Face Recognition

Object Detection

Self-driving car

Language translation
At Their Core: ML Training and Serving

Training Data → ML Training → ML Model → ML Serving → Serving Data

Key Objectives: Low Latency and Low Cost

Image classification
Many ML Data are Highly Distributed and Rapidly Growing
Challenge: ML Serving on Rapidly-Growing Data

**High Cost** to Process Data at Ingest Time

**Long Latency** to Process Data at Query Time

ML Serving System

Traffic cameras in a city

Find video frames with trucks

Computation Challenge
Challenge: ML Training on Highly-Distributed Data

What happens to ML if training data partitions are not IID (independent and identically distributed)?

Communication Challenge

Statistical Challenge
Thesis Statement

The latency and cost of ML training and serving on highly-distributed and rapidly-growing data can be improved by one to two orders of magnitude by designing ML systems that exploit the characteristics of ML algorithms, ML model structures, and application data.
Overview of Our Approach

ML Serving over Large, Rapidly-Growing Datasets [OSDI’18]
- Address the computation challenge

ML Training over Geo-Distributed Data [NSDI’17]
- Address the communication challenge

Understanding The Non-IID Data Partition Problem for Decentralized ML
- Address the statistical challenge
Overview of Our Approach

ML Serving over Large, Rapidly-Growing Datasets [OSDI’18]
- Address the computation challenge

ML Training over Geo-Distributed Data [NSDI’17]
- Address the communication challenge

Understanding The Non-IID Data Partition Problem for Decentralized ML
- Address the statistical challenge
Video Data are Rapidly Growing

Massive video recordings are happening everywhere
Querying Objects in Videos using ML Serving

- Find all trucks among traffic videos in a city last week
- Find all people in garage videos in a company last night

→ Query execution requires running detector & classifier CNNs

→ It is slow and costly on massive videos
Ingest Time Analysis: Too Costly

• Analyzing live videos at ingest time can make query fast
  • But it is costly
  • Potentially wasteful (ingest all garage cameras vs. query one)

$380/month/stream
Query Time Analysis: Too Slow

• Analyzing videos at query time can save cost
  • Frame down-sampling / skipping
  • CNN specialization / cascading
  • But it still very slow (5 hr for a month-long video [1])

1. Kang et al., NoScope, PVLDB’17
Our Goal

Enable **low-latency** and **low-cost** querying over rapidly growing video datasets

Low-Latency and Low-Cost Video Querying System

CNN, Accuracy target

Query object class

Frames with trucks
A Convolutional Neural Network (CNN) outputs the probability of each class.

Based on the extracted features (high-level representation).
Focus System: Low-latency query with low-cost ingest

- Approximate indexing via cheap ingest
- Redundancy elimination for fast query
- Trading off ingest cost vs. query latency
Low-Cost Ingestion: Cheaper CNNs

• Process video frames with a cheap CNN at ingest time
  • Compressed and Specialized CNN: fewer layers / weights and are specialized for each video stream
Challenge: Cheap CNNs are Less Accurate

• Cheaper CNNs are less accurate than the expensive CNNs

The best result from the expensive CNN is within the top-K results of the cheaper CNN

<table>
<thead>
<tr>
<th>Rank</th>
<th>Expensive CNN</th>
<th>Cheap CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Truck</td>
<td>Moving Van</td>
</tr>
</tbody>
</table>
Recall: Fraction of relevant objects that are selected

Precision: Fraction of selected objects that are relevant

Ground-truth CNN: YOLOv2 (80 classes)

Cheap CNNs can achieve high recall with small top-K results.
Solution: Split Ingest- and Query-time Work

Ingest-time

Query-time

Query-time work is done only on queried videos (reduce waste)

Frames → Objects → Specialized, Compressed CNN → CNN specialization → Top-K Index

Object Class → [Objects] → Object → Frame

Expensive CNN → Querying for trucks

Frames with trucks

High Recall

High Precision
Focus System: Low-latency query with low-cost ingest

- Approximate indexing via cheap ingest
- Redundancy elimination for fast query
- Trading off ingest cost vs. query latency
Low-Latency Query: Redundancy Elimination

- Approximate indexing $\rightarrow$ non-trivial work at query time
  - A larger $K$ $\rightarrow$ more query-time work
- Images with similar feature vectors are visually similar
- Minimize the work at query time $\rightarrow$ clustering similar objects based on the extracted features

Query-time work is done *only once per cluster*
Adding Feature-based Clustering

**CNN specialization**

- Frames
- Objects
- Specialized, Compressed CNN

**Features**

- Ingest-time
- Query-time

**Clusters**

- Top-K results
- Top-K Index

**Object Class → [Clusters]**

**Cluster → Centroid, [Objects], [Frames]**

**Reducing redundant work at query time**

- Querying for trucks
- Frames with trucks

- Centroid objects
- Expensive CNN
Experimental Setup

• **Video Datasets**
  - 11 live traffic and enterprise videos
  - Each video stream is evaluated for 12 hours

• **Accuracy Targets**
  - 99% recall and 99% precision w.r.t. YOLOv2

• **Baselines**
  - **Ingest-heavy**: Analyzes all frames with YOLOv2 at ingest time and stores the inverted index for query
  - **NoScope [VLDB’17]**: A query-optimized system that analyzes frames only at query time
Focus achieves low-latency query with low-cost ingest
Other Applications

Process **large and growing data** with CNNs to answer “after the fact” queries

**Other Video Apps**
- Face Recognition
- Emotion Detection

**Audio**
- find audio segments with a word

**Bioinformatics**
- Brain signals
- Medical images

**Geoinformatics**
- Satellite images
- ...
Overview of Our Approach

ML Serving over Large, Rapidly-Growing Datasets [OSDI’18]
  ➢ Address the computation challenge

ML Training over Geo-Distributed Data [NSDI’17]
  ➢ Address the communication challenge

Understanding The Non-IID Data Partition Problem for Decentralized ML
  ➢ Address the statistical challenge
ML Training on Geo-Distributed Data
Centralizing Data is Infeasible [1, 2, 3]

• Moving data over wide-area networks (WANs) can be extremely slow

• It is also subject to data sovereignty laws

1. Vulimiri et al., NSDI’15
2. Pu et al., SIGCOMM’15
3. Viswanathan et al., OSDI’16
Geo-distributed ML is Challenging

- No ML system is designed to run across data centers (up to 53X slowdown in our study)
Our Goal

• Develop a **geo-distributed ML system**
  • Minimize communication over wide-area networks
  • Retain the **accuracy and correctness** of ML algorithms
  • Without requiring changes to the algorithms
Background: Parameter Server Architecture

• The parameter server architecture has been widely adopted in many ML systems.

Synchronization is critical to the accuracy and correctness of ML algorithms.
Deploy Parameter Servers on WANs

- Deploying parameter servers across data centers requires a lot of communication over WANs (up to 53X slowdown)
Gaia System Overview

- **Key idea:** Decouple the synchronization model *within* the data center from the synchronization model *between* data centers.

![Diagram showing the decoupled synchronization model between Data Center 1 and Data Center 2.](image-url)

- Communicate over WANs only significant updates.
- Approximately Correct Model Copy in both Data Centers.
Key Finding: Study of Update Significance

The vast majority of updates are insignificant.
Approximate Synchronous Parallel

The significance filter
  • Filter updates based on their significance

ASP selective barrier
  • Ensure significant updates are read in time

Mirror clock
  • Safe guard for pathological cases
The Significance Filter

Worker Machine

Parameter Server

Parameter X

Value

Aggregated Update

Update ($\Delta_2$) on X

Significance Function

Other Parameters

Significance Threshold

$\frac{Agg.\ Update}{Value} > \frac{10}{\sqrt{T}}$

$\Delta_1 + \Delta_2$
Approximate Synchronous Parallel

The significance filter
- Filter updates based on their significance

ASP selective barrier
- Ensure significant updates are read in time

Mirror clock
- Safeguard for pathological cases
ASP Selective Barrier

Only workers that depend on these parameters are blocked.
Experimental Setup

• Applications
  • Matrix Factorization with the Netflix dataset
  • Topic Modeling with the Nytimes dataset
  • Image Classification with the ILSVRC12 dataset

• Hardware platform
  • 22 machines with emulated EC2 WAN bandwidth
  • We validated the performance with a real EC2 deployment

• Baseline
  • IterStore (Cui et al., SoCC’14) and GeePS (Cui et al., EuroSys’16) on WAN

• Performance metrics
  • Execution time until algorithm convergence
  • Monetary cost of algorithm convergence
Performance and WAN Bandwidth

Gaia is at most 1.23X of LAN speeds
Results – EC2 Monetary Cost

Gaia is 2.6-59.0X cheaper than Baseline
Overview of Our Approach

ML Serving over Large, Rapidly-Growing Datasets [OSDI’18]
- Address the computation challenge

ML Training over Geo-Distributed Data [NSDI’17]
- Address the communication challenge

Understanding The Non-IID Data Partition Problem for Decentralized ML
- Address the statistical challenge
Real-World Data can be Highly Skewed (Non-IID)

What happens to ML if data partitions are not IID?
Studying ML over Non-IID Data Partitions

ML Application

- Image Classification (with various DNNs and datasets)
- Face recognition

Decentralized Learning Algorithm

- Gaia [NSDI’17]: only send updates above a threshold
- Federated Average [AISTATS’17]: local steps then average
- Deep Gradient Compression [ICLR’18]: send top 0.1% of updates

Degree of Deviation from IID
Non-IID Data: Setup

- Task: Classify an image into one of the object classes

- Each data center has some classes of images

Show results for 2 to 5 partitions...Only gets worse with more partitions
Results: GoogleNet over CIFAR-10

All decentralized ML approaches lose significant accuracy.

Tight synchronization (BSP) is accurate but too slow.
Even BSP cannot solve this problem for DNNs with Batch Normalization Layers.

Non-IID data is a pervasive and fundamental problem.
The Degree of Deviation from IID is a Key Factor

More deviation from IID makes the problem more difficult
Quick Summary (so far)

• Non-IID Data Quagmire: non-IID data partition is a pervasive and fundamental problem

• Even communicating everything cannot solve this problem for DNNs with batch normalization

• The degree of deviation from IID is a key factor
Solution for Batch Normalization
Background: Batch Norm Layer

Input: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_1...m\}$; Parameters to be learned: $\gamma, \beta$

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \quad \text{// mini-batch mean}$$

$$y_i \leftarrow \gamma \bar{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation $x$ over a mini-batch.
Each partition uses very different mean for normalization – no way to reconcile in the end

\[
\text{Diff / Mean} = \frac{\text{ABS}(\text{Mean}_1 - \text{Mean}_2)}{\text{AVG}(\text{Mean}_1, \text{Mean}_2)}
\]
Alternatives to Batch Normalization

- Weight Normalization [Salimans et al., NeurIPS’16]
- Layer Normalization [Ba et al., arXiv’16]
- Batch Renormalization [Ioffe, NeurIPS’17]
- Separate approaches for Mean and Variance
- Gradient clipping

All of them fail on non-IID data
Solution: Use Group Normalization
[Wu and He, ECCV’18]

Introduced for better training with super-small batches
We apply as a solution to the Batch Norm problem for non-IID data
GroupNorm significantly improves accuracy for all decentralized learning algorithms with non-IID data.

GroupNorm recovers the accuracy loss in BatchNorm with non-IID data using BSP.
On-going Work: Towards a Solution for Arbitrarily Non-IID Data
Solution Overview

1. **Periodic model traveling**
   - To measure Accuracy Gap
   - Underperforming data

2. **Communication tightness control**
   - Less deviation from IID data requires less communication
Promising Preliminary Results

- All results achieve the same accuracy as BSP

![Graph showing Communication Saving over BSP (times) for different data types.]

- IID Data
  - Communication Control: 140X
  - Oracle: 337X
- Non-IID Data
  - Communication Control: 44X
  - Oracle: 122X

AlexNet, 2 partitions, Gaia
Summary

Low-Latency and Low-Cost ML on Highly-Distributed and Rapidly-Growing Data

- Computation Challenge
  - ML Serving over Large, Rapidly-Growing Datasets (video) [OSDI’18]

- Communication Challenge
  - ML Training over Geo-Distributed Data [NSDI’17]

- Statistical Challenge
  - Understanding The Non-IID Data Partition Problem for Decentralized ML
Machine Learning Systems for Highly-Distributed and Rapidly-Growing Data

Acknowledgements

Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodik, Gregory R. Ganger, Phillip B. Gibbons, Aaron Harlap, Dimitris Konomis, Onur Mutlu, Amar Phanishayee, Matthai Philipose, Shivaram Venkataraman, Nandita Vijaykumar
Backup Slides for Gaia
WAN: Low Bandwidth and High Cost

- WAN bandwidth is 15X smaller than LAN bandwidth on average, and up to 60X smaller
- In Amazon EC2, the monetary cost of WAN communication is up to 38X the cost of renting machines
Running ML systems on WANs can seriously slow down ML applications.

- **Normalized Execution Time until Convergence**
  - IterStore
  - Bösen

- **Matrix Factorization**
  - LAN: 3.7X
  - EC2-ALL: 3.5X
  - V/C WAN: 23.8X
  - S/S WAN: 24.2X

1. **11 EC2 Regions**
   - 11 EC2 Regions

2. **Cui et al., “Exploiting Iterative-ness for Parallel ML Computations”, SoCC’14**
   - Wei et al., “Managed Communication and Consistency for Fast Data-Parallel Iterative Analytics”, SoCC’15
Problem: Broadcast Significant Updates

Communication overhead is proportional to the number of data centers
Mitigation: Overlay Networks and Hubs

Save communication on WANs by aggregating the updates at hubs
Gaia achieves 3.7-6.0X speedup over Baseline
Gaia is at most 1.40X of LAN speeds
## Compare Against Centralizing Approach

<table>
<thead>
<tr>
<th></th>
<th>Gaia Speedup over Centralize</th>
<th>Gaia to Centralize Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix Factorization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>1.11</td>
<td>3.54</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>2.13</td>
<td>1.17</td>
</tr>
<tr>
<td><strong>Topic Modeling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>0.80</td>
<td>6.14</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.02</td>
<td>1.26</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>1.25</td>
<td>1.92</td>
</tr>
<tr>
<td><strong>Image Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>0.76</td>
<td>3.33</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.12</td>
<td>1.07</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>1.86</td>
<td>1.08</td>
</tr>
</tbody>
</table>
SSP Performance – 11 Data Centers

Matrix Factorization

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulation-EC2</td>
<td>2.0X</td>
<td>2.0X</td>
<td>2.0X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>1.5X</td>
</tr>
<tr>
<td>Emulation-Full-Speed</td>
<td>1.8X</td>
<td>3.8X</td>
<td>3.7X</td>
<td>1.3X</td>
<td>3.0X</td>
<td>2.7X</td>
</tr>
</tbody>
</table>

- Amazon-EC2
- Emulation-EC2
- Emulation-Full-Speed

Normalized Execution Time
SSP Performance – 11 Data Centers

Topic Modeling

Baseline Gaia LAN Baseline Gaia LAN

Emulation-EC2 Emulation-Full-Speed

Normalized Execution Time

Baseline: 2.0X Gaia: 3.7X LAN: 4.8X
Baseline: 1.5X Gaia: 2.0X LAN: 3.5X

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
SSP Performance – V/C WAN

Matrix Factorization

- **BSP**
  - Baseline: 3.7X
  - Gaia: 3.5X
  - LAN: 2.6X

- **SSP**
  - Baseline: 2.3X
  - Gaia: 3.9X
  - LAN: 3.2X

Topic Modeling

- **BSP**
  - Baseline: 3.7X
  - Gaia: 3.9X
  - LAN: 3.1X

- **SSP**
  - Baseline: 3.2X
  - Gaia: 3.0X
  - LAN: 3.0X
SSP Performance – S/S WAN

Matrix Factorization

- Baseline
- Gaia
- LAN

BSP: 25X 24X
SSP: 16X 14X

Topic Modeling

- Baseline
- Gaia
- LAN

BSP: 14X 17X
SSP: 17X 21X
Backup Slides for Focus
Focus System: Low-latency query with low-cost ingest

- Approximate indexing via cheap ingest
- Redundancy elimination for fast query
- Trading off ingest cost vs. query latency
Ingest Cost vs. Query Latency

• Parameter selection → trading off ingest cost vs. query latency
  • The cheap CNN at ingest time
  • $K$ in the top-$K$ approximate indexing
  • Clustering threshold for feature-based clustering
  • …

△ / ▲ A set of configurations

Enable trade-offs to meet application’s need
Both techniques are important to Focus
Ingest Cost by Video

- **Ingest-NoScope**
- **Focus**

<table>
<thead>
<tr>
<th>Location</th>
<th>Traffic</th>
<th>Surveillance</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>auburn_c</td>
<td>35X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>auburn_r</td>
<td>66X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bellevue_d</td>
<td>67X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bellevue_r</td>
<td>84X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bend</td>
<td>40X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jackson_h</td>
<td>38X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jackson_ts</td>
<td>92X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coral</td>
<td>60X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lausanne</td>
<td>44X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oxford</td>
<td>84X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sittard</td>
<td>53X</td>
<td></td>
<td>57X</td>
</tr>
</tbody>
</table>

- Ingest cheaper than ingest-heavy by (factor)
Query Latency by Video

<table>
<thead>
<tr>
<th>Location</th>
<th>Traffic</th>
<th>Surveillance</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>auburn_c</td>
<td>304X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>auburn_r</td>
<td>46X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bellevue_d</td>
<td>444X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bellevue_r</td>
<td>75X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bend</td>
<td>40X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jackson_h</td>
<td>288X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jackson_ts</td>
<td>46X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coral</td>
<td>122X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lausanne</td>
<td>322X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oxford</td>
<td>607X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sittard</td>
<td>350X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>162X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query faster than NoScope by (factor)
Sensitivity – Number of Classes

• We study the sensitivity to the number of object class using 1,000 ImageNet classes

• The results show that Focus is
  • 15× faster in query latency
  • 57× cheaper in ingest cost than the baseline systems
Implementation Architecture

Frame / object extraction

Ingest Processor (IP1)

Ingest CNN evaluation

Objects

IP2

Ingest CNN and Feature vector clustering

Approximate index

IP3

Model and parameter selection

Specialized model training

ST1

GT-CNN, Accuracy target

ST2

Stream Tuner (M2)

Trade-off policy

Query object class

Frames with queried object class

Centroid object selection

GT-CNN evaluation

Query Processor (M3)

QP1

QP2
Backup Slides for Skewed Data Partitions
The Skewness of Partitions is a Key Factor

We can relax communication with less skewed data partitions.

Face Recognition
Result: Model Comparison

Models only work well for the classes that they have seen
How Do Models Diverge?

• Key operation in these DNNs: Convolution

Small model differences can result in completely different models

Input  Param 1 (1% diff vs. Param 0)
+20 * +0.495 = -0.6545
+19  -0.5555 45% diff!!
Why ResNet? Batch Norm Layer

**Input:** Values of \( x \) over a mini-batch: \( B = \{x_1...m\} \);
Parameters to be learned: \( \gamma, \beta \)

**Output:** \( \{y_i = \text{BN}_{\gamma,\beta}(x_i)\} \)

\[
\begin{align*}
\mu_B & \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i & \quad \text{// mini-batch mean} \\
\sigma_B^2 & \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 & \quad \text{// mini-batch variance} \\
\hat{x}_i & \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} & \quad \text{// normalize} \\
y_i & \leftarrow \gamma \hat{x}_i + \beta = \text{BN}_{\gamma,\beta}(x_i) & \quad \text{// scale and shift}
\end{align*}
\]

**Algorithm 1:** Batch Normalizing Transform, applied to activation \( x \) over a mini-batch.

(test time uses global mean/average)

Scale and shift are implemented as another layer (Scale) to learn \( \gamma \) and \( \beta \). This layer is not required if the following activation layer is linear (e.g., ReLU)
Why Batch Norm is Sensitive to Sampling?

- When applying Batch Norm on SGD, we use mini-batch mean and variance to normalize each minibatch.
- If sampling is stratified at training time, the internal variance of each mini-batch will be reduced.
  - It will still be normalized but it is not a good estimate of underlying data distribution.
  - Also, at test time we use global mean and variance, which is different from the stratified sampling at training time.
Alternatives to Batch Normalization

- **Weight Normalization**
  - Use L2 norm to normalize weights
  - May need a mean-only batch normalization layer

- **Layer Normalization**
  - Use the sum of all neurons to estimate mean and variance
  - May not work if there is a large difference between different neurons in the same layer (such as CNNs)

- **Self-Normalizing Neural Networks**
  - Use a scaled exponential linear units to make the NN normalize by itself
  - Still trying to normalize the mean and variance within a layer
  - Works better for FNNs with mostly fully connected layers
Top-1 validation accuracy (CIFAR-10) varying Gaia’s T0 hyper-parameter

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td><strong>BSP</strong></td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td><strong>T0 = 2%</strong></td>
<td>73.8%</td>
<td><strong>70.5%</strong></td>
<td>78.4%</td>
<td><strong>56.5%</strong></td>
</tr>
<tr>
<td><strong>T0 = 5%</strong></td>
<td>73.2%</td>
<td><strong>71.4%</strong></td>
<td>77.6%</td>
<td><strong>75.6%</strong></td>
</tr>
<tr>
<td><strong>T0 = 10%</strong></td>
<td>73.0%</td>
<td><strong>10.0%</strong></td>
<td>78.4%</td>
<td><strong>68.0%</strong></td>
</tr>
<tr>
<td><strong>T0 = 20%</strong></td>
<td><strong>72.5%</strong></td>
<td><strong>37.6%</strong></td>
<td>77.7%</td>
<td><strong>67.0%</strong></td>
</tr>
<tr>
<td><strong>T0 = 30%</strong></td>
<td><strong>72.4%</strong></td>
<td><strong>26.0%</strong></td>
<td>77.5%</td>
<td><strong>23.9%</strong></td>
</tr>
<tr>
<td><strong>T0 = 40%</strong></td>
<td><strong>71.4%</strong></td>
<td><strong>20.1%</strong></td>
<td>77.2%</td>
<td><strong>33.4%</strong></td>
</tr>
<tr>
<td><strong>T0 = 50%</strong></td>
<td><strong>10.0%</strong></td>
<td><strong>22.2%</strong></td>
<td><strong>76.2%</strong></td>
<td><strong>26.7%</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CIFAR-10 Top-1 validation accuracy with various Federated Averaging hyper-parameters.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td>BSP</td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 5$</td>
<td>73.7%</td>
<td>62.8%</td>
<td>75.8%</td>
<td>68.9%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 10$</td>
<td>73.5%</td>
<td>60.1%</td>
<td>76.4%</td>
<td>64.8%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 20$</td>
<td>73.4%</td>
<td>59.4%</td>
<td>76.3%</td>
<td>64.0%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 50$</td>
<td>73.5%</td>
<td>56.3%</td>
<td>75.9%</td>
<td>59.6%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 200$</td>
<td>73.7%</td>
<td>53.2%</td>
<td>76.8%</td>
<td>52.9%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 500$</td>
<td>73.0%</td>
<td>24.0%</td>
<td>76.8%</td>
<td>20.8%</td>
</tr>
<tr>
<td>$\text{Iter}_\text{Local} = 1000$</td>
<td>73.4%</td>
<td>23.9%</td>
<td>76.1%</td>
<td>20.9%</td>
</tr>
</tbody>
</table>
CIFAR-10 Top-1 validation accuracy with various DeepGradientCompression hyper-parameters

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td>BSP</td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 8$</td>
<td>75.5%</td>
<td><strong>72.3%</strong></td>
<td>78.3%</td>
<td><strong>10.0%</strong></td>
</tr>
<tr>
<td>$E_{\text{warm}} = 4$</td>
<td>75.5%</td>
<td>75.7%</td>
<td>79.4%</td>
<td><strong>61.6%</strong></td>
</tr>
<tr>
<td>$E_{\text{warm}} = 3$</td>
<td>75.9%</td>
<td>74.9%</td>
<td>78.9%</td>
<td><strong>75.7%</strong></td>
</tr>
<tr>
<td>$E_{\text{warm}} = 2$</td>
<td>75.7%</td>
<td>76.7%</td>
<td>79.0%</td>
<td><strong>58.7%</strong></td>
</tr>
<tr>
<td>$E_{\text{warm}} = 1$</td>
<td>75.4%</td>
<td>77.9%</td>
<td>78.6%</td>
<td><strong>74.7%</strong></td>
</tr>
</tbody>
</table>
Prior Work

- Distributed ML training with centralized data
  - Examples: DistBelief [NIPS’12], Petuum [SIGKDD’15],
  - Example: Federated Learning [AISTATS’17]
  - ML training algorithms to reduce the dependency on intensive parameter updates
  - Examples: TensorFlow Serving, Clipper [NSDI’17]
  - Do not focus on serving large, rapidly-growing data
  - Query latency on large-scale data is still slow with

It is challenging to achieve low-latency and low-cost ML over highly-distributed and rapidly-growing data
Other Contributions

Processing-in-Memory

- Automatic offloading [ISCA’16, SC’17]
- Pointer chasing accelerator [ICCD’16]
- Cache coherence [CAL’17, ISCA’19]
- Bulk bit-wise ops [CAL’16]

Cross-layer abstractions

- Expressive Memory [ISCA’18]
- Locality descriptor in GPUs [ISCA’18]
- GPU programmability [MICRO’16]

Memory

- Variable DRAM latency [SIGMETRICS’16]