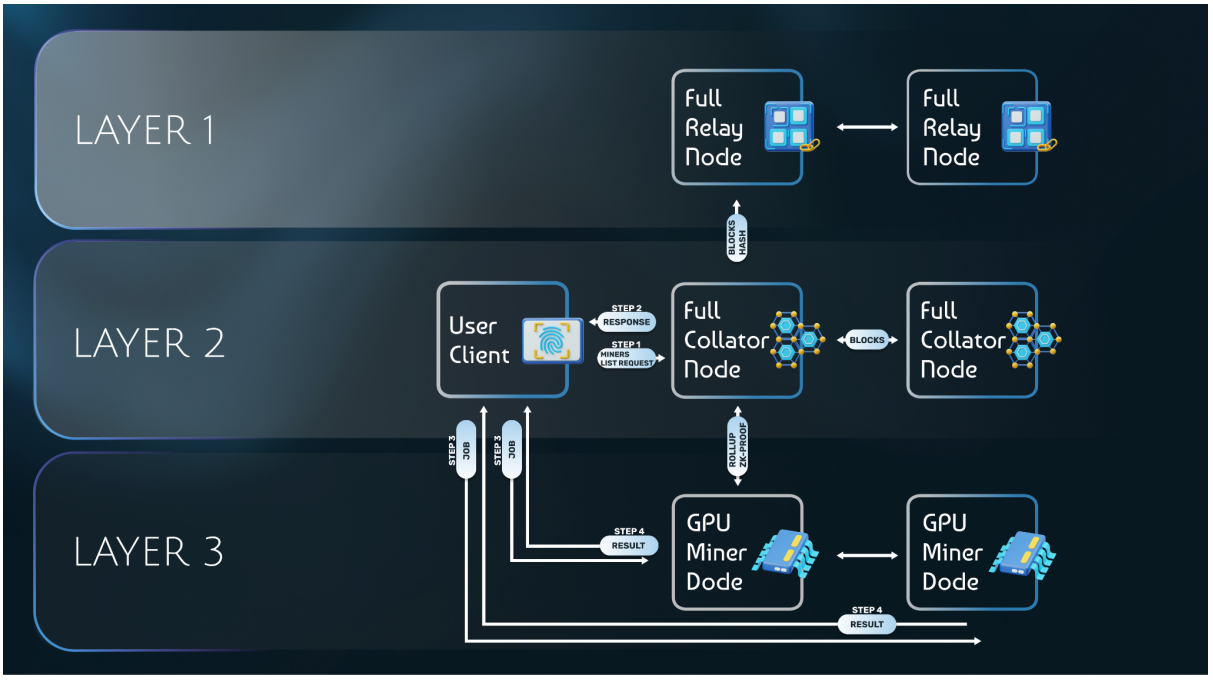
WendyAI - a parachain for neural networks

General information

WendyAI is a decentralized infrastructure for launching any neural network models on GPUs, their training, and task distribution among network nodes. It is built as a parachain on Polkadot and Kusama. The network does not have its native token; all fees are paid in KSM and DOT and are distributed among worker nodes, model owners, and network owners. To join the network, a workstation with a GPU is required, and all configuration is done through an executable file, lowering the entry barrier. All user network tasks are distributed among miners on the parachain, which serves as the main hub for work distribution. To prove miner execution, a ZK-proof is recorded in the blockchain.



Information about the team

Alex PromoTeam - Chief Business Developer
Polkadot Head Ambassador, PromoTeam member, Entrepreneur, Crypto and Blockchain Expert. Serial entrepreneur, for more than 15 years, had more than 50 projects in IT, Sales, Services, helped to establish more than 30 different businesses as a consultant, investor and / or founder. In crypto space from 2016 , full time from 2019. Helped to develop and run several Ambassador programs, including Bifrost

Ambassador Program. In collaboration with PromoTeam developed and executed several marketing campaigns for Integritee, InvArch and many other projects.

Vlady Limes - Chief Marketing Officer
Polkadot Senior Ambassador, PromoTeam member, Entrepreneur. Talented Photographer, Videographer and Youtuber. In Crypto from 2018, participated as content creator, content manager, community developer, investor and ambassador. Built PromoTeam and many other communities from scratch, with a total audience of more than 100k+.

Eugene Piker - Chief Technical Officer
Mathematician, Senior Blockchain Developer with strong Rust, C++, Substrate skills. Blockchain analytics and programming for over 7 years, Haskell programming over 12 years. C programming over 18 years, Rust programming over 4 years. Substrate development over 3 years.

Naikee Andy - Chief Commercial Officer

Expertise in marketing, mobile development and highload systems, investor. Participated in more than 50 blockchain projects as a marketing specialist and business developers. Worked with top-class banks in mobile development.

Technology

General description of the technology

The key equipment required for a miner is a GPU, a graphics card on which neural networks can be run and trained, and a hard disk for storing the chain of blocks and the weights of the neural network.

The runtime of the chain of blocks contains additional checks that test for the presence of the GPU and data on the disk. The weights on the disk have a Merkel root and possible probabilistic Monte Carlo checks for the presence of data.

There are text data located in IPFS, which have a PACK ROOT (a Merkel tree root). This hash is registered in the chain of blocks and gets its ID. This can be data used, for example, for training a translator. Another use case is an expert system in the form of an intelligent search, for example, simply through documentation, etc.

Next, there is a configuration of the neural network (systems like GPT), where it is indicated what it does. For example, that it is a translator that will be trained on data from this ID.

Collators, upon receiving this order, which could also have been voted on by some units, begin their training based on the data. Training is of course deterministic, and collators come to a single result, which has a consensus protection.

Also, the architecture of the neural network is designed to be compatible with ZK proof, using ZK allows for data processing to be done in parallel by each collator. It also opens up such a feature as checking calculations at the relay chain level, which means that chain of block blocks containing ZK proof, which is much smaller than the data and calculations themselves, has a certification property at the Layer0 (relay chain) level.
Of course, requests to the service itself (for example, a translator) do not need to be certified at Layer0. At the same time, requests are sent directly to backup nodes or collectors that contain a GPU and can process the request. These can be RPC requests.

Proving that requests are being processed is done in lottery mode, a collator may get a request that will be a winning ticket, then he certifies his request and places a ZK proof in the chain of blocks.

Since backup nodes have a chance to win the lottery, they are incentivized to keep GPUs and weights at their site.

Protection against requests fraud is achieved in such a way that other nodes must know the hash of the request, but only one node calculates the answer. Hashes of requests are distributed before finding out whether there is a win or not. Liveness and special consensus are not needed to distribute such hashes, as there is no need to pick up the win immediately, it can be picked up later.

Working principle:

- There is a set of nodes that have three sets of software:
    - A parachain collator that can make block proposals.
    - An application that performs full and efficient neural network and GPT computations, and can work using both CPU and GPU. It is compatible with popular weight stacks and can receive what has already been trained.
    - A ZK plonky mir application that creates proof for the calculation performed by the neural network program. All modern and standard functions such as GELU, TANH, SOFTMAX, and many more are supported. There is also the possibility to quickly implement your own functions as the root of the Merkle tree.
- The client has software that communicates via API with the nodes. The client has its account in the parachain, which contains payment for the work of these computations. By sending a request to the node, the client signs it, which allows the node to take a fee for the work.

Security, ZK-proof

Main points of the WendyAI security concept:

- Any work of any volume or complexity that is paid for through the parachain must be fully verifiable through ZK (without compromising performance).
- The concept of signing output data, which is done by a node only when it is confident that it can prove its calculations through ZK if necessary and that the proof will be successful.
- The concept of node risk before issuing incorrect output data and signing it. The node holds a stake that allows it to operate at a serious level, and its signature has significance and a certain stake weight.
    - If the node is not hacked, the hardware is working well, and everything is done correctly, then these risks are zero because any correct calculation is always verifiable through ZK.

Implications of protecting neural calculations:

- Ability to use large-scale GPU mining equipment and perform very large-scale calculations.
- Ability to create easy-to-use mining software and expand the network of GPU miners.
- There is no need for additional verification of calculations, and there is no need to duplicate the same calculations. This results in greater power and parallel servicing of clients by miners.

Frequently Asked Questions:

Even if the computation occurs on a GPU, is it still possible to create a full ZK proof for the entire computation and prove that the output vector is exactly as obtained?

- Is it impossible to transfer and prove computations that do not happen within the blockchain, and by giving output data, are we missing the contents of the computation?
    - All neural computations affect the output layer, and the correctness of the output layer is proven in ZK.
    - Even if the computation occurs on a GPU, it is still possible to create a full ZK proof for the entire computation and prove that the output vector is exactly as obtained.
- If a node has performed neural calculations and realized that it does not lead to a solve, then what is the point of sending this data to the client?
    - This is solved by the fact that the node cannot immediately understand if there is a solve or not. The node also needs to perform a VDF calculation that takes time.
    - A VDF calculation is 1% of the total volume of calculations, but it provides an effective time delay.
    - The node sends the answer to the user before the VDF is solved because the client, in case of receiving a quick response, will sign a positive compliment for the node.
    - Thus, this encourages every calculation of the node to necessarily have a chance of solve, which is possible only with correct calculation.
- Who wants their history of silly drunk Google searches to be publicly saved in the blockchain forever?

    - Not all queries end up in the blockchain, only those that are "solve" queries. Most queries receive a response directly from the node through an RPC, and the blockchain serves the functions of control and orchestration.
    - The input data for the model is never saved in clear text, it is always done in the form of a hash.
- And this is VERY EXPENSIVE in terms of overhead compared to just "setting up a server and selling a service."
    - But not in our case, in our case it's about orchestrating GPU computations and making them more parallel and secure. In our case, it's possible to even process large models, i.e. already billions of weights.
    - ZK proofs as well as additional intermediate probabilistic checks occur in random challenge mode to better control and minimize the delay of real computations. These checks take less than one thousandth or even less.

A node that participates in the parachain can be called a peculiar miner of neural network computations.
In Bitcoin, a nonce is iterated and hashed, while in our case, signed requests from clients are accepted and neural network computation is performed.
In Bitcoin, a solve is when the output result (hash) contains enough zeros, while in our case, it is a similar hash of the neural network output layer that takes into account the client's public key and the miner's public key, and this hash passes a threshold (similar to the number of zeros in Bitcoin).

In our case, proving the neural computation in ZK is required only when it is clear that the solution is found.

Payment model for request

Each user request in the form of a signed input layer for neural computation is an account action that includes the ability to pay a fee. The most common way is if the account has a set of native units (ROC, KSM, DOT) that are used for the fee.

These fees go to a common account that is a system account (like a smart contract account). When a miner finds a solve, it is a neural computation for certain weights (models) where the output was proven for a given input.

It is clear that the solve belongs to a certain model and there should be a reward for it. Obviously, this should not be just any model, but one for which a reward can be given.

It is already clear that there is a common fee account, and the obvious question for the model is whether to take funds for the model from this common account. It seems obvious that if the transaction that gave this fee could have been intended to use a particular model.

In short, there are things like:

- Fees that arose as a need to use a model

- Proof that the model was used and taking funds from the common account to reward it
- A common account containing funds for different models

Starting from simple to complex, it is usually efficient to load all GPUs with one model and declare that the miner is only doing calculations for that model. Then, from the blockchain history, it is clear that all attempts to find the solve will be made for this particular model. This approach is simple and practical to begin with.

Suppose there are several different models being paid for and the concept of sufficiency of resources working with this model, whether more resources are needed for the model. Suppose a model is already fully serviced, then other miners who want to do something are forced to take another model that requires miner resources to work.

This can even be called mining a neural network model as proof of work to use funds from the common account.

Dilemma of incentivization

Brief definition: if funds come from fees, but in order for fees to come, mining needs to be done.

Steps to reach a solution:
- Miners need to be incentivized when there are no fees yet.
- When there is incentivization and miners are working, that means requests are being made and fees are coming in.
- If fees are coming in, then the problem is solved, which means that the incentive has fulfilled its role and can be returned to the one who made it.
- The one who provides the incentive wants to benefit from it, since their funds are freed up when fees are received.

Now, with these steps in mind, let's try to present a smart function for this (substrate pallet).
- Call "deposit_to_model(m_id: ModelID, dep_a: Balance)" which means that the calling account has more funds than dep_a, and is willing to incentivize the model m_id. The funds go to a common account, and they are marked as specifically for model m_id. The account receives "pool tokens" or special assets, which are exactly the same amount as dep_a (or potentially more, with interest).
- Call "withdraw_from_model(m_id: ModelId, plt_a: Balance)" which means that the calling account wants to withdraw "pool tokens" in the amount of plt_a as value. However, they cannot withdraw everything they deposited with the "deposit_to_model" call all at once.

Variables used in the solution:
- Percents: the percentage profit that the stimulator wants.
- Cycles: the number of cycles required to release all pool tokens.

The miner has funds to receive payment, and there are funds in the common account.

Clients have funds for paying fees, and they send requests to the miner. The miner will not publish these requests as transactions because there would be too many transactions. Instead, the miner will wait for a solve and then attach these payments to the solve as a closing of a payment channel. Channel closures are also published when there is no GPU solve yet, but there is a solve for a PPK, which is even the main volume. (The next heading explains what a PPK is.)

The miner performs their solve and guaranteedly receives their payment from the common account for this solve, and all attached fees release the stimulus. In other words, fees also go to the common account but, at the same time, allow the withdrawal of stimuli in the amount of the fee (or if the cycles are specified, in the amount divided by the number of cycles), that is, to perform the operation withdraw_from_model.

The profit that the stimulator wants can simply come from an increased fee. While the stimulus is needed, the fee is increased by a certain percentage (for example, by 20%). The pool tokens are issued by 20% more. It is also interesting that if two cycles are specified, and a 20% profit is desired, it is enough to raise the fee by 10%, as it will continue to be released in the second cycle. By doing so, this can be 10 cycles, and then the addition to the percentages is almost imperceptible, about 2%.

There is an effect of momentum, where the initial stimulus occurs, and the fee starts to fill the common account. To make this effect more stable, cycles should be set to no less than two, preferably more. At the same time, the profit margin should be maintained at 20%; the fee should simply pass more "circles," and it will leave a new stimulus reserve for itself without external help. The initial stimulator will receive all their pool tokens "earned."

A payment channel-like mechanism which is used as a way to pay for fees (or a probabilistic payment channel)

To begin with, an explanation of why such payment channels are needed instead of classical ones. What is a Payment Channel Key (PPK)?

- It works probabilistically and incentivizes the client to lock a sum of money with a margin.
- It allows attaching fewer data to solve but adding more money to the common account.
- It eliminates the need for state tracking, achieved by using VRF (replacing the state with probability).
- It allows for automated unlocking of funds in the channel and can be done in continuous mode (which greatly simplifies the protocol and makes it independent of state tracking).

It has the following fields:

- Miner's public key.
- Model identifier.

- Hash of the input vector (or set of input data).
- A set of any data that you don't want to look up in the database.
    - This could be, for example, the hash of the weights from the model.
    - Also, the shape of the weight tensor, the number of layers, etc.

These fields are signed by the client. But before making such signatures, he opens a payment channel (PPK) for the miner. He deposits funds into this PPK specifically for the miner, for example, $10. For simplification, operations such as opening channels for all miners serving a particular model can be done in one transaction.

Similar to a regular payment channel, funds cannot be immediately withdrawn, and a certain time must be waited for this. For example, 50% of the funds can be released from the payment channel every day. This can happen according to a continuous formula, as the user still needs to make a claim (this operator can be simple, like "withdraw all from all my channels").

It is not beneficial for the client to make multiple signatures for the same input vector, whether it's just two signatures of the same vector or the signature of the same vector for a different miner. This causes increased fee spending, but the client pays the excess anyway. Therefore, he must clearly determine which miner he will send this input vector to. He can send another input vector to another miner, or to the same one, but the key is to send it to only one miner.

Now, let's talk about how the PPK is closed. When the miner receives the signature of the fields, he applies the VRF function to this signature, and of course, the public key must correspond to the miner's key. The VRF function is deterministic and provable.

Now, the variables that exist in the channel are:
- The amount of money in the channel
- The complexity of the VRF solution
- How much money is taken out of the channel when a solution is found

Usually, these variables are interdependent - the more money in the channel, the more complex the VRF solution, and all funds are taken out of the channel when a solution is found. If the variable for how much can be taken out is half the funds in the channel, then the VRF solution will be twice as easy, and so on.

This can be configured so that funds are taken out of the payment channel in increments of $1, for example by depositing $10. In this case, the solution will be 10 times easier than for $10, or the complexity of the solution will be the same as for $1.

How the principle of the fisher works

The term "fisher" is used in Polkadot, it is a kind of police controller for trolleybuses, a sort of inspector who can come across incorrect data and report a violation. This principle has

optimistic probabilistic properties. Here we are talking about the Fisher principle for mining neural network models.

PPK SOLV is a VRF that the miner creates. However, to publish and transfer the fee funds to the common account, this is not enough. Additional data must be attached to the VRF, which are signed and must have the property of verifiability. Typically, this is the hash of the output vector, for which the miner is responsible and declares by signing that the output is correct. The meaning of the Fisher principle is to publish some certified data more often than ZK proof, and also to have probabilistic protection at a "pre-ZK" level. This also allows to compactly track the dynamics of the miner's work until solve.

Since the verification is probabilistic, there is no task of checking everything indiscriminately, the collator, having its VRF function, can secretly know what it was probabilistically verified for. If he encounters a problem and is going to prove the mismatch, he can prove that it was precisely the VRF that was used for this verification. Of course, this is only possible for those models that the collator loads for himself.

Calculations of required mining equipment

Here is an example of a mining farm (text from a mining equipment store).

The total power consumption of a turnkey mining farm with 4 cards purchased from our online store is 585 watts, taking into account the operation of the motherboard, hard drive, and other peripherals. The total hashrate is 118 Mh/s, with an efficiency of 4.9 W/MH. The model's payback period for Ethereum mining will be 7-8 months, assuming a stable exchange rate and access to low-cost electricity.

Now, more about the graphics card:

The main structural element of the adapter is the GeForce GTX 1660 SUPER video processor. The device has a memory capacity of 6 GB, and the memory bandwidth is 336 GB/s.
Number of universal processors (ALU): 1408
*In total, we have 1408 * 4 = 5632 total number of cores*
*We have 6 * 4 = 24 gigabytes of video memory (more than enough for GPT-4, where there are several hundred million weights).*
Power consumption: 585 watts

A billion weights **are more than enough** for GPT-4, which may have several hundred million weights.
Even larger models can be considered, for example, farms of larger size or even several farms. And even multiple miners who load a giant model, each with their own piece, making their own super-stack.

If the miner has good internet, then he can receive and send requests. Unlike traditional mining, network requirements are higher here.

In the super-stack, it is important to have a connection with the main peer-to-peer center that accepts and returns final requests, but that is another story for another article.
In the future, there may be specialized software for such things, for example, to create pools for those who do not have large capacities but want to earn by mining a large model with their piece.

The average electricity rate in Texas is 14.82 cents per kilowatt-hour (kWh). This rate is 5% lower than the national average.
It can be concluded that operating such a farm will cost about $60 just for electricity.

In Texas, the cost of unlimited internet access varies depending on the provider. Generally, prices range from $39.95 to $49.95 per month. Some providers, such as CenturyLink, offer bundle discounts that can reduce the cost of internet access. Additionally, there are other providers such as ViaSat that offer unlimited data plans for around $150 per month.

Let's suppose a miner decided to set up two mining rigs so that the cost of the rigs would be higher than the cost of internet. This results in more power and more requests that can be processed, and the miner can handle multiple models by creating multiple accounts.
Let's say the internet cost is $50, so the total cost would be $170 ($50 + $60 + $60).
Let's also assume the miner wants a decent profit, say 50%. So the total cost would be $256 ($170 + $86).
The equipment cost is approximately $2000, which means it would take 23 months to break even (2000 / 86). This seems too long.
The miner wants to break even in at most 6 months. So the maximum cost per month would be $333.3 (2000 / 6).
Adding that to the previous cost, the total cost would be around $500 ($170 + $333.3), or $503.3 to be exact.
Now let's calculate the coefficients.
Each rig has 24 GB of video memory, and since there are two rigs, there is a total of 48 GB of video memory. The 2 rigs combined have 11,264 cores.
Per month, the cost would be:

- $10 per GB of video memory ($500 / 48)
- 22.5 cores per dollar (11,264 / 500)

Of course, there are many miners (let's say at least 50 for starters), all of whom want to break even. Their equipment operates 24/7 for months without interruption, and their internet connection will be filled with requests for a certain percentage of their total bandwidth (if it is completely filled, increasing the power of the rigs no longer makes sense for the miner).

A parachain block comes out every 12 seconds, assuming each block has some miner's solve who has been lucky.
Of course, there will usually be empty blocks without a solve since our consensus does not depend on GPU work.

In theory, one solve should correspond to 1000 real requests, or even more than 1000 (logic of not less than 1000).

If all blocks are filled with solves at a difficulty of 1000 requests per solve, then this is 1000/12 = 83.3 requests per second.

83 requests per second are 83 responses from the chatbot. It seems that about 1000 users use the service and each sends their question with a frequency of 12 seconds.

If there are more requests and you don't want to place more than one solve in a block, then the difficulty will have to be increased (for example, 2000 requests per solve or even more). But miners also wanted increased payment proportional to the increased difficulty of their work, so that their desired equipment payback parameters would be maintained.
So there are two variables, difficulty and solve price, which are linked to a common variable (the fee for processing one request for the model).

Each model can have its own difficulty and solve price and a common variable. That is, there are three variables.
It is clear that the difficulty of all models will tend to be roughly the same, but this does not mean that a common difficulty should be set.

Each miner has a set of accounts, where each account can only be responsible for one model and also has its own declared throughput, such as requests per hour. If a miner wants to rest, he sends a "chill" transaction and officially goes on vacation, requests are now distributed differently. And vice versa, if he wants to start accepting requests, he does a "wakeup".

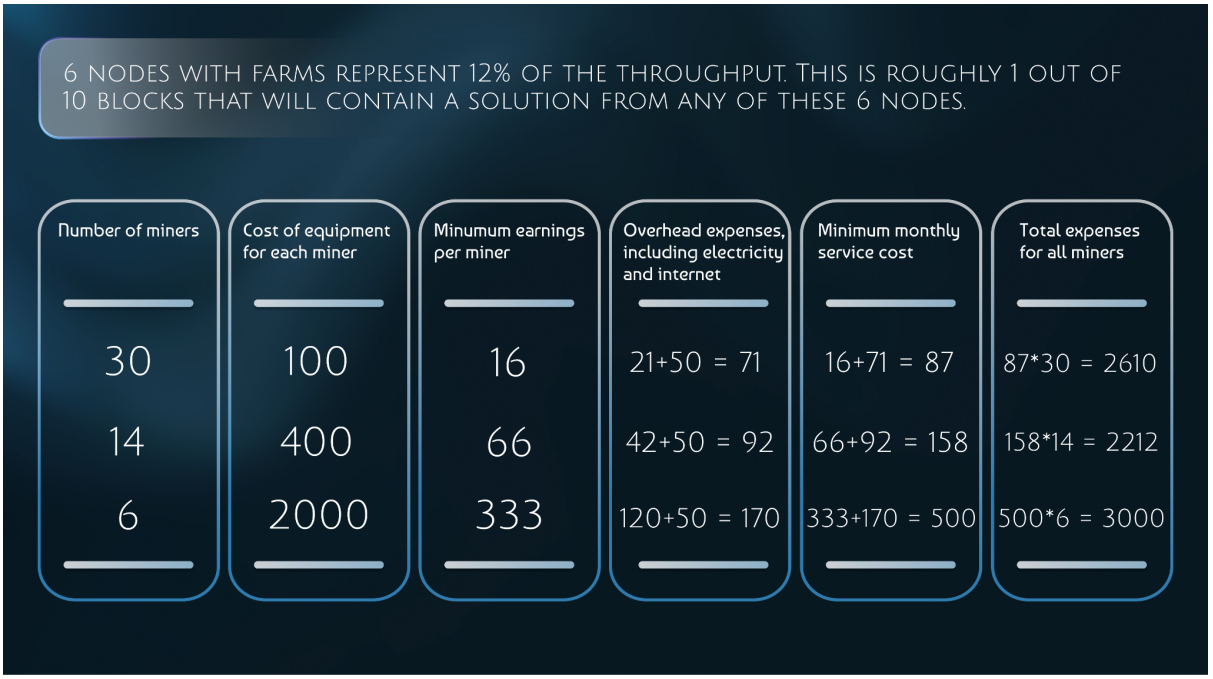
The application that accesses the API extracts planning information from the blockchain, who has what models, and who has what throughput. Based on this, it makes conclusions about where to send requests.

Let's say there are currently 50 miners in the WendyAI network working on 4 models. The large model has 1 billion weights, while two medium models have 200 million weights each, and one small model has 50 million weights (or there could be a set of small models). Further development of the idea could allow for a small model to be used as pre-training for a large model, such as a custom additional layer or layers.

Miners who have a whole GPU miner (a stack of video cards) can take the large model, as well as all the others. Miners who have one powerful GPU card on board can take one of the two medium models and still have resources to take one small model. Miners who have a regular GPU card and good internet can simply take the small model.

Let's assume the composition of the 50 miners is as follows: 30 regular, 14 medium, and 6 with farms.

For those with farms, if the payback period is more than 6 months, they will leave. This should be kept in mind. It's clear that everyone loves bonuses and is always willing to get more, but the most important thing is that profitability never hits rock bottom, preferably. Miners with farms have fewer competitors, so they take the large model and share the profit from it among themselves.



6 nodes with farms represent 12% of the throughput. This is roughly 1 out of 10 blocks that will contain a solution from any of these 6 nodes.

Number of miners	Cost of equipment for each miner	Minumum earnings per miner	Overhead expenses, including electricity and internet	Minimum monthly service cost	Total expenses for all miners
30	100	16	21 + 50 = 71	16 + 71 = 87	87 * 30 = 2610
14	400	66	42 + 50 = 92	66 + 92 = 158	158 * 14 = 2212
6	2000	333	120 + 50 = 170	333 + 170 = 500	500 * 6 = 3000

2610 + 2212 + 3000 = 7822 dollars as the minimum conditions for the network so that no one starts to leave.

Without increasing the complexity of solve, i.e. this is 1 solve complexity for 1000 requests, with 1000 users where each sends approximately 1 request every 12 seconds.
It is clear that the user will not send requests around the clock, suppose he logs in 16 times during the daytime and makes 16 requests every 12 seconds.

This is 16 * 16 * 1000 = 256 * 1000, i.e. 256 solves.
There can be 7200 blocks per day, 7200 / 256 = 28.1 or 3.5% network load.

It is clear that the load may fluctuate on certain days, for example, up to 50%, or fall.
If the network load stays at a high level for a long time, it may be necessary to increase the complexity, and the factor of new miners can also be taken into account (to create conditions and/or open up opportunities for an increase in the number of miners).

Now, taking into account the usual network load as an average, we calculate that 256 * 1000 * 30 ~= 2^23 requests per month.
7822 / 2^23 ~= 0.1 cents per one request (a tenth of a cent).
A user spends 24 cents per day for his 256 requests.
If the number of miners does not change but the number of requests grows, the complexity simply increases.

If the miners do not experience any problems with increasing the number of requests and are able to process them in time, the profitability remains the same and the cost of one request decreases. And this means that the price of solve does not change, but the complexity increases.

That is, this is an example of a case when the more it is used, the cheaper it gets; this is the phase when using it means mastering the technology. The phase of "absorbing" customers, making sure they are satisfied and bringing even more customers.

Next, the dynamics may be such that the number of requests may start to increase so much that one of the miners will already feel discomfort, for example, he cannot process requests in time, and the price of solve at such a high complexity is also unsatisfactory.

In theory, this should be predicted by certain curves, the system should be ready in advance. This indicates that more miners are needed, and conditions should be included when it will be profitable for new miners to join (conditions created, opportunities opened, possibly automatically).

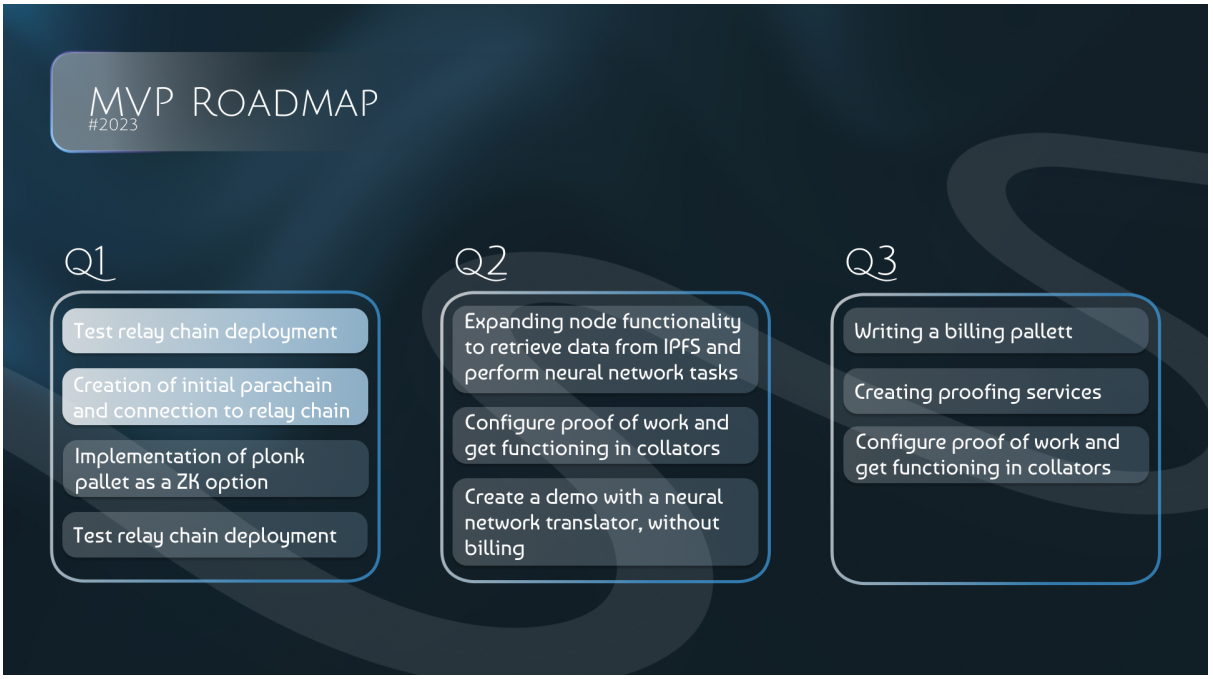
Suppose the number of miners has doubled, now there are 60 + 28 + 12 = 100 miners, where each category has simply doubled.
The complexity will be twice as much, i.e. 2000 requests per solve.
Now let's think about what will happen to the price per solve and per one request.
(7822 * 2) / (2^23 * 2) = the same price per request.
This means that the price of solve should be twice as high.

Simply put, we can explain it as follows: let's take some miner from the old network and compare his conditions with those he will have in the new network. He will receive approximately the same number of requests, the network has grown and requests to the network have grown, but he gets approximately the same number of requests. Then the complexity increased, but so did the price of solve. That is, he just feels the growth of complexity.

Clearly, a well-designed system of curves that can effectively amortize network parameter variability is beneficial. However, no matter how perfect the system may be, it will require governance intervention. This means that using governance referendums from the standard substrate of Polkadot is a good idea. Adjustments can be made to the curve system parameters.

MVP Roadmap

Test relay chain deployment - **done**
Creation of initial parachain and connection to relay chain - **done**
Implementation of plonk pallet as a ZK option **- done**
Writing a GPT compatible with ZK **- done**
Expanding node functionality to retrieve data from IPFS and perform neural network tasks
Configure proof of work and get functioning in collators
Create a demo with a neural network translator, without billing **- done**
Writing a billing pallett
Creating proofing services
Expanding use cases, creating an example expert system



Use cases and demo

There is a program called fast_gpt2. It already has trained neural network weights and a variety of different GPT elements and layers.

To quickly demonstrate a demo, we need to involve the blockchain for only one component. The Gpt2 Layers are used, and one of the first layers is processed through the blockchain. Then, the program continues to work independently.

This will allow us to develop features such as clients using neural network computations through blockchain:

- debugging fee payments,
- debugging proof and validation of correct operation,
- helping to develop a software version that uses GPU.
- We can also introduce penalties for incorrect behavior of blockchain nodes, which have already been debugged in the Substrate Polkadot system and can be used.

Next, we can formulate plans for a full transfer of the application to blockchain

Also, we have plans for other applications, such as:

- collaborating with Robonomics for object recognition with drones and route planning,
- a neural network for checkers game,
- a recommendation system for retail trade in online stores (module),
- a service for estimating the value of any NFT from an image,
- a chatbot with a mobile application,
- a tasking system similar to Jira with a virtual project,
- collaborating with RMRK for generating images with NFT minting on RMRK.

Business strategy

From the very beginning, WendyAI has been focused on full-fledged work and active use of its infrastructure. We are aimed at bringing practical benefits and truly attracting users to the web3 ecosystem.

1. Develop a Clear Revenue Model: Define the network's revenue model and identify how it will generate revenue. This could include transaction fees, membership fees, or other revenue streams.
2. Build a User Base: Develop a user acquisition strategy that attracts a broad user base, including individuals, businesses, and developers. Offer incentives and user-friendly features to attract new users and retain existing ones.
3. Create Value-Added Services: Develop value-added services that incentivize users to remain on the network and generate additional revenue streams. This could

include mobile applications, web applications and services such as data analytics, smart contract development, or other blockchain-related services.

4. Establish Strategic Partnerships: Forge partnerships with businesses, investors, and other parachains to expand the network's reach and add value to the network. These partnerships could include joint ventures, mergers, or other strategic collaborations.
5. Prioritize Security and Compliance: Ensure that the network is secure and compliant with all relevant regulations. This will help build trust and attract a larger user base.
6. Continuously Innovate: Invest in research and development to continuously improve the network and stay ahead of the competition. This could include developing new technologies, improving user experience, or adding new features.
7. Monitor Metrics and KPIs: Regularly track key performance indicators (KPIs) and metrics to evaluate the network's success and make data-driven decisions. This could include metrics such as user acquisition, revenue, and user engagement.
8. Enhance Polkadot's Capabilities: Position the blockchain network as a valuable addition to the Polkadot ecosystem by providing all the features of neural networks to its parachains. This will allow other projects within the Polkadot ecosystem to utilize the network's capabilities, generating additional revenue streams and expanding the network's reach. Additionally, this will help establish the network as a leader in the blockchain space and attract further investment and partnerships.

By implementing these strategies, the blockchain network can generate sustainable revenue and provide value to shareholders while maintaining a strong user base and expanding its reach.

Market analysis

Total economic impact of AI in the period to 2030- $15.7 trillion game changer
What comes through strongly from all the analysis we've carried out for this report is just how big a game changer AI is likely to be, and how much value potential is up for grabs. AI could contribute up to $15.7 trillion1 to the global economy in 2030, more than the current output of China and India combined. Of this, $6.6 trillion is likely to come from increased productivity and $9.1 trillion is likely to come from consumption-side effects.

According to PWC
- $15.7tr Potential contribution to the global economy by 2030 from AI
- Up to 26% boost in GDP for local economies from AI by 2030
- 300 AI use cases identified and rated are captured in our AI Impact Index

The artificial intelligence market was valued at USD 58.15 billion in 2021, and it is expected to reach USD 271.48 billion by 2027, registering a CAGR of 31.45% during the forecast

period 2022 - 2027. COVID-19 positively impacted the market as few industries witnessed an increase in AI adoption; however, others faced a decline. The COVID-19 debacle has educated business executives with vital insights about digital transformation. The potential that data analytics and artificial intelligence bring to an organization is one of the most compelling teachings. For instance, AI helps push processes, people, and services online in the public sector and compels local, regional, and national governments to embrace AI. In a few months, governments globally have learned to use AI as a weapon to fight against the virus, from educating the public and screening patients to tracking and tracing contacts.

- Artificial intelligence (AI), including computer vision and machine learning (ML), is changing the landscape of the industries. The market for artificial intelligence is growing across the globe owing to the growth of data-based AI and advancements in deep learning, and the need to achieve robotic autonomy to stay competitive in a global market is expected to drive the adoption of AI solutions and services.
- Moreover, the continuous research and innovation directed by the tech giants are driving the adoption of advanced technologies in industry verticals, such as automotive, healthcare, retail, finance, and manufacturing. For instance, in November 2020, Intel Corporation acquired Cnvrg.io., an Israeli company that develops and operates a platform for data scientists to build and run machine learning models to boost the artificial intelligence (AI) business. However, technology has always been an essential element for these industries, but AI has brought technology to the center of organizations. With the growing number of enterprises and increasing competition, companies are rigorously trying to integrate artificial intelligence (AI) technology within their applications, businesses, analytics, and services.

Risks

As with any blockchain network, there are a number of risks that should be considered when implementing a system that allows for decentralized neural networks.

One risk is the potential for smart contract bugs, which can allow malicious actors to exploit vulnerabilities and steal funds or data. This risk can be mitigated through thorough testing and auditing of smart contracts before they are deployed on the network.

Additionally, there is the risk of regulatory uncertainty, as governments around the world are still grappling with how to regulate blockchain and cryptocurrency technologies. This could lead to changes in regulations or even outright bans, which could have a significant impact on the viability of the network.

Finally, there is the risk of market volatility, which can impact the value of any cryptocurrencies used within the network. This could impact the ability of users to access the network or even result in a loss of value for investors.

Overall, while there are risks associated with implementing a blockchain network that allows for decentralized neural networks, these risks can be mitigated through careful planning, testing, and adherence to best practices.