Eco-FL: Adaptive Federated Learning with Efficient Edge Collaborative Pipeline Training

Shengyuan Ye, Liekang Zeng, Qiong Wu, Ke Luo, Qingze Fang, Xu Chen

School of Computer Science and Engineering
Sun Yat-sen University
Federated Learning

- Federated Learning workflow
  - Each client uses its local data and available IoT computing resource to learn model parameters
  - A central server aggregates parameters to update the global model
  - Periodically synchronizes global model with each client
Challenges of Training on Devices

- Hetero. hardware
- Limit memory
- Limit bandwidth
- Limit computing capacity
- Dynamic resource
- Limit energy

How to train?

- DNN Training is extremely computation-intensive and resource-demanding
- Edge devices are resource-constrained and heterogeneous

Credit: Google Image
Challenges of Training on Devices

• **Existing Literature:**
  - Model Compression and Pruning
  - Model Quantization
  - Applying *light-weight* model on edge

• **Drawback:**
  - Defect the model test accuracy as well as FL’s training convergence
  - Need to optimize specially for a particular model and is not easily expandable
Edge Collaboration DNN Training for FL

• **Issues:**

  • **Collaboration mechanism** to orchestrate distributed edge devices
  • **Dynamics** computing resource of IoT devices
  • **Heterogeneous** computing capability of collaborated devices
Edge Collaboration DNN Training for FL

Data Parallelism

- Each device holds a complete model, which is expensive for memory-constrained IoT devices.
- Parameters transmission overhead can occupy nearly 66.3% in data parallelism training.

Pipeline Parallelism

- Each device is responsible for a subset of model layers caching and computing.
- Transmission overhead can be efficiently overlap by forward and backward computation.
Eco-FL Framework Overview

**Server-side**

- **FL Participants**
- **Eco-FL Cloud Server**
  - Grouping Scheduler (§5.2)
  - Asynchronous Aggregator (§5.1)
  - Sync. Aggregator
  - Sync. Aggregator
  - Sync. Aggregator
- **Adaptive Grouping**
- **Smart Home Groups**
  - Group 1
  - Group 2
  - Group 3

**Client-side**

- **Edge Pipeline Training in Smart Home**
  - Stage 0
  - Stage 1
  - Stage 2
  - Stage 3
- **Dynamic Profiler**
- **Hetero.-Aware Workload Partitioner (§4.2)**
- **Adaptive Pipeline Scheduler (§4.3, §4.4)**

**Hierarchical grouping-based FL**

**Pipeline edge collaboration**
Collaborative Edge Training Via Pipeline Parallelism

• Pipeline Parallelism Workflow

Partition DNN into stages → Split mini-batch into micro-batches → Inject micro-batches concurrently → Update stage model (Pipeline flush)

The number represents the micro-batch ID

Bubble means the device’s idle time
Collaborative Edge Training Via Pipeline Parallelism

**Traditional Pipeline Strategy**

- Backward pass only start after finishing all forward pass (**BAF Strategy**)
- Activations produced by forward tasks have to be kept for all micro-batches until backward pass begin, which is memory-unfriendly for IoT devices

**Eco-FL Resource-Efficient Pipeline Strategy**

- Schedule **one** Forward pass followed by **one** Backward pass (**1F1B Strategy**)
- Employ an early backward scheduling to release memory produced by forward pass for reuse
- Maintain the same throughput as BAF strategy
Collaborative Edge Training Via Pipeline Parallelism

• Heterogeneity-Aware Workload Partitioning

  • Step 1: Profiling
    • Monitor the computation time across forward pass and backward pass on heterogeneous IoT devices
    • Collect layer message of DNN model

  • Step 2: Workload Partitioning:
    • Global throughput of the pipeline is determined by the execution time of slowest stage (lagger)
    • Partition the model into balanced stages with dynamic programming algorithm
Collaborative Edge Training Via Pipeline Parallelism

• Pipeline bubbles analysis
  
  • **Synchronous Static Bubble (SSB)**
    • Caused by the periodic pipeline flush, *inevitable* in synchronous strategy
  
  • Can be *minimized* by increasing the number of micro-batches injected concurrently

  • **Data Dependency Bubble (DDB)**
    • Caused by the *data dependency* of micro-batches in pipeline training.
  
    • The occurrence of DDB is *periodic* and cannot be eliminated by increasing the number of micro-batches
Collaborative Edge Training Via Pipeline Parallelism

• Trade off between training throughput and peak memory usage
  • DDB is determined by the the number of forward pass (FP) in start-up phase
  • If the number of FP in start-up phase too small, DDB will occur. But if too many FP reside concurrently in stages, it will cause memory pressure to IoT devices.

• Best micro-batch scheduling strategy
  • Eq1: # of FP in start-up = 2(# of stages – stage index) – 1
  • Avoid the occurrence of DDB while minimizing memory pressure of each stage
Collaborative Edge Training Via Pipeline Parallelism

• **Issue: Dynamic edge resources**
  - IoT devices usually have **high variation** in available computing capability and memory resources
  - The maximum throughput of the pipeline is **greatly determined by the lagger**
Collaborative Edge Training Via Pipeline Parallelism

• **Issue: Dynamic edge resources**
  - IoT devices usually have high variation in available computing capability and memory resources
  - The maximum throughput of the pipeline is greatly determined by the lagger

• **Solution: Adaptive workload migration**
  - Training worker will periodically report the execution time of FP and BP
  - If there is a large deviation between the current and historical execution time of any device, pipeline will adaptively self-rebalance and migrate workload according to new scheduling.
Grouping-based Hierarchical FL Aggregations

Traditional Sync. & Async. FL Architecture

- **Sync. FL:**
  - Achieve high training performance
  - The slowest client (straggler) can significantly prolong the training time

- **Async. FL:**
  - Alleviate the straggler issue
  - Sacrifice accuracy and convergence speed

Eco-FL Hierarchical Architecture

- The available trusted devices that each smart home can collaborate with usually vary, which causes severe straggler issue

- Hybrid Hierarchical FL combine the best of both Sync. and Async. mechanisms, while efficiently alleviate straggler issue.
Grouping-based Hierarchical FL Aggregations

Adaptive Client Grouping:
- Group smart homes according to their training performance and data distribution

Intra-group Sync. Aggregation

Inter-group Async. Aggregation
Adaptive Client Grouping:
- Group smart homes according to their training performance and data distribution

Intra-group Synchronous Aggregation:
- Synchronous aggregation is applied to aggregate model updates from the clients with similar response latency within a same group
Adaptive Client Grouping:
- Group smart homes according to their training performance and data distribution

Intra-group Synchronous Aggregation:
- Synchronous aggregation is applied to aggregate model updates from the clients with similar response latency within a same group

Inter-group Asynchronous Aggregation:
- Asynchronous aggregation is made for global model aggregation among different groups
Grouping-based Hierarchical FL Aggregations

• Heterogeneity-aware Client Grouping

  • System heterogeneity: Stragglers will prolong the synchronous training time intra-group

  • Data heterogeneity: Non-IID characteristics can harm the convergence of model training during synchronous process intra-group

• Grouping Target:

  • Let the response latency of the members in the group be as close as possible while having an associated data distribution as close as possible to the I.I.D. distribution
Grouping-based Hierarchical FL Aggregations

- **Dynamic Client Re-grouping**

  - The response latency of each client can be varying occasionally due to the changes in its collaborative device resources, which can disable the static grouping method.

  - Eco-FL server will monitor each client in run-time and dynamically re-group client according to their real-time response latency.

  

  ![Diagram of Grouping](image)

  

  **Algorithm 1: Adaptive Grouping Process**

  ```
  Process Eco-FLServer():
  Collect and monitor response latency of each client;
  if Client n in group g satisfy |L⁹ - Lₙ| > RT⁹ then
    Regroup(n);
  end

  Function Regroup(n):
  MinCost ← +∞, t ← −1;
  for g ∈ {0, 1, ..., |G| − 1} do
    if COSTₙ⁹ < MinCost and |L⁹ - Lₙ| ≤ RT⁹ then
      t ← g;
      MinCost ← COSTₙ⁹;
    end
  end
  if t ≠ −1 then
    Move client n to group t;
  else
    Drop out client n until its response latency Lₙ meets the threshold range of any group;
  end
  ```

```
Evaluation

• Experimental Setting
  • Models:
    • CNN with two 5x5 convolution layers
  • Baselines:
    • FedAvg, FedAsync, FedAT, Astraea
  • Datasets:
    • Cifar10, MNIST, Fashion-MNIST
  • Testbed:
    • Virtual machine instance (48 vCPUs and 64GB memory)
    • Use Docker to deploy FL server and clients. Each client gets assigned 2 vCPU cores

• Models:
  • EfficientNet, MobileNetv2
• Baselines:
  • PipeDream, Gpipe, Single device
• Testbed:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Power Mode</th>
<th>GPU Max Frequency</th>
<th>Memory</th>
<th>Network Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jetson Nano</td>
<td>5W (L)</td>
<td>640MHz</td>
<td>4GB</td>
<td>100Mbps</td>
</tr>
<tr>
<td></td>
<td>10W (H)</td>
<td>921.6MHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jetson TX2</td>
<td>Max-Q</td>
<td>850MHz</td>
<td>8GB</td>
<td>100Mbps</td>
</tr>
<tr>
<td></td>
<td>Max-N</td>
<td>1.3GHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation

• Federated Learning Performance
  • Training performance
    • Eco-FL hierarchical architecture outperforms the baselines with faster convergence and higher achieved accuracy
    • Eco-FL with the adaptive scheduler can still maintain a high performance under the IoT environment with dynamic nature
  • Heterogeneity-aware client grouping
    • FedAT: Group clients only based on response latency
    • Astraea: Grouping clients only based on data distribution
    • Eco-FL heterogeneity-aware grouping method outperform both FedAT and Astraea up to 26.3% on testing accuracy

Training performance with different datasets

Effectiveness of heterogeneity-aware client grouping
Evaluation

• Pipeline Training Performance

  • Training Results
   • Evaluation on a 2-stage pipeline and a 3-stage pipeline
   • Eco-FL pipeline efficiently collaborates the computation power of all IoT devices and reaches the target accuracy $2.6 \times$ faster than data parallelism.
Evaluation

• Pipeline Training Performance
  • Dynamic pipeline re-scheduling and workload migration
    • An external GPU workload to device 2 at the 100-th timestamp.
    • Without pipeline re-scheduling, the training speed of device 2 will significantly slow down and become lagger in the pipeline.
    • With our adaptive pipeline scheduler, device 2 will migrate part of model layers to device 1 and 3 to rebalance the workload across each stage.
Conclusion

- We devise a novel edge collaborative pipeline parallelism to achieve edge resource pooling over trusted devices in proximity for local FL model training acceleration.
- We propose Eco-FL, a hierarchical FL framework upon the edge collaborative pipeline training, which jointly considers both the response latency and data distribution divergence.
- We feature adaptive scheduling in both FL server and client sides to tackle system dynamics inherent in edge scenarios.
- Experimental results show that Eco-FL can upgrade the training accuracy by up to 26.3%, reduce the local training time by up to 61.5%, and improve the local training throughput by up to $2.6\times$ against state-of-the-art baselines.

Thanks!

Shengyuan Ye, Liekang Zeng, Qiong Wu, Ke Luo, Qingze Fang, Xu Chen
yeshy8@mail2.sysu.edu.cn, Sun Yat-sen University