한국정보전자통신기술학회논문지(jkiiect)'24-8, Vol.17 No.4
논문 24-17-04-213 http://dx.doi.org/10.17661/jkiiect.2024.17.4.213

WASM을 활용한 디바이스 및 엣지 클라우드 기반 Federated Learning의 최적화 방안

최종석*

Optimization Strategies for Federated Learning Using WASM on Device and Edge Cloud

Jong-Seok Choi*

요 약 본 논문에서는 WebAssembly(WASM)를 활용하여 디바이스와 엣지 클라우드 간의 Federated Learning을 수행하는 최적화 전략을 제안한다. 제안된 전략은 일부 학습을 디바이스에서 수행하고, 나머지 학습을 엣지 클라우드에서 수행하여 효율성을 극대화하는 것을 목표로 한다. 특히, GPU 메모리 세그먼트 간 데이터 이동과 연산 작업의 중첩을 최적화하여 전체 학습 시간을 줄이고 GPU 사용률을 향상시키는 방법을 수학적으로 설명하고 평가한다. 다양한 실험 시나리오를 통해 비동기 데이터 전송과 연산 중첩이 학습 시간을 단축하고 GPU 사용률을 향상시키며, 모델 정확도를 증가시킴을 확인하였다. 모든 최적화 기법을 적용한 시나리오에서 학습 시간이 47% 단축되었고, GPU 사용률은 91.2%로 향상되었으며, 모델 정확도는 89.5%로 증가함을 확인하여 비동기 데이터 전송과 연산 중첩이 데이터 전송을 기다리는 GPU 유휴 시간을 줄이고, 병목 현상을 완화할 수 있음을 확인하였다. 본 연구는 향후 Federated Learning 시스템의 성능 최적화에 기여할 수 있을 것으로 사료된다.

Abstract This paper proposes an optimization strategy for performing Federated Learning between devices and edge clouds using WebAssembly (WASM). The proposed strategy aims to maximize efficiency by conducting partial training on devices and the remaining training on edge clouds. Specifically, it mathematically describes and evaluates methods to optimize data transfer between GPU memory segments and the overlapping of computational tasks to reduce overall training time and improve GPU utilization. Through various experimental scenarios, we confirmed that asynchronous data transfer and task overlap significantly reduce training time, enhance GPU utilization, and improve model accuracy. In scenarios where all optimization techniques were applied, training time was reduced by 47%, GPU utilization improved to 91.2%, and model accuracy increased to 89.5%. These results demonstrate that asynchronous data transfer and task overlap effectively reduce GPU idle time and alleviate bottlenecks. This study is expected to contribute to the performance optimization of Federated Learning systems in the future.

Key Words : Artificial Intelligence, Edge Cloud, Federated Learning, WASM, WebAssembly

1. 서론

Federated Learning은 여러 디바이스가 로컬 데이터를 사용하여 개별적으로 모델을 학습한 후, 학습된 모델 파라미터를 중앙 서버에 집계하여 글로벌 모델을

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the National Program for Excellence in SW) (2024-0-00071) supervised by the IITP(Institute of Information & communications Technology Planning & Evaluation)
* Spartan Software Educational Institute, Soongsil University
Received July 31, 2024 Revised August 07, 2024 Accepted August 12, 2024

업데이트하는 분산 학습 기법이다. 이는 데이터 사생활 보호와 데이터 전송 비용 절감의 이점을 제공한다. 디바이스가 로컬 데이터를 외부로 전송하지 않고, 모델 파라미터만 전송함으로써 데이터의 프라이버시를 보호할 수 있다. 또한, 네트워크 대역폭을 효율적으로 사용하여 통신비용을 줄일 수 있다. 그러나 Federated Learning은 여러 가지 문제를 가지고 있다. 디바이스의 자원이 제한되어 있고, 데이터 전송과 연산 작업 간의 병목 현상이 발생할 수 있다. 특히, 데이터 이동과 연산 작업이 겹치지 않으면 GPU의 사용률이 떨어지고, 전체 학습 시간이 늘어나는 문제가 발생한다. 이러한 문제를 해결하기 위해서는 디바이스와 엣지 클라우드 간의 효율적인 연산 분담과 최적화가 필요하다. 이에따라, 본 논문에서는 디바이스와 엣지 클라우드 간의 효율적인 연산 분담과 최적화를 통해 Federated Learning의 성능을 향상시키고자 한다. GPU의 메모리 세그먼트 간 데이터 이동과 연산 작업의 중첩을 극대화하여 전체 학습 시간을 줄이고 GPU 사용률을 향상시키는 방법을 제안하고자 하며, 이를 통해 데이터 전송을 기다리는 GPU 유휴 시간을 줄이고, 모델 학습 속도와 정확도를 개선할 수 있다.

이를 통해, WASM을 활용하여 디바이스에서 일부 학습을 수행하고, 나머지 학습을 엣지 클라우드에서 수행하는 최적화 전략을 제안한다. WASM은 다양한 플랫폼에서 고성능 연산을 가능하게 하며, 이를 통해 디바이스의 연산 능력을 최대한 활용할 수 있다. 또한, CUDA 스트림과 통합 메모리(UVM)를 활용하여 연산과 데이터 전송을 병렬로 수행함으로써 최적화를 달성하고 다양한 실험 시나리오를 통해 제안된 최적화 전략의 성능을 평가하고, Federated Learning의 효율성을 극대화할 수 있는 방법을 제시한다. 이를 통해 제안된 방법의 실질적인 효과를 입증하고, 향후 연구 방향을 모색한다.

2. 관련 연구

2.1 WASM

WASM(WebAssembly)는 웹 브라우저에서 실행되는 저수준 이진 형식의 가상 기계로, 다양한 플랫폼에서 고성능 연산을 가능하게 한다. 최근에는 WASM을 활용하여 Federated Learning의 성능을 향상시키기 위한 연구들이 활발히 진행되고 있다. WASM은 이식성측면에서 WASM은 다양한 플랫폼에서 동일한 코드가 실행될 수 있어, 디바이스 간의 호환성을 보장한다. 또한, WASM은 네이티브 코드에 근접한 성능을 제공한다. 연산 집약적인 작업에 적합하며, 샌드박스 환경에서 실행되어 메모리 격리를 통해 악의적인 버그나 메모리 침범으로 인한 시스템 영향도를 낮출 수 있고 격리 및 제한된 실행권한을 가질 수 있어 리소스에 직접 접근할 수 없기 때문에 보안성을 높일 수 있다. WASM은 이러한 장점을 기반으로 표 1과 같은 연구가 진행되고 있다.

표 1. WASM 관련 기술적 연구
Table 1. Technical Research Using WASM

Game Engine Optimization	Optimizing Game Engine Performance with WASM
	Cross-Platform Game Development with WASM
Machine Learning Model Deployment	Deploying Machine Learning Models with WASM
	Lightweight Machine Learning Inference with WebAssembly
Blockchain Technology	Optimizing Smart Contracts with WebAssembly
	Blockchain WebAssembly Runtime

Lin et al. (2021)[1]는 WASM을 사용하여 웹 기반 게임 엔진의 성능을 최적화하는 연구를 수행하였다. WASM의 빠른 실행 속도와 낮은 오버헤드를 활용하여 복잡한 물리 연산과 그래픽 처리를 효율적으로 수행하였다. Smith et al. (2022)[2]는 WASM을 활용하여 게임 개발 프로세스를 간소화하고, 다양한 플랫폼에서 동일한 게임을 실행할 수 있는 Cross-Platform Game Development with WASM을 제안하였다. Patel et al. (2021)[3]는 WASM을 활용하여 웹 브라우저에서 머신러닝 모델을 실행하는 Deploying Machine Learning Models with WASM을 제안하였다. 이를 통해 클라이언트 측에서 머신러닝 모델을 실시간으로 실행할 수 있으며, 서버 부하를 줄이고 응답 속도를 향상시켰다. Chen et al. (2020)[4]는 WASM을 사용하여 경량화된 머신러닝 모델을 웹 브라우저에서 실행하는 Li

ghtweight Machine Learning Inference with WebAssembly를 제안하였다. Nakamoto et al. (2021)[5]는 WASM을 사용하여 블록체인 네트워크에서 스마트 계약의 성능을 최적화하는 연구를 수행하였다. WASM의 빠른 실행 속도와 보안성을 활용하여 스마트 계약의 실행 효율성을 높이고, 네트워크의 전체 성능을 향상시켰다. Li et al. (2022)[6]는 WASM을 활용하여 블록체인 네트워크에서 스마트 계약을 실행하는 Blockchain WebAssembly Runtime을 제안하였다. 이를 통해 스마트 계약의 이식성과 보안성을 강화하고, 다양한 블록체인 플랫폼에서 호환성을 보장하였다.

2.2 Federated Learning

Federated Learning은 데이터가 로컬 디바이스에 분산되어 있는 상황에서 중앙 서버에 데이터를 전송하지 않고, 각 디바이스에서 로컬 모델을 학습한 후 모델 파라미터를 중앙 서버에 전송하여 글로벌 모델을 업데이트하는 분산 학습 기법이다. 이 기법은 데이터 사생활 보호와 데이터 전송 비용 절감의 이점을 제공하며, 기본 프로세스와 장단점은 표 2, 3과 같다.

표 2. Federated Learning Process
Table 2. Federated Learning Process

Initialization	The central server initializes the global model and distributes it to all devices.
Local Training	Each device trains the model on its local data and updates the model parameters.
Transmission and Aggregation	Devices send their updated model parameters to the central server, which aggregates them to update the global model.
Iteration	This process is repeated multiple times to improve the global model's performance.

이러한 장단점을 통해 최근 많은 연구들이 최적화를 위한 다양한 방법을 제안하고 있다. 연합 최적화를 위한 방안으로 Li et al. (2020)[7]은 Federated Learning의 최적화 문제를 해결하기 위해 FedAvg 알고리즘을 제안하였다. 이 알고리즘은 각 디바이스에서 로컬 모델을 일정 기간 학습한 후, 중앙 서버에서 평균을 내어 글로벌 모델을 업데이트하는 방법이다. 이 방법은 통신 비용을 줄이면서도 모델 성능을 유지할 수 있는 장점이 있다. 또한 Park et al. (2021)[8]은 IoT 디바이스에서 WASM과 연계하여 Federated Learning을 수행하는 Federated Learning with WASM for IoT Devices를 제안하였다. 연구에서는 IoT 디바이스의 제한된 자원에서 효율적으로 모델을 학습하고 업데이트할 수 있는 방법을 제시하였다. WASM을 통해 다양한 IoT 디바이스에서 일관된 성능을 유지할 수 있었다. 또한, WASM의 보안적인 측면과 연계하여 Zhang et al. (2022)[9]은 WASM의 샌드박스 환경을 활용하여 Federated Learning의 보안을 강화하는 Secure Federated Learning with WASM을 제안하였다. 이를 통해 학습 과정에서 발생할 수 있는 보안 문제를 최소화하였다. 연구에서는 WASM의 보안 모델을 활용하여 데이터 유출이나 모델 도용을 방지하는 방법을 제시하였다.

표 3. Federated Learning의 장단점
Table 3. Advantages and Disadvantages of Federated Learning

Advantages	Data Privacy Protection	Since local data remains on the devices, data privacy is preserved
	Reduced Communication Costs	model parameters are transmitted, saving network bandwidth
	Scalability	Multiple devices can train models in parallel, making it suitable for large-scale data training
Disadvantages	Limited Device Resource	Devices have limited computational power and memory, making it challenging to train complex models
	Communication Delays	There can be delays in communication between devices and the server
	Data Imbalance	If the data distribution across devices is uneven, it can lead to biased model training

3. WASM기반 디바이스 및 엣지 클라우드 기반 Federated Learning의 최적화

본 논문에서는 WASM을 활용하여 디바이스와 엣지

클라우드 간의 Federated Learning을 수행하는 최적화 전략을 제안한다. 일부 학습을 디바이스에서 수행하고, 나머지 학습을 엣지 클라우드에서 수행하여 효율성을 극대화하는 것이며, 특히, GPU 메모리 세그먼트 간 데이터 이동과 연산 작업의 중첩을 최적화하여 전체 학습 시간을 줄이고 GPU 사용률을 향상시키는 방안을 제안하고자 한다.

3.1 데이터 분할 및 작업 할당

Federated Learning에서 각 디바이스 i는 로컬 데이터를 가지며, 로컬 데이터셋 Di는 다음과 같이 정의된다.

$$D_i = \{(x_j, y_j) | j \in \mathcal{I}_i\}$$

Ii는 디바이스 i에서의 데이터 인덱스 집합이며, 각 디바이스는 로컬 데이터셋 Di 를 사용하여 로컬 모델 θi 를 업데이트하며 아래와 같이 정의된다.

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \nabla_{\theta_i} \mathcal{L}_i(\theta_i^{(t)}; D_i)$$

여기서 η은 학습률이며, L_i는 디바이스 i에서의 손실 함수이다. 각 디바이스에서 업데이트된 모델 파라미터를 엣지 클라우드에서 집계한다. 이를 글로벌 모델 θ에 업데이트는 다음과 같이 정의된다. 여기서 N은 디바이스의 수, n_i 는 디바이스 i의 데이터 샘플 수, n은 전체 데이터 샘플 수이다.

$$\theta^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{n} \theta_i^{(t+1)}$$

3.2 비동기 데이터 전송과 연산 중첩

GPU의 비동기 데이터 전송을 통해 데이터 전송 시간 $T_{transfer}$와 연산 시간 $T_{compute}$를 중첩시켜 전체 시간을 줄이며, 총 소요시간은 다음과 같다.

$$T_{total} = \max(T_{transfer}, T_{compute})$$

이를 위해 CUDA 스트림을 사용하여 데이터 전송과 연산을 비동기적으로 수행한다. CUDA 스트림을 사용하면 메모리 전송과 커널 실행을 병렬로 수행할 수 있어, 데이터 전송 시간과 연산 시간이 겹치지 않게 된다. 이는 다음과 같은 코드로 구현할 수 있다.

표 4. CUDA 비동기 전송과 연산중첩 코드
Table 4. CUDA asynchronous and computation overlap code

Line	Code
1	cudaMemcpyAsync(dst, src, size, cudaMemcpyHostToDevice, stream);
2	matrixMultiplicationKernel<<<grid, block, 0, stream>>>(args);
3	softmaxKernel<<<grid, block, 0, stream>>>(args);

또한, 통합 메모리(UVM, Unified Virtual Memory)를 사용하면 메모리 관리가 단순해지고, 메모리 이동이 자동으로 관리된다. 통합 메모리는 CPU와 GPU 간의 메모리 전송을 자동으로 처리하여 개발자의 부담을 줄이고, 성능 최적화를 돕는다. 이를 통해 GPU의 효율적인 사용을 보장할 수 있다.

표 5. CUDA 통합 메모리 코드
Table 5. CUDA unified virtual memory code

Line	Code
1	cudaMallocManaged(&data, size);
2	matrixMultiplicationKernel<<<grid, block>>>(data);
3	softmaxKernel<<<grid, block>>>(data);
4	cudaDeviceSynchronize();

3.3 교차 수행

행렬 곱셈과 소프트맥스 연산을 교차 수행하여 병목현상을 줄인다. 행렬 곱셈 T_{mm}과 소프트맥스 연산 T_{sm}의 총 소요 시간은 다음과 같다.

$$T_{mm} = \frac{M \cdot K \cdot N}{GPU_{performance-mm}}$$

$$T_{sm} = \frac{N \cdot log \quad (\sum \exp(x))}{GPU_{performance-sm}}$$

교차 수행은 두 가지 연산 간의 데이터 종속성을 최소화하고, GPU 자원을 최대한 활용할 수 있도록 한다. 이를 통해 GPU의 연산 능력을 최대한 끌어올리고, 데이터 전송과 연산 간의 병목 현상을 줄일 수 있다.

3.4 Federated Learning 워크플로우

Federated Learning 워크플로우는 다음과 같은 단계로 이루어진다.

표 6. Federated Learning Workflow
Table 6. Federated Learning Workflow

Local Training	Each device i updates its local model θ_i using its own local dataset D_i
Model Transmission	Each device transmits its local model θ_i to the edge cloud
Global Model Aggregation	The edge cloud aggregates the model parameters received from each device to update the global model θ
Global Model Distribution	The updated global model θ is then distributed back to each device.

로컬 학습은 디바이스의 자원을 최대한 활용하여 데이터의 사생활을 보호하면서 학습을 진행한다. 이후 모델 전송을 통해 나타난 전송된 모델 파라미터는 엣지 클라우드에서 집계되어 글로벌 모델을 업데이트하는 데 사용된다. 글로벌 모델 집계에서 업데이트된 모델은 각 디바이스의 로컬 모델을 조정하여 전체 학습 과정의 일관성을 유지한다. 또한, 각 디바이스는 전송받은 글로벌 모델을 기반으로 다음 학습 단계로 진행된다.

이에 따라 전체 Federated Learning 프로세스의 최적화로 총 학습 시간을 최소화하고, GPU 사용률을 최대화하며, 모델 정확도를 향상시킬 수 있게 되며, 효율적인 연산과 데이터 전송을 통해 최적화된 모델의 성능을 보장 할 수 있다.

4. 성능평가

성능 평가는 학습 시간, GPU 사용률, 모델 정확도, 데이터 전송 시간, 병목 현상 분석의 항목으로 이루어진다. 평가를 위해 PyTorch를 사용하여 구현하고, 다양한 실험 시나리오를 설정하였다. 디바이스는 WASM을 지원하는 디바이스로 제한하며, 엣지 클라우드는 K8s기반의 오케스트레이션 엣지 클라우드를 사용하였다. 또한 CUDA를 통해 런타임 환경을 제공하였다.

실험의 각 시나리오는 표 7와 같으며, 10개의 상이한 클래스를 포함하여 다양한 패턴을 학습하고 일반화하는 능력을 평가하는 데 적합한 CIFAR-10 데이터셋과 복잡한 패턴을 학습하고, 장기적인 종속성을 효과적으로 처리할 수 있으며 GPU 자원을 효율적으로 활용할 수 있는지 확인할 수 있는 Transformer 모델을 통해 성능평가를 진행하였다.

표 7. Performance Evaluation Scenarios
Table 7. Performance Evaluation Scenarios

Scenario 1	All training performed on devices
Scenario 2	Partial training on devices, remaining training on edge cloud
Scenario 3	Training with asynchronous data transfer and computation overlap
Scenario 4	Softmax operation optimization on devices
Scenario 5	Matrix multiplication optimization on edge cloud
Scenario 6	All optimizations applied

4.1 학습시간, GPU사용률, 모델정확도

학습 시간은 디바이스와 엣지 클라우드에서 전체 학습을 완료하는 데 소요된 총 시간을 측정한 것이다. 각 시나리오에서 비동기 데이터 전송과 연산 중첩이 학습 시간에 어떻게 영향을 미치는지를 평가하였다. GPU 사용률은 학습 과정에서 GPU가 얼마나 효율적으로 사용되었는지를 나타낸다. 높은 GPU 사용률은 메모리 전송과 연산 작업의 중첩이 효과적으로 이루어졌음을 의미한다. 모델 정확도는 최종 학습된 Federated Learning 모델의 정확도를 나타낸다. 각 시나리오에서의

모델 학습 성능을 평가하여 엣지 클라우드에서의 추가 학습이 모델 성능에 미치는 영향을 분석하였다.

표 8. 성능평가 결과1
Table 8. Performance Evaluation Results1

Scenario	Time to Train (Sec)	GPU Utilization(%)	Model Accuracy(%)
Scenario 1	1032.45	61.3	85.2
Scenario 2	842.12	71.8	87.1
Scenario 3	623.98	84.7	88.3
Scenario 4	762.34	75.1	86.4
Scenario 5	645.87	79.6	87.8
Scenario 6	547.61	91.2	89.5

학습 시간의 결과 디바이스에서 모든 학습을 수행하는 시나리오 1에서 가장 긴 학습 시간이 소요되었다. 이는 디바이스의 제한된 연산 능력과 자원으로 인한 것이다. 일부 학습을 엣지 클라우드에서 수행한 시나리오 2는 엣지 클라우드의 고성능 연산 자원을 활용하여 학습 시간이 단축되었다. 비동기 데이터 전송과 연산 중첩을 적용한 시나리오 3은 데이터 전송과 연산이 병렬로 수행되어 학습 시간이 더욱 단축되었다. 소프트맥스 연산과 행렬 곱셈의 최적화를 각각 적용한 시나리오 4와 5에서도 학습 시간이 감소하였으며, 모든 최적화 기법을 적용한 시나리오 6에서 가장 짧은 학습 시간이 기록되었다.

GPU 사용률의 결과는 디바이스에서 모든 학습을 수행한 시나리오 1에서는 GPU 사용률이 가장 낮았다. 이는 디바이스의 자원 한계로 인해 GPU가 충분히 활용되지 못했기 때문이다. 일부 학습을 엣지 클라우드에서 수행한 시나리오 2에서는 GPU 사용률이 향상되었다. 비동기 데이터 전송과 연산 중첩을 적용한 시나리오 3에서는 GPU 사용률이 크게 향상되었으며, 이는 데이터 전송과 연산이 병렬로 수행되어 GPU의 유휴 시간이 감소했기 때문이다. 소프트맥스 연산과 행렬 곱셈의 최적화를 각각 적용한 시나리오 4와 5에서도 GPU 사용률이 증가하였다. 모든 최적화 기법을 적용한 시나리오 6에서는 GPU 사용률이 가장 높게 나타났다.

모델의 정확도는 모든 학습을 디바이스에서 수행한 시나리오 1에서는 모델 정확도가 가장 낮았다. 이는 디바이스의 자원 한계와 데이터 분포의 불균형으로 인한 것이다. 일부 학습을 엣지 클라우드에서 수행한 시나리오 2에서는 모델 정확도가 향상되었다. 비동기 데이터 전송과 연산 중첩을 적용한 시나리오 3에서는 모델 정확도가 더욱 향상되었으며, 이는 데이터 전송과 연산의 병렬 수행으로 학습 효율성이 증가했기 때문이다. 소프트맥스 연산과 행렬 곱셈의 최적화를 각각 적용한 시나리오 4와 5에서도 모델 정확도가 증가하였다. 모든 최적화 기법을 적용한 시나리오 6에서는 모델 정확도가 가장 높게 나타났다.

4.2 데이터 전송시간, 병목 현상 분석

데이터 전송 시간은 디바이스와 엣지 클라우드 간의 데이터 전송에 소요된 시간을 측정한 것이다. 실시간 데이터 스트리밍을 통해 데이터 전송 시간이 어떻게 감소하는지를 평가하였다. 병목 현상 분석은 소프트맥스 연산과 행렬 곱셈 연산 간의 병목 현상을 분석하는 것이다. 각 시나리오에서 최적화 전후의 성능 차이를 비교하여 병목 현상을 어떻게 줄일 수 있는지 평가하였다.

표 9. 성능평가 결과2
Table 9. Performance Evaluation Results2

Scenario	Data Transfer Time	Bottleneck Analysis	
		Matrix Multiplication	Softmax
Scenario 1	N/A	403.45	629.87
Scenario 2	52.76	301.23	505.12
Scenario 3	32.15	251.67	372.31
Scenario 4	41.89	351.89	416.45
Scenario 5	37.54	304.76	358.98
Scenario 6	27.83	202.54	305.78

데이터 전송시간은 시간이 측정되지 않은 시나리오 1을 제외하고, 일부 학습을 엣지 클라우드에서 수행한 시나리오 2에서는 데이터 전송 시간이 소요되었다. 비동기 데이터 전송을 적용한 시나리오 3에서는 데이터

전송 시간이 크게 감소하였다. 이는 데이터 전송과 연산이 병렬로 수행되어 전송 시간이 단축되었기 때문이다. 소프트맥스 연산과 행렬 곱셈의 최적화를 각각 적용한 시나리오 4와 5에서도 데이터 전송 시간이 감소하였다. 모든 최적화 기법을 적용한 시나리오 6에서는 데이터 전송 시간이 가장 짧게 나타났다.

모든 학습을 디바이스에서 수행한 시나리오 1에서는 행렬 곱셈과 소프트맥스 연산 모두에서 병목 현상이 크게 나타났다. 일부 학습을 엣지 클라우드에서 수행한 시나리오 2에서는 병목 현상이 일부 완화되었다. 비동기 데이터 전송과 연산 중첩을 적용한 시나리오 3에서는 병목 현상이 많이 감소하였다. 소프트맥스 연산을 최적화한 시나리오 4에서는 소프트맥스 연산의 병목 현상이 크게 감소하였으며, 행렬 곱셈을 최적화한 시나리오 5에서는 행렬 곱셈 연산의 병목 현상이 크게 감소하였다. 모든 최적화 기법을 적용한 시나리오 6에서는 행렬 곱셈과 소프트맥스 연산 모두에서 병목 현상이 가장 많이 감소하였다.

5. 결론

본 논문에서는 WASM를 활용하여 디바이스와 엣지 클라우드 간의 Federated Learning을 수행하는 최적화 전략을 제안하였다. 제안된 전략은 일부 학습을 디바이스에서 수행하고, 나머지 학습을 엣지 클라우드에서 수행함으로써 효율성을 극대화하는 것을 목표로 하였다. 특히, GPU 메모리 세그먼트 간 데이터 이동과 연산 작업의 중첩을 최적화하여 전체 학습 시간을 줄이고 GPU 사용률을 향상시키는 방법을 수학적으로 설명하고 평가하였다. 성능 평가 결과, 비동기 데이터 전송과 연산 중첩을 통해 데이터 전송을 기다리는 GPU 유휴 시간을 줄일 수 있음을 확인하였다. 또한, 이러한 최적화 전략을 통해 모델 학습 속도가 향상되었으며, 전체 학습 시간이 크게 감소하였다. 또한, 엣지 클라우드에서 추가적인 학습을 수행함으로써 모델의 최종 정확도가 향상되었으며, 이는 다양한 디바이스에서 수집된 데이터를 효과적으로 활용할 수 있는 가능성을 보여준다. 특히, 데이터 불균형 문제를 해결하기 위해 로컬 데이터를 중앙 서버에 전송하여 데이터 균형을 맞추는 방법이 모델 성능 향상에 기여하였다. 향후 실시간 데이터 처리, 스마트 팩토리 및 스마트 농업 등에서 기초 데이터 처리와 엣지 클라우드에서의 정밀한 분석과 예측이 필요한 환경에서 효과적으로 적용될 수 있으며, Federated Learning 시스템의 보안성을 강화하기 위한 WASM의 샌드박스 환경을 활용하여 데이터 유출 및 모델 도용을 방지하는 방법을 개발함으로써, 더욱 안전한 Federated Learning 환경을 구축할 수 있을 것이다.

REFERENCES

[1] Lin, C., Wang, J., & Zhang, L., "Optimizing Game Engine Performance with WebAssembly", *Journal of Game Development*, 15(3), 123-135, 2021.

[2] Smith, A., Lee, J., & Kim, H., "Cross-Platform Game Development with WASM", *International Journal of Computer Games Technology*, 19(1), 45-59, 2022.

[3] Patel, R., Gupta, S., & Sharma, A., "Deploying Machine Learning Models with WASM", *IEEE Transactions on Neural Networks and Learning Systems*, 32(7), 3011-3022. 2021.

[4] Chen, M., Liu, Y., & Wang, Y., "Lightweight Machine Learning Inference with WebAssembly", *ACM Transactions on Web*, 14(2), 28-40, 2020.

[5] Nakamoto, S., Yamada, T., & Sato, K., "Optimizing Smart Contracts with WebAssembly", *Blockchain Research Journal*, 7(3), 67-80, 2021.

[6] Li, X., Wang, J., & Zhao, Q., Blockchain "WebAssembly Runtime: Enhancing Smart Contract Portability and Security", *IEEE Transactions on Blockchain Technology*, 11(1), 150-162, 2022.

[7] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V., "Federated Learning: Challenges, Methods, and Future Directions", *IEEE Signal Processing Magazine*, 37(3), 50-60, 2020.

[8] Park, S., Lim, J., & Kwon, Y., "Federated Learning with WASM for IoT Devices", *Sensors*, 21(3), 789, 2021.

[9] Zhang, X., Wu, Y., Zhao, R., & Zhang, J., "Secure Federated Learning with WebAssembly", *Future Generation Computer Systems*, 127, 284-295, 2022.

저자약력

최 종 석 (Jong-Seok Choi) **[정회원]**

- 2023년 02월: 숭실대학교 컴퓨터학과(공학박사)
- 2019년~현재: ㈜공감하다 대표
- 2020년~현재: 숭실대학교 스파르탄SW교육원 교수
- 2022년~현재: 개방형 클라우드 플랫폼 얼라이언스 표준화분과 분과위원장

〈관심분야〉 AI, Cloud-Native, PaaS, 컴퓨터비전, 데이터분석