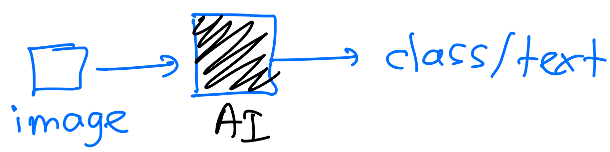
AI is vision

Intelligent is to recognize people, scenes, objects patterns

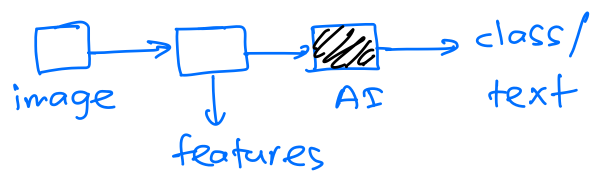
- Face recognition
- Object recognition
- Auto captioning of images
- Robot navigations

Given a digital image, 2 possible ways for recognition

① AI approach



② CV/AI approach



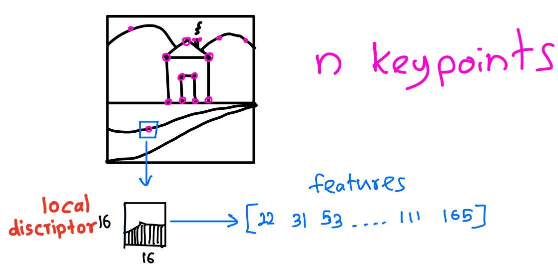
Feature extraction

① Key point oriented (SIFT, SURF, ...)

→ several hundred/thousands of features

② Histogram oriented (LBP, HOG, ELP) → one histogram

Example :



for SIFT you get many feature vectors of length 128.

$$\text{Image} = \bigcup_{i=1}^{n_{\text{keypoints}}} v_i$$ ← union

spare sampling (only sample keypoints)

Challanges: 1) Data is too large

2) Data may not be discriptive

Solutions: embedding / pooling / encoding

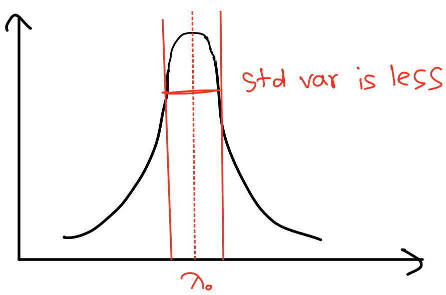
→ Fisher vector (embedding and pooling)

→ VLAD (encoding and increasing discrimination)

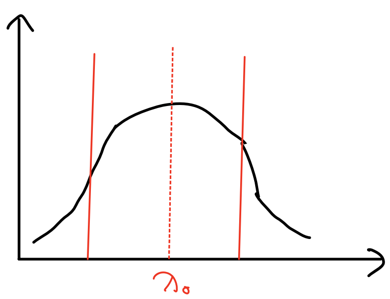
$X = \{x_t, t = 1, 2, \dots, T\}$

u_λ = probability density function which models the generative process of elements of X.

$\lambda = [\lambda_1, \lambda_2 \dots \lambda_M]^T \in \mathbb{R}^M$ parameters of u_λ



fits the data more accurately



Statistics : The score function is the gradient (partial derivative) wrt parameter λ of the natural logarithm of the likelyhood function.

Score function $= \underbrace{\nabla_\lambda}_{\text{partial derivative wrt } \lambda} \log u_\lambda(x)$

a,b independent and identically distributed random variables.

$f(a,b|u) = f(a,u) * f(b,u)$ is likelyhood

function a,b given distribution u

$\log(f(a,b|u)) = \log(f(a,u)) + \log(f(b,u))$ log likelyhood

Score function $= \nabla_\lambda \log u_\lambda(x)$

$= \nabla_\lambda \log P(x|u_\lambda)$

Let X be the set of D-dimensional local descriptors extracted from an image (e.g SIFT) Don't have intelligence

$g_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t)$ fisher vector

function g that has parameter λ operating on X

Fisher vector is a sum of normalized gradient statistics compute for each descriptor (feature vector)

The operation $x_t \longrightarrow f_{FK}(x_t) = L_\lambda \nabla_\lambda \log u_\lambda(x_t)$

read the data and put it on another space

f uses fisher kernal for each data point x_t

is an embedding of local discriptors x_t in a higher dimentional space which is easier for a linear classifier.

L_λ : Cholesky Decomposition

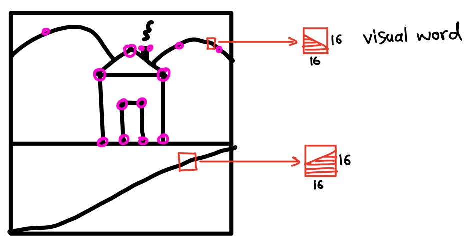
$$F_\lambda^{-1} = L_\lambda^{T} L_\lambda$$

$$K_{FK}(X,Y) = G_\lambda^{X^T} F_\lambda^{-1} G_\lambda^{X}$$

Fisher kernel - take 2 numbers and tell how similaraty it is in higher dimentional space.

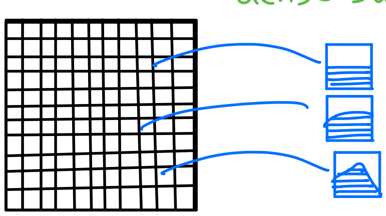
VLAD (Vector for locally aggregated descriptors)

How to recognize images? ⟶ Bag-of-visual-words



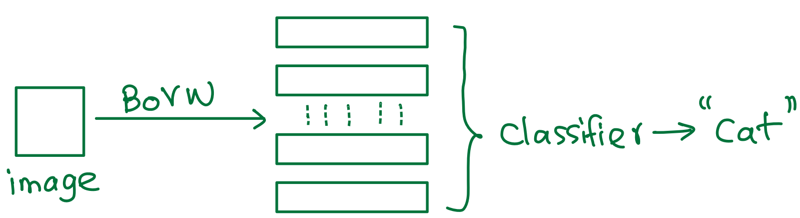
Given an image I, divided it into small cells/windows of size nxn (eg: 16x16)

dense sampling



We vectorize visual words for convinient calculations.

Theory:



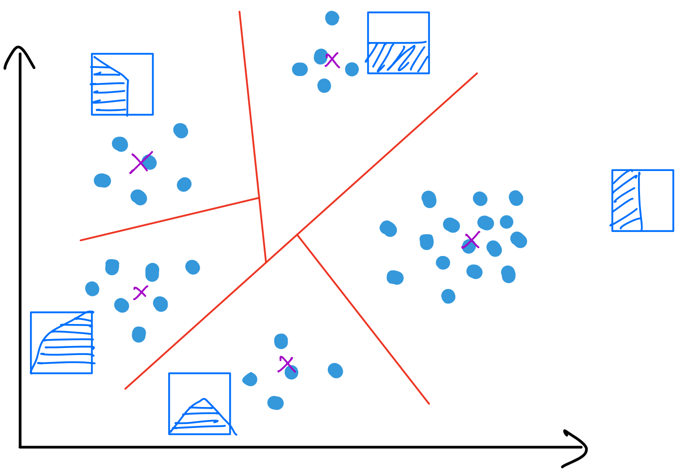
Practice:
- too many vectors
- Redundancy
- Noise

General approach

Build a codebook (dictionary) $C = \{c_1, c_2, \dots, c_n\}$ from $m \ggg n$ feature vectors (vectorized visual words)

idea : Use a clustering algoritm like k-means

c_i is the centre of the n classes found in the data



× class center

Core idea of VLAD

Accumulate for each visual word c_i, the difference of the vectors x assigned to c_i, $x - c_i$ ie distribution of data wrt the class centers.

$$V_{i,j} = \sum_{NN(x) = c_i} x_j - c_{i,j}$$

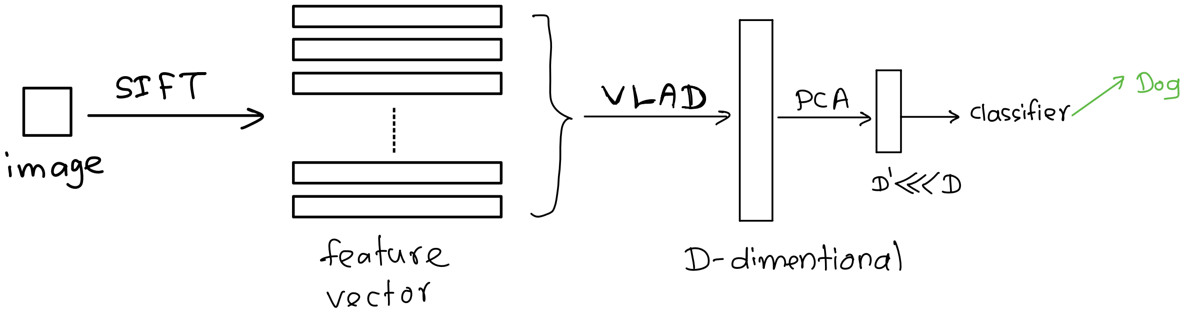
use these in dictionary

jth component of the discriptor x

corresponding cluster centre

L2-normalization, $V : \dfrac{V}{\|V\|}$

Final chain


SIFT
image
feature
vector
VLAD
PCA
D-dimentional
D'⋘D
classifier
Dog