







DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression

Jisung Park, ETH Zürich; Jeonggyun Kim, Yeseong Kim,
and Sungjin Lee, DGIST; Onur Mutlu, ETH Zürich

Background

➤ Data Reduction

- Three methods to save physical storage

Logic file			
Physical storage			
	Data Deduplication	Lossless Compression (zip,...)	Delta Compression

The higher the similarity between the two blocks, the better the delta compression.

Background

➤ Combination of 3 methods

1-3 : Deduplication

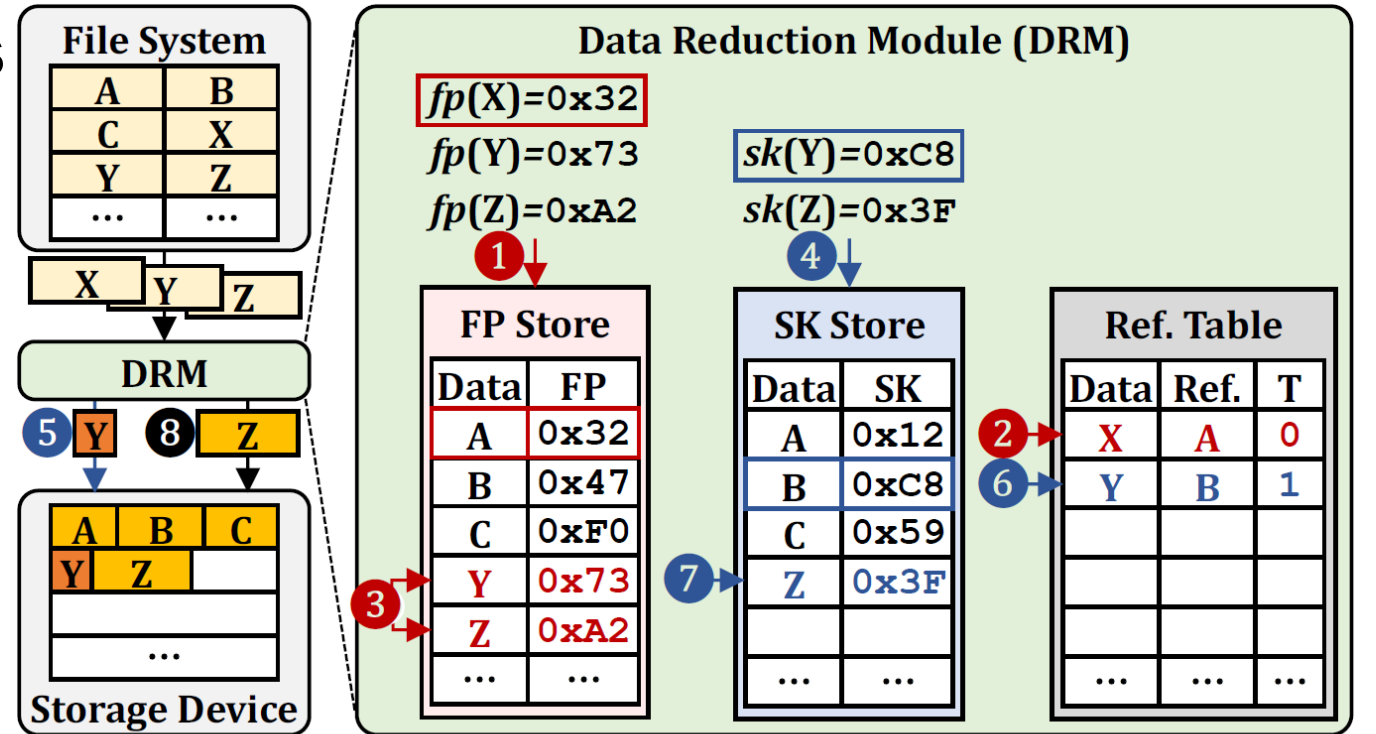
4-7 : Delta Compression

8 : Lossless Compression

fp(): Calculate the fingerprint of a chunk
 $fp(A) == fp(B) \iff A \text{ and } B$
 have the same content

sk(): Calculate the sketch of a chunk

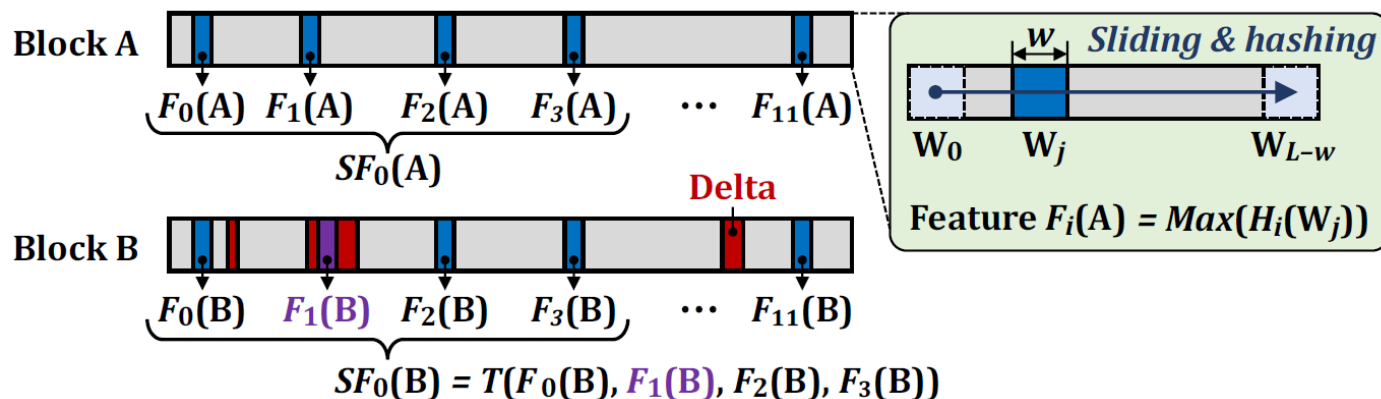
$sk(A) == sk(B) \iff A \text{ and } B$ are **similar enough**, system processes delta compression between A and B



Problem

➤ LSH(locality-sensitive hash) sketching

- Use a **sliding window** and **hash function $H(i)$** to calculate $L-w+1$ hashes, and take the maximum of these hashes as the feature **F_i**
- Calculate features **F_1, F_2, \dots, F_n** with the above algorithm using different hash functions
- Process these features to get the sketch (transpose algorithm): **$sk(A) = T(F_1, F_2, \dots, F_n)$**



If the location where the feature hash is generated is modified, the sketch is modified



Reduce the delta compression effect

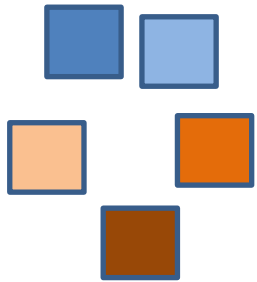
Problem

- **Theoretical optimal algorithm(brute-force search, BFS)**
 - Traverse over all existing chunks and select the chunk with the best effect
- **LSH vs. BFS**
 - FNR: **LSH = null** ,BFS != null
 - FPR: **LSH != BFS**
 - DRR: data-reduction ratio(normalized to BFS)

Workload		PC	Install	Update	Synth	Sensor	Web	Avg.
FNR		35.3%	51.8%	56.3%	75.5%	48.1%	5.5%	35.7%
FPR		21.1%	15.8%	11.3%	14.1%	47.3%	60.6%	23.1%
DRR	FN cases	0.474	0.488	0.578	0.639	0.567	0.539	0.562
	FP cases	0.621	0.608	0.644	0.683	0.798	0.674	0.669

Idea

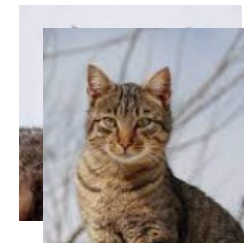
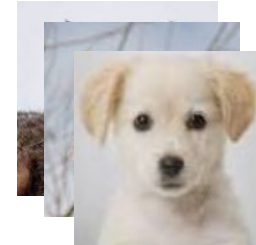
➤ Sketching and machine learning



Similarity-Based Classification
&
Key feature extraction



Sketching and delta compression

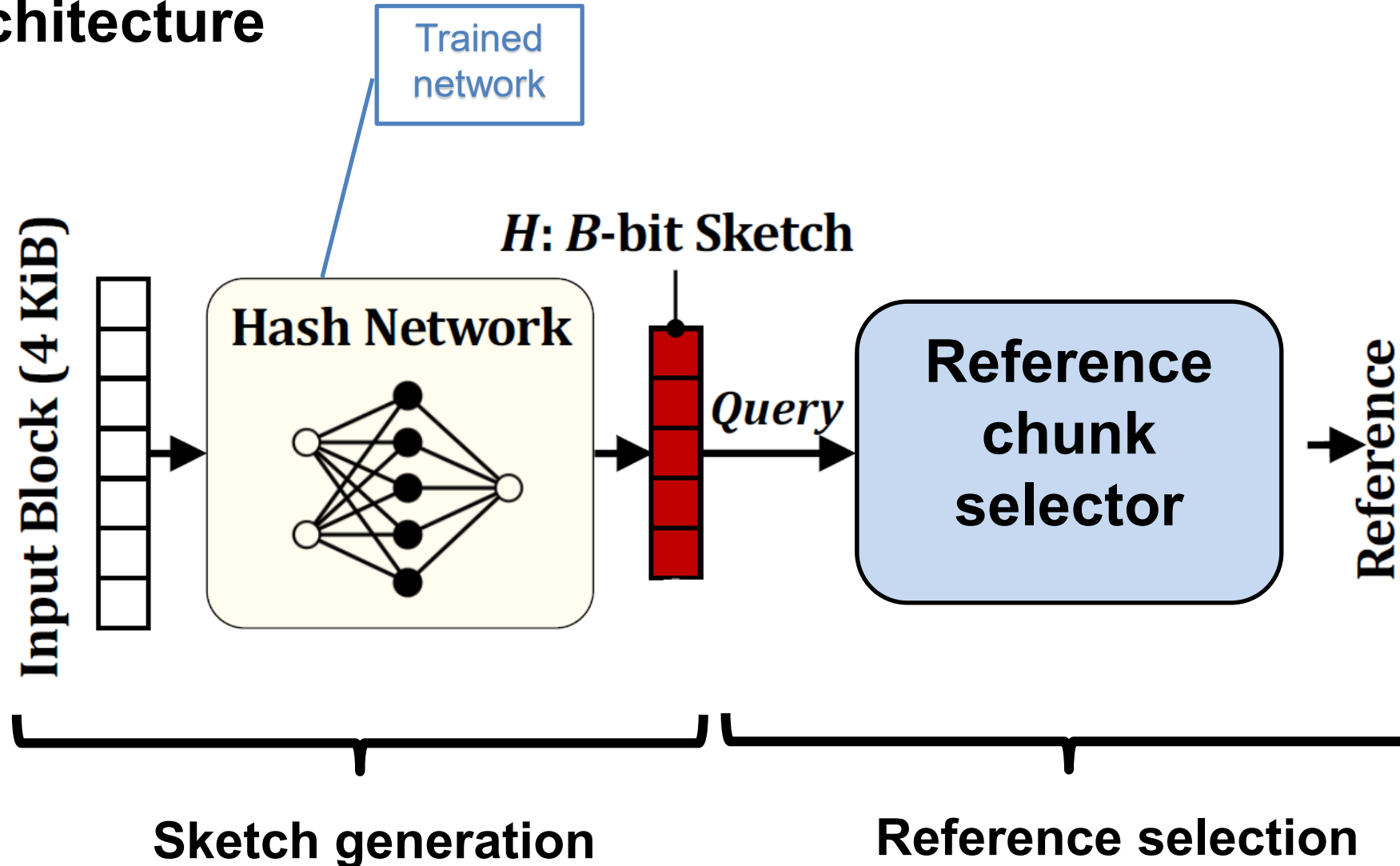


Cat and Dog Identification

IDEA: Use machine learning to generate hashes for chunks, so that the more similar two chunks are, the more similar their hashes(sketch) are.

Design

➤ Architecture



Design

➤ Sketch generation



Classification Model

- supervised learning
- Provide knowledge(W) to the hash network

Input: (chunk ,class)

Output = CNN(input)

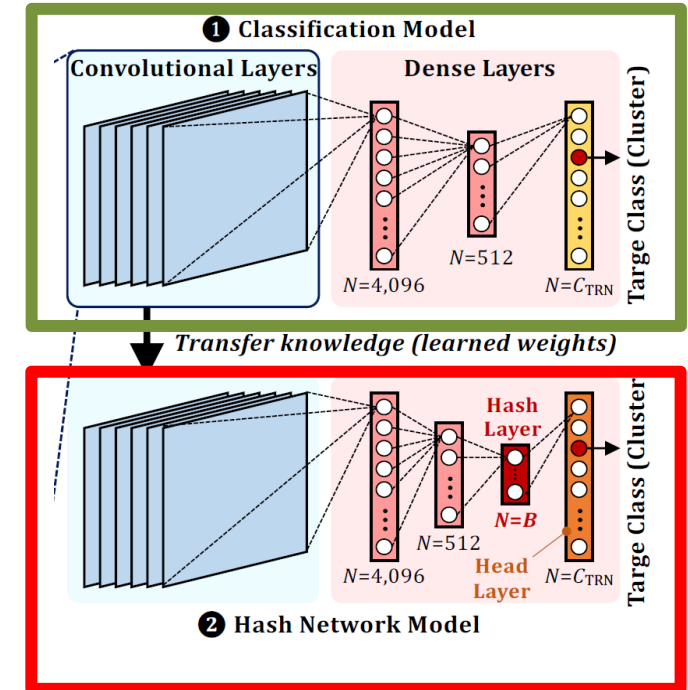
$W = \text{Backward}(W)$



Hash network model (based on greedyHash)

- supervised learning
- generate hash

The hash layer produces the sketch of the chunk



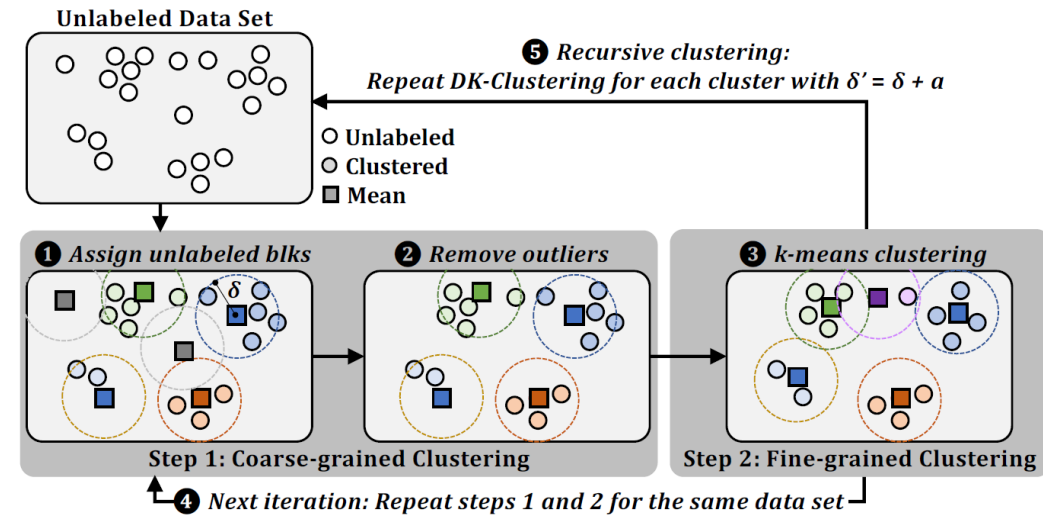
 (convolution + BN + Max Pooling) * 3

Feature extraction

Design

➤ Dynamic K-Means Clustering (for classification)

- Based on K-means
- two-level clustering
- Repeat clustering until convergence



0. Initialize a difference(or distance) threshold D

1. For any new chunk B , Exists a nearest cluster? **YES**: add B to this cluster **NO**: create a new cluster

2. Remove single chunk

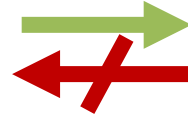
3. Adjusting the mean chunk and reassign each chunk

4. $D = D + d$, Repeat the above process until **convergence** $DRR(\text{cluster}) = DRR(\text{sub-clusters})$

Design

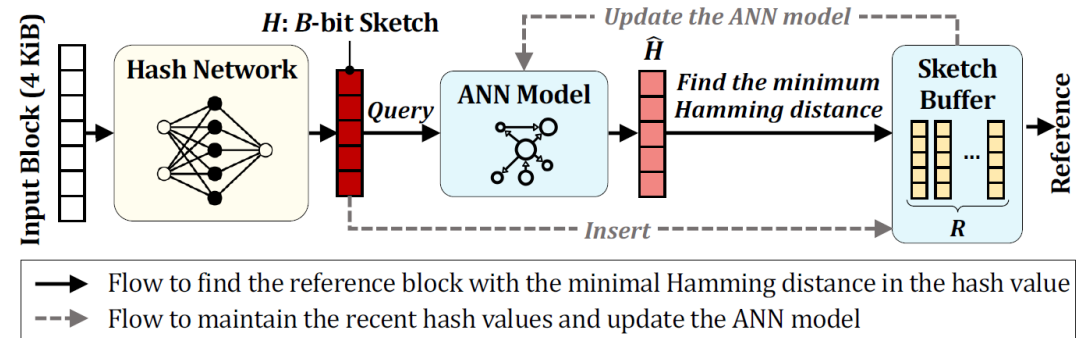
➤ Reference block query

- Problem $sk(A) == sk(B)$
- ANN



A and B are similar enough

- approximate nearest neighbor search in high dimensional vector data space
- NGT library
- Input: sketch H
- Output: a similar sketch H'
- high update overhead



Find a most similar hash from ANN and buffer, choose the better one

The new hash will not be added directly to the ANN, but to the buffer, refresh the buffer when it is full

➤ Hyper-Parameter Exploration

Adjust the hyper-parameters(layers number, etc) of the network and select the optimal one

Evaluation

➤ Platform

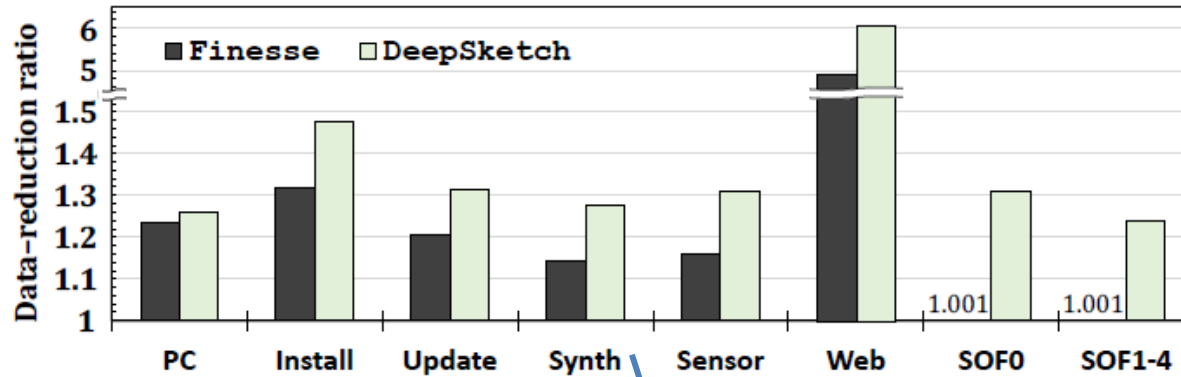
- Hardware: Xeon 4110 + 128GB DDR4 DRAM + 8 * 1TB SSDs + RTX 2080
- Software: Storage system with combination of three data reduction schemes
 - Fingerprint Algorithm: MD5 Delta Algorithm: Xdelta
 - Sketch algorithm: DeepSketch vs. Finesse

• Workloads

Workload	Description	Size	Dedup. ratio	Comp. ratio
PC	General Ubuntu PC usage	1.57 GB	1.381	2.209
Install	Installing & executing programs	8.83 GB	1.309	2.45
Update	Updating & downloading SW packages	3.73 GB	1.249	2.116
Synth	Synthesizing hardware modules	653 MB	1.898	2.083
Sensor	Sensor data in semiconductor fabrication	91.2 MB	1.269	12.38
Web	Web page caching	959 MB	1.9	6.84
SOF0	Storing Stack Overflow database [33] as of 2010 (SOF0) and 2013 (SOF1–4)	8.98 GB	1.007	2.088
SOF1		13.6 GB	1.01	1.997
SOF2		13.6 GB	1.01	1.996
SOF3		13.6 GB	1.01	1.997
SOF4		13.6 GB	1.01	1.996

Evaluation

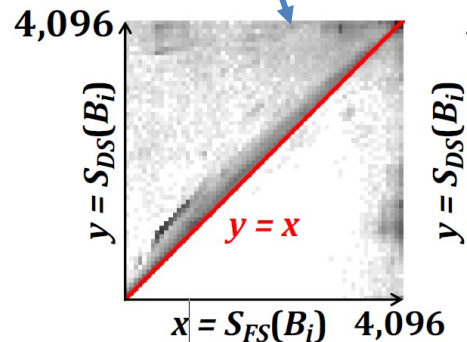
➤ Overall Data Reduction



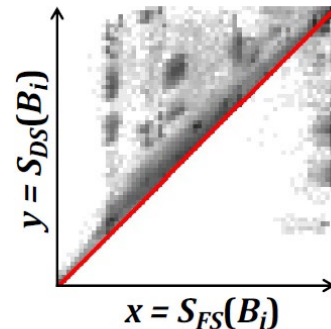
Ratio(dedup+ delta compression + lossless compression)

Data-reduction ratio =

Ratio(dedup+ lossless compression)



(a) PC.



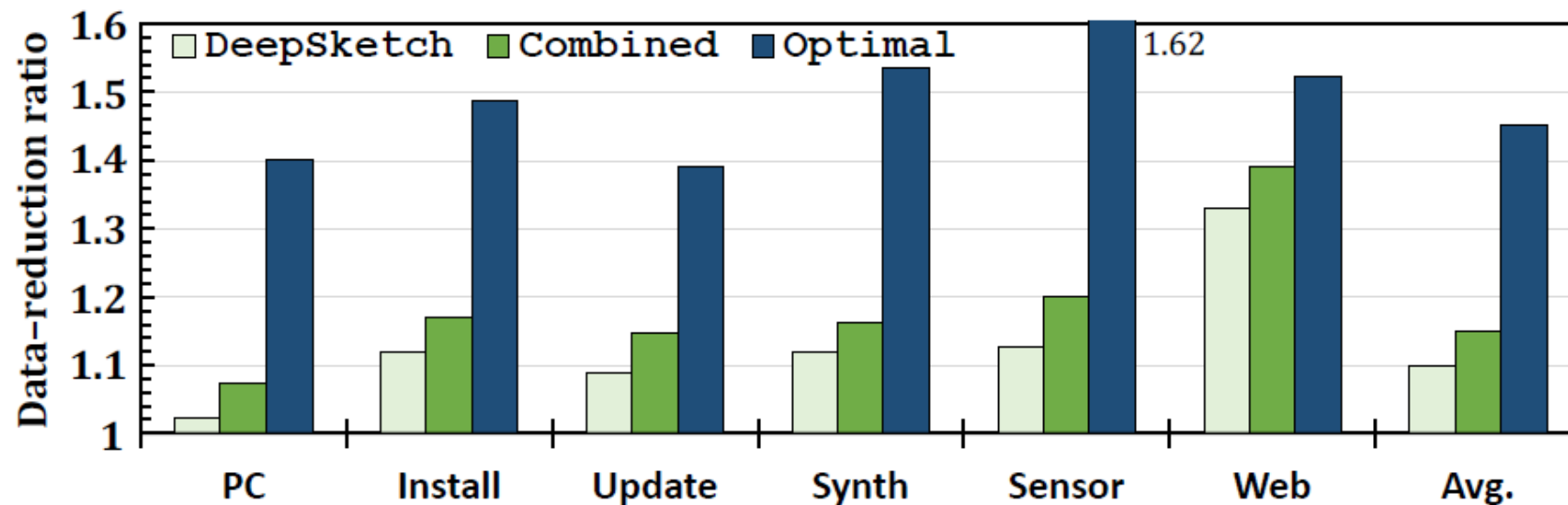
(d) Synth.

Point (x,y) for a same chunk
X: saved bytes by DeepSketch
Y: saved bytes by Finesse

Evaluation

➤ DeepSketch vs Combined vs Optimal

- Combined: choose the better one between Deepsketch and Finesse
- Optimal: brute-force search(Theoretical Optimality Ignoring Overhead)

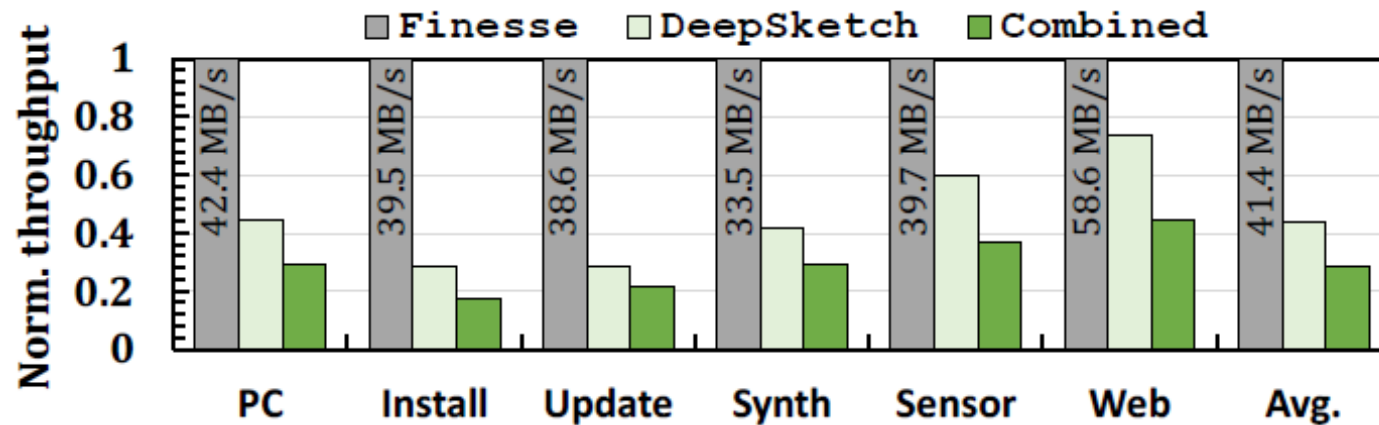


Combining the two methods produces better results

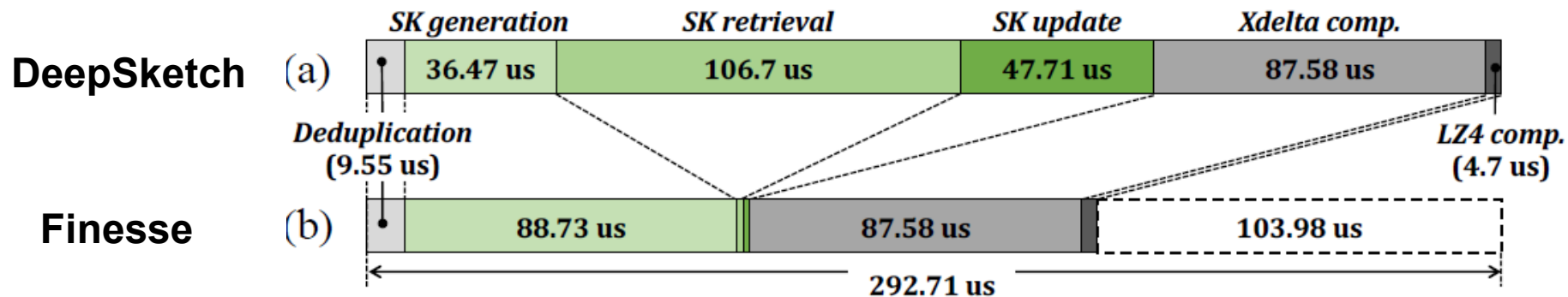
There is still a certain gap between the existing methods and the theoretical optimal

Evaluation

➤ Throughput



➤ Average latency



Conclusion

➤ Problem

- Traditional sketch generation methods do not work well in delta compression

➤ Idea

- Neural networks(NN) can perform feature extraction, so they can also be used in sketch generation

Conclusion

