Semi-Asynchronous Energy-Efficient Federated Prototype Learning for Client-Edge-Cloud Architectures

***Abstract*—**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

I. Introduction

THIS

II. Related Work

III. Motivation

IV. Method

A. Proposed Algorithm

Below is a table of symbols used in the algorithms:

TABLE I
SYMBOL TABLE

Symbol	**Description**
J	Number of classes
T	Global communication rounds
E	Edge communication rounds
K	Local train epochs
L	Number of edge servers
B	The buffer of the cloud server with a static length
N^l	Number of clients in the l-th each edge server
$\mathcal{N}_j^l$	Number of clients in edge l containing class j that have participated in aggregation
$\mathcal{N}_j^{l,\text{prev}}$	Previous number of clients in edge l containing class j that have participated in aggregation
S^l	Set of clients participating in training in the l-th edge server
$\bar{C}_j$	Aggregated prototype of class j in the cloud edge server
C_j^l	Aggregated prototype of class j from the l-th edge server
$C_j^{l,\text{prev}}$	Previous version of aggregated prototype of class j from the l-th edge server in the cloud server
$c_{i,j}^l$	Aggregated prototype of class j from the client i in the l-th edge server
$c_{i,j}^{l,\text{prev}}$	Previous version of the aggregated prototype of class j from client i in the l-th edge server in the edge server
$D_{i,j}^l$	A subset of the local dataset D_i^l of the i-th client in the l-th edge server, containing training instances of class j.
$\mathcal{X}_i^l$	The feature and label set of the i-th client in the l-th edge server, containing all features of D_i.
G^l	The global classifier in the l-th edge server.

The following algorithm demonstrates how to calculate the factorial of a number.

Algorithm 1 Hierarchical Federated Prototype Learning -Part 1

```
procedure CLOUD SERVER EXECUTES
    Initialize weights for clients with heterogeneous models.
    All edge servers execute in parallel.
    for t = 1, ..., T do
        Clear the buffer B
        while B is not full do                          ▷ Async process
            Receive a triple (C^l, 𝒩^l, 𝒳^l) from one edge server.
            Populate B with the received triple.
        end while
        C̄, G ← CloudUpdate(B)
        Send C̄, G to edge servers participating in the current global aggregation.
        These edge servers re-execute.
    end for
end procedure
procedure EDGE SERVER EXECUTES
    Receive C̄, G from the cloud server
    Choose a set of clients S^l to train in parallel.
    for e = 1, ..., E do                                ▷ E now is static 1
        Send C̄, G to client i ∈ S^l
        for each client i in parallel do
            (c_i^l, 𝒳_i^l) ← ClientUpdate(i, C̄, G)
        end for                                         ▷ Wait for all clients
        (C^l, 𝒩^l, 𝒳^l) ← EdgeAggregate({(c_i^l, 𝒳_i^l)}_{i∈S^l})
        C̄ ← EdgeUpdate(C̄, C^l)                          ▷ not used now
    end for
    Send a triple (C^l, 𝒩^l, 𝒳^l) to the cloud server
end procedure
```

Algorithm 2 Hierarchical Federated Prototype Learning -Part 2

procedure CLOUDUPDATE(B)
  for $j = 1, \dots, J$ **do**
    $\hat{C}_j \leftarrow \sum_{l=1}^{L} \mathcal{N}_j^{l,\text{prev}} \cdot \bar{C}_j$ ▷ Extend the aggregated prototypes $\bar{C}$
    for $(C^l, \mathcal{N}^l) \in B$ **do**
      $\hat{C}_j \leftarrow \hat{C}_j + \mathcal{N}_j^l \cdot C_j^l$
      if $C_j^{l,\text{prev}}$ is not empty **then**
        $\hat{C}_j \leftarrow \hat{C}_j - \mathcal{N}_j^{l,\text{prev}} \cdot C_j^{l,\text{prev}}$
      end if
      $\mathcal{N}_j^{l,\text{prev}} \leftarrow \mathcal{N}_j^l$
    end for
    $\bar{C}_j \leftarrow \frac{\hat{C}_j}{\sum_{l=1}^{L} \mathcal{N}_j^{l,\text{prev}}}$
  end for
  $C^{l,\text{prev}} \leftarrow C^l$ for $l \in B$
  $\mathcal{X} \leftarrow \bigcup_{l \in B} \mathcal{X}^l$
  Train the global classifier G by using $\mathcal{X}$.
  return $\bar{C}, G$
end procedure
procedure EDGEAGGREGATE($l, \{(c_i^l, \mathcal{X}_i^l)\}_{i \in S^l}$)
  for $j = 1, \dots, J$ **do**
    $\hat{C}_j^l \leftarrow \mathcal{N}_j^l \cdot C_j^l$
    for each c_i^l **do**
      $\hat{C}_j^l \leftarrow \hat{C}_j^l + c_{i,j}^l$
      if $c_{i,j}^{l,\text{prev}}$ is not empty **then**
        $\hat{C}_j^l \leftarrow \hat{C}_j^l - c_{i,j}^{l,\text{prev}}$
      else
        $\mathcal{N}_j^l \leftarrow \mathcal{N}_j^l + 1$
      end if
    end for
    $C_j^l \leftarrow \frac{\hat{C}_j^l}{\mathcal{N}_j^l}$
  end for
  $c_i^{l,\text{prev}} \leftarrow c_i^l$ for $i \in S^l$
  $\mathcal{X}^l \leftarrow \bigcup_{i \in S^l} \mathcal{X}_i^l$
  return $(C^l, \mathcal{N}^l, \mathcal{X}^l)$
end procedure
procedure CLIENTUPDATE($i, \bar{C}, G$)
  Receive $\bar{C}, G$ from the edge server
  for $k = 1, \dots, K$ **do**
    DVFS to be implemented...
    for batch $(x, y) \in D_i$ **do**
      Compute client prototypes by Eq.?.
      Compute loss by Eq.? using client prototypes and the global classifier G.
      Update client model according to the loss.
    end for
    Store the features and labels of D_i in $\mathcal{X}_i^l$.
  end for
  return $(c_i^l, \mathcal{X}_i^l)$
end procedure

V. EXPERIMENTS

VI. CONCLUSION