https://doi.org/10.7840/kics.2025.50.2.195

# 클라이언트의 서로 다른 학습 능력을 고려한 분산학습 알고리즘 연구

류 지 현◆, 양 희 철°

# A Study on Distributed Learning Algorithm for Heterogeneous Client Settings in Computing Capabilities

Ji-Hyun Ryu◆, Heecheol Yang°

요 약

본 논문에서는 서로 다른 학습 능력을 가진 클라이언트와 서버로 구성된 분산학습 환경을 고려한다. 클라이언트의 컴퓨팅 능력이 제한되고 그 능력이 서로 다른 경우, 많은 클라이언트가 학습에 참여할 수 있도록 서버의 학습 능력을 활용하는 분할 학습과 순서화된 드롭아웃(Ordered dropout) 방식을 결합한 분산학습 알고리즘을 제안한다. 이를 통해 클라이언트가 학습하는 클라이언트 모델의 크기가 서로 다르더라도 모두 분산학습에 참여하여 전역 모델 성능 향상에 기여할 수 있음을 밝혔다. 기존에 제안된 ResNet50 모델을 통해 CIFAR-10 데이터셋에 대한 이미지 분류를 수행하는 실험을 설계하였으며, 클라이언트의 수와 데이터셋의 분포에 따른 분류 성능을 확인하였다. 클라이언트의 컴퓨팅 능력이 서로 다른 상황을 가정하여 클라이언트마다 다른 크기의 클라이언트 모델을 학습하였을 때, 모든 클라이언트가 전역 모델 학습에 기여함을 보였다.

**키워드 :** 분산 학습, 연합 학습, 분할 학습
**Key Words :** Distributed learning, Federated learning, Split learning

ABSTRACT

In this paper, we address a distributed learning system with a server and clients with heterogeneous computational capabilities. We propose a new distributed learning algorithm that combines split learning with ordered dropout, enabling clients with limited and heterogeneous computing capabilities to participate in training. This approach allows all clients, even with different client-model sizes, to contribute to the improvement of the global model's performance. We conduct experiments on image classification using ResNet50 on the CIFAR-10 dataset, examining classification performance given the number of clients and the distribution of the dataset. With heterogeneous client settings in computational capacities, simulation results demonstrate that all clients with various client-side model sizes effectively contribute to global model training.

※ 본 연구는 한국연구재단 이공분야기초연구사업(2022R1C1C1005749), 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(RS-2022-II221200)에 의해 수행되었습니다.
◆ First Author : Chungnam National University Department of Computer Science and Engineering, ryjihyunmail@gmail.com, 학생회원
° Corresponding Author : Chungnam National University Division of Computer Convergence, hcyang@cnu.ac.kr, 종신회원
논문번호 : 202411-274-A-RU, Received November 8, 2024; Revised November 21, 2024; Accepted November 25, 2024

## Ⅰ. 서 론

디지털 시대의 급격한 발전과 함께 다양한 스마트폰, 태블릿 PC, IoT 기기의 등장으로 다양한 기기에서 얻을 수 있는 데이터의 양은 기하급수적으로 증가하고 있다. 이러한 데이터를 활용하여 기계학습 또는 딥러닝 모델의 성능을 향상하는데 기여할 수 있지만, 개인의 민감한 정보를 포함하는 경우 데이터의 보안과 프라이버시 문제를 일으킨다. 특히 의료, 금융, 소셜 네트워크 서비스(SNS)와 같은 분야에서는 데이터의 보호를 우선시하고 있다. 이러한 상황에서 제안된 분산학습 방식 중 하나가 연합학습(Federated Learning, FL)이다[1].

연합학습은 서버와 여러 클라이언트로 구성된 환경에서 학습을 진행하는 방식이다. 서버에서 여러 클라이언트에 공통의 전역 모델을 전달하고 클라이언트는 각자의 데이터를 가지고 지역 모델을 학습하여 모델의 파라미터를 서버에 전송한다. 서버는 각 클라이언트에서 전달받은 파라미터를 취합하여 전역 모델에 대한 업데이트를 진행한다. 이를 다시 각 클라이언트에 전달하는 과정을 반복하여 학습을 진행한다. 이러한 접근 방식을 통해 클라이언트의 데이터를 서버에 공개하지 않고 전역 모델에 대한 학습이 진행되어 데이터를 보호할 수 있는 장점이 존재한다. 하지만, 최근 거대 언어 모델(Large language model)과 같이 모델의 크기가 과도하게 증가하는 경향이 있어 모델 파라미터를 전달하는 통신 과정에서 발생하는 통신 부담이 증가하는 단점이 있다. 또한, 각 클라이언트의 컴퓨팅 능력이 제한되는 상황에서 일부 클라이언트가 학습에 참여하지 못하거나 특정 클라이언트의 학습 지연으로 인해 전체 학습이 정상적으로 진행되지 못하는 문제가 발생할 수 있어 이로 인해 전역 모델의 성능이 만족스럽지 못한 결과를 얻을 수 있다.

연합학습에서의 단점을 보완하기 위해 새로운 분산학습 방식인 분할학습(Split Learning, SL)이 제안되었다[2]. 분할학습은 전체 모델을 서버 부분 모델과 클라이언트 부분 모델로 나누어 각각에 대해 서버와 클라이언트에서 학습을 진행하게 된다. 먼저 클라이언트가 학습 데이터를 입력으로 하여 클라이언트 부분 모델에 대한 학습을 진행하고 중간 출력을 서버가 전달받아 서버 부분 모델의 학습을 진행한다. 이를 통해 클라이언트의 컴퓨팅 능력이 제한되는 상황에서도 서버의 컴퓨팅 능력을 빌려 전체 모델에 대한 학습이 가능하다는 장점이 존재하지만, 연합학습과 달리 클라이언트가 순차적으로 학습을 진행하여 학습 진행의 지연이 크다는 단점이 존재한다.

최근에는 특히 클라이언트의 컴퓨팅 능력이 제한된 환경과 열악한 통신 환경으로 인한 클라이언트 간의 이질성을 효과적으로 대처하는 분산학습 알고리즘이 제안되었다[3-7]. [3]에서 제안하는 FjORD는 기존의 모델 학습과정에서 과대적합을 막고자 제안된 임의 드롭아웃(Random Dropout; RD)과 달리, 순서화된 드롭아웃(Ordered Dropout; OD)을 적용하여 클라이언트 간 이질성을 대처하고자 하였다. OD는 RD와 달리 모델의 개별 노드에 대한 중요도 기반 드롭아웃 방식으로 중요도에 따라 정렬하여 순차적으로 드롭아웃을 진행한다. 전체 모델에 OD를 적용하여 성능이 낮은 클라이언트는 작은 서브모델을, 성능이 높은 클라이언트는 더 큰 서브모델을 학습할 수 있는 방식을 제안하였다. FjORD가 클라이언트의 컴퓨팅 자원 제약을 해결하기 위해 너비 기반 서브모델 축소 방식을 제안하였다면, DepthFL에서는 깊이 기반 서브모델 축소 방식을 제안하였다 [4]. 이 방식은 전체 모델의 깊은 레이어들을 가지치기(pruning)함으로써 성능이 낮은 클라이언트는 얕은 서브모델을 학습하고 성능이 높은 클라이언트는 깊은 서브모델을 학습함으로써 전체 모델 학습이 진행될 수 있다는 것을 보여준다. FedHM에서는 글로벌 모델을 저랭크 팩터화(low-rank factorization)를 통해서 서브모델을 생성하는 방식을 제안하였다[5]. 이를 통해서 컴퓨팅 능력이 제한된 클라이언트에서는 저랭크의 서브모델을 학습하고, 서버에 업데이트를 전송하는 방식으로 학습을 진행한다. 서버에서는 저랭크 서브모델을 다시 풀랭크(full-rank)형태로 복원하여 가중치 기반 평균 방식으로 글로벌 모델을 업데이트한다. FedRolex는 롤링 서브모델 추출 방식을 사용하여 글로벌 모델에서 서브모델을 추출하는 방식을 제안한다. 이 방식은 롤링 윈도우(Rolling window)기법을 사용하여 모델의 서로 다른 부분이 선택되도록 하며 모든 파라미터가 균등하게 학습됨을 보여준다[6]. 또한, SplitFed 기법은 분산학습 환경에서 활용할 수 있는 FL과 SL의 장점을 결합한 학습 방식이다[7]. 기존 SL 방식에서 클라이언트가 순차적으로 학습에 참여하는 단점을 FL의 병렬 학습 방식을 적용하여 학습 지연 효과를 줄이면서 서버의 학습 능력을 활용함으로써 컴퓨팅 능력이 부족한 클라이언트도 학습에 참여할 수 있음을 보여준다.

본 논문에서는 SplitFed 기법을 활용하는 분산학습 환경에서 클라이언트 간 이질성을 효과적으로 대처하는 알고리즘을 제안하고자 한다. 기존 연구로 SplitFed 알고리즘에 클라이언트 간 이질성 또는 클라이언트의 학습 능력 제한을 추가적으로 고려한 방안이 제안되었다[8-10]. NeFL 알고리즘은 SL과 같이 전체 모델을 클라

이언트 부분 모델과 서버 부분 모델로 분할하지 않고, 클라이언트의 컴퓨팅 능력에 따라서 전체 모델을 너비와 깊이 모두 축소하여 학습할 수 있는 방안을 제안한다[8]. FedSplitX에서 제안한 방식은 전체 모델을 서버 부분 모델과 클라이언트 부분 모델로 분할하는 과정에서 클라이언트의 컴퓨팅 능력을 고려하여 다양한 깊이의 분할 지점을 설정하여 클라이언트마다 서로 다른 깊이의 클라이언트 부분 모델을 전달한다[9]. 또한, AdaptSFL은 분산학습 환경에 적응적인 알고리즘으로 전체 모델을 서버와 클라이언트가 서로 나누어 연산 부담을 분산시키면서 클라이언트의 컴퓨팅 능력을 추가로 고려하게 된다[10]. 주기적으로 진행되는 클라이언트 부분 모델의 통합 과정에서 주기에 따라 모델의 정확도와 일관성의 차이가 발생하여 적절한 통합 주기를 설정하여 정확도와 통신 비용을 조절할 수 있다. 추가로, 학습 진행 방식에 관한 연구 외에 서버와 클라이언트 간의 통신 비용에 대한 다양한 연구가 진행되었다[11-13].

본 논문에서는 논문[14]에서 제안한 분산학습 기법을 확장하여 각 클라이언트의 이질적이고 제한된 컴퓨팅 능력을 고려하여 모든 클라이언트가 전체 모델 학습에 참여할 수 있도록 SplitFed 방식과 FjORD 알고리즘을 활용한 분산학습 기법인 HeteroSplitFed를 제안한다. 기존 SplitFed 방식에서는 분할 레이어(cut layer)를 기준으로 전체 모델을 클라이언트 부분 모델과 서버 부분 모델로 나누어 학습을 진행한다. 본 연구에서는 SplitFed 방식에서 클라이언트 부분 모델에 대한 OD를 적용하여 각 클라이언트의 이질적인 컴퓨팅 능력을 추가로 고려해 클라이언트의 제한된 컴퓨팅 능력에 효과적으로 대응할 수 있도록 한다. 성능 확인을 위해 ResNet50 모델을 서버 부분 모델과 클라이언트 부분 모델로 분할하고, 클라이언트 부분 모델에 대해 채널 기준 OD를 적용하였다. CIFAR-10 데이터셋을 각 클라이언트에 분포하고 전체 모델의 이미지 분류 성능을 확인하였다. 서로 다른 데이터 분포와 클라이언트의 수, 클라이언트 부분 모델의 크기를 조정해가며 성능을 확인하였고, 이를 통해 서로 다른 컴퓨팅 능력을 갖춘 클라이언트가 모두 전체 모델 학습에 기여할 수 있음을 보였다.

## Ⅱ. 본 론

### 2.1 순서화된 드롭아웃(Ordered Dropout: OD)

OD는 [3]에서 제안하는 FjORD 알고리즘에서 활용하는 드롭아웃 방법이다. 기존의 학습 과정에서 과대적합을 막고자 제안된 RD에서는 개별 노드에 대해 임의로 드롭아웃을 진행한다. 하지만 OD에서는 RD와 다르게 특정 기준(노드 순서, 채널 순서 등)에 따라 순차적으로 드롭아웃을 진행하여 전체 모델에서 크기가 작은

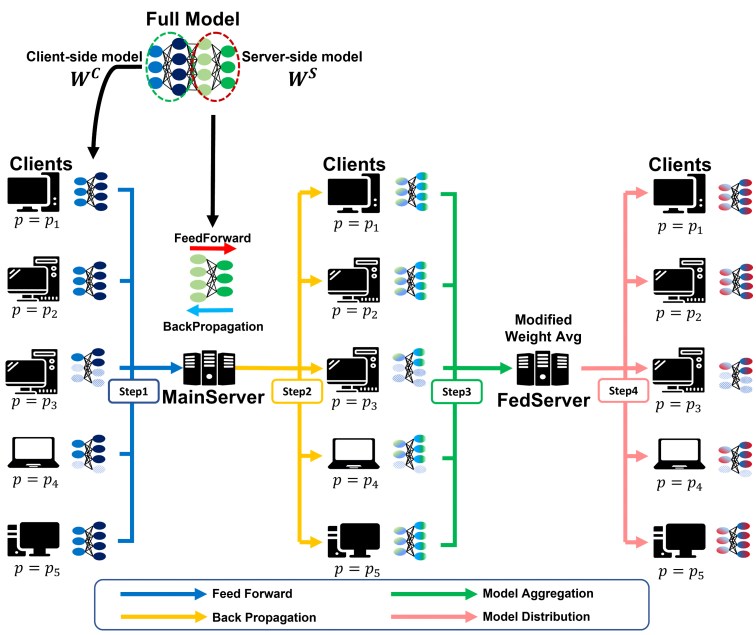


그림 1. 시스템 모델과 훈련 과정
Fig. 1. System model and training procedure

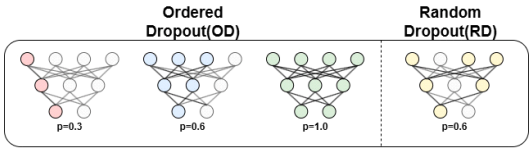


그림 2. Ordered Dropout과 Random Dropout 비교
Fig. 2. Comparison of Ordered Dropout and Random Dropout

효과적인 서브모델을 추출할 수 있도록 한다. Figure 2에서 $p$는 노드의 드롭아웃 비율을 나타내며 각 텐서의 채널에서 드롭아웃 이후 남는 노드의 비율을 의미한다. 본 논문에서 제안한 HeteroSplitFed에서는 채널 기준 OD를 적용하였으며 클라이언트 부분 모델에 해당하는 레이어 전체에 적용하게 된다.

### 2.2 시스템 모델

본 연구에서의 시스템 모델은 Figure 1과 같이 구성하였다. 서버 역할은 MainServer와 FedServer로 나눌 수 있으며, 둘은 하나의 서버에서 역할을 분리하거나 개별적인 두 서버를 둘 수도 있다. 전체 모델 $W$를 분할 레이어를 기준으로 클라이언트 부분 모델 $W^C$와 서버 부분 모델 $W^S$로 분할한다.

서버는 매 라운드마다 클라이언트 부분 모델 $W^C_{p_k}$을 클라이언트에 전달한다. 클라이언트는 전달받은 서브 모델로 개별 학습 데이터셋 $X_k$에 대한 FeedForward 과정을 수행하고, 분할 레이어까지의 FeedForward 결과인 $A_k$를 MainServer로 전달한다. MainServer는 클라이언트에서 전달받은 $A_k$를 사용하여 서버 부분 모델 $W^S$로 FeedForward 과정을 수행한다. FeedForward 결과인 모델의 출력 $\hat{Y}_k$와 학습 데이터 라벨 $Y_k$을 통해 손실을 계산한다. 계산된 손실을 통해 모델의 파라미터 업데이트를 위한 역전파(backpropagation)를 분할 레이어까지 수행한다. 분할 레이어에서의 그래디언트(gradient)를 클라이언트에 전달하며, 클라이언트는 클라이언트 부분 모델의 파라미터 업데이트를 위한 역전파를 수행한다. 모든 클라이언트는 클라이언트 부분 모델에 대한 역전파 결과를 FedServer로 전달하고, FedServer는 클라이언트 부분 모델 업데이트 정보를 수합(aggregation)하여 전역 클라이언트 부분 모델의 업데이트를 수행한다. 이를 통해 한 번의 학습 라운드가 종료되며, 앞선 과정을 반복함으로써 전체 모델에 대한 학습이 진행된다.

### 2.3 HeteroSplitFed 알고리즘

HeteroSplitFed의 전체 알고리즘은 알고리즘 1과 같다. Algorithm 1에서 수행되는 ClientUpdate와 ClientBackProp 알고리즘은 Algorithm 2에서 표현하였다. MainServer, FedServer 및 클라이언트에서 수행되는 학습 과정을 나누어 설명하고자 한다. Table 1은 Algorithm 1과 Algorithm 2에서 사용된 변수명을 정리하였다.

MainServer는 매 라운드마다 라운드 $t$에서의 서버 부분 모델 $W^S_t$을 초기화한다. 클라이언트는 클라이언트 $k$의 라운드 $t$에서의 부분 모델 $W^C_{k,t,p_k}$의 FeedForward 과정의 결과인 $A_{k,t,p_k}$를 MainServer로 전달한다. $A_{k,t,p_k}$의 크기는 클라이언트 드롭아웃 비율 $p_k$에 따라 서로 다르므로 서버 부분 모델 $W^S_t$의

표 1. 변수 표기
Table 1. Notations

| *Notation* | *Description* |
|---|---|
| $R$ | The total training rounds |
| $E$ | The local training epochs of client $k$ |
| $C$ | The set of clients |
| $W^C_t$ | The client-side model at round $t$ |
| $W^S_t$ | The server-side model at round $t$ |
| $Y_k$ | The true labels of the client $k$ |
| $\hat{Y}_k$ | The predicted labels of the client $k$ |
| $\nabla \ell_k$ | The gradient of the loss for the client $k$ |
| $A_{k,t}$ | The smashed data of client $k$ at $t$ |
| $Z_{k,t}$ | The tensor for zero padding in server-side model |
| $V^C_{k,t}$ | The model for zero padding with client-side model of the client $k$ |
| $p_k$ | The odrdered dropout rate for each client $k$ |
| $\eta$ | The step size factor for model update |
| $X_k$ | The dataset in each client $k$ |

**Algorithm 1** HeteroSplitFed

**MainServer:**
**for** each Round $t = 1, 2, \dots R$ **do**
  Initialize $W_t^S$ server side model
  **for** each client $k \in C$ **do**
    **for** each local epoch $e = 1, 2, \dots E$ **do**
      $A_{k,t,p_k} \leftarrow$ **ClientUpdate**$(W_{k,t,p_k}^C)$
      $\hat{A}_{k,t,p_k} \leftarrow A_{k,t,p_k} \cup Z_{k,t}[:, :, p_k :]$ (zero-padding)
      $\hat{Y}_k \leftarrow$ **ServerForward**$(W_t^S)$
      Loss calculation with and $Y_k$ and $\hat{Y}_k$
      Calculate Back-propagation $\nabla\ell(W_t^S)$
      Send $\nabla\ell(W_t^S)$ to client $k$ for **ClientBackprop**
      Server-side model update : $W_{t+1}^S \leftarrow W_t^S - \eta\nabla\ell(W_t^S)$
    **end for**
  **end for**
**end for**

**FedServer:**
**for** each Round $t = 1, 2, \dots R$ **do**
  Initialize $W_t^C$ (global client model)
  **for** each client $k \in C$ **do**
    Initialize model $V_{k,t}^C$ with the same size as $W_{k,t,p_k}^C$
    **if** $|W_{k,t,p_k}^C| \neq |V_t^C|$ **then**
      $V_{k,t,1-p_k}^C \leftarrow V_{k,t}^C[:, :, p_k :]$
      $W_{k,t}^C \leftarrow W_{k,t,p_k}^C \cup V_{k,t,1-p_k}^C$ (zero-padding)
    **end if**
  **end for**
  **for** each layer $l_i \in W_t^C$ **do**
    **for** each weight $w_{i,j} \in l_i$ **do**
      $w_{i,j} \leftarrow \frac{\sum_{k \in C} w_{i,j}^k \cdot 1(w_{i,j}^k \neq 0)}{\sum_{k \in C} 1(w_{i,j}^k \neq 0)}$
    **end for**
  **end for**
**end for**

알고리즘 1. HeteroSplitFed 알고리즘
Algorithm 1. HeteroSplitFed Algorithm

FeedForward를 위한 입력에 맞추기 위해 제로-패딩(zero-padding)으로 크기를 조정한다. MainServer는 $A_{k,t,p_k}$를 이용해 서버 부분 모델에 대한 FeedForward 과정인 ServerForward를 수행하며, 그 결과인 모델 출력 $\hat{Y}_k$와 데이터 라벨 $Y_k$를 통해 손실을 계산한다. 이를 통해 서버 부분 모델 $W_t^S$에 대한 역전파를 진행하고 분할 레이어에서의 그래디언트를 클라이언트에 전달한다. 클라이언트는 클라이언트 부분 모델에 대해 역전파를 수행하여 클라이언트 부분 모델을 업데이트한다. MainServer는 서버 부분 모델 $W_t^S$에 대한 업데이트를 진행하고 라운드를 종료한다.

FedServer는 매 라운드마다 라운드 $t$에서의 전역 클라이언트 부분 모델 $W_t^C$를 초기화한다. FedServer는 클라이언트 $k$의 라운드 $t$에서의 부분 모델 $W_{k,t,p_k}^C$이 클라이언트마다 서로 다른 크기를 가지기 때문에 모델 수합 전에 크기 조정 과정을 거친다. $W_t^C$와 크기가 같고 모든 가중치가 0인 모델 $V_t^C$를 정의하고 각 클라이언트에 대해 $W_{k,t,p_k}^C$와 $V_t^C$의 크기가 다른 경우 $W_{k,t,p_k}^C$에서 드롭아웃한 부분을 채우기 위해 이와 같은 크기의

**Algorithm 2** ClientUpdate and ClientBackProp

**ClientUpdate:**
Model updates $W_{k,t,p_k}^C \leftarrow$ **FedServer**()
$W_{k,t,p_k}^C \leftarrow W_{k,t}^C[:, :, : p_k]$
Forward propagation with dataset $X_k$
$A_{k,t,p_k} \leftarrow$ **ClientForward**$(W_{k,t,p_k}^C, X_k)$
Send $A_{k,t,p_k}$ to the **MainServer**
Wait for the completion of **ClientBackprop**()

**ClientBackprop:**
$\nabla\ell_k(W_t^S) \leftarrow$ **MainServer**()
Calculate Back-propagation $\nabla\ell_k(W_{k,t,p_k}^C)$
Update $W_{t,k,p_k}^C \leftarrow W_{t,k,p_k}^C - \eta\nabla\ell_k(W_{k,t,p_k}^C)$
Send $W_{k,t,p_k}^C$ to the **FedServer**

알고리즘 2. ClientUpdate와 ClientBackProp
Algorithm 2. ClientUpdate and ClientBackProp

텐서를 $W_{k,t,p_k}^C$에 제로-패딩하며, 이를 모든 레이어에 대해 적용한다. 클라이언트 부분 모델 수합 및 업데이트 과정에서는 제로-패딩을 통해 채워진 채널이 모델 수합에 영향을 주지 않도록 하기 위해 변형된 파라미터 평균화(averaging) 방법을 사용한다. 즉, 실제 모델 학습을 통해 얻은 파라미터 업데이트만을 클라이언트 부분 모델 업데이트에 사용한다. FedServer에서의 제로-패딩 및 변형된 파라미터 평균화 과정은 Figure 3으로 표현하였다.

클라이언트는 매 라운드마다 FedServer에서 업데이트된 클라이언트 부분 모델을 전달받아 클라이언트의 컴퓨팅 능력을 고려하여 클라이언트 부분 모델에 비율 $p_k$로 OD를 적용한 서브 모델 $W_{k,t,p_k}^C$를 생성한다. 생성한 $W_{k,t,p_k}^C$를 통해 데이터 $X_k$에 대한 FeedForward 과정을 수행하며, 분할 레이어에서의 출력인 $A_{k,t,p_k}$를 MainServer로 전달한다. 클라이언트는 ClientBackprop가 종료될 때까지 대기한다.

클라이언트 부분 모델의 역전파를 위해 우선 MainServer의 서버 부분 모델에 대한 역전파 결과를 전달받는다. 전달받은 분할 레이어에서의 그래디언트

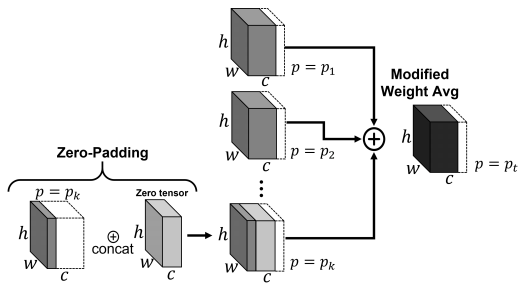


그림 3. FedServer에서의 제로-패딩과 가중치 평균화 과정
Fig. 3. Zero-padding and weight averaging process in FedServer

를 이용해 클라이언트 부분 모델에 대한 역전파를 진행하고, 이를 통해 각 클라이언트의 지역 클라이언트 부분 모델을 업데이트한다. 업데이트한 지역 클라이언트 부분 모델을 FedServer로 전달한다.

## Ⅲ. 실 험

### 3.1 시뮬레이션 환경

시뮬레이션은 CIFAR-10 데이터셋에 대한 이미지 분류 성능을 확인하는 방식으로 진행하였다. 디리클레 분포를 사용하여 전체 데이터셋을 개별 클라이언트에 분배하였고, 디리클레 분포의 $\alpha$값을 조정하여 클라이언트별 데이터 분포가 이질적인 경우와 균질한 경우를 모두 가정하였다. Figure 4는 $\alpha$값에 따른 클라이언트별 데이터 라벨 분포를 클라이언트 수가 5인 상황에 대해 나타내었다. 클라이언트의 수는 5 또는 10으로 설정하였고, 클라이언트별 드롭아웃 비율 $p$는 Table 2와 같이 설정하였다. 전체 모델 학습은 100 라운드까지 진행하였으며 지역 클라이언트 부분 모델 학습을 위한 local epoch은 10으로 설정하였다. 전체 모델은 ResNet50으로 설정하였으며 Figure 5와 같이 클라이언트 부분 모델과 서버 부분 모델을 분할하였다. 손실 함수로 Cross Entropy 손실을 사용하였고, Adam optimizer를 파라미터 최적화를 위한 옵티마이저로 사용하였다. 배치 크기는 256, 학습률은 0.0001로 설정하여 진행하였다.

### 3.2 시뮬레이션 결과

Table 2은 다양한 클라이언트 수, 클라이언트 간 데이터 분포의 차이, 클라이언트의 드롭아웃 비율에 따른 FedAvg[1]와 HeteroSplitFed의 모델 정확도를 나타낸다. 100 라운드까지의 모델 학습 과정에서 검증 데이터에 대한 가장 높은 정확도를 표시하였다. 성능 비교를 위해 클라이언트 수가 5이고 $\alpha = 10$일 때 FedAvg의 성능을 확인하였다. 클라이언트 부분 모델의 드롭아웃 비율이 [0.8, 0.8, 0.8, 0.8, 0.8]일 때, 클라이언트의 컴퓨팅 능력이 모두 균일하고 충분한 상황에서 학습한 결


그림 4. 5개 클라이언트의 CIFAR-10 데이터셋 분포
Fig. 4. CIFAR-10 dataset distribution with 5 clients

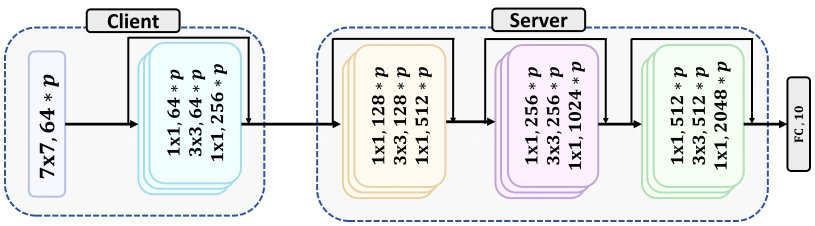


그림 5. ResNet50의 클라이언트 부분 모델과 서버 부분 모델
Fig. 5. ResNet50 with client-side model and server-side model

과로 64.887%의 높은 정확도를 보였다. 드롭아웃 비율이 [0.2, 0.2, 0.2, 0.2, 0.2]인 경우 45.874%의 정확도를 보였다. 이는 클라이언트의 컴퓨팅 능력이 부족한 경우 전체 모델에 대한 충분한 학습이 이루어지지 않음을 뜻한다. 또한 서로 다른 컴퓨팅 능력을 갖춘 클라이언트가 혼재한 경우 높은 컴퓨팅 능력을 가진 클라이언트의 학습 결과로 인해 정확도 향상을 얻을 수 있지만, 충분히 효과적이지 않음을 확인하였다.

HeteroSplitFed은 클라이언트 수가 5이고 $\alpha = 10$일 때, 드롭아웃 비율이 [0.8, 0.8, 0.8, 0.8, 0.8]인 경우 FedAvg와 유사한 정확도를 달성하였다. 클라이언트가 상대적으로 제한된 컴퓨팅 능력을 갖춘 상황을 나타내는 드롭아웃 비율 [0.2, 0.2, 0.2, 0.2, 0.2]과 [0.4, 0.4, 0.4, 0.4, 0.4]인 상황에서는 FedAvg와 비교해 더 높은 정확도를 달성하였으며, 이는 클라이언트의 컴퓨팅 능력이 제한된 상황에서 서버의 도움을 통해 더 뛰어난 모델을 얻을 수 있음을 뜻한다. 서로 다른 컴퓨팅 능력을 가진 클라이언트가 혼재하는 경우, OD를 적용하여 낮은 컴퓨팅 능력을 가진 클라이언트도 충분히 모델 학습에 기여를 할 수 있는 성질을 가져 모델 성능의 향상을 꾀할 수 있음을 확인하였다. $\alpha = 1$인 경우 비균일한 데이터 분포의 영향으로 $\alpha = 10$일 때에 비해 전체적으로 모델 성능이 저하되는 결과를 보였다. OD를 적용하지 않은 기존 SplitFed 알고리즘[7]의 경우 모든 클라이언트가 같은 크기의 클라이언트 부분 모델을 학습하는 상황을 가정하므로 모든 클라이언트가 가장 낮은 컴퓨팅 능력을 가지는 상황에서의 성능을 얻을 것으로 예상할 수 있다. 따라서, HeteroSplitFed에서 서로 다른 컴퓨팅 능력을 가진 클라이언트가 혼재하는 경우의 성능과 모두 가장 낮은 컴퓨팅 능력을 가지는 경우의 성능을 비교함으로써 HeteroSplitFed의 우수성을 확인할 수 있다.

클라이언트 수가 10일 때, 클라이언트 수가 5이고 같은 드롭아웃 비율을 갖춘 경우와 비교하였을 때 클라이언트 수 증가에 따른 일부 모델 성능 향상을 확인할 수 있고, 데이터 분포가 비균일할수록 클라이언트 수 증가에 따라 높은 성능 향상을 보였다. 또한, 클라이언트 수가 증가하였을 때도 이전과 마찬가지로 클라이언트의 드롭아웃 비율이 서로 다르더라도 낮은 컴퓨팅 능력을 가진 클라이언트가 학습에 효과적으로 참여함으로써 모델 성능 향상에 기여함을 확인하였다.

Figure 6-1에서 Figure 6-5은 다양한 클라이언트 수, 클라이언트 간 데이터 분포의 차이에 대해 라운드 진행에 따른 모델 정확도를 나타낸다. 주어진 상황에서 대부분 10라운드 이내에 빠른 정확도 상승을 보여 이후 모델 성능이 수렴하는 경향을 보였다. Figure 6-1에서의 결과를 보면 클라이언트의 컴퓨팅 능력이 충분한 상황인 [0.8, 0.8, 0.8, 0.8, 0.8]인 경우에 빠르게 모델 성능이 수렴하는 경향을 보였으며, 클라이언트의 컴퓨팅 능력이 서로 다른 경우 상대적으로 느리게 수렴하는 것을 확인하였다. Figure 6-2에서도 마찬가지로 클라이언트의 컴퓨팅 능력이 충분한 [0.8, 0.8, 0.8, 0.8, 0.8]인 상황에서 가장 높은 정확도를 보였으며, 컴퓨팅 능력이 서로 다른 경우 상대적으로 느리게 수렴하는 경향을 보여 수렴 속도에 대한 이점은 보이지 않았다. Figure 6-3에서는 데이터의 분포가 non-i.i.d.인 상황에서 i.i.d.한

표 2. 시뮬레이션 결과
Table 2. Simulation results

| $p$ | Accuracy | | |
|---|---|---|---|
| | FedAvg[1] | HeteroSplitFed | |
| | $\alpha = 10$ | | $\alpha = 1$ |
| [0.8, 0.8, 0.8, 0.8, 0.8] | 64.887% | 63.707% | 57.398% |
| [0.4, 0.4, 0.4, 0.4, 0.4] | 55.968% | 59.499% | 51.779% |
| [0.8, 0.8, 0.8, 0.2, 0.2] | 50.587% | 56.818% | 51.226% |
| [0.8, 0.2, 0.2, 0.2, 0.2] | 46.100% | 51.630% | 45.914% |
| [0.2, 0.2, 0.2, 0.2, 0.2] | 45.874% | 50.896% | 39.101% |
| [0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8] | | 64.801% | 61.475% |
| [0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.2, 0.2, 0.2, 0.2] | | 56.203% | 51.782% |
| [0.8, 0.8, 0.8, 0.8, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2] | | 51.757% | 48.122% |
| [0.8, 0.8, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2] | | 52.118% | 48.542% |
| [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2] | | 52.486% | 48.341% |

상황보다 정확도가 낮은 경향을 보이며, 컴퓨팅 능력이 서로 달라도 수렴 속도의 차이는 없는 것을 알 수 있다. Figure 6-4와 Figure 6-5는 클라이언트 수가 10인 경우 라운드 진행에 따른 모델 정확도를 나타내며, 학습 수렴 속도 측면에서 이전 결과와의 차이점은 발견하지 못하였다.

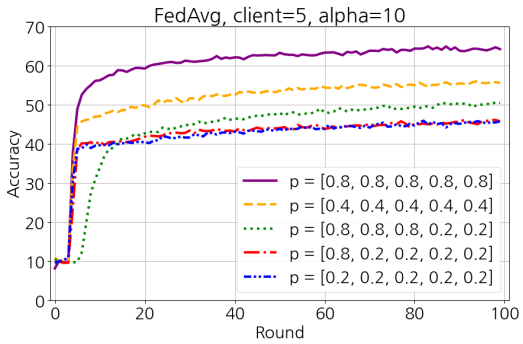


그림 6-1. FedAvg의 정확도, clients=5, $\alpha = 10$
Fig. 6-1. Classification accuracy of FedAvg with clients=5, $\alpha = 10$

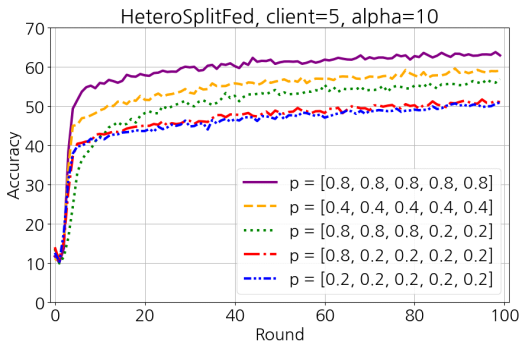


그림 6-2. HeteroSplitFed의 정확도, clients=5, $\alpha = 10$
Fig. 6-2. Classification accuracy of HeteroSplitFed with clients=5, $\alpha = 10$

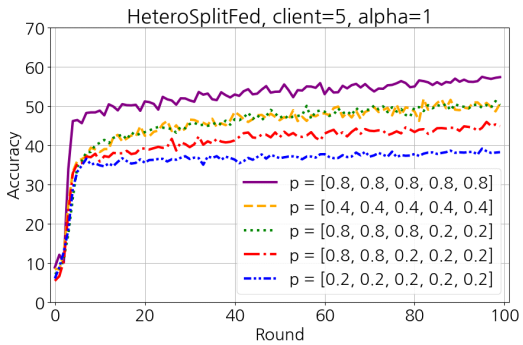


그림 6-3. HeteroSplitFed의 정확도, clients=5, $\alpha = 1$
Fig. 6-3. Classification accuracy of HeteroSplitFed with clients=5, $\alpha = 1$

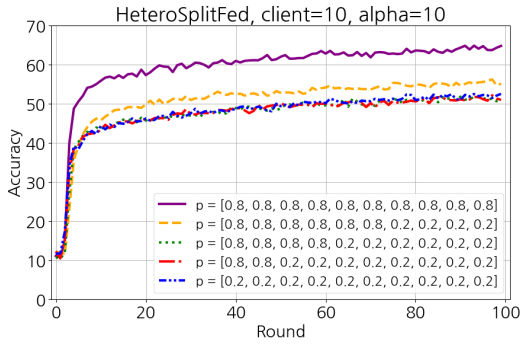


그림 6-4. HeteroSplitFed의 정확도 ,clients=10, $\alpha = 10$
Fig. 6-4. Classification accuracy of HeteroSplitFed with clients=10, $\alpha = 10$

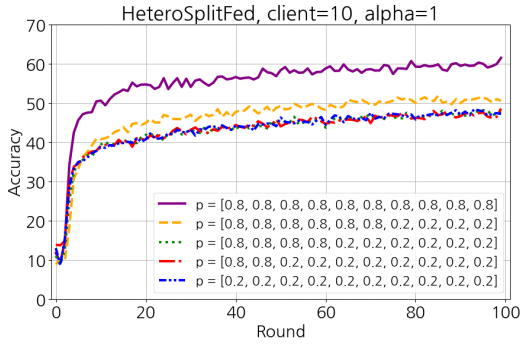


그림 6-5. HeteroSplitFed의 정확도, clients=10, $\alpha = 1$
Fig. 6-5. Classification accuracy of HeteroSplitFed with clients=10, $\alpha = 1$

## Ⅳ. 결 론

본 연구에서는 분산학습 시스템에 참여하는 클라이언트의 컴퓨팅 능력이 제한되고 서로 이질적인 상황에서 이를 고려한 분산학습 알고리즘 HeteroSplitFed를 제안하였다. 이는 SplitFed 알고리즘에 서로 다른 크기를 갖춘 클라이언트 부분 모델을 얻기 위해 OD를 적용하였고, 다양한 실험을 통해 클라이언트의 컴퓨팅 능력이 제한되고 이질적이더라도 모든 클라이언트가 효과적으로 모델 학습에 참여함으로써 전체 모델의 성능 향상에 기여할 수 있음을 확인하였다. 본 연구를 통해 다양한 모바일 기기가 참여할 수 있는 분산학습 시스템을 구성할 방안을 제시하며, 분산학습 시스템이 차세대 이동통신 시스템, 무선 네트워크 등을 비롯한 다양한 환경에서 구현될 수 있는 토대를 마련하였다[15].

## References

[1] H. B. McMahan, E. Moore, D. Ramage, S.

Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Int. Conf. AISTATS*, 2017. (https://doi.org/10.48550/arXiv.1602.05629)

[2] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, “*Split learning for collaborative deep learning in healthcare*,” (2019), Retrieved Sep. 3, 2024, from https://arxiv.org/abs/1912.12115 (https://doi.org/10.48550/arXiv.1912.12115)

[3] S Horváth, S. Laskaridis, M. Almeida, I. Leontiadis, S. I. Venieris, and N. D. Lane, “FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout,” in *Advances in NIPS(NeurIPS)*, 2021. (https://doi.org/10.48550/arXiv.2102.13451)

[4] M. Kim, S. Yu, S. Kim, and S.-M. Moon, “DepthFL : Depthwise federated learning for heterogeneous clients,” in *ICLR*, 2023. (https://openreview.net/forum?id=pf8RIZTMU58)

[5] D. Yao, S. Tang, X. Zheng, Y. Zhao, and Y. Gu, “*FedHM: Efficient federated learning for heterogeneous models via low-rank factorization*,” (2021), Retrieved Nov. 19, 2024, from https://arxiv.org/abs/2111.14655. (https://doi.org/10.48550/arXiv.2111.14655)

[6] S. Alam, S. Jiang, A. Al-Aradi, M. Fatourechi, and A. Wong, “*Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction*,” (2022), Retrieved Nov. 19, 2024, from https://proceedings.neurips.cc/paper_files/paper/2022/hash/fedrolex (https://doi.org/10.48550/arXiv.2212.01548)

[7] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, “Splitfed: When federated learning meets split learning,” in *Proc. AAAI Conf. Artificial Intell.*, pp. 8485-8493, Vancouver, Canada, Jun. 2022. (https://doi.org/10.48550/arXiv.2004.12088)

[8] H. Kang, S. Cha, J. Shin, J. Lee, and J. Kang, “*NeFL: Nested federated learning for heterogeneous clients*,” (2023), Retrieved Sep. 3, 2024, from https://arxiv.org/abs/2308.07761. (https://doi.org/10.48550/arXiv.2308.07761)

[9] J. Shin, J. Ahn, H. Kang, and J. Kang, “*FedSplitX: Federated split learning for computationally-constrained heterogeneous clients*,” (2023), Retrieved Sep. 3, 2024, from https://arxiv.org/abs/2310.14579 (https://doi.org/10.48550/arXiv.2310.14579)

[10] Z. Lin, G. Qu, W. Wei, X. Chen, and K. K. Leung, “Adaptsfl: Adaptive split federated learning in resource-constrained edge networks,” (2024), Retrieved Sep. 3, 2024, from https://arxiv.org/abs/2403.13101 (https://doi.org/10.48550/arXiv.2403.13101)

[11] G. Zhu, Y. Deng, X. Chen, H. Zhang, Y. Fang, and T. F. Wong, “ESFL: Efficient split federated learning over resource-constrained heterogeneous wireless devices,” *IEEE Int. Things J.*, vol. 11, no. 1, pp. 123-135, Jan. 2024. (https://doi.org/10.48550/arXiv.2402.15903)

[12] G. Yang, K. Mu, C. Song, Z. Yang, and T. Gong, “*Ringfed: Reducing communication costs in federated learning on non-iid data*,” (2021). Retrieved Sep. 3, 2024, from https://arxiv.org/abs/2107.08873. (https://doi.org/10.48550/arXiv.2107.08873)

[13] Z. Y. Chai, C. D. Yang, and Y. L. Li, “Communication efficiency optimization in federated learning based on multi-objective evolutionary algorithm,” *Evolutionary Intell.*, vol. 16, no. 3, pp. 1033-1044, Sep. 2023. (https://doi.org/10.1016/j.eswa.2022.116963)

[14] J. Ryu and H. Yang, “HeteroSplitFed: A study on distributed learning techniques for heterogeneous client settings on computing capability and communcation bandwidth,” in *Proc. KICS Winter Conf.*, pp. 1229-1230, Feb. 2024. (https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE11737563)

[15] H. Lee, B. Lee, H. Yang, J. Kim, S. Kim, W. Shin, B. Shim, and H. V. Poor, “Towards 6G hyper-connectivity: Vision, challenges, and

key enabling technologies," *KICS JCN*, vol. 25, no. 3, pp. 344-354, Jun. 2023. (https://doi.org/10.48550/arXiv.2301.11111)

**류 지 현** (Ji-Hyun Ryu)

2024년 : 충남대학교 컴퓨터공학과 졸업
2024년~현재 : 충남대학교 컴퓨터공학과 석사과정
<관심분야> 분산학습
[ORCID:0009-0004-2633-3740]

**양 희 철** (Heecheol Yang)

2013년 : 서울대학교 공학사
2018년 : 서울대학교 공학박사
2018년~2019년 : 삼성전자 삼성리서치 표준연구팀 Staff Engineer
2019년~2021년 : 금오공과대학교 전자공학부 조교수
2021년~현재 : 충남대학교 컴퓨터융합학부 조교수
<관심분야> 무선통신, 분산컴퓨팅, 분산학습
[ORCID:0000-0002-2802-2143]