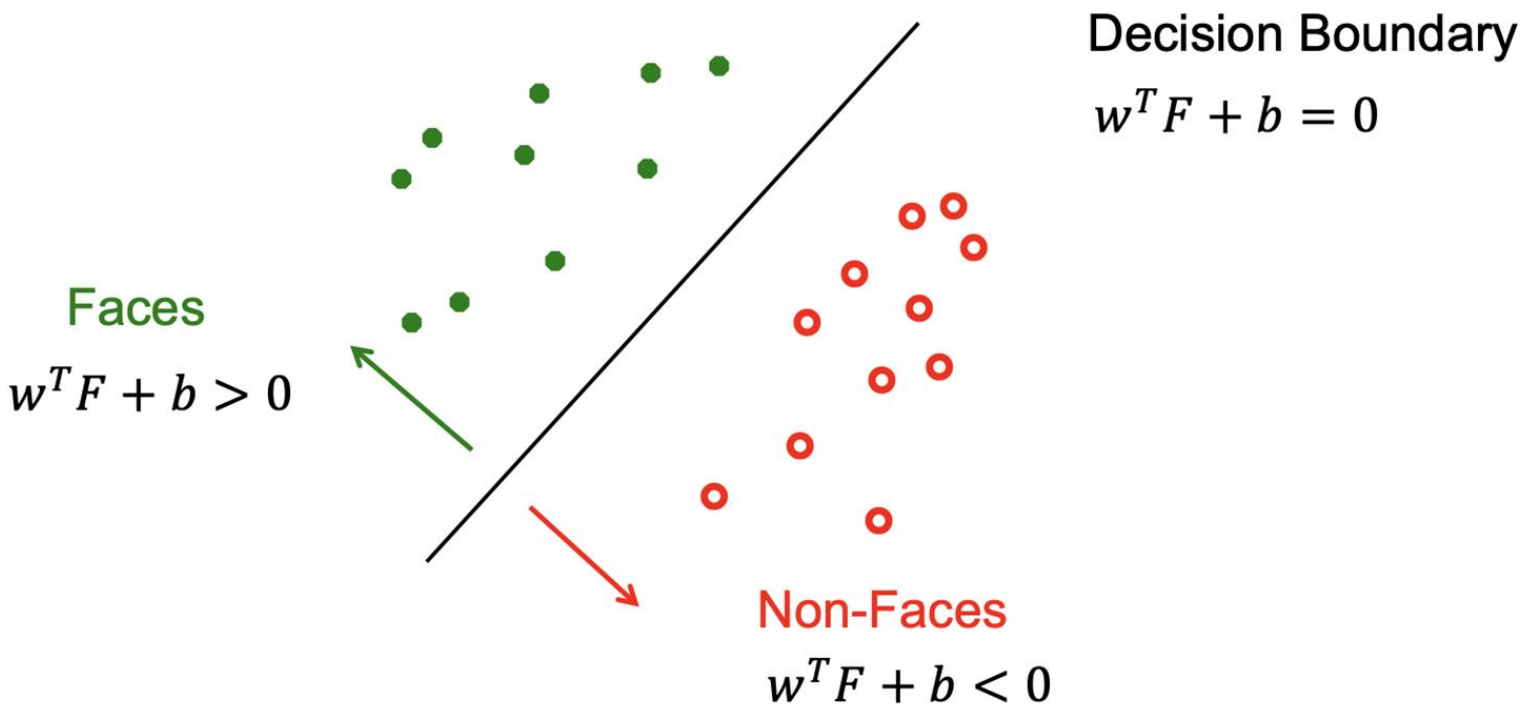
- A classifier maps from the feature space to label
  - Training labels dictate that two examples are the same or different, in some sense
  - We want the simplest function that is confidently correct



(Content credit: Gunhee Kim's CV lecture slide)

Decision Boundary

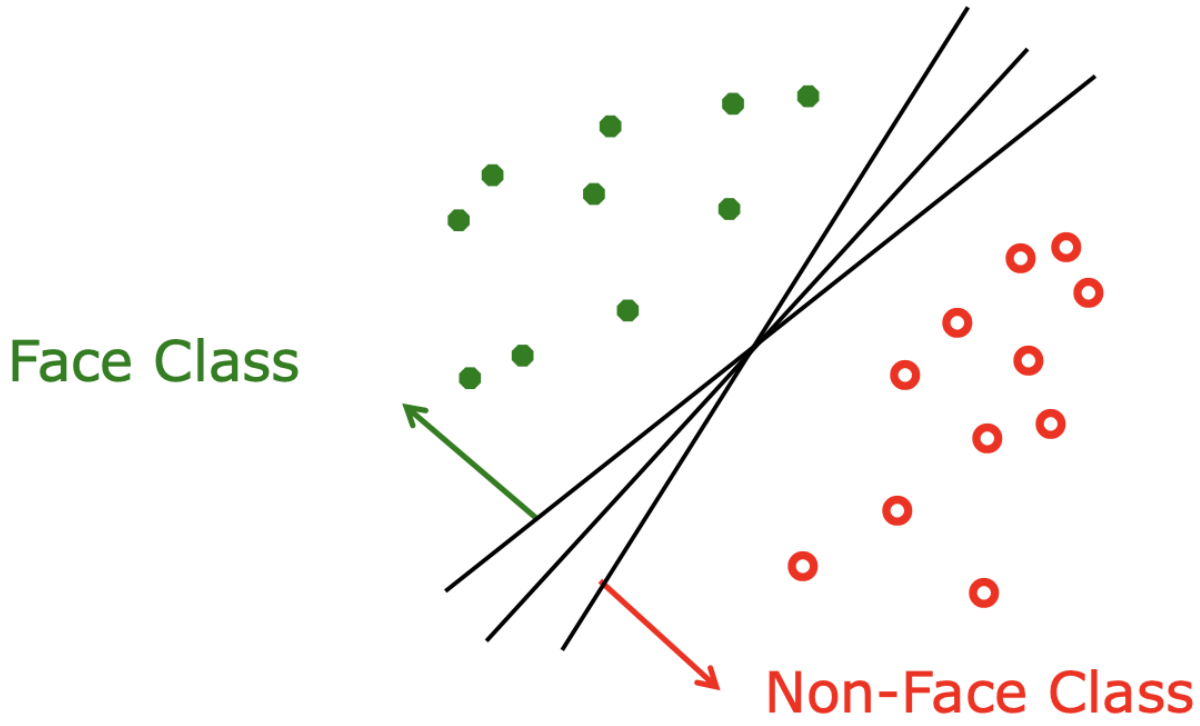
- Find a decision boundary in a feature space



(Content credit: Gunhee Kim's CV lecture slide)

Decision Boundary

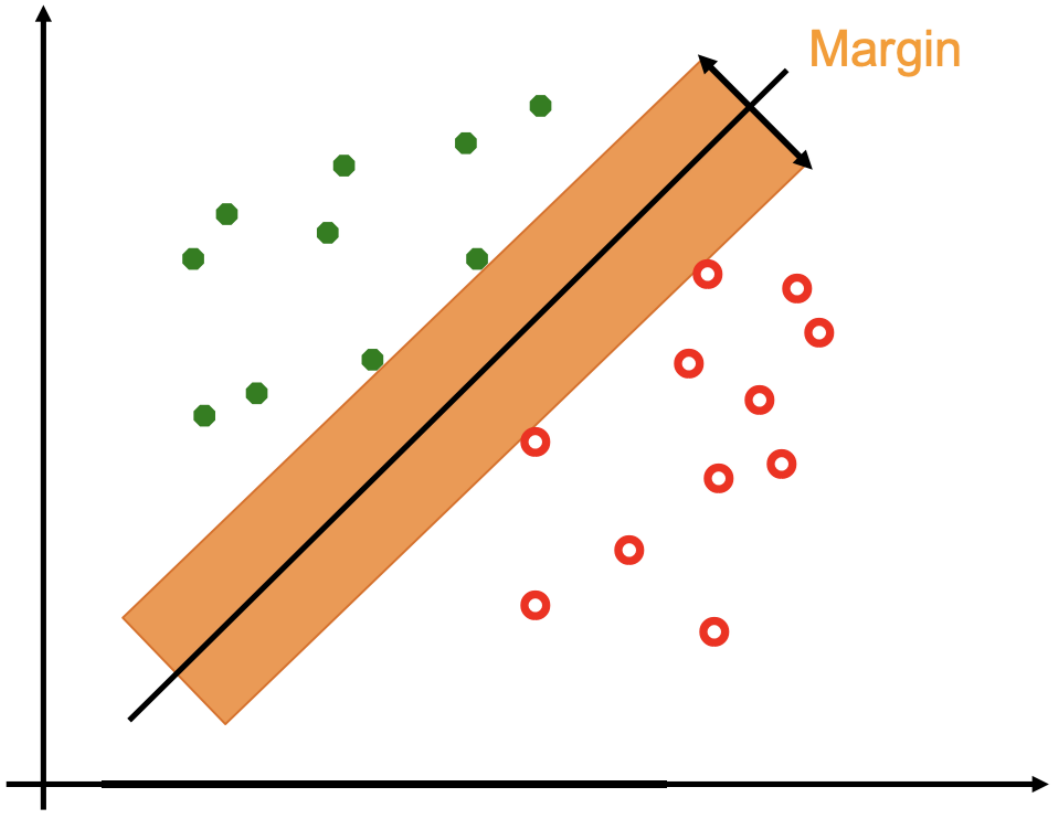
- How to find the optimal decision boundary?



(Content credit: Gunhee Kim's CV lecture slide)

Evaluating a Decision Boundary

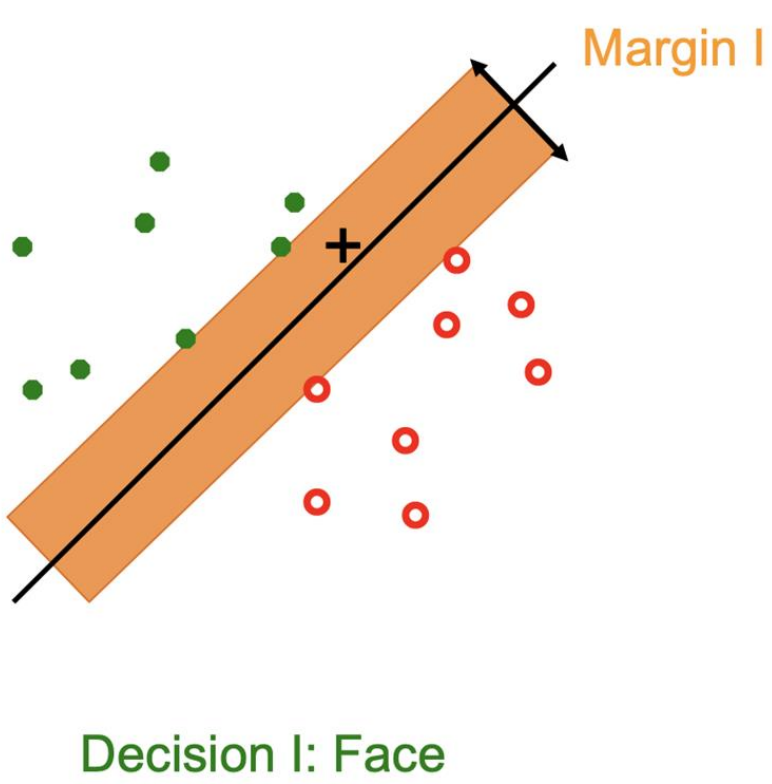
- Margin or safe zone: The width that the boundary could be increased by before hitting a data point

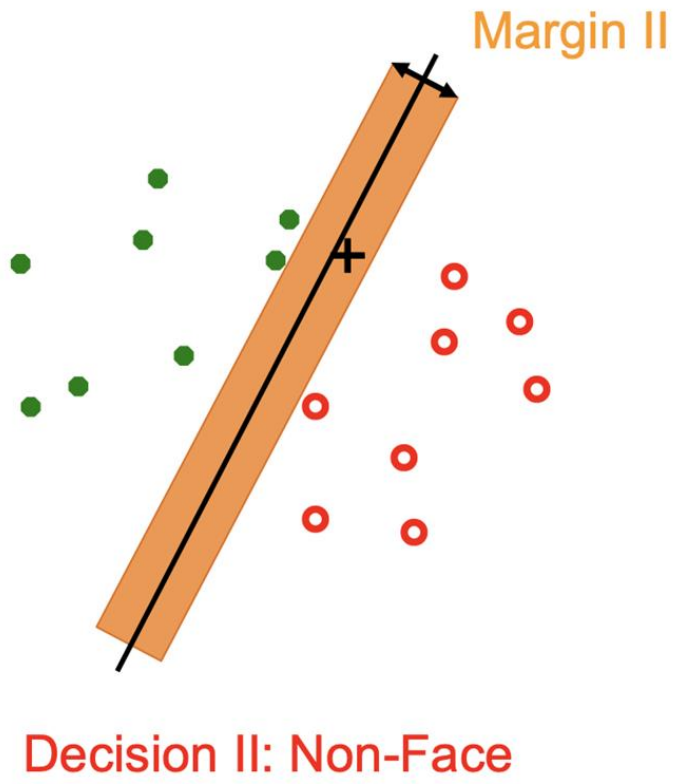


(Content credit: Gunhee Kim's CV lecture slide)

Evaluating a Decision Boundary

- Choose decision boundary with the maximum margin



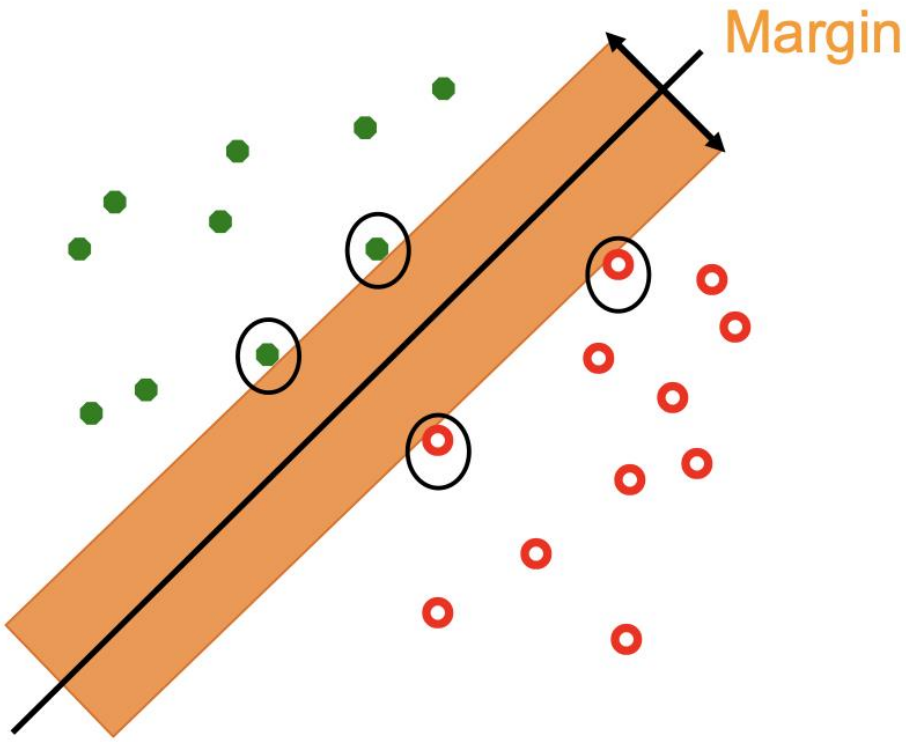


(Content credit: Gunhee Kim's CV lecture slide)

Support Vector Machine (SVM)

- Support vectors: Closest data samples to the boundary

- Decision boundary and the margin depend only on the support vectors



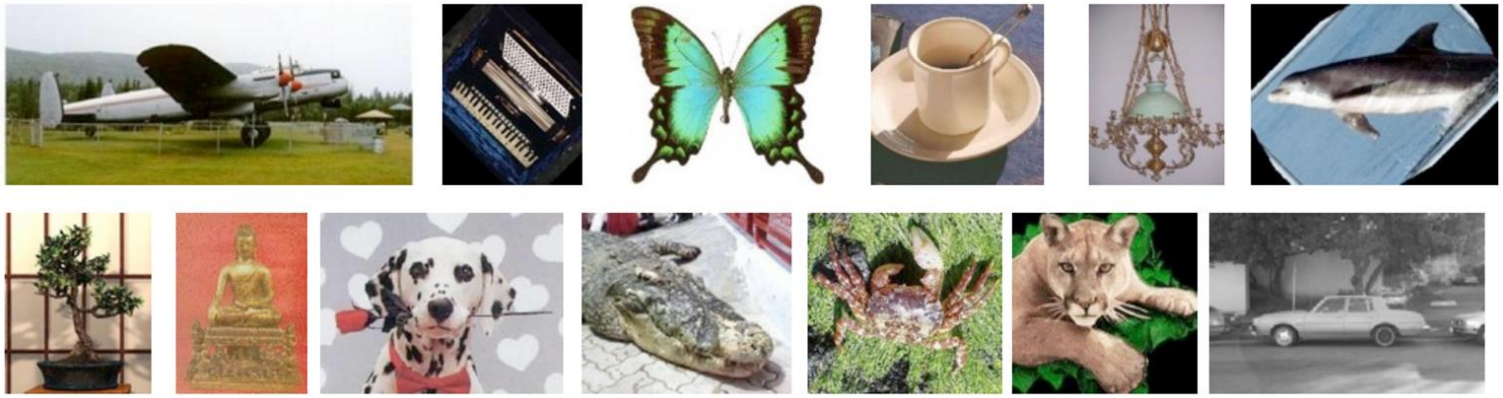
(Content credit: Gunhee Kim's CV lecture slide)

Bag-of-Words based Image Classification

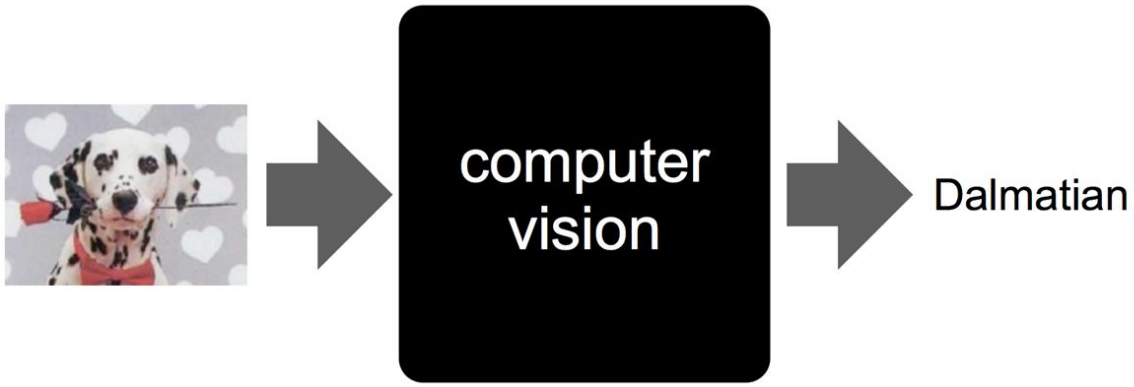
- Image classification
- Image feature
  - Dense SIFT(PHOW)
  - VBoW
  - Spatial histogram
- Classifier

Image Classification

- The Dataset – Caltech 101



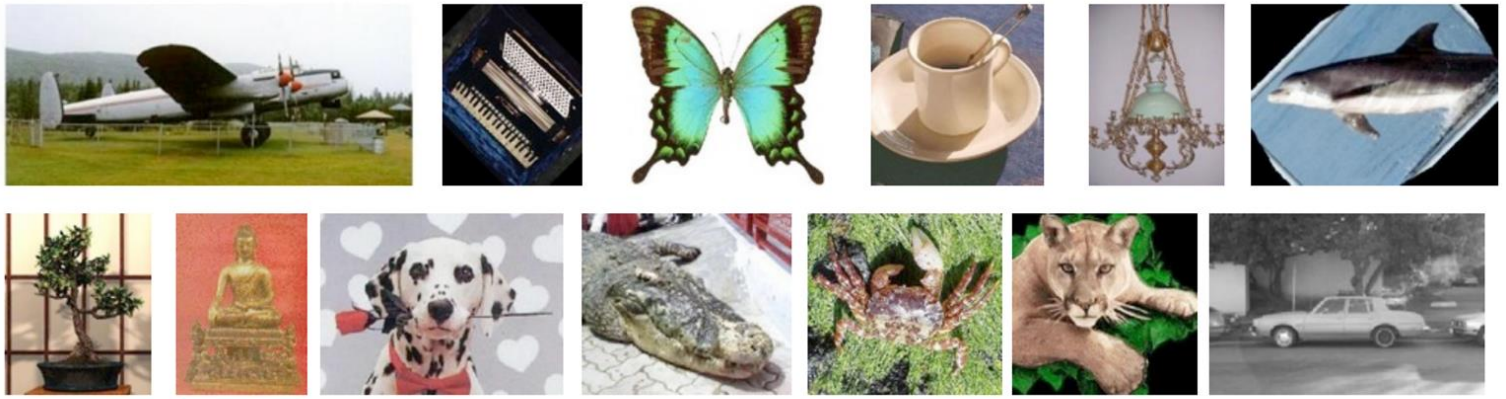
[Fei-Fei et al. 2003]



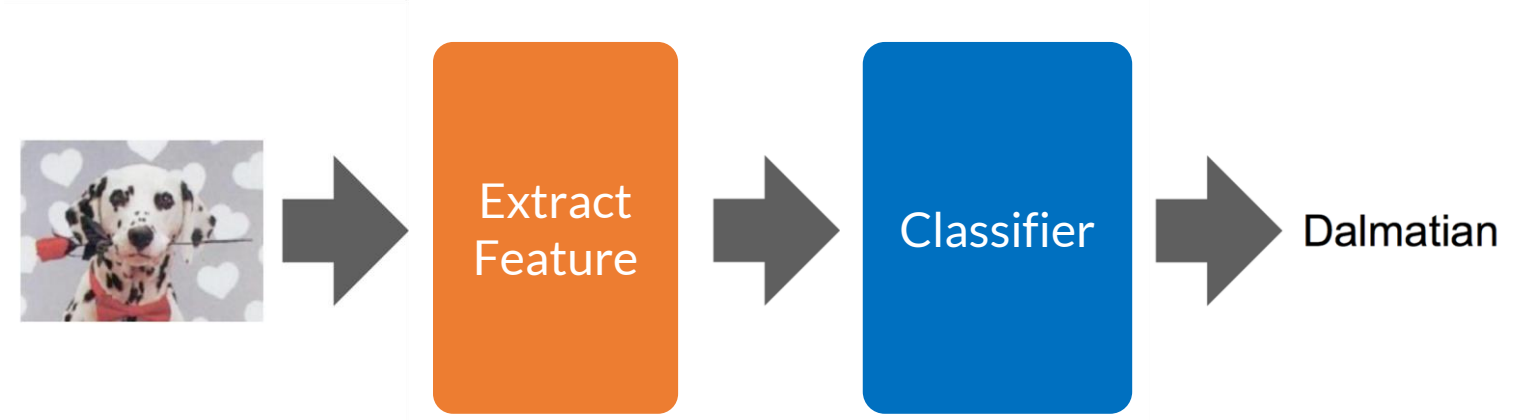
Andrea Vedaldi, Brian Fulkerson, *VLFeat Tutorial* [http://vision.ucla.edu/~vedaldi/assets/pubs/vedaldi10vlfeat-tutorial.pdf]

Image Classification

- The Dataset – Caltech 101



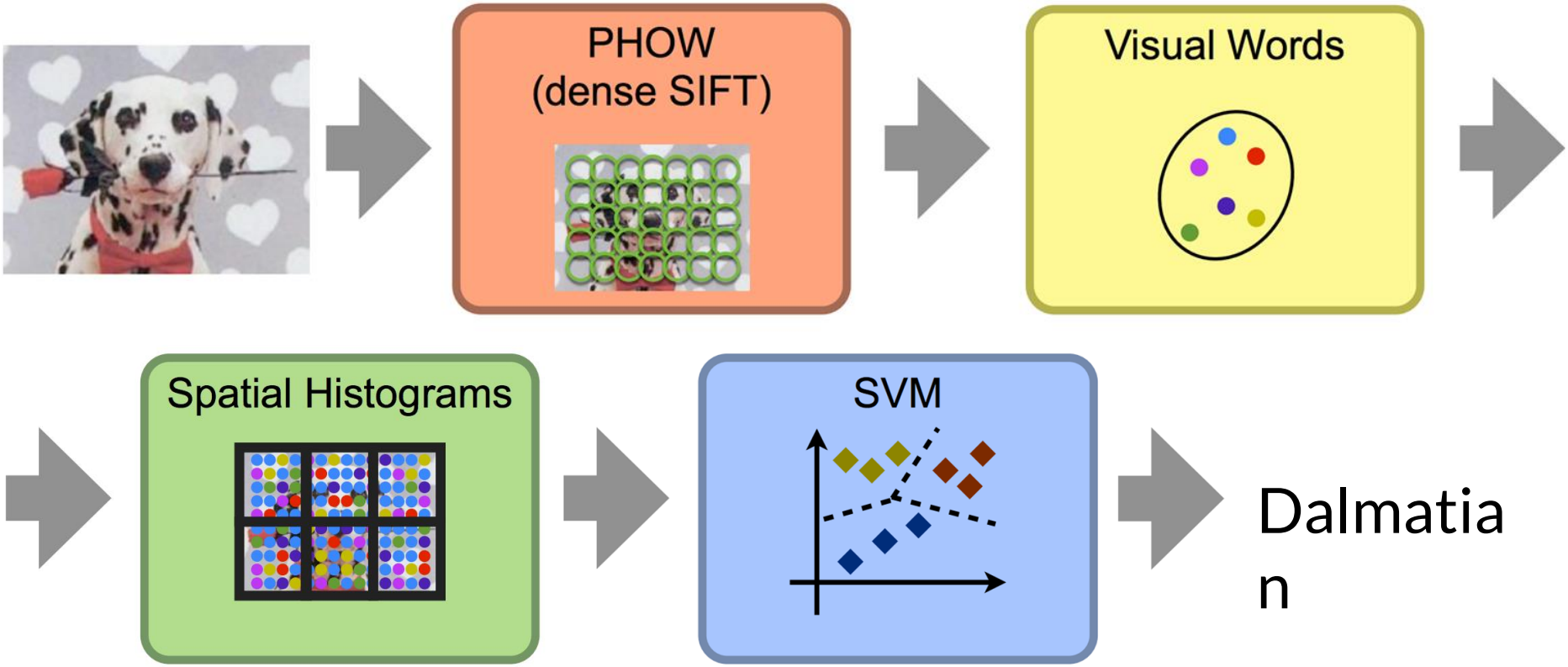
[Fei-Fei et al. 2003]



Andrea Vedaldi, Brian Fulkerson, *VLFeat Tutorial* [http://vision.ucla.edu/~vedaldi/assets/pubs/vedaldi10vlfeat-tutorial.pdf]

Image Classification

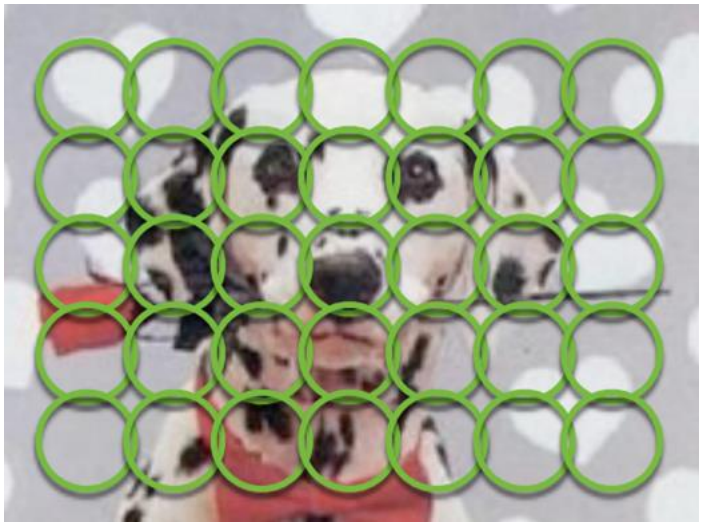
- At a Glance - A Pipeline

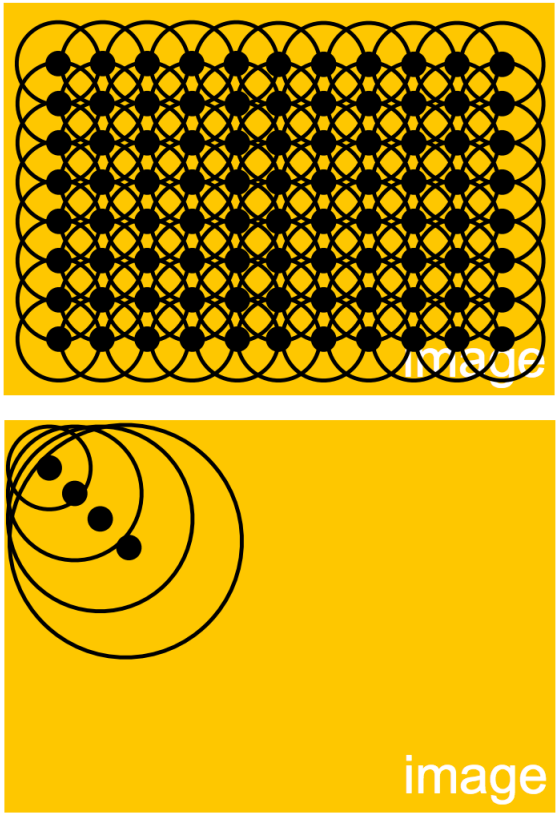


Andrea Vedaldi, Brian Fulkerson, *VLFeat Tutorial* [http://vision.ucla.edu/~vedaldi/assets/pubs/vedaldi10vlfeat-tutorial.pdf]

Feature Extraction

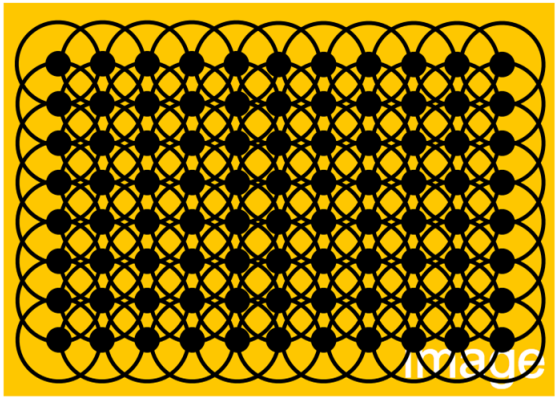
- Dense SIFT
  - Uniform keypoints – No detection
  - Dense multiscale SIFT
  - Descriptors from the uniform keypoints





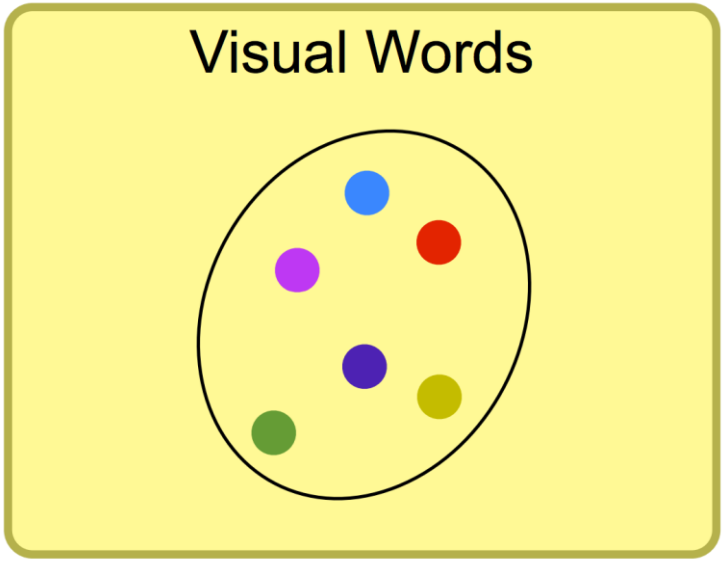
Feature Extraction

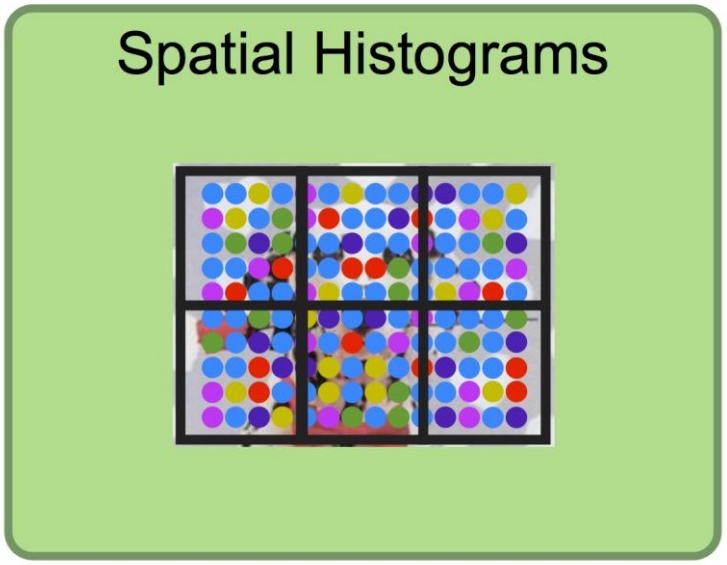
- Dense SIFT
  - Uniform keypoints – No detection
  - Dense multiscale SIFT
  - Descriptors from the uniform keypoints




```
sift = cv2.xfeatures2d.SIFT_create()
# Dense SIFT(Extract SIFT descriptor in grid points over an image)
def denseSIFT(img, step = 5, size = 7):
    rows, cols = img.shape[:2]
    kp = []
    for x in xrange(step,cols,step):
        for y in xrange(step,rows,step):
            kp.append(cv2.KeyPoint(x, y, size))
    kp, des = sift.compute(img, kp)
    return des
```

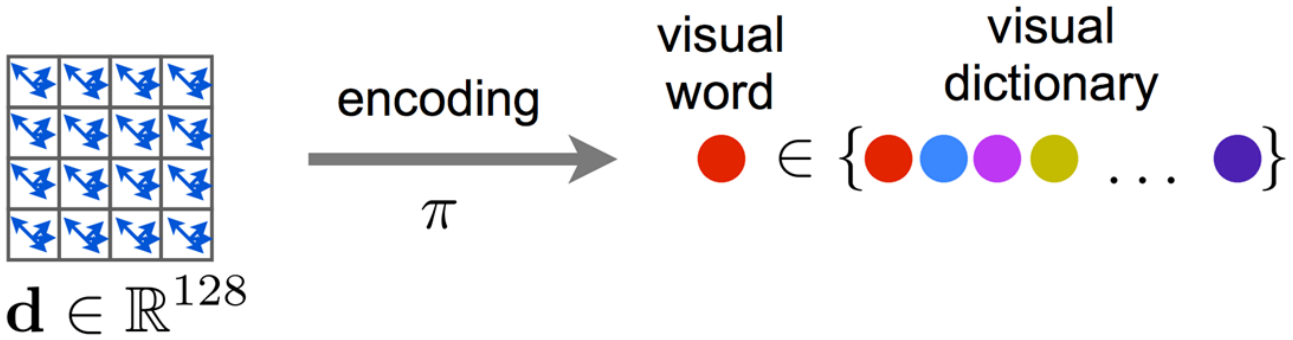
Feature Extraction



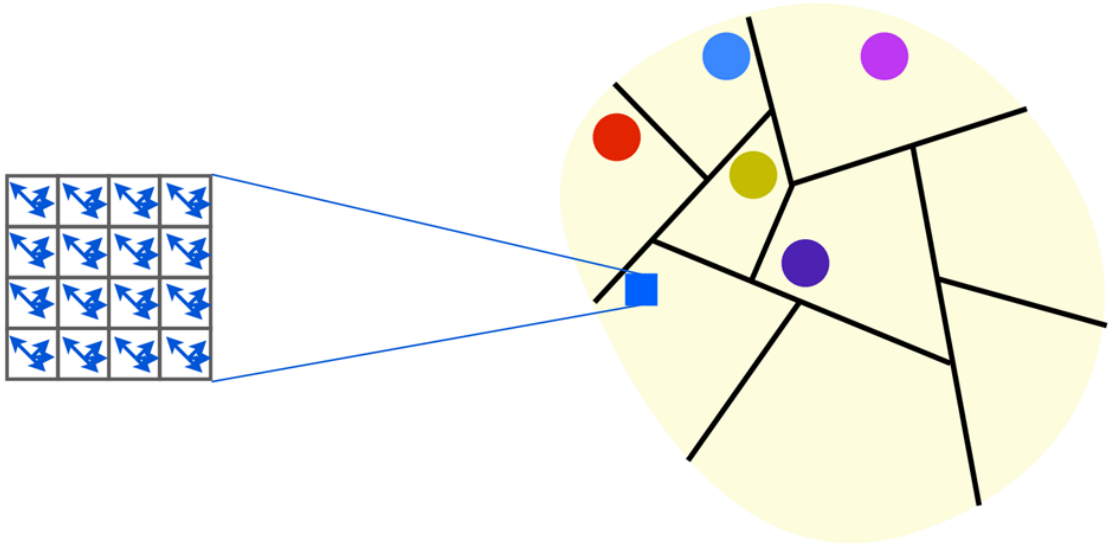


Feature Extraction

- Visual words



- Encoding = clustering [Sivic and Zisserman 2003]
  - vector quantization (k-means) [Lloyd 1982]
  - agglomerative clustering [Leibe et al. 2006]
  - affinity propagation [Frey and Dueck 2007]
  - ...

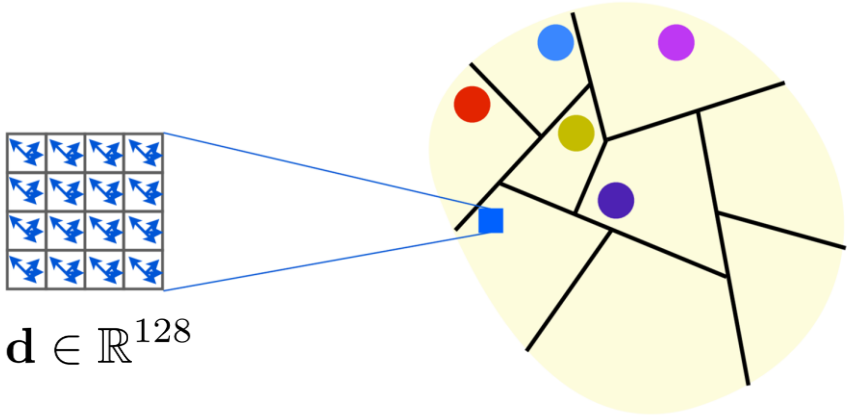


Feature Extraction

- K-means clustering in OpenCV

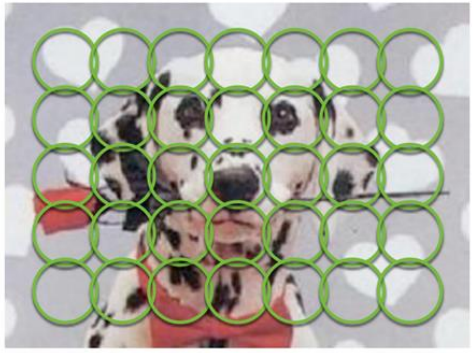
```
# Quantize the descriptors to get the visual words
print "Running K-means clustering (%d -> %d)..." % (PHOW_descrs.shape[0],
                                                    numWords)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 500, 1.0)
attempts = 10
flags = cv2.KMEANS_RANDOM_CENTERS
retval, bestLabels, vocab = cv2.kmeans(PHOW_descrs, numWords,
                                       None, criteria, attempts, flags)
```

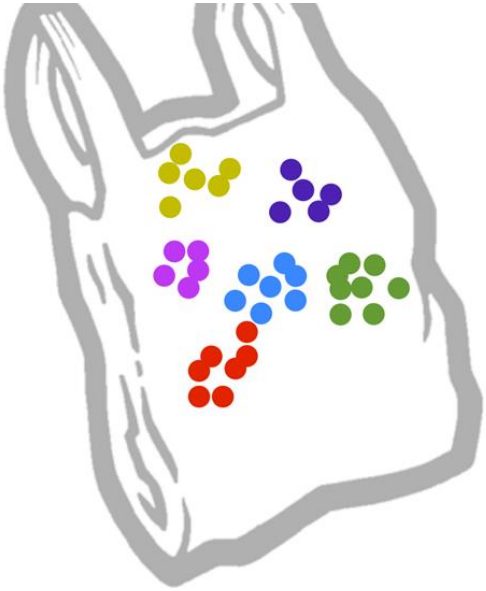
```
# match SIFT features to the words from K-means
bf = cv2.BFMatcher()
matches = bf.knnMatch(des, vocab, k=1)
words = [m[0].trainIdx for m in matches]
```

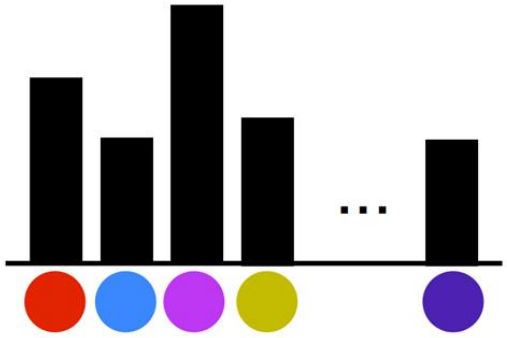


Feature Extraction

- Visual Bag of Words





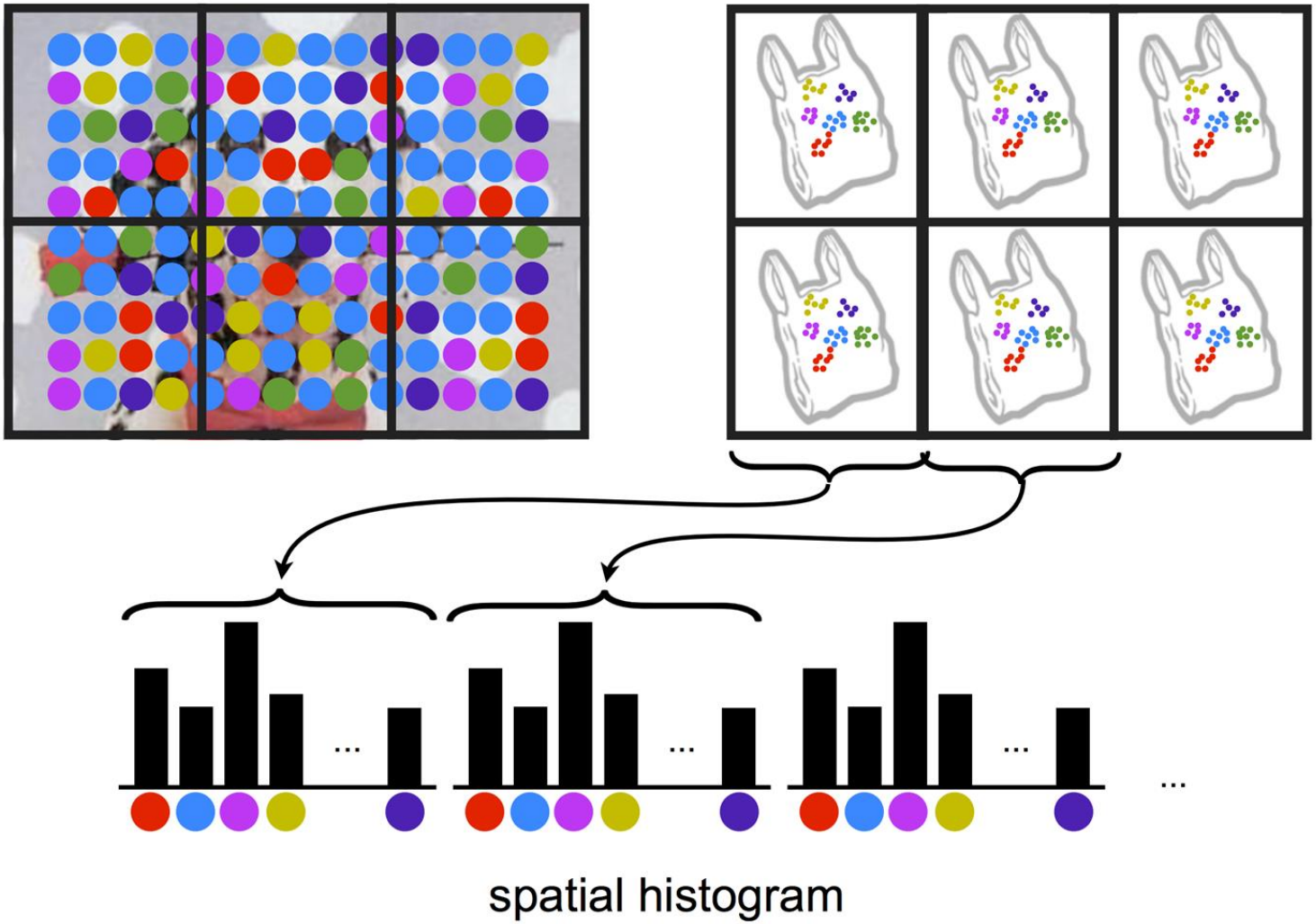


[Csurka et al. 2004]

histogram (bag) of visual words

Feature Extraction

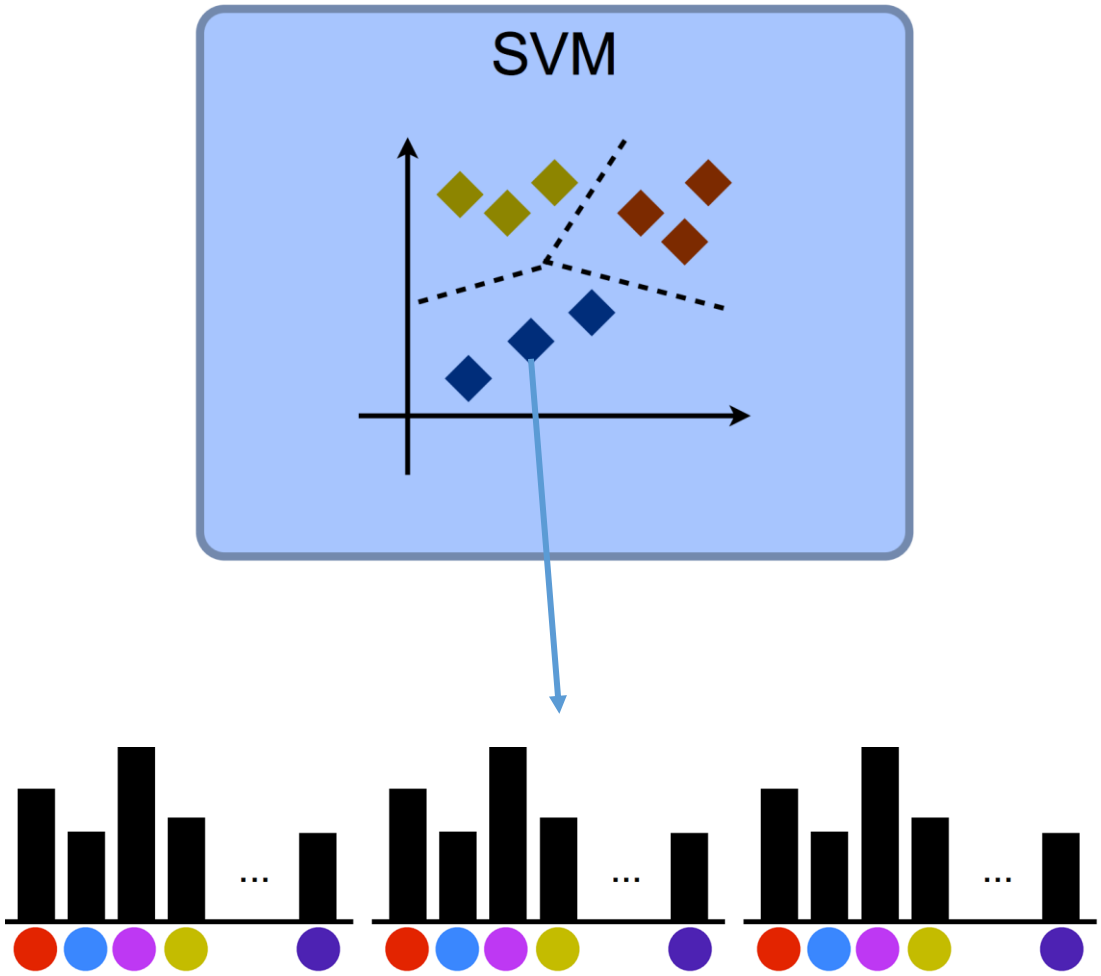
- Spatial Histogram



[Lazebnik et al. 2004]

Classifier

- SVM



Classifier

- SVM in OpenCV
  - Training SVM

```
# Train SVM
print 'Training SVM...'
svm = cv2.ml.SVM_create()
svm.setType(cv2.ml.SVM_C_SVC)  # classification(n > 2)
svm.setKernel(cv2.ml.SVM_LINEAR)  # linear kernel
svm.setC(0.01)
svm.setTermCriteria((cv2.TERM_CRITERIA_COUNT, 10, 1.0)) # term. criteria

svm.train(train_bow, cv2.ml.ROW_SAMPLE, train_labels)
```

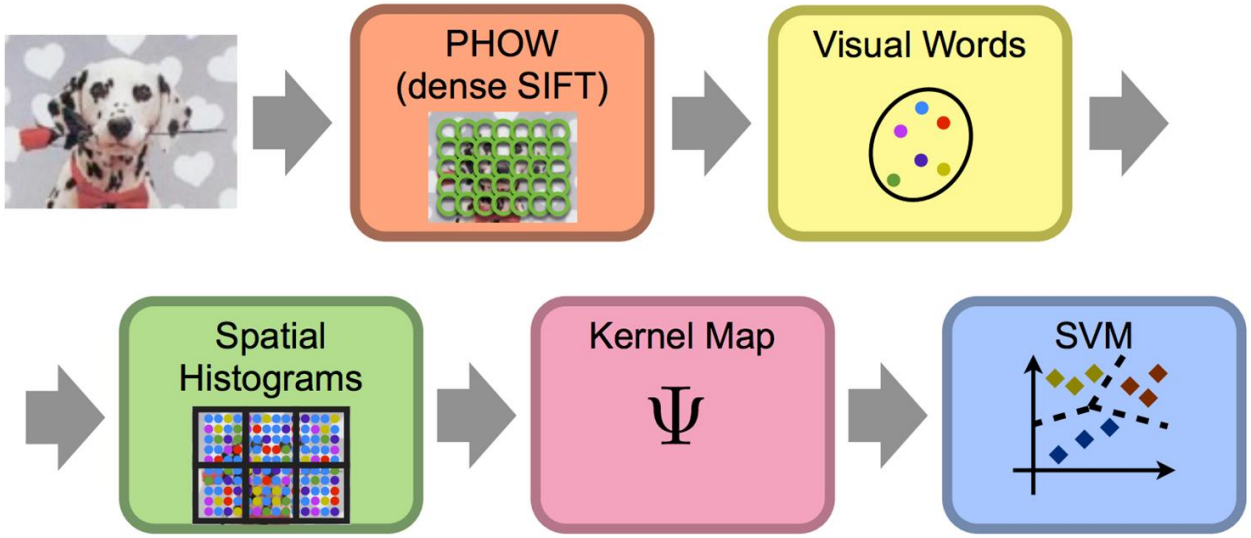
  - Predict with SVM

```
train_preds = svm.predict(train_bow)[1]
print('Training Accuracy: %.6f' % np.average(train_preds == train_labels))
```

```
Training Accuracy: 0.990850
```

Image Classification

- Results
  - ~60% Accuracy using Dense SIFT / VBoW / Spatial Histogram



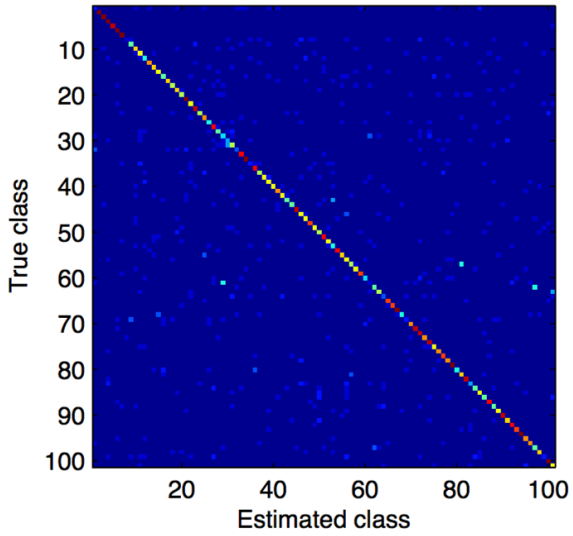
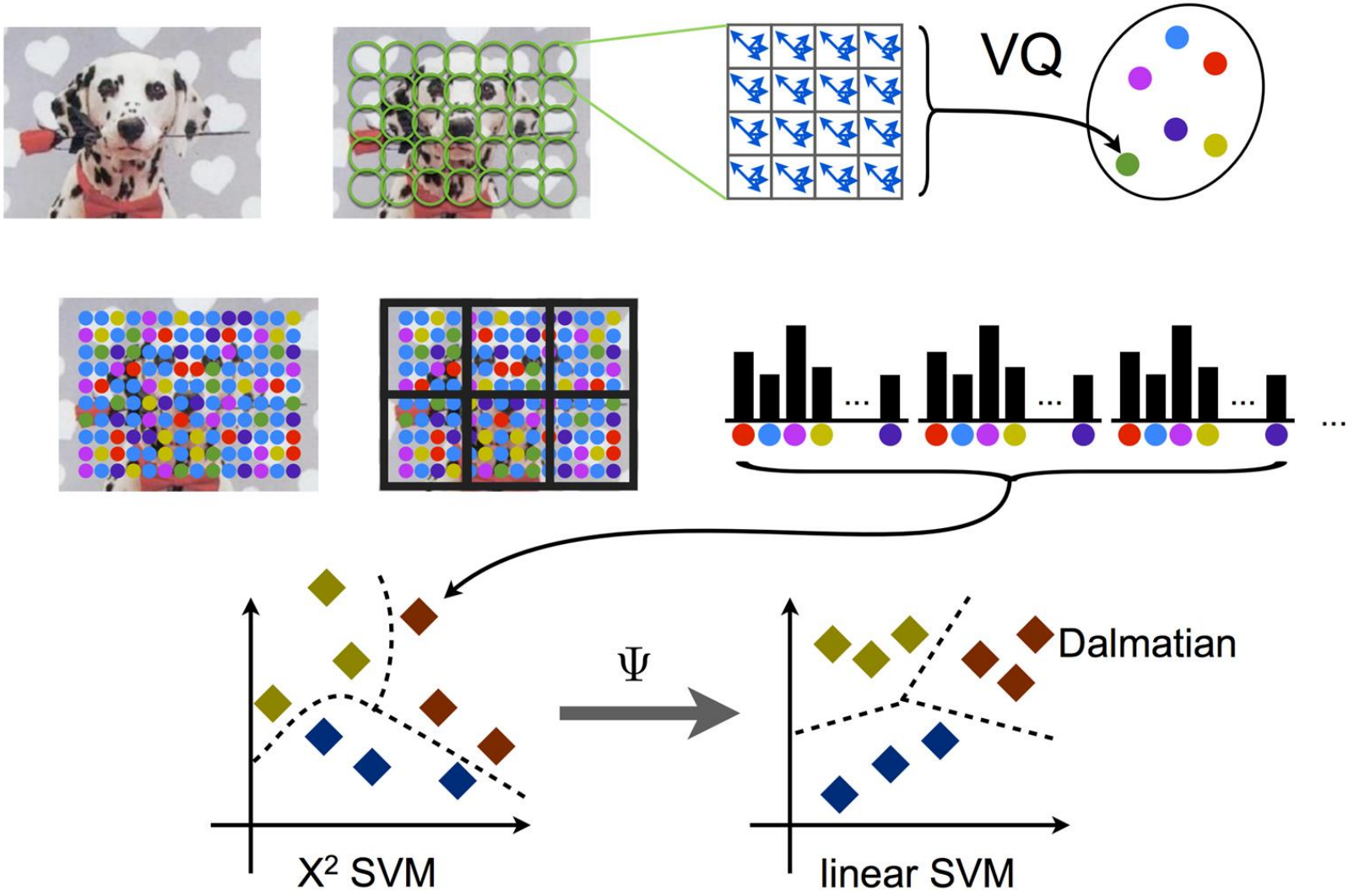


Image Classification

- Image Classification Summary



Let’s Check the Code

3_classification.ipynb

Trouble-shooting

- Kernel crashes with ModuleNotFoundError on 'prompt_toolkit.formatted_text'
  - https://github.com/jupyter/notebook/issues/4050
- OpenCV(3.4.3) /io/opencv_contrib/modules/xfeatures2d/src/sift.cpp:1207: error
  - https://stackoverflow.com/questions/52305578/sift-cv2-xfeatures2d-sift-create-not-working-even-though-have-contrib-instal