Efficient Image Retrieval via Decoupling Diffusion into Online and Offline Processing

Fan Yang[1,2], Ryota Hinami[1,2], Yusuke Matsui[1], Steven Ly[2,3], Shin'ichi Satoh[2,1]

[1]The University of Tokyo, Japan

[2]National Institute of Informatics, Japan

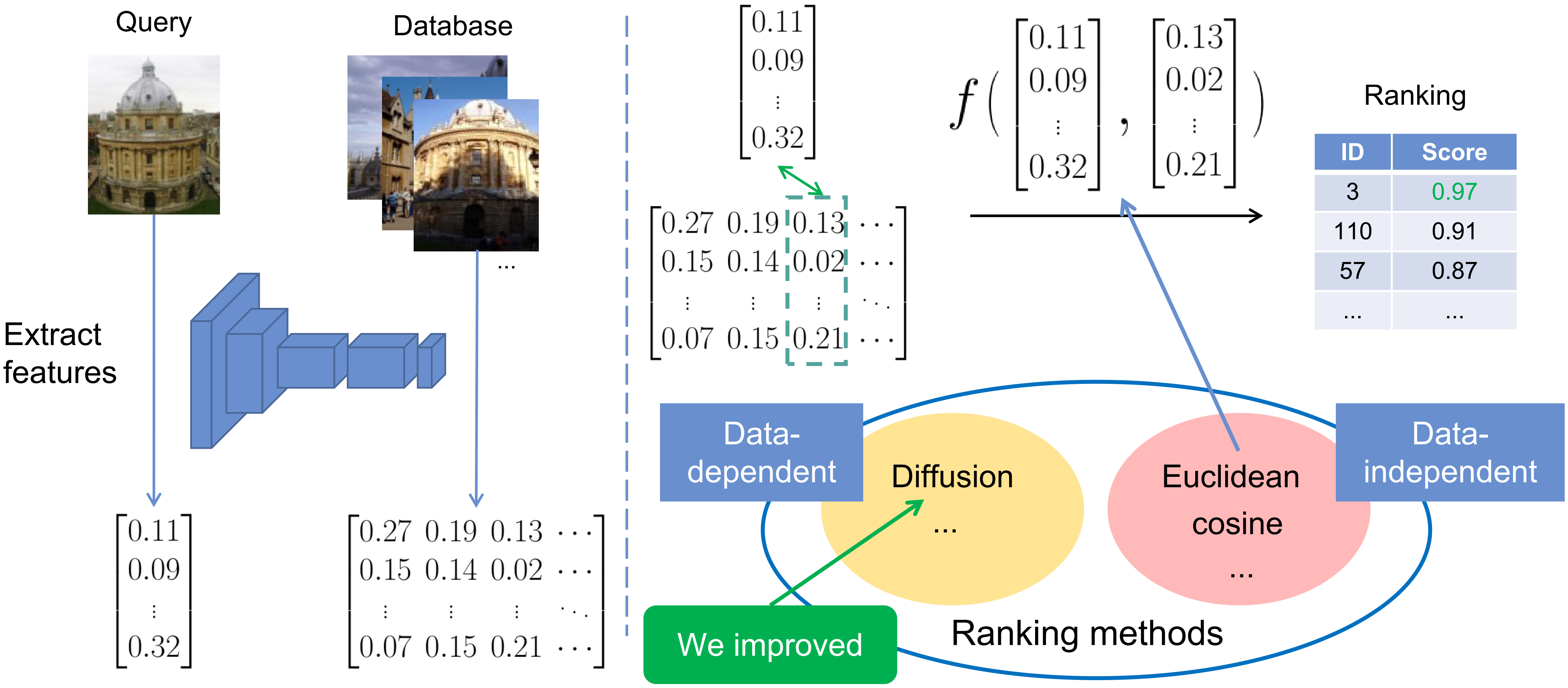
[3]University of Southern California, USA

Introduction to Diffusion

Mainly follows Donoser et al. [CVPR2013] and Iscen et al. [CVPR2017]

Diffusion processes for retrieval revisited, Donoser et al., CVPR 2013
Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations, Iscen et al., CVPR 2017

Image Retrieval



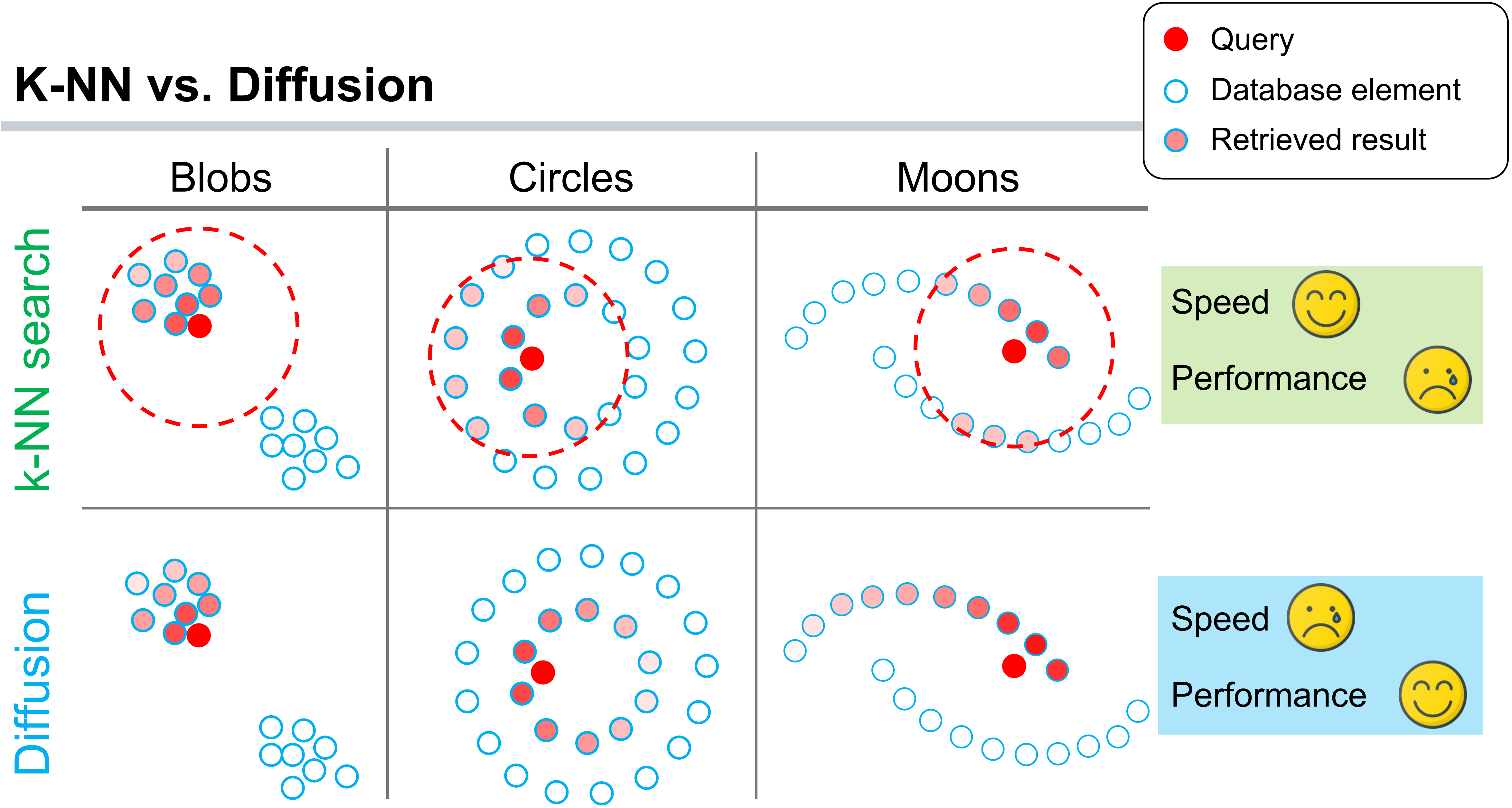
$$\begin{bmatrix}0.11\\0.09\\\vdots\\0.32\end{bmatrix} \quad \begin{bmatrix}0.27 & 0.19 & 0.13 & \cdots\\0.15 & 0.14 & 0.02 & \cdots\\\vdots & \vdots & \vdots & \ddots\\0.07 & 0.15 & 0.21 & \cdots\end{bmatrix}$$

$$f\left(\begin{bmatrix}0.11\\0.09\\\vdots\\0.32\end{bmatrix}, \begin{bmatrix}0.13\\0.02\\\vdots\\0.21\end{bmatrix}\right) \longrightarrow$$

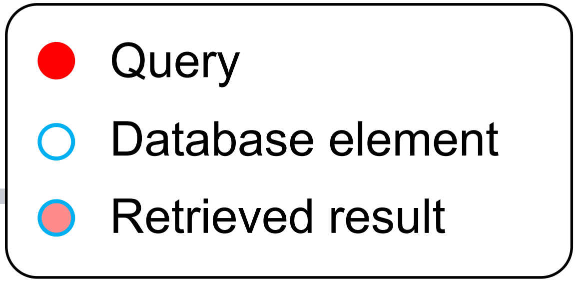
Ranking

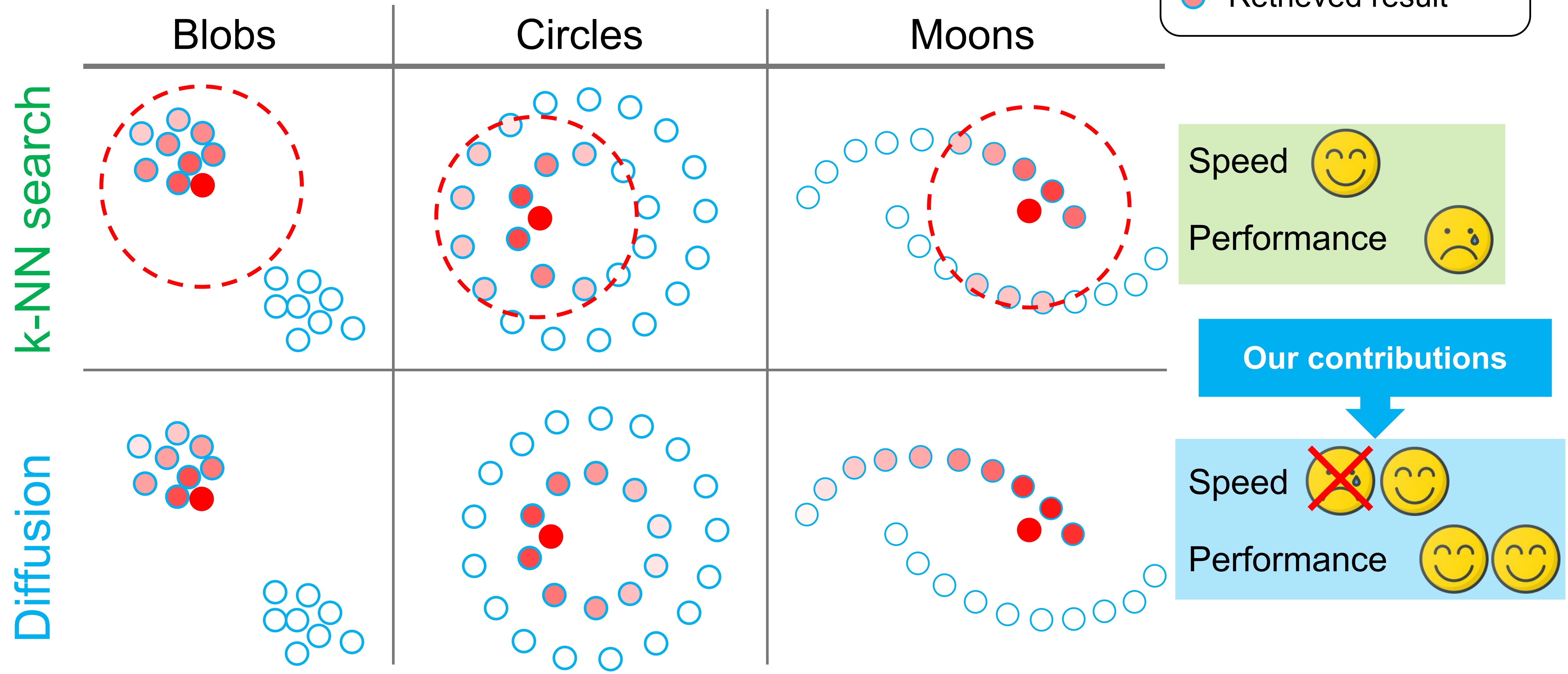
ID	Score
3	0.97
110	0.91
57	0.87
...	...

K-NN vs. Diffusion



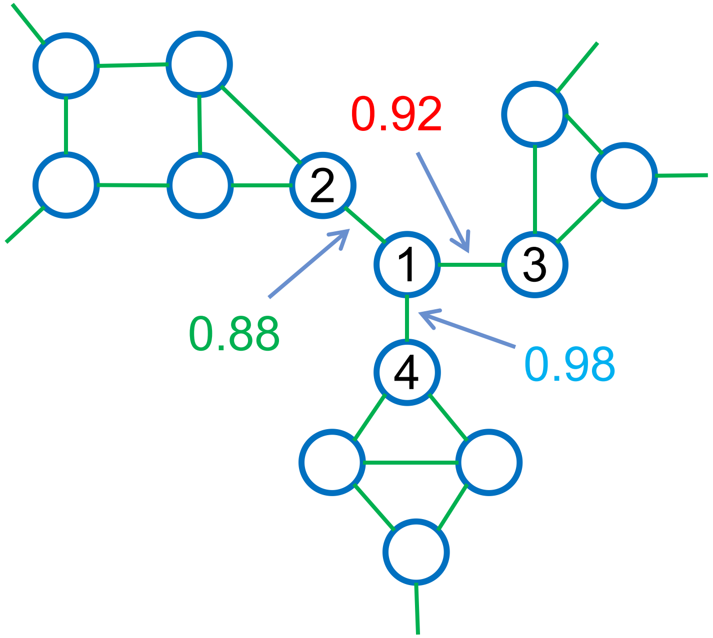
K-NN vs. Diffusion





Preliminaries of Diffusion

- Graph construction



Node: image

Edge: similarity between images

- Step1: Extract features of images

- Step2: Perform k-NN search, using cosine similarity as the metric

  e.g.

  query: ①

  neighbors: ② ③ ④

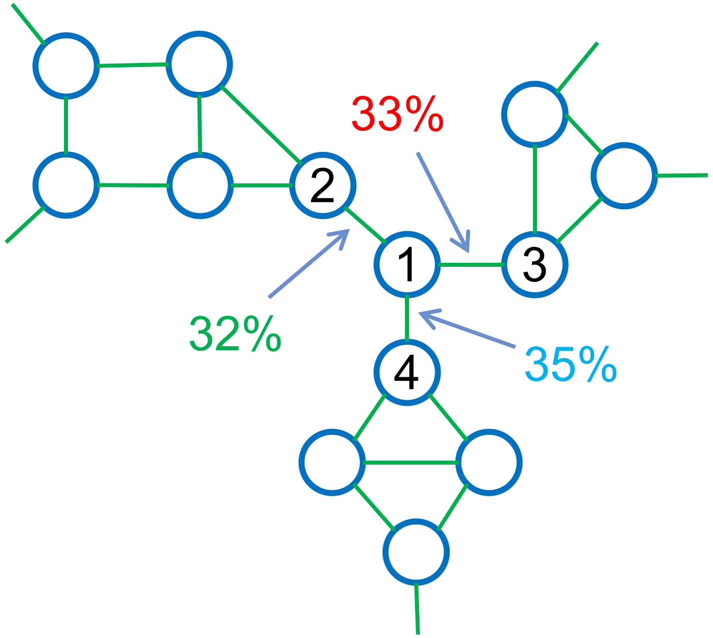
  similarities: ①-② 0.88, ①-③ 0.92, ①-④ 0.98

  affinity matrix $\mathbf{A} = \begin{bmatrix} 0 & 0.88 & 0.92 & 0.98 & \cdots \\ 0.88 & 0 & 0 & 0 & \cdots \\ 0.92 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$

- Step3: Construct graph with k-NN search results

Preliminaries of Diffusion

- Graph normalization



Node: image

Edge: transition probabilities

- Normalize similarities to transition probabilities:
  e.g.
  [0.88, 0.92, 0.98] -> [32%, 33%, 35%]

- Popular choices of normalization

$$\mathbf{S} = \mathbf{D}^{-1}\mathbf{A}$$

$$\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

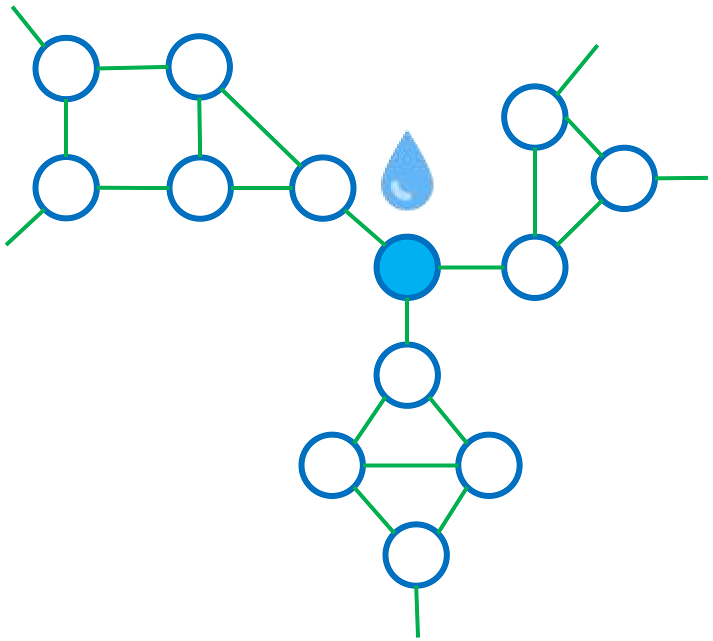
$$\mathbf{D}_{ii} = \sum_{j=0}^{n} \mathbf{A}_{ij}$$

transition matrix / stochastic matrix

degree matrix

Preliminaries of Diffusion

- Random walk



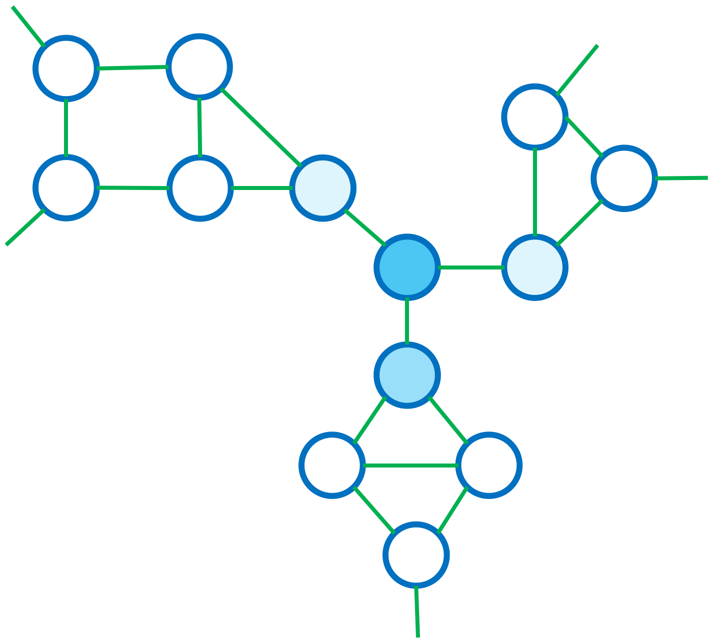
- Start from a single node (place a droplet of ink onto the node)

Node: image

Edge: transition probabilities

Preliminaries of Diffusion

- Random walk



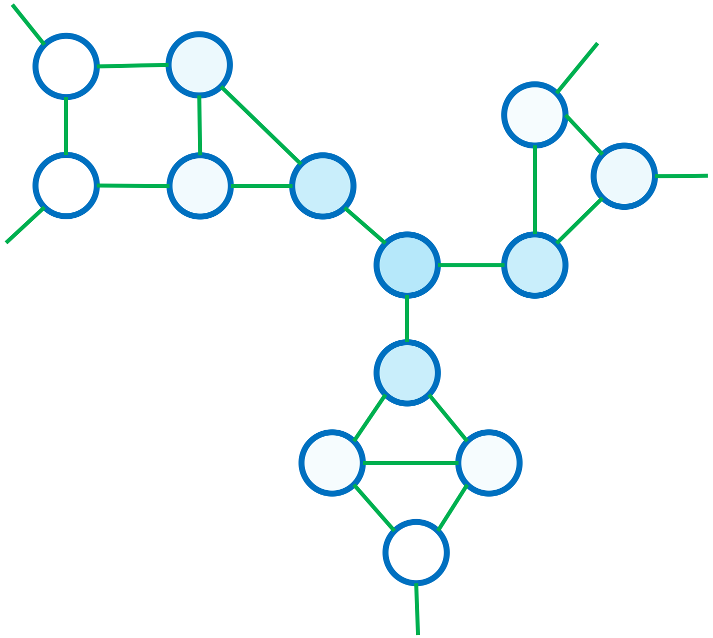
- Start from a single node (place a droplet of ink onto the node)
- Random transition to other nodes

Node: image

Edge: transition probabilities

Preliminaries of Diffusion

- Random walk



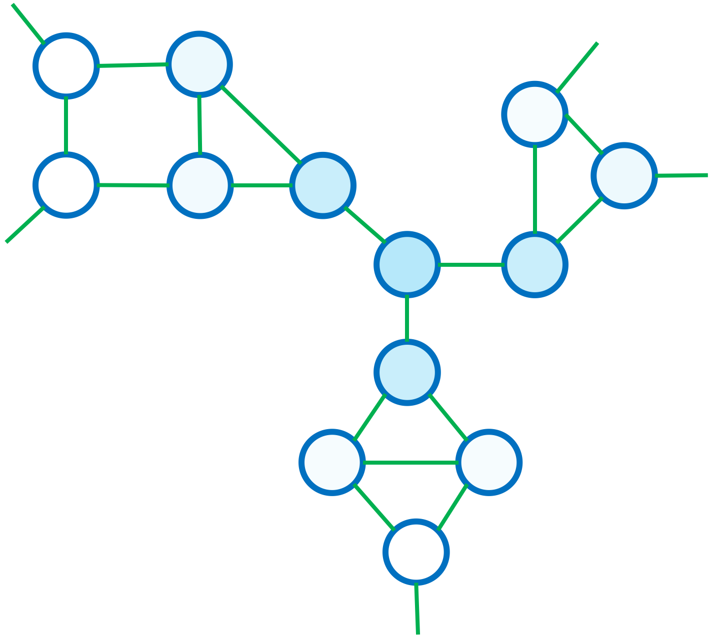
- Start from a single node (place a droplet of ink onto the node)
- Random transition to other nodes
- Converge to a stable state

Node: image

Edge: transition probabilities

Preliminaries of Diffusion

- Random walk



Node: image

Edge: transition probabilities

- Start from a single node (place a droplet of ink onto the node)
- Random transition to other nodes
- Converge to a stable state
- Obtain ranking scores (amount of ink on each node)

ID	Score
1	0.45
4	0.23
3	0.17
2	0.11
...	...

$$\mathbf{f}^* = [0.45, 0.11, 0.17, 0.23, \cdots]^\top$$

score vector

Preliminaries of Diffusion

- Random walk step

the first droplet of ink

$$\mathbf{f}^{t+1} = \underbrace{\alpha \mathbf{S} \mathbf{f}^{t}}_{\text{99\% probability to transition}} + \underbrace{(1-\alpha)\mathbf{f}^{0}}_{\text{1\% probability to restart}}, \mathbf{f}^{0} = [1, 0, 0, \cdots]^{\top}$$

when $\alpha = 0.99$

Preliminaries of Diffusion

- Random walk step

the first droplet of ink

$$\mathbf{f}^{t+1} = \underbrace{\alpha \mathbf{S} \mathbf{f}^t}_{\text{99\% probability to transition}} + \underbrace{(1-\alpha)\mathbf{f}^0}_{\text{1\% probability to restart}}, \mathbf{f}^0 = [1, 0, 0, \cdots]^\top$$

when $\alpha = 0.99$

iterate steps

- Closed-form solution

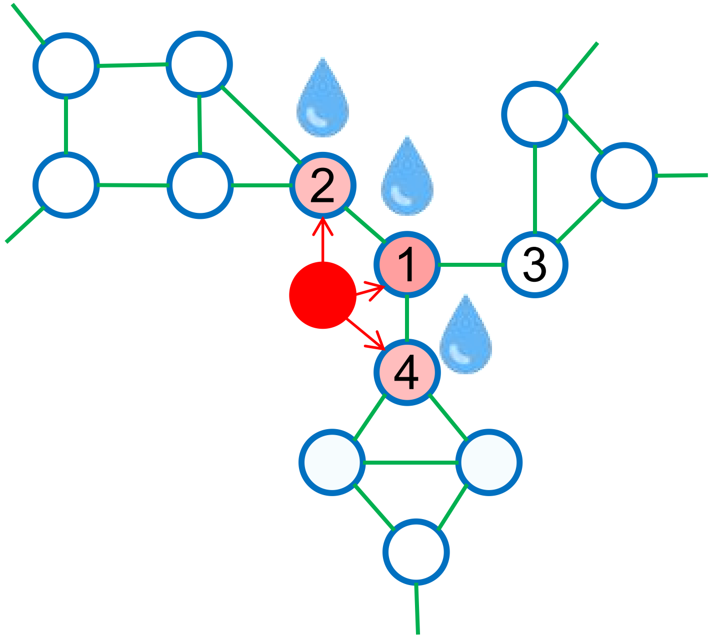
$$\underbrace{\mathbf{f}^*}_{\text{score vector}} = (1-\alpha)(\underbrace{\mathbf{I} - \alpha \mathbf{S}}_{\text{Laplacian matrix}})^{-1}\mathbf{f}^0$$

$$\mathcal{L}_\alpha = \mathbf{I} - \alpha \mathbf{S}$$

$\mathbf{f}^* \propto (\mathbf{I} - \alpha \mathbf{S})^{-1}\mathbf{f}^0 \Rightarrow \boxed{\mathcal{L}_\alpha \mathbf{f}^* \propto \mathbf{f}^0} \Rightarrow$ solve $\mathbf{A}\mathbf{x} = \mathbf{b}$

Preliminaries of Diffusion

- Handling unseen queries



Images in database

A query image

- Decompose transition/stochastic matrix

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{qq} & \mathbf{S}_{qd} \\ \mathbf{S}_{dq} & \mathbf{S}_{dd} \end{bmatrix}$$

- Closed-form solution for any new queries

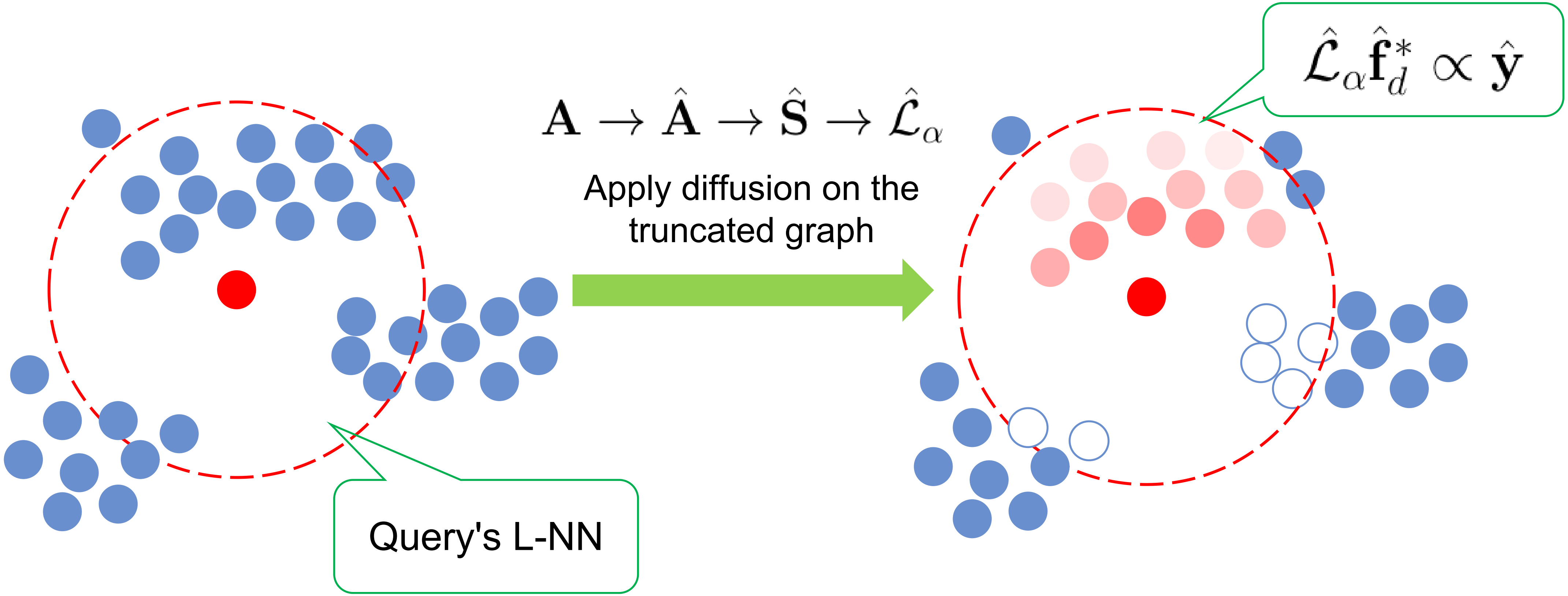
$$(\mathbf{I} - \alpha \mathbf{S}_{dd})\mathbf{f}_d^* \propto \mathbf{y}, \underline{\mathbf{y} = \mathbf{S}_{dq}\mathbf{f}_q^0}$$

$$[0.97, 0.88, 0, 0.79, 0, 0, \cdots]^\top$$

query's NN: 1 2 4

Preliminaries of Diffusion

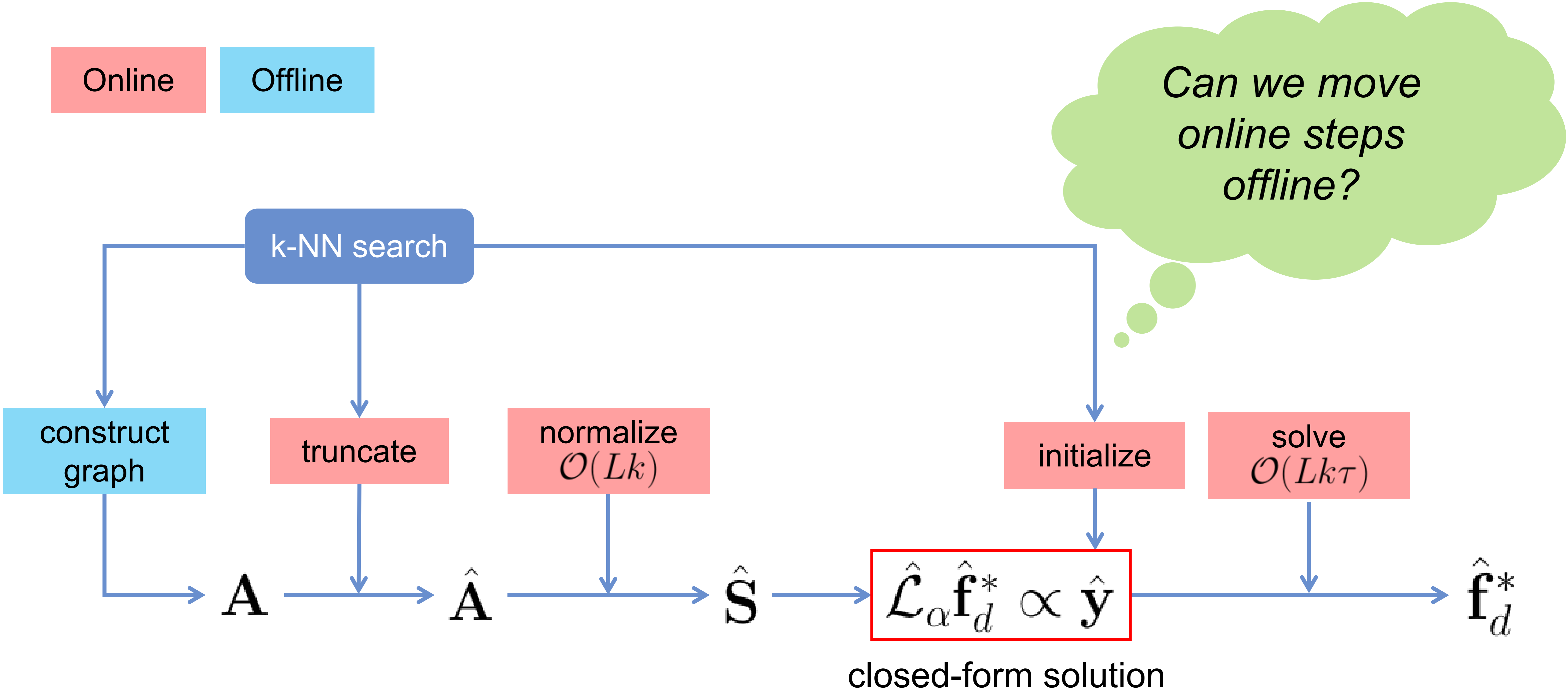
- Truncation
  - An approach to scale up diffusion for large datasets



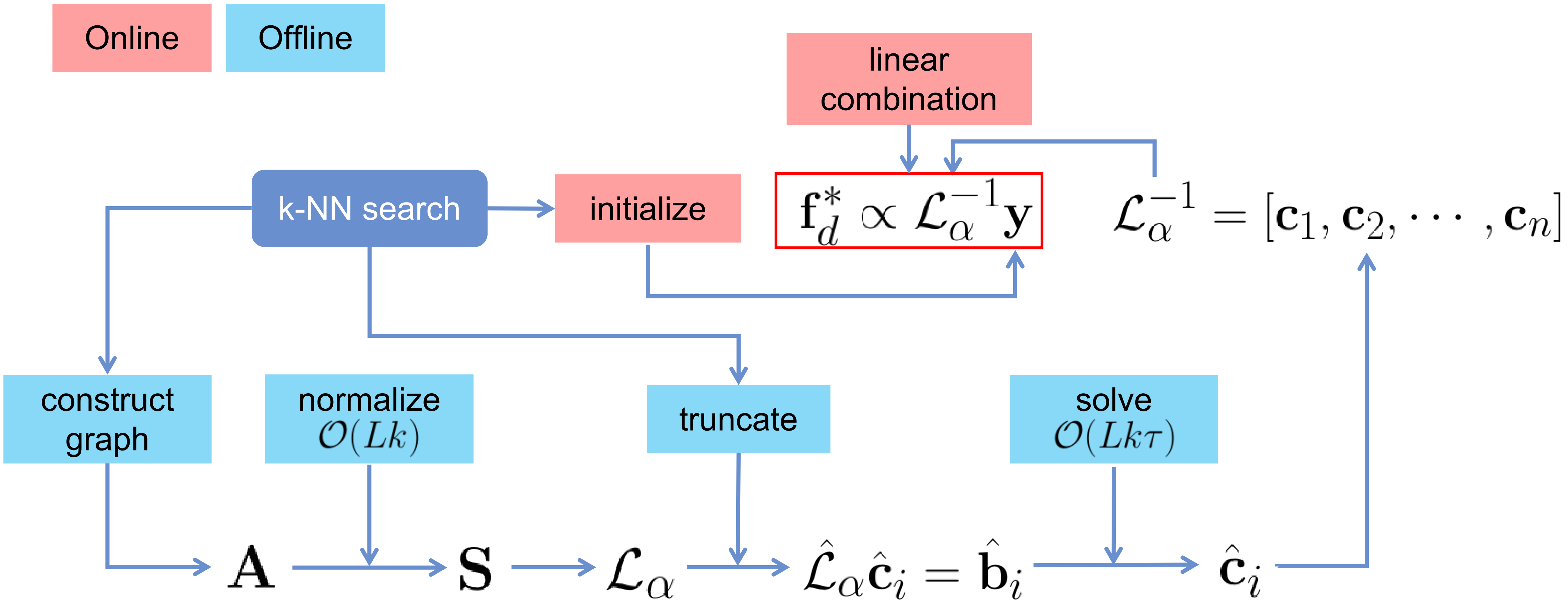
Proposed Method

We propose an efficient diffusion with significant improvement in retrieval performance

Overview of Previous Method



Overview of Proposed Method

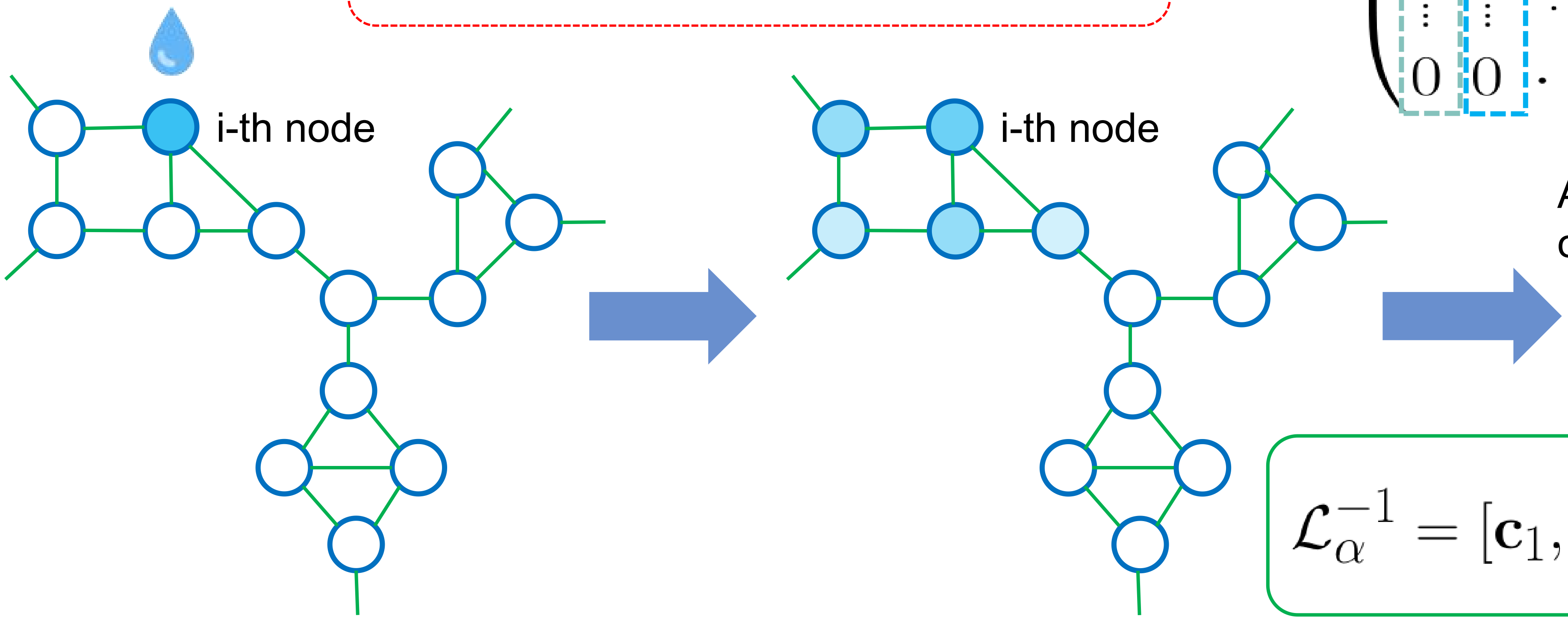


Proposed Method

- Offline Pre-computations

$$\mathbf{c}_i = \mathcal{L}_\alpha^{-1}\mathbf{b}_i \qquad \mathbf{f}_d^* \propto \mathcal{L}_\alpha^{-1}\mathbf{y}$$

$$\begin{array}{cccc} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ \hline \end{array}$$
$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}_{n \times n}$$



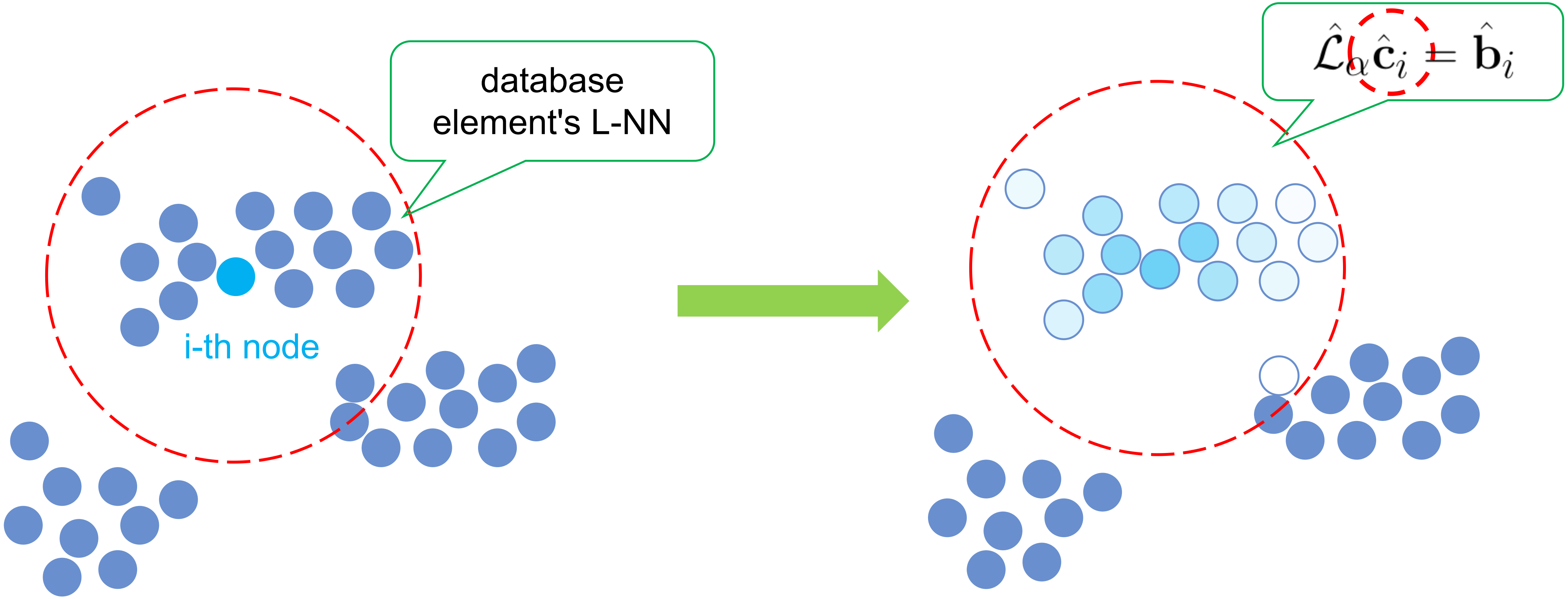
Amount of ink on each node

$$\mathbf{c}_i$$

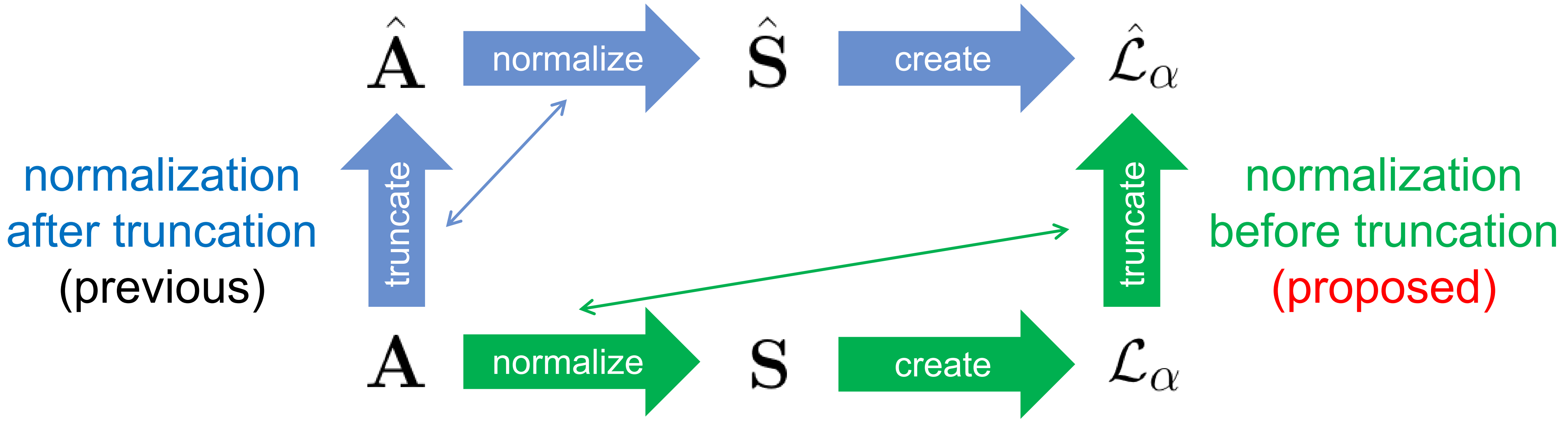
$$\mathcal{L}_\alpha^{-1} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n]$$

Proposed Method

- Database-side truncation



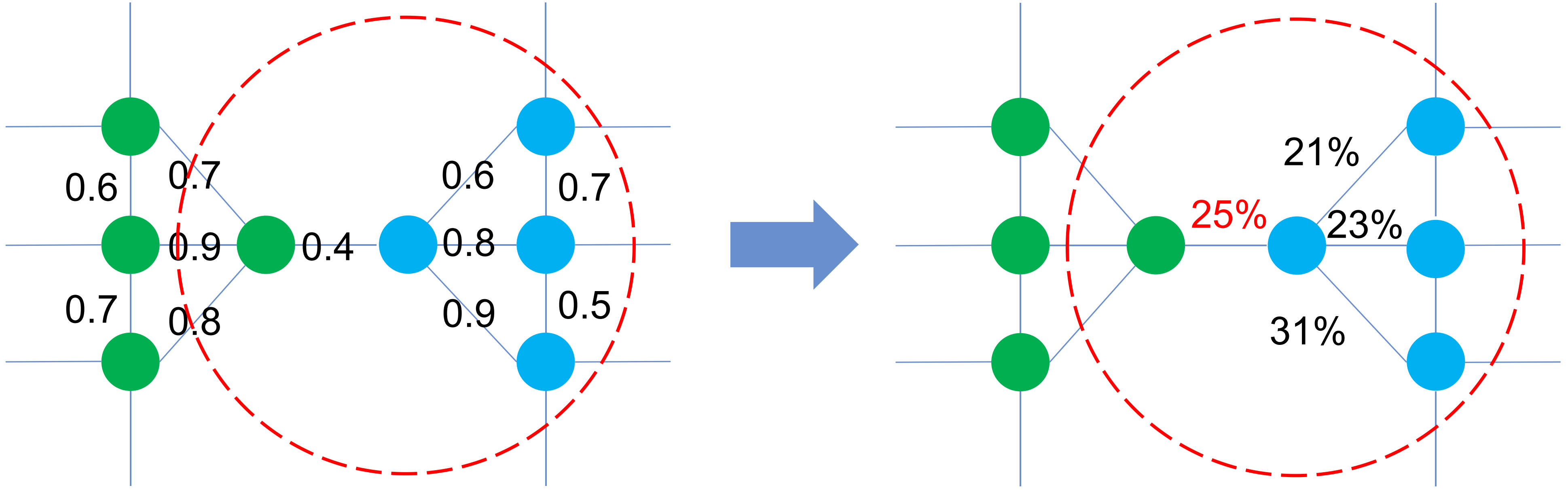
Early Truncation vs. Late Truncation



Early Truncation

$$\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

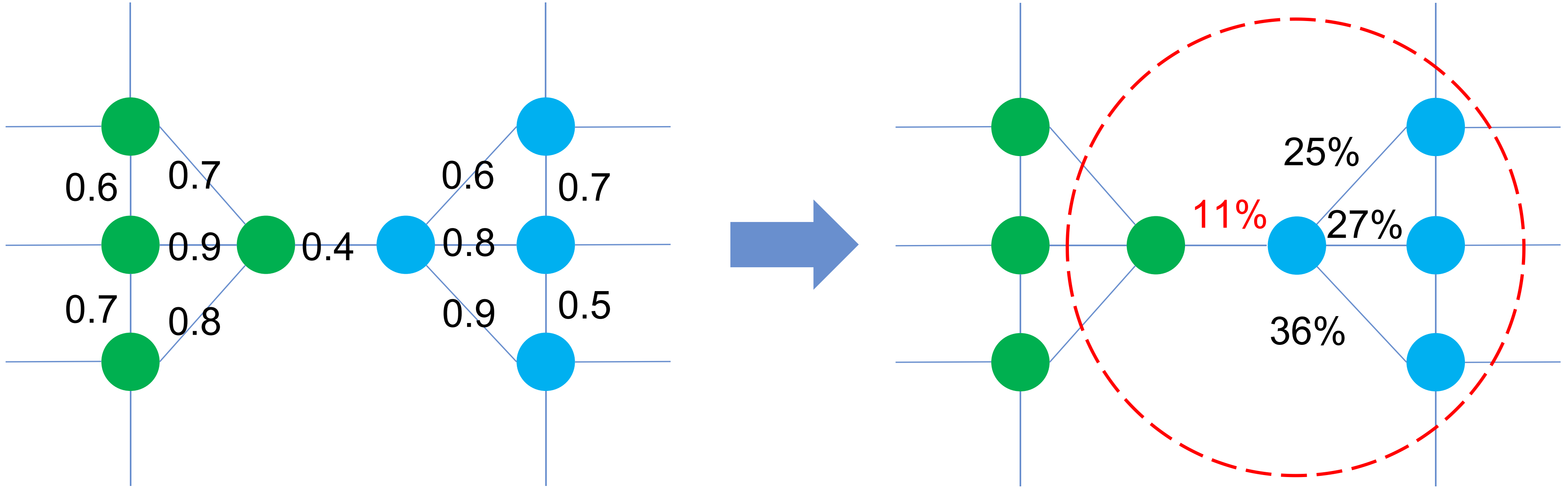
$\mathbf{A}$ truncate $\hat{\mathbf{A}}$ normalize $\hat{\mathbf{S}}$ We call this subgraph normalization



Late Truncation

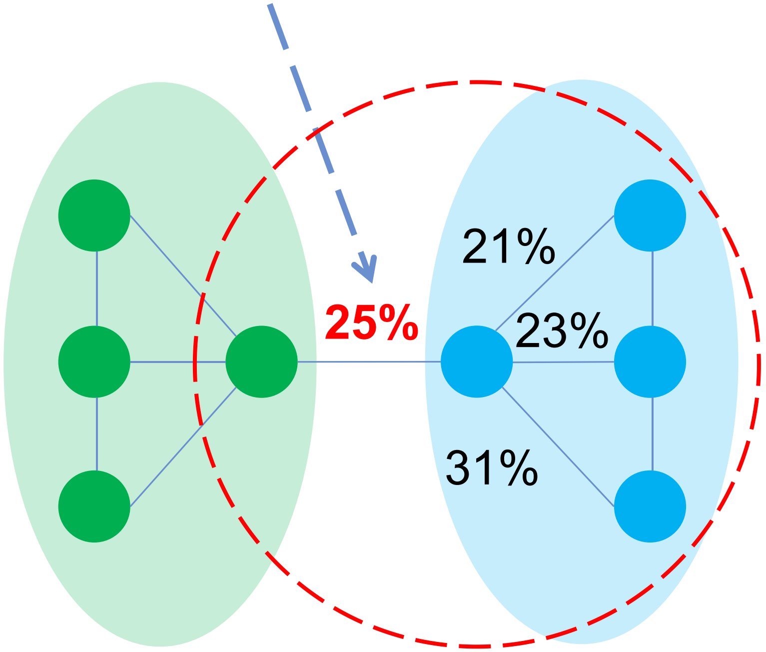
$$\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

In real implementation, we apply truncation to Laplacian

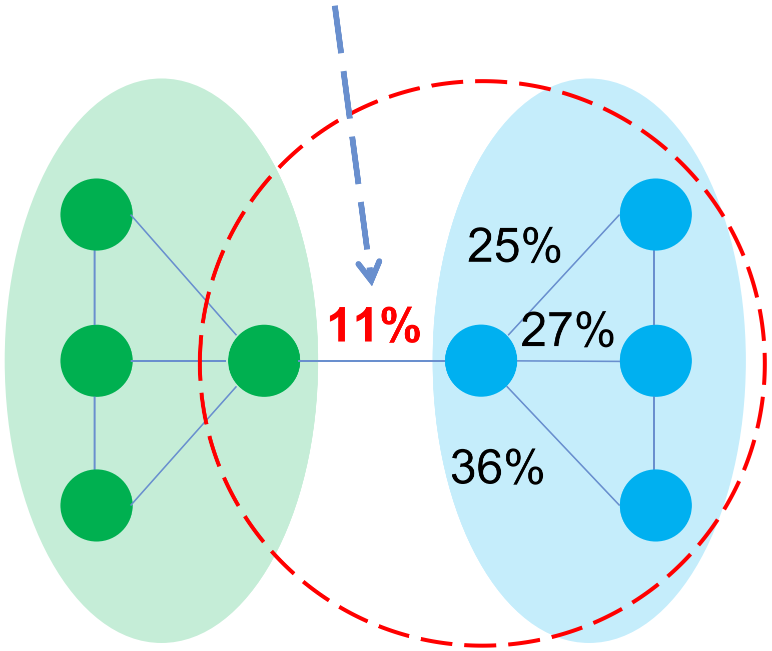


Early Truncation vs. Late Truncation

Early truncation utilizes the **truncated graph** in normalization and results in misleading high transition probabilities to partial manifolds after truncation



Late truncation utilizes the **complete graph** in normalization which avoids the problem in early truncation

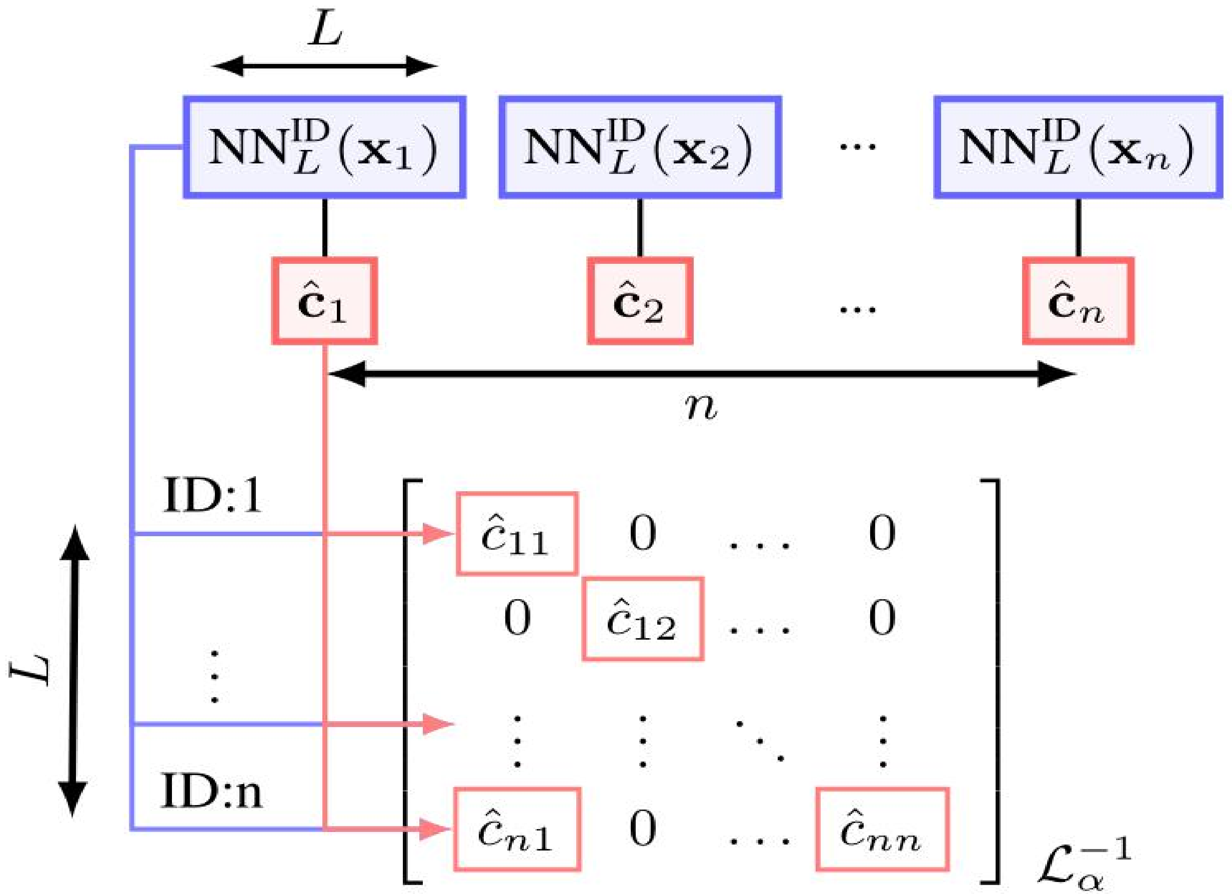


Proposed Method

- Sparsified structure

$$\hat{\mathcal{L}}_\alpha \hat{\mathbf{c}}_i = \hat{\mathbf{b}}_i$$

We maintain a sparsified inverse of Laplacian, which takes O(L*n) memory. L<<n (e.g. L=2k, n=100k)



Proposed Method

- Online search

Algorithm 1 Online search

Input $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\} \leftarrow$ a new query
Output $\mathbf{f}_d^* \leftarrow$ a new array of n zeros
for $i \leftarrow 1$ to m **do**
  obtain $\mathrm{NN}_k^{\mathrm{SIM}}(\mathbf{q}_i), \mathrm{NN}_k^{\mathrm{ID}}(\mathbf{q}_i)$ by k-NN search
  for $j \leftarrow 1$ to k **do**
    $\texttt{col\_id} \leftarrow \mathrm{NN}_k^{\mathrm{ID}}(\mathbf{q}_i)[j]$
    $\texttt{weight} \leftarrow \mathrm{NN}_k^{\mathrm{SIM}}(\mathbf{q}_i)[j]$
    $\texttt{row\_ids} \leftarrow \mathrm{NN}_L^{\mathrm{ID}}(\mathbf{x}_{\texttt{col\_id}})$ from sparsified $\mathcal{L}_\alpha^{-1}$
    $\mathbf{f}_d^*[\texttt{row\_ids}] \leftarrow \mathbf{f}_d^*[\texttt{row\_ids}] + \texttt{weight} * \hat{\mathbf{c}}_{\texttt{col\_id}}$
Aggregate scores in $\mathbf{f}_d^*$ to image level if needed

$$\mathbf{f}_d^* \propto \mathcal{L}_\alpha^{-1}\mathbf{y}$$

$$\mathcal{L}_\alpha^{-1} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n]$$

This algorithm computes the above linear combination

Experiments and Results

We show that our method improved both the speed and the performance

Datasets

The Oxford Buildings

James Philbin, Relja Arandjelović and Andrew Zisserman

Overview

The Oxford Buildings Dataset consists of 5062 images collecte
manually annotated to generate a comprehensive ground tru
55 queries over which an object retrieval system can be evalu

Groundtruth Queries

The following image shows all 55 queries used to evaluate pe

The Paris Dataset

James Philbin and Andrew Zisserman

Overview

The Paris Dataset consists of 6412 images collected from Flickr by searching for particular Paris landmarks.

55 query
5062 gallery

Flickr 100k as distractors

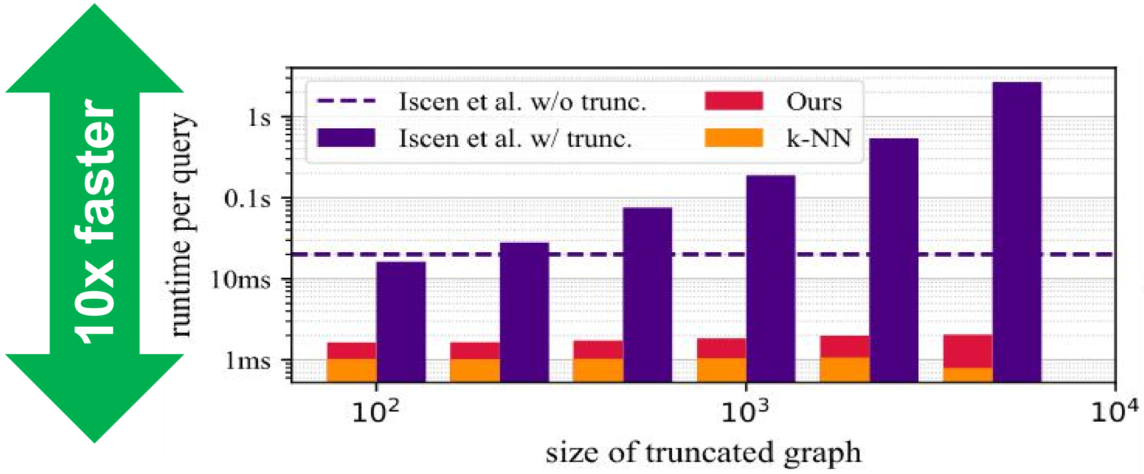
55 query
6412 gallery

Experiment Settings

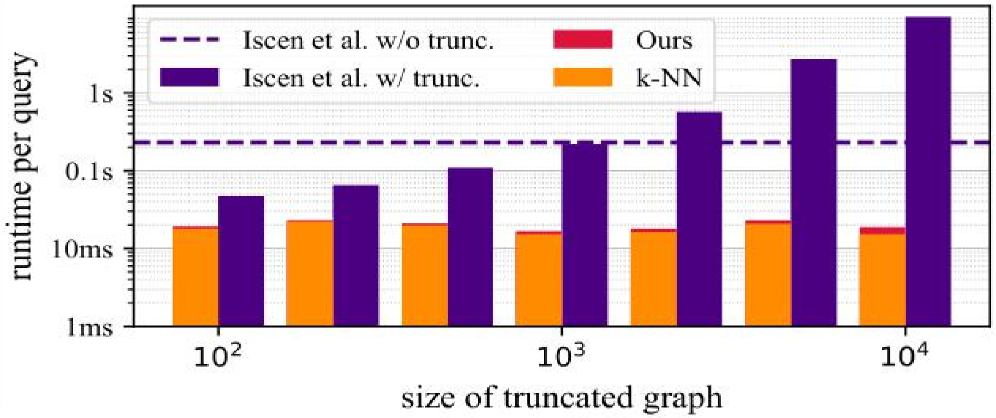
- Features
  - 512-d and 1,024-d R-MAC descriptors based on VGG and ResNet respectively (provided online by Iscen et al. https://github.com/ahmetius/diffusion-retrieval)
- Parameters
  - We follow the parameters in previous works and we comfirm that deviating from these parameters results in worse performance
- Library
  - We use Facebook FAISS (https://github.com/facebookresearch/faiss) to conduct k-NN search

Runtime Computational Efficiency

- We evaluate the computational efficiency between k-NN search, Iscen's method [CVPR2017], and our proposed method



(a) Oxford5k



(b) Oxford105k

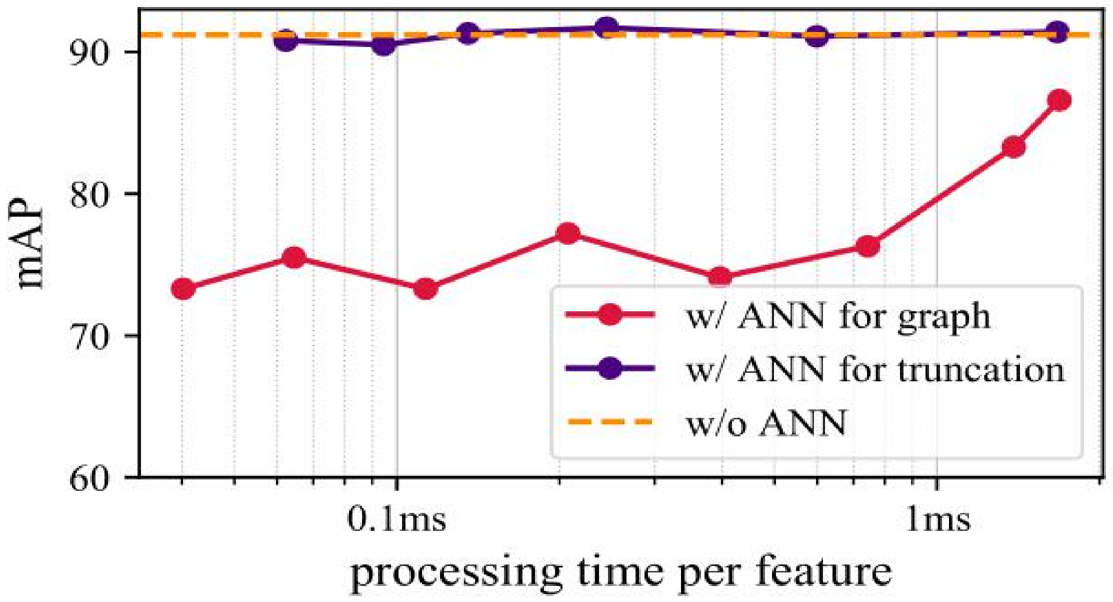
speed

proposed method ≈ k-NN search

Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations, Iscen et al., CVPR 2017

Pre-computational Efficiency

- We adopt ANN (approximate nearest neighbor) search to accelerate the offline computation



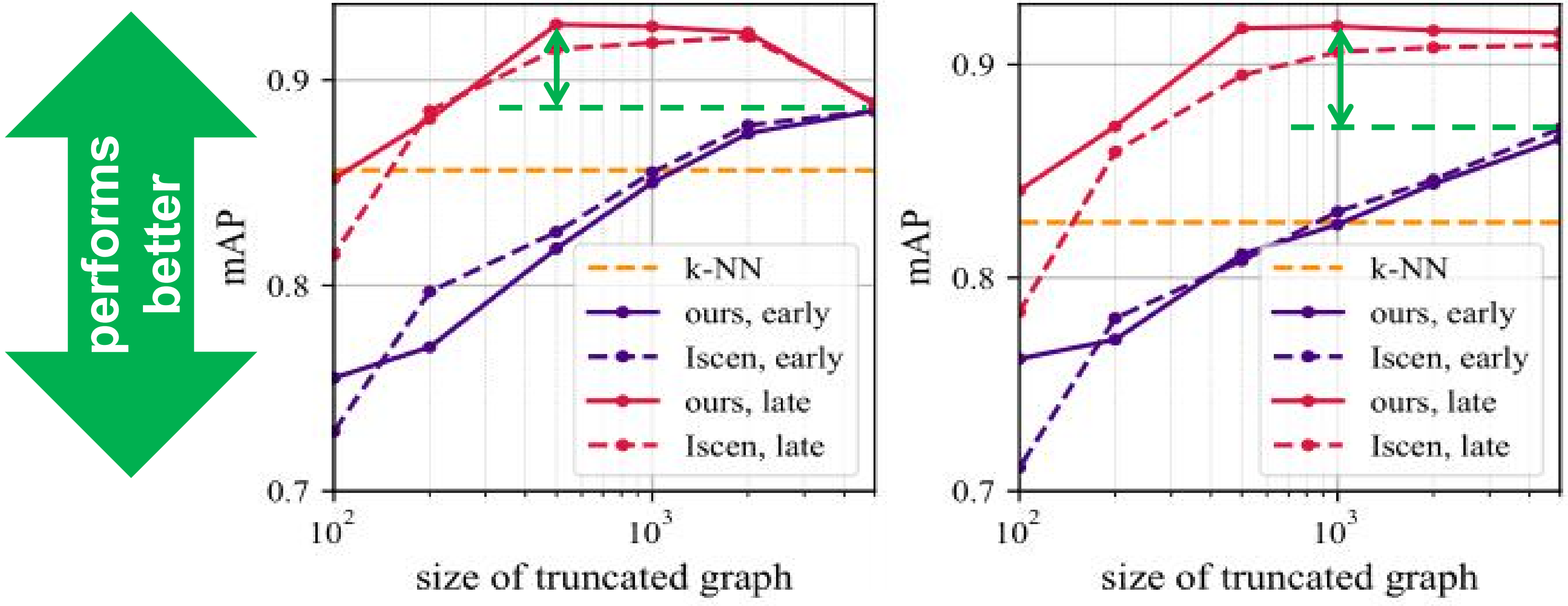
- Truncation only requires a coarse set of the top results
- Graph construction is negatively affected by even small differences in the scores

100k~ Images	Ours	Iscen's [CVPR2018]
Offline Time Cost	~6mins	a few hours

Fast spectral ranking for similarity search, Iscen et al., CVPR 2018

Influence of Subgraph Normalization

- We show the effectiveness of our proposed late truncation by using our proposed method and Iscen's method [CVPR2017]



(a) Oxford5k (b) Oxford105k

Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations, Iscen et al., CVPR 2017

Comparison to Other Methods

	Method	Feature	Global	Regional	Oxf5k	Oxf105k	Par6k	Par106k
w/o regional feature	k-NN search	R-MAC (VGG)	✓		79.5	72.1	84.5	77.1
	k-NN + AQE (Chum et al. 2007)		✓		85.4	79.7	88.4	83.5
	Iscen's diffusion (Iscen et al. 2017)		✓		85.7	82.7	94.1	92.5
	Proposed diffusion		✓		**89.7**	**86.8**	**94.7**	**92.9**
	k-NN search	R-MAC (ResNet)	✓		83.9	80.8	93.8	89.9
	k-NN + AQE (Chum et al. 2007)		✓		89.6	88.3	95.3	92.7
	Iscen's diffusion (Iscen et al. 2017)		✓		87.1	87.4	96.5	95.4
	Proposed diffusion		✓		**92.6**	**91.8**	**97.1**	**95.6**
w/ regional feature	R-match (Razavian et al. 2016)	R-MAC (VGG)		✓	81.5	76.5	86.1	79.9
	R-match + AQE (Chum et al. 2007)			✓	83.6	78.6	87.0	81.0
	Iscen's diffusion (Iscen et al. 2017)		✓	✓	93.2	90.3	**96.5**	92.6
	Proposed diffusion			✓	91.8	88.6	93.9	89.2
	Proposed diffusion w/ late fusion		✓	✓	**93.5**	**91.2**	96.1	**93.8**
	R-match (Razavian et al. 2016)	R-MAC (ResNet)		✓	88.1	85.7	94.9	91.3
	R-match + AQE (Chum et al. 2007)			✓	91.0	89.6	95.5	92.5
	Iscen's diffusion (Iscen et al. 2017)		✓	✓	95.8	94.2	96.9	95.3
	Proposed diffusion			✓	95.9	94.8	97.6	95.6
	Proposed diffusion w/ late fusion		✓	✓	**96.2**	**95.2**	**97.8**	**96.2**

Table 1: Performance comparison with the state of the art. We used R-MAC features extracted with VGG (Radenović, Tolias, and Chum 2016) and ResNet101 (Gordo et al. 2016).

Thank you!

Fan Yang

2018/11/13