Ensuring Verifiable and Secure Outsourced Training of Deep Neural Networks

Xu Kang, Zhuoyi Hu, Xianchao Wang, Kejie Zhao
Presented By Bahen at July 4, 2023

1 Introduction

Deep Neural Networks (DNNs) have revolutionized a wide range of fields from healthcare to finance to entertainment. However, the process of training these models requires a substantial amount of computational power, typically provided by Graphics Processing Units (GPUs). To meet this demand, many entities resort to outsourcing their training tasks to third-party workers, equipped with the necessary resources.

Our project, Lab Apex, serves as a platform bridging the gap between clients with training tasks and workers with available GPU resources. Clients upload their training scripts and data onto the platform, and workers undertake the task of running the scripts and training the models. While this system optimizes resource allocation and broadens accessibility to machine learning capabilities, it also introduces critical concerns around verifiability, security, and privacy.

The issue we address in this paper is the development of a robust, efficient, and secure mechanism for "Proof of Learning." This protocol will validate that a worker has honestly and correctly executed the job it claims to have done. Formally, once a worker commences the training process using its GPU resources, a method must be in place to verify the completion and authenticity of the training task.

The integrity of this process relies on two primary conditions: first, that the worker has indeed performed the task rather than presenting a distillation of a trained model, and second, that the model has been trained with the original model and hyperparameters setup. A secure and reliable verification process is fundamental to uphold the integrity of outsourced DNN training. This paper proposes the integration of proof learning techniques to address these challenges. Our goal is to establish a method that not only verifies the correct and honest execution of the outsourced training process but also ensures that no sensitive information about the model or data is leaked during the process.

2 Method

Our proposed method for validating the integrity of the outsourced training tasks consists of three parts, each designed to tackle a unique aspect of the problem and the integration would help generate a more stable and safe solution.

2.1 Real-Time Monitoring for Training Integrity

The first part of our method involves continuous validation through real-time logging of the training operations, underpinned by a secure execution environment.

As the worker runs the training script, critical metrics such as training loss and GPU utilization metrics are monitored and logged. These logs are regularly sent to a dedicated server for evaluation and storage. This approach serves two main purposes. Firstly, it monitors the activity of the GPU throughout the training process, ensuring that the worker's GPU is being utilized. Secondly, it verifies that the number of logged training losses aligns with the total number of iterations run, as any discrepancies could indicate issues or anomalies in the training process.

To ensure the integrity of this logging system, and to prevent the worker from potentially modifying the training script or forging logs, we enforce a secure execution environment. This environment is achieved by encapsulating the scripts into encrypted executable files before they are sent to the worker. The encrypted executables are designed to be resistant to reverse engineering attempts, making it difficult for the worker to view or modify the original training scripts. This approach ensures that the logs generated and sent to the server truly represent the performance and progression of the training process, eliminating the risk of log forgery.

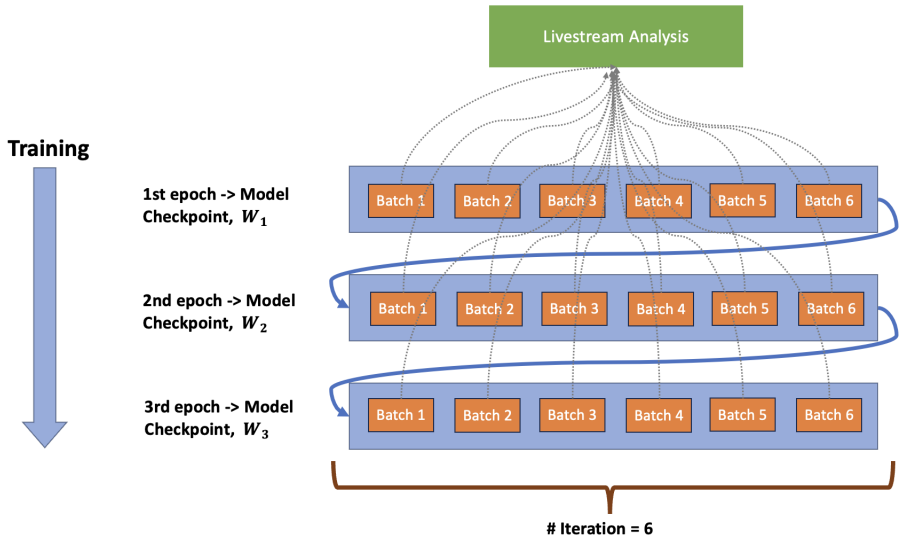

Figure 1: An Overview of Deep Neural Network Training with Live-stream Monitoring

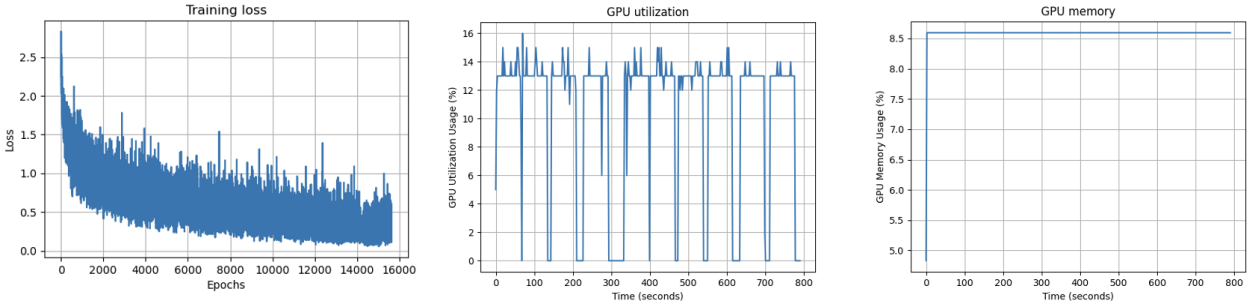

Figure 2: An Example of Training Metrics Generated by ResNet-18 Training with CIFAR-100 on NVIDIA TESLA V100. The left figure is training loss vs iterations; The middle figure is the GPU Utilization vs. time as training starts; The right figure is the GPU Memory vs. time as training starts

As shown in Figure 1, it demonstrates an example of training a neural network where the tasks is running for 3 epochs. For each epoch, there are 6 iterations and each iteration would generate one piece of log of training loss. Thus, a total of 6*3 training loss would be sent to the server for training integrity analysis. In addition, three model checkpoints would be generated after each epoch and these checkpoints would be used for later validation process. Figure 2 is an example of metrics generated by the training ResNet-18 [1] with CIFAR-100 on Nvidia Tesla V100. Given the authenticity of logs, some integrity checks could be as following.

- Let N be the number of training loss logs, E the total number of epochs, and I the number of iterations per epoch. The condition that the logs correctly reflect the training process is $N = E \times I$.

- Let $U = \{u_1, u_2, \ldots, u_n\}$ represent the series of GPU utilizations and let $\bar{U}$ represent the average GPU utilization. Let α be the proportion of GPU utilization measurements that should be larger than $\bar{U}$, which we heuristically set to 0.4. This condition can be expressed as:

$$\frac{|\{u \in U : u > \bar{U}\}|}{|U|} \geq \alpha$$

  Here, $|\{u \in U : u > \bar{U}\}|$ represents the number of elements in U that are larger than $\bar{U}$, and $|U|$ represents the total number of elements in U.

- Let $M = \{m_1, m_2, \ldots, m_n\}$ represent the series of GPU memory usage over time. We

expect the sequence to have a steep increase in the beginning, followed by a relatively stable period. This can be expressed as:

$$m_{i+1} - m_i > 0 \text{ for small } i,$$

and

$$|m_{i+1} - m_i| \approx 0 \text{ for large } i.$$

Here, $m_{i+1} - m_i > 0$ for small i captures the expectation of a steep increase in GPU memory usage at the beginning of the sequence. The second condition $|m_{i+1} - m_i| \approx 0$ for large i expresses the expectation that the changes in GPU memory usage become small (i.e., the usage is stable) later in the sequence.

In essence, the combined mechanism of logging and secure execution environment provides a layer of transparency, accountability, and security during the training process. This enables real-time checks and ensures the correctness and integrity of the task's execution. It forms the foundational layer of our validation method, upon which the subsequent parts build further.

2.2 Probabilistic Epoch Validation (PEV)

Despite the use of encrypted executables, there is always a non-zero risk of system infiltration and data tampering. To further bolster our system's security, we introduce an additional layer of validation - *Probabilistic Epoch Validation* (PEV).

In PEV, for every epoch, not only is the training loss evaluated, but the model checkpoints are also validated. Specifically, for each model checkpoint, we randomly select $n\%$ of the data and recompute the training loss for these selected samples. This re-computed loss is then compared against the training loss computed for the entire epoch. This procedure provides an additional guard against potential spoofing attempts by requiring not just a global, but also a granular match between logged and recomputed losses.

In mathematical terms, let $\bar{L_e}$ represent the average of training loss for epoch e, and let $\bar{L'_e}$ represent the average of recomputed loss based on the $N\%$ randomly selected data. PEV verifies the following condition:

$$|\bar{L_e} - \bar{L'_e}| < \epsilon$$

where ϵ is a small threshold, ensuring that the discrepancy between the two losses is within acceptable limits. Heuristically, ϵ is chosen as:

$$\epsilon = \frac{\sigma_e \times 100}{N}$$

where σ_e is the standard deviation of the training loss for epoch e.

This method provides significant security benefits. Even if an adversary were to hack the real-time monitoring system, they would require substantial computational resources to forge the generated model – in many cases, potentially more resources than running the task honestly [2].

Additionally, the PEV method is computationally efficient for validation purposes. Since we only select $N\%$ of the training data for re-computing the training loss.

Given an inference cost C_{inf}, which is approximately $\frac{1}{5}$ to $\frac{1}{10}$ of the original training cost C_{train} [3][4], i.e.,

$$\frac{1}{5}C_{\text{train}} \leq C_{\text{inf}} \leq \frac{1}{10}C_{\text{train}},$$

Therefore,

$$\frac{N}{500} \times C_{\text{train}} \leq C_{\text{ver}} \leq \frac{N}{1000} \times C_{\text{train}}.$$

For a common choice of $N = 10$, i.e. we select 10% of the training data to calculate the loss, the verification cost is only **1%** to **2%** of the original training cost. This computation suggests that the verification process is significantly more efficient than the original training process.

2.3 Verifiable Model Ownership

The final part of our proposed method, Verifiable Model Ownership (VMO), draws inspiration from the paper titled "Proof-of-Learning: Definitions and Practice" [2]. This paper emphasizes the importance of model ownership, as modern machine learning training procedures can be expensive and complex, making them significant targets for wrongful appropriation. It addresses the challenge of proving the ownership of a trained model, as well as verifying the integrity of training procedures, particularly in decentralized (distributed) training scenarios where many workers perform training tasks in parallel.

The core concept in the paper is that model ownership implies an entity has expended resources

(computational resources, in this case) for training the model. Thus, observing the "secret information" i.e., the intermediate states during training, which cannot be guessed in advance due to the inherent randomness of training, serves as evidence for the effort expended in model training. This secret information, specifically the sequence of intermediate states (or checkpoints) of model parameters, can only be known to the entity that honestly trained the model. By performing step-wise verification of these states and comparing the results against claimed checkpoints, the paper proposes a method to verify the integrity of training procedures. The computational and storage costs of this verification process are mitigated by storing weights at every k^{th} step and verifying only the top one or two steps in each epoch, where the largest change in weights occurs, which is as below:

$$\|W_{t+k} - W'_{t+k}\|_2 < \epsilon$$

where $\epsilon = \frac{1}{m} \sum_{i=t}^{t+m} \|\nabla \mathcal{L}(W_i)\|_2$ for some suitable choice of m.

This method employs a similar approach for an added layer of security and verification, providing a robust solution for establishing the proof of honest and resource-intensive model training. Future improvements in this direction could focus on further reducing the computational cost for verification and storage costs for the prover.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016.

[2] Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy*, 2021.

[3] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.