iCACHE: An Importance-Sampling-Informed Cache for Accelerating I/O-Bound DNN Model Training

Weijian Chen[*], Shuibing He[*], Yaowen Xu[*], Xuechen Zhang[#], Siling Yang[*], Shuang Hu[*], Xian-He Sun[$], Gang Chen[*]

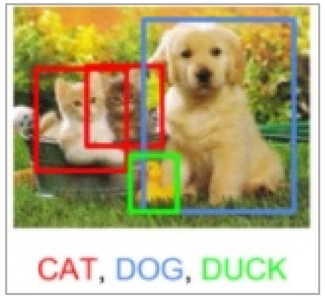
[*] 浙江大学 Zhejiang University

[#] Washington State University

[$] Illinois Tech

Deep Neural Network (DNN) Training

- DNN has been applied in a range of fields



Google Translate

- DNN training pipeline

Data Loading → 1st epoch: batch 1, batch 2, … …; 2ed epoch: … … → Data Preprocessing → Forward and Backward Computation

Deep Neural Network (DNN) Training

➢ **Characteristics of each stage**

Data Loading

- **Poor temporal locality.** (Access each data item only once in each epoch)
- **Poor spatial locality.** (Fully random access)

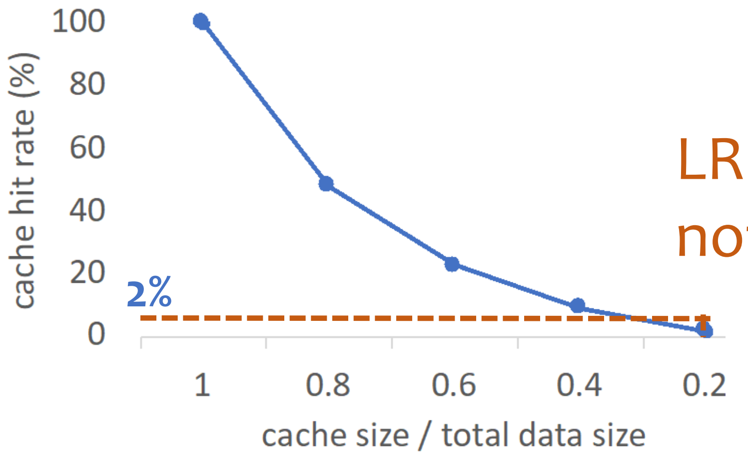
Data Preprocessing

- Operators are **usually lightweight**

Forward and Backward Computation

- DL accelerators are **getting faster**: GPU V100, A100, TPU, ASIC…

➢ **When memory is insufficient for growing dataset**



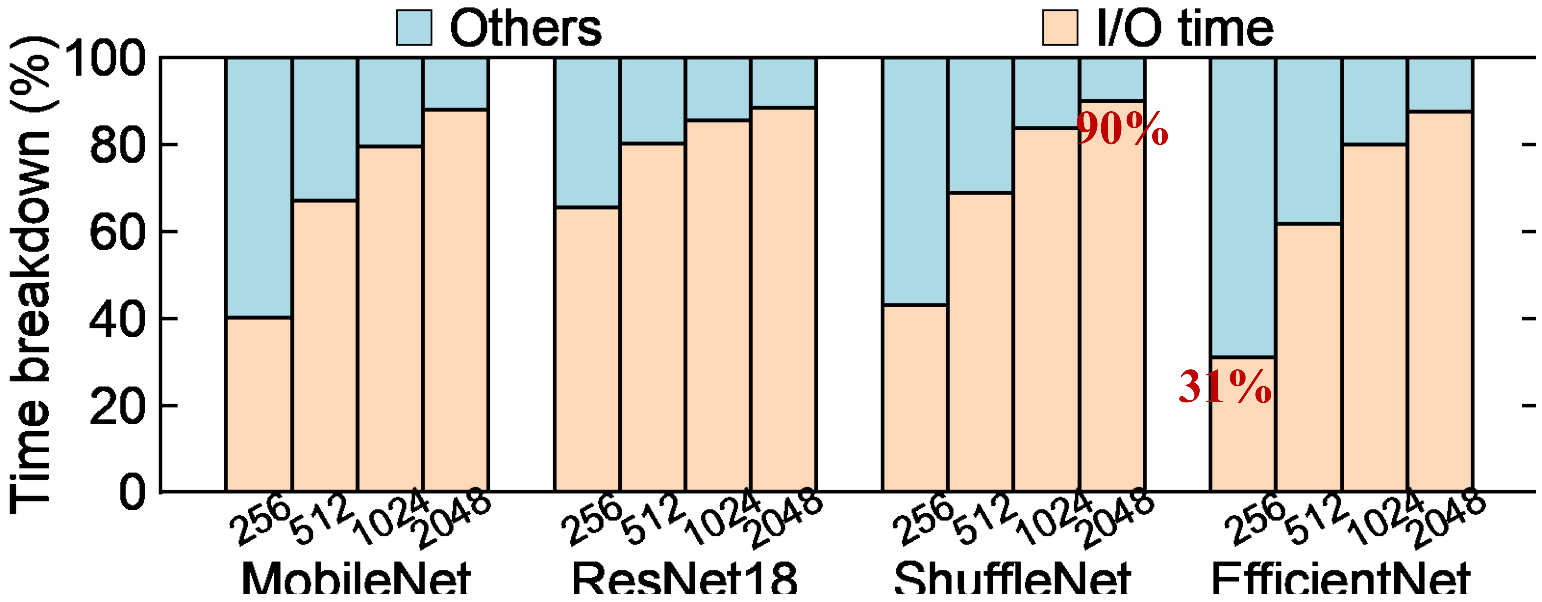
LRU-based cache is not practice.

Data Loading is becoming the training bottleneck ! ☹

Deep Neural Network (DNN) Training

➢ Common techniques to accelerate DNN training

- Data prefetching
- Traditional data caching
- Batch size adjustment
- Multi-GPU training



These widely used techniques are inefficient for I/O-bound DNN tasks.

Related Work: DNN Cache Optimization

- **Explore data locality in more depth.**
  - between **epochs** → CoorDL [VLDB' 21]: A static cache.
  - between **multiple jobs** → OneAccess [HotCloud' 19], et al.: Sharing cached data.

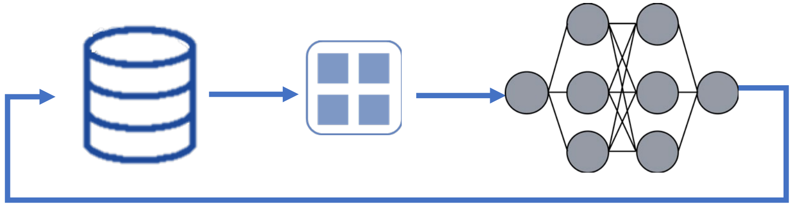
- **Exploit data substitutability of DNN training.**
  - DeepIO [MASCOTS' 18], Quiver[FAST' 19]: Replace cache missed data with data in the cache

These techniques are not sufficient when data size is huge.
DNN applications in all of these work need to
fetch all data from cache/storage for each epoch training.

Opportunity from Importance Sampling

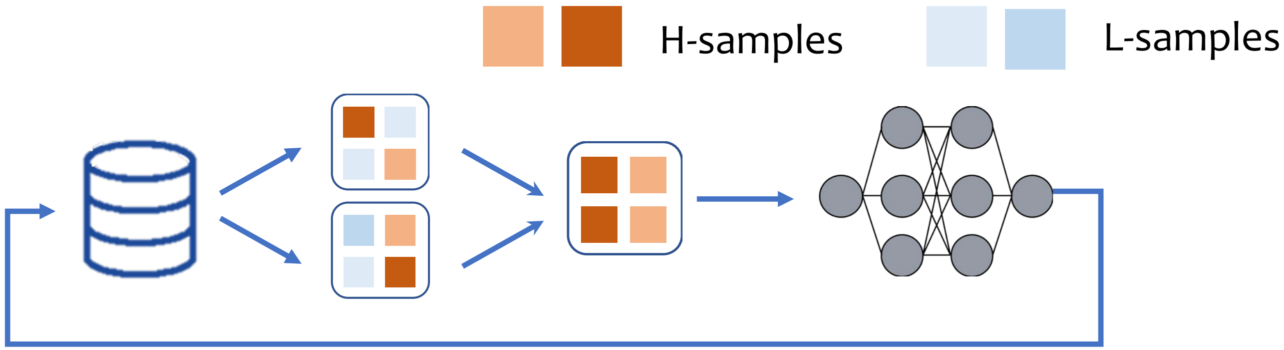
➢ For each epoch training:

a. Default DNN training:



Original accuracy

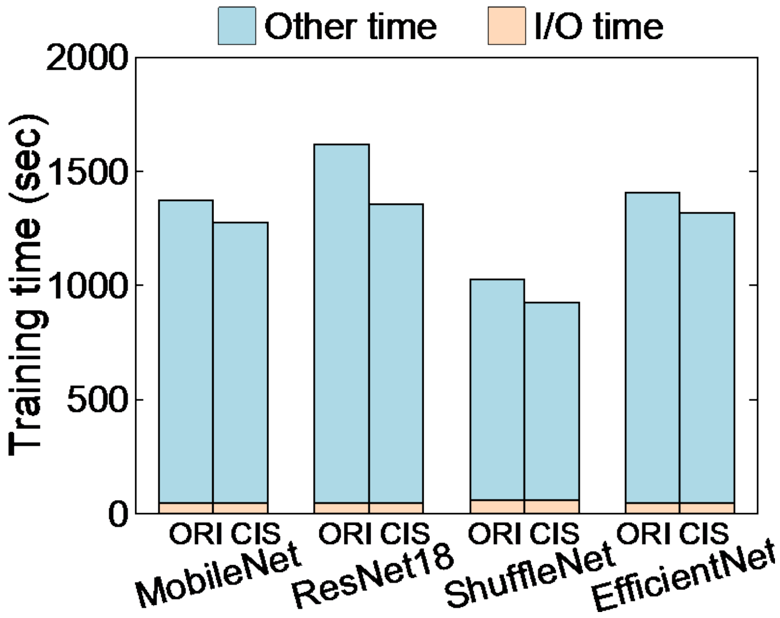
b. Importance sampling-based DNN training:



Comparable accuracy

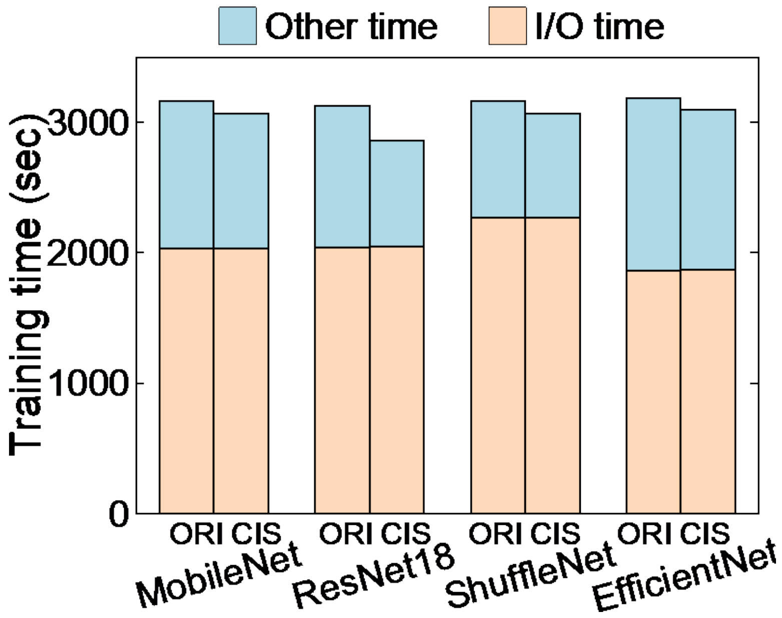
Opportunity from Importance Sampling

- However, existing IS algorithms are designed for computing-bound tasks (We name them CIS).



a. Computing-bound training (cache size 100%)

CIS Speed up training 1.3x ☺



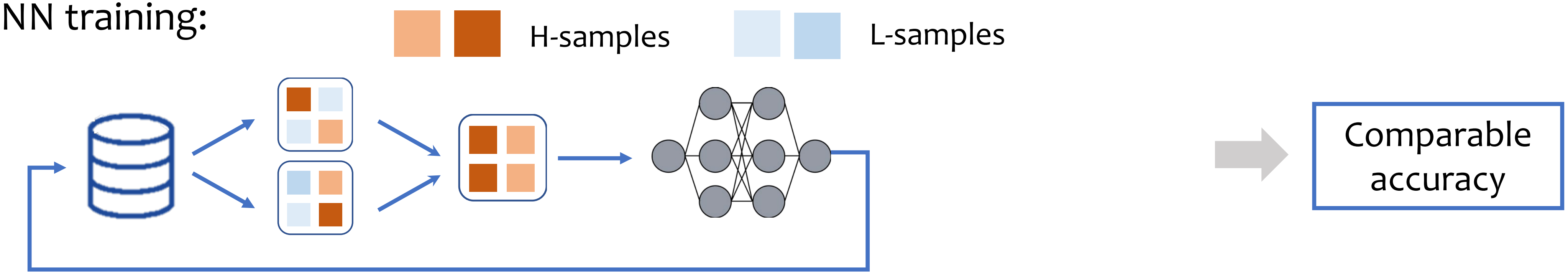
b. I/O-bound training (cache size 20%)

CIS Speed up training 1.02x ☹

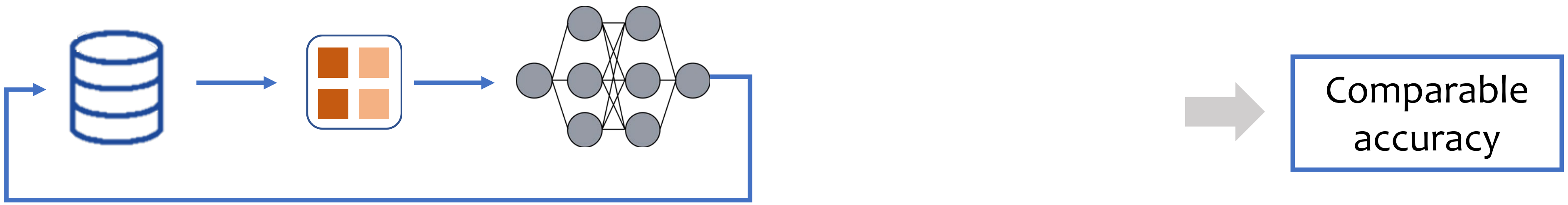
I/O-oriented Importance Sampling Algorithm

- Inspired by CIS, we propose I/O-oriented importance sampling (IIS).

b. CIS DNN training:



c. IIS DNN training:



The necessity of re-design cache optimization

➢ **It seems promising to combine IIS and cache optimization…**

IIS	Reduce **# of data items loaded**	Mitigate I/O bottleneck of DNN training
+		
Cache optimization	Reduce data loaded **from storage**	

IIS	select data items based on their **impact on model accuracy**
OS cache Quiver CoorDL… **Existing DNN cache system**	cache replacement based on **locality**

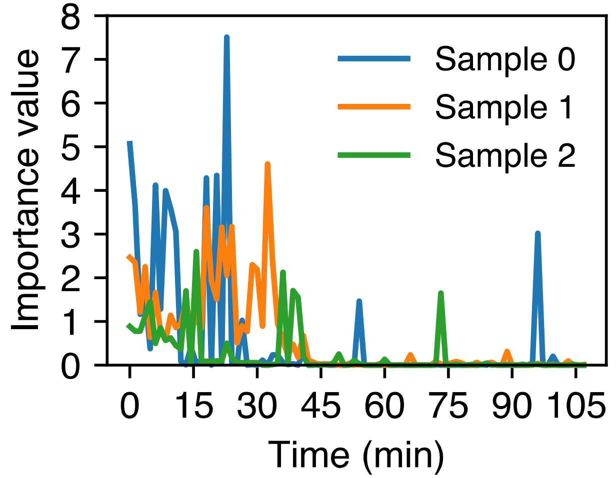
Unmatched

It is necessary to re-design cache management considering importance sampling.

Challenges

➢ **Intuitively, caching H-samples as much as possible. However…**

1. Importance value of a specific data item fluctuates during training.



❓ How to keep a maximum number of H-samples in the cache when the importance values changes to achieve high cache hit rate ?

2. Cache capacity is limited and L-samples are likely to be cache missed.

❓ How to deal with poor I/O efficiency when accessing L-samples ?

3. The cache needs to serve multiple jobs.

❓ How to coordinate samples cached between multiple jobs ?

Outline

- Background & Motivation
- **Design of iCACHE: an cache system to accelerate DNN training**
- Implementation & Evaluation
- Summary & Conclusion

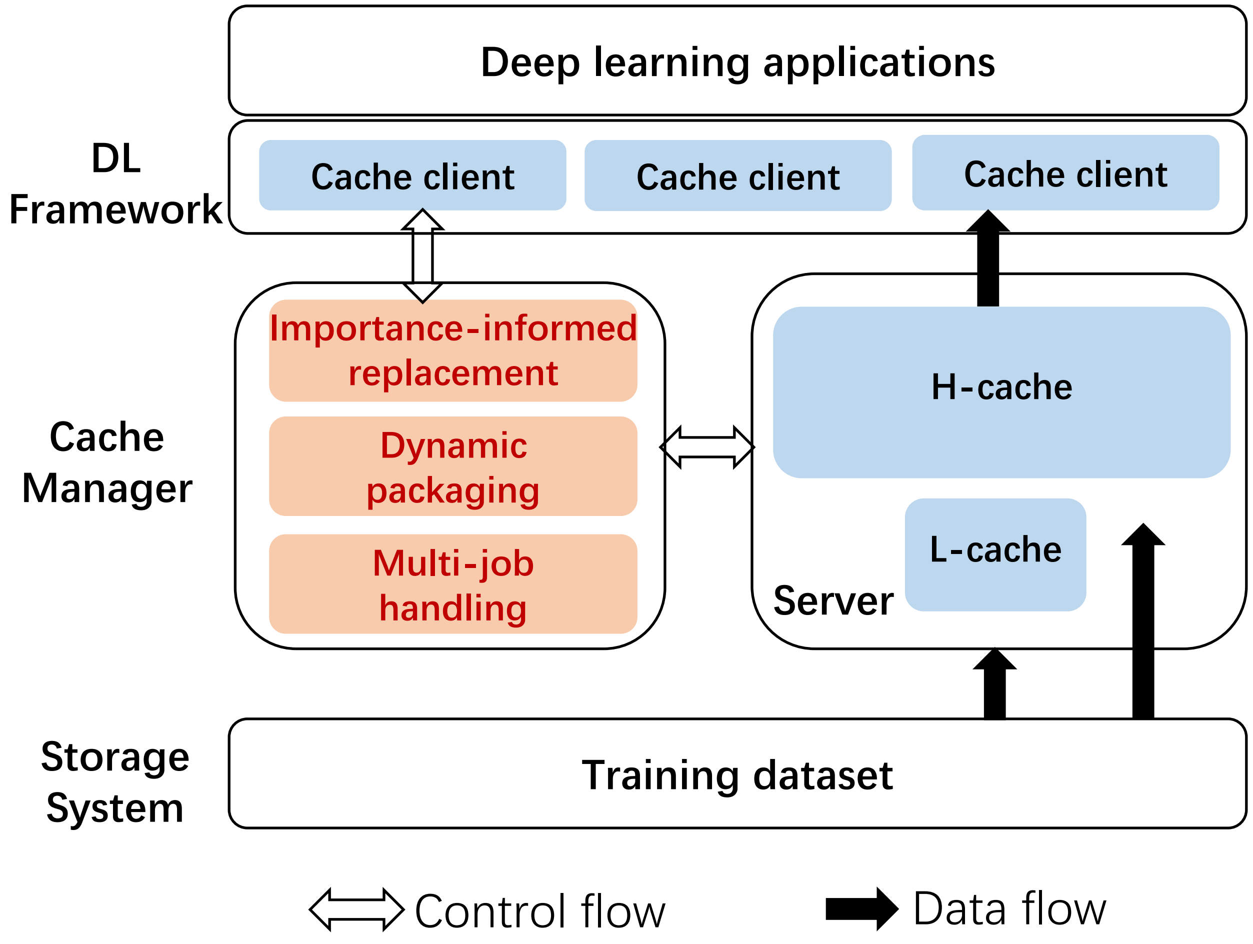
iCACHE Architecture

Cache clients

- ➢ Request data items based on Importance sampling algorithm
- ➢ Maintains each data item's importance value

Cache server

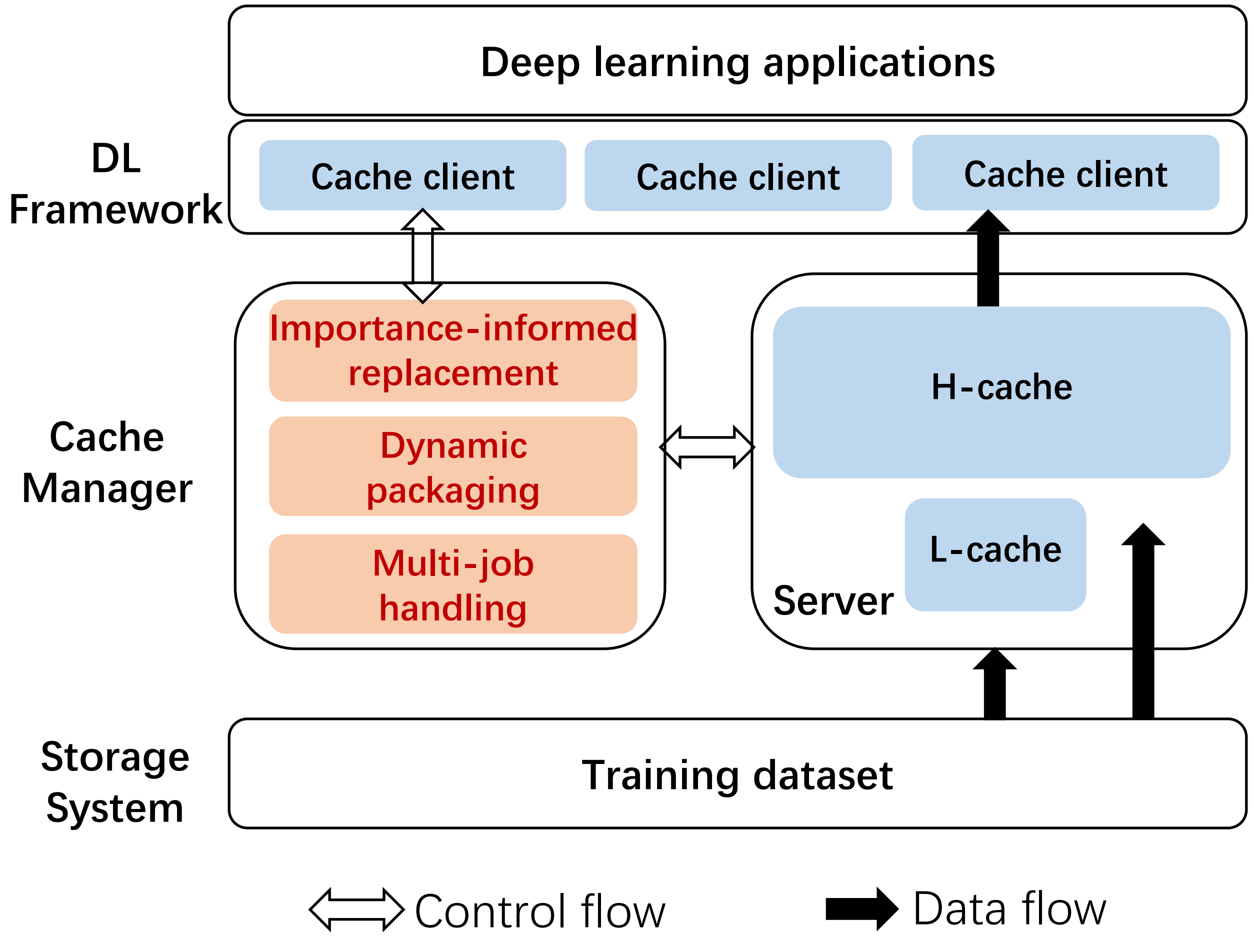
- ➢ User-level cache
- ➢ H-cache: cache H-samples
- ➢ L-cache: cache L-samples



iCACHE Architecture

Cache Manager (Key ideas)

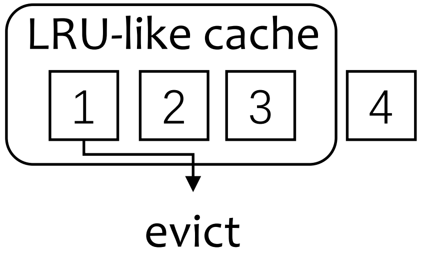
- Importance-informed cache replacement
- Dynamic packaging to server L-sample requests
- Multi-job handling module



1. Importance-Informed Cache Algorithm

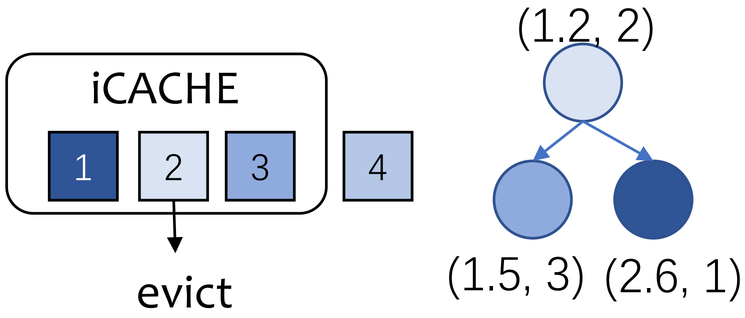
- Aims to serve H-sample requests and improve H-cache hit ratio.

➢ **Use a small-top-heap for cache replacement.**
- O(1) to find the data item with smallest importance value.



(a) LRU-like algorithm

➢ **Tracks samples' importance value and refresh.**



(b) iCACHE

➢ **Build shadow-heap to asynchronously update importance value.**
- The additional space overhead is less than 0.5% of the cache size.

2. Dynamic Packaging

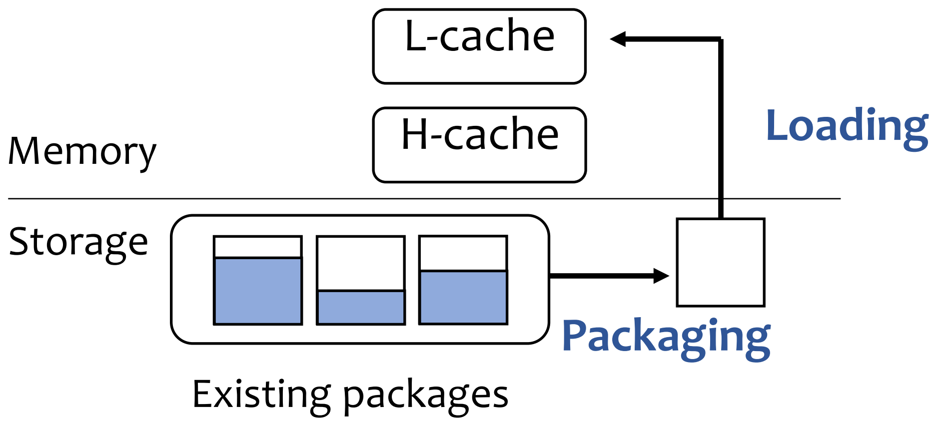
- Aims to serve L-sample requests.

➢ Key idea:

- **apply substitutability on L-samples** has minor impact on model accuracy while reducing data fetch time.

TABLE III
MODEL ACCURACY ON CIFAR10.

Models	Top-1 Acc.(%)			Top-5 Acc.(%)		
	Def	ST_{HC}	ST_{LC}	Def	ST_{HC}	ST_{LC}
ResNet18	92.70	91.89	92.14	99.81	99.77	99.80
ShuffleNet	87.76	86.73	86.96	99.59	99.52	99.57



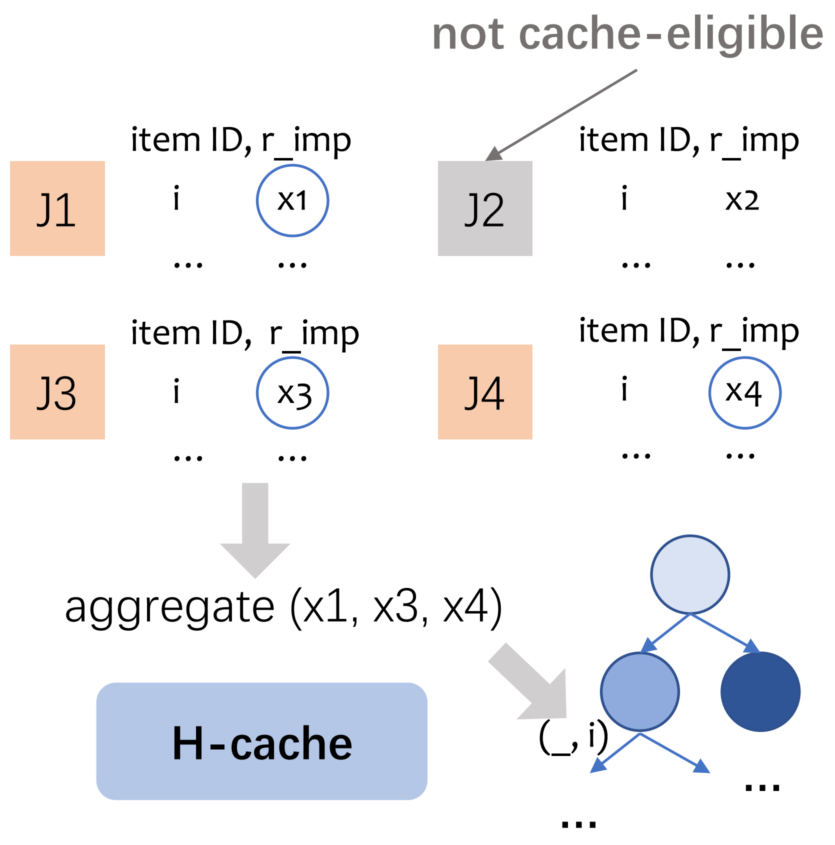
* The white area represents L-samples; the blue area denotes H-samples.

➢ **Two asynchronous concurrent threads**

- three benefits:
  a) Alleviate random small I/O
  b) Improving effective storage bandwidth
  c) Increase randomness of training sequence

3. Multi-Job Handling

- One data item may receive different importance value
  a) Different model has different fit capacity
  b) Importance values tend to decrease during training



1. Evaluate the cost-effectiveness of caching for each job by profiling

2. Adjust importance value:

- use relative importance value
- calculate aggregated importance value

Implementation

- Cache client (2000 LOC)
  - New *Dataset* interface of PyTorch

- Cache server (3500 LOC)
  - Key-value structure in Golang
  - dynamic packaging & multi-job handling

- Easy to deploy iCACHE.

- We also extend iCACHE to the **distributed version.**

PyTorch (1.8.0)

Golang

Experimental Setup

- System configuration

CPU	2× AMD EPYC 7742 CPUs
GPU	8× NVIDIA A100
Dataset store	OrangeFS (Remote PFS), 10Gbps Ethernet.

- Workloads and datasets

Datasets	CIFAR10, ImageNet-1k
DNN Models	ShuffleNet, ResNet18, MobileNet, ResNet50, VGG11, MnasNet, SqueezeNet, and DenseNet121.

- Compared systems

Default	PyTorch + LRU user-level cache
Base	CIS + LRU user-level cache
Quiver [FAST'20]	Uses sample substitutability & Coordinated eviction
CoorDL [VLDB'21]	Does not evict already cached data
iLFU	IIS + LFU to compare different cache strategies

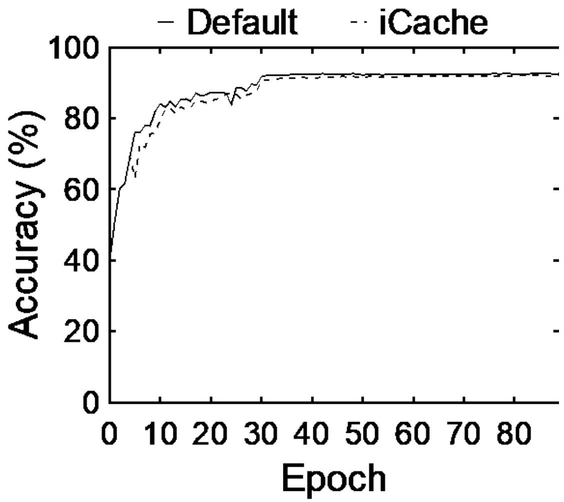
State-of-the-art

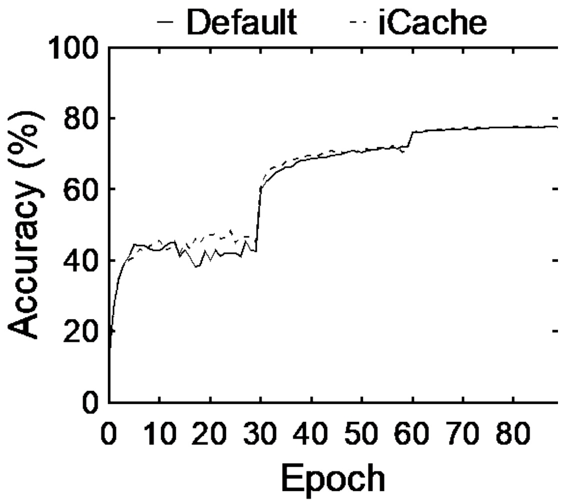
- Default cache size: 20% of total training dataset

Accuracy

TABLE I
MODEL ACCURACY ON CIFAR10.

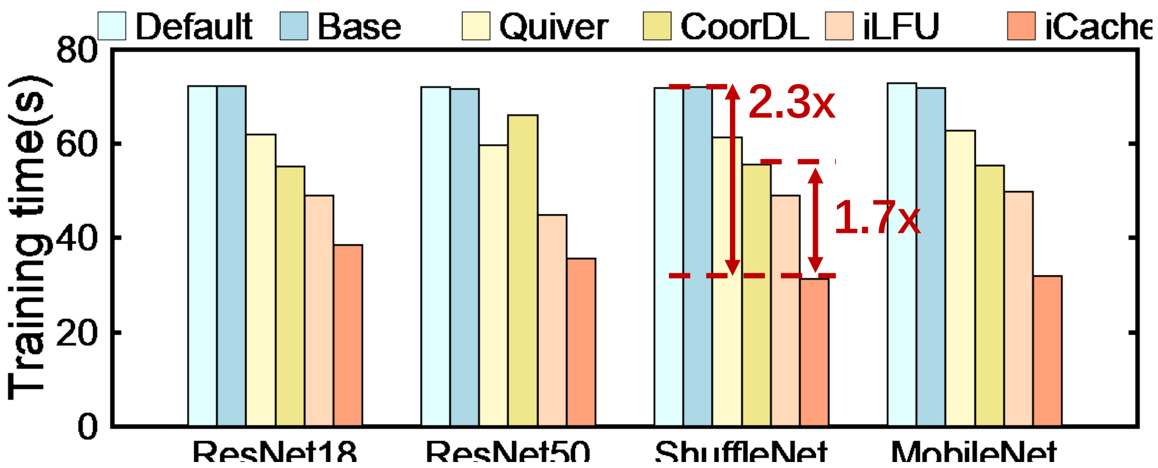
Models	Top-1 Acc.(%)		Top-5 Acc.(%)	
	Default	iCACHE	Default	iCACHE
ShuffleNet	87.76	86.96	99.59	99.57
ResNet18	92.70	92.14	99.81	99.80
MobileNet	92.37	92.01	99.87	99.77
ResNet50	89.91	89.36	99.68	99.69

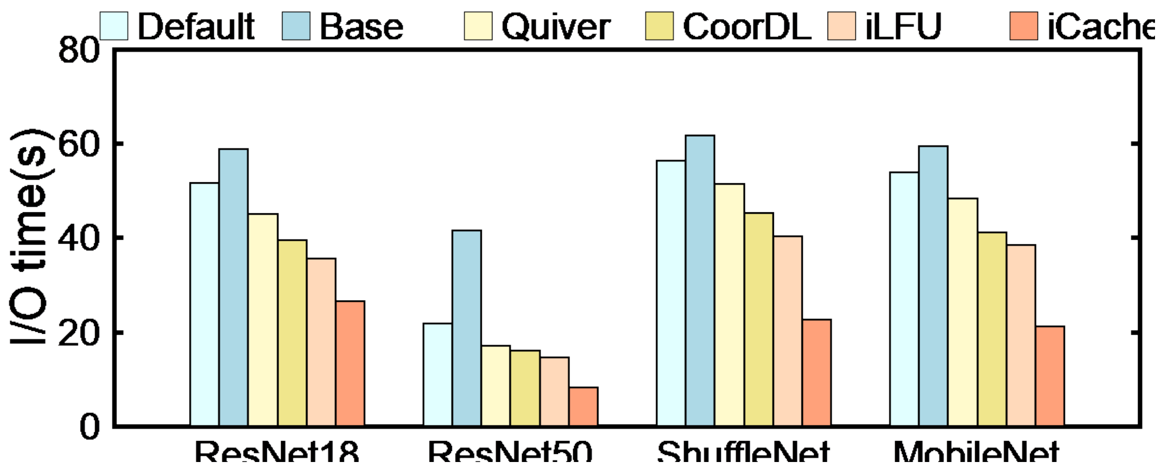




Comparable accuracy is achieved on different models and datasets

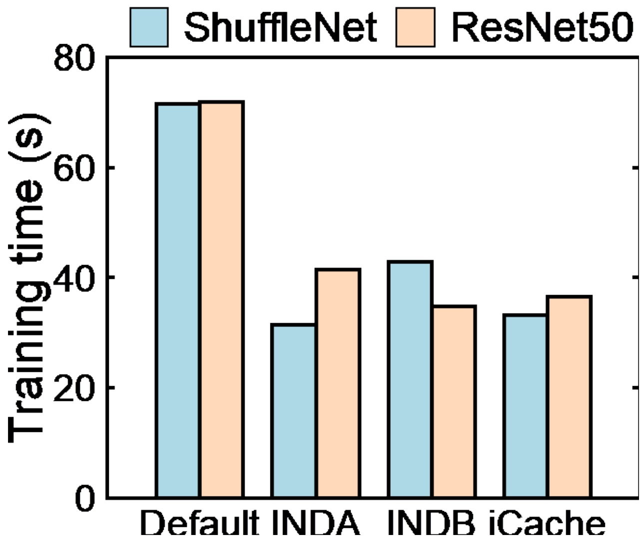
Overall Performance

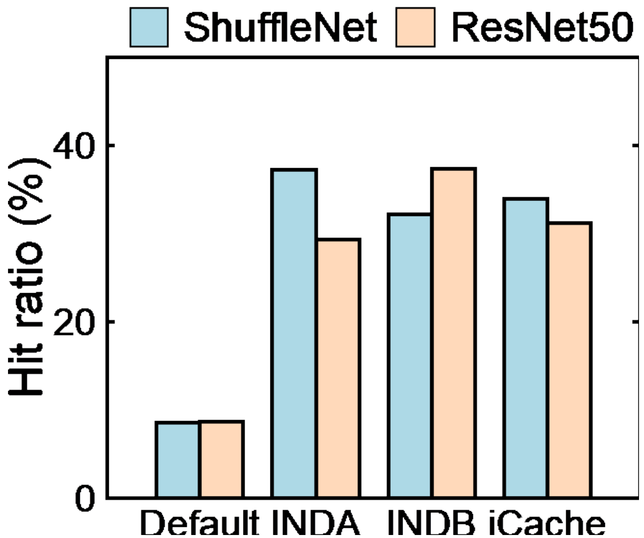




iCACHE speeds up the overall training time by 1.7x compared to SOTA, and 2.3x to Base. Compared to Default, iCACHE reduces the I/O time by 2.4x on average.

Multi-job Training Performance

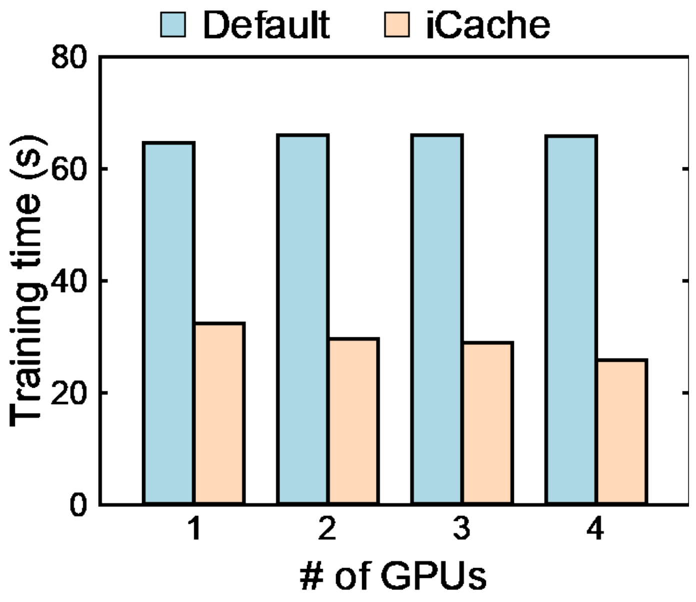




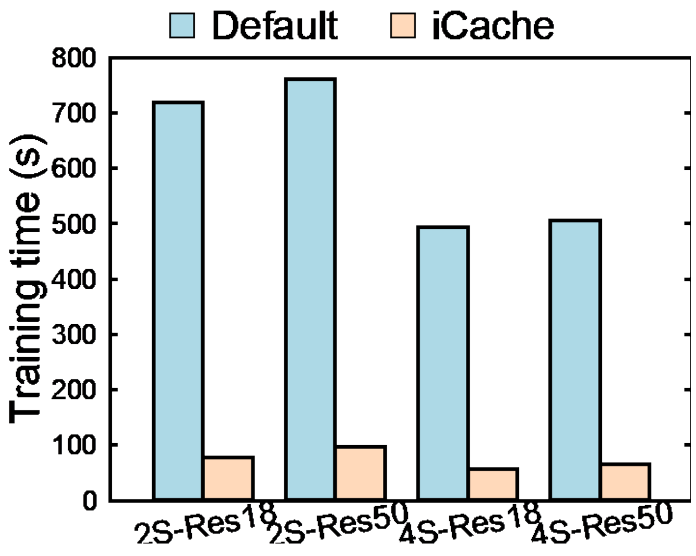
INDA: Manage cache simply based on importance value given by ShuffleNet.
INDB: Manage cache simply based on importance value given by ResNet50.

iCACHE speeds up the jobs completion time
in multi-job scenario by up to 1.2x.

Multi-GPU and multi-node training



(a) Multi-GPU training



(b) Multi-node training

iCACHE always performs better than Default on Multi-GPU training.
iCACHE speeds up at least 8.6x and 7.6x under 2-server and 4-server configurations.

More evaluations: checkout our paper.

Summary & Conclusion

- **Problem**
  - I/O is becoming the bottleneck in DNN training
- **Key idea**
  - Introduce I/O-oriented importance sampling (IIS) and optimize cache management considering importance values.
- **Techniques in iCACHE**
  - Importance-Informed Cache Algorithm
  - Dynamic Packaging
  - Multi-Job Handling
- **Results**
  - iCACHE alleviates I/O bottleneck of DNN training in various training scenarios.
  - iCACHE outperforms state-of-the-arts while maintaining comparable accuracy.

Thanks & QA

iCACHE: An Importance-Sampling-Informed Cache for Accelerating I/O-Bound DNN Model Training

浙江大学
Zhejiang University

WASHINGTON STATE UNIVERSITY

ILLINOIS TECH

Contact Information: weijianchen@zju.edu.cn