CSE 598 - Data Intensive Systems for Machine Learning
Leaving Backpropagation Behind: Forward Gradients for Distributed Private Learning

Tanaya Lad
tlad2@asu.edu

Vatsal Gaurang Shah
vshah65@asu.edu

Parva Vipul Barbhaya
pbarbha1@asu.edu

Abstract

The aim of this project is to explore the forward gradient algorithm as an alternative to backpropagation for distributed machine learning. The forward gradient method uses forward-mode automatic differentiation and Jacobian vector products to estimate gradients, without requiring transfer of raw training data between nodes. A privacy-preserving framework is proposed where clients compute local Jacobian vectors and send only these to a central server for gradient aggregation and model updates. Experimental evaluation using convolutional and multilayer neural networks on the MNIST dataset is outlined to assess the algorithm's performance in terms of accuracy, training time, and communication overhead compared to backpropagation baselines in a simulated distributed setting. The investigation aims to determine the feasibility and potential benefits of adopting forward gradients for distributed, privacy-preserving deep learning.
Relevant keywords: Forward Gradient, Distributed Machine Learning, Jacobian Vector Product Aggregation.

Problem Investigation

The investigation focuses on the application and efficacy of the forward gradient algorithm within distributed machine learning environments. This topic is of great significance as it addresses critical challenges faced in the realm of distributed computing, where data is inherently decentralized across multiple computational nodes. Such environments are commonplace in real-world applications ranging from healthcare systems with data privacy constraints to large-scale industrial setups requiring data processing across geographically dispersed locations.

Challenges in Distributed Machine Learning
Distributed machine learning presents unique challenges primarily due to the separation of data and the need for frequent communication between nodes:

- Communication Overhead: Each node in a distributed system may need to send and receive large volumes of data during the training process, leading to significant communication overhead.
- Synchronization Costs: Ensuring that all nodes are synchronized in terms of model updates and state can be both time-consuming and resource-intensive, potentially leading to bottlenecks.
- Data Privacy Concerns: When training data contains sensitive information, transferring raw data between nodes can pose serious privacy risks, making it crucial to find methods that reduce or eliminate the need for raw data exchange.

Relevance and Interest
Exploring the forward gradient algorithm in this context is particularly interesting for several reasons:

- Efficiency in Gradient Computation: Unlike traditional backpropagation that relies on reverse-mode differentiation, the forward gradient approach utilizes forward-mode differentiation, which can be more efficient in terms of computational resources when dealing with functions that have high input dimensionality and lower output dimensionality.
- Potential for Reduced Communication: By computing gradients locally and only exchanging necessary gradient information, the forward gradient method can significantly reduce the amount of data that needs to be transmitted between nodes, thus lowering communication overhead.
- Enhanced Privacy: The forward gradient method can potentially enhance data privacy since it does not require sharing of raw data between nodes; instead, only gradient information or updates are exchanged.

Broader Implications
The investigation into the forward gradient algorithm is not just a technical endeavor but also addresses broader implications for the field of machine learning and distributed systems:

- Scalability: Successfully implementing and optimizing this algorithm could lead to more

scalable machine learning models that can be trained efficiently over larger distributed systems.

- Accessibility of Machine Learning: By reducing the requirements for high bandwidth and intensive communication, smaller organizations or those with limited resources could also leverage advanced machine learning models.
- Innovation in Algorithmic Approaches: Exploring alternative gradient computation methods like the forward gradient could pave the way for further innovations in algorithmic strategies, potentially leading to new paradigms in machine learning.

Literature Survey

The paper by Baydin et al. [1] presents a method for computing gradients in neural networks that does not rely on the traditional backpropagation algorithm, potentially reducing computational overhead and improving parallelism. Werbos et al. 's [2] study delves into optimizing the backpropagation through time algorithm for training recurrent neural networks, focusing on efficiency improvements and stability in learning sequences. Zhang et al. [3] propose a novel approach to train spiking neural networks using a modified back propagation technique that accommodates the temporal dynamics of spike sequences, enhancing learning accuracy and temporal data processing. "Deep Learning with PyTorch" [4] is a practical guide that teaches Python programmers how to build, train, and fine-tune neural networks for applications like cancer detection, using PyTorch's user-friendly yet powerful tools, with insights from PyTorch contributors and experts in the field. The research by Albawi et al. [5] reevaluates the design of lightweight convolutional neural networks aiming at maintaining high performance while significantly reducing computational cost and model size for mobile and embedded applications. The paper by Jia et al. [6] introduces a novel Discriminable Squeeze and Excitation Graph Convolutional Network (D-SEGCN) for semi-supervised node classification on graphs, which incorporates a Squeeze and Excitation module to enhance feature representation and address the issue of over-smoothing in graph convolutional networks, demonstrating improved performance on three citation network datasets. Li et al. [7] reevaluate the design of lightweight convolutional neural networks aiming at maintaining high performance while significantly reducing computational cost and model size for mobile and embedded applications. The paper by Shi et al. [8] introduces a communication-efficient method for distributed deep learning, focusing on reducing the bandwidth requirements without compromising the convergence rate or accuracy of the learning process. Zhang et al. [9] discuss a method for accelerating the training of deep learning models using distributed data parallelism that effectively handles the increasing model sizes and training data volumes. Lin et al. [10] explore a compression technique for gradients in distributed training environments that reduces the network communication load while preserving the essential information for accurate model updates.

Method Used

The proposed method involves using a privacy-preserving framework that leverages forward gradient descent in a distributed environment. The framework outlines steps from server initialization, client computation involving Jacobian vector products, to the aggregation of these vectors at a central server. This method eschews traditional backpropagation in favor of forward-mode automatic differentiation, which allows for efficient gradient computation without needing access to raw training data, thus preserving privacy.

1. Initialization: Setting the Stage for Distributed Computation - The first step in implementing the forward gradient algorithm in a distributed setting is the initialization phase. During this phase, the central server plays a crucial role in establishing the baseline for the entire learning process.
   - **Distribution of Initial Parameters**: The central server initializes the model by setting up the initial parameters, which include the weights and biases of the neural network. These parameters are crucial as they represent the starting point of the learning process.
   - **Random Seed Distribution**: Along with the initial parameters, the server also distributes a random seed to all participating client nodes. This seed is vital for ensuring that all nodes generate the same random perturbation vectors in a synchronized manner, which is essential for the consistency and comparability of the computations performed by each client.

2. Client Computation: Local Processing and Gradient Estimation - Once the initialization is complete, each client node begins its local computations, which are pivotal for the distributed training process.
   - **Generation of Perturbation Vectors**: Using the shared random seed, each client generates a perturbation vector. This vector is used to slightly alter the input data or the parameters, enabling the estimation of how changes in inputs affect the

outputs, which is a key aspect of gradient computation.

- **Jacobian Vector Product (JVP) Computation**: Each client uses its local dataset to compute the Jacobian vector product. The JVP represents the first derivative of the output with respect to the input, providing a directional gradient estimate. This computation is done using forward-mode automatic differentiation, which is typically more suitable for functions where the number of input parameters is less than the number of outputs.

3. Aggregation: Synthesizing Global Updates from Local Contributions - The final step in the forward gradient computation process involves the aggregation of the locally computed Jacobian vector products at the central server.
   - **Naive Method**: Here, the server sequentially updates the model parameters by applying each client's JVP one after the other. This method, while straightforward, might not be the most efficient in terms of convergence speed.
   - **Simple Average**: The server calculates the mean of all received JVPs and uses this average for updating the model parameters. This approach assumes equal contribution and influence from all clients, regardless of their data size or diversity.
   - **Weighted Average**: In this more sophisticated approach, weights are assigned to each client's JVP based on factors such as the size of their local dataset or the variance in their data. This method aims to optimize the learning process by giving more significance to potentially more informative or reliable gradients

Experimental Setup

The experimental design is structured to validate the effectiveness of the forward gradient algorithm in a realistic distributed machine learning scenario. By leveraging the MNIST dataset and a variety of neural network architectures, the experiments aim to closely mimic real-world distributed systems. This approach ensures that findings are applicable and scalable to actual distributed environments where data and computational resources are spread across multiple nodes.

Detailed Experimental Setup:

1. **Model Architectures** - Two primary types of neural network architectures will be employed:
   - Convolutional Neural Networks (CNNs) - These models are well-suited for handling image data and will demonstrate the algorithm's performance on more complex tasks.
   - Multilayer Neural Networks (MNNs) - Simpler than CNNs, these models will allow us to evaluate the algorithm's effectiveness with fundamental architectures.
2. **Dataset Distribution**: The MNIST dataset will be split into 50,000 training images and 10,000 test images. To simulate a distributed setting, the training data will be randomly distributed among the clients. This distribution mimics the challenges faced in real-world scenarios where data is not centrally located but rather dispersed across different geographical locations.
3. **Baseline Comparisons**:
   - Centralized Backpropagation: A model trained on the full dataset using traditional backpropagation will serve as a baseline to compare against the centralized training approach.
   - Decentralized Backpropagation: Additionally, a decentralized model trained using backpropagation with periodic averaging among the nodes will provide insights into how traditional distributed training stacks up against the proposed method.
4. **Training Procedure**:
   - Each client will train their model using the locally available data and compute the necessary gradients using the forward gradient algorithm.
   - After local training, gradients are sent to a central server where they are aggregated. This step tests the algorithm's efficiency in handling and merging information from multiple sources, a critical aspect of distributed systems.
5. **Communication Overhead and Efficiency**: An essential part of the experiments will involve measuring the communication overhead specifically, the total volume of data transferred between the clients and the central server per training epoch. This metric will help quantify the efficiency improvements provided by the forward gradient method in reducing the need for extensive data transmission, which is a significant bottleneck in distributed environments.
6. **Performance Metrics**: The models will be evaluated based on their accuracy on the MNIST test set, the total training time per epoch (measured in wall clock time), and the communication overhead involved. These metrics will provide a comprehensive view of the model performance, efficiency, and practicality.

Dataset Description

Dataset Selection

For this study, the MNIST dataset, which is a widely recognized benchmark in the machine learning community, will be utilized. This dataset consists of 60,000 images in total, divided into 50,000 training images and 10,000 test images. Each image is a 28x28 pixel grayscale representation of handwritten digits from 0 to 9. The MNIST dataset is chosen for its clarity in demonstrating the effectiveness of machine learning algorithms due to its moderate complexity and well-understood characteristics.

Distribution Among Clients

To simulate a distributed learning environment, the MNIST training dataset will be evenly partitioned and distributed among multiple client nodes. This distribution is intended to mimic a realistic scenario where data is not centrally located but is instead scattered across various locations, each with computational capabilities. This setup tests the algorithm's ability to handle data spread and ensure consistency and accuracy in learning from decentralized data sources.

Data Privacy Considerations

Given the distributed nature of the experiment and the emphasis on privacy preservation, the MNIST dataset offers an excellent opportunity to demonstrate the forward gradient algorithm's capability to train models effectively without needing access to centralized data. This aspect is crucial for scenarios where data privacy and locality are paramount, such as in medical or financial sectors where data may be sensitive and not transferable due to regulatory requirements.

Use of Data in Training and Testing

The training phase will involve each client node using its portion of the MNIST dataset to compute local updates to the model using the forward gradient method. This method involves calculating the Jacobian vector product locally, which is then sent to a central server for aggregation. The advantage here is that only the necessary computed updates are transmitted rather than the raw data, significantly enhancing data privacy.

During the testing phase, the model's performance will be evaluated using the separate set of 10,000 test images. This phase is crucial for assessing the generalizability and accuracy of the trained model across unseen data, which represents a realistic assessment of how the model would perform in actual deployments.

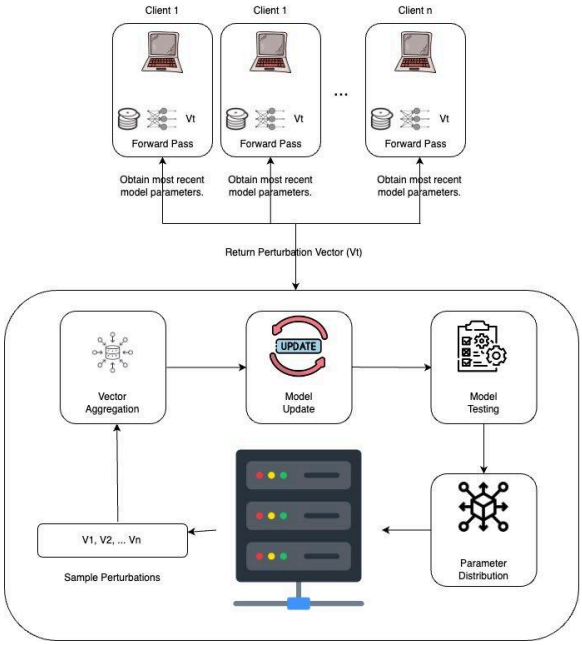

Fig 1. System Architecture Diagram

Results

Performance comparison of backpropagation and forward gradient

Centralized Back Propagation

	CNN	MNN
Time	61.15 min	55.24 min
Accuracy	0.8793	0.8658

Centralized Forward Gradient

	CNN	MNN
Time	58.45 min	57.77 min
Accuracy	0.825	0.846

De-centralized Forward Gradient

	CNN	MNN
Time	1hr 19 min	1 hr 8 min
Accuracy	0.834	0.881

Algorithm Performance Overview

1. **Centralized Back Propagation**: This method showed stable performance.
2. **Centralized Forward Gradient**: Similar stability in performance to centralized backpropagation.
3. **De-centralized Forward Gradient**: Noted a significant drop in accuracy and an increase in training time. This decline is attributed to the high communication overhead, particularly due to the serialization and deserialization of model parameters.

Effects of Decentralization

1. **Communication Overhead**: Major performance degradation in decentralized settings due to the overhead from handling data serialization (e.g., using pickle files).
2. **Scaling Benefits**: Increasing the number of clients could potentially enhance accuracy and reduce relative training time due to parallelization advantages.

Comparative Analysis

1. **Speed and Efficiency**: No significant speedup was observed between backpropagation and forward gradient methods.
2. **Model Type Impact**:
   - Convolutional Neural Networks (CNNs): Saw a considerable decline in accuracy, possibly due to the complexity and convergence challenges.
   - Multilayer Neural Networks (MNNs): Showed better performance with the forward gradient method, benefiting from simpler architectures.

Parameter Aggregation in Decentralized Setting:

CNN:

CNN	Naive Mathod	Weighted Average	Average
Time	61.15 min	1 hr 19 min	60.32 min
Accuracy	0.827	0.835	0.833

MNN:

MNN	Naive Mathod	Weighted Average	Average
Time	58.50 min	1 hr 15 min	56.88 min
Accuracy	0.851	0.881	0.876

The Weighted Average gained the highest accuracy but it took more time in comparison in parameter aggregation in a decentralized setting.

Analysis

The investigation into the forward gradient algorithm revealed several critical insights into its performance relative to traditional backpropagation in distributed machine learning environments:

1. Impact on Model Accuracy and Training Time:
   - **Centralized Settings**: Both the forward gradient and backpropagation demonstrated stable performance in a centralized setup. However, no significant speed advantages were observed for the forward gradient method.
   - **Decentralized Settings**: The decentralized implementation of the forward gradient showed a considerable drop in accuracy and an increase in training time. This degradation was primarily due to the high communication overhead involved in serializing and deserializing model parameters.
2. Differential Impact on Neural Network Architectures:
   - **Convolutional Neural Networks (CNNs)**: CNNs experienced a notable decrease in accuracy under the forward gradient approach, likely due to the

complex nature of these models which makes them more sensitive to the perturbations introduced by the forward gradient method.

- **Multilayer Neural Networks (MNNs)**: In contrast, MNNs, with their relatively simpler structures, responded better to the forward gradient method. This difference underscores the importance of model architecture in choosing the gradient computation technique.

3. Communication and Computational Overheads:
   - The experiments underscored the significant impact of communication overhead on the performance of distributed learning algorithms. The forward gradient's approach to local computation of gradients and centralized aggregation can potentially reduce data transmission volumes, but the serialization process itself becomes a bottleneck.
   - The need for synchronization and the associated delays in aggregating gradient information from multiple nodes also contributed to longer training times in decentralized environments.
4. Parameter Aggregation Techniques: The aggregation of parameters in a decentralized setting was another area of focus. The use of a weighted average for aggregating parameters emerged as a method that, while time-consuming, resulted in higher accuracy. This approach indicates a trade-off between computational time and model performance, particularly in environments where data is not uniformly distributed across nodes.

Conclusion

The study underscores the forward gradient method as a promising alternative to backpropagation, especially in distributed learning scenarios where data privacy and communication efficiency are crucial. This method enhances privacy by allowing the computation of gradients locally at client sites without needing to transfer sensitive raw data, making it particularly advantageous for use in industries like healthcare and finance. Additionally, it reduces communication overhead by minimizing the amount of data exchanged between nodes, which is beneficial in environments with limited bandwidth. However, the performance of the forward gradient method varies significantly with the complexity of the neural network model and the specific configurations of the distributed system. It tends to perform better with simpler architectures such as Multilayer Neural Networks due to their reduced sensitivity to the noise introduced by gradient estimations. Despite these benefits, challenges such as increased computational demands and potential accuracy losses in decentralized settings highlight the need for further refinement and optimization of the method, particularly in terms of effective parameter aggregation and adaptation to diverse network architectures.

Future Work

1. Algorithm Optimization: Further research will focus on optimizing the forward gradient calculation and aggregation methods to enhance performance and reduce computational and communication overhead even further.
2. Broader Application Scope: Expanding the testing framework to include more complex datasets and different types of neural network architectures to validate the algorithm's applicability and performance in a broader range of scenarios.
3. Integration with Other Privacy - Enhancingn Technologies: Combining the forward gradient approach with technologies like federated learning and differential privacy could potentially open up new avenues for highly secure and efficient distributed machine learning.

The exploration of forward gradients in distributed settings not only challenges traditional approaches but also provides a foundation for future innovations in machine learning that could transform how data is processed and utilized across diverse and large-scale environments. This research could lead to significant advancements in making machine learning more adaptable, privacy-focused, and efficient in real-world applications.

References

[1] Baydin, A.G., Pearlmutter, B.A., Syme, D., Wood, F. and Torr, P., 2022. Gradients without backpropagation. arXiv preprint arXiv:2202.08587.

[2] Werbos, P.J., 1990. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), pp.1550-1560.

[3] W. Zhang and P. Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," in Advances in Neural Information Processing Systems, vol. 33, 2020. [4] Eli Stevens, Luca Pietro Giovanni Antiga, and Thomas Viehmann, Deep Learning with PyTorch, Manning, 2020.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6.

[6] N. Jia, X. Tian, Y. Zhang, and F. Wang, "Semi-Supervised Node Classification With Discriminable Squeeze Excitation Graph Convolutional Networks," in IEEE Access, vol. 8, pp. 148226-148236, 2020.
[7] K. Li et al., "Rethinking Lightweight Convolutional Neural Networks for Efficient and High-Quality Pavement Crack Detection," in IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 1, pp. 237-250, Jan. 2024.
[8] S. Shi et al., "Communication-Efficient Distributed Deep Learning with Merged Gradient Sparsification on GPUs," in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Toronto, ON, Canada, 2020, pp. 406-415.
[9] X. Zhang, C. Zhang, and M. Jiang, "Distributed Data Parallel Training Based on Cumulative Gradient," in 2022 2nd International Conference on Computer, Control and Robotics (ICCCR), Shanghai, China, 2022, pp. 202-206.
[10] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in Proc. of International Conference on Learning Representations, 2018.