A Proof of Useful Work for Artificial Intelligence on Blockchain

ANDREI LIHU
JINCHENG DU
IGOR BARJAKTAREVIC
PATRICK GERZANICS
MARK HARVILLA
**Motivation**

Blockchain is secure & Blockchain is decentralised & PoW mining is wasting energy because Hashing is meaningless

What we want: Replace proof of work (PoW) with proof of useful work (PoUW).

How to achieve this: Provide machine learning (ML) services secured by the blockchain.
**Environment**

The environment is the PAI blockchain, a hybrid POW/POS blockchain.

<table>
<thead>
<tr>
<th>Clients</th>
<th>Miners</th>
<th>Supervisors</th>
<th>Evaluators</th>
<th>Verifiers</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Nodes interested in training their models using the PAI blockchain.</td>
<td>- Nodes that train a local model, but also mine with special nonces obtained from executing a ML iteration.</td>
<td>- Nodes that record all messages exchanged during training, but also detect malicious behaviour.</td>
<td>- Nodes that test the final models from miners and decide upon splitting the rewards.</td>
<td>- Nodes that verify if a mined block is the result of honest and useful work.</td>
<td>- Nodes that do not perform any of the functions described so far, just regular transactions, are called peers.</td>
</tr>
</tbody>
</table>
ML Service

Client submits a ML task along with his/her preferences

Worker nodes are matched to task based on preferences

Miners train a ML model and mine with nonces built from by-products of training

Worker nodes are rewarded with the client’s fee

Client dataset is split into:

- Training
- Validation
- Test

Training dataset is split into mini-batches:

- Mini-batch 1
- ... 
- Mini-batch n

A mini-batch is processed in each iteration by a miner. An epoch is a full training cycle, i.e. all mini-batches from the training dataset are processed.
Except regular peers, all nodes must stake PAI Coins to participate. Stakes are returned once nodes performed honestly all their duties, otherwise they are confiscated.

If stakes expire, they are returned in full.
Tickets

Tickets = Stakes = locked money that are returned along with extra fees once the participants properly carry out their duties. They can be used once they become "live" (after a maturation period).

Interested nodes issue BUY TICKETS transactions containing the task preferences and the desired role: miner, supervisor, evaluator or verifier.

Clients issue a PAY FOR TASK transaction that includes the task description, along with a fee.

Evaluators issue a CHARGE FOR TASK transaction to pay the worker nodes after they evaluate their final ML models.
Workflow
A Typical Task Roadmap

- **REGISTRATION**
  - client submits a task
  - client dataset is prepared for ML training

- **INITIALISATION**
  - worker nodes are matched to the task
  - the expected order of messages is established

- **TRAINING**
  - training starts
  - miners improve their local models by sharing and receiving weight updates
  - supervisors look for malicious behaviour and record the messages
  - miners also mine with nonces

- **FINALISATION**
  - evaluators select the best final model
  - evaluators reward the worker nodes
Registration

1. **Task Submission**
   - The client first submits the task as a transaction in the mempool (PAY_FOR_TASK)

2. **Task Block Mining**
   - A miner includes the task transaction in a mined block => task definition block

3. **Participants Selection**
   - Miners and supervisors are chosen to perform and oversee the training
Initialisation

1. **DATASET PREPARATION**
   - the task definition block hash is used as a random seed to split the client dataset into training, validation and test. The training dataset is split into mini-batches.

2. **ESTABLISH THE ORDER OF MESSAGES**
   - we establish in advance what is the order of mini-batches processing at each epoch for each miner.

3. **KEY EXCHANGE**
   - the miners and supervisors that participate in the training will exchange their public keys, in order to verify if the signed sent messages are genuine.
ML training is distributed, asynchronous and decentralised.

Each miner processes a separate allocated mini-batch, updates his/her local model, then sends his/her weight updates to other miners and supervisors.

Every other miner receives the peers' updates and applies them to his local model, while supervisors record the messages.
Miners update their local model using Stochastic Gradient Descent (SGD) as the optimiser of choice.
Deep Neural Networks

The model trained by the miners is a deep neural network, for example:

+ A Multilayer Perceptron (MLP)
+ A Convolutional Neural Network (CNN)
+ A Recurrent Neural Network (RNN)
Main Loop Procedure

- **DO BACKPROPAGATION**
  - **CALCULATE RESIDUAL GRADIENT**
  - **CALCULATE LOCAL WEIGHTS AND BUILD MESSAGE MAP**
  - **APPLY LOCAL WEIGHTS TO LOCAL MODEL**
  - **SEND RESULTS (MESSAGE MAP) TO PEERS, THEN MINE WITH NONCES**

**Receive and Apply Peers’ Weight Updates to Local Model**

**Load Minibatch**

**Dead-reckoning scheme:**
A miner will only send the indices in the local model (the message map) whose values are above/under a specified threshold value (±T).
The residual gradient accumulates values under the threshold, that are later added to the model.
Finalisation

The evaluators are selected using a matching algorithm similar to the one used to select miners and supervisors.

- Training finished & evaluators are called in
- Client provides test dataset to evaluators & miners provide their final models
- Evaluators test the final models and produce reports
- Worker nodes are rewarded by evaluators (CHARGE_FOR_TASK)
- Evaluators must produce identical conclusions that are announced to the blockchain using the CHARGE_FOR_TASK transaction. If less than 2/3 of the evaluators produce identical reports, then another round of evaluators is drawn until a 2/3 consensus is reached. All evaluators that were right from any round are paid in the end.
Message History
Recording Messages

Supervisors record the messages into chunks per time slots. Lost messages are recovered using a gossip protocol for synchronisation.

A chosen slot master publishes the hash of the messages in the chunk on the blockchain (as a MESSAGE_HISTORY transaction).
At each time slot supervisors vote using a preferential voting system that will yield the order of the messages in the current slot.

A supervisor's stake is not freed until the training process is finished and he/she cast all his/her votes.

While the order of messages is pre-established at initialisation, the supervisors will record their actual order and their contents.

Schulze voting
Mining
Nonces

How to obtain the nonce: double hash the local model at the end of the iteration and the local weight updates.

Miner is allowed to try several nonces based on mini-batch size and model complexity: nonce + interval
At every iteration $i$, a miner commits to mine using a zero-nonce block after $k$ iterations.
PoUW Blocks

The miner fills in the previous zero-nonce block with the nonces obtained from the ML iteration.

If the hash is lucky, he/she stores:
- the initial model state
- the mini-batch
- the gradient residual
- the peer updates
- the relevant message history chunks

The mined block contains the hash of the message and the hash of the message history chunk as additional fields.
Fill-in Tasks

To ensure continuous mining on the PAI blockchain, if there are no active tasks, the network acts as a client and starts some predefined tasks.

Miners do not get a client's fee, but they get the block reward.

Supervisors, verifiers and evaluators get paid a small fee from the block's reward.
Verification
Delegated Verification

1. Based on the mined block hash, 10 verifiers are called in using the tickets system.

2. A verifier obtains the lucky iteration's inputs from the lucky miner.

3. All verifiers check the message history, the block commitment and re-run the iteration to verify if the nonce was calculated correctly.

4. Verifiers will post their encrypted digests (the calculated nonce precursor) to the mempool, that they will reveal when the miner decides to post a COLLECT_VERIFICATIONS transaction.

Only a few delegated verifiers will pursue all the verification steps because it's computationally expensive to rerun iterations.
Verification by Peers

1. Regular peers can perform all the verification steps for themselves if they want to do so.

2. Most peers will only check if the nonce precursor reported by the verifiers hashes to the value of the block's nonce.

Through verification we obtain the proof of useful work (PoUW) from the lucky miner.
Adversarial Models
Malicious Clients & Miners

Supervisors run heuristics in real-time to detect malicious behaviour described by the adversarial models.
Other malicious actors

* details for each adversarial model are provided in the whitepaper
Implementation

ML code implemented in Python with Gluon/MXNet.

Blockchain code implemented in C++.
Conclusion

We believe that this is a system that would democratise artificial intelligence using the power of blockchain.

We presented a POW for a decentralised machine learning system on blockchain.

It can be easily extended to other AI algorithms.

We believe that this is a system that would democratise artificial intelligence using the power of blockchain.
Thank you

For more feel free to contact me at:
mark@oben.com