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*

* Zhejiang University, # Washington State University, $ Illinois Tech
Deep Neural Network (DNN) Training

- DNN has been applied in a range of fields

- DNN training pipeline

```
1st epoch
Data Loading
batch 1
batch 2
...
...
2nd 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!
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:

b. Importance sampling-based DNN training:
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 😞
Inspired by CIS, we propose I/O-oriented importance sampling (IIS).

b. CIS DNN training:

```
| Database | H-samples | L-samples | DNN | Comparable accuracy |
```

c. IIS DNN training:

```
| Database | H-samples | DNN | Comparable accuracy |
```
The necessity of re-design cache optimization

- It seems promising to combine IIS and cache optimization...

IIS + Cache optimization

**IIS**
- Reduce # of data items loaded
- Reduce data loaded from storage

Mitigate I/O bottleneck of DNN training

**OS cache**
- Quiver
- CoorDL...
- Existing DNN cache system

select data items based on their **impact on model accuracy**

**Unmatched**

cache replacement based on **locality**

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
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.

- **Tracks samples’ importance value and refresh.**

- **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.

- Two asynchronous concurrent threads
- Three benefits:
  a) Alleviate random small I/O
  b) Improving effective storage bandwidth
  c) Increase randomness of training sequence

---

### TABLE III

Model accuracy on CIFAR10.

<table>
<thead>
<tr>
<th>Models</th>
<th>Top-1 Acc. (%)</th>
<th>Top-5 Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Def</td>
<td>ST(_{HC})</td>
</tr>
<tr>
<td>ResNet18</td>
<td>92.70</td>
<td>91.89</td>
</tr>
<tr>
<td>ShuffleNet</td>
<td>87.76</td>
<td>86.73</td>
</tr>
</tbody>
</table>

---

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

- **System configuration**
  - CPU: 2x AMD EPYC 7742 CPUs
  - GPU: 8x NVIDIA A100
  - Dataset store: OrangeFS (Remote PFS), 10Gbps Ethernet.

- **Workloads and datasets**
  - Datasets: CIFAR10, ImageNet-1k

- **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

- Default cache size: 20% of total training dataset

State-of-the-art
Accuracy

Comparable accuracy is achieved on different models and datasets

### TABLE I
**Model Accuracy on CIFAR10.**

<table>
<thead>
<tr>
<th>Models</th>
<th>Top-1 Acc. (%)</th>
<th>Top-5 Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default</td>
<td>iCACHE</td>
</tr>
<tr>
<td>ShuffleNet</td>
<td>87.76</td>
<td>86.96</td>
</tr>
<tr>
<td>ResNet18</td>
<td>92.70</td>
<td>92.14</td>
</tr>
<tr>
<td>MobileNet</td>
<td>92.37</td>
<td>92.01</td>
</tr>
<tr>
<td>ResNet50</td>
<td>89.91</td>
<td>89.36</td>
</tr>
</tbody>
</table>

![Accuracy Graphs](image1.png)

![Accuracy Graphs](image2.png)
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**

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

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

INDB: Manage cache simply based on importance value given by ResNet50.
Multi-GPU and multi-node training

(a) Multi-GPU training

(b) Multi-node training

iCACHÉ always performs better than Default on Multi-GPU training. iCACHÉ 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.
iCACHE: An Importance-Sampling-Informed Cache for Accelerating I/O-Bound DNN Model Training