Quadra-Embedding:
Binary Code Embedding with Low Quantization Error

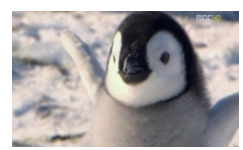
Youngwoon Lee*, Jae-Pil Heo, Sung-Eui Yoon
KAIST

ACCV 2012

Large-scale Image Retrieval

Query

Database



Large-scale Image Retrieval

Query

➡ [1,0,2,…]

High-dim. descriptor
- GIST (> 300 dim)
- BoW (> 1000 dim)

Database

➡ [0,4,3,…]

➡ [1,0,2,…]

➡ [2,1,2,…]

➡ [0,2,1,…]

BRING YOUR CUP ➡ [1,0,0,…]

Large-scale Image Retrieval

Query

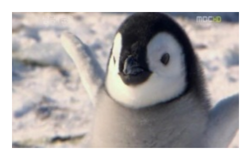
➡ [1,0,2,…]

Database

➡ [0,4,3,…]

➡ [1,0,2,…]

➡ [2,1,2,…]



➡ [0,2,1,…]

➡ [1,0,0,…]

Challenges

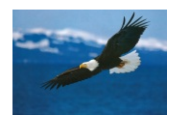
- Slow exhaustive search
- Huge memory requirement

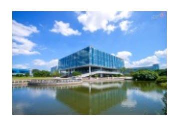
Compact Binary Code [Torralba et al. 2008]

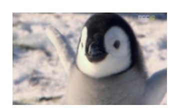
- Embed image descriptor
  to compact similarity-preserving binary codes
- Both time and storage efficient

Query

Database

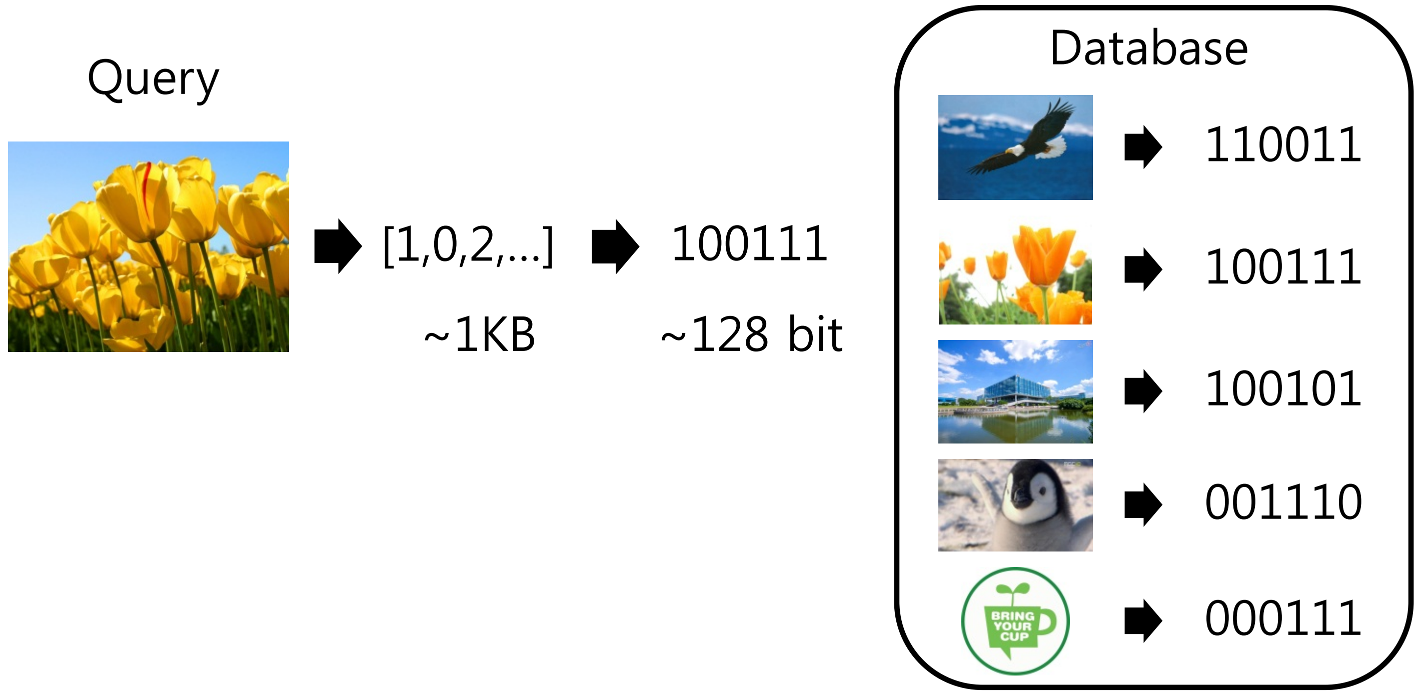






Compact Binary Code [Torralba et al. 2008]

- Embed image descriptor
  to compact similarity-preserving binary codes
- Both time and memory efficient



Compact Binary Code

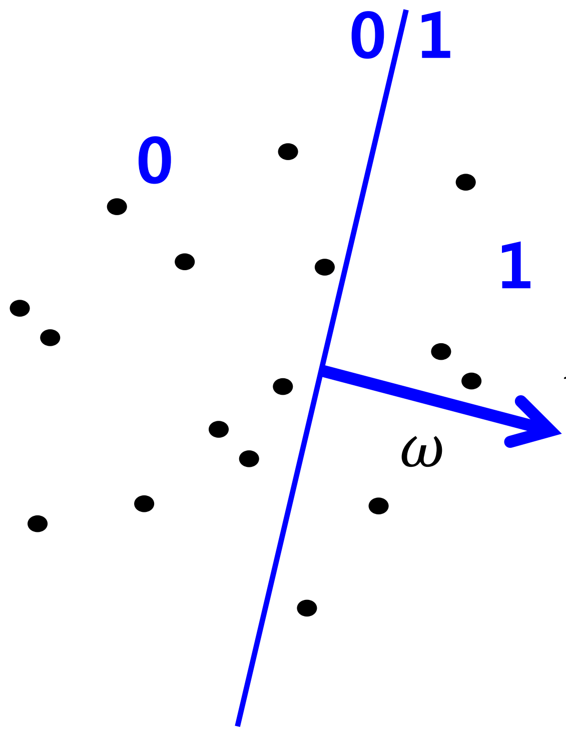
- How to encode an image to a binary code?

- Data-independent methods
  - Locality-Sensitive Hashing (LSH) [Datar et al. 2004]
  - Shift-invariant Kernel LSH (SKLSH) [Raginsky et al. 2009]

- Data-dependent methods
  - Spectral Hashing (SH) [Weiss et al. 2008]
  - Iterative Quantization (ITQ) [Gong et al. 2011]

Locality-Sensitive Hashing

- Randomly generated hyperplanes
- Data-independent method

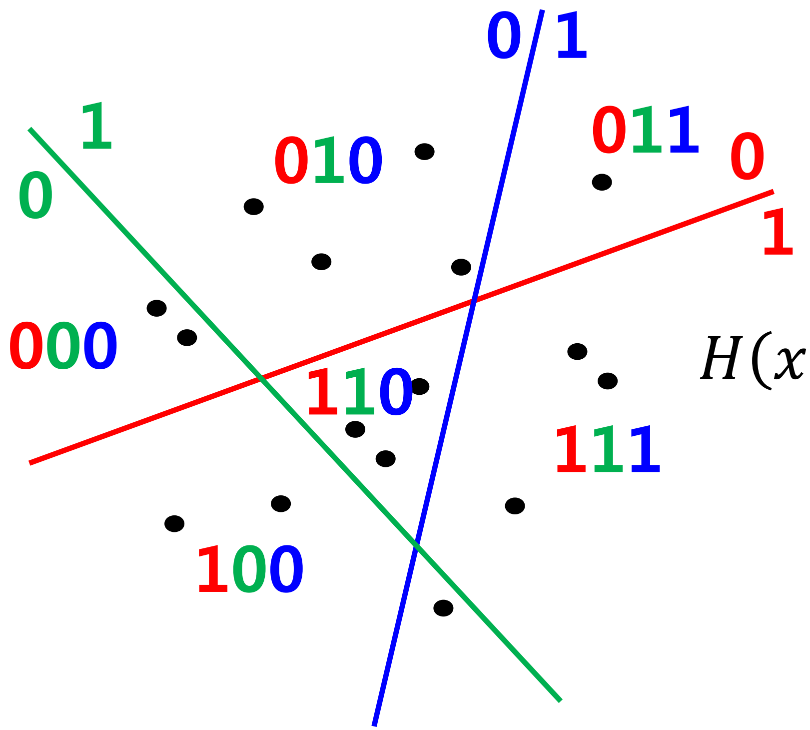


$H(x) = sgn(\underline{\omega^T x + b})$

Projection value

Locality-Sensitive Hashing

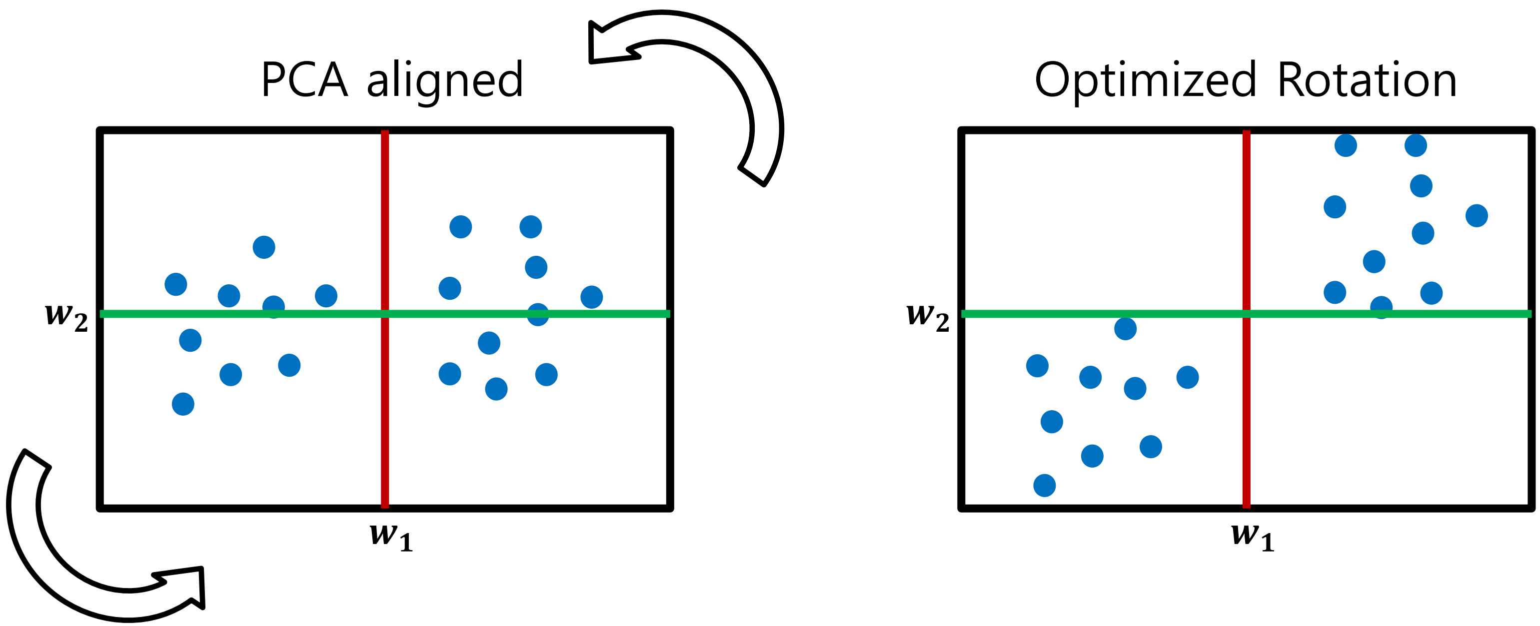
- Randomly generated hyperplanes
- Data-independent method



$H(x) = sgn(\omega^T x + b)$

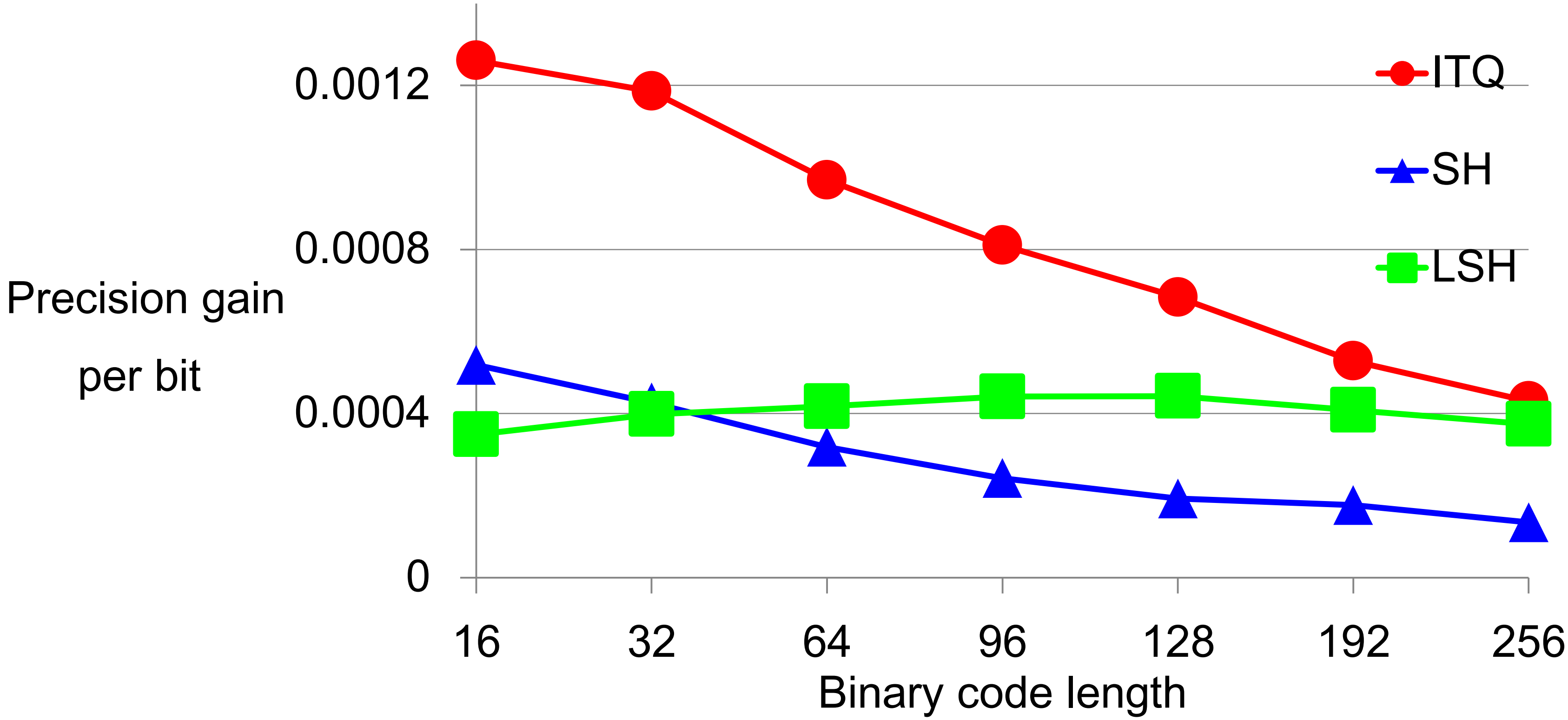
Iterative Quantization

- Rotate the PCA-projected data to minimize quantization error
- Data-dependent method



Problem

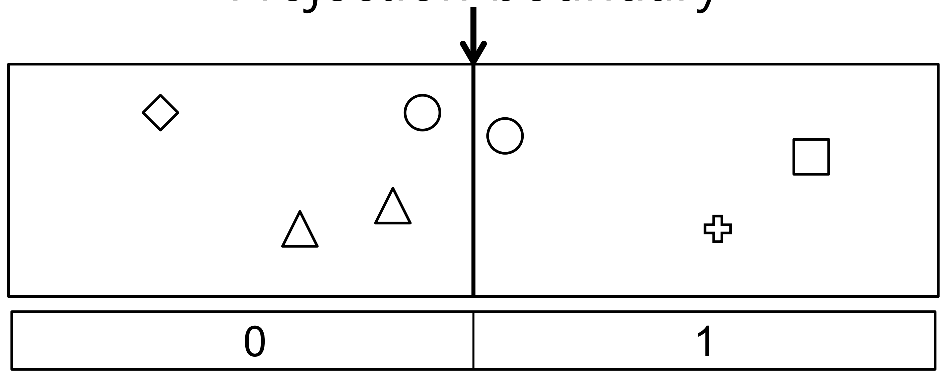
- Diminishing efficiency of bits



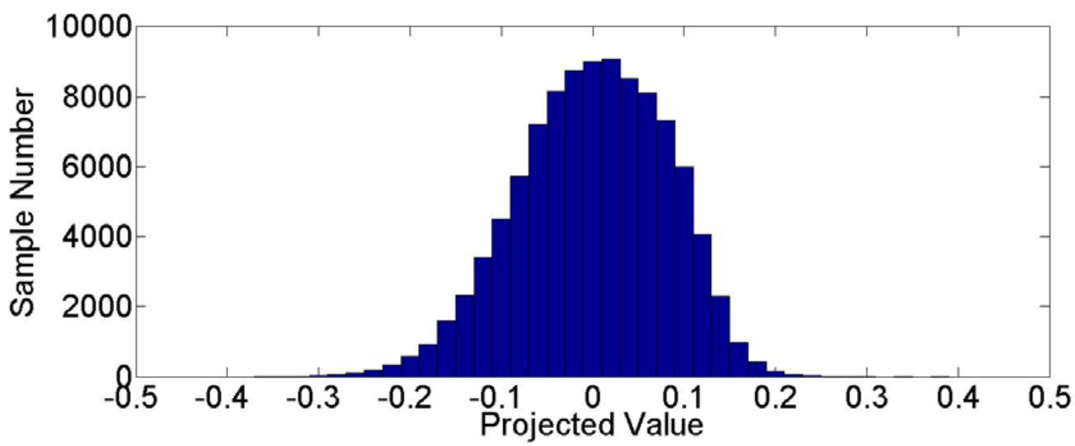
Problem

- High quantization error near boundary

Projection boundary



Bit value



Problems

- Diminishing efficiency
- High quantization error near boundary

Problems of Single-Bit Quantization (SBQ)

- One bit value per one projection

Solutions

- Diminishing efficiency
- High quantization error near boundary

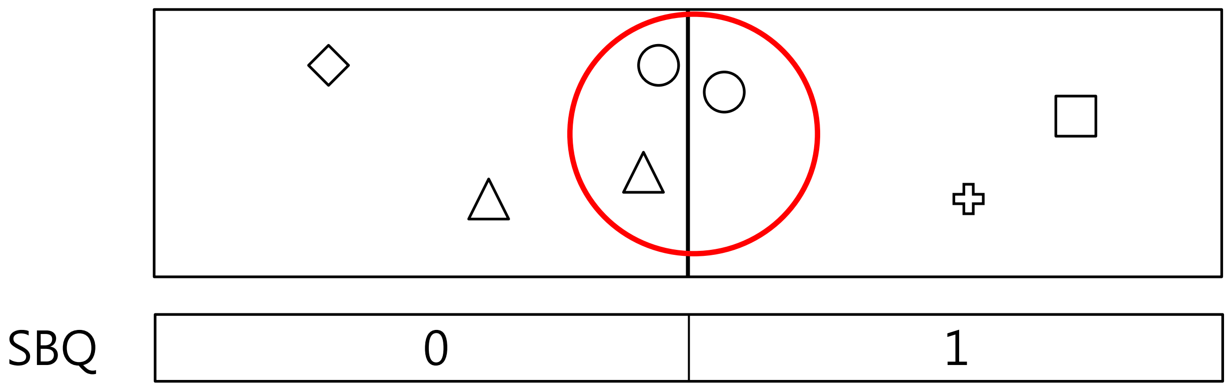
Depart from Single-Bit Quantization (SBQ)

⇩

- Assign 2 bits for each projection
  - Use only half projections
    to get a good set of projections
  - Utilize 1 bit for reducing quantization error

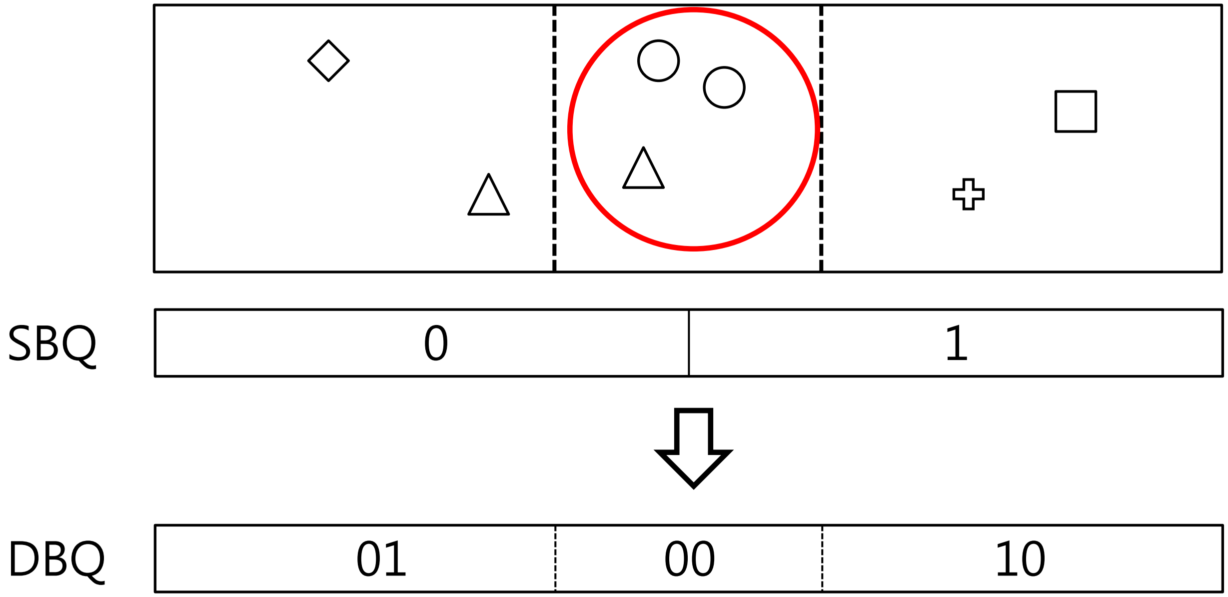
Double-Bit Quantization [Kong and Li. 2012]

- Assign same code along a boundary



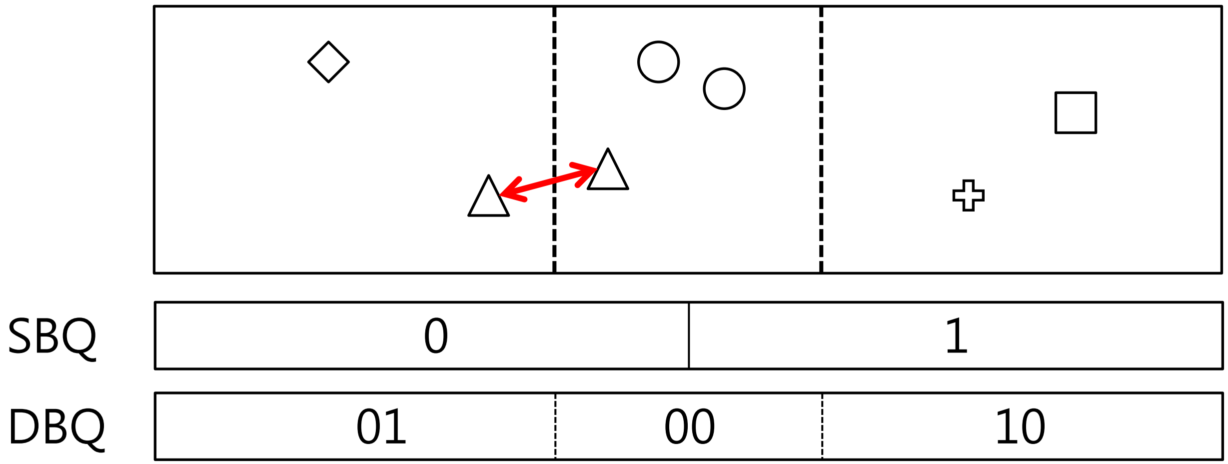
Double-Bit Quantization [Kong and Li. 2012]

- Assign same code along a boundary



Limitations

- Double-Bit Quantization (DBQ)
  - Additional quantization error near new boundaries
  - Cannot fully utilize 2 bits
    - Only encodes 3 regions



Our Approach

Quadra Embedding

Fully utilize 2 bits with 4 regions

Threshold optimization

Determine boundary suitable to Quadra Embedding and distance

Distance metric

Provide low quantization error near boundary

Our Approach

Quadra Embedding

Fully utilize 2 bits with 4 regions

Threshold optimization

Determine boundary suitable to Quadra Embedding and distance

Distance metric

Provide low quantization error near boundary

Our Approach

Quadra Embedding

Fully utilize 2 bits with 4 regions

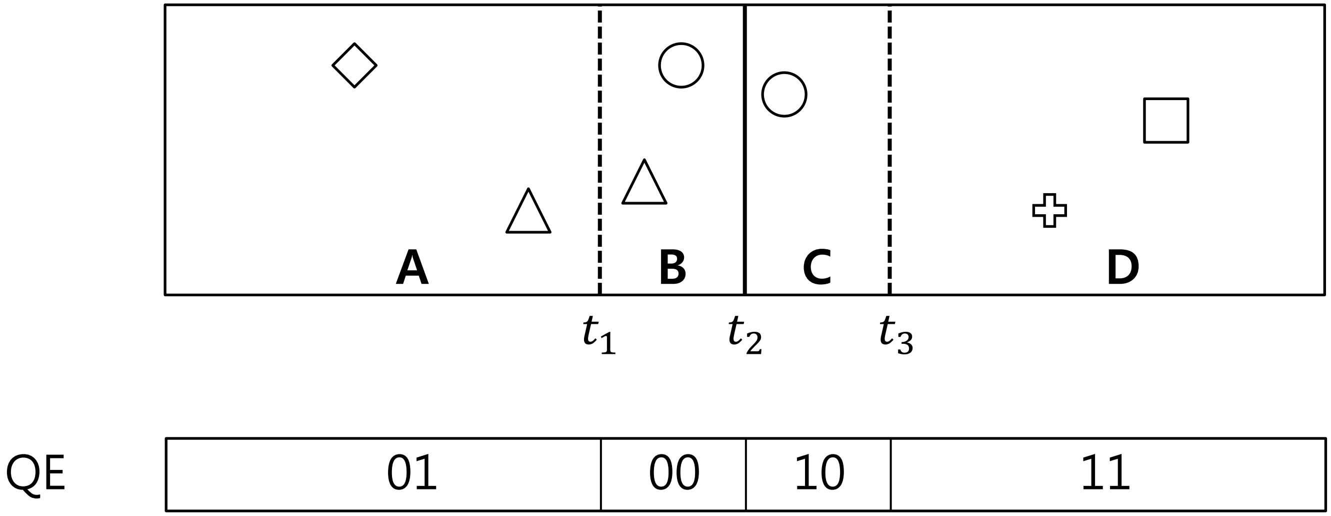
Threshold optimization

Determine boundary suitable to Quadra Embedding and distance

Distance metric

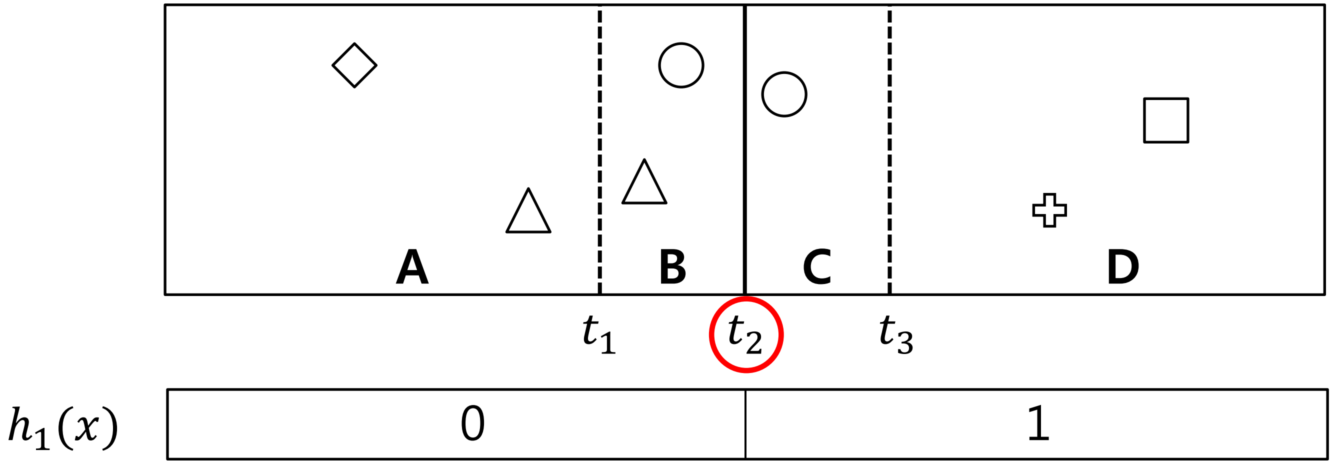
Provide low quantization error near boundary

Quadra Embedding (QE)



- Partition space into four regions
- Determine thresholds through optimization

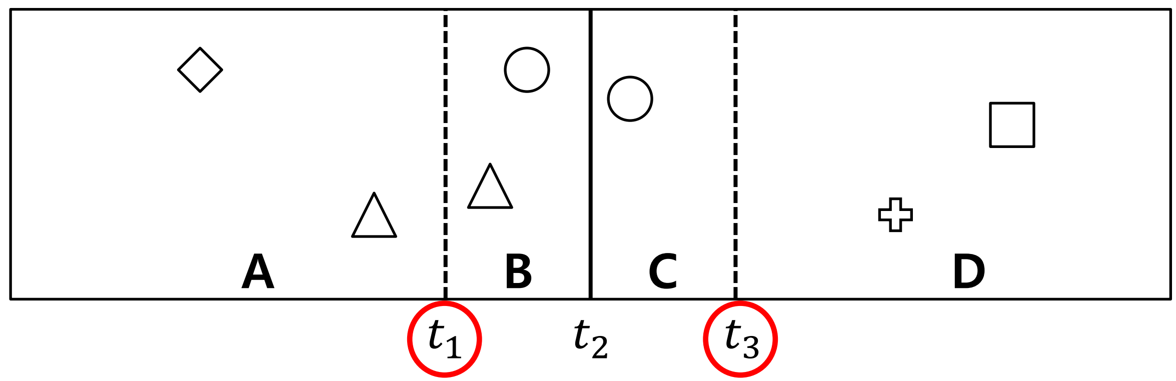
Quadra Embedding



Binary code $X = (H_1(x), H_2(x))$

$$= (h_1^1(x), \cdots, h_1^{m/2}(x), h_2^1(x), \cdots, h_2^{m/2}(x))$$

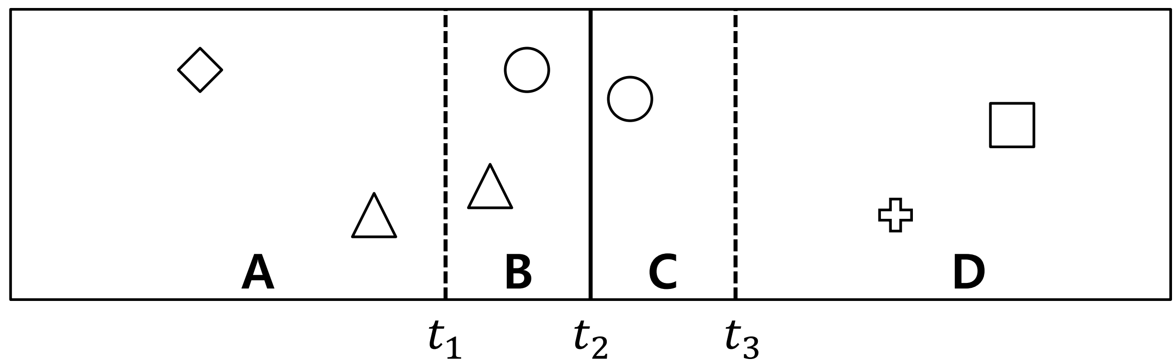
Quadra Embedding



$h_2(x)$	1	0	0	1

Binary code
$$X = (H_1(x), H_2(x))$$
$$= (h_1^1(x), \cdots, h_1^{m/2}(x), h_2^1(x), \cdots, h_2^{m/2}(x))$$

Quadra Embedding



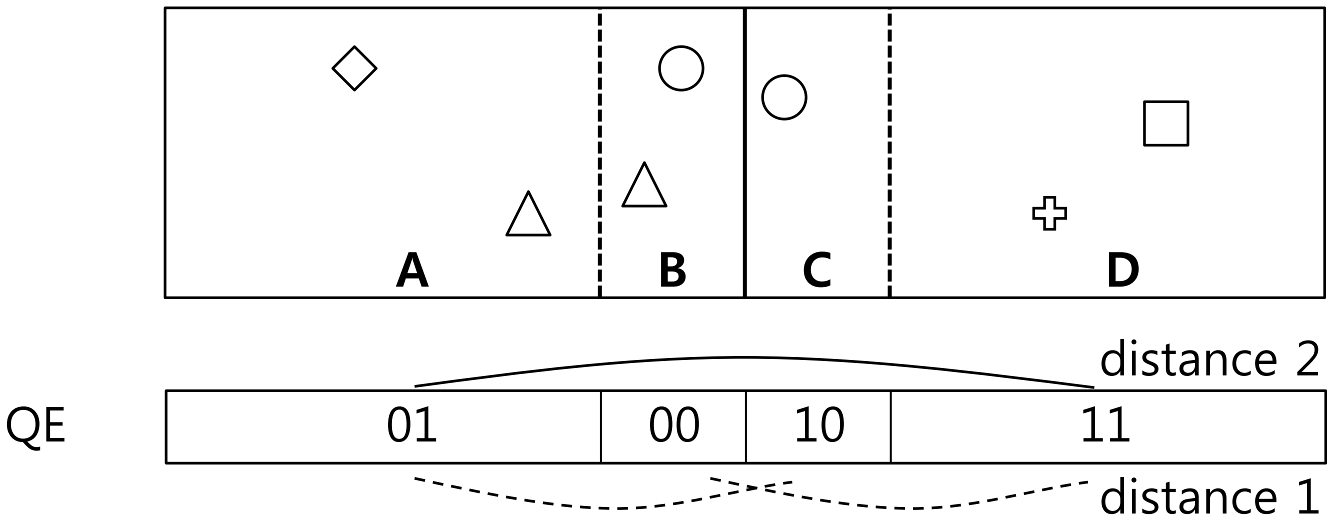
$h_1(x)$	0		1	
$h_2(x)$	1	0	0	1
QE	01	00	10	11

Binary code
$$X = (H_1(x), H_2(x))$$
$$= (h_1^1(x), \cdots, h_1^{m/2}(x), h_2^1(x), \cdots, h_2^{m/2}(x))$$

KAIST

Quadra Embedding Distance

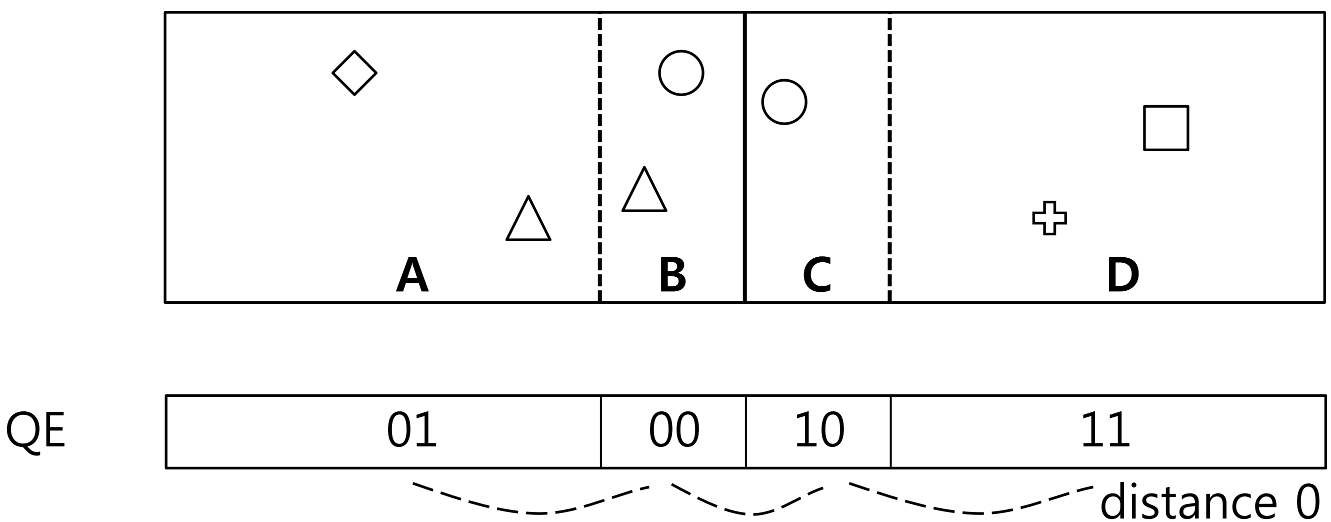
$$d(X,Y) = \underline{\#\ of\ regions\ between\ X\ and\ Y}$$
$$= 2|(X_1 \oplus Y_1) \wedge (X_2 \wedge Y_2)| + |(X_1 \oplus Y_1) \wedge (X_2 \oplus Y_2)|$$



- Give distance 1 and 2 according to the number of regions between two binary codes

Quadra Embedding Distance

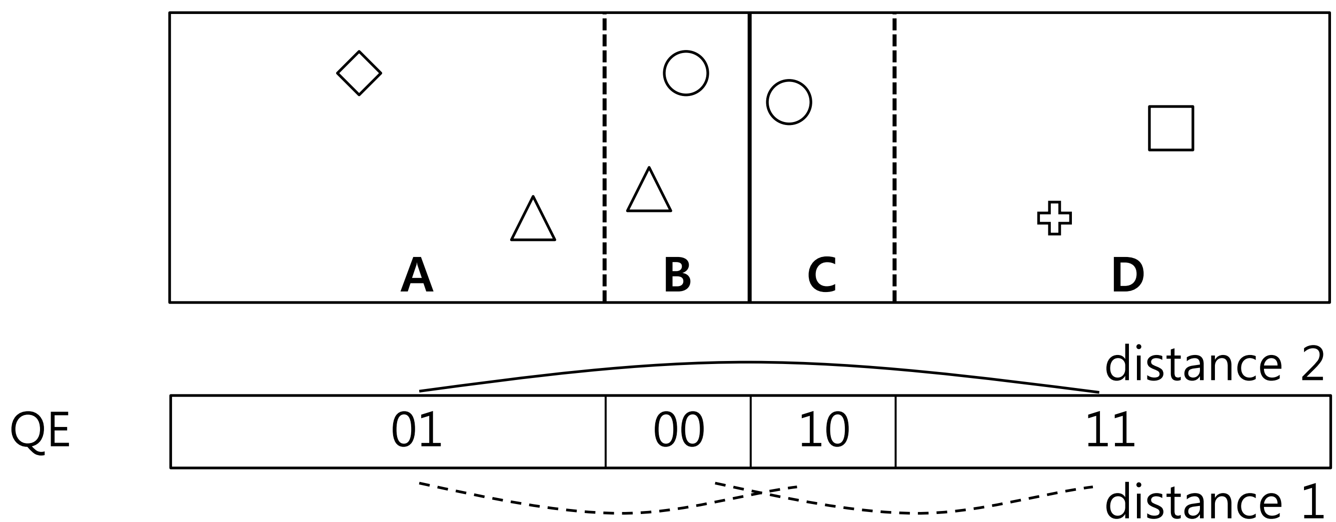
$$
\begin{aligned}
d(X,Y) &= \#\ of\ regions\ between\ X\ and\ Y \\
&= 2|(X_1 \oplus Y_1) \wedge (X_2 \wedge Y_2)| + |(X_1 \oplus Y_1) \wedge (X_2 \oplus Y_2)|
\end{aligned}
$$



- Give distance 0 between neighboring regions to prevent additional quantization error

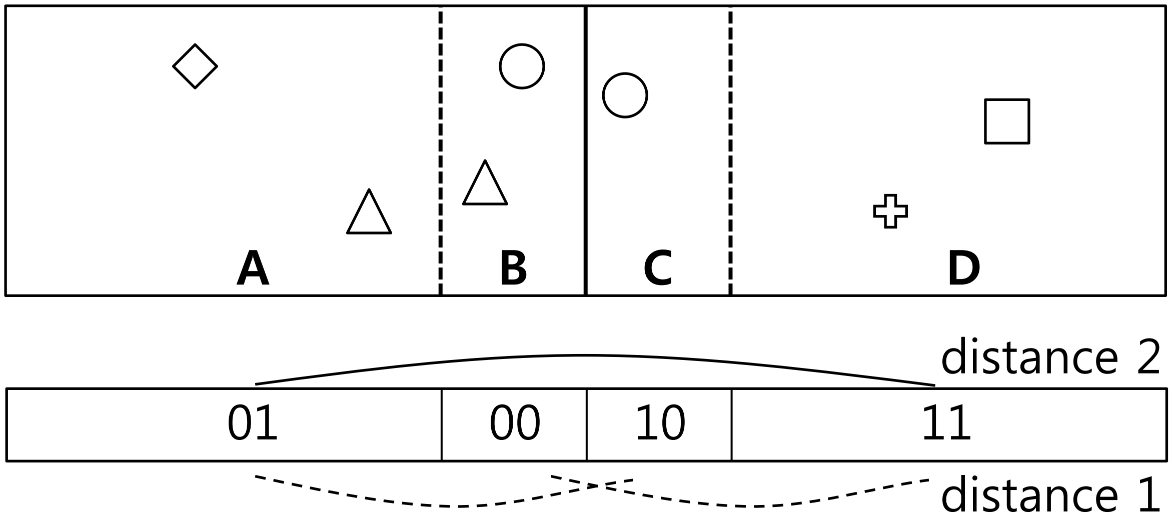
Quadra Embedding Distance

$$d(X,Y) = \#\ of\ regions\ between\ X\ and\ Y$$
$$= \underline{2|(X_1 \oplus Y_1) \wedge (X_2 \wedge Y_2)| + |(X_1 \oplus Y_1) \wedge (X_2 \oplus Y_2)|}$$



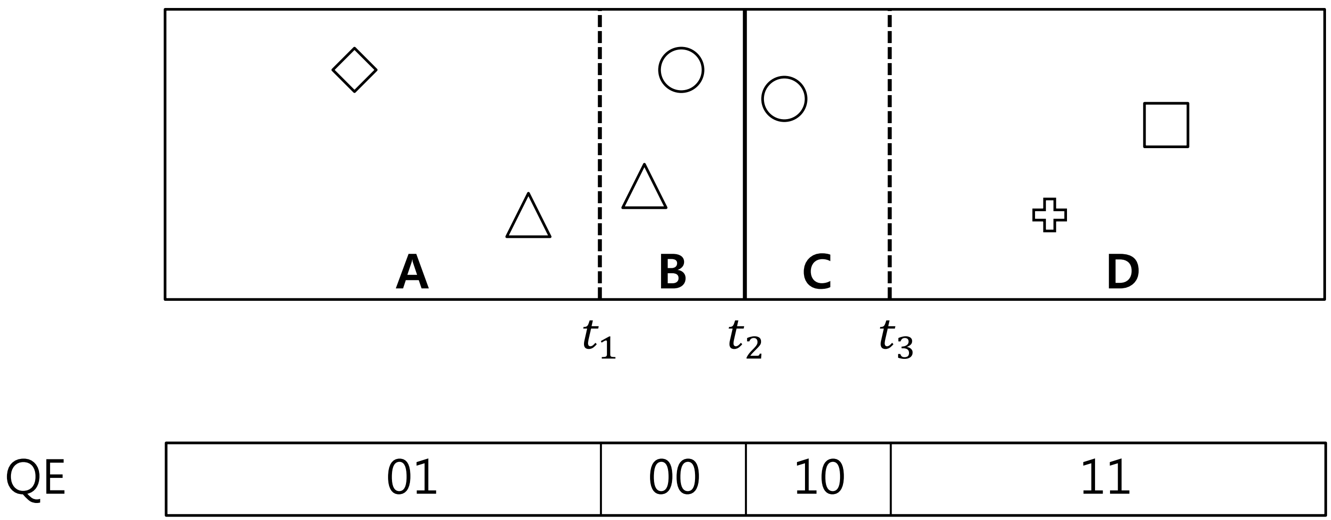
Quadra Embedding Distance

$$d(X,Y) = \#\ of\ regions\ between\ X\ and\ Y$$
$$= 2|(X_1 \oplus Y_1) \wedge (X_2 \wedge Y_2)| + |(X_1 \oplus Y_1) \wedge (X_2 \oplus Y_2)|$$



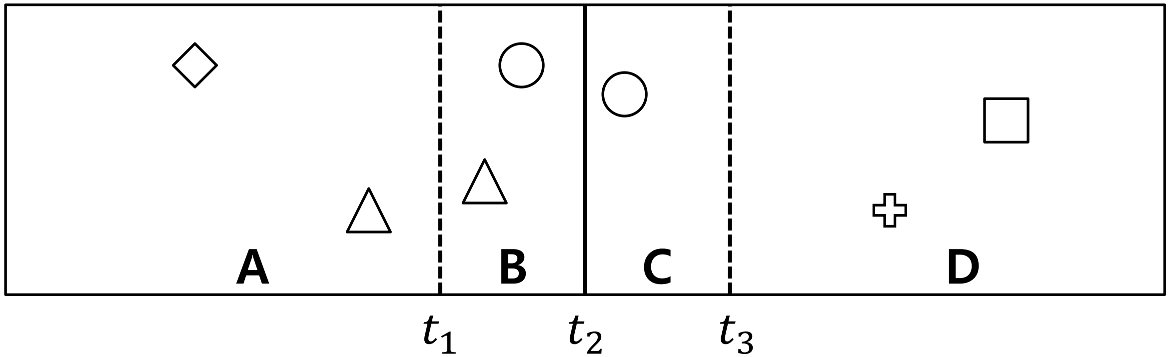
1M distance computation time (8.3 ms) is comparable to the Hamming distance (7.4 ms)

Threshold Optimization



- Minimize variance within each region
- Only consider points near boundaries

Threshold Optimization



QE

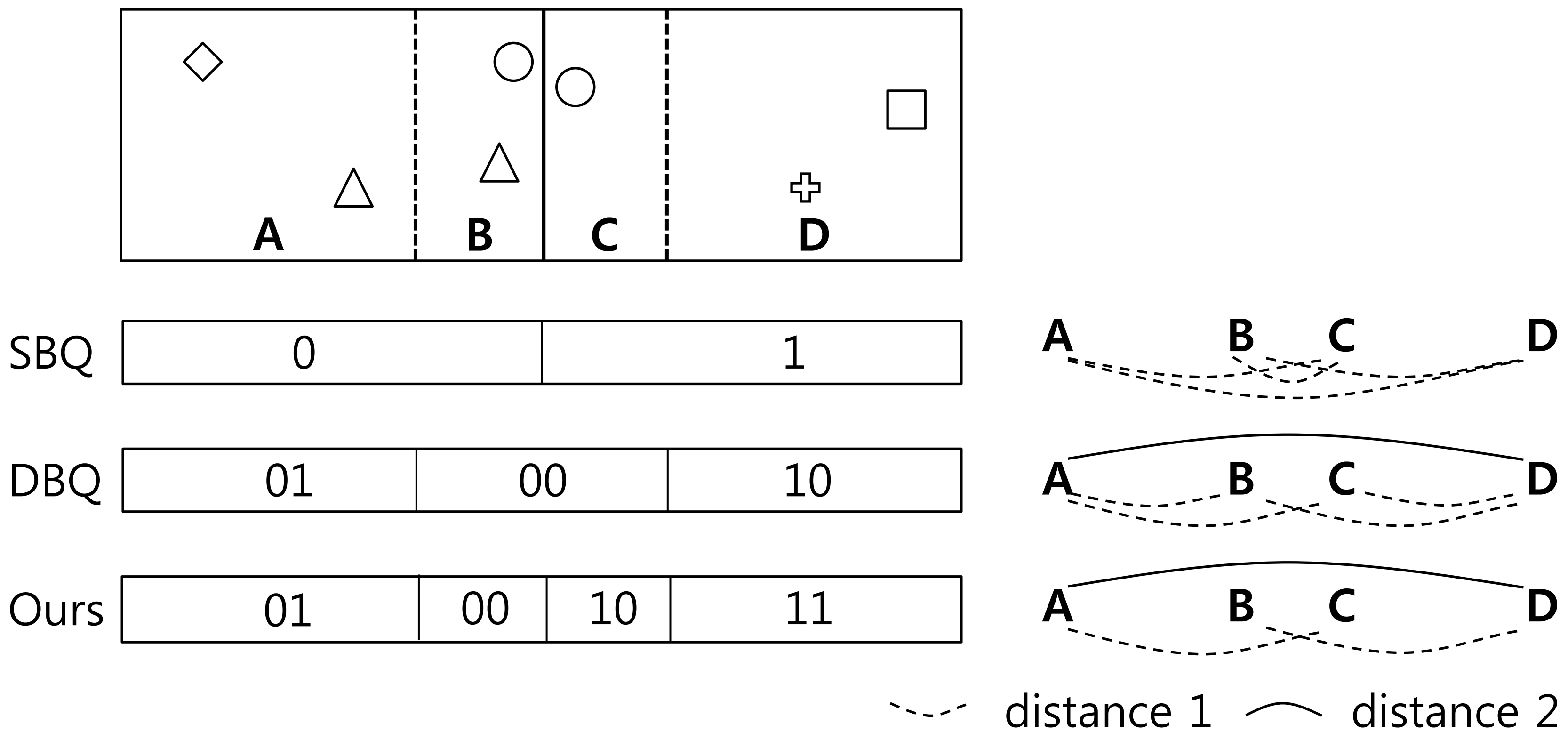
01	00	10	11

$$J = \sum_{p \in P_1} \ell(p-\mu_1)^2 + \sum_{p \in P_2} \ell(\mu_2 - p)^2 + \sum_{p \in P_3} \ell(p-\mu_3)^2 + \sum_{p \in P_4} \ell(\mu_4 - p)^2$$

Evaluation

- Nearest Neighbors (NN) search
  - Find similar images in the image descriptor space
- Protocols
  - k-NN: find points closer than k-th NN
  - ε-NN: find points closer to the distance ε
- mean Average Precision (mAP)
  - Area under the recall-precision curve

Compared Quantization Schemes



Compared Methods

Hashing methods

- LSH [Datar et al. 2004]
- SKLSH [Raginsky et al. 2009]
- SH [Weiss et al. 2008]
- ITQ [Gong et al. 2011]

Quantization schemes

- SBQ
- DBQ
- Ours

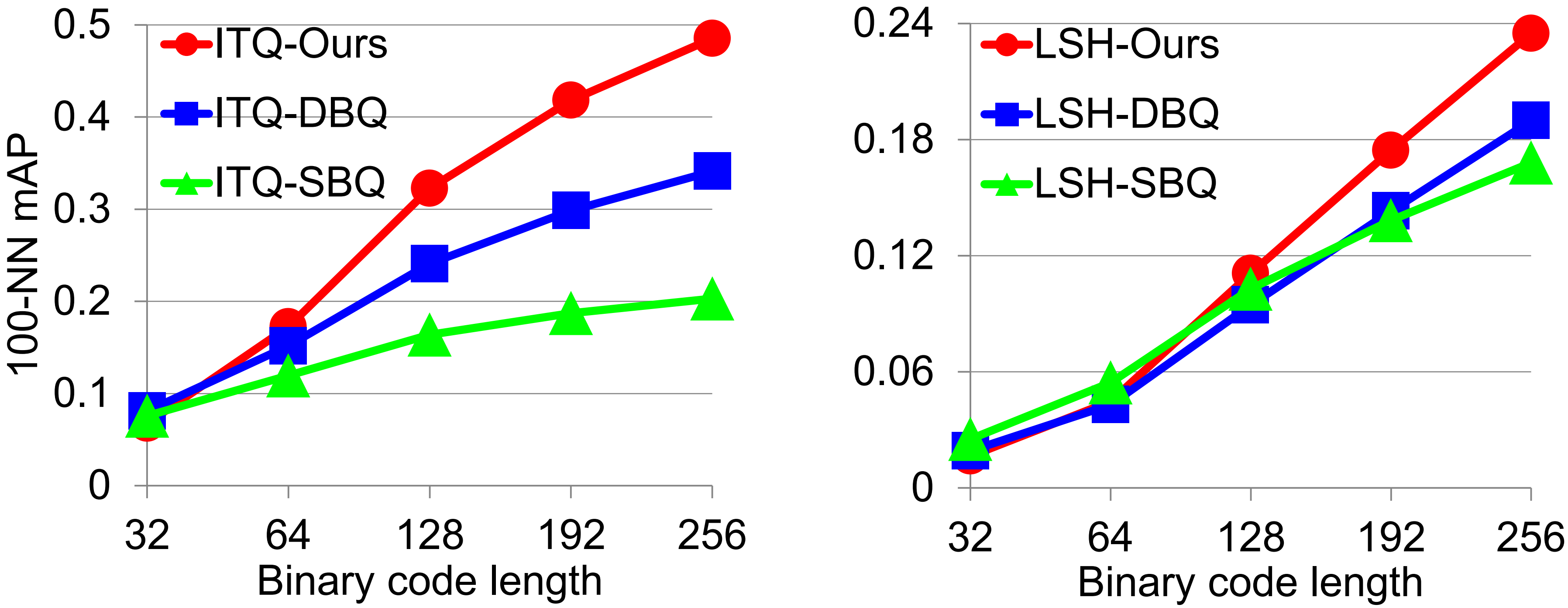
- Every combination of hashing methods and quantization schemes
  - e.g., ITQ-Ours, ITQ-DBQ, ITQ-SBQ

Three varying datasets

- CIFAR-60K-512D
  - 60K images, 512D GIST features
- GIST-1M-960D
  - 1M images from Tiny images, 960D GIST features
- GIST-75M-384D
  - 75M images from Tiny images, 384D GIST features

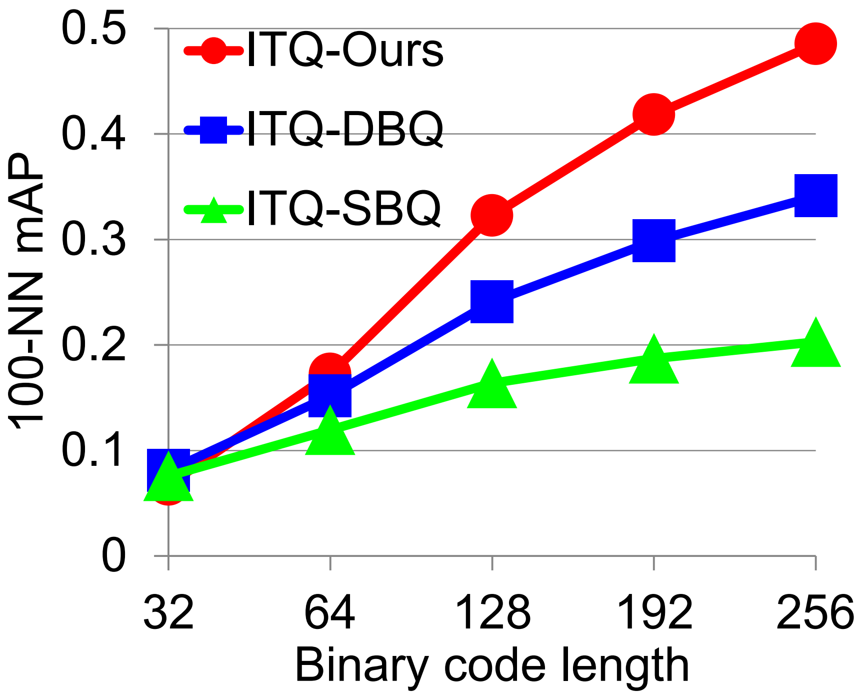
Result on CIFAR-60K dataset

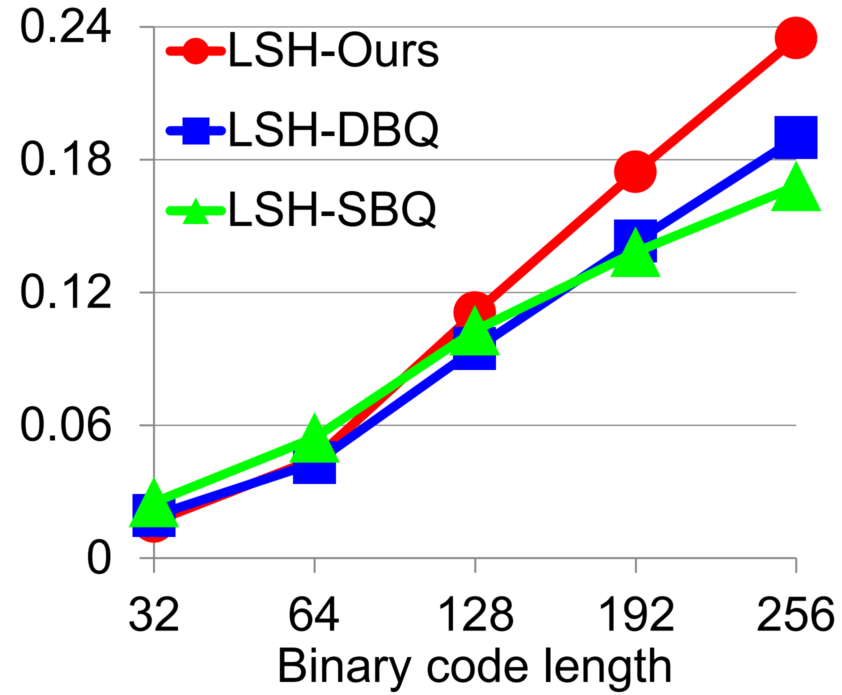
- k-NN



Result on CIFAR-60K dataset

- k-NN

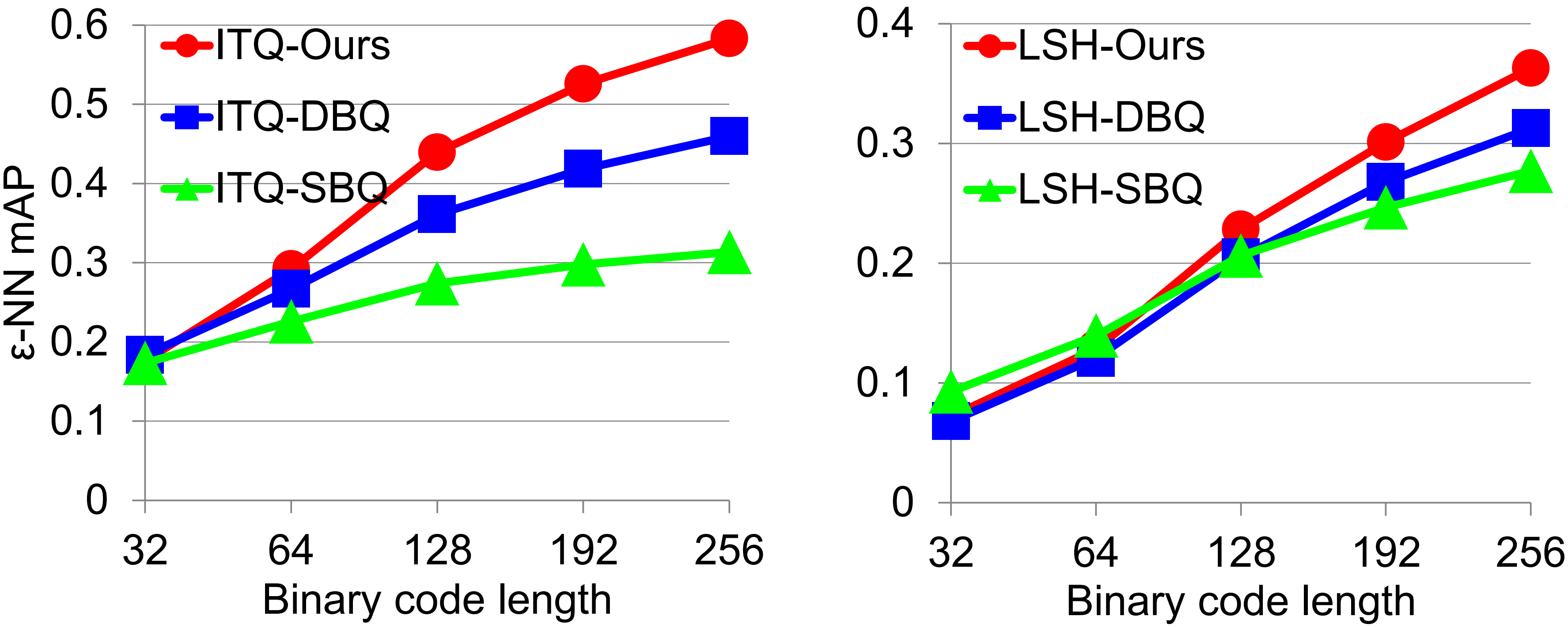




Improvement on both data-dependent and –independent methods

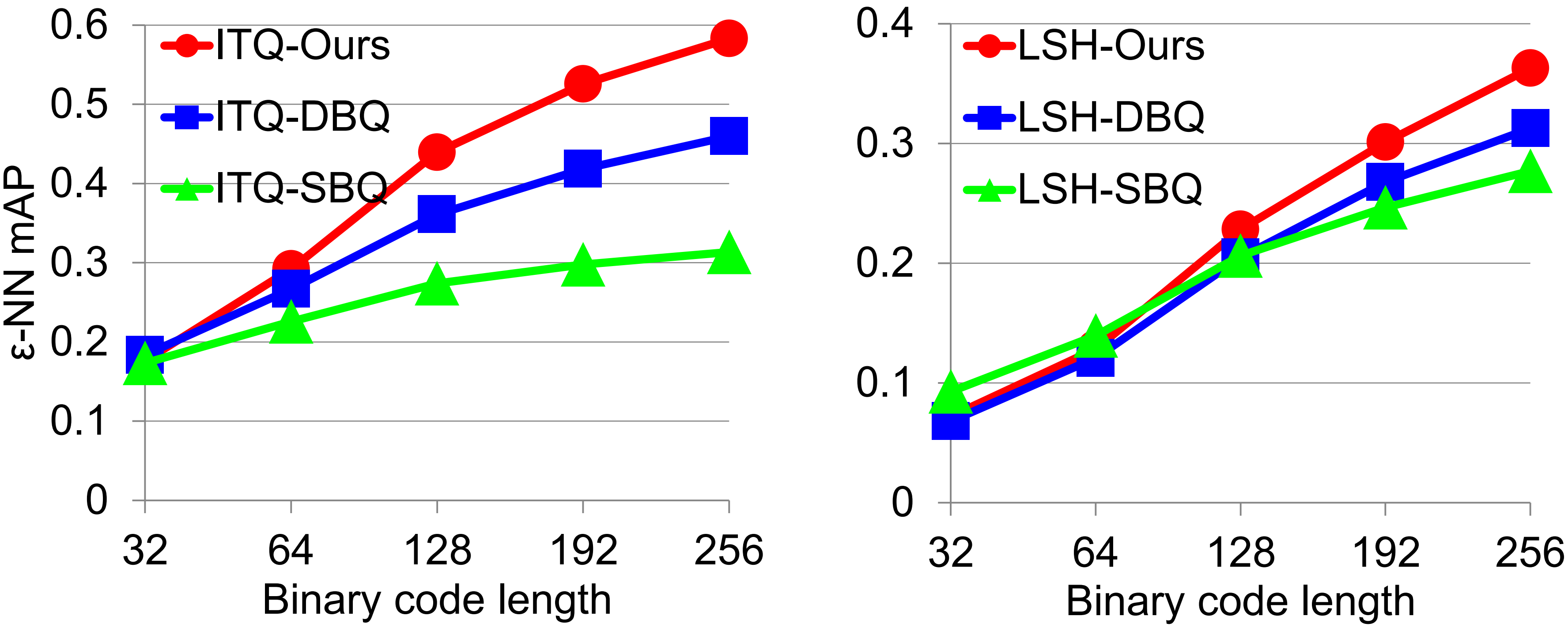
Result on CIFAR-60K dataset

- ε-NN



Result on CIFAR-60K dataset

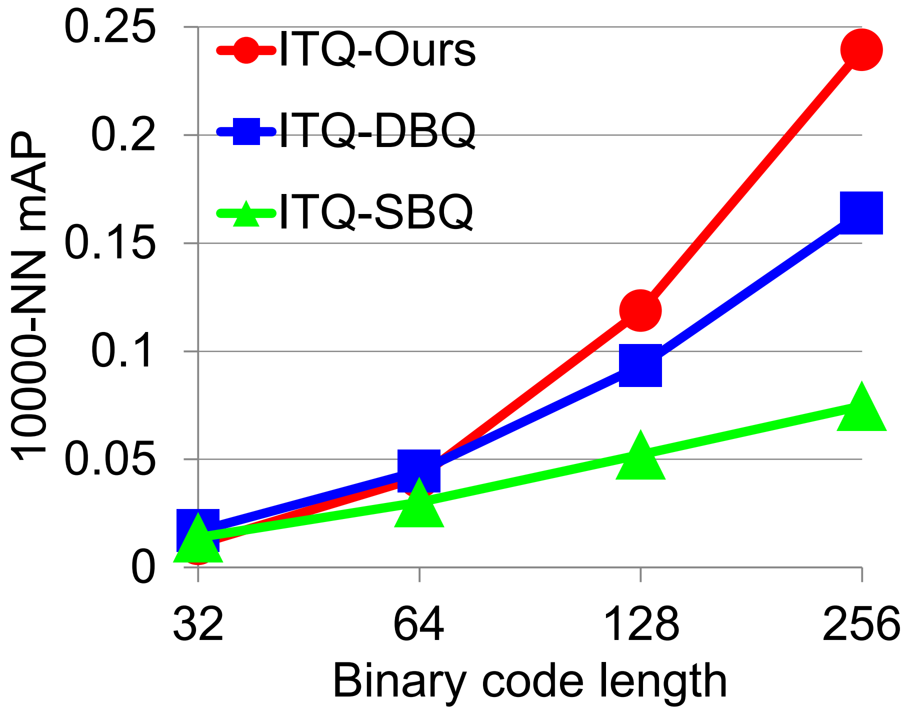
- ε-NN

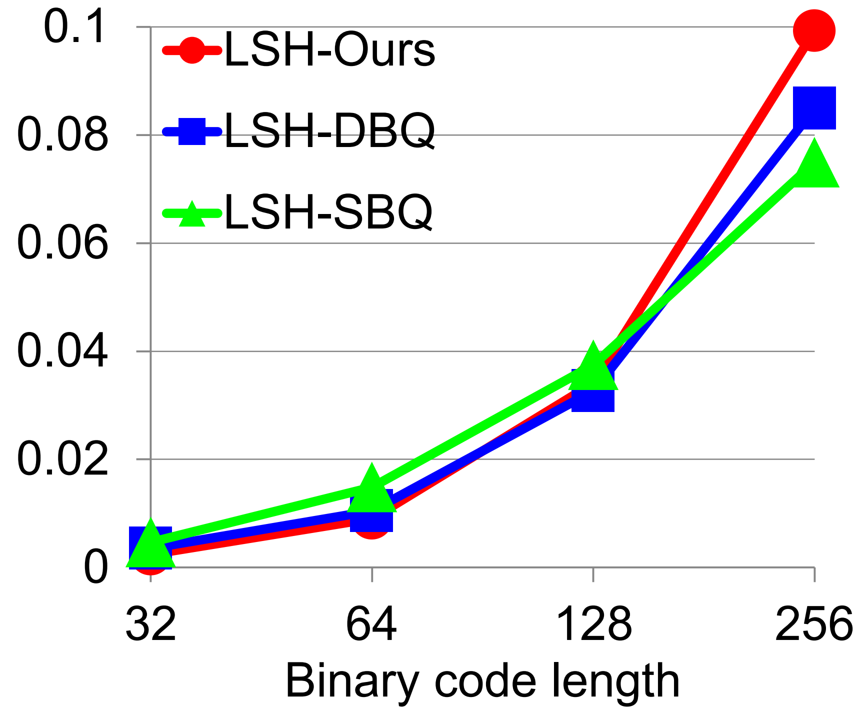


Improvement on both k-NN and ε-NN

Results on Tiny-75M dataset

- k-NN





Conclusions

- Quadra-Embedding
  - Utilizes two bits for each projection
  - Reduces quantization errors with Quadra-Embedding distance
  - Can apply to prior hashing methods

- Outperforms other *state-of-the-art* methods

Future Work

- Optimize hashing functions and encoding scheme simultaneously

- Quantitatively measure quantization errors to directly minimize the quantization error

Thank you.

Quadra-Embedding:
Binary Code Embedding with Low Quantization Error

Source codes are available at
http://sglab.kaist.ac.kr/quadra

ACCV 2012