Hashing Techniques

윤성의 (Sung-Eui Yoon)

Professor

KAIST

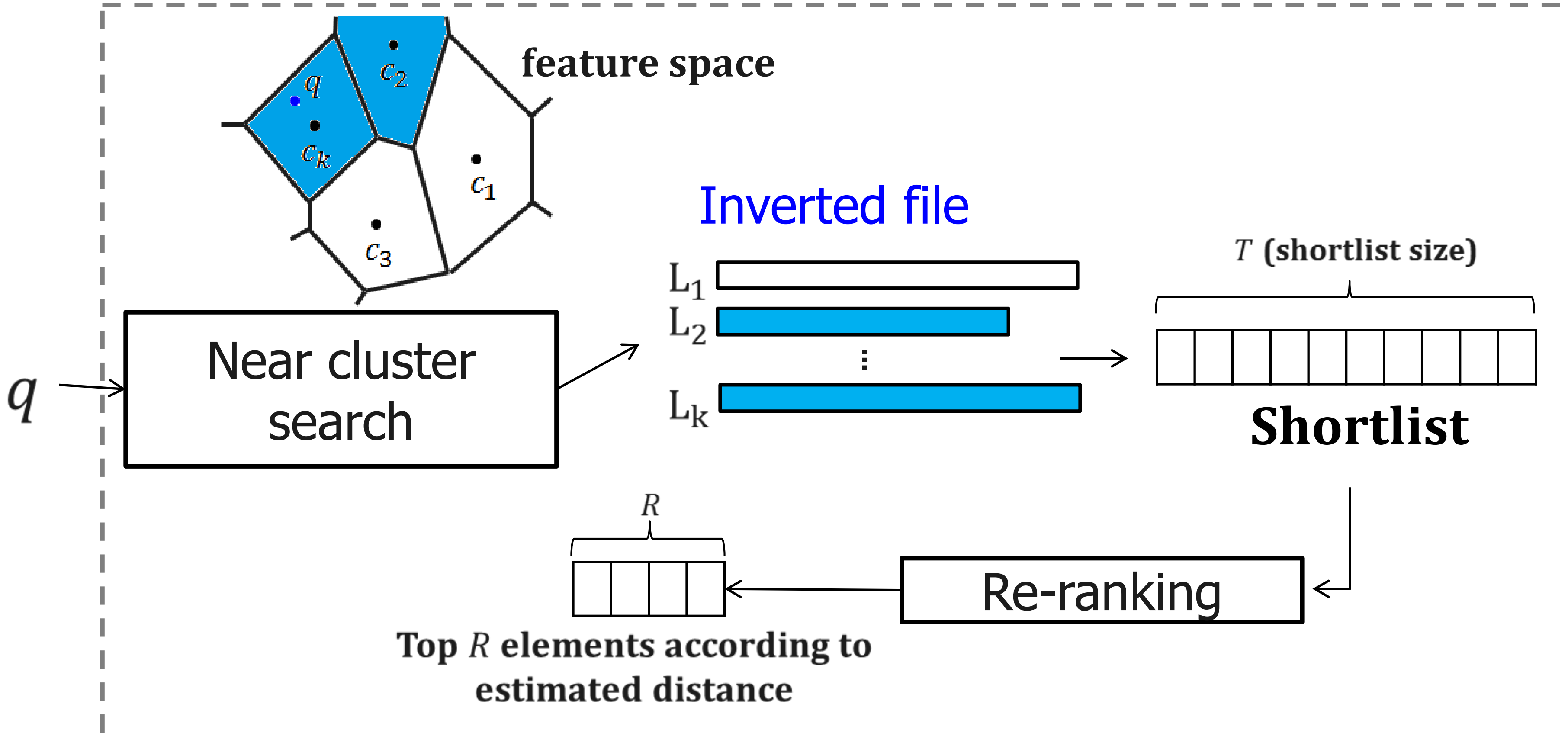
http://sgvr.kaist.ac.kr

Class Objectives

- **Understand the basic hashing techniques based on hyperplanes**
  - **Unsupervised approach**
- **Sematic hashing using deep learning**

- **At the last class:**
  - **Discussed re-ranking methods: spatial verification and query expansion**
  - **Talked about inverted index**

Review of Basic Image Search



Ack.: Dr. Heo

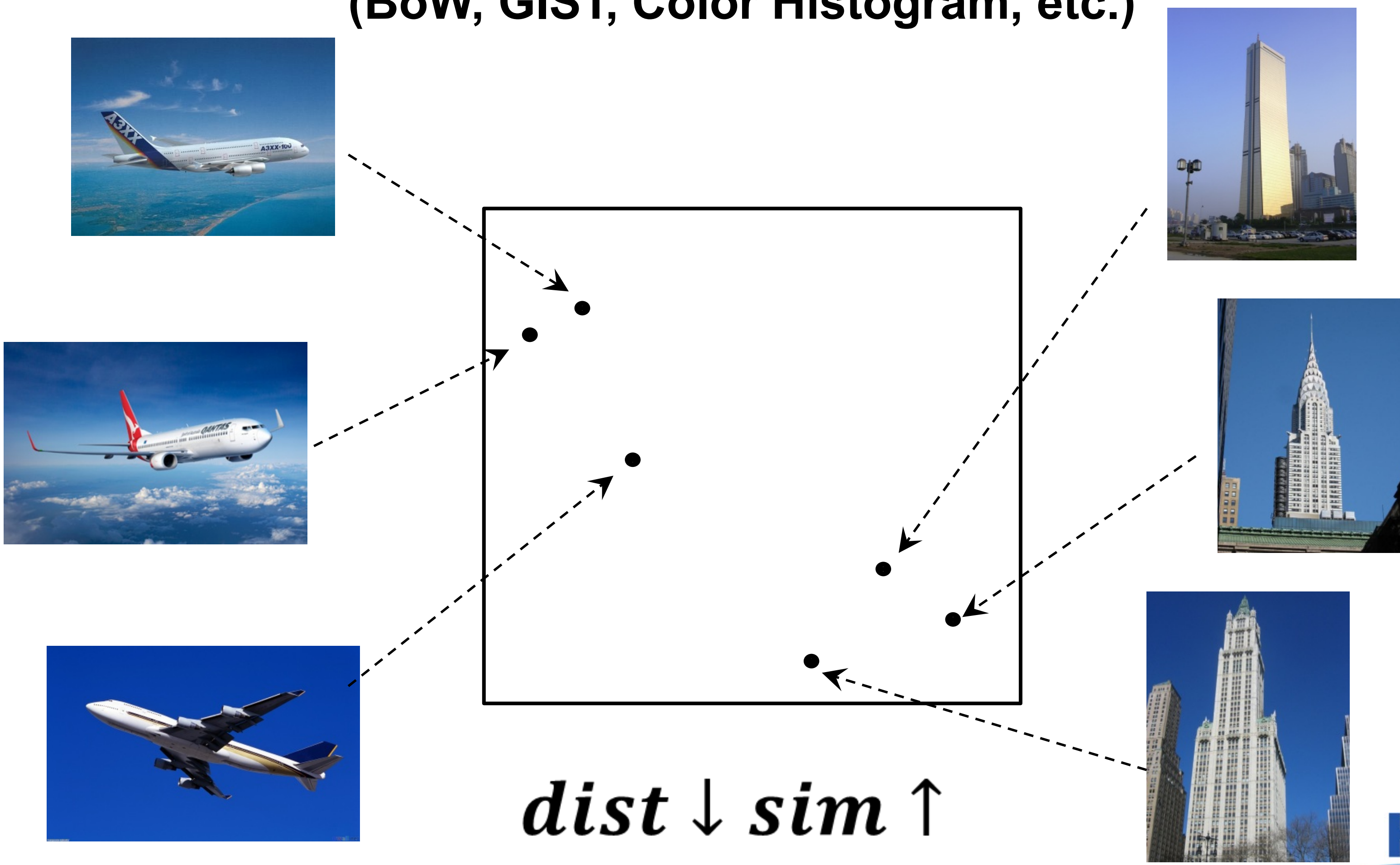
Image Search

Finding visually similar images

Image Descriptor

High dimensional point

(BoW, GIST, Color Histogram, etc.)

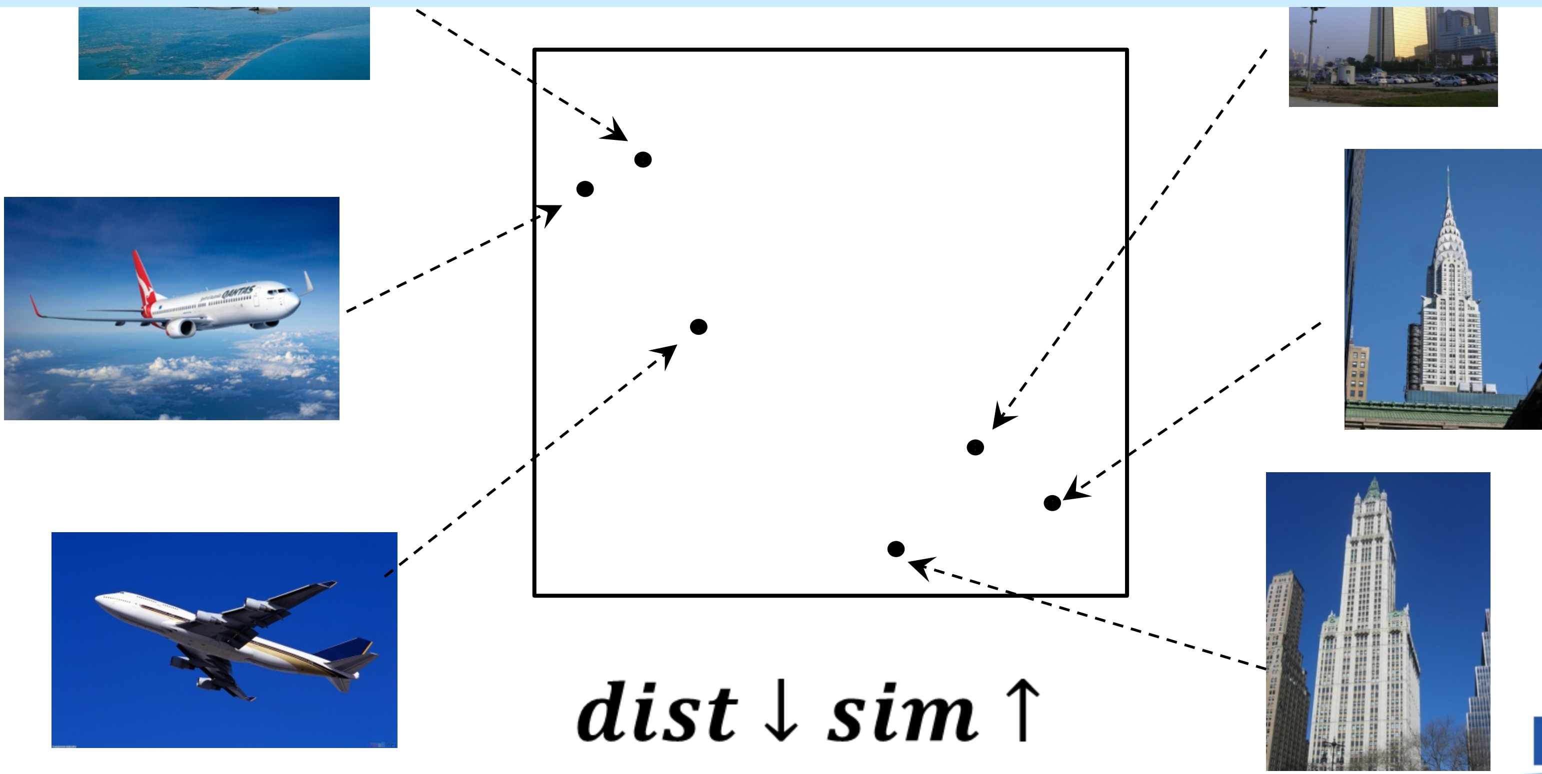


$$dist \downarrow \; sim \uparrow$$

Image Descriptor

High dimensional point

Nearest neighbor search (NNS) in high dimensional space



$$dist \downarrow sim \uparrow$$

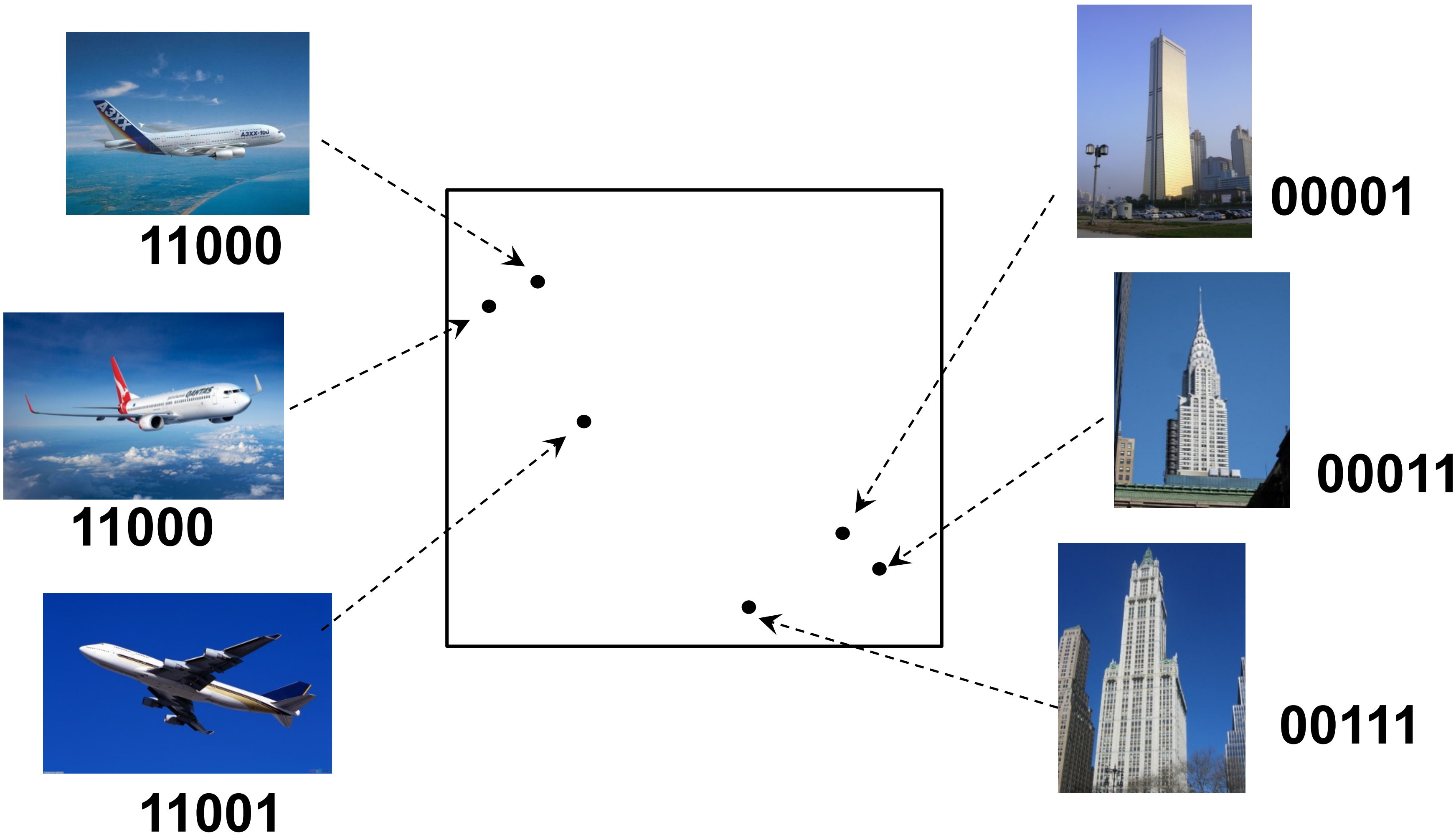
Challenge

	BoW	CNN
Dimensions	1000+	4000+
1 image	4 KB+	16 KB+
1B images	4 TB+	16 TB+

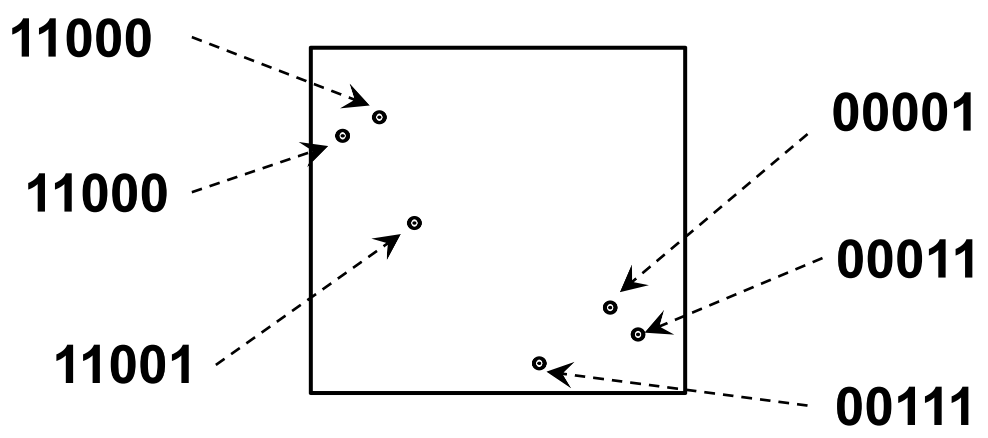
$$\frac{144\ GB\ memory}{1\ billion\ images} \approx \frac{128\ bits}{1\ image}$$

KAIST

Binary Code



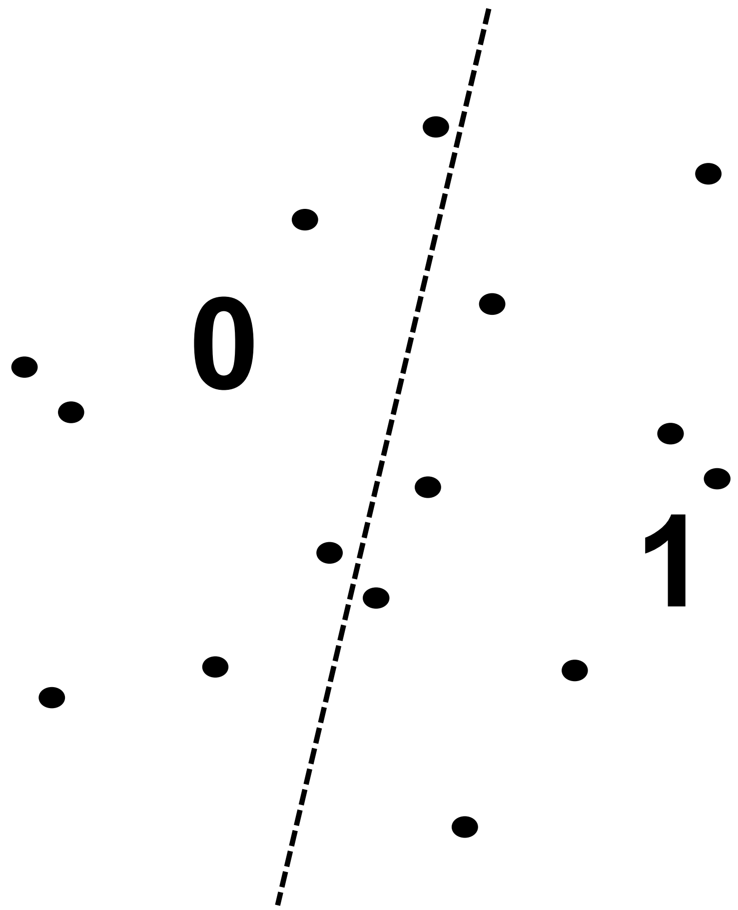
Binary Code



* Benefits

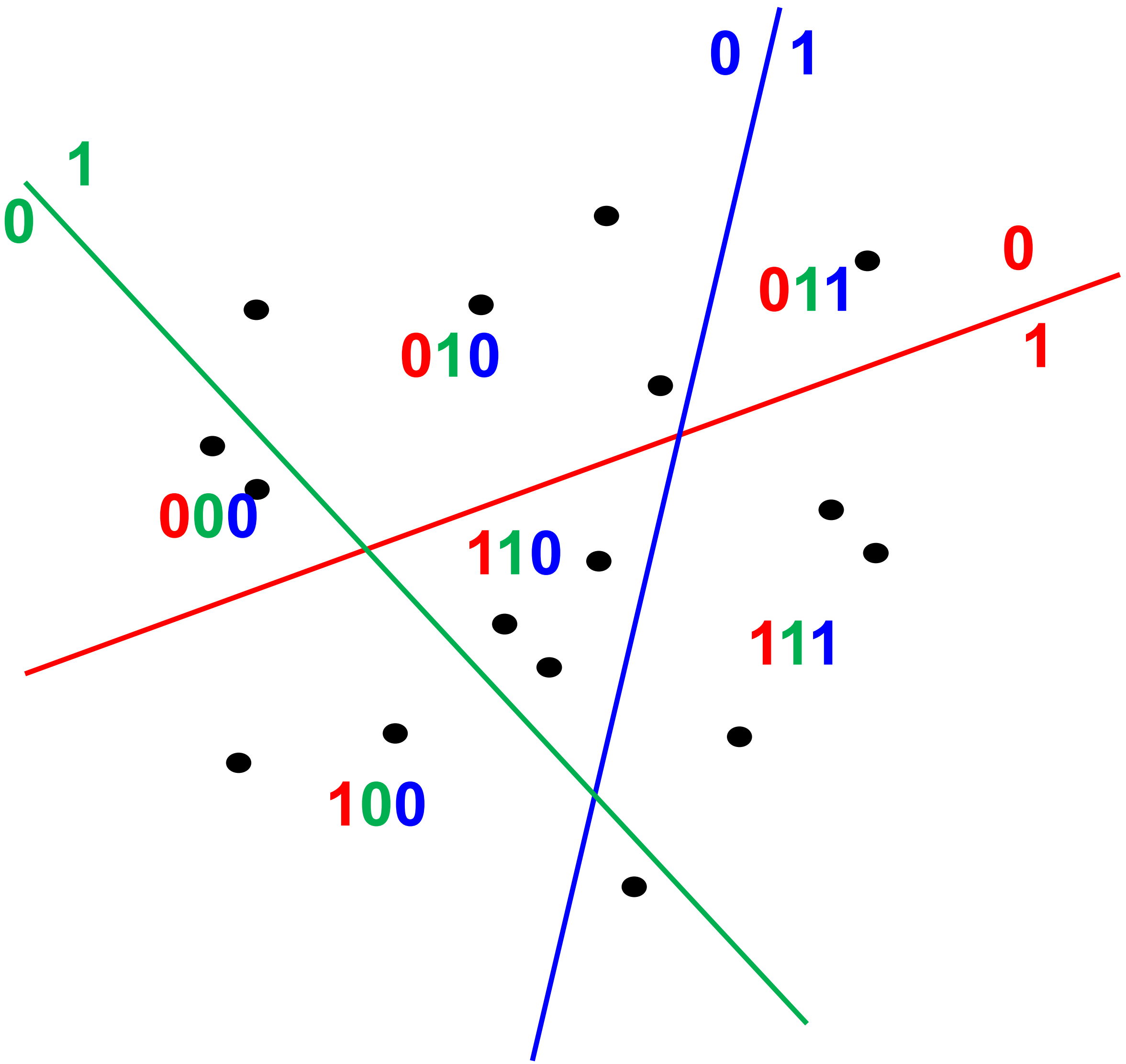
- Compression
- Very fast distance computation (Hamming Distance, XOR)

Hyper-Plane based Binary Coding



KAIST

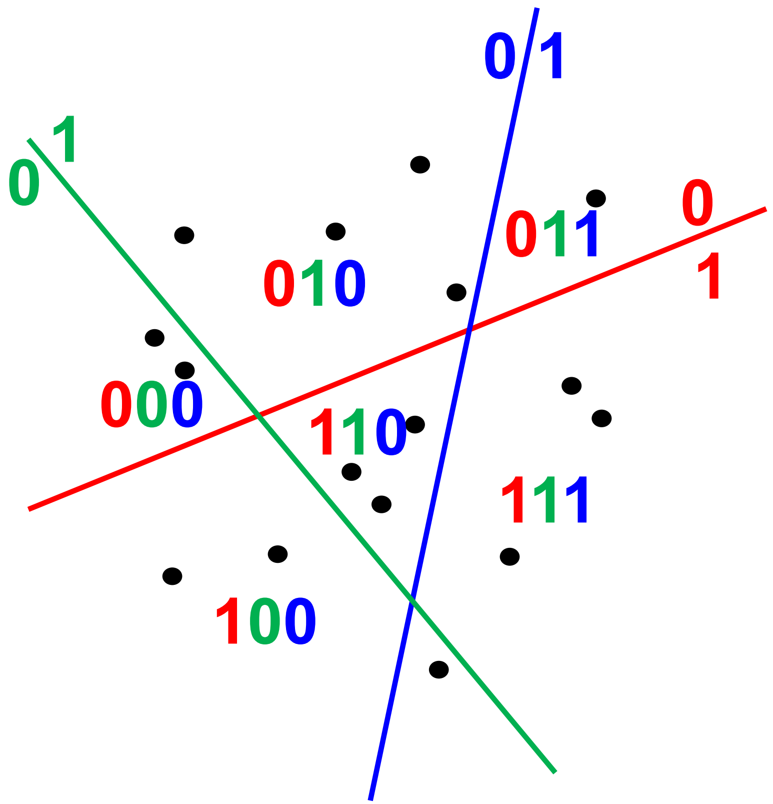
Hyper-Plane based Binary Coding



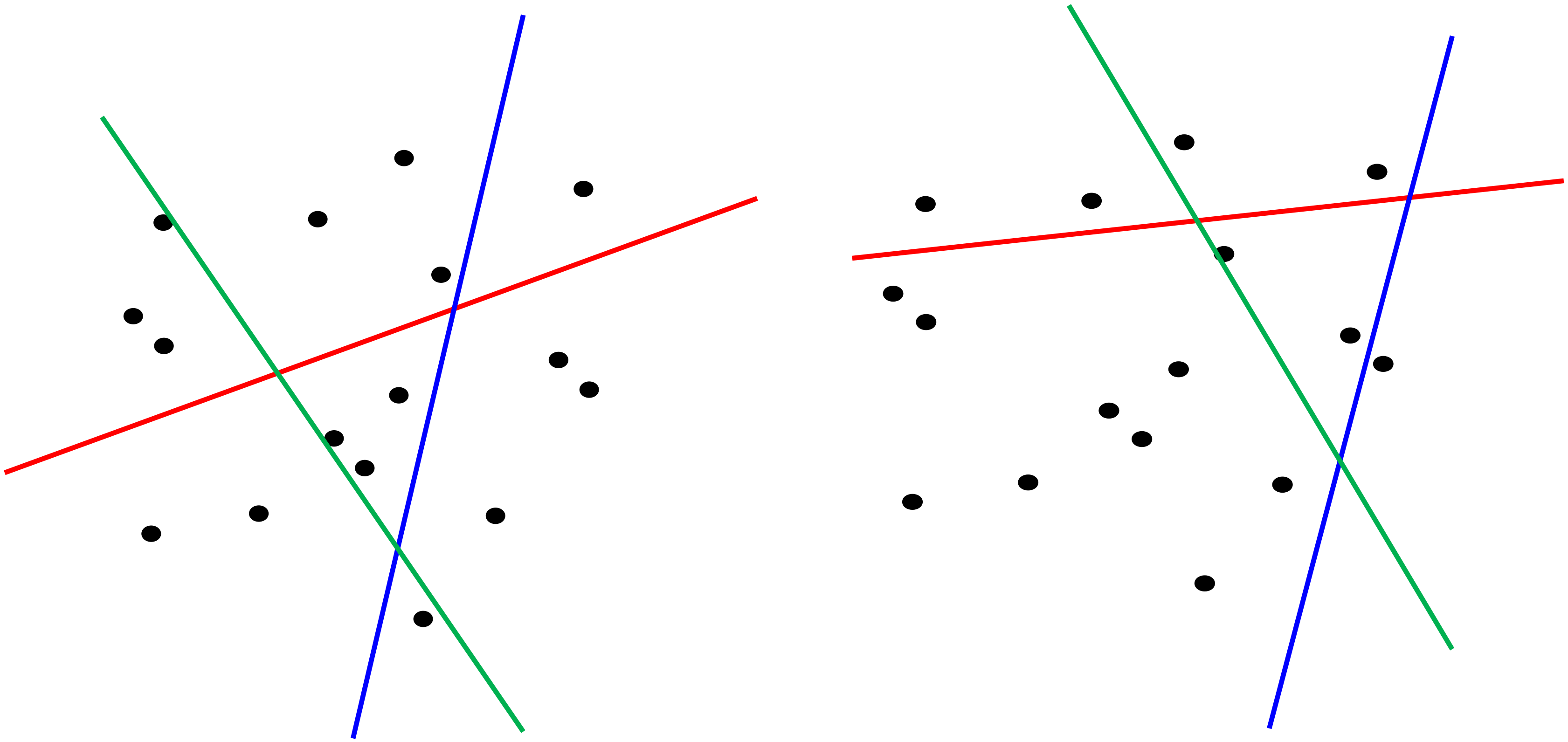
Distance between Two Points

- Measured by bit differences, known as Hamming distance
- Efficiently computed by XOR bit operations

$$d_{hd}(b_i, b_j) = |b_i \oplus b_j|$$



Good and Bad Hyper-Planes



Previous work focused on how to determine good hyper-planes

Components of Spherical Hashing

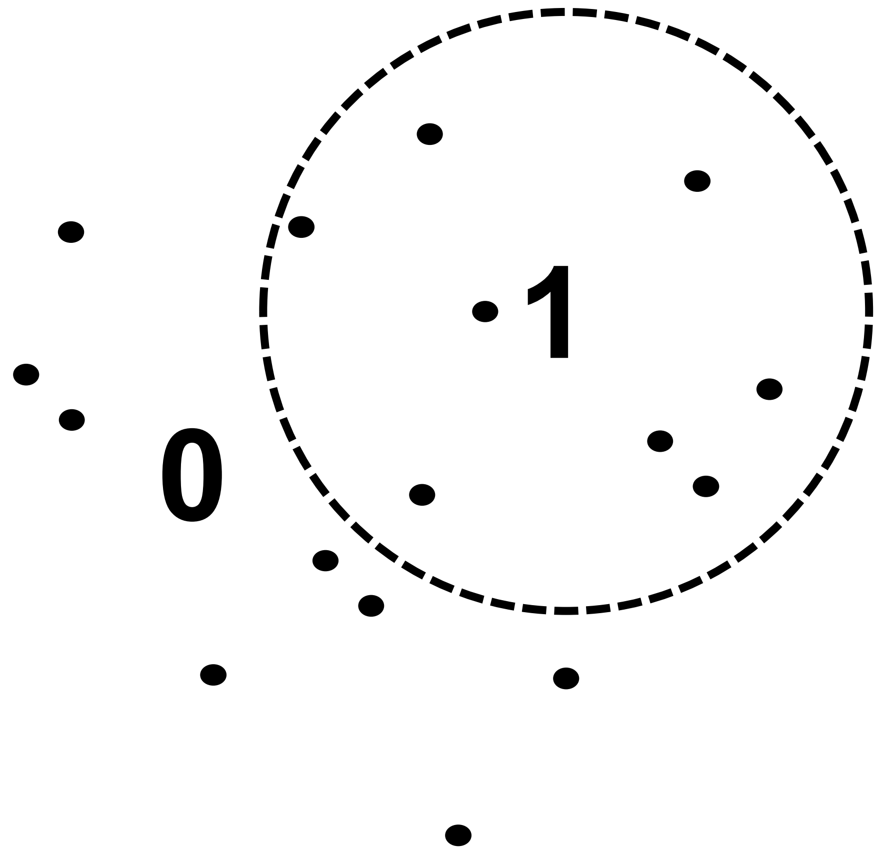
- Spherical hashing
- Hyper-sphere setting strategy
- Spherical Hamming distance

Components of Spherical Hashing

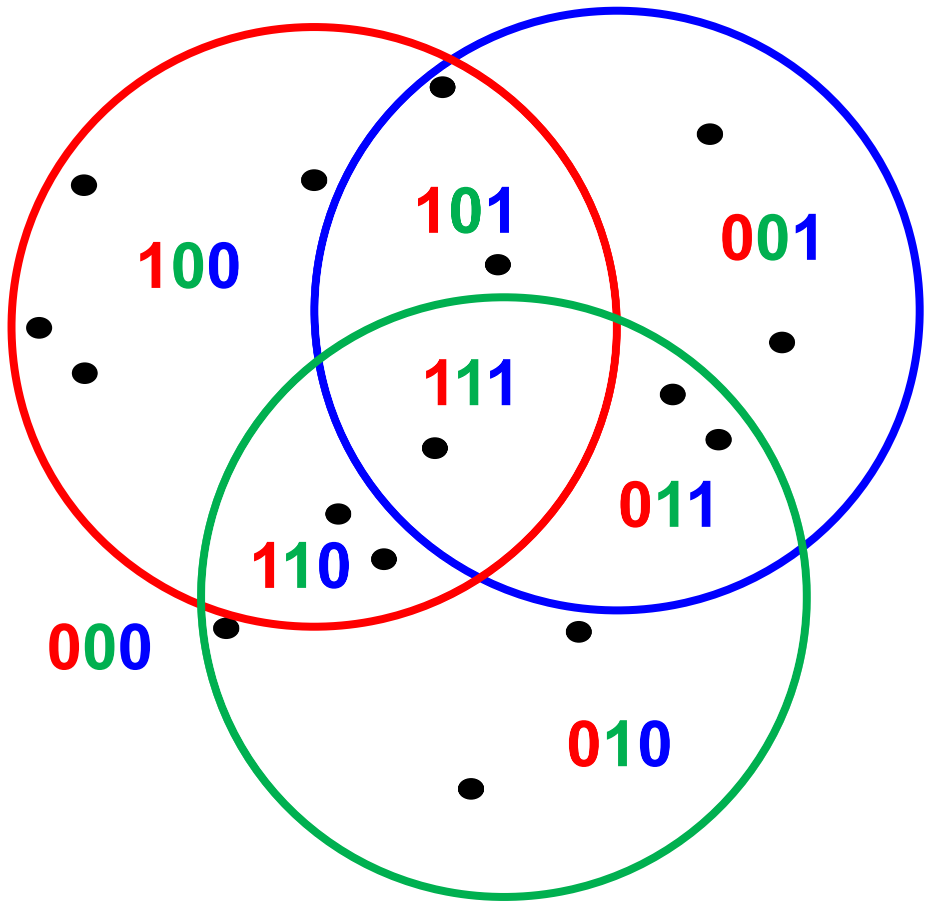
- **Spherical hashing**
- Hyper-sphere setting strategy
- Spherical Hamming distance

KAIST

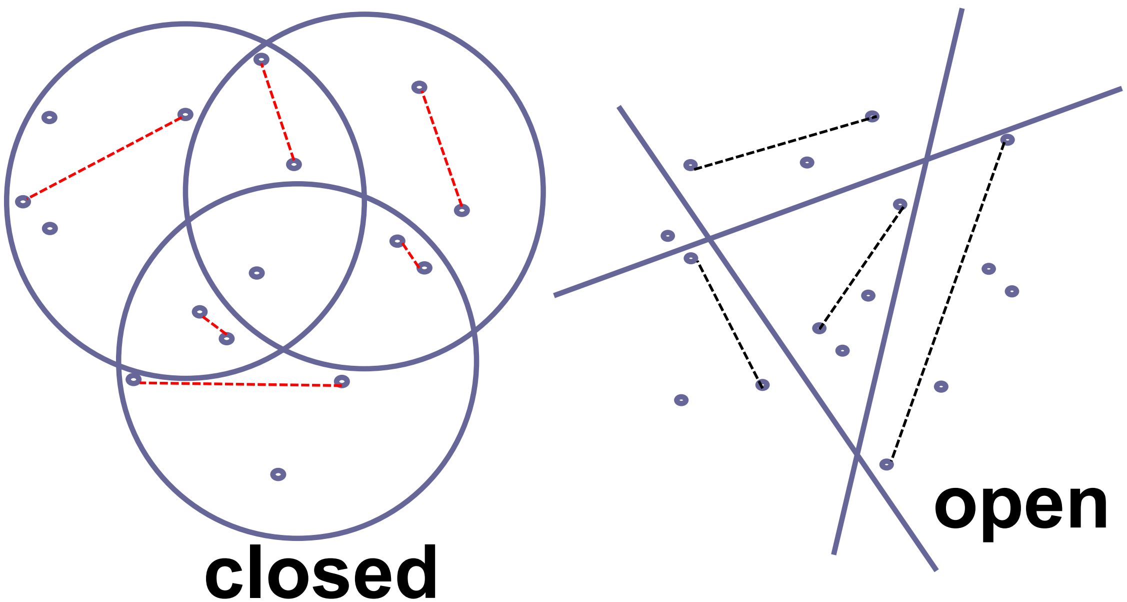
Spherical Hashing [Heo et al., CVPR 12]

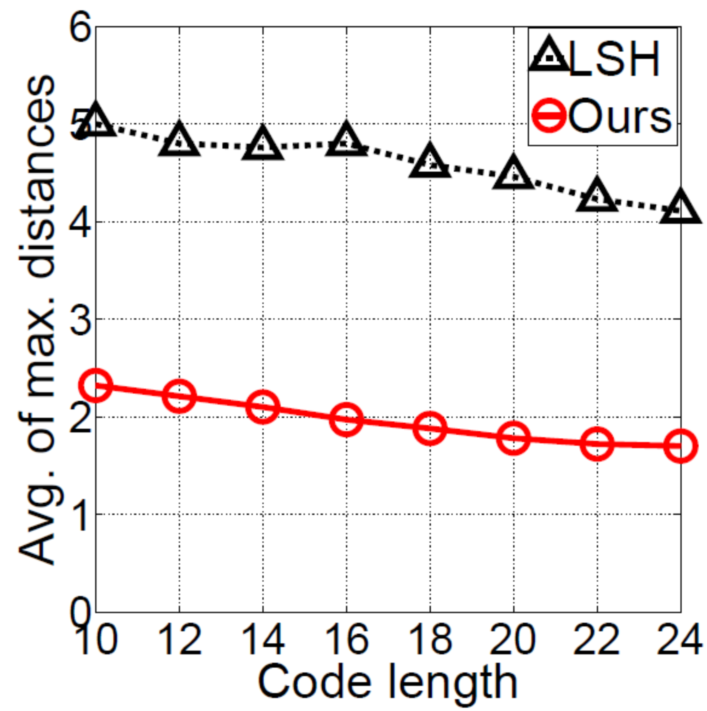


Spherical Hashing [Heo et al., CVPR 12]



Hyper-Sphere vs Hyper-Plane





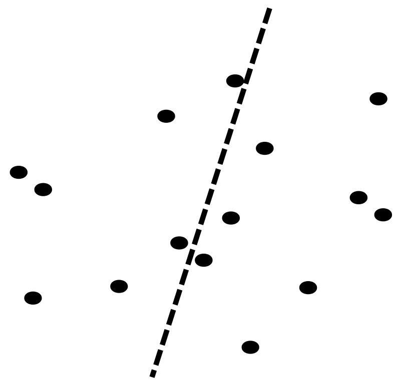
Average of maximum distances within a partition:
- Hyper-spheres gives tighter bound!

Components of Spherical Hashing

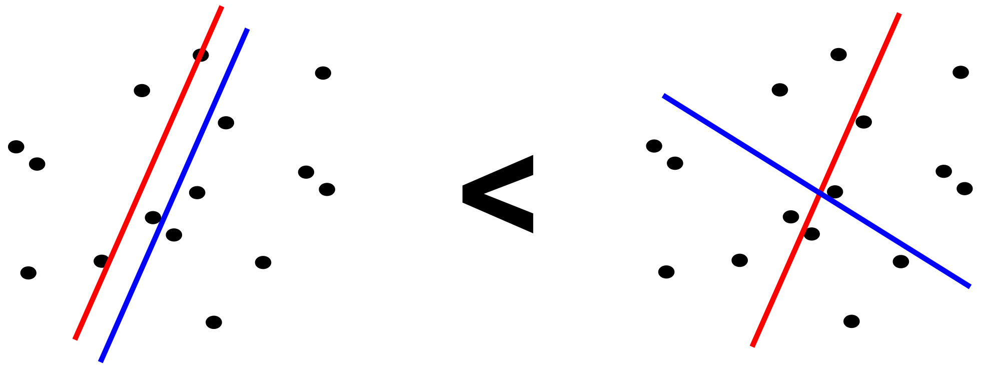
- Spherical hashing
- **Hyper-sphere setting strategy**
- Spherical Hamming distance

Good Binary Coding [Yeiss 2008, He 2011]

1. Balanced partitioning

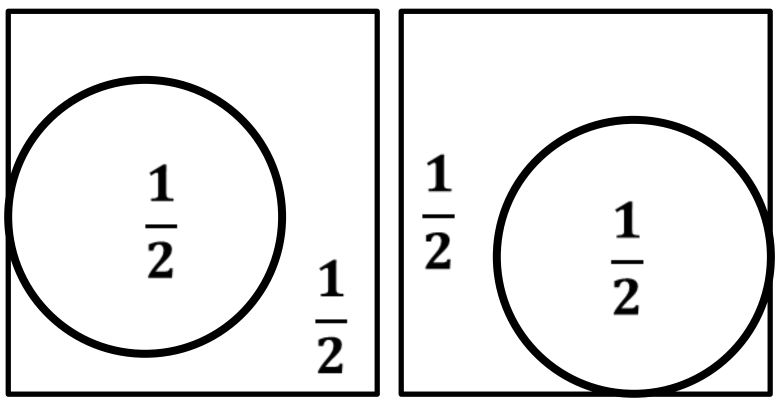


2. Independence

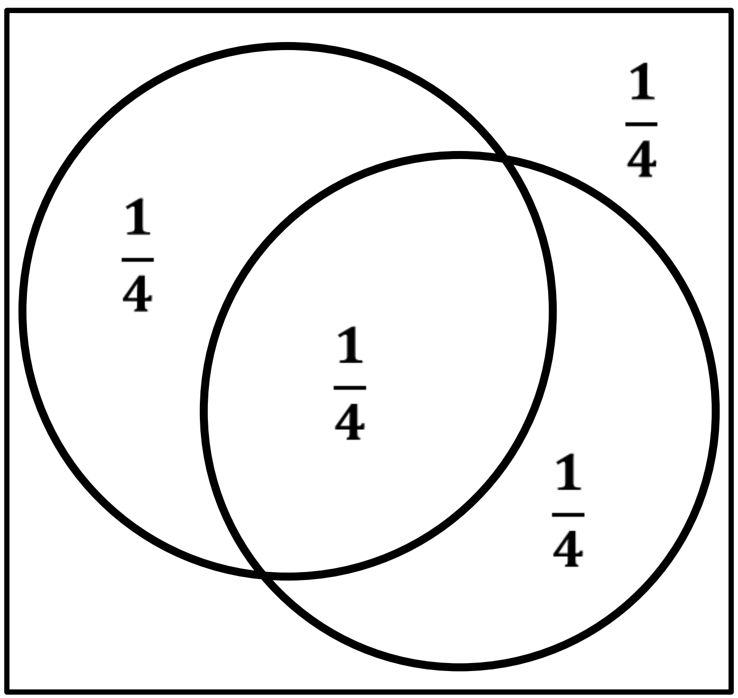


Intuition of Hyper-Sphere Setting

1. Balance

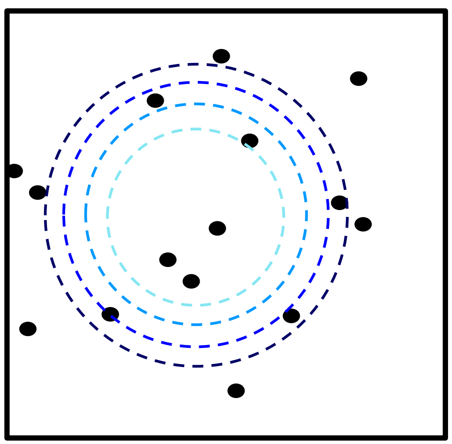


2. Independence

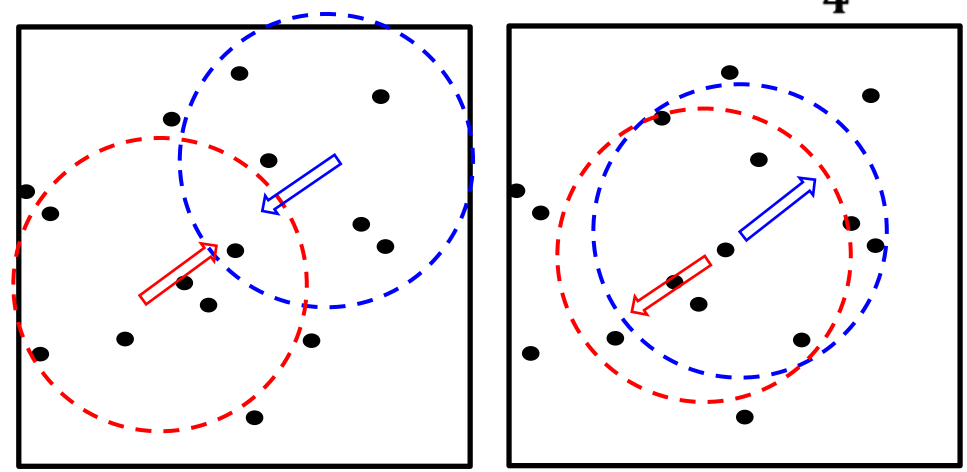


Hyper-Sphere Setting Process

1. Balance
- by controlling radius for $n(S) = \frac{N}{2}$



2. Independence
- by moving two hyper-spheres for $n(S_1 \cap S_2) = \frac{N}{4}$

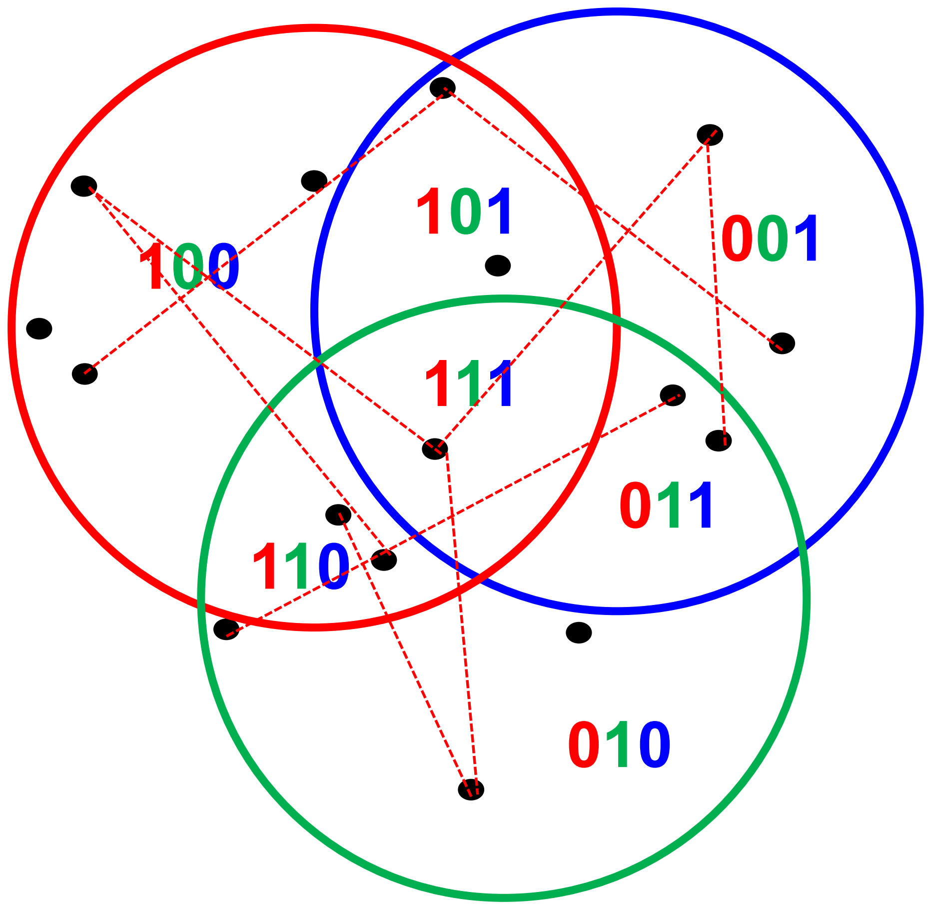


Iteratively repeat step 1, 2 until convergence.

Components of Spherical Hashing

- Spherical hashing
- Hyper-sphere setting strategy
- **Spherical Hamming distance**

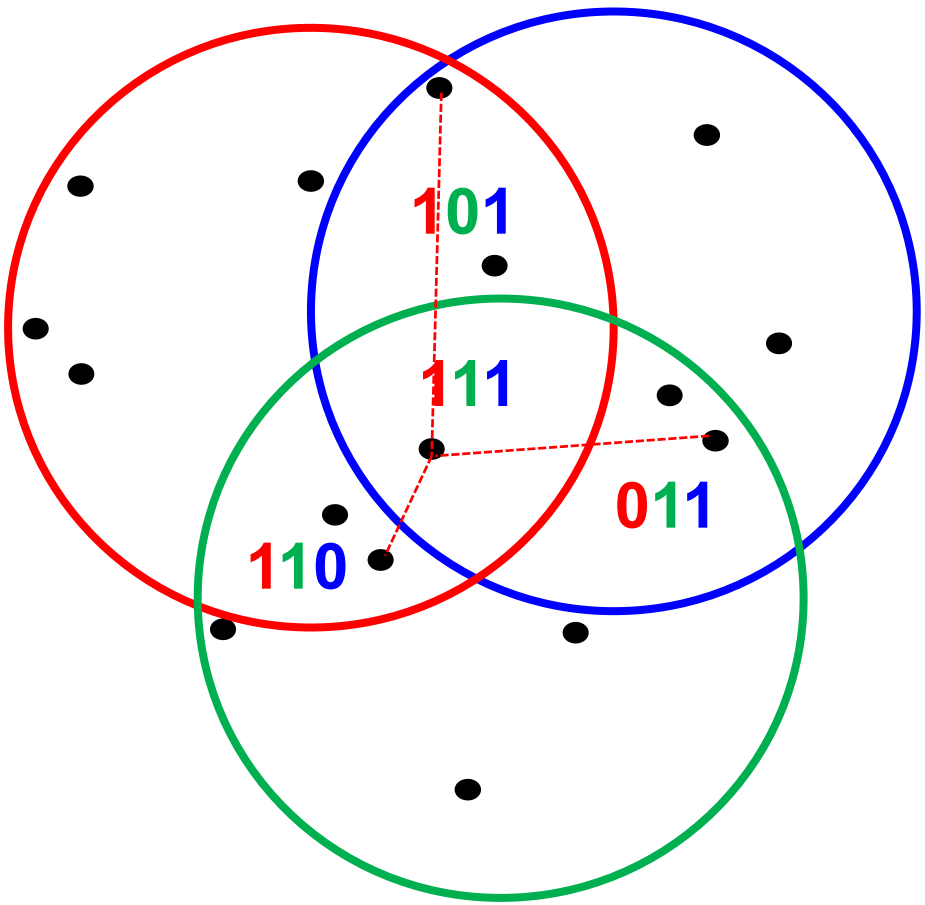
Max Distance and Common '1'



Common '1's

: **1**

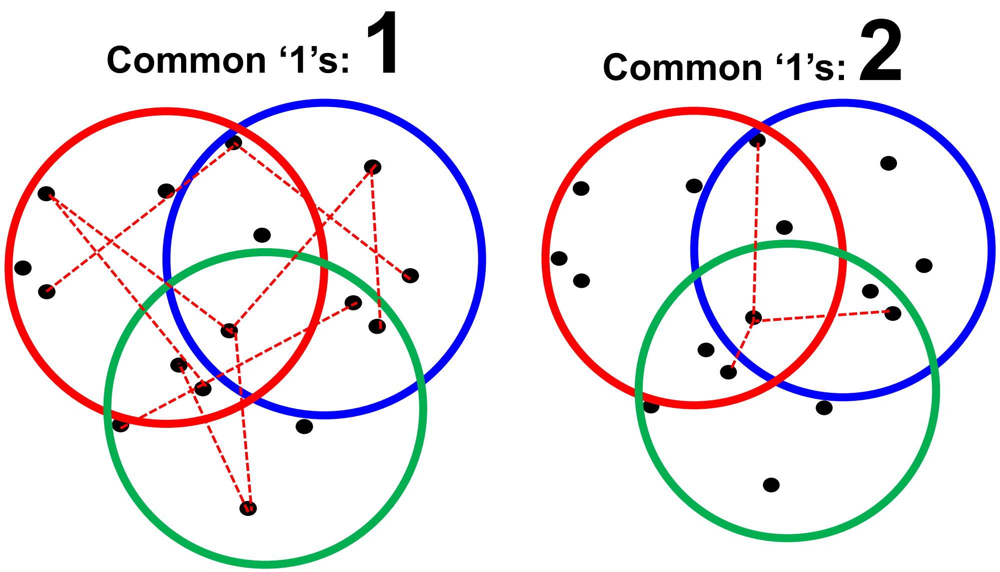
Max Distance and Common '1'

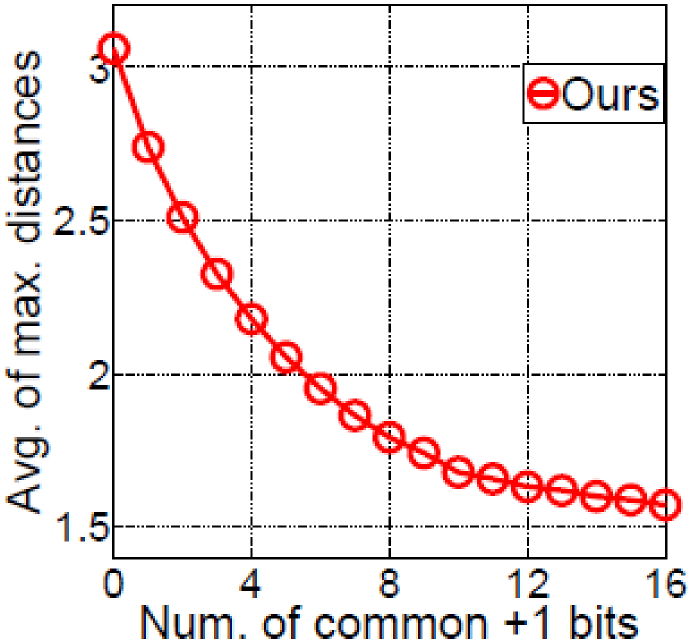


Common '1's

: **2**

Max Distance and Common ‘1’





Average of maximum distances between two partitions: decreases as number of common ‘1’

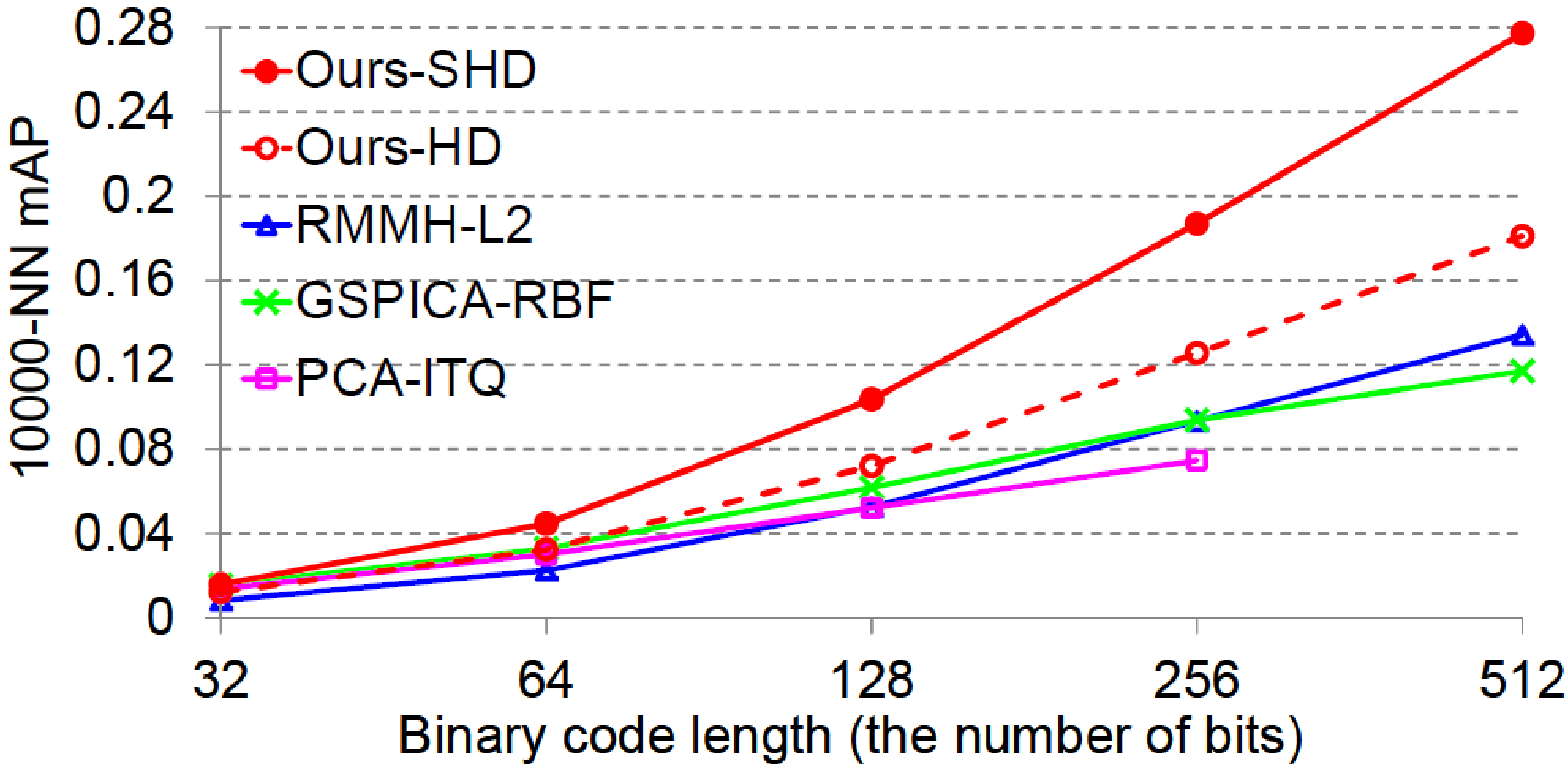
Spherical Hamming Distance (SHD)

$$d_{shd}(b_i, b_j) = \frac{|b_i \oplus b_j|}{|b_i \wedge b_j|}$$

SHD: Hamming Distance divided by the number of common '1's.

KAIST

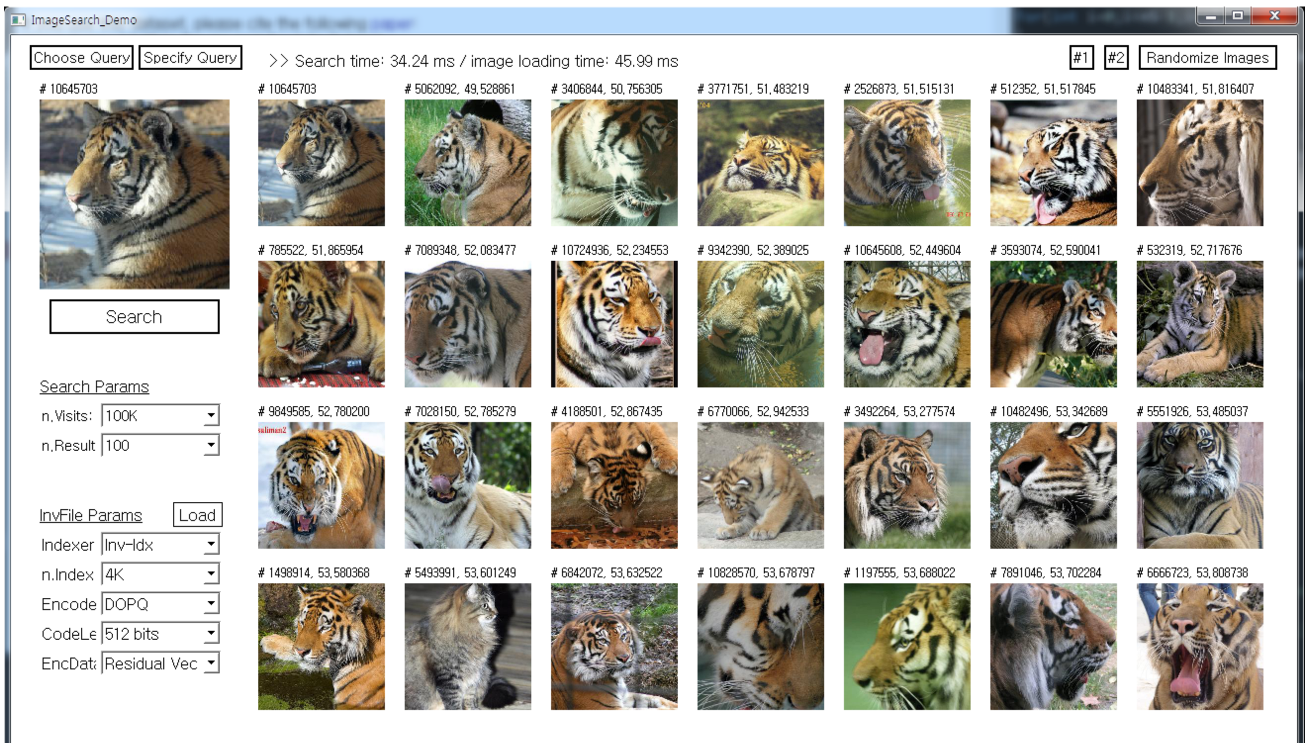
Results



384 dimensional 75 million GIST descriptors

Results of Image Retrieval

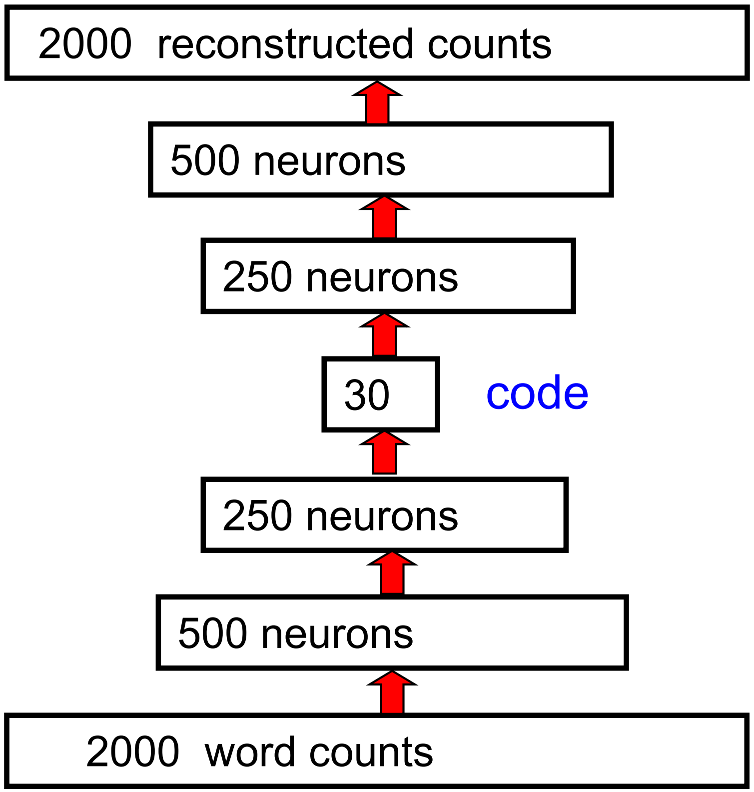
- **Collaborated with Adobe**
  - **11M images**
  - **Use deep neural nets for image representations**
  - **Spend only 35 ms for a single CPU thread**



KAIST

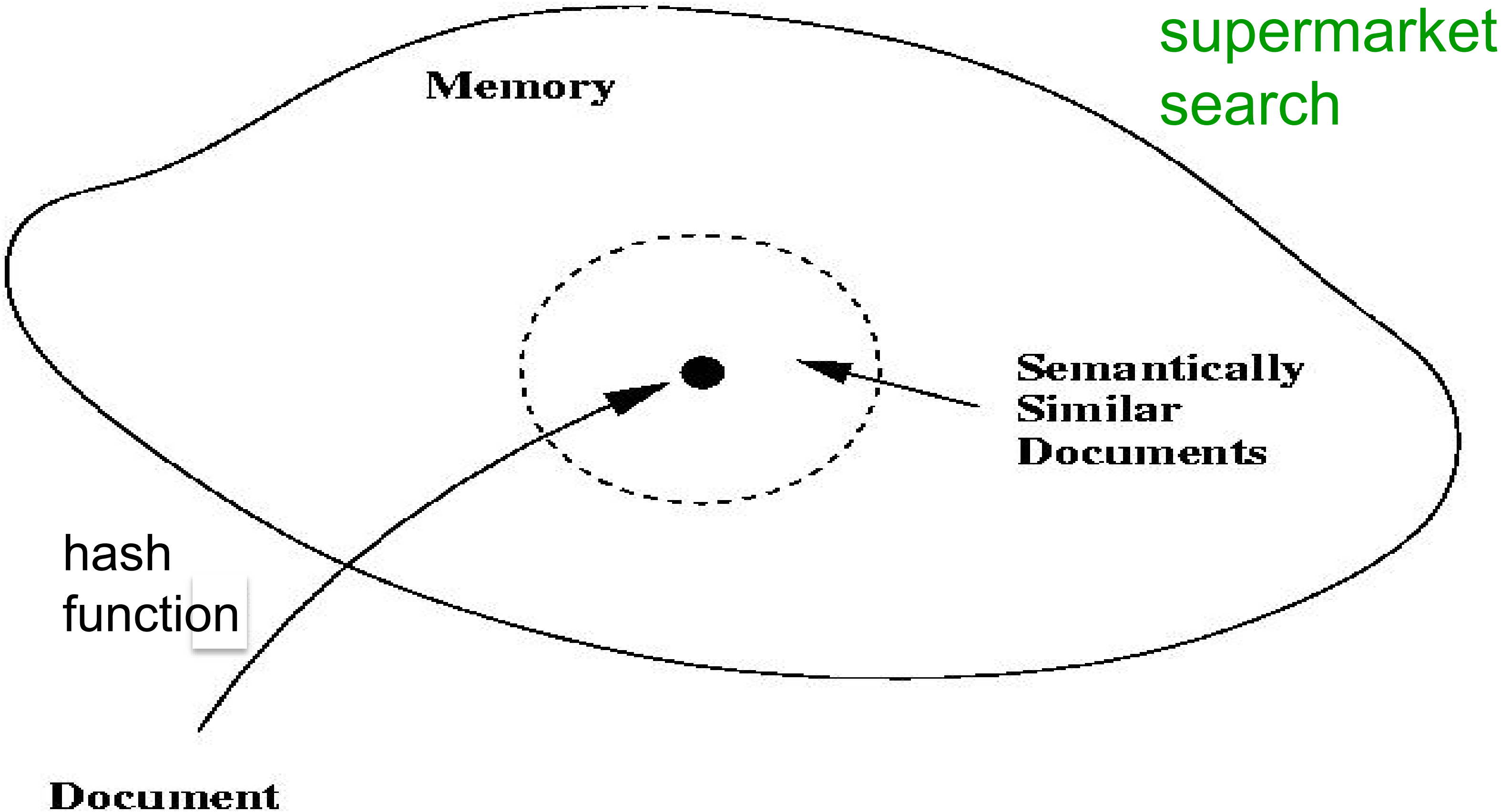
Semantic Hashing:
Finding binary codes for BoW model

- Train an auto-encoder using 30 logistic units for the code layer.
  - We simply threshold the activities of the 30 code units to get a binary code.



Ack. Hinton

Using a deep autoencoder as a hash-function for finding approximate matches



Binary codes for image retrieval

- Maybe we should extract a real-valued vector that has information about the content?
  - Matching real-valued vectors in a big database is slow and requires a lot of storage.
- Short binary codes are very easy to store and match.

A two-stage method

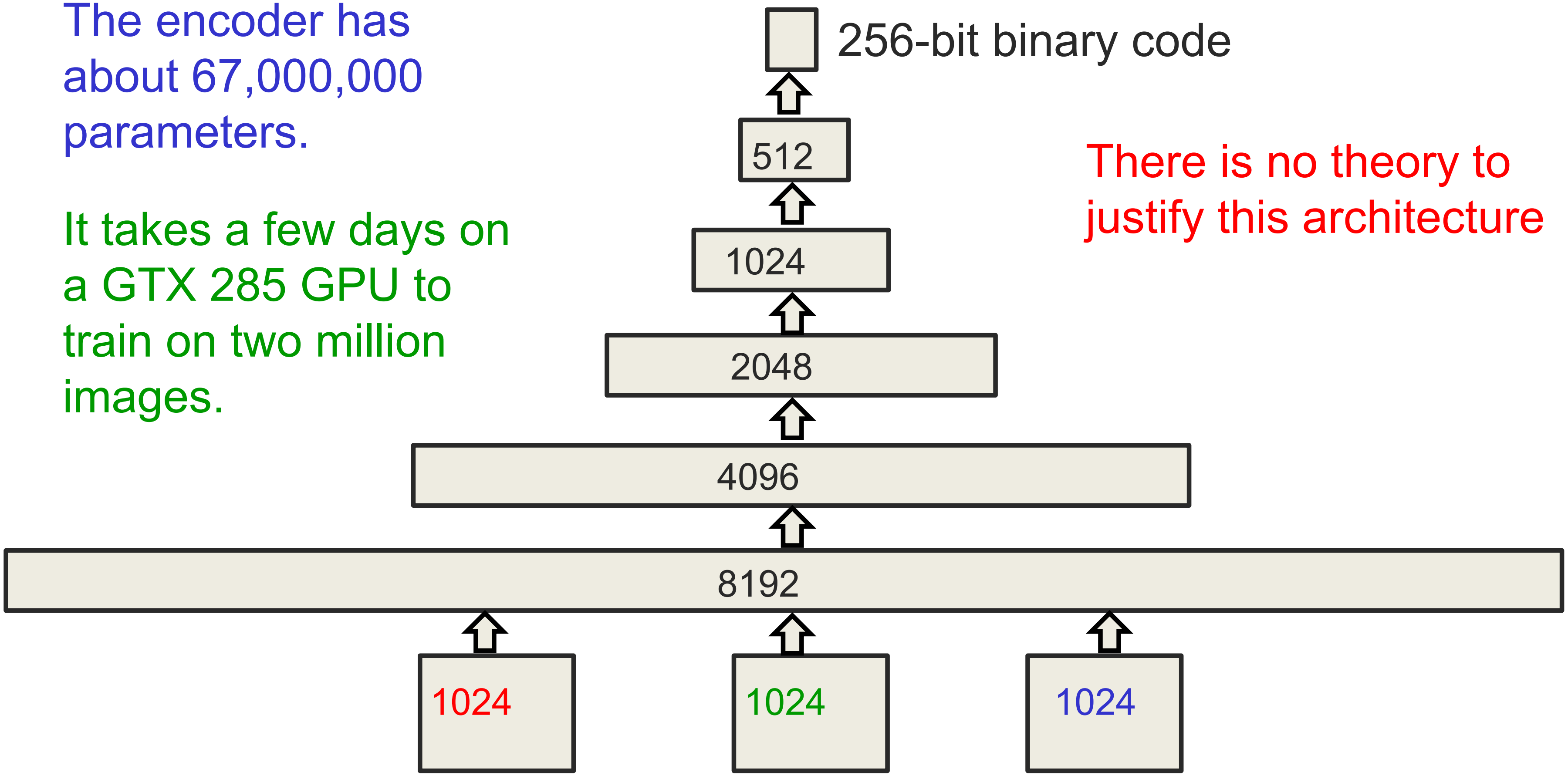
- First, use semantic hashing with 28-bit binary codes to get a long "shortlist" of promising images.
- Then use 256-bit binary codes to do a serial search for good matches.
  - This only requires a few words of storage per image and the serial search can be done using fast bit-operations.
- But how good are the 256-bit binary codes?
  - Do they find images that we think are similar?

Krizhevsky's deep autoencoder

The encoder has about 67,000,000 parameters.

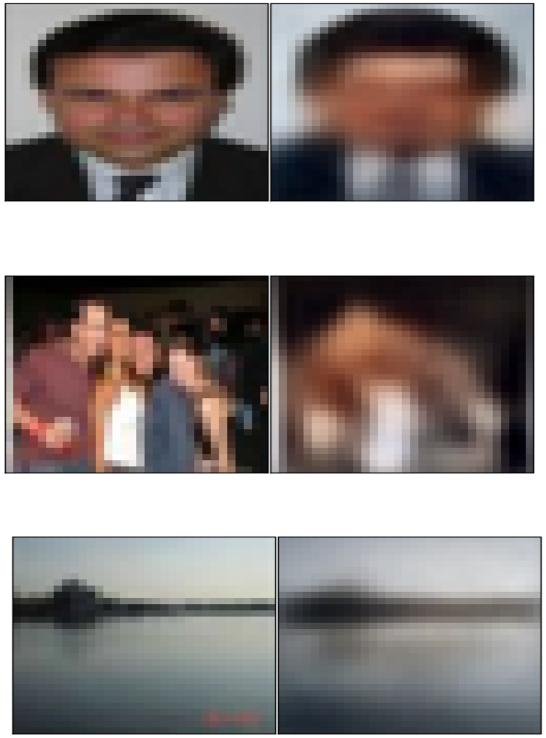
It takes a few days on a GTX 285 GPU to train on two million images.

There is no theory to justify this architecture



Start w/ 32 by 32 image patch

Reconstructions of 32x32 color images from 256-bit codes



retrieved using 256 bit codes

retrieved using Euclidean distance in pixel intensity space

retrieved using 256 bit codes

retrieved using Euclidean distance in pixel intensity space

Class Objectives were:

- **Understand the basic hashing techniques based on hyperplanes**
  - **Unsupervised approach**
- **Semantic hashing**
- **Codes are available**

http://sglab.kaist.ac.kr/software.htm

Homework for Every Class

- **Go over the next lecture slides**
- **Come up with one question on what we have discussed today**
  - **Write questions three times**
- **Go over recent papers on image search, and submit their summary before Mon. class**

Next Time…

- **Person Re-Identification, Re-ID**