Tangle Ledger for Decentralized Learning

Robert Schmid, Bjarne Pfitzner, Jossekin Beilharz, Bert Arnrich and Andreas Polze
Hasso Plattner Institute, University of Potsdam, Germany
{firstname.lastname}@hpi.uni-potsdam.de

***Abstract*—Federated learning has the potential to make machine learning applicable to highly privacy-sensitive domains and distributed datasets. In some scenarios, however, a central server for aggregating the partial learning results is not available. In *fully decentralized learning*, a network of peer-to-peer nodes collaborates to form a consensus on a global model without a trusted aggregating party. Often, the network consists of Internet of Things (IoT) and Edge computing nodes.**

Previous approaches for decentralized learning map the gradient batching and averaging algorithm from traditional federated learning to blockchain architectures. In an open network of participating nodes, the threat of adversarial nodes introducing poisoned models into the network increases compared to a federated learning scenario which is controlled by a single authority. Hence, the decentralized architecture must additionally include a machine learning-aware fault tolerance mechanism to address the increased attack surface.

We propose a *tangle* architecture for decentralized learning, where the validity of model updates is checked as part of the basic consensus. We provide an experimental evaluation of the proposed architecture, showing that it performs well in both model convergence and model poisoning protection.

***Index Terms*—machine learning, consensus protocol, edge computing, information security, fault tolerance**

I. Introduction

In federated learning, the training task of a machine learning model is distributed to multiple participants with private data. Training is organized in rounds of communication and the aggregation of local updates is typically performed by a central, *trusted* authority [1].

In fully decentralized learning, this authority does not exist and a consensus on the aggregated model has to be continuously reached by the peers in the network. In an open network, malicious node behavior must be expected and hence protection mechanisms are required to prevent fabricated updates from poisoning the trained model which would otherwise reduce its performance on benign datasets.

Decentralized learning is anticipated to be mainly performed by IoT and Edge devices, following the assumption that for privacy reasons, raw data must not be aggregated across users or organizations and hence must be analyzed as close to its origin as possible.

We propose to adopt the tangle architecture of the IOTA distributed ledger [2] which is designed for low power compute devices. In contrast to traditional blockchains, the approved transactions of a tangle are represented by a directed acyclic graph (DAG) rather than a linear chain of blocks. Each new transaction that is inserted into the tangle approves two previous transactions, thereby contributing to a global consensus. As part of this approval step, we incorporate an algorithm that verifies the proposed model weights. An overview of this approach is presented in Fig. 1. We argue that a tangle architecture is well suited to build a robust decentralized learning system and to the best of our knowledge, this paper is the first to propose learning in such a way.

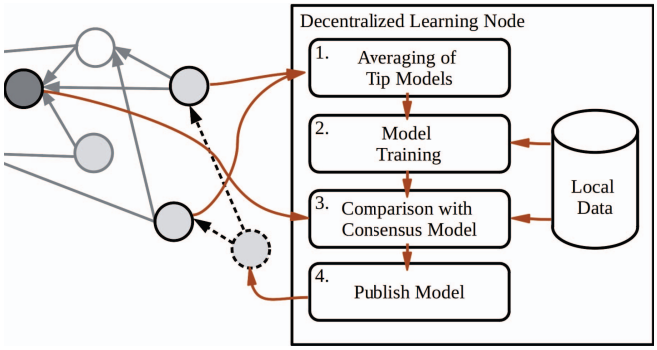


Fig. 1: Using a tangle as consensus mechanism, federated learning can be decentralized. A participating node averages the models published in two tips, trains this model with local data and publishes the new model if it is better than the current consensus model.

The remainder of this paper is structured as follows. Section II introduces the relevant related work. Our concept of a tangle for decentralized learning is described in Section III before Section IV introduces our prototypical implementation, which is used to evaluate our approach. The results of this evaluation are presented in Section V. Finally, Section VI concludes this paper.

II. Related Work

This section provides an overview of the areas of related work by introducing the basic concepts of federated learning, previous approaches for decentralized learning using distributed ledger architectures and tangle-based ledgers as a specific instance thereof.

A. Federated Learning

The training of machine learning models relies on the availability of large datasets, and to some extent, the model performance scales with the amount of data used for training. Nowadays, large datasets are technically available, but they often exist in a distributed way. That means, data owners are scattered and may not have the trust or option to share their data, for example due to privacy regulations such as the *General Data Protection Regulation (GDPR)* [3].

978-1-7281-7445-7/20/$31.00 ©2020 IEEE
DOI 10.1109/IPDPSW50202.2020.00144

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.

An approach to work with distributed datasets in a privacy-preserving way is *federated learning* [1]. It describes an algorithm for training machine learning models on massively distributed data. A federated learning system consists of a server and multiple clients (up to millions) who jointly train a machine learning model. First, the machine learning model is defined in terms of its type (e.g. convolutional neural network (CNN)) and model hyperparameters like the number of layers and their activation functions. The model is then distributed to each participant, followed by several global epochs in which the server instructs a fraction of available clients to perform a few epochs of local optimization and report the resulting model parameters (in reality just the parameter difference) back to the server. There, the parameter updates are aggregated, weighted by the number of local training samples used to find a particular update. After a predefined number of epochs, or sufficient model convergence, the training process concludes and the model can be used for prediction. Alternatively, it can be continuously updated when more and more data becomes available for participants.

There are some challenges when using federated learning, which are imposed by the characteristics of federated datasets:

- **Massively distributed**: The data can be distributed across millions of clients, so there are challenges with client availability or communication efficiency to consider.
- **Unbalanced**: While some clients possess large datasets themselves, others might only have a few samples, also effecting their weight updates.
- **Non-IID**: Data collected by different clients cannot be expected to be independent and identically distributed (*non-IID*), which results in a potentially high variance in weight updates reaching the server.

Another challenge for federated learning architectures as previously described is its susceptibility to different types of *poisoning attacks*, in which adversaries can effectively manipulate the predictions on one or many of the participating client devices. Emerging from data poisoning of centralized machine learning models, where an adversary injects malicious data into the training set such that retraining the model makes it less effective, model poisoning is an attack considered in a federated learning scenario. It can be a result of data poisoning of a client, or one or multiple clients being adversarial towards the network and sending poisonous model updates.

To make the distributed learning system robust against data poisoning attacks, a byzantine fault tolerance (BFT) mechanism must be implemented as part of the architecture. Byzantine faults refer to the broadest class of faults in system components, where a participating node may affect the stability of the entire system in any possible way.

Previous approaches to ensure byzantine fault tolerance in federated learning assume a batch of incoming gradients and a trusted authority to verify the gradients. The median-based Krum algorithm [4] provably converges given a maximum of f byzantine workers in a distributed stochastic gradient descent (SGD) process. However, its support for non-IID datasets, which are common in federated learning, is limited since it tends to discard outlier gradient updates even if they are not to be considered malicious.

There are still many open questions regarding defense against poisoning attacks, including how to prevent targeted attacks that only affect a minority of participating clients.

Further security concerns in federated learning related to user privacy include the ability to reconstruct private data from published model updates [5] and to detect updates by the same participant [6].

B. Decentralized Learning using Blockchain Architectures

Open networks for decentralized learning have an increased attack surface for poisoning attacks since there is no authority that restricts participation in the network. As in the centralized setting, a mechanism must ensure that updates received from the clients verifiably improve the quality of the overall model, while privacy of the training data is maintained [7].

As a first step towards decentralized learning, concepts from blockchain technologies were adopted for training a global machine learning model [5], [8]–[12]. In these architectures, a linear chain of blocks is created through block mining. The current network consensus is defined by the state contained in the latest transaction that has a sufficiently high probability of being part of the longest chain of blocks.

Fundamentally, the federated averaging algorithm can be mapped onto the blockchain by first letting client nodes submit their gradient updates into the ledger until a fixed number of updates are collected, in which case an aggregation step is triggered and a transaction with the aggregated results is published into the network. Additionally, the median-based BFT protocols from federated averaging can be applied to the decentralized architecture by removing outliers from the batch of gradients prior to the averaging step.

The block mining process in these architectures can be implemented by a Proof-of-Work algorithm [9], which is not well suited for constrained IoT and Edge devices. Other approaches make use of Proof-of-Stake in combination with verifiable random functions (VRFs) to build mining committees from a random subset of nodes [8], [13]. The mining committee is authorized to propose the next block in the chain within a given timeframe by using a voting scheme internally. However, this approach requires high availability of the participating nodes in order to conduct the synchronized block selection process, which usually cannot be guaranteed in IoT and Edge scenarios.

Some of the blockchain architectures assume that the ledger is short-lived until a certain training task has been accomplished. After the training process, nodes can be compensated according to their contributions [9]. For our approach, we envision a long-lived, evolving learning network, that incentivizes participants to continuously contribute by providing a global model in return which over time adapts to shifts in the underlying data distribution.

One previous work by Lu et al. [14] tries to address privacy and security concerns for federated learning in mobile networks. An asynchronous federated learning scheme is proposed, where all nodes publish model updates and the

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.

corresponding mean absolute error in a random sub-gossip scheme. Interesting for our work is the use of a directed acyclic trust graph by Lu et al. to mitigate the risk of malicious nodes publishing wrong mean absolute errors, which would lead to biased averaging in their system.

C. Tangle Consensus

Recent distributed ledger technologies have replaced the linear blockchain with a DAG in order to accommodate higher throughput of the distributed consensus protocol. The tangle is a DAG that was introduced for the cryptocurrency IOTA [2] and is now also used in various other protocols such as Byteball, Avalanche or SPECTRE [15]–[17].

Like a blockchain, the tangle can create statistical consensus on the ordering of events between many distributed nodes. But unlike a blockchain, a tangle does not rely on one single chain being the single source of truth, the idea is rather to reach consensus on a partially ordered set of transactions. [2]

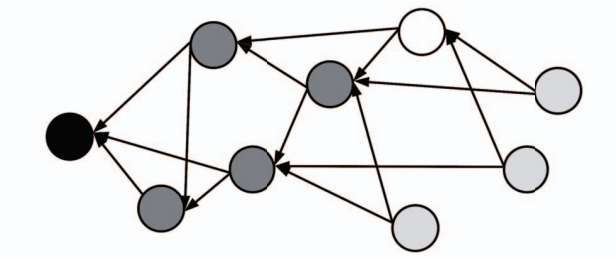


Fig. 2: A tangle is a DAG where vertices represent transactions and the edges denote approval of another transaction. The genesis-transaction is black, transactions that are approved by all tips (and as such part of the consensus) are dark gray and tips are light gray. The white transaction is neither a tip nor approved by all tips.

In the tangle, each published transaction approves two other transactions, as illustrated by Fig. 2. In the resulting DAG, the vertices are transactions and the directed edges are approvals of other transactions. A transaction that is not approved by any other transaction is called a tip. For a new transaction, two (not necessarily distinct) tips are chosen to be approved according to a tip selection algorithm. This new transaction is said to approve these two tips *directly*, while it also *indirectly* approves all transactions that are approved by the two tips. In other words, the approval-relationship is transitive.

While many different tip-selection algorithms are discussed, this paper will focus on the widespread algorithm of a weighted random walk from the genesis transaction (thus moving opposite the direction of approvals) where the weights are the number of approvers for a given transaction. [2], [18]–[20]

The distributed consensus is reached on those transactions that have a high probability of being directly or indirectly approved by all future transactions in the ledger. This is typically approximated by sampling the current tips in the network and evaluating their direct and indirect approvals of other transactions.

Additionally, a proof-of-work mechanism is used to prevent adversaries from flooding the network with crafted transactions that could subvert the consensus in a Sybil attack.

III. CONCEPT

The tangle architecture proposed in this paper adopts the basic working principle described in the previous section to implement a cross-device decentralized learning network. As such, we assume a network population with hundreds or thousands of devices that are not typically available at the same time to perform training; furthermore, limitations in compute and storage resources, as well as network bandwidth, are to be expected [21]. The training dataset is horizontally partitioned, i.e. devices have different sets of non-IID training and validation examples that include a common set of features.

A direct but inefficient implementation approach would be to map the gradient batching and aggregation steps of federated averaging to the distributed tangle ledger, similar to the examples introduced in Section II-B. To prevent adversarial participants from manipulating the global model, median-based outlier detection can then be applied within the batch of proposed gradients [4], [8].

However, instead of artificially introducing batches to support poisoning defenses, we chose to leverage the asynchronous consensus mechanism unique to the tangle to not only record the incoming stream of updates but also to directly perform model aggregation and validation steps. In particular, we make the following changes to distributed tangle ledgers:

Firstly, each transaction in the tangle consists of a full set of parameters for a shared machine learning model, rather than recording transfers of assets or cryptocurrency. The model parameters represent a snapshot of the current best training results on a given node.

Secondly, the legitimacy of a transaction correlates with the resulting predictive performance of the contained model parameters with regards to the global data distribution. However, since each node only has access to a non-IID subset of the data, it is not able to make a binary decision about the legitimacy of a given transaction as in distributed ledgers with well-defined soundness criteria.

Instead, well-behaved nodes incorporate other nodes' results into the training process and only publish the resulting parameters if a prediction improvement is achieved on local validation data. Thus, the chosen parent transactions are implicitly endorsed by the node once it publishes its training results.

The following sections introduce the core algorithms on each participating node in more detail.

A. Choosing a Reference Transaction from the Tangle

For making predictions on unlabeled data, a participating node needs to obtain the *current best* model weights from the tangle, as defined by the network consensus. This process is not specific to the learning tangle per se, however, we highlight some implementation choices in the following.

In order to choose a reference transaction, the node initially computes *confidence* and *rating* for every transaction in the graph.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.

The confidence of a transaction is determined by running the tip selection multiple times, thereby counting how often a given transaction is hit during the random walk. The final confidence is a number between zero and one, which is obtained by dividing the hit count by the number of sampling rounds.

In addition to the transaction confidence, the rating of a given transaction is defined by the number of other transactions that it directly or indirectly approves. The IOTA tangle allows different transactions to contribute to ratings in different degrees, depending on the hardness of the provided proof-of-work. This implies that some transactions in IOTA are more important to the consensus than others. In the prototypical learning tangle implementation, however, all transactions contribute equally to the rating.

Algorithm 1 shows how the transaction confidences and ratings are used to derive the current consensus from a tangle graph.

Algorithm 1 Choosing reference weights from tangle graph

ChooseReferenceWeights(G)**:**
$C \leftarrow$ ComputeConfidences(G)
$R \leftarrow$ ComputeRatings(G)
PriorityQueue Q
for each transaction t in G **do**
Q.insert$(t, C(t) * R(t))$
return Q.top()

To achieve a smoothing effect on the retrieved weights, a possible variation is to perform an averaging step on the top n weights, as determined by the algorithm. This would lead to fewer fluctuations in validation accuracy especially in the early stages of the network since the individual weight sets are more heavily biased towards the local dataset of the publishing node. As model convergence is reached in later training stages, this effect diminishes since there are only minor differences in the published weights.

B. Model Training and Transaction Validation

To ensure the integrity of the network consensus, a continuous stream of benign transactions is required. Each node can take part in the consensus multiple times. Furthermore, depending on the locally available compute resources, it is not required that a node takes part in training only when receiving new labeled training data.

Provided that access to the network preserves user anonymity, the identity of the user is not revealed by participating in training multiple times. Section III-D will discuss this in more detail.

Algorithm 2 shows how a participating node emits updates into the network based on averaged weights and local training results. This algorithm implicitly validates the parent transactions by publishing training results that incorporate the parent weights only if the resulting model parameters perform better on the local test data set than the reference model which is obtained from the current tangle consensus.

Algorithm 2 Basic Model Training and Parameter Validation

Node executes:
reconstruct tangle G
$w_r \leftarrow$ ChooseReferenceWeights(G)
$(w_1, w_2) \leftarrow$ TipSelection(G)
$w_{avg} \leftarrow \frac{1}{2}w_1 + \frac{1}{2}w_2$
$w_{new} \leftarrow$ Train$(w_{avg}, epochs, lr)$
if ValidationLoss$(w_{new}) <$ ValidationLoss(w_r) **then**
Broadcast(w_{new})

This basic implementation is not yet robust against model poisoning attacks: The probability of discarding a training result and not publishing a new transaction for well-behaved nodes is proportional to the probability of selecting a malicious tip. This has a self-reinforcing effect since fewer benign transactions are emitted into the network and the probability to select malicious tips increases. Quickly, the tangle consensus is taken over by adversarial nodes. Section III-E provides an extension to the algorithm which addresses this issue.

C. Convergence and Efficiency

Because new transactions are only published if they improve on the reference model as evaluated on the local data, it is expected that the model converges on a good global solution. In Section V, we evaluate the convergence rate on two benchmarking datasets in comparison to federated averaging. Furthermore, the combination of averaging and training ensures that the number of tips in the network remains constant given a fixed rate of incoming updates. This can be seen as the distributed and asynchronous equivalent of the aggregation step in centralized federated averaging.

In the decentralized training process, each node performs the training on a typically unique set of base model parameters which are determined by the randomly selected parent tips of the tangle transaction. Therefore, it is not possible to average the model gradients as in federated averaging. One possible approach to compensate different base model parameters is asynchronous stochastic gradient descent [22].

For the initial version of the tangle architecture, averaging is performed using the model parameters, which yields equivalent aggregation results. In contrast to federated learning, published models are equally weighted in the averaging step. In a real-world implementation, the communication costs of sending the full model parameters are likely to be higher than exchanging gradients, since compression is more effective on gradients [23]. However, an in-depth analysis of the communication costs in the network is not part of this paper.

In decentralized learning, nodes only have access to a small set of local data that exhibit node-specific characteristics. Consequently, they are interested in actively participating in the tangle network to build up a global model that incorporates knowledge from other nodes in order to prevent over-fitting on the local data. On the other hand, this means that no single node can judge the quality of any proposed update in the network with regards to the global data distribution. Hence,

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.

it depends on other nodes to verify its proposed transactions to ensure that the prediction accuracy on unseen local data improves.

D. Privacy

One benefit of fully decentralized approaches like the tangle, compared to federated learning, is the lack of a trusted central authority, which minimizes the risk associated with having a single party directly communicate with all participants for training. A malicious server in federated learning can abuse his power over parameter aggregation and distribution to violate the clients' privacy. One possibility for the adversary is a *reconstruction attack* targeting sensitive local data [5]. By knowing which participant sent which updates over time, a malicious server can collect sufficient information about local datasets to rebuild a close representation. The server can also direct an attack at a single client by sending him a specific model such that the resulting parameter updates reveal more information to the adversary.

To prevent this type of privacy breach, federated learning research includes algorithms which use one or multiple middle-men between clients and server [24], [25], however, the requirement of trust between client and facilitator still stands. Another possible defense is the introduction of *differential privacy* into the system, which essentially adds noise to client updates, scaled to their specific datasets [26]. The noise hides the contribution of specific data points to the update and reduces the risk of identifying or reconstructing the data [27], [28]. While encryption schemes can also provide provable security against a malicious party, encryption and decryption algorithms, as well as computing with ciphertext greatly increase the time required for computation, making it a poor choice for a tangle application that is targeted at devices with limited computational power.

Our proposed tangle architecture greatly reduces the risk for a reconstruction attack, since every participant publishes their model parameters anonymously. Additionally, the randomized parent tip selection for choosing the base model for training prevents targeted attacks as explained above.

Another type of attack against federated learning is a *linkability attack*, where the attacker can identify model updates sent by the same person (also via multiple different devices) through the update values alone and thus increase his knowledge about client updates [6]. This attack relies on updates by the same client in different rounds being similar to each other. We leave it up to future work to investigate the relatedness of transactions published by the same participant of our system, but the mitigation options mentioned in [6] in the form of differential privacy or data augmentation still apply.

E. Robustness and Security

A decentralized learning system needs to be able to withstand attacks by malicious nodes that publish fabricated model parameters, so-called poisoning attacks. There are two main goals the adversary can try to achieve in this situation: an *indiscriminate attack* aims at reducing the overall model performance, whereas *targeted attacks* (also called *backdoor attacks*) have a specific misclassification of individual samples or whole classes in mind, which could later be exploited by an evasion attack [29]. We anticipate and evaluate one example for each type of poisoning attack.

For the former type, the indiscriminate attack, malicious clients publish transactions consisting of random noise which reset the weights and hinder model convergence. An example of a targeted attack is a *label-flipping attack* [30], where malicious clients possess a local training dataset entirely consisting of mislabeled samples, according to the targeted misclassification. If the goal is to misclassify samples from class 0 as class 6, then the dataset will have samples of class 0, which are labeled as 6s. Incorporating this poisoned data in the dataset, the training process pushes the model towards the targeted misclassification.

While the comparison with a consensus model described in Section III-B ensures that obvious poisonous models are not approved by honest nodes, it leads to honest nodes having a low probability of publishing well-performing models and thus the tangle being overtaken by the malicious nodes.

To increase the robustness against poisoning attacks we alter the way tips are selected as the basis of model training. We propose to run the tip selection algorithm described above multiple times and validate the models of each tip on the local data. Then, the two best performing models are averaged and their transactions approved. As shown in Section V-B, this approach provides robustness against poisoning attacks while maintaining model convergence and consensus.

IV. Implementation

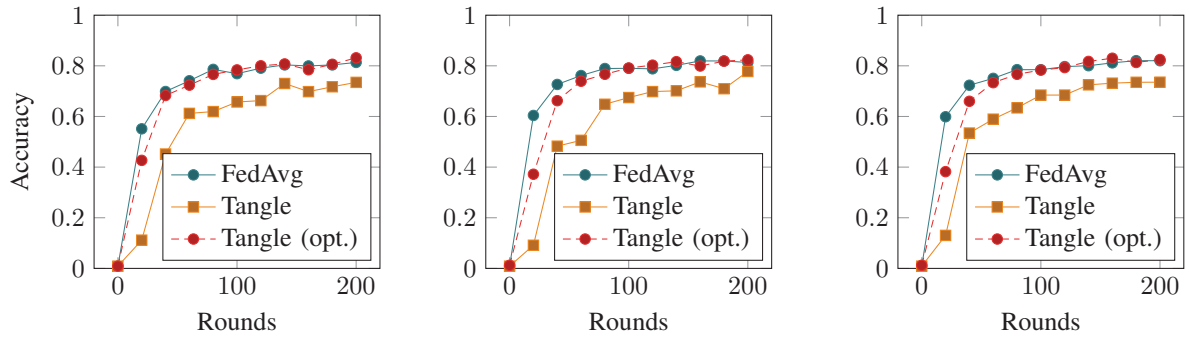
Our tangle implementation is meant for benchmarking the architecture for different machine learning tasks and to compare the efficiency of the tangle with a centralized federated averaging algorithm. It extends the LEAF framework [31], which provides reference models for a number of federated datasets for analysis of the federated averaging algorithm. The tangle implementation is available on GitHub[1].

The evaluation prototype does not implement certain aspects of tangle ledgers that ensure scalability and resource efficiency in a real-world implementation. For instance, the tip selection algorithm always starts at the genesis transaction whereas the original tangle authors propose to start tip selection from *particles* within a window of transactions that are *reasonably deep* within the tangle [2].

Furthermore, the defense against Sybil attacks, where the network is flooded with illegitimate transactions, is not addressed by the prototype. In future work, publishing transactions may be restricted to nodes who provide a proof-of-work, as in IOTA, or a proof-of-stake that potentially measures the previous contributions to the consensus model. For the scope of this prototype, the focus is to evaluate the efficiency of fully decentralized aggregation, training and its resistance to model

[1] https://github.com/osmhpi/learning-tangle

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.



(a) 10 nodes per round (b) 35 nodes per round (1% of all nodes) (c) 50 nodes per round

Fig. 3: Mean prediction accuracy of federated averaging (baseline) and tangle learning processes on the FEMNIST dataset. The first tangle trial was run with 2 selected tips and a single model chosen as consensus model. After hyperparameter optimization, nodes selected 3 tips and used a reference model averaged from 10 models. After 80 rounds of training, an accuracy of more than 60% is achieved by the non-optimized tangle in all three configurations and after 200 rounds, the prediction accuracy is within 0.1 of the baseline. Hyperparameter optimization lead to a performance of the tangle comparable to FedAvg.

TABLE I: Characteristics of the benchmarking datasets and training parameters.

	FEMNIST	Shakespeare
Train/Test Split	0.8	0.9
Labels	62	80
Users	3500	1058
Minimum Samples Per User	0	64
Model Type	CNN	Stacked LSTM
Learning Rate	0.06	0.8
Local Epochs	1	1

poisoning attacks with an assumed percentage of adversarial network participants.

Therefore, we assume a given number of active nodes to actively participate in training at a time. Although it wouldn't be necessary for the learning tangle algorithm, training is organized in rounds, where the published transactions from a given round are only visible to the nodes participating in the next round. This simplifies comparisons with the round-based federated averaging process.

V. EVALUATION

This section evaluates the learning tangle based on its convergence properties and robustness against poisoning attacks.

A. Neural Network Convergence

To evaluate the efficiency of the architecture, we compare the convergence rate of the consensus model in the tangle network with a federated averaging benchmark. Evaluation was conducted using two datasets that were pre-processed to show non-IID data characteristics. The federated EMNIST (FEMNIST) dataset is used for handwritten digit and letter recognition, where the data is partitioned by the authors of the character samples. A CNN is trained during the learning process. In contrast, on the *Shakespeare* dataset, a recurrent neural network (RNN) architecture makes next-character predictions for the texts of different characters in Shakespeare's plays. Table I shows the parameters used during preprocessing and training of the datasets and models provided by LEAF.

Figure 3 displays the validation accuracy for the FEMNIST dataset for federated averaging and the learning tangle using 10, 35 and 50 nodes per round, respectively. Validation is performed every 20 training rounds using the test datasets of a random selection of 10% of all nodes. Note that the validation on the tangle is performed on the current consensus weights, i.e. one transaction (or an average of 10 transactions) that resides several rounds deep in the tangle. Thus, the results of the last few training rounds are usually not incorporated in the validation.

The benchmarking results show that the convergence characteristics of the distributed averaging process in the tangle yield slightly inferior but still acceptable results compared to the batch-oriented federated averaging process. Moreover, as in federated averaging, the convergence rate in the first 200 rounds is mostly independent of the number of nodes that are active at a given time.

In order to observe the tangle behavior on a different type of model architecture, we train a global model on the Shakespeare dataset with 10 active nodes per round (Fig. 4). The results show a similar offset to the federated averaging baseline as the FEMNIST results: After an initial bootstrapping phase of the network, the tangle accuracy catches up until the final gap after 200 training rounds amounts to 5pp (0.55 vs. 0.50).

Furthermore, we investigated the effect of tangle learning hyperparameters on the convergence speed of the consensus model. Table II shows the number of rounds required to reach 70% accuracy of the reference model on the validation set. We found that increasing the number of tips selected by the nodes to three reduced the number of rounds required by an average of 12. Another observation is that choosing more published models as the current reference model of the tangle improves the convergence by up to 51 rounds (see Table II, row 3). Finally, increasing the sample size of tips being considered

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.

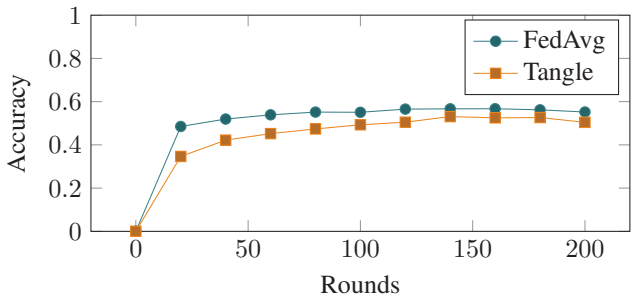


Fig. 4: Mean accuracy per round of federated averaging (baseline) and tangle learning (without hyperparameter optimization) processes on the Shakespeare dataset, 10 active nodes per round (1% of all nodes).

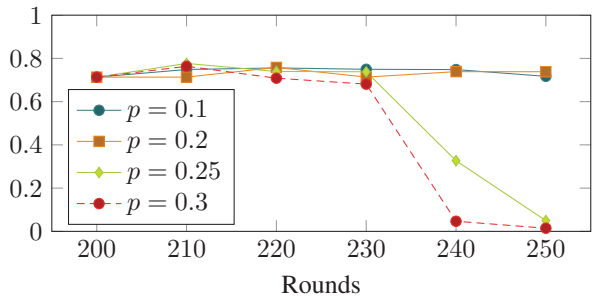


Fig. 5: Development of model accuracy when injecting transactions with random model weights into the tangle network, starting from round 200.

TABLE II: Effects of hyperparameters on the convergence speed of tangle learning, measured on the FEMNIST dataset.

# tips (n)	sample size	# Transactions chosen as reference model			
		1	2	10	50
2	n	89	73	59	57
	$2n$	77	73	59	53
	$5n$	108	77	57	57
3	n	77	53	**44**	45
	$2n$	85	57	47	**44**
	$5n$	86	64	45	**44**

by a node for aggregation showed a negative impact on the convergence speed. More experimentation is required to explore the hyperparameter space for tangle learning.

In conclusion, the tangle implementation provides acceptable convergence rates compared to the federated averaging baseline for both CNNs and RNNs.

B. Model Poisoning Attack

For investigating the effects of model poisoning attacks, a fraction $p \in \{0.1, 0.2, 0.3\}$ of nodes are declared as malicious. After 200 rounds of benign training on the FEMNIST dataset, the adversarial nodes generate poisoning transactions to add to the tangle whenever they are chosen for a training round.

The robustness of the learning tangle depends on a careful parameterization of the nodes. This includes the *randomness factor* of the tip selection algorithm [32] as well as the number of sampling rounds for selecting tips for training and during consensus selection. For the benchmarking measurements, we set the number of sampling rounds for establishing the consensus and for selecting the parent tips for training equal to the number of active nodes per round (35). To better inform the choice of these parameters, a structured analysis of the effects of the tangle parameters on the robustness should be conducted in the future.

Fig. 5 shows the effects of an indiscriminate, random poisoning attack, where the adversarial nodes simply submit model parameters generated by a standard normal distribution. The results indicate that the tangle can sustain 20% of adversarial nodes without an impact on the consensus prediction accuracy. However, 25% and 30% of adversarial nodes are able to take over the consensus within 50 or 40 rounds, respectively.

The tangle's robustness against a targeted label-flipping attack is demonstrated in Fig. 6. Malicious nodes adapt their local dataset to only include samples exhibiting the targeted misclassification (in this case samples from class 3 labeled as class 8). While we can identify a success of the attack with 20% and more adversarial nodes, the tangle manages to recover to a more accurate state after 50 rounds. In the case of $p = 0.1$, the label-flipping attack fails.

VI. CONCLUSION AND OUTLOOK

This paper discusses the fundamental suitability of a tangle architecture for decentralized learning. The initial evaluation of the approach showed high prediction accuracies for two non-IID datasets. Moreover, tangle learning demonstrates a model-agnostic resistance against poisoning attacks.

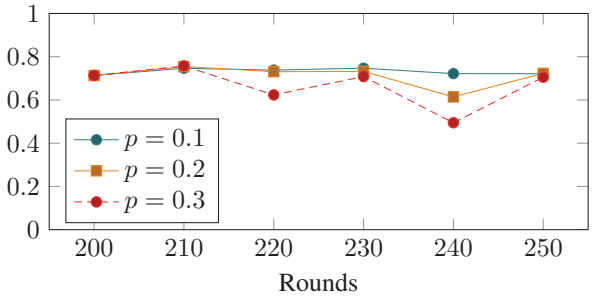
Possible next steps are to investigate how the tangle structure could support creating individualized models for clusters of similar nodes. One approach would be to evaluate the model on local data during the tip selection algorithm, introducing model performance as a bias in the weighted random walk. This could lead to clusters of federated nodes with similar data working on separate sub-tangles and would thus discard the idea of creating consensus between all nodes. The implications of this on robustness need to be evaluated.

Another interesting research direction is the security against different attacks on machine learning systems. The results of this paper have to be extended with more in-depth experimentation about different classes of poisoning attacks.

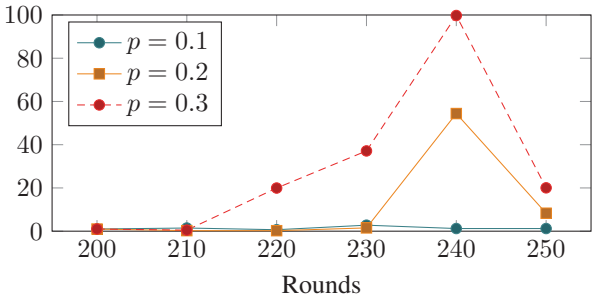
Moreover, the conceptual prototype could be translated into a distributed implementation which can be benchmarked in a simulation environment, thereby considering faults introduced by real-world network conditions. In addition, such a simulation allows for an evaluation of the resource efficiency of the approach.

Finally, it should be investigated whether it is possible to adapt the training goal in the tangle to perform meta-learning such that the resulting model can quickly be trained to perform optimally on a given nodes' dataset.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.



(a) Development of model accuracy



(b) Average target validation misclassification percentage

Fig. 6: Effects of a label-flipping attack targeting the misclassification $3 \rightarrow 8$ on tangle, pre-trained for 200 rounds.

ACKNOWLEDGMENT

This research was partly funded by the Federal Ministry of Education and Research of Germany in the framework of KI-LAB-ITSE (project number 01IS19066).

REFERENCES

[1] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv e-prints*, Feb. 2016. arXiv: 1602.05629 [cs.LG].

[2] Serguei Popov, *The tangle*, 2018. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf.

[3] European Union. (2016). "General data protection regulation (GDPR)," [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30*, I. Guyon *et al.*, Eds., Curran Associates, Inc., 2017, pp. 119–129. [Online]. Available: http://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent.pdf.

[5] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," *CoRR*, 2018. arXiv: 1812.00535.

[6] T. Orekondy, S. J. Oh, B. Schiele, and M. Fritz, "Understanding and controlling user linkability in decentralized learning," *CoRR*, 2018. arXiv: 1805.05838.

[7] Y. Dong, J. Cheng, M. J. Hossain, and V. C. M. Leung, "Secure distributed on-device learning networks with byzantine adversaries," *IEEE Network*, pp. 1–8, 2019. DOI: 10.1109/mnet.2019.1900025.

[8] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A ledger for private and secure peer-to-peer machine learning," *arXiv preprint arXiv:1811.09904*, 2018.

[9] X. Chen, J. Ji, C. Luo, W. Liao, and 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, Dec. 2018. DOI: 10.1109/bigdata.2018.8622598.

[10] H. Kim, J. Park, M. Bennis, and S. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, pp. 1–1, 2019, ISSN: 2373-7891. DOI: 10.1109/LCOMM.2019.2921755.

[11] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sep. 2019, pp. 1–4. DOI: 10.23919/APNOMS.2019.8893114.

[12] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019, ISSN: 2160-9209. DOI: 10.1109/TDSC.2019.2952332.

[13] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand," in *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17*, ACM Press, 2017. DOI: 10.1145/3132747.3132757.

[14] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020, ISSN: 1941-0050. DOI: 10.1109/TII.2019.2942179.

[15] A. Churyumov, *Byteball: A decentralized system for storage and transfer of value*, 2016. [Online]. Available: https://byteball.org/Byteball.pdf.

[16] T. Rocket, *Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies*, 2018.

[17] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[18] B. Kusmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the Tangle for Uniform Random and Random Walk Tip Selection," *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 228–236, Jul. 2019. DOI: 10.1109/Blockchain.2019.00037. arXiv: 2001.07734.

[19] B. Kusmierz, *The first glance at the simulation of the Tangle: Discrete model*, 2017.

[20] S. Popov, O. Saa, and P. Finardi, "Equilibria in the Tangle," *Computers & Industrial Engineering*, vol. 136, pp. 160–172, 2019.

[21] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *arXiv e-prints*, Dec. 2019. arXiv: 1912.04977 [cs.LG].

[22] S. Zheng *et al.*, "Asynchronous Stochastic Gradient Descent with Delay Compensation," *arXiv e-prints*, Sep. 2016. arXiv: 1609.08326 [cs.LG].

[23] J. Konečný, H. Brendan McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," *arXiv e-prints*, Oct. 2016. arXiv: 1610.05492 [cs.LG].

[24] C. Fung, J. Koerner, S. Grant, and I. Beschastnikh, "Dancing in the dark: Private multi-party machine learning in an untrusted setting," *CoRR*, 2018. arXiv: 1811.09712.

[25] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1175–1191, ISBN: 9781450349468. DOI: 10.1145/3133956.3133982.

[26] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318, ISBN: 9781450341394. DOI: 10.1145/2976749.2978318.

[27] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private language models without losing accuracy," *CoRR*, 2017. arXiv: 1710.06963.

[28] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, 2017. arXiv: 1712.07557.

[29] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," *arXiv e-prints*, Jul. 2018. arXiv: 1807.00459 [cs.CR].

[30] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," *arXiv e-prints*, Jun. 2012. arXiv: 1206.6389 [cs.LG].

[31] S. Caldas *et al.*, "LEAF: A Benchmark for Federated Settings," *arXiv e-prints*, Dec. 2018. arXiv: 1812.01097 [cs.LG].

[32] Alon Gal, *Alpha: Playing with randomness*, 2018. [Online]. Available: https://blog.iota.org/alpha-d176d7601f1c.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:32:30 UTC from IEEE Xplore. Restrictions apply.