Towards Human Society-inspired Decentralized DNN Inference

Dimitrios Papaioannou, Vasileios Mygdalis, Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

{*mygdalisv, pitas*} *csd.auth.gr*

Abstract

In human societies, individuals make their own decisions and they may select if and who may influence it, by e.g., consulting with people of their acquaintance or experts of a field. At a societal level, the overall knowledge is preserved and enhanced by individual person empowerment, where complicated consensus protocols have been developed over time in the form of societal mechanisms to assess, weight, combine and isolate individual people opinions. In distributed machine learning environments however, individual AI agents are merely part of a system where decisions are made in a centralized and aggregated fashion or require a fixed network topology, a practice prone to security risks and collaboration is nearly absent. For instance, Byzantine Failures may tamper both the training and inference stage of individual AI agents, leading to significantly reduced overall system performance. Inspired by societal practices, we propose a decentralized inference strategy where each individual agent is empowered to make their own decisions, by exchanging and aggregating information with other agents in their network. To this end, a "Quality of Inference" consensus protocol (QoI) is proposed, forming a single commonly accepted inference rule applied by every individual agent. The overall system knowledge and decisions on specific manners can thereby be stored by all individual agents in a decentralized fashion, employing e.g., blockchain technology. Our experiments in classification tasks indicate that the proposed approach forms a secure decentralized inference framework, that prevents adversaries at tampering the overall process and achieves comparable performance with centralized decision aggregation methods.

Keywords: Decentralized DNN inference, Distributed consensus, Distributed systems, Decentralized decision making, Blockchain

1. Introduction

There is a growing need of executing Deep Neural Networks (DNNs) on edge devices, since such architectures typically outperform other type of classifiers in a wide range of applications, including computer vision, natural language processing, and speech recognition [1]. Due to their computational/memory complexity, DNNs are typically trained and maintained on powerful servers, while the data used for inference are obtained by edge devices such as tablets, smartphones, autonomous vehicles, and sensors [2]. Adapting DNN Inference across multiple allocated devices frequently necessitates offloading the input sensor data to a cloud or a centralized fusion server for executing all DNN Inference there, or dynamically distributing DNN computations between the edge devices and the master server. The master server typically employs centralized ensemble learning methods to aggregate the received information collected by the individual edge devices [3]. The aforementioned strategies frequently raise concerns about associated costs in communication and latency, as well as privacy concerns [4, 5].

The installation of DNNs on edge devices, motivated by such issues, enables inference on the same user device rather than sending samples to a centralized cloud server. These architectures are required primarily because edge devices such as smartphones, smart cameras and so on, frequently have memory and processing power limitations, thus existing approaches work around that concept to reduce computation costs and decrease inference time for real-time applications [6]. To address such issues, leading strategies propose compressing highly parameterized DNNs using pruning [7] and quantization [8] methods or combining statistical model-based domain knowledge into DNN-aided systems that use compact networks [9, 10]. Such solutions are frequently focused on a single edge device, without considering the fact that numerous users can comfortably and securely collaborate in order to benefit from their shared computational resources.

Moreover, approaches that enable device collaboration involve the division and partitioning of the centralized master server between multiple nodes (e.g., devices) that can communicate directly with each other by incorporating peer-to-peer networks, ad-hoc networks, gossiping protocols, and so on. Computation offloading is the partitioning of a multi-layered DNN among multiple nodes to jointly form a large network during inference [7], whereas collaborative intelligence is the division of a DNN between an edge device and its edge server [11, 12]. In both scenarios, each participating device only keeps a subset of the DNN layers and transmits its output features to the specific device that keeps the succeeding layers, which may be compressed to save overhead. The fundamental disadvantage

of such designs is that each device cannot infer on its own, and all of the devices that comprise the DNN must be present. As a result, there is a high reliance on connectivity, which may cause latency issues.

However, research in that field is still in narrow steps because existing approaches are limited to local inference via previously mentioned architectures, making device collaboration practically impossible and relying heavily on assumptions of constant reliable communications. In other words, these architectures rely on mutual trust between all nodes, resulting in high-risk failures such as crash failures, computational errors, highly biased data, or even malicious attacks such as poisoned models or tamper and steal user's private data [13]. In distributed DNN settings, a common approach for dealing with such issues is to replicate the centralized master server to avoid it becoming a single point of failure. However, for the generated replicas to synchronize, a classical state machine replication (SMR) approach must be considered. A total order of updates must be established in all replicas via consensus, to eliminate the need of it and thus, benefit from the resilient nature of the multiplicity of underlying replicas [14]. Such approaches, as seen in DNN training settings, particularly in decentralized SGD optimization algorithms, can result in significant latency and computational/memory overhead because, in order for replicas to agree on a common state, they must also agree on the overall order of model updates. This can lead to multiple re-transmissions of gradients and parameter vectors, burdening the network with large transactions of many megabytes.

Blockchain, on the other hand, can be viewed as a decentralized storage system that can be used for highly secure decentralized training under consensus assumptions, making it a well-established alternative approach. The underlying Proof of Work consensus protocol limits the proposal rate of blocks in permissionless blockchain by allowing unauthenticated users to propose blocks, if and only if, they prove that they have performed the necessary computations [15]. Due to the ability to prevent numerous attacks, including Sybil and Distributed Denial of Service (DDoS) attacks, this approach offers fault-resilient properties, at the cost of increased computational / energy consumption. Research focuses on the idea of Proof of Useful Work. Rather than solving the alleged computational puzzle, a machine learning task is tackled. The squandered energy during the consensus process is utilized for something that actually matters, while the blockchain structure gives a highly protected and privacy - preserving setting for the machine learning operations [16, 17]. Despite the fact that such methods have a major impact on the total blockchain and AI combo, big scale DNN models cannot be applied since they require a sizable quantity of storage space from the blockchain nodes.

In our work, a society-inspired DNN inference architecture is designed to simulate how people make decisions based on their own opinion while taking into account the experts opinions in a relevant field. In particular, a fully decentralized framework over a multi-agent based system is proposed, by leveraging peer-to-peer communications to form an ensemble of individual experts. A novel SMR-based distributed consensus protocol is proposed in addition, ensuring the agreement of each AI agent on a widely accepted "prediction" history. The Quality of Inference (QoI) consensus protocol aims at establishing consensus between the individual agents without the need of any centralized structure. A SMR strategy is employed, inspired by the consensus protocols operating in blockchain, where each agent stores and maintains a copy of all previously agreed-upon predictions. Our experiments demonstrate that by individualizing an ensemble approach can increase the accuracy of each agent and thereby enhance generalization performance, whereas the QoI can offer a commonly accepted agreement between the individualized agents, with outcomes comparable to typical centralized ensemble aggregation methods.

Our main contributions through that work are:

- A decentralized decision-making process influenced by human society, where each agent is empowered to decide whether or not the accumulate the knowledge of other agents, simulating how individuals make decisions based on their own beliefs while also taking expert advice into account.
- A novel consensus protocol for distributed DNN Inference, in which a commonly accepted agreement is established between AI agents in the same prediction history working as a single inference rule.
- A fault-tolerant inference architecture in which misbehaving AI agents are penalized, reducing their influence on the decision making process of honest agents.

The rest of the paper is structured as follows. The problem of decentralized DNN inference is formulated in Section 2, which also provides a brief review of centralized and decentralized ensemble approaches, as well as a review of distributed consensus protocols. In Section 3 we describe the individualized ensemble decision making process as well as the proposed QoI consensus protocol. Section 4 contains experimental results from both the individualized ensemble approach and the QoI consensus protocol. Finally, Section 5 provides conclusion remarks and future research directions.

2. Centralized, distributed and decentralized inference

Let $\mathcal{G} = \{\mathcal{A}, \mathcal{E}\}$ be a direct acyclic graph consisting of M collaborating AI agents described in a set $\mathcal{A} = \{\alpha_1, \alpha_2, ..., \alpha_M\}$, that are employed to perform some inference task, e.g., classification, and $\mathcal{E}$ defined as a set of fixed communication links allowing them to communicate with each other. It is assumed that all agents have obtained access to the same test sample $\mathbf{x}$, while their goal for them is to produce a single prediction $\hat{y}$. This work differentiates the following strategies:

Centralized inference refers to the case where each AI agent α_i produces an intermediate prediction y_i for a given test sample $\mathbf{x}_i$. A master node thereby collects and aggregates the individual AI agent predictions and produces the final system output, using e.g., an average/median rule or majority voting.

Distributed inference refers to the case where individual nodes

only perform computational tasks, i.e., the inference task is divided between each of the nodes and/or a master node, and the final output of the system for a given test sample is provided by the master node.

Decentralized inference refers to the case where individual nodes α_i make their own predictions for a given sample as in the centralized inference case, however, the aggregation is performed by all participating AI nodes, using a consensus protocol.

A conceptual diagram differentiating the different problem variants in shown in Figure 1.

Figure 1: A conceptual diagram of centralized, distributed and decentralized inference.

2.1. Centralized Inference

Centralized inference is the simplest form of decision fusion and is tightly coupled with ensemble learning, involving numerous techniques that vary across the input data structure or the heterogeneity of the base models. The combination of a set of weak learners can be used to produce a stronger model, which will improve the generalization performance and reduce the biases accross base models. These methods can be divided into three core families: bagging, boosting, and stack generalization [18]. Regardless of employed technique, decision fusion is responsible for the aggregation rules that need to be applied for the combining step [19]. The most simple and common approach for agent decision fusing is the averaging rule. The final prediction is produced by averaging the outputs of the base models [20] while weighted variants obtained by optimizing objective functions [21] or the median rule. Another approach is majority voting. Instead of calculating the average of the probability outcomes, the central node counts the votes of the base models and predicts the final class as the class with the majority of votes.

In addition, ensemble selection methods [22, 23] (also referred to as ensemble pruning methods) are often required when the AI agents form a collection of various heterogeneous models. These methods search for a subset of ensemble models that performs similarly to the original ensemble model in order to decrease the number of base models [24, 25]. Several strategies for choosing an ideal subset of models are based to the fact that it is possible to identify the class label that has the majority of support without taking into account all votes [26, 27]. Distributed structures make use of ensemble selection and centralized decision fusion methods for the final aggregation stage on a fusion or cloud server.

Despite the simplicity of centralized structures, decentralized architectures are required in some applications to address issues such as security, storage limitations, computational costs, privacy, or handling large volumes of data. To overcome the aforementioned issues, data or models are distributed across multiple physical nodes (e.g., data centers, sensors, devices). As a result, algorithms capable of dealing with non-collocated data must be developed. Those type of algorithms can be divided into two main categories: Distributed and Decentralized Algorithms.

2.2. Distributed inference

Modern devices are boosted with high-level hardware components allowing them to partially execute DNN models directly on them. Often a highly reliable communication scheme is assumed for the individual devices to securely communicate between each other their locally produced information. The most computationally challenging operations and the general coordination is ensured via a powerful cloud-based server. That innovation provides solution to centralized issues since the DNN computations can be allocated to multiple devices allowing then to collaborate, secure the DNN process and preserve privacy of the input data. A novel approach for Distributed DNN Inference [4] is proposed based on distributed computing hierarchies, involving cloud server, edge network and devices. Same as in [28], the core idea is that the computational graph of pretrained DNN models is partitioned and distributed over edge nodes along an edge to cloud path to achieve forward propagation in the network while the data traverses toward the cloud.

Furthermore, methods for improving the existing models' sensor fusion capability, fault tolerance, and data confidentiality are proposed in [29, 30, 5]. A byzantine resilient fault-tolerant inference algorithm is proposed in [29], under an environment of multiple sensor cameras. Specifically, each sensor has a local feature extractor and each produced feature is weighted based on the importance in terms of likelihood. A weighted average scheme is used in back-end layers in order to make the global prediction of the captured model. Other fault tolerant approaches [30, 5, 3] adopt the concept of skip hyper-connections where they skip one or more physical nodes in the vertical hierarchy of distributed DNN providing alternate paths for the information propagation when a node is detected as faulty. In the field of unsupervised learning, a distributed approximate method for performing variational inference over a network of independent agents is proposed [31]in order to address the problems observed in symmetry and dependencies. Specifically, Bayesian models are employed to involve approximate inference methods, so that agents can efficiently use the Bayesian rule to combine the local posterior probabilities. In DELCO [32] a new approach for aggregating the predictions of the base models using a probabilistic model relying on Gaussian copulas under the coordination of a global centralized server is investigated.

2.3. Decentralized Inference

As we can observe, fault-resilient behavior can be established in the previous architectures if and only if it is assumed

that the cloud-based server is not tampered and the communication links between edge devices are reliable [33]. Fully decentralized architectures, on the other hand, correspond to setups in which a centralized server does not exist. In such architecture, individual nodes are collectively tasked to generate final decisions based on information exchange among them in the so-called "on-device" algorithms. In parallel computing, fully decentralized training approaches such as [34, 35] often involve concepts where in each iteration, each worker exchanges their locally computed gradients with its neighbors and combining them by averaging methods. Despite the fact that a centralized server is no longer required, synchronization through a common clock is vital. However, in Asynchronous Decentralized Parallel SGD [36], there is no need for each worker to wait for the rest ones to finish their execution process but instead, a ring-based network topology is used where after each iteration, a worker observes and selects a neighbor for averaging. Both workers replace their local gradients with the averaged one. In [37] in order to reduce communication between nodes, predefined epochs in which workers execute SGD in parallel are used and all individual results are averaged. The generated model is then used as reference for the next epoch. LEARNAE [38] proposes a fault-tolerant peer-to-peer architecture in which training data is propagated across the network using resilient gossiping protocols.

Other methods make use of Deep Ensembles, a class of architectures that employ several models whose outputs are aggregated to improve performance by using DNNs as individual models [39]. Specifically, the input sample is processed in parallel by each of these DNNs, and their outputs are aggregated into a single decision. Deep ensemble models are typically made up of a trainable encoder that encodes the input data into a continuous space and a dedicated decoder that compresses the features into a discrete representation. A scalable element-wise quantization can be used to extract the representation [40]. Later that year, another method called Edge Ensembles was introduced, claiming that it can achieve collaborative inference in a fully decentralized manner by leveraging Deep ensembles with quantized features [41]. Those models are comprised of a shared encoder and a user specific decoder. Let θ_E represent the shared parameters of encoder, the shared quantizer as Q and the unique and diverse parameters of decoder as $\theta_{\{D,j\}}$ for the j^{th} user. At any given time t a user j_t observes a data sample $\mathbf{x}_{i_t}$ to be used as inference. In order to collaborate with the neighbor devices, it encodes and quantizes the input sample $\mathbf{x}_{i_t}$ as:

$$z_{j_t} = f_Q\left(f_{\theta_E}\left(\mathbf{x}_{i_t}\right)\right). \tag{1}$$

The j_t broadcasts the quantized features z_{j_t} in all the neighbors in the set $\mathcal{S}^t_{j_t}$, where set $\mathcal{S}^t_{j_t}$ represent the set of all users in which user j can reliably communicate at a given time t. Then, each available user in $\mathcal{S}^t_{j_t}$ applies its local decoding model to z_{j_t} and conveys the resulting mapping $f_{\theta_{D,j}}\left(\mathbf{z}_{j_t}\right)$ back to user j_t which then aggregates them into a prediction $\hat{y_{i_t}}$. Specifically, for classification tasks, with N different labels, the output of each individual network $f_{\theta_{D,j}}\left(\mathbf{z}_{i_t}\right)$ is a $N \times 1$ vector whose values are estimated by the conditional distribution of each label given z_{i_t} [41]. The predicted value $\hat{y_{i_t}}$ is given as the label which maximizes the average condition distribution of the form:

$$\underset{n=1,\ldots N}{\operatorname{argmax}} \frac{1}{\left|\mathcal{S}^t_i\right|} \sum_{j \epsilon \mathcal{S}^t_i} \left[f_{\theta_{D,j}}\left(\mathbf{z}_{i_t}\right)\right]_n. \tag{2}$$

As an alternative approach, blockchain has been employed in a multi-agent decentralized ensemble approach based on sample exchange [42]. Each agent is assumed to have a small-scale initial local dataset, interacting with each other based on the exchange of their local data samples. When performing data iteration, the results are stored in a blockchain to ensure the validity and security of the data iteration. Anti-D chain is investigated in [43], a novel approach for detecting DDoS attacks in a blockchain framework. Lightweight classifiers are used with AdaBoost and a random forest ensemble strategy to create AI blockchain, connected with the original based on VR parallel tactics. The aim of that work is to achieve stronger generalization, performance and universality in order to overcome DDoS attacks in a P2P network. Danku Protocol [44] offers a blockchain-based approach for model selection on the Ethereum blockchain, allowing AI researchers to exchange machine learning models while adhering to highly secure principles. Since each node in the Danku ecosystem serves as a baseline model and the best performing model is chosen among the committees to be rewarded, the method is capable of working into the concept of ensemble model selection, allowing AI researchers to use the blockchain as a highly secure database of well-trained models. These architectures are mostly build on the application layer [45] of a blockchain system, in which the underlying trust of the consensus layer is used as a collateral in order to secure the DNN Inference process.

2.4. Decentralized Consensus Protocols

Decentralized consensus protocols can be employed for establishing a commonly accepted agreement between the individualized devices for a given specific task operating in a decentralized DNN framework. Such protocols have been used frequently in distributed computing, in order to protect and prevent any system abuse. The theory of decentralized consensus protocols begin with the introduction of the *Byzantine General's Problem*, which is formulated by Lamport et al. [46]. It is an allegory used to describe the challenges observed in the establishment and maintenance of a consensus between nodes in a decentralized system where the communications links are highly unreliable. Under that assumption, several type of failures can be detected, which may occur during the consensus process such as:

- *Crash Failures:* A case in which a node operating in the system may suddenly goes offline.
- *Fail-to-Stop Failures:* A case in which a node operating in the system may fail to stop its execution.
- *Byzantine Failures:* A case in which a node seems to operating normally but it fails maliciously to deliver a correct result.

To address the previously aforementioned issues, a series of *Byzantine Fault Tolerance (BFT)* replication protocols [47, 48] have been adapted, capable to detect and prevent faulty nodes to tamper the overall consensus process. In particular, BFT describes the ability of a system to normally function, even if some of the underlying components fail to respond or act maliciously. In BFT protocols, the implementation of the consensus among a set of nodes adopts the idea of the *State Machine Replication* where the state of a system is replicated across n deterministic replicas (e.g., nodes). Additionally, a traditional BFT consensus protocol must specifically guarantee -*Agreement*-, where no two non-faulty nodes arrive at different decisions, -*Termination*- where all non-faulty nodes eventually have a decision and -*Validity*- where the decision is proposed by some nodes [49]. To address a Byzantine consensus problem, one can extend an algorithm that satisfies the aforementioned properties. On the other hand, a BFT SMR algorithm needs to ensure the accuracy of the *Safety* and *Liveness* properties [49]. Safety states that every node successfully executes the same sequence of requests while Liveness states that all requests should be served.

SMR approaches often require a small number of nodes in order to prevent scalability and thus latency issues. Under that settings, several approaches have been introduced in order to improve the efficiency of classical BFT protocols by avoiding the use of expensive cryptographic signatures. For example, recent protocols like SBFT[50], Zyzzyva[51], BFT-Smart[52] and DR-BFT[53] have reduced the amount of transmitted and proceed signatures over the network by $O(n^2)$. Despite the fact that due to the message communication complexity, the classical SMR approaches cannot be employed in a DNN training application. Instead, the possibility of applying such approaches in a pure DNN inference scenario, where the computation complexity is limited, should be investigated. The SMR approaches are dealing with the FLP impossibility result [54] which states that there is no deterministic solution for the consensus problem in any asynchronous system even with a single crash failure. In order to over come this limitation, almost all BFT systems rely on partially synchronous assumptions in order to ensure liveness.

Practical BFT proposed by M.Castro and B.Liskov et al. [55] has been long-termly a state of the art consensus protocol for coping with Byzantine systems. It can tolerate up to a $1/3$ fraction of Byzantine faults in a system. When a client asks for the execution of any operation, the primary/leader replica receives the request, performs the requested operations and then, transmits the requests to rest replicas. The order of requests is then agreed by all replicas working together in a three-phase (pre-prepare/prepare/commit) agreement process. Every request is processed by each replica, which then replies to the relevant client. Because progress solely depends on the primary, the PBFT protocol ensures that safety is maintained even when there are timing violations [55]. The replicas initiate a view-change process to choose a new leader to coordinate the consensus procedure, after they determine that the current primary replica is flawed through the consensus procedure. When there are few participating replicas, the leader-based protocol performs admirably, but scalability problems might arise. We took inspiration from PBFT when developing the suggested AI - enhanced QoI consensus protocol since it is widely considered as the baseline for practically all BFT protocols developed so far.

A use case for merging between AI and blockchain can arise from the consensus layer, which can offer DNN training solutions under completely decentralized and verifiable settings, which aim to provide highly secure and efficient architectures. Deep learning models concentrate on training these data and making accurate predictions, while blockchain, as a fault-tolerant technology, secures and prevents data leakage, making training more reliable compared to centralized or cloud approaches since various types of attacks or data noise issues are prevented. In consensus layer, BlockML [17] aims at leveraging PoW in a concept where miners are being forced to solve a task by using the same input data and generating a competitive ML model. Once the miners successfully train the ML models, they immediately publish their results in blockchain and the test set is revealed. The miner with the highest-ranked accuracy model is winning the competition, its block is appended in the chain and is getting rewarded. In, Committee Consensus mechanism[13], consensus protocol aims at updating the global model using the federated learning approach. WecaCoin[56] on the other hand is a cryptocurrency developed to provide a decentralized and publicly available database of ML models. Miners are divided according to the DNN task and the Weca block holds the transactions, the previous hash and the machine learning model, used to verify the previous block. Proof of learning[57], attempts to solve large-scale DNN complexity issues using a secure mapping layer approach where consensus nodes are providing processing power for the given DNN task. Once the minimum training requirements are full-filed the miner is broadcasting the generated block and is rewarded.

3. Human society-inspired Decentralized DNN Inference

This Section describes the proposed Human society-inspired Decentralized DNN Inference, which consists of the: a) *Individual Agent Decision Aggregation* method in which each individual agent is requested to detect and eventually select the neighboring agents that will indeed help him to improve his individual performance and b) *Quality of Inference (QoI) Consensus Protocol* in which all agents are requested to collaborate with each other in order to reach a common decision agreement about the content of a given sample, and thus serve as a single inference rule.

3.1. Individual Agent Decision Aggregation (IADA)

In our pipeline, individual agents exchange information with other agents about their predictions at a given data sample $\mathbf{x}$ in order to calculate their own decisions. Let $\mathcal{A} = \{\alpha_1, \alpha_2, ..., \alpha_M\}$ is an index set of M agents performing decentralized inference and they have fixed communication links allowing to communicate directly to each other. We assume that each agent has a different locally installed classification model that produces a prediction for a given sample $\mathbf{x}$, i.e., $\hat{\mathbf{y}}_i = \boldsymbol{f}_j\left(\mathbf{x}; \boldsymbol{\theta}_j\right)$, where

θ_j are the learnable parameters of the j^{th} agent's model. At the inference stage, each agent generates a prediction based on the observed sample and transmits it to the rest of the network. A predetermined time interval, t, governs the communication process. Once enough transmissions have taken place between the agents and enough time has passed, each one makes a decision individually based on their own observations and the ones received by the rest of the agents.

After the time interval t has passed, agent α_i observes the prediction vectors received from the rest $M-1$ agents and based on a predefined condition decides whether to include them in the final aggregation step, or not. An agent is disqualified from the network if, for whatever reason, it was unable to successfully transmit its prediction over the designated time delay t. Moreover, if the agents' α_j prediction does not follow the predefined formation, is recognized from the rest of the agents as null and is totally ignored.

Specifically, in the set of the non-excluded predictions generated by each agent, we assumed that a soft-max activation function has been applied in order to transform the outputs into probability distributions, i.e., $\hat{\mathcal{Y}} \in [0,1]^C$, where C is the number of classes supported by the models. Aggregation with the predictions of other agents is performed for each class, based on two predefined conditions:

- **Probability-based Condition:** For a given sample $\mathbf{x}$, the prediction of agent a_j only takes into account agents that are more confident than itself, i.e.:

$$\hat{y}'_j = \frac{1}{M'} \sum_{i=1}^{M} b_i \hat{y}_i, \tag{3}$$

  where $b_i = 1$ if $\hat{y}_j < \hat{y}_i$ and $b_i = 0$, otherwise, and $M' \leq M$ is the number of agent predictions taken into account, based on the above condition. Additionally, by introducing a med$(\cdot)$ function, we can apply a median rule for the aggregation step formally defined as:

$$\hat{y}'_j = \underset{i \epsilon M}{\text{med}}(b_i \hat{y}_i), \quad \forall b_i \neq 0. \tag{4}$$

- **Weighted-based Condition:** As an alternative approach, we can consider a weight condition where weights are assigned to each agent based on their performance on training or on validation set. Let $\mathcal{W}$ represent a set of weights of the form $\mathcal{W} = \left\{w_j, \ \forall j \in M\right\}$ assigned for each agent in set $\mathcal{A}$. Then the condition for b_i parameter is defined as:

$$b_i = \begin{cases} 1, & \text{if } \ w_j \hat{y}_j < w_i \hat{y}_i \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

  The aggregation rule is set to be as described in Eq. (3), (4). The weights can be assigned manually by using e.g., the accuracy scores produced by each agent during the training process. The above two conditions can be combined or used individually.

3.2. Quality of Inference (QoI) Consensus Protocol

In this section, we propose a novel consensus protocol to be formed as a single inference rule and provide coordination in the individualized agent decisions under a fully decentralized structure, thus eliminating the need of any kind of centralized coordination. The QoI protocol can be thought as a hybrid consensus mechanism, where the core process is achieved by adapting the traditional BFT SMR approach, where $M \leq 2f+1$ total agents are needed to tolerate f *faulty* agents who may fail during execution or who may behave maliciously by transmitting tampered data to the neighboring agents. The normal operating agents are considered *honest*.

The QoI protocol addresses byzantine failure settings, i.e., a classical byzantine failure model applied in Decentralized Inference scenario is assumed. We assume a strong adversary model in which faulty agents may behave arbitrary in order to compromise the entire DNN Inference process. In particular, each agent receives a data sample $\mathbf{x}$ as input. Its goal is to communicate with the rest $M-1$ agents to reach a common agreement about the content of that sample as well as, the total ordering of the predicted classification results. The adversaries aim at achieving full control over the DNN Inference process and may even collude to each other in order to achieve the most damage and subvert the rules of the protocol. The Proof of Work byzantine agreement model, organizes and assigns decision-making authority to the most trustworthy agents so far, in order to handle samples in which the decision is in contention.

The system is designed under synchronous assumptions where the delivery schedule of predictions (e.g., messages) are guaranteed. Communication delays are minimized. We assume that each agent broadcasts a prediction (e.g., state machine command) to each neighbor agents ensuring that all agents receive all predictions in same order and thus each one of them maintains a complete list of the prediction history. So, QoI must guarantee:

- **Validity:** If an individual honest agent broadcasts a prediction $\hat{y}$, then every honest agent eventually receives $\hat{y}$.
- **Agreement:** If an individual honest agent decides a prediction $\hat{y}$, then every other honest agent must also decide $\hat{y}$.
- **Integrity:** A prediction $\hat{y}$ for sample $\mathbf{x}$ appears at most once in the delivery sequence of any honest agent.
- **Total Order:** The ordered sequence of predictions $\hat{y}_i$ and $\hat{y}_{i+1}$ for samples $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ must be the same for all honest agents.

QoI, as already mentioned, is a state machine replication protocol based on three sub operations: a) *view change*, b) *normal operation*, and c) *conflict decision agreement*. The view change operation coordinates the primary election process. The primary agent is responsible for beginning the consensus process by broadcasting its prediction about the given sample to the rest of the agents. The normal operation is the core execution of the protocol, where the proposed decision of the primary is evaluated in order to be universally accepted or rejected. In the case of universal acceptance, the primary is responsible for delivering the final decision of the network. When the proposed

primary's decision is universally rejected, a view-change process is triggered. Otherwise, when view change fails to elect a universally accepted primary, a conflict decision agreement mechanism is provisioned. *Conflict decision agreement* is used in order to handle situations that arise due to the nature of the DNN models themselves and not via a malicious behavior. In a conflict decision scenario, total ordering is not guaranteed in the specific decision time.

Agents operate through a sequence of actions known as *views*. Given that, QoI protocol operate in rounds in which each consensus round is defined as one execution of the normal operation process regardless if it is successful or not. Views describe the consensus rounds that are required in order for the network to reach a consensus about a given sample and defined as an index of the form $v \epsilon \mathcal{V}$, containing a sequence of testing pairs whose predictions have been scheduled in the time interval t. At each view, one agent is operating as *primary* while the rest $M-1$ agents are operating as *validators*. In the reminder of this work and for simplicity reasons, each view represents a single sample and its prediction of the form $(\mathbf{x}_i, y_i)$. Our goal is that every honest agent in M maintains an identical prediction history set defined as $\hat{\mathcal{Y}} = \left\{\hat{y}_{ij},\ \forall v \in \mathcal{V} \text{ and } j \in C\right\}$ given a set of $C = \{c_1, c_2, ..., c_C\}$ classes where $\|C\| = C$.

3.2.1. View Change

Leader Election. At any given time, all agents must be synchronized and begin from the same view or consensus round. At the beginning, a primary agent is selected from the set $\mathcal{A}$ to begin the consensus process for the first round. From this moment onwards, the rest $M-1$ agents will be working as "Validators". The primary agent is elected in circulating order, from the first to the last one, ensuring that everyone has a chance to be elected as long as they strict to the consensus rules. Let $a_p \epsilon \mathcal{A}$ be the primary agent, the election formula is defined as:

$$a_p = v \bmod |\mathcal{A}|, \tag{6}$$

where $|\mathcal{A}| = M$ and $v \epsilon \mathcal{V}$ represent the view we are currently working on.

View Change. Once the primary agent of the current view is detected as faulty, view change is performed in order to be replaced. Specifically, in the v^{th} view, the primary agent is promoting a prediction for the i^{th} sample of the form:

$$\hat{y_p} = \text{argmax}\left(f_p(\mathbf{x}_i)\right). \tag{7}$$

He is communicating its prediction $\hat{y_p}$ to the validators by constructing and broadcasting a pre-prepared message of the form $pre-prepare < a_p, \hat{y_p}, v_p, r_p >$ where a_p is the primary's identifier, $\hat{y_p} \in C$ is its predicted value for the current sample, v_p is the view index and r_p is the rewards he have collected so far.

Let $a_j \epsilon \mathcal{A}$ represent a random validator which has just received the primary's message. He calculate its prediction value as:

$$\hat{y_j} = \text{argmax}\left(f_j(\mathbf{x}_i)\right),\ a_j \in \{\mathcal{A} | a_j \neq a_p\}. \tag{8}$$

If its predicted value $\hat{y_j} \neq \hat{y_p}$ or $v_j \neq v_p$ then, from now onwards, the j^{th} agent recognizes the primary as faulty and so, he must immediately multi-cast a view-change message to the rest of the validators of the form $view-change < a_j, v_j + 1, vote_j, r_j >$. The parameter $vote_j$ is calculated via the following voting process as:

$$vote_j = \begin{cases} 1, & \text{if } \hat{y_j} \neq \hat{y_p} \text{ or } v_j \neq v_p \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

Once the validators receive that view change messages, they append them in a local log. If $\frac{\sum_{i=1}^{M-1} vote_i}{|\mathcal{A}|} \geq 0.5$ then the primary is globally recognized as faulty since it has lost the favor of the majority. The consensus round at this moment has failed, so the agents are switching to a new view in which a new primary will be elected via the leader election process and the consensus round will be restarted in the new view.

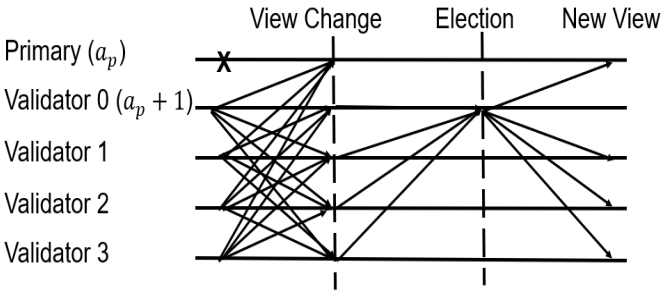

Figure 2: An illustration of the view change protocol applied to QoI.

Reward System. The reward system aims at working as an incentive mechanism for each primary, managing to retain the majority of the validators with its share. In other words, at any given time, if the current primary is honest, it is rewarded by a predetermined amount of quality points (q) as a reward for its honest work. However, if at any given time, the primary loses the favor of the majority of the validators, it is penalized according to a predetermined amount. As a result, each agent, must locally maintain a record of the rewards acquired by each agent so far defined as $\mathcal{R}$ where $\mathcal{R} = \{r_1, r_2, \ldots, r_M\}$. For a given primary a_p, the reward and the penalty are calculated as:

$$r_p = \begin{cases} q, & \text{if } \frac{\sum_{i=1}^{M-1} vote_i}{|\mathcal{A}|} < 0.5 \\ 0.5 r_p, & \text{if } \frac{\sum_{i=1}^{M-1} vote_i}{|\mathcal{A}|} \geq 0.5 \end{cases}. \tag{10}$$

Half of the collected primary points will be lost if it fails to get the majority's support. This stage serves as a quality control measure and an assurance that the primary agents would continue to perform honestly in order to be rewarded. However, if they fail to behave as such at any point, they will lose their high-quality status, which could harm them in the subsequent steps. In this way we are creating a quality index to guide us to the "by accident" faulty agents (e.g., poor trained models) or those with the malicious behavior in order to further exclude them for the conflict decision process.

3.2.2. Normal Operation

In a normal operation protocol, the determined primary is responsible for observing the decisions of the validators, deciding which of them is going to take under consideration and finally, constructing its final decision which will be then multicasted to all the honest validators. Given a primary agent a_p, we consider the following three predetermined conditions for the decision making process:

- **Class-based Condition:** Here, the primary agent compares its decisions with the rest $M-1$ validators. Specifically, let $\hat{y_p}$ be the prediction determined by the primary for the i^{th} sample calculated as described in Eq. (7) and $\hat{y_j}$ be the prediction calculated by a random validator $a_j \epsilon \mathcal{A}$ as determined by equation (8). The class-based condition defines that if $\hat{y_p} = \hat{y_j}$ then:

$$\hat{\mathbf{y}_p} = \left\{ \left(f_j(\mathbf{x}_i) \right), \text{ where } a_j \in \mathcal{A} \right\} \tag{11}$$

  and the final decision is produced by combining the selected validators decisions using average 3 or median rule 4.

- **Weight-based Condition:** We can assume a weight-based condition where the weights are assigned to each agent based on their performance on training or validation set. In such a set up, each agent must be able to provide evidence to the rest $M-1$ agents whose assigned weight is not randomly selected but is determined based on the true performance of its locally trained model. Let $\mathcal{W}$ represent a set of weights of the form $\mathcal{W} = \{w_i, \ \forall i \in M\}$ assigned to each agent in the set $\mathcal{A}$ and w_p represent the assigned to the primary agent a_p weight and w_j the weight assigned to a random validator $a_j \epsilon \mathcal{A}$. Then, the weight-based condition states that if $w_p < w_j$ then the primary considers the decision of the j^{th} agent as reable and then:

$$\hat{\mathbf{y}_p} = \left\{ \left(f_j(\mathbf{x}_i) \right), \text{ where } a_j \in \mathcal{A} \right\} \tag{12}$$

  and the final decision is produced by combining the selected validators decisions using the average 3 or the median rule 4. Regarding the weight-based condition, the primary is considering only the agents achieving greater overall performance than him.

- **Hybrid Condition:** Here the decision rule is mixed among the weight and the classed-based condition. Specifically, the hybrid rule states that if $w_p < w_j$ and $\hat{y_p} = \hat{y_j}$ then:

$$\hat{\mathbf{y}_p} = \left\{ \left(f_j(\mathbf{x}_i) \right), \text{ where } a_j \in \mathcal{A} \right\}. \tag{13}$$

  and the final decision is produced by combining the selected validators decisions using average 3 or median rule 4. Hybrid Condition ensures that the primary is in consensus for the predicted value while focusing on the validator who has greater overall performance, comparing to the primary, boosting in that way the decision process.

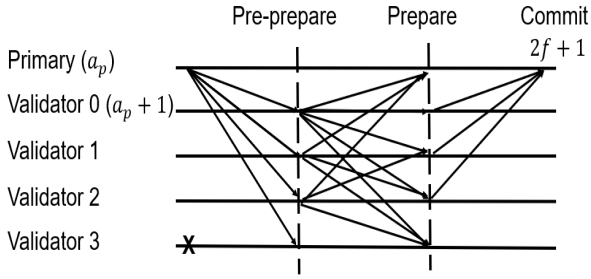

Figure 3: An illustration of the Normal Operation applied to QoI.

Regadless the condition, the final decision of the primary is then calculated as:

$$\hat{y}_{final_p} = \text{argmax}\left(\hat{\mathbf{y}_p}\right). \tag{14}$$

Once the primary has reached to a final decision, he multi-casts to all agents, including himself, an encrypted *prepare* message of the form $prepare < a_p, \hat{y_p}, \hat{y}_{final_p}, v_p, r_p >$ where a_p is the primary's sequence number, $\hat{y_p}$ is its initial predicted value for the current sample, $\hat{y}_{final_p}$ is the final agreed decision, v_p is the view index and r_p is the collected so far rewards. Once a validator receives a prepared message from the primary, it first ensures its validity by observing if the v_p matches with its own view number and weather $\hat{y_p}$ matches its own locally produced prediction. If the prepare message is indeed valid, it is transmits the prepared message to the rest validators. The validators awaits for $2f+1$ identical messages from different agents in order to proceed to the commit phase, where a $commit < a_j, \hat{y_j}, \hat{y}_{final_p}, v_j, r_j >$ is transmitted. Once the validators ensure its validity, it transmits it back to the primary. If the primary receives $2f+1$ identical commit messages from different validators, it recognizes that the consensus is achieved for that specific sample. The consensus round is initialized.

3.2.3. Conflict Decision Agreement

In this subsection, weather the agents working in QoI protocol will be unable to classify a sample at all is discussed. Conflict Decision Agreement occurs when the sample's content is too complex for the majority of the agents to understand. A sample in which the majority of the agents are unable to agree on a commonly accepted decision is referred as conflict sample.

Specifically, for any given sample, if all possible primaries have failed to maintain the majority of the validators with their share during view change operation, the framework cannot establish any decision for that specific sample. At this point, for the conflict sample i^{th}, the agents according the rewards $\mathcal{R}$ they have collected so far are ordered, in a descending order of the form:

$$r_{j+1} < r_j, \quad \text{where } r_{j+1}, r_j \epsilon \mathcal{R}. \tag{15}$$

The final decision for that sample is achieved via *Group of Experts Rule*, where agents, are organized in groups of experts depending on the correlation of their decisions, where the most qualified group eventually decides for the sample. If the process fails, the *Most Honest Rule* is applied where the agent with

the highly recorded rewards is qualified to perform the final decision.

- **Group of Experts Rule:**

  A normal operation is executed for each agent in the set $\mathcal{A}$, starting from the agent with highest reward score and all the way to the agent with the lowest. When the method is applied for the first agent, it will simultaneously operate as primary agent $a_p \in \mathcal{A}$ and produce a prediction via the Eq. (7) of the form $\hat{y_p}$.

  At this point, the primary agent observes the predictions produced by the rest agents. If any of the other agents agrees with him then they group up together and jointly propose the prediction $\hat{y_p}$. Let $group_i$ be a group of agents and $a_j \in \mathcal{A}$ be an agent that already jointed the group, then the rewards he has collected so far are summed up with the rewards of the primary's group, reflecting in that way the total reward of the group as $group_i = r_p + r_j$. From now on, agent a_j is excluded from the rest process since he is already part of a group and there is no reason for him to create it's own. These steps are applied on every agent in the set $\mathcal{A}$ forming groups of decided predicted values and total rewards for the i^{th} conflict sample.

  The final decision for the i^{th} sample is achieved once, the consensus for the next sample $i + 1$ is reached. Specifically, if agent a_j is determined as primary agent for the sample $i+1$ and successfully passes the view change process, he is deciding for the sample i+1. Its given reward is added in the total reward of the group that is member for the previous sample i. At this point, if agent $a_j \in group_i$ and if:

  $$\frac{group_i}{\sum_{i=1}^{M} r_i} \geq 0.51 \tag{16}$$

  then the agent a_j, is responsible to decide for the i^{th} conflict sample as:

  $$\hat{y_j} = \text{argmax}\left(f_j(\mathbf{x}_i)\right). \tag{17}$$

- **Most Honest Rule:** If the group of the primary agent for the sample $i + 1$ fails to pass the 0.51 quality threshold, the decision is made by the agent with the highest reward score achieved. Specifically, the primary agent is determined as:

  $$a_p = \text{argmax}(r_j),\ \forall r_j \in \mathcal{R} \tag{18}$$

  Then, the decision for the conflict sample i is the prediction achieved by the agent a_p.

Finally, our protocol can be considered as a hybrid consensus mechanism based on DNN Inference architecture, aiming at providing the Safety, Liveness and Fault Tolerant properties when dealing with non-conflict samples. Specifically, the conditions are formulated as:

- **Safety:** Agents will agree on the same value, which is proposed by the primary agent.
- **Liveness:** Eventually, every honest agent should decide upon a prediction value.
- **Fault Tolerance:** All honest agents must and will agree on the same value.

When the decision for non-conflict samples is reached, Safety and Liveness property is guaranteed. If for a primary agent a_p the proposed prediction $\hat{y_p}$ is true, any prediction $\hat{y_{p'}}$ is false for any honest agent a_j (even if $a_i = a_j$), for any $\hat{y_{p'}} \neq \hat{y_p}$. This is guaranteed because two honest agents agree on the same sequence number of samples that commit locally on their prediction history record. Honest agents also agree on the same sequence of views since a prediction for a view $v^{'} > v$ without the first decision for the view v is not accepted. In a conflict sample scenario, the Liveness property is no longer valid as the decision on that point can not be based on deterministic assumptions. Instead a probability-based setting is adopted, by requiring the agents to agree on the probability that some predicted values are correct. In order to distinguish between the agents with poor performance and the ones with the best achieved so far, the Reward System is vital.

4. Experiments

In this section experimental results in various datasets are presented among with illustration approaches. The benefits of applying the proposed Individual Agent Decision Aggregation approach are presented. The performance of the decentralized QoI consensus protocol is compared with the traditional centralized DNN inference structure [19]. Experiments are applied in both synthetic and real-world datasets. The process of developing synthetic datasets is covered first. The output of a DNN model is built directly, simulating the generated prediction set of an applied pre-trained model over a specified test set, rather than following the more conventional method of creating a feature space using Gaussian rules to generate synthetic data. Each individual agent performance is then observed and discussed as we proceed to the numerical results for the IADA method. Last but not least, we implement the QoI consensus protocol in both the base AI-empowered agents and the enhanced versions of them, after implementing the collaborative inference approach. We compare the outcomes with centralized aggregation rules such as weighted average and majority voting, under the premise of a straightforward "committee of experts" ensemble approach.

4.1. Synthetic Data Generation

Our objective is to directly generate a set of predictions, simulating the predicted values calculated by hypothetical DNN models of the form $\mathcal{F} = \{f_1, f_2, ..., f_n\}$. These predictions serve as the baseline agents, solving a specific computer vision classification task (e.g., Image Recognition) using a pre-defined set of true labels.

In a supervised driven method, the dataset is defined as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), \forall i \in N\}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in [0,1]^C$, here, we focus to construct the space y and totally ignore the space $\mathbf{x}$. So, in the final form, our true label set will be defined as $\mathcal{Y} = \left\{\left(y_{ij}\right), \forall i \in N \text{ and } j \in C\right\}$. Given a number of C classes, defined as $C = \{c_1, c_2, ..., c_c\}$ where $|C| = C$. We create the space $\mathcal{Y}$ by randomly choosing values in a range $[1, C]$. Each generated number is encoded by using the One Hot Encoding method in order to serve as a vector of the form $\mathbf{y}_i \in [0,1]^C$. Given a number of M agents, the set $\mathcal{Y}$ is cloned for each agent and the global prediction set is formally defined as $\hat{\mathcal{Y}} = \left\{\left(\hat{y}_{ijk}\right), \forall i \in M, j \in N \text{ and } k \in C\right\}$. For the i^{th} agent and the j^{th} sample, the predicted value $\hat{y}$ is replaced with a random selected number calculated using a uniform distribution between two float numbers a and b. For generating agents with diversity on their predictions, we must define the follow strategies:

- **High Range:** The random float assigned to the prediction $\hat{y}$ is calculated in a range for $a = 0.8$ and $b = 1.0$. We assume that models with such probabilities are quite confident about their decisions.
- **Medium Range:** The random float assigned to the prediction $\hat{y}$ is calculated in a range for $a = 0.7$ and $b = 0.8$. We assume that models with such probabilities are less confident about their decisions.
- **Low Range:** The random float assigned to the prediction $\hat{y}$ is calculated in a range for $a = 0.5$ and $b = 0.7$. We assume that models with such probabilities are unconfined about their decisions.

The rest values in the positions $\hat{y}_k$, $\forall k \in C$ and $\hat{y_k} \neq \hat{y}$, are replaced with random floats calculated in a range of $a = 0$ and $b = 0.1$ until the overall summation of vector $\hat{\mathbf{y}}_k$ results in 1.0. This strategy gives us the ability to generate agents with diverse probability distributions and also provides us with a way to define agents with different confidence levels on their produced predictions.

Finally, we should be able to create misclassified samples in order to reduce and customize the accuracy score produced by each agent. This is achieved by calculating, for each agent, a random float value $rand_{value}$ in a range of our choice and then constructing a boolean condition of the form $count \leq N * rand_{value}$. While the condition is true, we select a random sample $j \in N$ and use the vector $\hat{\mathbf{y}}_{jk}$ to calculate the predicted value and the position of the predicted value as:

$$pred_{value} = \max\left(\hat{\mathbf{y}}_k\right), \forall k \in C, \tag{19}$$

$$pred_{pos} = \text{argmax}\left(\hat{\mathbf{y}}_k\right), \forall k \in C. \tag{20}$$

then we replace the predicted value with the one in a random position in range $[1, C]$.The predicted value is also decreased by a random float calculated in a range $[0, 1]$. The procedure described above is applied for each agent constructing the global set of predictions $\hat{\mathcal{Y}}$, in which we apply a soft-max activation function in order for it to follow a probability distribution.

Table 1: Synthetic Datasets Set-Up.

Dataset	n_{Samples}	n_{Classes}
D1	1k	10
D2	6k	100
D3	10k	10
D4	10k	25
D5	10k	120
D6	15k	60
D7	100k	10

4.2. Evaluation on Synthetic Data

Experiments are conducted with varied numbers of agents for each dataset based on the configurations shown in Table 1. Some of them simulate well-trained models with high confidence scores in their predictions. Others are less accurate but nevertheless handle high levels of confidence and others are mixed based on agents with high accuracy but low confidence in the produced predictions and vice versa. The purpose of those experiments is to demonstrate, as illustrated in Table 2, that the decentralized character of collaborative decision-making has the potential to be advantageous for all agents.

To clarify some of the following notations, we refer to the AI-powered agents as Base Agents and demonstrate their individual accuracy performance over the corresponding dataset. Secondly, DA stands for Decentralized Agents after the application of the proposed IADA inference framework over the probability condition, as stated in Subsection 3.1. DWA stands for Decentralized Weight-based Agents after the application the weight-based condition. The number of agents M varies between $5 - 11$ depending on the experiment. In order to keep track of each individual agent performance, their total number is kept small. In our method, both in DA and DWA conditions, the generally accepted aggregation rule of the median or average is applied. Median appears to outperform averaging in most of the experiments. Thus, our results are presented under the median aggregation rule.

The base agents overall accuracy scores are presented in Table 2. The agent performance in each dataset is varying between 49% to 87%. As observed in the experiments where the base line agents appear to have a well-established "knowledge" about the content, the DA and DWA results demonstrate than we can distinguish them in a descending order and the top rated agents can still gain a boosted information improving their accuracy and thus generalization performance by around 0.20 to 1%. The poorly performing agents gain an overall boost between 2% to 5%. This occurs due to the fact that we are focusing on improving them on each class rather than applying the method directly on final prediction. On the other hand, when we involve low performance agents with average 45% to 60% , the individual agents gain an boost from 1% to 7%. Such behavior is observed mostly because the divergence between them is high enough in order to allow the poorly performing agents to receive an incremental rise of their ability to recognize the underlying content of a given sample $\mathbf{x}_i$.

Table 2: Results on Synthetic Data.

Experiment[a]	Dataset	Accuracy (%)										
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
Base Agents		53.30	50.30	49.60	52.10	48.90	53.40	50.81	-	-	-	-
DA	D1	53.80	53.10	51.30	52.80	51.10	54.30	53.20	-	-	-	-
DWA		54.00	52.60	51.00	52.90	51.10	53.30	53.50	-	-	-	-
Base Agents		53.10	55.80	51.50	55.41	54.30	52.80	51.10	51.90	54.20	-	-
DA	D1	54.80	57.10	54.50	55.10	55.20	54.50	53.60	53.30	55.40	-	-
DWA		54.60	55.70	54.60	55.10	54.90	54.70	53.80	53.60	53.50	-	-
Base Agents		69.02	68.30	73.75	71.45	70.00	73.53	70.47	68.07	67.12	-	-
DA	D2	71.62	71.40	74.07	72.62	71.73	73.92	72.35	70.97	70.40	-	-
DWA		71.52	71.38	73.89	72.12	71.56	73.45	72.21	70.64	70.06	-	-
Base Agents		52.63	53.03	53.52	50.37	54.35	51.25	54.54	54.62	53.54	54.41	50.51
DA	D3	54.17	54.23	55.00	52.40	54.80	52.80	55.25	55.12	55.01	55.41	52.79
DWA		54.34	54.21	54.91	52.91	54.68	53.27	54.75	54.23	54.91	54.83	53.34
Base Agents		85.46	84.34	85.09	84.04	84.83	84.68	85.22	-	-	-	-
DA	D3	87.47	86.80	87.31	86.85	87.08	87.21	87.43	-	-	-	-
DWA		87.09	86.89	87.19	86.97	87.00	87.38	87.24	-	-	-	-
Base Agents		87.55	87.91	88.00	87.88	87.81	-	-	-	-	-	-
DA	D4	88.68	88.90	89.12	89.03	88.99	-	-	-	-	-	-
DWA		88.70	88.90	89.10	89.01	88.76	-	-	-	-	-	-
Base Agents		69.71	68.68	68.75	68.18	67.89	-	-	-	-	-	-
DA	D5	70.01	68.94	68.81	68.61	68.21	-	-	-	-	-	-
DWA		69.91	68.80	68.85	68.49	68.22	-	-	-	-	-	-
Base Agents		84.29	85.11	84.55	84.87	83.70	84.71	85.69	-	-	-	-
DA	D6	84.70	85.21	84.87	85.25	84.43	85.05	85.79	-	-	-	-
DWA		84.67	85.29	84.84	85.16	84.52	85.01	85.81	-	-	-	-
Base Agents		81.58	82.28	82.17	82.37	81.76	82.54	81.88	81.63	82.56	83.37	82.49
DA	D7	84.07	84.52	84.55	84.65	84.16	84.75	84.31	84.13	84.68	85.22	84.70
DWA		84.37	84.72	84.74	84.84	84.42	84.95	84.63	84.39	84.89	84.87	84.88
Base Agents		75.22	74.87	74.93	76.28	73.79	75.72	73.71	-	-	-	-
DA	D7	77.09	76.86	76.92	77.67	76.24	77.37	76.17	-	-	-	-
DWA		77.25	77.13	77.23	77.21	76.54	77.55	76.53	-	-	-	-

[a] DA: Decentralized Agents; DWA: Decentralized Weight-based Agents;

4.3. Evaluation on Real Datasets

We decided to use a combination of DNN models in order to demonstrate a heterogeneous model environment and evaluate our framework in real-world scenarios. In some experiments, the pre-trained on a specific set of data agents, reflect their true performance as best as possible, whereas when trained from scratch with the intention of producing poor performance.

Experiments were conducted on five well-known image classification datasets: Fashion MNIST [58], Cifar 10 and 100 [59], STL-10 [60] and Street View Housing Numbers (SVHN) [61]. Fashion MNIST consists of 60K training and 10K testing gray scale images equally distributed among 10 object classes. Cifar-10 and 100 consists of 50K training and 10K testing gray scale images equally distributed among 10 and 100 object classes respectively. STL-10 consists of 5k 96×96 labeled images for training and 8k for testing equally distributed among 10 object classes. Finally, SVHN dataset consists of 73257 training and 26032 testing images of labeled digits of street view numbers distributed in 10 classes. In F-MNIST the used pre-trained models are VGG16, DesNet161, EfficientNet, MobileNet v2, ShuffleNet v2. In Cifar10-100 the used pre-trained models are VGG11, VGG16, ResNet20, ResNet32, MobileNet v2, ShuffleNet v2, RepVgg-a1. Additionally, in STL-10 dataset, the used pre-trained models are ResNet20, ResNet32, MobileNet v2, ShuffleNet v2, RepVgg-a1. Finally in SVHN, the used pre-trained models are VGG11, VGG16, ResNet20, ResNet32, MobileNet v2, ShuffleNet v2, RepVgg-a1.

As we can see in Table 3, in F-Mnist and Cifar-10, all of the agents benefit from an increase in accuracy, with the highly performing agents in each dataset gaining around 1% overall, while the poorly performing ones gain around $4-5\%$. In addition, in datasets such as STL-10 and Cifar-100, the accuracy boost is increasing significantly, with the highly performing agents achieving an overall increase around $5-8\%$, while the lower performing agents can achieve an overall increase of up to 10% in some cases. In those experiments we use median as the general aggregation rule for the combination of the chosen agents in DA condition while in DWA conditions the rule is set to be the averaging.

Also, in Table 9 we further investigate the behavior of the IADA method for each agent in per class level results. We choose the STL-10 dataset since it achieves the best performance in real world scenarios. As we see, each base agent is learning a specific class better than the others. By applying

Table 3: Results on Real Datasets.

Experiment	Dataset	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base	F-MNIST [58]	90.63	87.85	90.99	90.34	91.02	-	-
DA		91.26	91.07	91.90	91.93	92.14	-	-
DWA		91.27	91.08	91.91	91.94	92.15	-	-
Base	STL-10 [60]	73.30	68.70	66.84	71.43	70.65	-	-
DA		78.12	77.39	76.89	77.55	78.00	-	-
DWA		77.70	77.12	76.71	77.61	78.06	-	-
Base	SVHN [61]	90.90	91.27	88.36	91.69	90.72	89.43	94.33
DA		93.11	93.12	92.69	93.28	93.05	92.76	94.37
DWA		92.97	93.06	92.66	93.32	93.19	92.89	94.34
Base	Cifar-10 [59]	92.18	92.65	91.53	93.68	92.57	89.96	94.51
DA		94.68	94.88	94.38	94.73	94.74	94.27	95.32
DWA		94.77	94.73	94.61	94.79	94.99	94.53	95.11
Base	Cifar-100 [59]	62.67	63.41	66.15	69.81	64.47	59.83	69.62
DA		70.40	73.24	73.23	73.80	73.42	72.90	74.75
DWA		70.32	73.34	73.37	73.78	73.57	73.02	74.93

the proposed methods the, shared among the agents, knowledge improves the generalization performance of each agent individually. This occurs due to the fact that each agent has a diverse discrete probability distribution where the knowledge of each class is divided under the setting of the given base line model.

It is also critical to present a mix of well-trained and poorly trained agents in order to investigate their interactions. The low performing agent should not influence the decisions produced by the well trained ones. We can assume that poor agent behavior results from either poorly trained base models or malicious behavior, in which some agents broadcast completely tampered results to the others, poisoning either the model itself or the produced prediction set. As we see in Tables 10,12 we simulate 2 different scenarios and testing it in both synthetic and real datasets. To begin, we distinguish between honest and faulty agents. Honest agents typically have good performance, whereas faulty agents are tampered. Their performance is significantly reduced in comparison to the honest. In such settings, as a general aggregation method, both in probability and weight basted condition, the average rule is used. As we can see, honest agents can successfully ignore the faulty ones and even improve their performance through collaboration. In addition the faulty ones can correct themselves through the well-established knowledge honest agents provide. This behavior can provide us with a fault-tolerant resilient schemes. While some agents consistently make incorrect predictions on a given sample, the well-trained ones are unaffected and will correct their performance by receiving a significant boost from the "experts."

In particular, when the performances are mixed in a "honest-faulty" scenario, the accuracy of each honest agent is still enhanced but with lower rates such as 0.1 to 0.4%, whereas the faulty agents' performance can be boosted up to 40% in some cases. This demonstrates that we can provide substantial assistance to those in need while also preventing faulty actors from tainting the good actors as long as they are in the minority. We use two different scenarios to investigate the number of faulty agents that our system can handle. In the first scenario, faulty agents are set to be 3 out of 7 total agents, yielding the previously discussed results, whereas in the second scenario, faulty agents are set to be 5 out to 7 total agents, yielding the previously discussed results. In the second scenario, we can clearly see that performance remains constant, but the faulty agents gain less accuracy boost than before.

4.4. Cost vs Accuracy

In this section we study the efficiency of our IADA method in an economic costly environment. Specifically, we assume that each agent has the ability to cost their produced predictions applied in a decentralized collaborative market place for exchanging decisions. Under that settings, each individual agent, in order to be able to improve itself, they must select a subset of neighboring agents to take into account and thus to pay a fee in them in order to gain the ability to use their predictions. The centralized majority voting aggregation rule application in that setting, for each agent individually, requires all agents involvement into the decision making process thus resulting a total cost of 100%. In the IADA method 3.1, each agent can instead select who would lessen or not, reducing significantly the overall cost, in some cases even to a 40% or 60%, for him self as we can see in Table 4.

Furthermore, Tables 6, 7, 8 present the results of the experiments conducted by bounding the associated cost in thresholds of 50%, 35% and 25% in order to study the cost-efficiency ratio. In particular, when the cost is bounded on 50% and 35% threshold, the observed drop in the agents accuracy is relatively small, making them capable to atomically improve themselves with a significant lower cost. On the other hand, when the cost threshold is dropping to 25%, the finally boosting for each of them, drops significantly, showing the amount of agents each one should uses in order to present well-established final decisions.

Table 4: Total Cost per Agent.

Experiment	Dataset	Cost (%)						
		A1	A2	A3	A4	A5	A6	A7
DA	F-MNIST [58]	86.75	46.25	67.50	37.88	57.00	-	-
DWA		86.13	47.87	72.38	35.00	56.75	-	-
DA	STL-10 [60]	61.25	59.22	53.47	67.72	56.63	-	-
DWA		59.59	61.06	55.06	67.44	56.84	-	-
DA	SVHN [61]	49.20	58.21	60.63	60.89	54.11	46.25	70.71
DWA		51.16	61.25	58.57	59.38	53.84	46.43	69.38
DA	Cifar-10 [59]	69.02	54.91	71.79	48.21	61.70	63.13	31.25
DWA		76.87	57.50	65.98	46.16	59.02	67.95	26.52
DA	Cifar-100 [59]	52.12	65.84	67.23	79.51	49.03	50.94	35.33
DWA		54.33	66.28	67.99	76.22	51.75	48.93	34.49

Table 5: Comparison of Aggregation Methods in Real Datasets

Experiments	Dataset	Centralized Voting Rules		QoI Consensus Protocol		
		Weight Average	Majority Voting	Class Rule	Weight Rule	Hybrid Rule
Base Agents		94.09	93.75	93.58	94.12	93.62
DA	SVHN [61]	-	-	94.02	**94.20**	93.99
DWA		-	-	93.97	94.16	93.95
Base Agents		95.12	95.05	95.04	95.27	94.97
DA	Cifar-10 [59]	-	-	95.05	95.21	95.14
DWA		-	-	95.16	**95.29**	95.17
Base Agents		74.65	73.96	71.47	71.42	71.64
DA	Cifar-100 [59]	-	-	73.84	74.01	73.85
DWA		-	-	74.80	**74.96**	74.74
Base Agents		**92.51**	92.01	91.94	92.33	92.01
DA	F-MNIST [58]	-	-	92.15	92.28	92.17
DWA		-	-	92.16	92.13	92.14
Base Agents		**80.11**	79.29	78.19	78.21	78.49
DA	STL-10 [60]	-	-	79.49	79.44	79.34
DWA		-	-	79.19	79.07	79.27

4.5. Comparison with centralized Aggregation

In this section, we compare the QoI consensus protocol to simple centralized ensemble aggregation techniques such as a "commit of experts" architecture with weighted average and majority voting as combine rules. Our main goal here is to present performance that is nearly on pair with centralized techniques. After all, the main difference between the centralized and decentralized approaches we proposed is in the architecture of the setting. We investigate how many faulty agents we can tolerate in our protocol and discuss how they affect the overall performance. Firstly, in Table 5, we present results in real-word datasets constructed by pre-trained models, with the goal of producing results as close as possible to the centralized aggregation methods. Secondly we proceed by employing in each experiment a set of "faulty" agents, defined either as low performance trained models or under completely malicious settings.

As we see in Table 5, in most of the cases, we achieve a comparable performance with centralized architectures, and in some datasets such as SVHN and Cifar-10 we manage to improve the accuracy levels in base line agents. In almost all of the experiments, after we apply the IADA technique in the base line agents we are in a situation to improve the decentralized consensus performance. The reason of that observation is because each agent firstly improves their own decisions individually by applying the decision making process introduced in the section 3.1. Each agent can indeed distinguish between the agents that will have a positive effect on him and those that will have a negative effect in any given sample. After that, using the consensus process in the already enhanced agents, we can provide a final decision under a fully decentralized structure, simulating how humans make decisions based on feedback from their environment. This architecture does not rely on a straightforward decision-making process, as we see in centralized architectures, but rather on a collaborative setting that can improve the final process by using the QoI protocol, where the occasionally observed ties in majority voting, can be completely avoided through the reward system and the conflict sample management system. In other words, ties are resolved by the agent or agents determined to be the most honest during the consensus process, rather than by a convince rule.

Continuing, in Tables 11 and 13 we study two attack types. In the first one, faulty agents are defined as bad trained models attempting to achieve inference. In that case, the predicted probability of each class in each sample has a small correla-

tion with the honest agents, which are presented as well trained models, and thus their generalization and prediction probability is significantly higher. As expected, the centralized weighted average technique outperforms the majority voting and QoI consensus in the base agent situation. However, after boosting each agent individually, better consensus is reported. This emphasizes how important is for each agent to observe and take accurate decisions about the agents that would be able to improve him or not. In Tables 12,13, a scenario in which the faulty agents are attempting to tamper the overall consensus process by transmitting totally randomized prediction results is applied. In Table 12 we observe how the IADA method is autocorrecting the malicious agents. The amount of the faulty ones are within the bounds of the $n \leq 2f + 1$ rule. Both the IADA method and the QoI protocol can recognize them and reduce the influence of them in the whole process. On the other hand, when the rule is no longer applicable, the consensus protocol can no longer work properly. Those experiments work as a visualized proof for the faulty/honest ratio in which our algorithm can tolerate.

5. Conclusion

In this work, a decentralized decision-making process for a DNN inference framework has been proposed. Inspired by human societies, this framework allows each individual agent to make their own decisions by exchanging and aggregating information with other agents in their network, in an effort to improve individual performance. In order to build an agreement among the several AI-agents that would ultimately form one inference rule, a novel consensus protocol is also presented. All individual agents maintain a replica of the prediction history that establishes the overall system knowledge and judgments regarding particular samples. Additionally, it has been shown that by adopting a fault-tolerant inference architecture, missbehaving AI agents can be punished in a way that dramatically lessens their ability to influence the decisions of good agents. Our classification task studies have demonstrated that the suggested methodologies forms a secure decentralized inference framework, which hinders adversaries from interfering with the whole process and delivers performance that is comparable to centralized decision aggregation techniques.

Particularly, it was demonstrated that by choosing the neighbors who will benefit them the most, individual agents can actually increase their performance greatly. Additionally, a cost-efficiency approach is described, in which the process of choosing individual agents can significantly lower the overall inference cost. The QoI consensus mechanism has also shown its resistance to malicious attacks while its accuracy performance is comparable with centralized aggregation techniques. However, since the experiments are not carried out on an actual decentralized DNN system, neither the algorithm's scalability nor the latency experienced during communication among AI agents are thoroughly investigated. Finally, since the QoI protocol is particularly created be utilized as a consensus layer algorithm, the outcomes of this work reveal significant potential benefits of implementing a genuinely decentralized DNN inference process within a blockchain network.

The following study areas could be further explored after our work. First, the feasibility of direct implementation in an AI blockchain network must be examined. That could be considered as a challenging task because the blockchain network needs to be designed in a way that enables AI task to be carried out and allows the penalties and rewards amassed by the agents during the process to actually be used in something that matters. Additionally, the algorithm needs to be coupled with other consensus protocols in order to scale sufficiently. Another research path, in order to study how SMR-based approaches can be used in training situations in order to prevent the overload of message communications, must be considered.

Acknowledgment

This work has received funding from the European Union's European Union Horizon 2020 research and innovation program under grant agreement 951911 (AI4Media). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–44.

[2] J. Chen, X. Ran, Deep learning with edge computing: A review, Proceedings of the IEEE 107 (8) (2019) 1655–1674.

[3] A. Yousefpour, B. Q. Nguyen, S. Devic, G. Wang, A. Kreidieh, H. Lobel, A. M. Bayen, J. P. Jue, Resilinet: Failure-resilient inference in distributed neural networks, arXiv preprint arXiv:2002.07386 (2020).

[4] S. Teerapittayanon, B. McDanel, H. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 328–339.

[5] A. Yousefpour, B. Q. Nguyen, S. Devic, G. Wang, A. Kreidieh, H. Lobel, A. M. Bayen, J. P. Jue, Resilinet: Failure-resilient inference in distributed neural networks (2020).

[6] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, Mobile edge computing: Survey and research outlook, CoRR (2017).

[7] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, CoRR (2017).

[8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2704–2713.

[9] N. Shlezinger, N. Farsad, Y. C. Eldar, A. J. Goldsmith, Inference from stationary time sequences via learned factor graphs, CoRR (2020).

[10] N. Shlezinger, Y. C. Eldar, S. P. Boyd, Model-based deep learning: On the intersection of deep learning and optimization (2022).

[11] R. A. Cohen, H. Choi, I. V. Bajić, Lightweight compression of intermediate neural network features for collaborative intelligence, IEEE Open Journal of Circuits and Systems 2 (2021) 350–362.

[12] M. Merluzzi, A. Martino, F. Costanzo, P. Di Lorenzo, S. Barbarossa, Dynamic ensemble inference at the edge, in: 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6.

[13] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, Q. Yan, A blockchain-based decentralized federated learning framework with committee consensus, IEEE Network 35 (1) (2020) 234–241.

[14] X. Chen, J. Ji, C. Luo, W. Liao, P. Li, When machine learning meets blockchain: A decentralized, privacy-preserving and secure design, in: 2018 IEEE international conference on big data (big data), IEEE, 2018, pp. 1178–1187.

[15] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Cryptography Mailing list at https://metzdowd.com (03 2009).

[16] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics, M. Harvilla, A proof of useful work for artificial intelligence on the blockchain (2020).

[17] A. Merlina, Blockml: A useful proof of work system based on machine learning tasks, in: Proceedings of the 20th International Middleware Conference Doctoral Symposium, Association for Computing Machinery, New York, NY, USA, 2019, p. 6–8.

[18] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8 (02 2018).

[19] J. Kittler, M. Hatef, R. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.

[20] C. Ju, A. Bibaut, M. J. van der Laan, The relative performance of ensemble methods with deep convolutional neural networks for image classification (2017).

[21] S. Mao, L. Jiao, L. Xiong, S. Gou, B. Chen, S.-K. Yeung, Weighted classifier ensemble based on quadratic form, Pattern Recognition 48 (5) (2015) 1688–1706.

[22] Y. Zhang, S. Burer, W. Nick Street, K. P. Bennett, E. Parrado-Hernández, Ensemble pruning via semi-definite programming., Journal of machine learning research 7 (7) (2006).

[23] I. Partalas, G. Tsoumakas, I. P. Vlahavas, Focused ensemble selection: A diversity-based method for greedy ensemble selection, in: M. Ghallab, C. D. Spyropoulos, N. Fakotakis, N. M. Avouris (Eds.), ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings, Vol. 178 of Frontiers in Artificial Intelligence and Applications, 2008, pp. 117–121.

[24] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the twenty-first international conference on Machine learning, 2004, p. 18.

[25] Q. Hu, D. Yu, Z. Xie, X. Li, Eros: Ensemble rough subspaces, Pattern Recognition 40 (2007) 3728–3739.

[26] D. Hernández-Lobato, G. Martínez-Muñoz, A. Suárez, Statistical instance-based pruning in ensembles of independent classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 364–369.

[27] S. hyeun Park, J. Fürnkranz, Efficient prediction algorithms for binary decomposition techniques (2012).

[28] Z. Tao, Q. Li, esgd: Communication efficient distributed deep learning on the edge, in: I. Ahmad, S. Sundararaman (Eds.), USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2018, Boston, MA, July 10, 2018, USENIX Association, 2018.

[29] J. Choi, Z. Hakimi, J. Sampson, V. Narayanan, Byzantine-tolerant inference in distributed deep intelligent system: Challenges and opportunities, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9 (3) (2019) 509–519.

[30] A. Yousefpour, S. Devic, B. Q. Nguyen, A. Kreidieh, A. Liao, A. M. Bayen, J. P. Jue, Guardians of the deep fog: Failure-resilient dnn inference from edge to cloud, in: Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things, 2019, pp. 25–31.

[31] T. Campbell, J. P. How, Decentralized variational bayesian inference, CoRR (2014).

[32] J. Klein, M. Albardan, B. Guedj, O. Colot, Decentralized learning with budgeted network load using gaussian copulas and classifier ensembles, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 301–316.

[33] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28 (3) (2012) 583–592.

[34] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, J. Liu, Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent, Advances in Neural Information Processing Systems 30 (2017).

[35] Z. Jiang, A. Balu, C. Hegde, S. Sarkar, Collaborative deep learning in fixed topology networks (2017).

[36] X. Lian, W. Zhang, C. Zhang, J. Liu, Asynchronous decentralized parallel stochastic gradient descent (2017).

[37] H. Yu, S. Yang, S. Zhu, Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning (2018).

[38] S. Nikolaidis, I. Refanidis, Privacy preserving distributed training of neural networks, Neural Computing and Applications 32 (12 2020).

[39] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, 1st Edition, Chapman & Hall/CRC, 2012.

[40] N. Shlezinger, Y. C. Eldar, Deep task-based quantization (2019).

[41] M. Malka, E. Farhan, H. Morgenstern, N. Shlezinger, Decentralized low-latency collaborative inference via ensembles on the edge (06 2022).

[42] Y. Yu, J. Deng, Y. Tang, J. Liu, W. Chen, Decentralized ensemble learning based on sample exchange among multiple agents, in: Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure, Association for Computing Machinery, 2019, pp. 57–66.

[43] B. Jia, Y. Liang, Anti-d chain: A lightweight ddos attack detection scheme based on heterogeneous ensemble learning in blockchain, China Communications 17 (2020).

[44] A. B. Kurtulmus, K. Daniel, Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain, arXiv preprint arXiv:1802.10185 (2018).

[45] N. Deepa, Q.-V. Pham, D. C. Nguyen, S. Bhattacharya, B. Prabadevi, T. R. Gadekallu, P. K. R. Maddikunta, F. Fang, P. N. Pathirana, A survey on blockchain for big data: Approaches, opportunities, and future directions, Future Generation Computer Systems 131 (2022) 209–226.

[46] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, ACM Transactions on Programming Languages and Systems 4 (1982) 382–401.

[47] K. P. Kihlstrom, L. E. Moser, P. M. Melliar-Smith, The securering protocols for securing group communication, in: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences - Volume 3, HICSS '98, IEEE Computer Society, USA, 1998, p. 317.

[48] D. Malkhi, M. Reiter, Byzantine quorum systems, Distrib. Comput. (1998).

[49] V. Gramoli, From blockchain consensus back to byzantine consensus, Future Generation Computer Systems 107 (2020) 760–769.

[50] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D.-A. Seredinschi, O. Tamir, A. Tomescu, Sbft: a scalable and decentralized trust infrastructure (2018).

[51] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva: Speculative byzantine fault tolerance, ACM Trans. Comput. Syst. 27 (4) (2010).

[52] A. Bessani, J. Sousa, E. E. Alchieri, State machine replication for the masses with bft-smart, in: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014.

[53] Y. Fan, H. Wu, H.-Y. Paik, Dr-bft: A consensus algorithm for blockchain-based multi-layer data integrity framework in dynamic edge computing system, Future Generation Computer Systems 124 (2021) 33–48.

[54] M. J. Fischer, N. A. Lynch, M. S. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM (apr 1985).

[55] M. Castro, B. Liskov, Practical byzantine fault tolerance, in: Third Symposium on Operating Systems Design and Implementation (OSDI), USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS, New Orleans, Louisiana, 1999.

[56] F. Bravo-Marquez, S. Reeves, 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), IEEE, 2019, pp. 119–124.

[57] Y. Lan, Y. Liu, B. Li, Proof of learning (pole): Empowering machine learning with consensus building on blockchains (2020).

[58] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017).

[59] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009.

[60] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, Journal of Machine Learning Research - Proceedings Track 15 (2011) 215–223.
URL `http://cs.stanford.edu/~acoates/stl10`

[61] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Ng, Reading digits in natural images with unsupervised feature learning, NIPS (01 2011).
URL `http://ufldl.stanford.edu/housenumbers`

6. Appendix A: Tables

Table 6: Accuracy on 50% Total Cost Threshold.

Experiment	Dataset	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base	F-MNIST [58]	90.63	87.85	90.99	86.21	91.02	-	-
DA		91.69	90.50	91.64	88.84	91.88	-	-
DWA		91.54	90.53	91.63	88.68	91.84	-	-
Base	STL-10 [60]	73.30	68.70	66.84	71.43	70.65	-	-
DA		75.96	74.85	74.62	75.48	75.70	-	-
DWA		76.24	74.44	74.38	75.84	75.62	-	-
Base	SVHN [61]	90.93	91.36	88.51	93.11	90.64	89.54	94.20
DA		92.85	92.88	92.35	93.80	92.88	92.22	94.52
DWA		92.77	92.82	92.26	93.77	92.71	92.05	94.44
Base	Cifar-10 [59]	91.93	93.07	91.72	93.48	92.68	90.01	94.46
DA		94.49	94.48	94.13	94.39	94.46	93.63	95.07
DWA		94.09	94.36	93.93	94.09	94.32	93.33	94.92
Base	Cifar-100 [59]	62.79	63.41	65.98	70.39	64.39	59.02	69.77
DA		69.10	72.80	72.10	73.11	72.88	71.45	74.83
DWA		68.97	73.04	72.23	72.92	72.30	71.43	74.27

Table 7: Accuracy on 35% Total Cost Threshold.

Experiment	Dataset	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base	F-MNIST [58]	90.63	87.85	90.99	86.21	91.02	-	-
DA		90.97	88.51	91.18	86.65	91.38	-	-
DWA		90.67	88.52	91.20	86.46	91.37	-	-
Base	STL-10 [60]	73.30	68.70	66.84	71.43	70.65	-	-
DA		74.29	71.85	70.60	73.34	73.11	-	-
DWA		74.20	71.85	70.62	73.36	73.11	-	-
Base	SVHN [61]	90.93	91.36	88.51	93.11	90.64	89.54	94.20
DA		92.52	92.61	91.54	93.78	92.38	92.00	94.46
DWA		92.38	92.64	91.67	93.80	92.53	91.83	94.47
Base	Cifar-10 [59]	91.93	93.07	91.72	93.48	92.68	90.01	94.46
DA		93.89	94.28	93.48	94.24	93.86	92.99	94.99
DWA		93.82	94.19	93.32	94.20	93.91	92.88	95.02
Base	Cifar-100 [59]	62.79	63.41	65.98	70.39	64.39	59.02	69.77
DA		68.22	71.56	71.25	72.46	70.85	69.12	73.66
DWA		67.61	71.34	70.62	71.86	70.63	69.05	73.43

Table 8: Accuracy on 25% Total Cost Threshold.

Experiment	Dataset	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base	F-MNIST [58]	90.63	87.85	90.99	86.21	91.02	-	-
DA		90.59	87.41	90.88	85.11	90.83	-	-
DWA		90.59	87.41	90.88	85.11	90.83	-	-
Base	STL-10 [60]	73.30	68.70	66.84	71.43	70.65	-	-
DA		73.12	68.81	67.00	71.46	70.62	-	-
DWA		73.12	68.81	67.00	71.46	70.62	-	-
Base	SVHN [61]	90.93	91.36	88.51	93.11	90.64	89.54	94.20
DA		91.56	91.92	89.38	93.55	91.49	90.55	94.57
DWA		91.58	91.73	89.34	93.46	91.49	90.35	94.47
Base	Cifar-10 [59]	91.93	93.07	91.72	93.48	92.68	90.01	94.46
DA		92.35	93.08	92.11	93.73	93.05	90.62	94.42
DWA		92.28	93.00	92.20	93.55	92.79	90.60	94.41
Base	Cifar-100 [59]	62.79	63.41	65.98	70.39	64.39	59.02	69.77
DA		64.11	67.77	68.49	71.55	68.60	65.29	71.94
DWA		64.32	67.95	68.49	71.40	67.82	65.12	72.23

Table 9: Per-class Results on STL-10 based on F1-score

Agents	Method	Classes									
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
A1	Base	0.83	0.65	0.88	0.62	0.69	0.57	0.73	0.69	0.89	0.82
	DA	0.88	0.72	0.91	0.66	0.75	0.62	0.78	0.73	0.91	0.87
	DWA	0.88	0.72	0.91	0.66	0.75	0.62	0.78	0.73	0.91	0.87
A2	Base	0.84	0.65	0.87	0.46	0.62	0.47	0.66	0.63	0.86	0.78
	DA	0.88	0.73	0.91	0.64	0.73	0.58	0.80	0.71	0.90	0.86
	DWA	0.87	0.73	0.91	0.64	0.73	0.57	0.80	0.71	0.90	0.86
A3	Base	0.79	0.61	0.79	0.59	0.67	0.48	0.67	0.53	0.83	0.74
	DA	0.86	0.71	0.89	0.66	0.73	0.60	0.76	0.72	0.91	0.87
	DWA	0.86	0.71	0.89	0.66	0.72	0.59	0.76	0.72	0.91	0.87
A4	Base	0.84	0.68	0.86	0.57	0.66	0.54	0.71	0.67	0.85	0.82
	DA	0.87	0.75	0.90	0.62	0.75	0.61	0.78	0.74	0.90	0.87
	DWA	0.88	0.75	0.91	0.62	0.75	0.60	0.78	0.74	0.90	0.87
A5	Base	0.82	0.68	0.84	0.57	0.63	0.52	0.70	0.67	0.83	0.79
	DA	0.87	0.75	0.89	0.66	0.72	0.62	0.78	0.74	0.91	0.86
	DWA	0.87	0.76	0.89	0.66	0.72	0.62	0.78	0.74	0.90	0.86

Table 10: Results on Poisoned Experiments

Experiment	Dataset - Faulty Agents	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base Agents	SVHN - 3	90.83	42.60	88.45	32.97	46.76	89.49	94.18
DA (ours)		92.54	89.09	91.30	89.87	88.48	91.81	94.17
DWA (ours)		92.43	89.08	91.39	90.04	88.45	91.79	94.29
Base Agents	SVHN - 5	90.90	42.40	32.38	33.03	46.72	20.39	94.17
DA (ours)		88.88	89.83	91.52	91.25	88.42	91.78	94.31
DWA (ours)		88.86	89.87	91.31	91.10	88.49	91.66	94.336
Base Agents	D1 - 3	85.20	83.70	83.00	83.20	47.80	53.20	48.30
DA (ours)		86.30	84.70	84.80	84.10	86.20	85.70	85.90
DWA (ours)		86.30	84.40	84.60	83.40	87.50	86.40	86.60
Base Agents	D1 - 5	88.70	89.20	47.00	49.50	46.90	47.10	47.30
DA (ours)		90.20	90.50	65.70	66.70	66.40	69.20	63.60
DWA (ours)		89.40	89.80	66.30	68.70	67.00	70.00	64.80

Table 11: Comparison of Aggregation Methods in Poisoned Data

Experiments	Dataset - Faulty Agents	Centralized Voting Rules		QoI Consensus Protocol		
		Weight Average	Majority Voting	Class Rule	Weight Rule	Hybrid Rule
Base Agents		**93.21**	90.24	91.47	92.27	91.31
DA	SVHN - 3	-	-	92.55	93.04	92.66
DWA		-	-	92.66	93.18	92.64
Base Agents		**93.73**	55.14	59.48	72.83	59.64
DA	SVHN - 5	-	-	92.91	93.39	92.95
DWA		-	-	92.72	93.28	92.75
Base Agents		**90.80**	89.40	86.80	87.40	86.60
DA	D1 - 3	-	-	88.60	87.90	88.40
DWA		-	-	88.30	89.30	88.70
Base Agents		91.20	81.40	61.40	63.10	61.00
DA	D1 - 5	-	-	93.30	92.90	93.20
DWA		-	-	**93.40**	92.70	92.90

Table 12: Results on Confident Poisoned Agents

Experiment	Dataset - Faulty Agents	Accuracy (%)						
		A1	A2	A3	A4	A5	A6	A7
Base Agents		91.92	92.97	91.67	93.56	92.32	89.97	10.04
DA	Cifar-10 - 1	94.41	94.28	93.62	94.38	94.54	93.94	92.14
Base Agents		91.86	93.14	91.90	92.69	89.92	10.07	10.14
DA	Cifar-10 - 2	93.58	93.69	92.89	93.80	92.96	85.28	84.67
Base Agents		92.34	91.87	92.82	90.01	9.74	10.33	10.40
DA	Cifar-10 - 3	92.81	92.33	92.87	92.15	85.30	78.27	77.12
Base Agents		92.34	91.87	90.01	10.40	9.74	10.33	10.59
DA	Cifar-10 - 4	91.33	91.08	90.83	90.26	89.72	89.13	87.19
Base Agents		90.95	88.18	89.55	10.33	10.11	9.94	9.71
DWA	SVHN - 4	80.40	81.20	80.94	83.42	51.43	48.05	48.03
Base Agents		68.70	66.84	70.65	10.91	10.47	-	-
DWA	STL10-2	72.66	71.54	72.64	39.35	36.18	-	-
Base Agents		68.70	66.84	10.12	10.17	10.32	-	-
DWA	STL10-3	63.66	63.79	53.42	47.91	35.41	-	-

Table 13: Comparison of Aggregation Methods on Confident Poisoned Agents

Experiments	Dataset - Faulty Agents	Centralized Voting Rules		QoI Consensus Protocol		
		Weight Average	Majority Voting	Class Rule	Weight Rule	Hybrid Rule
Base Agents		94.78	94.57	94.46	94.37	94.41
DA	Cifar-10 - 1	-	-	**94.85**	94.80	94.82
Base Agents		93.77	94.12	93.74	93.58	93.64
DA	Cifar-10 - 2	-	-	94.08	**94.20**	94.06
Base Agents		92.61	93.29	92.43	92.37	92.52
DA	Cifar-10 - 3	-	-	92.36	**92.84**	92.33
Base Agents		90.91	89.69	74.66	74.86	73.59
DA	Cifar-10 - 4	-	-	90.29	**91.09**	90.30
Base Agents		55.68	88.25	90.39	90.49	**90.56**
DA	SVHN - 4	-	-	80.65	80.89	80.72
Base Agents		69.93	70.80	69.76	69.44	69.42
DA	STL10 - 2	-	-	73.62	**73.96**	73.60
Base Agents		62.50	56.46	45.39	42.48	42.43
DA	STL10 - 3	-	-	61.44	**63.50**	55.06