This is the preprint version. Please see IEEE for the final official version.

Ensuring Trustworthy Neural Network Training via Blockchain

1st Edgar Navarro*
Department of Computer Science
San Diego State University
3dgarnavarro@gmail.com

2nd Kyle J. Standing*
Department of Computer Science
Brigham Young University
kjstanding2@gmail.com

3th Gaby G. Dagher
Department of Computer Science
Boise State University
gabydagher@boisestate.edu

4rd Tim Andersen
Department of Computer Science
Boise State University
tandersen@boisestate.edu

***Abstract*—As Artificial Intelligence prevalence grows, it highlights the risk in relying on compromised models, thereby fueling a growing need to ensure the integrity of trained AI models. In this paper, we present a novel blockchain-based system, designed to authenticate the integrity of trained neural network models. The system addresses the risk of manipulation of a model by strategically re-computing intervals of the training process. Further, the blockchain network provides a traceable, immutable, trusted ledger for cataloging the intricate processes of training and validation. We consider two primary entities involved: 'submitters', who submit trained models, and 'verifiers', who re-train distinct sections of the submitted models to validate their integrity. The design of the blockchain system emphasizes efficiency by selectively targeting a portion of all training intervals. This is made possible through the use of an innovative weight-analysis algorithm, which applies an Absolute Change approach to identify outliers. We implement our solution to demonstrate that the proposed blockchain system is robust, and the weight-analysis algorithm is accurate and scalable.**

***Index Terms*—Blockchain; Machine Learning; Neural Networks;**

I. INTRODUCTION

Artificial Intelligence (AI) has become an integral part of our daily lives, influencing various sectors such as business, science, and education. Among the many branches of AI, deep learning, which employs large neural networks to perform complex tasks, has been particularly transformative. However, as AI continues to evolve and integrate into our society, it raises critical concerns about the integrity and reliability of AI models.

The motivation for our work stems from the increasing reliance on AI systems and the potential risks associated with compromised models. The process of designing and training AI models is complex and computationally intensive. It often requires expensive, specialized hardware and a significant amount of time. As the complexity and resource requirements of models increase, many researchers and developers are opting to delegate the training process to third parties. While this approach alleviates the computational burden, it introduces a new challenge: trust.

*These authors contributed equally.

There are two main factors that can affect the integrity of a third-party trained model. The first of which is whether the requested training was actually completed, and the second that the model was not manipulated or "poisoned". A poisoned model is one that has been manipulated to behave in unintended ways, typically through the introduction of manipulated data during training. While these are of particular concern when dealing with a third-party trained model, the threat is present in many other related fields.

In this paper we focus on the challenge of verifying the integrity of neural network models, which are a subset of AI. Neural nets pose a unique hurdle such that without a clear understanding of the training process, it is difficult to derive any information about the training that the model underwent. Further, it is infeasible to determine whether a model has been poisoned based solely off the model itself. This is known as the "black-box" nature of neural network models, making it nearly impossible to understand the inner workings of a model or how it arrives at a particular outcome.

Addressing these challenges is crucial for the continued growth and acceptance of AI technologies. Ensuring the integrity of trained neural network models is not just about maintaining the accuracy of predictions, but also about fostering trust in neural network systems. This is particularly important as neural nets and other deep learning technology continues to be integrated into critical areas such as healthcare, finance, and autonomous vehicles, where the consequences of a compromised model could be severe.

Current strategies for validating the integrity of neural network models are often constrained by their detection scope and a requirement of understanding of how a model might be compromised. These methods are typically designed to identify a specific type of poisoning or compromise, thereby limiting their effectiveness when faced with a diverse range of potential threats to a model's integrity. Moreover, these approaches often necessitate the user to possess some level of knowledge about the model's training process. This requirement presents a significant challenge, as it may not always be feasible or possible to obtain detailed information about

a model's training, particularly when dealing with third-party models or models trained on proprietary or confidential data.

In response to these challenges and concerns, we propose a novel method of model validation using blockchain technology. Our solution efficiently validates the integrity of a model without any prior knowledge of potential poisoning methods. It not only detects whether a model has been poisoned but also ensures that the model was trained as specified. Furthermore, our solution provides an immutable record of a model's evolution, enabling comprehensive analysis of any model validated on the network. This approach paves the way for a system of trusted AI models, fostering greater confidence in AI technology.

A. Contributions

This paper presents several significant contributions to the field of neural network model verification and blockchain technology:

- **Blockchain Network for Model Verification:** We propose a blockchain network specifically designed for the task of efficient verification of the integrity of neural network models. This network utilizes the distributed nature of blockchain technology, allowing many independent nodes to participate in the verification process. This collaborative approach also leverages the inherent transparency and immutability of blockchain technology to provide a robust and reliable platform for model verification.
- **Weight-Analysis Algorithm:** We introduce a novel weight-analysis algorithm that intelligently distributes recomputation tasks across the network. This algorithm is designed to optimize the verification process, balancing computational efficiency with effective model validation.
- **Comprehensive Implementation and Testing:** We developed a comprehensive implementation of our proposed blockchain network and weight-analysis algorithm. Furthermore, we conduct extensive experiments to rigorously test the system's capabilities. Our experimental results demonstrate that the proposed blockchain system is robust in the presence of malicious actors, and that our weight-analysis algorithm performs with high accuracy in detecting outlying training intervals and is scalable w.r.t. model complexity.

II. Related Work

A. Blockchain for AI

Blockchain technologies have emerged as an innovative solution to enable decentralized trust and ensure accountability and transparency in neural network model sharing. Several works of research have been done in this area, which demonstrate various applications and implications of using blockchain in the context of machine learning and artificial intelligence.

Research done by Bore et al. introduced a blockchain-enabled system to establish decentralized trust in machine learning and computational simulations. This system leverages blockchain to store, share, and maintain auditable logs and records of each step involved in a modeling process. The system also monitors worker outputs, and can rank and identify faulty workers to ensure the quality of models [4]. Similarly, Sarpatwar et al. described the use of blockchain for tracking the provenance of training models in artificial intelligence [1]. This approach could enable end-users to trust the received AI models by providing them with information about how the model was trained and the data it was trained on. Such a mechanism can potentially lead to better trusted AI.

Raman et al. introduced a framework for distributed trust in computations, which employs a novel combination of distributed validation of atomic computation blocks and a blockchain-based immutable audit mechanism [3]. This approach addresses the scalability problem by reducing the storage and communication costs using a lossy compression scheme. It also ensures the verifiability of the final results and the validity of local computations. In a slightly different approach, Baldominos et al. proposed Coin.AI [2], a proof-of-useful-work scheme for a blockchain-based distributed deep learning system. In this system, mining requires the training of deep learning models, and a block is only mined when the model's performance exceeds a certain threshold. This mechanism not only verifies the accuracy of the delivered models in an efficient way, but also promotes useful computation as a core feature of the blockchain system.

These systems underscore the potential of blockchain technology in enhancing the transparency, traceability, and reliability of shared machine learning models. However, they do not directly address the issue of model poisoning. While Raman et al. indirectly address the issue, a comprehensive solution that specifically targets model poisoning is absent. This gap can be attributed to the inherent complexity associated with identifying poisoned models. Our work fills this gap by proposing a blockchain network specifically designed for efficient verification of neural network models integrity.

B. Consensus algorithms

The consensus mechanism is a critical component of blockchain technology, serving as the protocol by which agreement is reached among the decentralized nodes in the network. It ensures the security, integrity, and robustness of the system, making it a compelling area of research. This section reviews the works of various researchers who have proposed innovative consensus algorithms that leverage machine learning and computational models. These studies are particularly relevant to our work as we seek to enhance the consensus mechanism in our blockchain network for efficient verification of neural network models.

A. Shoker introduces a novel consensus algorithm, the Proof of eXercise (PoX) [5], as a more sustainable and less energy-intensive alternative to the traditional Proof of Work (PoW). The author proposes that a matrix-based scientific computation problem could replace the "useless" computations in PoW, providing a more rational and efficient approach to consensus. This new paradigm, however, faces challenges as the "work"

Work	Poison Detection	Model Provenance	Training Validation	Storage	Blockchain Type
Sarpatwar et al. [1]	No	Yes	No	No	Public
Baldominos et al. [2]	No	No	No	Yes	Public
Raman et al. [3]	Yes	Yes	No	No	Private
Bore et al. [4]	No	Yes	No	Yes	Private
Our Work	Yes	Yes	Yes	No	Private

Table I: Comparison of our work with related works in the areas of: Poison Detection, Model Provenance, Training Validation (checking training for poisoning or lack of completion), Storage, and Blockchain Type.

remains far from being really "useful". The concept of Proof of Learning (PoL) was first proposed by Bravo-Marquez et al. as part of the WekaCoin project [6]. In this model, consensus is achieved by ranking machine learning systems for a given task. Unlike the hashing-based puzzles in PoW, this approach strives to create a public distributed and verifiable database of state-of-the-art machine learning models and experiments, thereby adding value to the computations involved. Liu et al. further develop the concept of PoL in their work, "Proof of Learning (PoLe): Empowering neural network training with consensus building on blockchains" [7]. They propose a new consensus mechanism that directs the computation spent for block consensus towards the optimization of neural networks. Their design involves the entire blockchain network's access to training and testing data, with the training of neural network models serving as proof of learning. This design includes a secure mapping layer (SML) to prevent consensus nodes from cheating. The authors claim that the PoLe protocol results in a more stable block generation rate, leading to more efficient transaction processing, without significantly sacrificing training performance.

Lihu et al. developed a Proof of Useful Work (PoUW) protocol [8], which is a novel consensus mechanism based on training machine learning models. Their approach reduces the resource-intensive process typical of Bitcoin mining and instead focuses on performing useful machine learning training work, thereby rewarding miners for their contributions to the network. This work presents an interesting model for blockchain-based systems that aspire to not just solve puzzles but to contribute towards meaningful work. Shafay et al. provided an insightful review of this subject [9]. The authors classify and categorize related literature by devising a thematic taxonomy based on parameters such as blockchain type, deep learning models, deep learning specific consensus protocols, application area, services, data types, and deployment goals. They argue that integrating blockchain technology with deep learning could offer operational transparency, traceability, reliability, security, and trusted data provenance, addressing the shortcomings of centralized deep learning systems.

In summary, the adoption of alternative consensus mechanisms in blockchain technologies has the potential to greatly enhance their efficiency and applicability. The consensus algorithms based on learning and useful work, such as PoX, PoL, PoLe, and PoUW, represent promising directions for future work.

C. Poisoned Model Detection

Poisoned model detection is a crucial task in ensuring the correct functioning of a trained neural network. The identification of whether a model has been trained maliciously is not a straightforward task due to the "black box" nature of many machine learning models. Below are some significant works that have made advancements in this area.

Gao et al. proposed an approach named STRIP (STRong Intentional Perturbation) to combat trojan attacks on deep neural networks [10]. Trojan attacks leverage the difficulty in interpreting a learned model to misclassify any inputs signed with a secret trojan trigger. STRIP works by intentionally perturbing incoming inputs, such as by superimposing various image patterns, and observing the randomness of predicted classes for these perturbed inputs. If the predictions demonstrate low entropy, violating the input-dependence property of a benign model, it suggests the presence of a malicious input, a characteristic of a trojaned input. To improve trojan backdoor detection in artificial intelligence systems, Guo et al. introduced an approach called TABOR [11]. Backdoor trojans are hidden patterns within a deep neural network (DNN) that force the model to behave abnormally when triggered. Existing detection techniques require an assumption of availability of the contaminated training database, which might not be practical. TABOR formulates trojan detection as an optimization problem, guided by explainable AI techniques and heuristics. It also uses an anomaly detection method to better identify intentionally injected triggers in the infected model and filter out false alarms.

Amarnath et al. presented a method called TESDA (Transform Enabled Statistical Detection of Attacks) for online detection of attacks on deep neural networks [12]. TESDA exploits the discrepancies caused by attacks in the distributions of intermediate layer features of DNNs. The method does not require dedicated hardware to run in real-time, nor the presence of a Trojan trigger to detect discrepancies in behavior. It has been shown to achieve detection coverages of above 95% with low overheads.

These methods highlight the importance and ongoing development in detecting and defending against poisoned or trojan-infected neural networks. The success of these methods also paves the way for more sophisticated techniques in ensuring the integrity of machine learning models. However, most of these algorithms require heavy computations or knowledge of how a neural network model may have poisoned. When working in a distributed blockchain system these are key points

of interest to address. Our poisoned model detection algorithm is efficient enough to work seamlessly on a blockchain network and makes no assumptions on the training process.

III. Background

The integrity of a machine learning model is crucial for its reliable operation. However, it can be compromised in several ways, one of which is model poisoning [13, 14]. Model poisoning involves the intentional introduction of misleading data into the training set, causing the model to make incorrect predictions. These attacks can lead to biased predictions or even complete failure of the model, making them a significant threat to the reliability of machine learning systems [15].

Deep learning models, particularly those based on neural networks, are often referred to as "black boxes" because their internal workings are not fully understood [16, 17]. This lack of transparency makes it difficult to verify the integrity of these models and to understand how they arrive at their decisions. We use the knowledge of how neural networks work and their individual components to obtain a measure of the models integrity [18].

We introduce an important component of neural networks, the individual neurons. Neurons in a neural network are designed to mimic the function of biological neurons. These neurons receive a number of inputs which are individually weighted, with the weight vector within the neuron. They are then summed and passed through an activation function that leads to the following neurons [19, 20].

Weights and biases are a set of parameters within the model that transform input data within the network's hidden layers. As an input enters the network, it is multiplied by a weight value and added to a bias before being passed onto a neuron in the next layer. The neuron sums all the inputs it receives, and if the sum surpasses a certain threshold, the neuron is activated and sends data to the next layer. The initial values of these weights and biases are typically set randomly and are adjusted during the training process [20]. The goal of training a neural network is to adjust these weights and biases in such a way that the error between the model's predictions and the actual output is minimized. This is done through a process called backpropagation, where the model learns from its errors by propagating the error backwards through the network, adjusting the weights and biases as it goes [21, 22, 23].

The depth of these networks is what enables them to learn from data and perform complex tasks, but it also contributes to their complexity and the computational resources required to train them [24].

Weight analysis can be used to identify abnormalities or signs of model poisoning [20]. If a model has been poisoned, the weights associated with the poisoned data may show unusual patterns compared to the rest of the weights [23].

Blockchain technology is a decentralized and distributed digital ledger that records transactions across multiple computers in such a way that the registered transactions cannot be altered retroactively [25]. This technology is designed to bring security, transparency, and efficiency to the exchange of digital assets. It works by grouping transactions into blocks, which are then linked together in a chain. Each block is secured using cryptographic techniques, and a consensus mechanism ensures that all transactions are agreed upon by the network.

One such consensus mechanism is quorum consensus [26]. The key differentiator in quorum consensus from other blockchain solutions is that it allows for transactions to be processed quickly and efficiently, making it suitable for use in a production environment. Quorum consensus achieves consensus through two mechanisms: QuorumChain, which is a voting-based mechanism, and Raft-based consensus, which is a majority-based mechanism. This flexibility allows quorum consensus to be used in a variety of different applications, including those that require high speed and high throughput.

In the context of model validation, blockchain technology can provide a secure and transparent way to record and verify the training process of a machine learning model [9]. Each transaction in this case could represent an update or change to the model, such as an adjustment of weights during training. This creates an immutable record of the model's training process. For example, if a model is trained over a distributed network, each update to the model's parameters could be recorded as a transaction on the blockchain. This would allow anyone to verify the entire training process, step by step. It can provide valuable insights into how the model was trained and can help identify any abnormalities or signs of model poisoning. Moreover, because the blockchain network is decentralized, it can be verified by anyone in the network. This means that third parties can independently verify the integrity of the model without needing to trust the entity that trained the model. This feature is particularly useful when the training process is outsourced to a third party[27, 28].

IV. SOLUTION: Diffusion

A. Solution Overview

This research encapsulates the steps towards creating a blockchain system that is expressly designed to authenticate the integrity of comprehensively trained neural network models. The purpose of this system is to function as a traceable, non-alterable, and a trusted ledger for cataloging the intricate processes of training and validation of these advanced neural network models. The primary concern addressed is the potential risk of the introduction of 'poisoned' data or adversarial manipulation during the training phase of a model. The proposed blockchain system serves to verify the safety and integrity of these models, thereby negating this concern and enhancing trust in model reliability.

The system involves two primary entities: submitters, who provide trained models with a detailed training provenance, and verifiers, who re-train distinct sections of the submitted models to validate their integrity. The blockchain system's design emphasizes balance between computational power and responsibility, selectively targeting specific training iterations for retraining instead of the whole model, ensuring computational efficiency.

Upon model submission, the submitter is required to provide snapshots of the model as it progressed through intervals of training. These snapshots represent the model in its entirety and capture the changes the model underwent during each iteration. The intervals are to be created so that each iteration of training between snapshots are relatively equal to each other.

Once all relevant information is made available to the network, an algorithm is run to delegate particular intervals to be recomputed by the verifiers. This process involves retraining the interval, a comparison of resulting models, and a report of any differences. Any difference between a submitted interval and the retrained interval indicates a breach of integrity.

To detect anomalies in the model, an algorithm based on four approaches is used: Absolute Change, Euclidean distance (L2 norm), Percent Change, and Cosines distance. The Absolute Change and L2 norm methods perform best in detecting potential poisoning attacks. However, the best algorithm can vary depending on the application.

B. Blockchain System

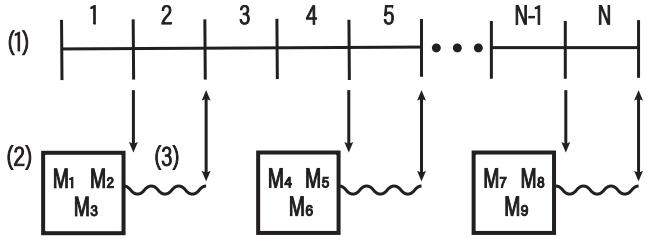

Figure 1: System Diagram illustrating the interaction between submitters and verifiers in the proposed blockchain system. The diagram shows the process from model submission (1), through intelligent selection of training intervals for retraining (3), to the multi-layered verification process (2).

The proposed blockchain system in this research consists of two primary entities: submitters and verifiers. Submitters are entities, either individuals or organizations, that contribute trained models to the system, accompanied by their detailed training provenance. This provenance acts as a testament to the safety and integrity of the model, ascertaining that no detrimental or adversarial data manipulation transpired during the training process. Contrastingly, verifiers operate as a consensus quorum. Their function involves the re-training of distinct sections of the submitted neural network models to authenticate their integrity and cross-verify the results against the data supplied by the submitters.

The architecture of the blockchain system is specifically designed to ensure a balance between responsibility and computational power. To enhance computational efficiency and curtail the extensive resource requirements inherent in comprehensive model retraining, an intelligent selection procedure is integrated. This procedure targets specific training intervals for retraining, thus circumventing the need for a complete model retraining procedure. The consensus algorithm that forms an important part of this system is meticulously designed to strike a balance between computational efficiency and robustness.

Method	Precision	Recall	F1-Score
Absolute Change	0.86	1.0	0.92
L2 Norm Difference	0.80	0.67	0.73
Percent Change	0.75	1.0	0.86
Cosine Distance	0.11	0.17	0.13

Table II: Performance metrics for different methods of analyzing changes in neural network weights.

Upon submission of a model and its associated training provenance for verification, the consensus quorum commences its operation. It selects discrete training intervals from the model for retraining. For the efficient execution of this task, the quorum bifurcates itself into smaller sub-quorums. Each subquorum is then tasked with the independent retraining of an assigned interval, with the subsequent results compared against the original model's data supplied by the Submitter.

This verification mechanism assures robustness. It significantly mitigates the risk of false positives or negatives, thereby guaranteeing the high integrity of models recorded in our blockchain system. We posit that this comprehensive verification process embodies the principles of trust, democratization, and computational efficiency, representing a significant advance in the application of neural networks across diverse fields, while ensuring reliability and trust in their outputs.

C. Weight-analysis Algorithm

The fundamental basis of our detection algorithm lies in the principle that a poisoned interval (an iteration of model training) causes a significant alteration to the learned parameters of a neural network. This abrupt shift is distinguishable from the comparatively gradual changes observed during normal training, and our method exploits this distinction for detecting poisoned intervals. We took 4 different approaches to interpreting the parameters: Absolute Change, Euclidean distance (l2 norm), Percent Change, and Cosine Distance.

1) Absolute Change: This method measures how much each individual weight has changed, without considering the direction of the change (increase or decrease). The resulting value is a cumulative representation of all changes in weights from one model to another. This can be a useful metric for identifying intervals where significant changes have occurred, such as a potential poisoning attack, because large changes in model weights could indicate a shift in the model's learning behavior.

The absolute change for weights $w_i^{(1)}$ and $w_i^{(2)}$ of two consecutive models can be computed as:

$$\Delta w = \sum_{i=1}^{n} \left| w_i^{(2)} - w_i^{(1)} \right| \tag{1}$$

Where n is the total number of weights in the model.

2) L2 Norm Difference: The L2 norm, also known as the Euclidean norm, is a mathematical concept that finds widespread usage in machine learning and data science. At

its core, the L2 norm measures the distance between two points in a space, and it does so by taking into account all dimensions of that space. It is mathematically defined as the square root of the sum of the squared differences between the individual elements of the vectors. In the context of neural networks, the L2 norm is often used as a measure of difference or distance between the parameters (weights and biases) of two models. When you apply the L2 norm to the difference between the parameters of two models, you get a single scalar value that represents the overall difference between the two models, accounting for all parameters simultaneously. This makes the L2 norm a useful tool for tracking changes in models during training or for comparing different models.

The L2 norm difference between the weights $w_i^{(1)}$ and $w_i^{(2)}$ of two consecutive models can be computed as:

$$\Delta w = \sqrt{\sum_{i=1}^{n} (w_i^{(2)} - w_i^{(1)})^2} \tag{2}$$

Where n is the total number of weights in the model.

3) Percent Change: This method calculates the percentage change in weights between two consecutive intervals. This method is similar to the Absolute Change method but normalizes the change by the magnitude of the weights. It can be more sensitive to relative changes in model parameters. However, if the weights of the model are small, even a small absolute change can result in a large percent change, leading to potential false positive.

The percent change for weights $w_i^{(1)}$ and $w_i^{(2)}$ of two consecutive models can be computed as:

$$\Delta w = \frac{\sum_{i=1}^{n} \left| w_i^{(2)} - w_i^{(1)} \right|}{\sum_{i=1}^{n} \left| w_i^{(1)} \right|} \times 100 \tag{3}$$

Where n is the total number of weights in the model.

4) Cosine Distance: This method involves the computation of the cosine of the angle between two weight vectors. We began by flattening the weights of two consecutive neural network models, thus converting multi-dimensional tensors into one-dimensional vectors. Subsequently, we calculated the cosine distance between these flattened weight vector. This cosine distance, a scalar value ranging from -1 to 1, signifies the cosine of the angle between the two weight vectors. A value closer to 1 indicates a smaller angle and thus a higher similarity between the two vectors. This method effectively measures the orientation changes between consecutive models during the training process. Any significant deviation from the expected cosine distance may be indicative of a poisoning attack or other anomalies in the neural network's training process.

The cosine distance between the weights $w_i^{(1)}$ and $w_i^{(2)}$ of two consecutive models can be computed as:

$$d = 1 - \frac{\sum_{i=1}^{n} w_i^{(1)} \cdot w_i^{(2)}}{\sqrt{\sum_{i=1}^{n} (w_i^{(1)})^2} \sqrt{\sum_{i=1}^{n} (w_i^{(2)})^2}} \tag{4}$$

Here, n is the total number of weights in the model.

D. Theoretical Analysis

Theoretical analysis involves a mathematical or logical examination of our proposed solution to gain insight into its effectiveness and potential limitations. For our blockchain-based system and anomaly detection algorithm, this involves a thorough analysis of computational complexity, accuracy, and resource requirements.

Firstly, the computational complexity of our proposed algorithm plays a crucial role in determining its efficiency. Both the Absolute Change and L2 Norm Difference approaches have a computational complexity of $\mathcal{O}(n)$, where n is the total number of weights in the model. These methods involve a single pass over all weights, which ensures that the computational complexity remains linear and manageable even for larger models. The Percent Change and Cosine Distance methods also have a computational complexity of $\mathcal{O}(n)$, despite additional mathematical operations, as these do not significantly affect the overall computational complexity.

As we have noted, the Absolute Change method performed the best among all the methods we evaluated. This method involves measuring the absolute differences in weights between two consecutive models. The key advantage of this method is its simplicity, which makes it computationally efficient. Here is a formal presentation of its efficiency using big O notation:

Theorem 1. *The Absolute Change method has a linear time complexity, specifically, $\mathcal{O}(n)$, where n represents the total number of weights in the neural network model.*

Proof. The Absolute Change method requires iterating over each weight in the neural network model. For each weight, the method computes the absolute difference between its values in two consecutive models. Therefore, if there are n weights in the model, the algorithm performs n computations.

Assuming that computing an absolute difference can be done in constant time, the total time complexity of the algorithm is proportional to the number of weights, hence $\mathcal{O}(n)$. The space complexity is also $\mathcal{O}(n)$ as we need to store the values of weights for two consecutive models. □

The Absolute Change method has the advantage of simplicity and computational efficiency. However, it does not take into account the direction or the relative magnitude of changes in weights, which may be significant in some applications. This may limit its effectiveness in detecting subtle or sophisticated poisoning attacks that cause minor but crucial changes in weights. In such cases, methods like Percent Change or Cosine Distance, which consider the direction and relative magnitude of changes, might be more effective, albeit with increased computational complexity.

Therefore, while the Absolute Change method's performance is superior in our experiments, it is essential to consider the specific requirements and constraints of individual applications when selecting the most suitable detection method.

Accuracy is another crucial factor in evaluating our algorithm. According to our experiments, the Absolute Change

method showed the highest F1-Score, which considers both precision and recall, making it the most accurate method among the four. However, the best performing method can vary depending on the application, the size and complexity of the neural network, and the nature of the poisoning attack.

The resource requirements of our algorithm, specifically memory usage, should also be considered. Since our algorithm only requires information about weights from two consecutive models during computation, it does not require significant additional memory overhead. The computational requirements are also manageable as the processes can be run on the GPUs available in most modern computer systems.

Furthermore, the system design ensures that the computational load of model verification is distributed across the network of verifiers. This design principle significantly reduces the burden on individual entities and enhances the overall efficiency of the system.

V. Experiments

We provide experimentation and demonstration of our solution based on three key factors: robustness against bad actors, accuracy and scalability of the weight-analysis algorithm. A complete implementation of our solution was built and utilized in this testing. Our results provide invaluable insights into our solution, specifically, how the weight-analysis algorithm accurately detects outlier training intervals and is linearly scalable, and that the blockchain network is resilient against the actions of participant bad actors.

A. Implementation and Setup

Our implementation is a blockchain network, tasked with accepting, validating integrity of and storing information of submitted neural network models. We built upon BlueChain [29], a blockchain research framework, to create our unique implementation. The network consists of a variable number of independent nodes that communicate with one another in a peer to peer environment. The nodes accept transactions from a client that can connect to any number of nodes within the network. The transactions accepted, in this case, are bundles of model data. Model data submitted to the network include access to snapshots of the model training process, dataset used, and any other relevant info to be provided to nodes for recomputation. Nodes arrange themselves into quorum and non-quorum members in a unified but randomized fashion using the most recent block in the chain. We utilize this quorum selection process to determine nodes to delegate retraining tasks to. The nodes within the quorum then agree on whether to validate or invalidate the model according to the verification process specified in our solution and then append a new block to the chain with all relevant info about the model.

The experiments and implementation were conducted using a system composed with Python and Java, leveraging libraries such as TensorFlow, Keras, NumPy, and pandas. The platform used for this study consisted of an Intel Core i7-13700k CPU paired with an Nvidia GeForce RTX 4090 GPU.

B. Weight-analysis Accuracy Experiment

A set of test scenarios was developed, including variable numbers of poisoned intervals, groups of consecutive poisoned intervals, and clean intervals. For each scenario, the model was trained, and then its snapshots from each interval were analyzed. The model architecture employed was a simple feed-forward neural network with a variable number of hidden layers, each containing a variable number of neurons, and a soft-max output layer for multi-class classification. The number of training intervals varied according to different test scenarios. The poisoning mechanism involved injecting a backdoor pattern into the training data for the poisoned intervals. The injected pattern was paired with the label of the target class.

The detection of poisoned intervals was based on the analysis of the model weights across intervals. The absolute change, as shown in equation 1, in weights between consecutive intervals was calculated, with significant changes potentially indicative of poisoning or lack of training execution. A dynamic threshold was used to classify intervals as problematic or clean based on these changes.

The evaluation of our detection mechanism involved calculating precision, recall, and F1 score. For each test scenario, we also tracked the number of successful detections (true positives) and failures (false negatives or false positives).

In addition to these standard evaluation metrics, we also used two accuracy metrics to evaluate the model's performance on clean and poisoned test data. This allowed us to gain insight into how well the model was performing its intended task and how effectively it had learned the backdoor pattern.

Results were accumulated across multiple runs of each test scenario, providing an aggregate view of the detection mechanism's performance. The final results, metrics, and visualizations provide valuable insights into the efficacy of our poisoning detection mechanism. This information is crucial in assessing the potential of our approach as a defensive tool against backdoor attacks on machine learning models.

C. Weight-analysis Scalability Experiment

Our study further probed the efficiency of the weight-analysis algorithm by conducting an experiment to understand its scalability relative to the complexity of the neural network models. Scalability describes how a system adapts or performs when it is subject to an increased load, which, in our case, corresponds to the complexity of the model as defined by the number of weights.

In this experiment, a range of neural network models were created, each with a varying number of layers. These models ranged from having a minimum of 5 hidden layers to a maximum of 200 hidden layers, with an increment of 5 layers for each successive model.

For each group of similarly structured models, comparisons were performed on the weights of the models within the group, with the objective of computing a single difference score. This score represented the aggregate of the absolute differences between the weights of the models being compared. The

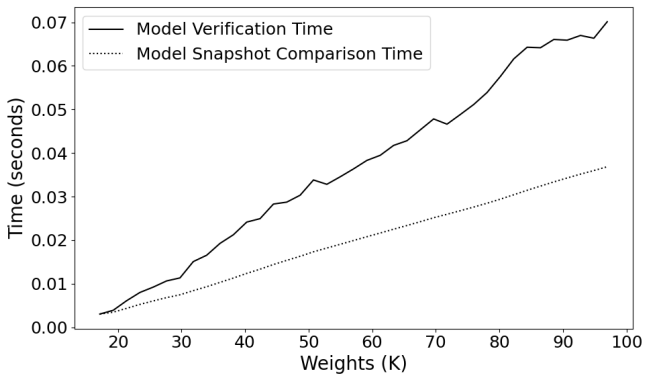

Figure 2: Scalability of the Poisoning Detection Mechanism: The graph displays the relationship between the complexity of artificial neural network models (measured by the number of weights) and the average computation time spent per model snapshot and total time per model. The linear growth pattern indicates a proportional increase in computation time with model complexity.

time taken to compute the difference scores was recorded and subsequently averaged over the number of comparisons performed for each group.

Our experiment's outcomes, as illustrated in figure 2, provide invaluable insights into the computational demands of our detection mechanism, a determinant factor for its applicability in practical, resource-constrained environments. The graph shows a linear growth pattern, indicating that the average computation time increases linearly with the model complexity. This observation is consistent with the theoretical time complexity of the Absolute Change method we analyzed earlier, reinforcing its computational efficiency.

Furthermore, the results also highlight the optimal model size for our detection mechanism and offer substantial guidance for prospective enhancements and improvements. The discrepancies seen in the comparison time line can be described as being a result of the different computation loads demanded by the varying complexity of the models. Some models may introduce more of a certain type of change in the parameters where the algorithm spends more time accounting for them.

These inconsistencies also hint at the existence of a potential threshold of model complexity beyond which the poisoning detection mechanism becomes computationally inefficient. For models approaching $70,000$ parameters, the computation times start to display a pronounced increase. This threshold indicates an optimal balance point where the complexity of the model and the efficiency of the detection mechanism can be maintained simultaneously.

This scalability experiment gives us crucial insights into the design constraints that would need to be considered when implementing this poisoning detection mechanism in a real-world setting. Balancing the complexity of the models and the computational efficiency of the detection mechanism will be critical for maintaining the viability and effectiveness of our system.

D. Robustness Experiment

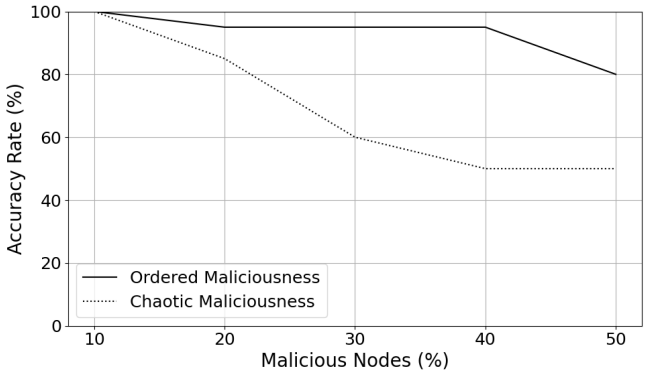

Figure 3: Validation accuracy at varying percentages of malicious nodes: The graph displays the relationship between the percentage of malicious nodes and the accuracy rate of the detection mechanism. Two scenarios are compared: 'Ordered Maliciousness' and 'Chaotic Maliciousness'. The accuracy rate, represented as a percentage, is plotted against the percentage of malicious nodes in the network. The differing patterns between the two scenarios highlight the effectiveness of the detection mechanism under different conditions.

To evaluate the robustness of our blockchain network, we conducted an experiment measuring how presence of bad actors affected performance. The percentage of malicious nodes present in the network was varied and compared against the percentage of how often the system correctly validated models. The data collected allowed an assessment of the network's resilience to adversarial attacks, that is its ability to maintain accurate functionality despite the presence of malicious nodes.

In this experiment, we created a series of scenarios wherein the proportion of malicious nodes in the network ranged from 0% to 50%, with increments of 10%. For each scenario, we ran a sample set of 20 neural network models through our system and observed the network's performance. The testing set was comprised of 50% clean models that had been trained properly, 25% that had been trained on poisoned data in random sections of training, and 25% that had been trained with poisoned data at a random sequence of sections.

Malicious nodes were designed to behave in a way that undermines the integrity verification process of the network. Specifically, this behavior is triggered when a node is assigned within a sub-quorum to recompute an interval. The malicious node may report to its sub-quorum peers that it found no issue with the interval when in fact the interval was invalid. Realistically, bad actor's behavior will vary according to their specific purposes. To model this varying behavior we defined two modes of malicious behavior. The first is "ordered maliciousness", where the bad node will attempt to validate any invalid interval that it is assigned to. The second is"chaotic

maliciousness", where the node will attempt to invalidate any valid interval in addition to reporting invalid intervals as valid. Both behaviors are a significant threat to the network, as they have the potential to significantly undermine the network's ability to correctly verify the integrity of models.

The performance of the network was evaluated based on its ability to correctly verify the integrity of the models, despite the presence of malicious nodes. Two separate networks were set up, the first of which whose malicious nodes were defined as ordered malicious and the second being chaotic malicious. The dataset was presented to both networks as the percent of malicious nodes was increased. We measured the accuracy of each network as a percentage of how often the network correctly verified the integrity of a model.

Figure 3 shows that the network maintained relatively high accuracy rates as percentage of malicious nodes increased. The experiment demonstrates that for both malicious scenarios, the network can validate models with 85% accuracy with the network containing up to 20% percent of malicious nodes. The network is more susceptible to chaotic maliciousness with accuracy dropping significantly at percentages of maliciousness greater than 20%. However, with ordered maliciousness the network maintains 80% accuracy even up to a 50% makeup of malicious nodes in the network.

These results demonstrate the robustness of our blockchain network against adversarial attacks. They show that the network can maintain its functionality and continue to correctly verify the integrity of models even in the presence of a significant proportion of malicious nodes. This robustness is a key strength of our network and an important factor in its potential for practical implementation.

VI. Conclusion and Future Work

In conclusion, this research presents a novel approach to ensuring the integrity of trained neural network models using a blockchain-based system. Our proposed solution addresses the critical issue of potential data poisoning or adversarial manipulation during the model training phase. The system, involving two primary entities: 'submitters' and 'verifiers', provides a robust and efficient mechanism for model verification, through intelligent assignment of training iterations to be recomputed for validation. Our weight-analysis algorithm, tested on four distinct approaches, has demonstrated promising results in detecting outlying training intervals. Our implementation is a novel contribution for testing and evaluation of our proposed solution. Our experiments utilizing the implementation have demonstrated robustness and the accuracy and scalability of the weight-analysis algorithm.

Despite these promising results, we recognize that there is room for improvement and expansion of our work. Future research could explore the following areas:

- **Enhanced Detection Methods:** While our current detection methods have shown promising results, there is potential for further refinement or the development of new methods. These could offer improved detection rates or efficiency, particularly in more complex or subtle poisoning scenarios.
- **Real-world Applications:** Our research has primarily focused on theoretical analysis and controlled experiments. Future work could involve applying our solution to real-world scenarios, which would provide valuable insights into its practical performance and potential limitations.
- **Scalability:** While our solution has demonstrated good scalability in our experiments, further research could explore its performance with larger and more complex models. This could involve developing methods to further optimize the computational efficiency of our solution.
- **Consensus Mechanism:** Our system currently uses a quorum consensus mechanism for model verification. Future work could explore ways to enhance this mechanism, such as by developing more efficient selection procedures or by incorporating additional checks to further reduce the risk of false positives or negatives. This could improve the robustness and reliability of our system.

In summary, our research represents a significant step towards ensuring the integrity of trained neural network models. We believe that our work provides a strong foundation for future research in this important area.

References

[1] K. Sarpatwar, R. Vaculin *et al.*, "Towards Enabling Trusted Artificial Intelligence via Blockchain," in *Policy-Based Autonomic Data Governance*, ser. Lecture Notes in Computer Science, S. Calo, E. Bertino, and D. Verma, Eds. Cham: Springer International Publishing, 2019, pp. 137–153.

[2] A. Baldominos and Y. Saez, "Coin.AI: A Proof-of-Useful-Work Scheme for Blockchain-Based Distributed Deep Learning," *Entropy*, vol. 21, no. 8, p. 723, Aug. 2019, number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[3] R. K. Raman, R. Vaculin *et al.*, "A Scalable Blockchain Approach for Trusted Computation and Verifiable Simulation in Multi-Party Collaborations," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 277–284.

[4] N. K. Bore, R. K. Raman *et al.*, "Promoting Distributed Trust in Machine Learning and Computational Simulation," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 311–319.

[5] A. Shoker, "Sustainable blockchain through proof of exercise," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, Oct. 2017, pp. 1–9.

[6] F. Bravo-Marquez, S. Reeves, and M. Ugarte, "Proof-of-Learning: A Blockchain Consensus Mechanism Based on Machine Learning Competitions," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, Apr. 2019, pp. 119–124.

[7] Y. Liu, Y. Lan *et al.*, "Proof of Learning (PoLe): Empowering neural network training with consensus building on blockchains," *Computer Networks*, vol. 201, p. 108594, Dec. 2021.

[8] A. Lihu, J. Du *et al.*, "A Proof of Useful Work for Artificial Intelligence on the Blockchain," Jan. 2020, arXiv:2001.09244 [cs].

[9] M. Shafay, R. W. Ahmad *et al.*, "Blockchain for deep learning: review and open challenges," *Cluster Computing*, vol. 26, no. 1, pp. 197–221, Feb. 2023.

[10] Y. Gao, C. Xu *et al.*, "STRIP: a defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, ser. ACSAC '19. New York, NY, USA: Association for Computing Machinery, Dec. 2019, pp. 113–125.

[11] W. Guo, L. Wang *et al.*, "TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems," Aug. 2019, arXiv:1908.01763 [cs].

[12] C. Amarnath, A. H. Balwani *et al.*, "TESDA: Transform Enabled Statistical Detection of Attacks in Deep Neural Networks," Oct. 2021, arXiv:2110.08447 [cs].

[13] Y. Li, Y. Jiang *et al.*, *Backdoor Learning: A Survey*.

[14] Z. Tian, L. Cui *et al.*, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," vol. 55, no. 8, pp. 1–35.

[15] Y. Li, Y. Bai *et al.*, "Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection."

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks."

[17] A. Barredo Arrieta, N. Díaz-Rodríguez *et al.*, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," vol. 58, pp. 82–115.

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

[19] P. Hajela and L. Berke, "Neural networks in structural analysis and design: An overview," vol. 3, no. 1, pp. 525–538.

[20] L. Wang, C. Wang *et al.*, "Explaining the behavior of neuron activations in deep neural networks," vol. 111, p. 102346.

[21] H. Leung and S. Haykin, "The complex backpropagation algorithm," vol. 39, no. 9, pp. 2101–2104.

[22] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," vol. 40, no. 4, pp. 967–969.

[23] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weight errors," vol. 1, no. 1, pp. 71–80.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," vol. 521, no. 7553, pp. 436–444.

[25] D. Yaga, P. Mell *et al.*, "Blockchain technology overview," p. NIST IR 8202.

[26] D. J. Sumpter and S. C. Pratt, "Quorum responses and consensus decision making," vol. 364, no. 1518, pp. 743–753.

[27] J. Liu, J. Huang *et al.*, "From distributed machine learning to federated learning: a survey," vol. 64, no. 4, pp. 885–917.

[28] H. B. McMahan, E. Moore *et al.*, "Communication-efficient learning of deep networks from decentralized data."

[29] P. Lundquist, "Bluechain," 2023, gitHub repository.