Image Similarity Analysis with LSH

Group Members:

Kerem Ayöz
Mert Epsileli
Safa Aşkın
Simon Arda Yuvarlak

Outline

- Introduction / Project Definition
- Feature Extraction Methods
  - Image Processing
  - CNN
  - SIFT & KAZE
- LSH
  - Jaccard Distance
  - Euclidean Distance
  - Cosine Distance
- Future Improvements

Introduction

Project Definition

Aim

-Get features from images
-Finding similar images (candidate pairs) by using Locality Sensitive Hashing.

Datasets

-Different classes of images from ImageNet
ex. (Truck, airplane, knife, orange, tree, car, church, dog…)

-Big enough to get inefficient performance from pairwise comparison algorithms.
In total 24.000 images (300 x 300)

Feature Extraction

Image Processing

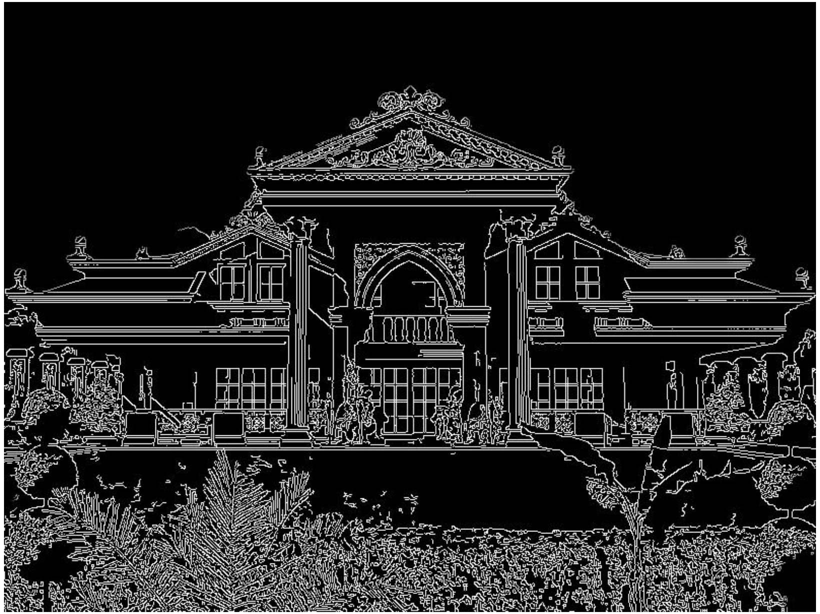
CNN

SIFT & KAZE

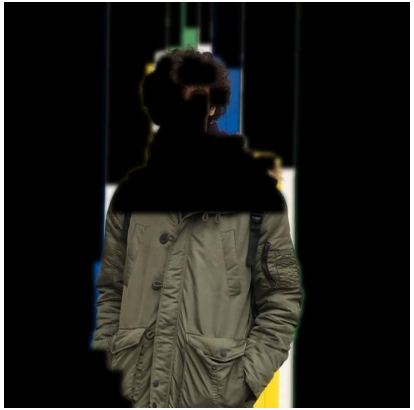
Image Processing

Edge Detection

- Simplest feature extraction method
- Useful result but ”NOISY”.



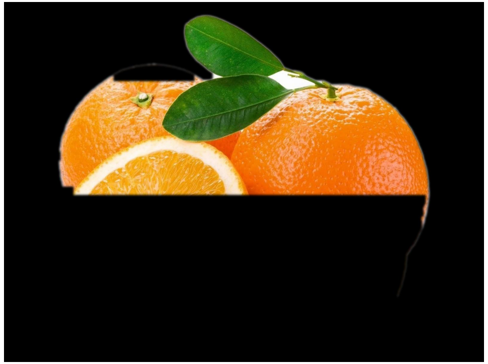
Segmentation



- First trial to get better (clear) results.
- Good results for home.jpg example.

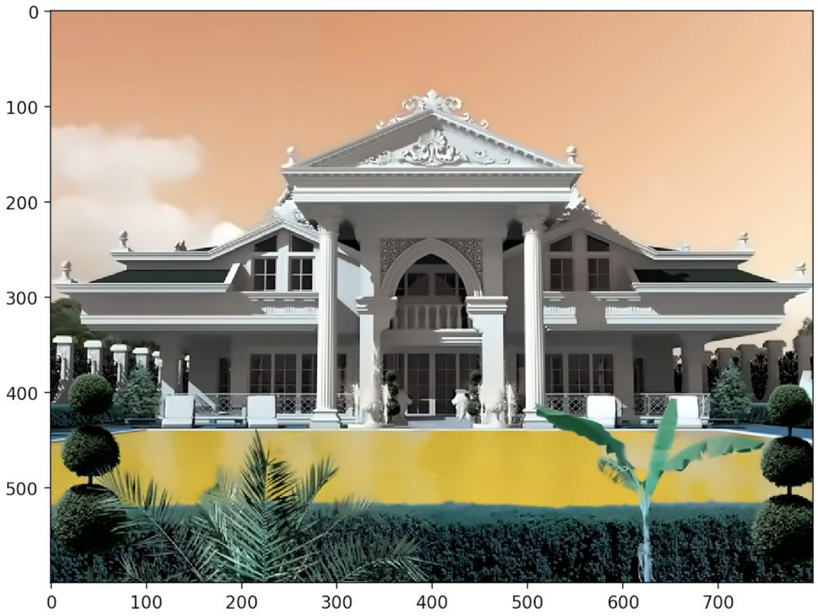
- Current practices are using Machine Learning for segmentation.

- Failed for some crucial datasets.
  (Human recognition, fruit recognition)



Noise Clearing

- Better results but not enough
- Still lots of insignificant detail..



Sobel Filtering

- Almost perfect result to binarize the features.
- Threshold value (120)

Value: 80 too many details
Value 150 missing important features

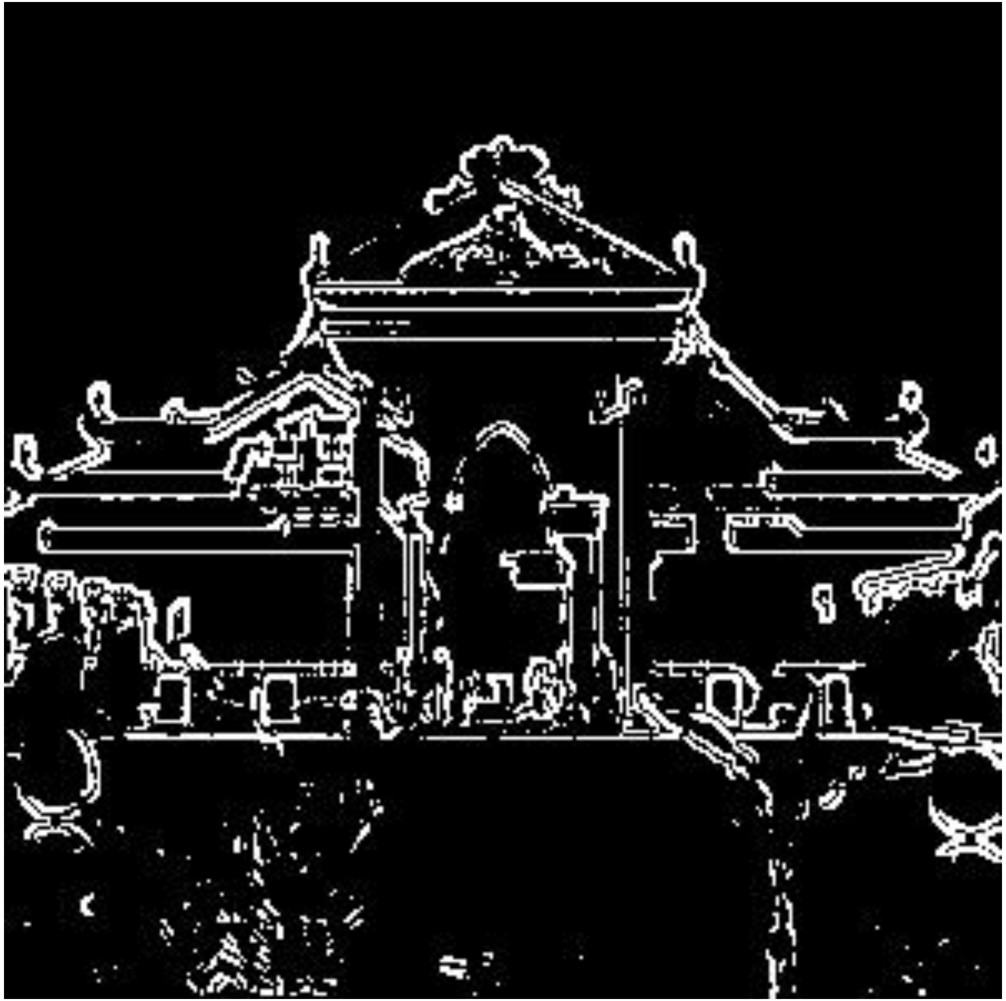
```
for i in range (height):
    print("\n")
    for m in range(width):
        if(grad[i][m] < 120):
            grad[i][m] = 0
            imageArray[counter].append(0)
        else:
            grad[i][m] = 255
            imageArray[counter].append(1)
        print(grad[i][m], end=" ")
counter = counter + 1
```

Binary Image Result

(Sobel Filtering)

Accuracy Rate with LSH

- 2-classes of images = %60.2
- 17-classes of images = %46.8



CNN

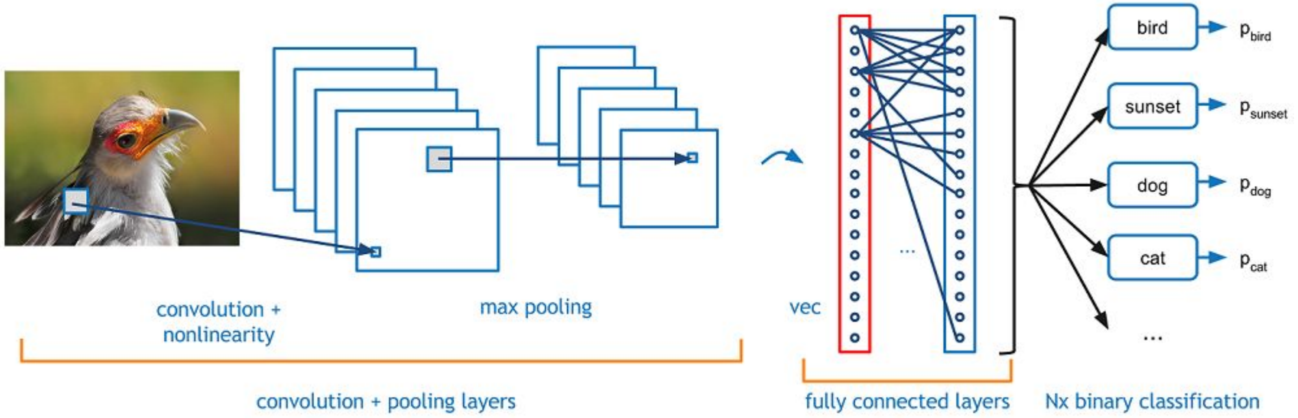
Convolutional Neural Network

What is Convolutional Neural Network？

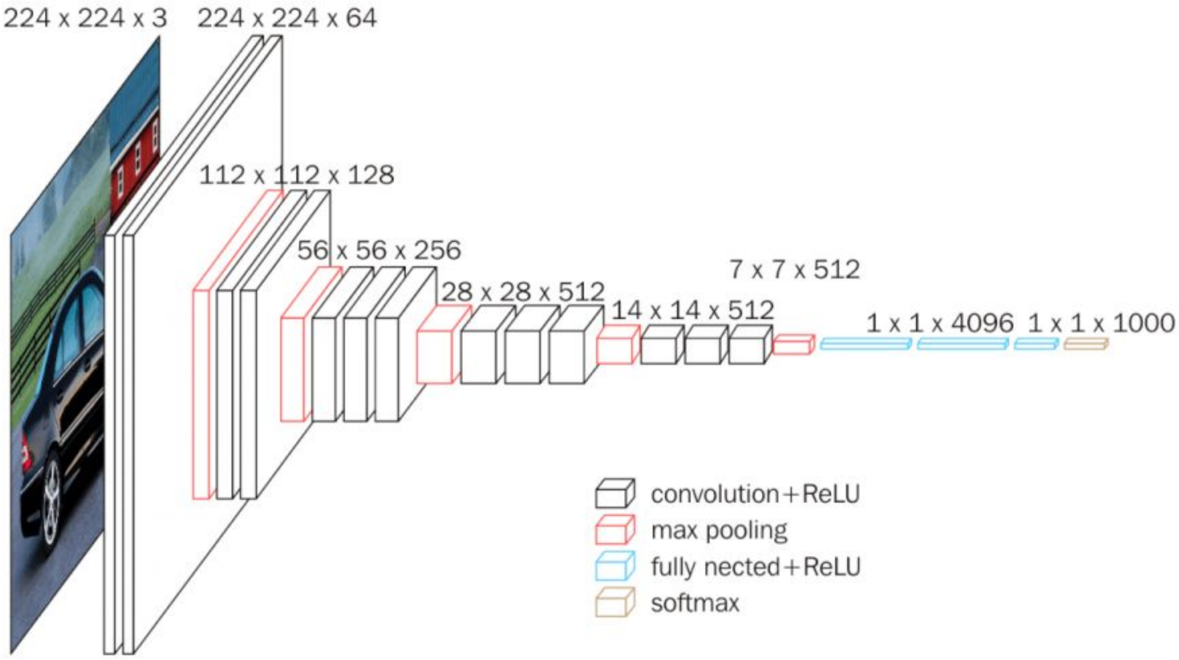
Used for image classification

Extracts local features by a technique called “Convolution”

Learns the features of the images by training.



How we extracted features ?



VGG-16

- Good accuracy on ImageNet
  - *Top-1 = %70.5*
  - *Top-5 = %90.0*
- Simple model compared to others
- Convolution layers represent features
- 7x7x512 = 25.088 features
- Binarized features

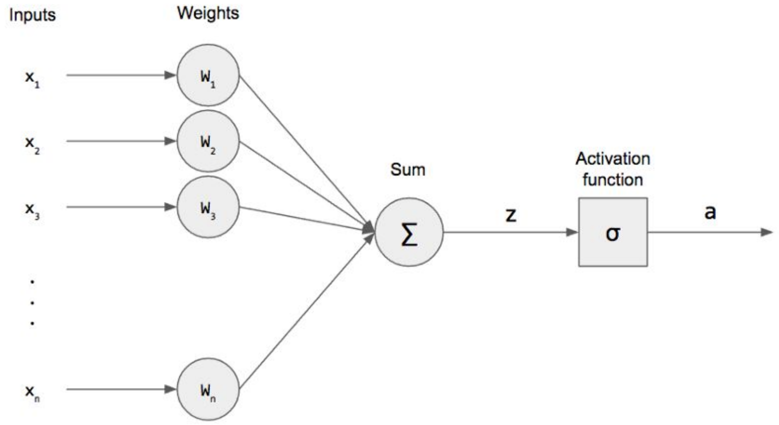
What are those features ?

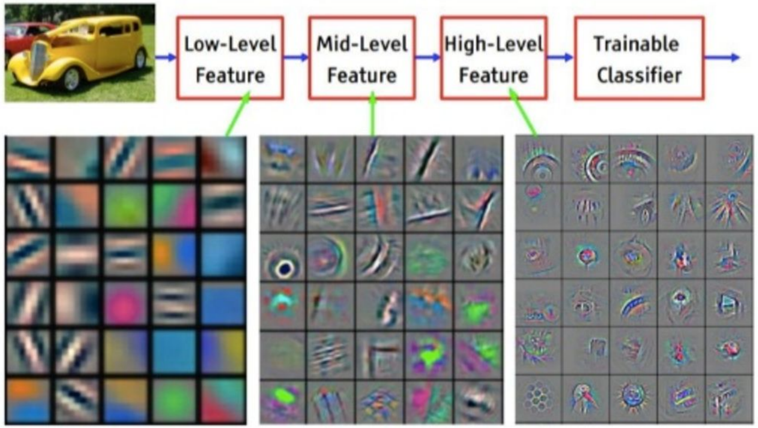
25.088 neurons in the last convolution layer.

On the average %7 of them are fired (non-zero) for single image.

Represents hidden features of input image.

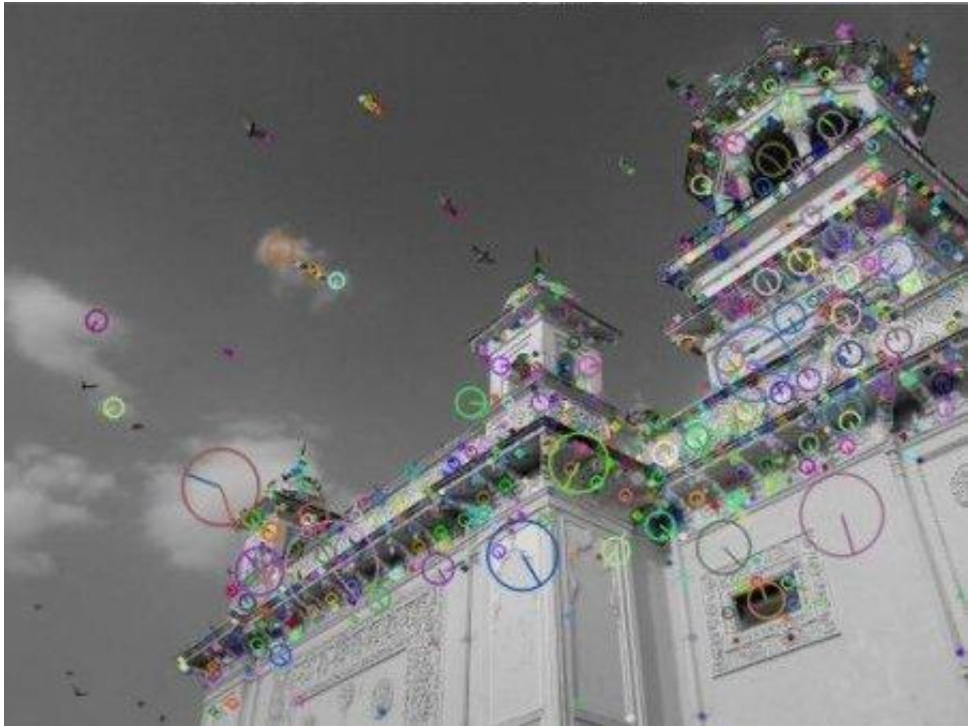
Each layer represents different set of features.





SIFT & KAZE

SIFT



- SIFT is an algorithm which finds key points in images.
- It defines every key point with a 128 dimensional vector.
- Hence, every image is defined by key point times 128 dimensional vectors.
- If Euclidean Distance is used, the similarity of images can be found.

KAZE

- KAZE is an algorithm which finds key points in images.
- It defines every key point with a 64 dimensional vector.
- Hence, every image is defined by key point times 64 dimensional vectors.
- If Euclidean Distance is used, the similarity of images can be found.

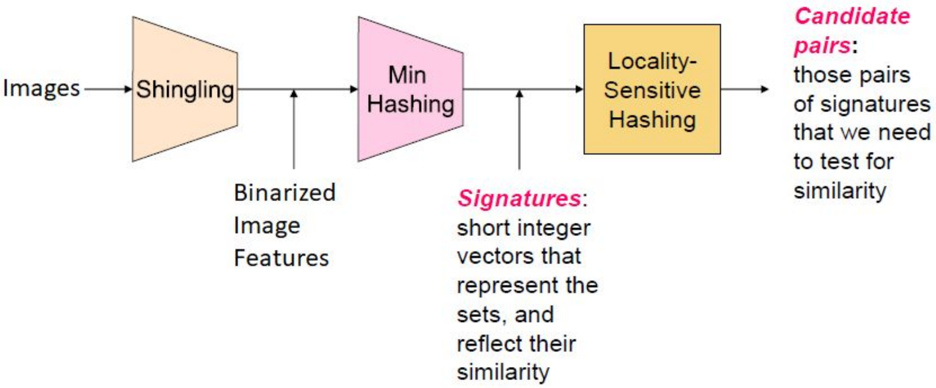
Locality Sensitive Hashing

Jaccard Distance

Euclidean Distance

Cosine Distance

LSH for Jaccard Distance



Hash(Hash(Hash(Hash(Hash(Hash(...))))))

Row Hashing

Generated random parameters for Universal Hash Function
For each hash function h(i);

- a(i), b(i) ~ Uniform(0,k), k is a hyperparameter
- p is the smallest prime number greater than N

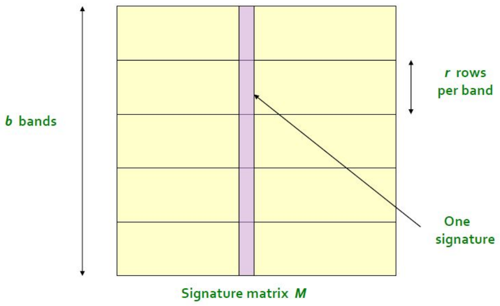
Universal hashing:
$h_{a,b}(x)=((a \cdot x+b) \bmod p) \bmod N$
where:
a,b … random integers
p … prime number (p > N)

Min Hashing

Take minimum row hash value for each image and for each h(i)
Signature size is a hyperparameter

Locality Sensitive Hashing

Hash each band of images into related hash table
b, r are hyperparameters



Using CNN Features

Accuracy Rate with Jaccard Distance LSH

2-classes of images

(b=25, r=4, signature=100) = %73.4

(b=20, r=5, signature=100) = %78.3

17-classes of images

(b=50, r=10, signature=500) = %66.7

(b=25, r=20, signature=500) = %71.2

Using Sobel Features

Accuracy Rate with Jaccard Distance LSH

2-classes of images

(b=25, r=4, signature=100) = %72.4

(b=20, r=5, signature=100) = %73.2

17-classes of images

(b=50, r=10, signature=500) = %55.3

(b=25, r=20, signature=500) = %60.1

Performance of Jaccard Distance LSH

LSH for Euclidean Distance

- Points in n-dimension to define images

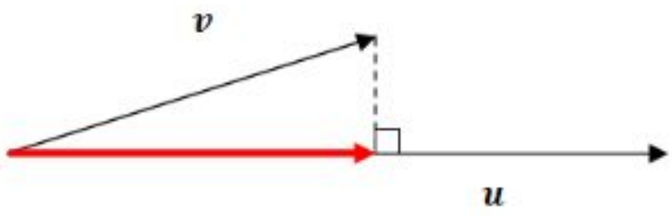
- Where n:
  - Width of image x Height of image
  - Feature array which comes from Convolutional Neural Network
  - Feature array which comes from Kaze
  - Feature array which comes from SIFT

Step 1: Find Projection Coordinate of the Point on a n-dimensional Random Vector

- Create a n-dimensional random vector with non zero values.
- Find the coordinates of projection of feature arrays on this vector by using;

$$proj_u v = \left(\frac{u \cdot v}{|u|} \right) \frac{u}{|u|}$$

Proof of Projection Coordinate Formula



$$\cos\theta = \frac{u \cdot v}{|u||v|}$$

$$\cos\theta = \frac{\text{scalar projection}}{|v|}$$

$$\frac{u \cdot v}{|u||v|} = \frac{\text{scalar projection}}{|v|}$$

$$\frac{u \cdot v}{|u|} = \text{scalar projection}$$

$$proj_u v = \left(\frac{u \cdot v}{|u|}\right)\frac{u}{|u|}$$

Step 2: Apply Dot Product to the Projection Point and a Random n-dimensional Vector

- We are using this method to find a hash value.
- Create a random n-dimensional vector with non zero values.
- Hash value = (((Random Vector) . (Projection Point)) / z) (mod k)
  (z is determined according to range of hash values.)
  (k is determined according to size of hash table.)

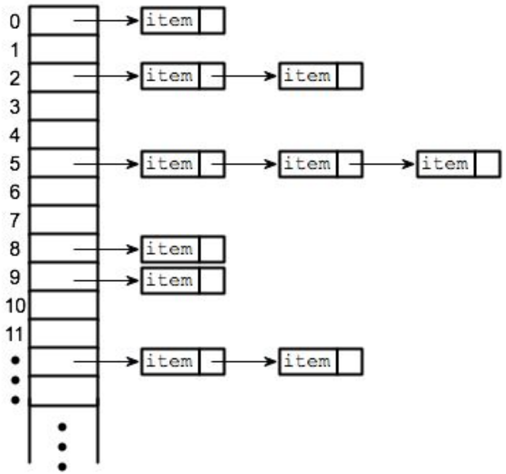
Unique Hash Values (Proof in 2D)

- We need to prove that the results of dot product of **the points on a line** and **a random vector** are different. Assume the random vector has non zero values. (m,n are not 0.)
- $ax+by = c$ (A line)
- (m, n) (Random Vector)
- $(x_1 , (c-ax_1)/b), (x_2 , (c-ax_2)/b)$ (2 Points on the line)
- $mx_1+n(c-ax_1)/b = mx_2 + n(c-ax_2)/b$ (Equality of dot product of 2 points and the random point)
- $m(x_1-x_2) = (n/b)((c-ax_2)-(c-ax_1))$
- $mb/n(x_1-x_2) = a(x_1- x_2)$
- $mb = na$
- $\mathbf{m/a = n/b}$

This shows that m/a must be equal to n/b to find same hash values for different points. This has really low probability in n-dimension.

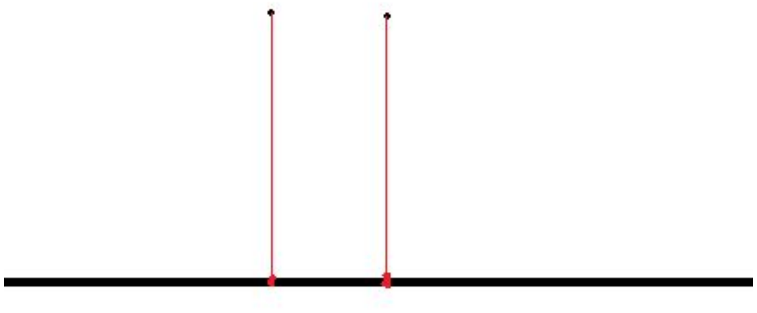
Step 3: Create a Hash Table

- Create a hash table, and add the ID of image to hashTable[Hash Value].
- Choose the images in same bucket as candidate pair.

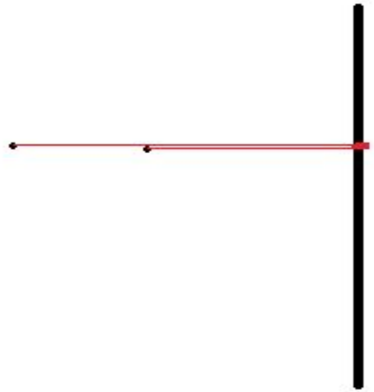


Step 4: Do First 3 Steps 100 Times

- Do first 3 steps 100 times to find candidate pairs which cannot find yet.
- For example, we assume 2 image which are similar. Their projection for 2 different random line.



In this case, they are not candidate pair because their projection are too far each other.



In this case, the are found as candidate pair, so this

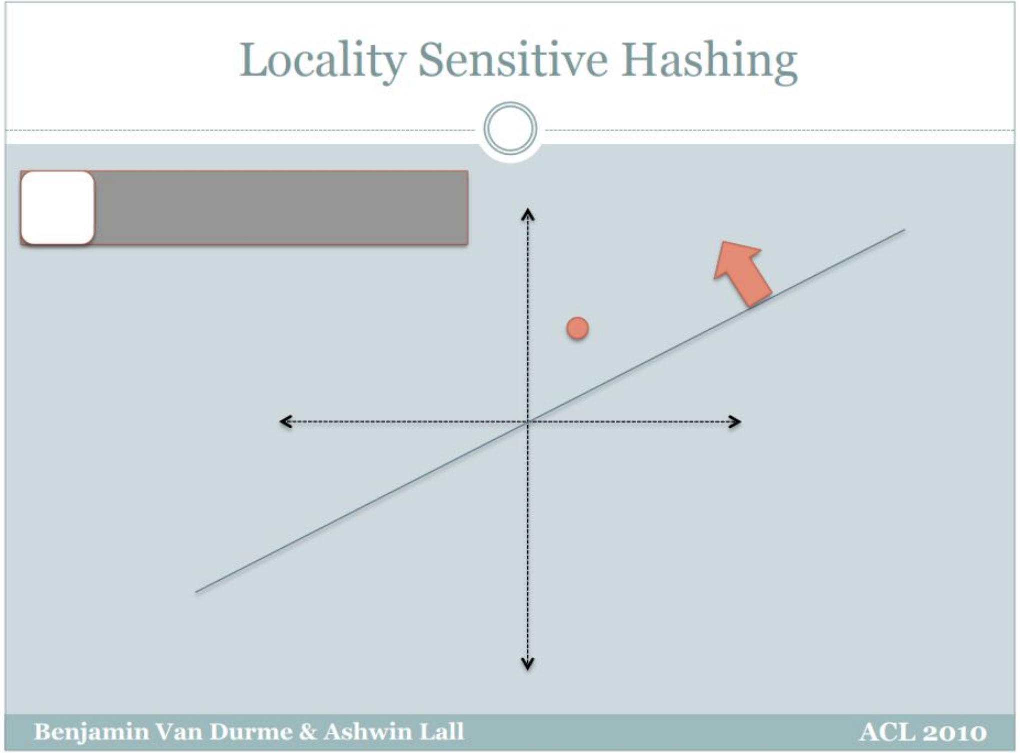
LSH for Cosine Distance

- Points in n-dimension to define images
- Where n:
  - Width of image x Height of image
  - Feature array which comes from Convolutional Neural Network
  - Feature array which comes from Kaze
  - Feature array which comes from SIFT

Idea of Cosine Distance LSH

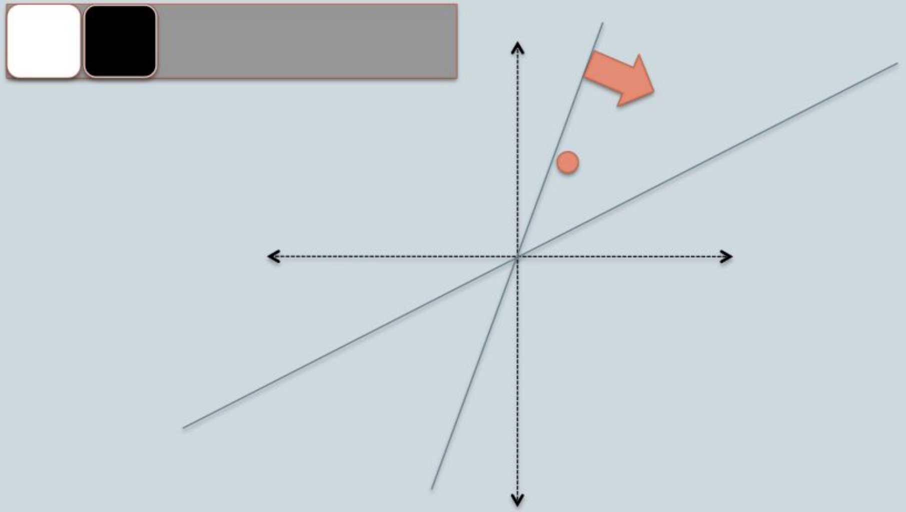
- Divide the n-dimensional space to regions by using many planes.
- Find the signature array of the feature arrays by using the place of them according to planes.

Visualization in 2D
(Left -> White , Right -> Black)

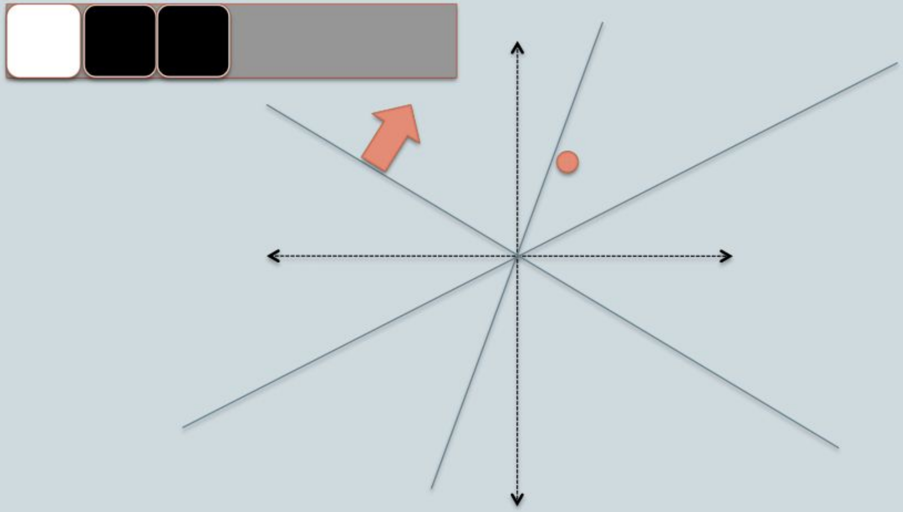


Reference:
http://www.cs.jhu.edu/~vandurme/papers/VanDurmeLallACL10-slides.pdf

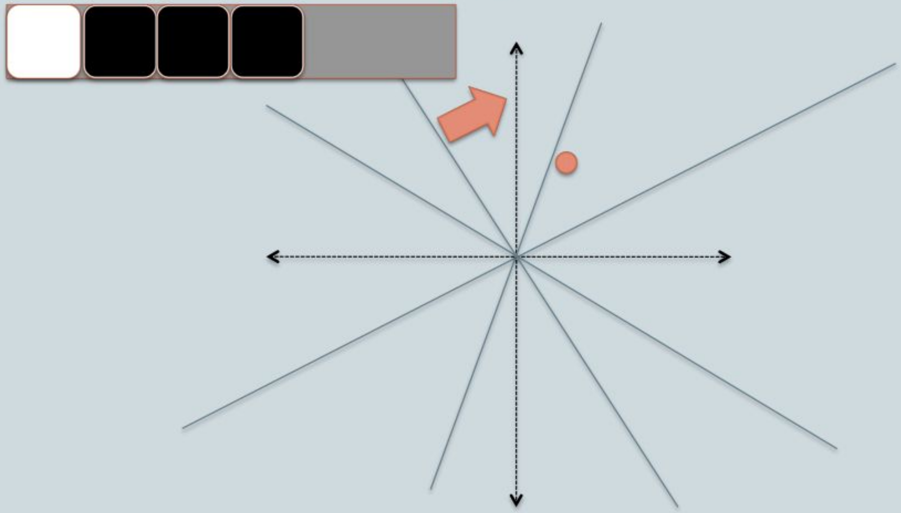
Locality Sensitive Hashing



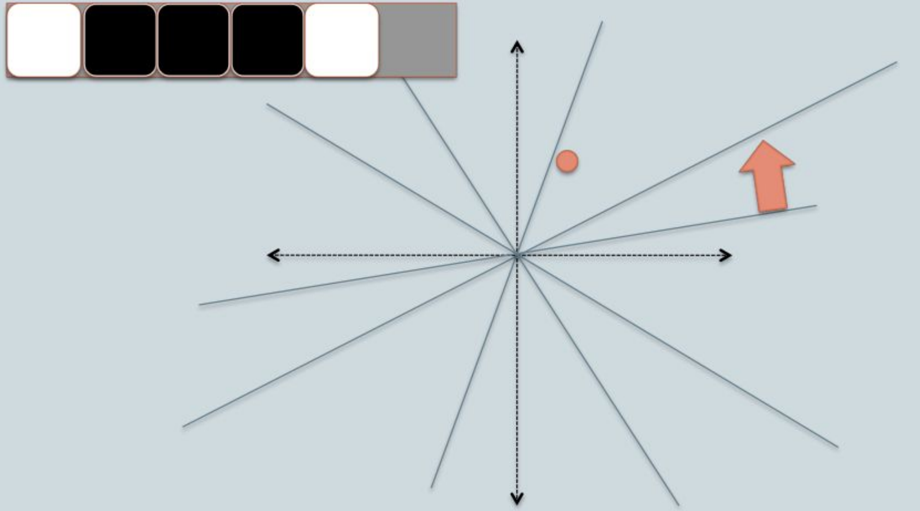
Locality Sensitive Hashing



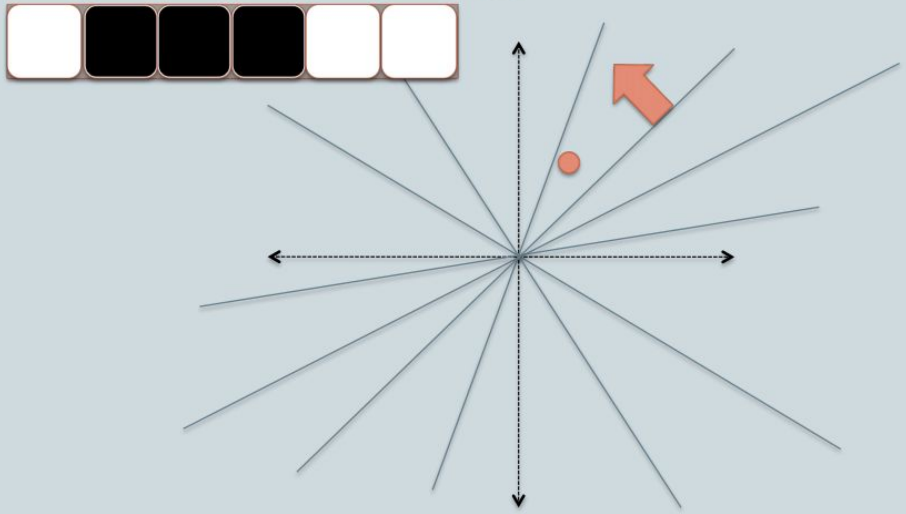
Locality Sensitive Hashing



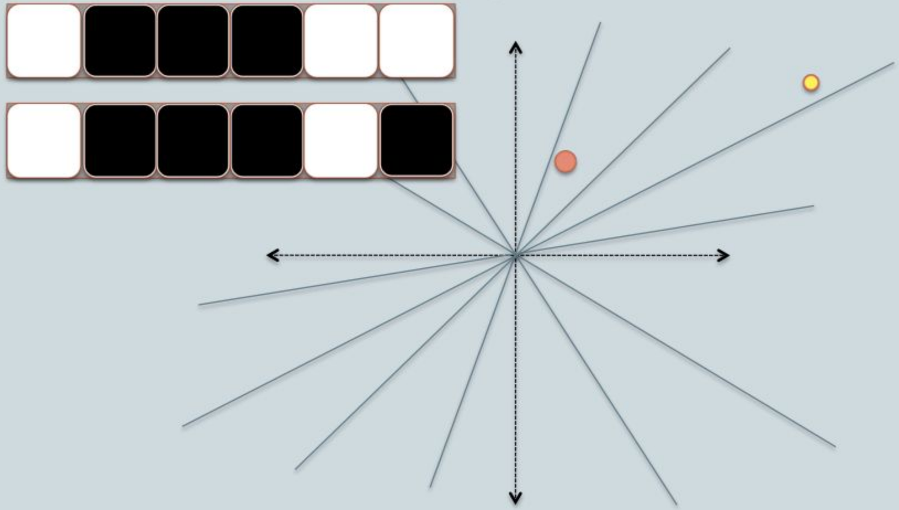
Locality Sensitive Hashing



Locality Sensitive Hashing



Locality Sensitive Hashing



Last Step

- If we write 0 for blacks and 1 for whites, we get a signature matrix for every point.
- In this case;
    - Red Point = (1,0,0,0,1,1)
    - Yellow Point = (1,0,0,1,0)
- After this, we can use min hashing to find candidate pairs.

Thanks for Listening