Decentralized Marketplace - Problem solving AI - Machine Learning Smart Contracts

Robert Michit, Mourad Redjah, Youssef Azzouzi, Leopold Lessasy, Olivier Gilles

Table of Contents

Abstract

Using blockchain technology, it is possible to create contracts that offer a reward in exchange for a trained machine learning model for a particular database. This would allow users to train machine learning models and be rewarded via blockchain.

The smart contract will use the blockchain to automatically validate the solution, so there would be no debate as to whether the solution was correct or not. Users who submit solutions will have no risk of not being paid for their work. Contracts can be easily created by anyone with a database.

This creates a market where parties capable of solving machine learning problems can directly monetize their skills and where any organization or agent who has a problem to solve with AI can solicit solutions from developers around the world. This will encourage the creation of better learning models and make AI more accessible to companies.

A consequence of the creation of this market is that there will be a well-defined price for GPU training for machine learning models. Cryptocurrency mining also uses GPUs in many cases. We can imagine a world where, at any time miners can choose to run their equipment to work on the most profitable workload: cryptocurrency mining or machine learning training.

1. Introduction

The problems solving and decision making model built by the Cyphai team comes from a wide range of experiences, formal and informal discussion and interactions between all Cyphai consultants and their counterparts (ranging from managers, students and parents looking for a solution to their day life or job activities).

The Cyphai model is a model from the research of a decision-making and problems solving specialist - Robert Michit - and his teams.

From the question for a manager weighing the opportunity to invest a large amount of capital in a billion dollar business project, to the definition of a portfolio investment allocation for an Asset management team, or not more easier to decide where to live, in which boroughs of the city and in what kind of housing, making a decision is roughly the same. The stake can be highly different but the rules of the decision are the same.

That’s roughly the building origin of the Cyphai Model from the beginning, more than 30 years ago.

The Cyphai model can be applied to any kind of organisation management (private and public affairs, small or large companies) but also to private and daily life problems (teaching and education, games addiction and of course relationships).

But right decision making is complex and requires in any case a lot of knowledge and information. Blockchain technology and Artificial Intelligence can really help in the matter and make the difference. They can really propel the model to a new dimension by increasing its reasoning capabilities and allow its use and practice by everyone.

We want ,as a team, provide to everyone to deal with it, in the most intuitive way , wether you are a computer scientist or consultant but also to anyone else interested in decision making.

The Cyphai development team has chosen Ethereum as the first platform to showcase its work and do all the necessary tests on the blockchain and smart contracts.

1.1. Bitcoin, Ethereum and cryptocurrencies

Bitcoin was first introduced in 2009 to create a decentralized method of storing and transferring funds from one account to another. He enforced the property using public key cryptography. Funds are stored in different addresses, and anyone with an address's private key could transfer funds from this account. To create such a system in a decentralized way, it was necessary to innovate on the way to reach a consensus between the participants, which was resolved using a blockchain. This created an ecosystem that allowed fast and reliable transactions between unreliable users.

Bitcoin has implemented a scripting language for simple tasks. This language was not designed to be complete. Over time, people wanted to implement more complex programming tasks on blockchains. Ethereum has introduced a comprehensive language to support a wider range of applications. This language was designed to use the decentralized nature of the blockchain. It is essentially an application layer above the Ethereum blockchain.

By having a more powerful and complete programming language, it has become possible to create new types of applications on top of the Ethereum blockchain: from escsyn systems, strike new parts, decentralized companies, etc. The Ethereum white paper talks about creating synchronized decesyn-col marketplaces, but focuses on things like identities and reputations to facilitate these transactions. In this market, especially for machine learning models, trust is a required feature. This approach is clearly different from the trustless trading system proposed in this article.

1.2. Breakthrough in machine learning

In 2012 Alex Krizhevsky, Ilya Sutskever and Geoff Hinton were able to form a deep neural network for classifying images using GPUs. Their submission for the Large Scale Visual Recognition Challenge (LSVRC) halved the best error rate at the time. The ability of GPUs to perform thousands of matrix operations in parallel was the breakthrough needed to form deep neural networks.

In ML, a variety of models and approaches are used to attack different types of problems. Such an approach is called a neural network (NN). Neural networks are made up of nodes, bias and weighted edges and can represent virtually all functions.

The construction of a new machine learning model takes place in two stages. The first step is training, which takes an input data set and adjusts the weights of the model to increase the accuracy of the model. The second step is the test, which uses an independent data set to test the accuracy of the model formed. This second step is necessary to validate the model and avoid a problem called over-fitting. An overadjusted model is very good for a particular database, but is bad for generalizing for the given problem.

Once it has been formed, an ML model can be used to perform tasks on new data, such as prediction, classification and clustering.

There is a huge demand for machine learning models, and companies that can access good machine learning models are benefiting from improved efficiency and new capabilities. Given that there is a high demand for this type of technology and a limited supply of talent, it makes sense to create a decentralized market place for machine learning models. Since machine learning is purely software and training requires no interaction with any physical system, the use of blockchain for coordination between users and the use of cryptocurrency for payment is a natural choice.

Ethereum white paper talks about on-chain decentralized marketplaces, using the identity and reputation system. It does not go into detail regarding the implementation, but mentions a marketplace being built on top of concepts like identities and reputations.

Here we introduce a new protocol on top of the Ethereum blockchain, where identities and reputations are not required to create a marketplace transaction. This new protocol establishes a marketplace for exchanging machine learning models in an automated and anonymous manner for participants.

ML models are verified and evaluated by running them in a forward pass manner on the Ethereum Virtual Machine. Participants using this protocol are not required to trust each other. Trust is unnecessary because the protocol enforces transactions using cryptographic verification.

2. Cyphai Protocol

To demonstrate a transaction, a simple Neural Network and forward pass capability is implemented to showcase an example use case.

Structure:

1: User Bob submits a database, an evaluation function and a reward amount to the Ethereum smart contract. The evaluation function takes into account a machine learning model and generates a score indicating the quality of this model. The reward is a reward in cryptocurrencies (e.g. ether).

2: Users download the database submitted by user Bob and work independently to form a machine learning model that can represent this data. When a Bob user successfully trains a model, he submits his solution to the blockchain.

3: At some point, the blockchain will evaluate the models submitted by users using the evaluation function and select a contest winner.

Note: Some extra steps are necessary in each of the phases in order to ensure trust and fairness of the competition. Details to follow in the “protocol” section.

The Cyphai protocol is proposed to allow users to solicit machine learning models for a reward in a decentralized manner. The protocol has 5 steps to ensure a contract is executed successfully.

2.1. Definitions

A user is defined by anyone who can interact with Ethereum smart contracts.

A CYPHC (Cyphai contract) is an Ethereum contract which implements the Cyphai protocol.

An organizer is defined by a user who creates the Cyphai contract.

A bidder is a user who submits solutions to the Cyphai contract for a reward.

A period is a unit of time which consists of the number of blocks extracted.

A data point is made up of inputs and predictions.

A data group is a matrix made up of data points.

A contract wallet address is an escrow account that holds the amount of the reward until the contract is finalized.

The contract uses the following protocol:

2.2 Contract Initialization

An organizer has a problem they wish to solve, database for testing and training, and an Ethereum wallet.

Contract creation: The organizer creates a contract with the following elements:

A machine learning model definition, a neural network is used in the demo

init1() , init2() and init3() functions.
Function get_training_index() to receive training indexes.
Function get_testing_index() to receive training indexes.
Function init3() for revealing the training database to all users.
Function submit_model() for submitting solutions to the contract.
Function get_submission_id() for getting submission id.
Function reveal_test_data() for revealing the testing database to all users.
Function get_prediction() for running a forward pass for a given model.
Function evaluate_model() for evaluating a single model.
Function cancel_contract() for canceling the contract before revealing the training database.
Function finalize_contract() for paying out to the best submission, or back to the organizer if best model does not fulfill evaluation criteria.

Init Step 1:

1.Reward deposited in the contract wallet address to payout winning solution.
2.Maximum lengths of the submission, evaluation and test reveal periods defined by the height of the blocks.
3.Hashed data groups.

Init Step 2: The organizer triggers the randomization function to generate the indexes for the training and testing data groups. These indexes are selected by using block hash numbers as the seed for randomization.

Init Step 3: The organizer reveals the training data groups and nonces by sending them to the contract via the init3() function. The data is cryptographically verified by the previously provided hash values for the data groups.

At this point, the Cyphai contract is initialized and the training database is public. The submitters can download the training database to start training their models. After successfully training they can submit their solutions.

2.3. Solution Submission

During this phase, any user can submit a potential solution. This is the only period during which submissions will be accepted.

Submitter(s) invoke the submit_model() function, providing a solution with the following elements:

The solution weights and biases.
The model definition.
The payment address for payout.

2.4. Evaluation

The evaluation stage can be initiated in one of two ways:

The organizer calls the function reveal_test_data() that reveals the testing database.
In this case, the testing database will be used for evaluation

The test reveal period has passed, but the organizer has not yet called reveal_test_data()
Since the testing database is unavailable, the training database will be used for evaluation

Once the evaluation begins, submissions will no longer be accepted and submitters may now begin evaluating their models:

Submitters call the evaluate_model() function with their submission id.
If the model passes the evaluation function, and is better than the best model submitted so far or is the first model ever evaluated it is marked as the best model.

2.5. Post-Evaluation

After the evaluation period ends, any user can call the finalize_contract() function to payout the reward to the best model submitter.
If best model does not exist, reward is paid back to the organizer.

3. Incentives and threat models

The key to any market is user trust. We need to design a system where no user can cheat or gain an advantage over another user. We need to look at this from the perspective of all parties involved: the organizer, the bidders and even the miners of the Ethereum blockchain.

We anticipate several risks that need to be mitigated:

3.1. Over-adjustment by the applicant

If the applicant has access to the test database, he can adapt his model to this database. The bidder can "cheat" by training hard on the test set, the downside being that the model will not apply to more general problems.

The solution to this problem is to keep the test database secret until the end of the submission period. In our contract, we ask the organizer to come back later and reveal the test set.

3.2. Handling of the test set by the organizer

If the training and test set does not come from the same input distribution, the organizer can deceive a modeller outside his costs. The organizer can provide false test data and collect the model weights without having to pay the bidders.

The solution to this problem is to make sure that the organizer cannot choose the training and test databases. The organizer submits the hashes of the data groups that represent the entire database. Cyphai's contract randomly selects a subset of this data that the organizer is required to reveal later. Since hashes are deterministic, the validity of the revealed data groups can be checked. This prevents the organizer from manipulating the training / test database.

3.3. Organizer does not reveal test database

For some reason, the organizer cannot reveal the test database. This would preclude calling the evaluation function and paying bidders for their work in these situations.

To alleviate this problem, we are giving the organizer some time to reveal the test database. If the organizer does not do so, the tenderers can always trigger the evaluation function on the training database.

Evaluating models with the training database is less than ideal, but it is a last resort necessary for bidders to be rewarded for their work in these rare circumstances.

3.4. Too many submissions

In the Ethereum network all transactions are executed by miners. With every transaction, a fee is required to compensate the miner for executing and saving it on the blockchain. This fee is called gas. Since each block is mined every 12 seconds on average, it is not practical to accept transactions that would take a long time. Accepting these types of long transactions would increase the risk of missing a block, and not being out-mined by other miners. To mitigate this issue, miners have their own self-imposed gas limits. If a transaction uses more than the gas limit, it will fail regardless of the amount being paid.

Originally the evaluation function was called by the organizer or submitter if the testing database was not revealed. This evaluation function would evaluate all submitted models. If the Cyphai contract had too many models to evaluate, the evaluation function was at the risk of running out of gas due to existing gas limits. At the time writing this the average gas limit was around ~8 million. This would create a vulnerability where the function caller can run out of gas pretty quick, or can get rejected immediately due to the large gas size not getting accepted at any nodes. This would mean that the contract would never be finalized.

To mitigate this problem, we allow users to call the evaluation function on any submitted model. This prevents the too-many-submissions problem since the evaluation function only runs on a single model at a time. This also reduces the amount of total computation needed to evaluate the models. This would create a situation where only the best model submitters would be incentivized to call the evaluation function on their own models.

Since evaluation can also be done offline, submitters can evaluate each model locally to determine if their model is indeed the best one or is in the top-n. It might make sense to call the evaluation function on your model if you’re in the top-n, since there might be a chance that the best model

submitter might not call the evaluation function on their model. In an ideal situation this should not happen, but if it does, the next best model submitter can also claim the prize if it ever happens.

3.5. Rainbow table attacks on hashed data groups

Initially the sha-3-keccak was only hashing the data group without a nonce. This made this hashing method susceptible to rainbow table attacks from the submitters.

One possible solution to this problem was to implement bcrypt, and use a salted hash instead. Password hashing functions do work, but they're computationally more expensive. Instead we decided to add a nonce to the end of the data group and hash it using sha3-keccak instead.

This method prevents rainbow table attacks, and keeps the gas cost at a lower price point.

There are probably other solutions to this problem, including like scaling data points and adding noise. This makes sense since real life data is rarely clean, and mostly noisy.

3.6. Block hash manipulation by miners

Initially we were only using the last block hash as a source of entropy for our randomization function. The problem with this approach is a miner can influence the resulting hash for a given block. This creates an opportunity for the organizer to cheat as mentioned in section (4.2). If they're a miner, they can deploy their contract right after mining a block. This doesn't guarantee that their contract will be initialized, but it gives them some influence over the selection process for the training and testing database.

Being a miner does not give you full control over how the training selection process works. But it would show you which data groups are going to be selected. Due to this, the organizer can decide to not mine a specific block hash that could result in undesirable training indexes.

To minimize the influence of the attacker, we require them to call init2() function within 5 blocks of calling init1() . If they fail to do so, the contract will get cancelled.

To read more about the implementation of hashing, and the probability of getting favorable training indexes, please refer to section 5.2.

3.7. Distributed reward system abuse

Depending on a selection criteria, the reward can be claimed by the first submitter who fulfills the evaluation criteria, the best model submitter or both. The reward could even be distributed among top solutions to incentivize more participation. The organizer has full control over picking the selection criteria for their Cyphai contract.

A distributed reward system would be similar to Ethereum's proportionate mining reward for stale blocks. A stale block is a solution to a block that is propagated to the network too late. These blocks are still rewarded to make sure miners still get paid for the work they contribute, and to keep the network stable. Only up to 7 stale blocks are rewarded. Stale blocks can happen due to many reasons like outdated mining software, and bad network connectivity.

A distributed reward system may de-incentivize submitters since a malicious submitter might resubmit the same solution with minimum changes to solution to steal the work of the original submitter. Due to this reason, it makes sense to only payout to the best submitter. If there are two

submissions with the same solution, and this solution is the best one, the first submitter will get paid out instead if they're both evaluated. This is to prevent malicious submitters from re-submitting the same solution and calling the evaluation function before the original submitter.

The ideal situation would be where these models are trained in pools, and the reward would be distributed to members among a pool when a solution is found. This would make collaboration and distributed rewarding possible.

4. Implementation

4.1. Hashing the database

The organizer breaks down the whole database into several data groups. A nonce is a randomly generated number that is only intended to be used once. Difference nonces are generated for each data group. Each individual nonce is pushed to the end of the corresponding data group.

The organizer hashes these data groups by passing them to a hashing function. For the Cyphai protocol sha3-keccak is chosen as the hashing function for creating hashed data groups.

Ideally each data group is made up of 5 data points. If there's 100 data points, that would make a total of 20 hashed data groups. This later allows the partitioning of the database into training and testing databases.

4.2. Determining the Training database

During Initialization step 2, the organizer calls the init2() function. This function uses the previously mined blocks hash numbers as a seed for randomization. This function randomly selects the training and testing groups. The default ratio is 80% for training and 20% for testing, but this can be changed in the contract.

The randomization function calculates the sha3-keccak of the previous block's hash number, and returns the hexdigest. The modulo of the hexdigest is used to randomly select an index. After selecting an index, we decrement the modulo and repeat the previous step until all training data group indexes are selected. The initial modulo is the total number of data groups.

Algorithm (in solidity) for randomly selecting hashes:

```
function randomly_select_index(uint[] array) private {
  uint t_index = 0;
  uint array_length = array.length;
  uint block_i = 0;
  // Randomly select training indexes
  while(t_index < training_partition.length) {
    uint random_index = uint(sha256(block.blockhash(block.number-block_i))) % array_length;
    training_partition[t_index] = array[random_index];
    array[random_index] = array[array_length-1];
    array_length--;
    block_i++;
    t_index++;
  }
  t_index = 0;
```

```
  while(t_index < testing_partition.length) {
    testing_partition[t_index] = array[array_length-1];
    array_length--;
    t_index++;
  }
}
```

Let’s call the number of data groups we have G
. This makes the probability of getting a favorable index 1G. This index is selected by getting the modulo G

of a given block hash.

Let’s call the training percentage as TP
. The number of training indexes are G∗ TP

.

With every selected index, we decrease the modulo G
by 1 for selecting the next index. We iterate to G∗ (1−TP)

until we have selected all training indexes.

Therefore, the probability of getting a unique sequence of training indexes are
G∏n=G∗ (1−TP)+11n

A malicious organizer would not be only interested in a sequence of training indexes. This is because any permutation of those indexes would yield the same training database. The permutation for the training indexes are: (G∗ TP)!

.

This makes the probability of getting ideal training indexes: (G∗ TP)!∗ G∏n=G∗ (1−TP)+11n

This probability can also be re-written as: G∏n=G∗ (1−TP)+1G−n+1n

Let’s call the block limit L. After calling init1() , the organizer has to call init2() within L blocks to limit the influence they have over selecting the training indexes. In the Cyphai contract, a default of 5 blocks is used, which should correspond to 1 minute on average.

Ideally speaking for the organizer, let’s assume that the contract gets deployed immediately. This gives the organizer L chances to call the init2() randomization function after init1() .

Hence, this makes the probability of getting ideal training indexes within L
blocks P=L∗ G∏n=G∗ (1−TP)+1G−n+1n

For a training partition of 80% and a block limit of 5, here are the chances of getting an ideal group of training indexes for the following number of data groups:

# of Data Groups G	Ideal probability P
5	100%
10	~11.11%

15 ~1.0989%
20 ~0.103199%
25 ~0.00941088%
30 ~0.00084207%

As seen in the table, the higher the number of data groups are, the less likely a malicious organizer can affect the outcome.

4.3. Forward Pass

A forward pass function is implemented for the given machine learning model definition. For the scope of this paper, we only included a simple neural network definition and a forward pass function. The general idea is to demonstrate that it should be possible to implement some if not most ML functions and models.

4.4. Evaluating the database

Depending on the type of problem we're trying to solve, we can use metrics like accuracy, recall, precision, F1 score, etc. to evaluate the success of the given model.

Based on the scope and requirements, any one of these metrics can be selected. An ideal scoring metric for every classification problem doesn't exist. Every ML problem should be evaluated within its own scope since it can have sensitivity towards different metrics. (eg. less tolerance towards false-negatives in cancer prediction)

For demonstration purposes, we chose a simple accuracy implementation for evaluating submitted solutions.

4.5. Math Functions

The Ethereum Virtual Machine (EVM) doesn't have a complete math library. It also does not support floating point numbers. The absence of these functions require to implement Machine Learning (ML) models using integer programming, fixed-point float point numbers, and linear activation functions.

Certain activation functions used in ML models such as sigmoid require functions such as exp() . These functions also require some floating point constants such as Euler's number e. Both of these math features are not implemented in EVM yet. Implementing them in a contract would significantly increase the gas cost. This makes non-linear type functions less desirable to use in evaluating ML models. For this reason, we've used ReLU instead of Sigmoid.

While implementing fixed float point numbers in solidity we noticed that shift functions were implemented with exponentiation functions, therefore they're actually more expensive than using division.

In solidity the right shift x>>y
is equivalent to x2y

. For division, this requires a lot more operations for a simple division. Due to this fact, we've extensively used division instead of shifting in fixed float point calculations.

Overall, since the EVM is low level language, and does not have a complete math library, most things needs to be implemented from scratch to get ML models working.

4.6. Working Around Stack Too Deep Errors

Solidity only allows to use around 16 local variables in a function. This includes function parameters and the return variable too. Due to this restriction, complex functions such as forward_pass() needed to be divided into several functions.

This required to use minimum number of local variables in functions. Due to this limit, in many places variables were accessed directly instead of be referred by more descriptive variables. This tradeoff was partially mitigated by adding more explanatory comments.

5. Miscellanea And Concerns

5.1. Complete anonymization of the database and Model Weights

Currently neither the database or model weights and biases are anonymized. Any user can access the submitted models on Cyphai contracts.

A possible way to solve this issue might be through the use of homomorphic encryption. One thing that stands out with homomorphic encryption is the use of real numbers. Since integers are real numbers, this would work with our implementation in solidity. This implementation also uses integers to compute the forward pass.

With homomorphic encryption, it's possible to encrypt the database for the sake of privacy. It's also possible to encrypt the weights of a model that is trained on an un-encrypted database. One thing to keep in mind is that even though homomorphic encryption can provide anonymity, it'll make the contract more expensive to execute. For the scope of this paper, homomorphic encryption is not included in the protocol.

5.2. Storage gas costs and alternatives

It is known that storing databases in the contract and validating them would require significant amounts of gas.

Here's a solidity contract that writes 1 KB of data to the Ethereum blockchain on solidity version 0.4.19:

```
pragma solidity ^0.4.19;

contract StorageTest {
  byte[1024] data;
  function store() public {
    for (uint i = 0; i < 1024; i++) {
        data[i] = 'A';
    }
  }
}
```

After creating the contract, the transaction cost for storing 1 KB of data is about 6068352 gas. This is currently below the gas limit of 8 million (as of Jan 2018). This means that it is possible to write large databases in increments instead of a single transaction.

MNIST is a popular handwritten digits database used for benchmarking optical character recognition algorithms. The whole database is around 11594722 bytes. The average gas price is around 4 gwei (as of Jan 2018). This makes the total cost of writing the MNIST database around 275 Ethereum. As of writing, Ethereum is worth around $1,100 (Jan 2018). This would make the total cost about $302,500.

The gas price is mostly consistent across different solidity versions for the same code instructions. The total price can change over time, since it's determined by the gas and Ethereum price.

A big portion of ML problems definitely have larger databases than MNIST. Storing these databases in the blockchain isn't a sustainable solution in majority of cases.

Alternatives like IPFS and swarm might be used for storing the databases elsewhere. These alternatives try to tackle the problem of storing large files and databases on the blockchain, while keeping the price at a reasonable level.

5.3. Model execution timeout

Like with all other Ethereum contracts, there's always a gas limit for running Cyphai contracts. Some complex models may not run due to high gas costs. Miners might reject these models if it requires gas more than the limit imposed by the miner. Accepting to execute such a model would make mining a block more likely to become stale.

Even though the gas limit goes up on average, the limit itself prevents running a set of deep neural networks. This means that running certain models may not be possible until the gas limit goes up, or EVM gets further optimized.

5.4. Re-implementation of Offline Machine Learning Libraries

Since solidity only works with integers, most popular machine learning libraries won't work out-of-the-box with these contracts. These libraries might need to be adapted to work with integers instead of floating points.

This might be possible if the activation functions are linear and the weights and biases are integers.

5.5. Cancelling a contract

Organizers are allowed to cancel a contract and take back their reward if they haven't revealed their training database yet.

6. Other Considerations and Ideas

6.1. GPU miner arbitrage and mining pools

A consequence of creating this market is that there will be a well-defined price of GPU training for machine learning models. Crypto-currency mining also uses GPUs in many cases. We can envision a world where at any given moment, miners can choose to direct their hardware to work on whichever workload is more profitable: cryptocurrency mining, or machine learning training.

GPU miners who choose to join these mining pools will be joining these pools that will be managed by Data Scientists. If that pool solves the contract, ideally the reward will be divided between the Data Scientists who manage the pool and the miners who provide the hardware.

Additionally the way to verify proof-of-work is a lot more simpler than in traditional cryptocurrency mining. In a pool a submitted solution and its accuracy can be considered the unit of work, which would make it easier to assess the amount of work done by each individual miner. Each unit of work can be easily assessed via the evaluation function.

6.2. Self-improving AI systems

Furthermore, we envision a world where intelligent systems can use these contracts to improve their own capabilities, by requesting training on new problem domains.

Since these contracts use cryptocurrencies to reward participants, all interactions with these contracts are digital, hence ideal for self-improving AI systems.

6.3. Raising Money for Computation Power for Medical Research

Cyphai contracts can also be used for medical research purposes. This has the advantage of being able to directly donate to the contract wallet address. This removes the requirement for a middleman or trusting a 3rd party. As the reward gets bigger, it’ll attract more participants to submit solutions to the given problem.

For example, a Cyphai contract could be created for a protein-folding problem, that might help with cancer research. This would create a new way to crowdsource funds for medical research.

6.4. Potential improvements

It’s worth mentioning that the Cyphai protocol has a lot of room for improvement. As mentioned before, introducing homomorphic encryption would definitely be a useful feature. Better designed Cyphai contracts could significantly reduce gas costs. The solidity language could introduce new features that would make Cyphai contracts faster and cheaper. The ever increasing gas limit will make running certain machine learning models possible that weren’t before. Improvements in Machine Learning such as using 8-bit integers will help further reduce gas costs for Cyphai contracts.

And possibly, a new language specifically designed for matrix multiplication for the Ethereum blockchain can significantly increase performance for Cyphai contracts.

7. Cyphai Token

Cyphai issues a fixed number of divisible Tokens (CYPH Tokens) which are used on the network as digital currency for all transactions, as well as for network operations.

The CYPH Token will be issued on Ethereum for the sale of Tokens, which allows the holder to generate Cyphai Tokens on the public test network for the purposes of development/testing and proof of work or staking. When the main network is released, the ERC-20 Tokens issued will be convertible to Cyphai Tokens.

Role of the Cyphai token

The Cyphai Token is the main method of exchanging securities on the Cyphai network. It is necessary for all network exchanges, and as a mechanism to provide added value to those doing work on the network.

Ability to access and develop blockchain-based AI / ML algorithms. The Cyphai Token allows you to develop and access a wide range of machine learning and artificial intelligence tasks available in the blockchain.

The operating costs in Cyphai are decoupled from the Cyphai Token in the same way as those for "gas" on the Ethereum network.

The ERC-20 Token

When generating CYPHAI Tokens, ERC-20 Cyphai Tokens will be issued.

Cyphai tokens can be used for many things, including, but not limited to:

Licensees can develop machine learning, artificial intelligence, applications services and have them run as part of proof of work and/or staking operations. Between these developers and node operators, these applications and services can be delivered to those who want them and the value exchanged accordingly.

Cyphai ERC-20 Token plays the key role in accessing the existing utility value from the test network, as well as the component facilitating the ability to develop and access the future utility value.

Token economy

The total number of Tokens generated is estimated at 1,000,000,000. No other Tokens will be created.

Company: 20%

Founders: 20%

Token Sale: (Private Seed and Sale & public sale) 20%

Future releases: 15%

Mining or staking operations: 15%

Advisors: 10%

Acquisition of Tokens:

Different acquisition periods apply to some of the Tokens issued.

8. Decision making

Making decisions - The different approaches to decision-making

https://medium.com/the-versatile-designer/decision-making-strategies-for-ux-product-designers-30040ab6e127

To explain the different approaches to decision-making, there's no better than begin with what our Cyphai consultant and their business counterparts, the middle management, face in their day to day decision making.

The role of a middle manager is to produce a certain result, negotiated with more or less success with his higher level management, as a chief of a profit center (business unit), and responsible of a team.

The scope of his job is related to 5 major activities (recruit-train-animate-control-report) that are translated into a series of daily tasks. The central question here is to determine what are the daily micros - decisions regarding the tasks to be accomplished in order to achieve the expected result of the manager.

In the long run, most organizations have developed a great number of tools for management evaluation in the quantitative and qualitative field objectives.

At the level of the quantitative objectives, it is essentially a question of measuring the managerial performance on the 5 major activity axes

- Recruitment
- Training
- Animation
- Control
- Reporting

In terms of qualitative objectives, the middle manager needs to meet pre-defined managerial priorities put forward by the higher level of management as part of its corporate strategy. The 4 priorities that our Cyphai consultant and their business counterparts have to tackle these days are always the same regardless of the business entity and sector.

- Deliver digital transition.
- acquire and develop business market share.
- Attract new talented people, train them toward success and retain them.
- Achieve a certain break-even point

From the objectives identified, it is therefore necessary to determine the actions and tasks to be performed and consequently to make the most relevant operational decisions.

Here is a list of actions and managerial decisions that are not exhaustive and have proven their worthiness and which generally help to achieve the expected objectives.

1. Define priorities for action
2. Plan the activity of employees individually and collectively
3. Identify the performance of each in the execution of tasks
4. Ensure regular monitoring of actions
5. Bring the corrective actions

When one analyzes the decision-making mode on these different items, one realizes that the best-performing managers have in fact internalized a certain number of reflexes and that they use a neuronal program called "automatic piloting" which allows them to detach oneself from the constraints of the action itself to refocus oneself on the results of the action. This "automatic piloting" is possible only because it obeys a series of arbitrations based on a number of prerequisites related to:

• Sorting the information
• A knowledge referential (usually called concept)
• A mode of action (know-how)
• Attitude and behaviors (knowing how to be)
• A situation analysis of either inductive or deductive type.

Managerial decision-making is just one example of human activity that involves making choices between several well-identified alternatives.

9. Conclusion

The Cyphai protocol is a byproduct of two emerging technologies that are disrupting their respective fields. It utilizes the anonymous and distributed nature of smart contracts, and the intelligent problem solving aspect of machine learning. It also introduces a new method for crowdsourcing funds for computational research.

The protocol helps users solicit machine learning models for a given fee. The protocol does not require trust and works completely on a decentralized blockchain.

The protocol creates interesting opportunities like GPU mining arbitrage, provides a more transparent platform for raising money for things like medical research, and introduces an automated self-improvement system for AI agents.

It is expected that open-source ML models will significantly benefits from this. We might see a sudden rise of publicly available ML models available in the open-source community.

The protocol will potentially create a new marketplace where no middlemen is required. It’ll further democratize machine learning models, and increase opportunity in acquiring these models. This new levelled playing field should hopefully benefits both blockchain technology and machine learning, as it’ll provide a more efficient mean of obtaining machine learning models and increase smart contract usage over time.

The team

Robert Michit, Mourad Redjah , Youssef Azzouzi, Leopold Lessasy, Olivier Gilles

Michit Consulting - Mc2R
7 Place ANDRE MALRAUX
38000 GRENOBLE - FRANCE