Ambient Litepaper

Abstract

Ambient is an SVM-compatible PoW L1 that will serve as a cornerstone of the agentic economy, unleashing Asimov-ian intelligence on chain. It is 10x more efficient than incumbent crypto AI systems, and features:

- Fully verified Inference with < .1% overhead but high security on one huge smart model (600b+ parameters) and its fine-tunes
- 10x better training performance than existing approaches
- Extremely high utilization of miners due to optimization on a single model for inference and validation
- A non-blocking proof of work consensus that foregrounds economic competition around the core activities of the network (inference, fine tuning, training) while maintaining extraordinary TPS

Introduction

Imagine a technology that could increase the baseline IQ of workers by twenty points, but which would only work if continuously utilized. Such a technology would have immediate productivity impacts on the workforce, and would become a de facto requirement for employment, the stickiest application ever. LLMs are that technology, and they are becoming a requirement for economic competitiveness. ChatGPT has been the fastest growing application in history.[1] To illustrate the progression: Mere weeks after its introduction, its phrasings

[1] https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

appeared in 10% of academic papers[2] and by 2023 17.3% of *sentences* in computer science papers were conservatively estimated to be written by ChatGPT[3].

We are headed toward an agentic economy, in which human / LLM synergy is continuous, and in which agents are going to be replacing an ever increasing proportion of human labor. AI is set to drive 13 trillion dollars in economic activity or 1.2% of additional GDP growth per year until 2030 and 66% of jobs are estimated to be subject to AI automation, with 25% being fully replaceable by AI.[4]

While closed source players initially dominated the field, it has become increasingly evident that 'there is no moat'[5] and that open source and open weights models (like DeepSeek R1) provide a credible alternative to closed frontier models. Such alternatives are necessary because closed models are under increasing pressure to monetize user data in hostile ways and have even begun cannibalizing popular applications built on their platforms (for example: Perplexity[6]). At the same time, even open weights model hosting providers have come under scrutiny for opaquely reducing service / model quality in order to reduce costs[7] (a serious concern for agentic businesses) and their centralization makes them vulnerable to censorship.

In a world where user interfaces are mostly APIs mediated by AI, Crypto is strongly positioned to become directly integrated into the broader software economy but perhaps surprisingly has not produced a credible AI solution. There seem to be four primary reasons for this, two technical and two related to product design / philosophy.

The first technical reason is that CryptoAI has put forward a series of inadequate solutions for 'verified inference,' the process of proving that a particular model generated a particular piece of text, that suffer either from severe overhead in some form or poor security.

---

[2] https://t.co/NQr7J6Yvdi
[3] https://t.co/5DUpPRI7kc
[4] https://www.nexford.edu/insights/how-will-ai-affect-jobs
[5] https://www.semianalysis.com/p/google-we-have-no-moat-and-neither
[6] https://www.theverge.com/2024/7/25/24205701/openai-searchgpt-ai-search-engine-google-perplexity-rival
[7] https://www.together.ai/blog/llama-31-quality

For example, ZK Proofs[8] use mathematical transformations to prove the inference happened on a particular model,[9] at a cost of vast quantities of memory and CPU time (something like 1000x overhead). Hybrid ZK approaches compromise on the security (by taking a sampling approach) but still suffer from something like 10x overhead and having unacceptable timing performance. Optimistic approaches, whilst having been continuously suggested for several years have not seen wide adoption or acceptance due to a combination of unpredictable overhead and performance and a severe issue with asymmetric risk involving scenarios wherein the inference consumer (say, a DeFi loan application provider looking at a $10 million loan) has much more to lose than the inference provider (who might only be slashed for $10,000). In addition, it does not appear that most optimistic approaches can even reproduce LLM calculations across machines, making their claims about auditing seem dubious at best. Trusted execution engines suffer from trusting providers that have been repeatedly compromised, that are likely providing backdoors to governments, and that use an inefficient system (the use of memory traces imposes a huge support burden for limited benefits).

The second technical reason is that approaches to decentralized training have historically been about ten times too inefficient to be considered usable and approaches to decentralized training have been underspecified. While recent advances such as OpenDiLoCo and a preliminary report from Nous have been suggestive (and may incline some to declare victory early), the reality is that no credible plan has been put forward by CryptoAI to train even a single model of the quality of say, Llama 3.1.

The third reason and first philosophical reason that Crypto AI has floundered has been a preoccupation with ‘marketplaces of models’ that invariably end up as ‘marketplaces without a market.’ From a user perspective, such marketplaces present a paralyzing ‘paradox of choice’ . From an agentic perspective, it is irrational to depend upon ‘mystery’ models of unknown

[8] https://docs.inferencelabs.com/zk-ml/benchmarks
[9] https://arxiv.org/pdf/2310.14848

provenance trained on unknown data with unknown performance characteristics. Agentic operators are not willing to risk capital on such things. From a miner's perspective, miners have struggled hugely with 'marketplace' approaches as they impose a huge cognitive load on the mining entity. A miner's goal in life is to get a steady, predictable rate of return on investment. Model marketplaces force a miner to continuously parse the model catalog to understand what is popular, and to attempt to optimize hardware deployments against multiple moving targets. Miners do not get good utilization from marketplaces, either, as model downloading and loading costs impose a substantial downtime burden.

The fourth reason and second philosophical reason is that the vast majority of Crypto AI approaches have attempted to innovate on the AI side while ignoring the economic model that historically produced the greatest miner engagement, Proof of Work. By adopting pure Proof of Stake systems to secure networks, Crypto AI has in fact adopted an incentive scheme that actively discourages miners, as 'incumbents' have already 'won' perpetual low-risk rewards, and the upside for miners themselves is limited.

Thus, the burgeoning agentic economy is left without a credible crypto AI solution competitive with closed source alternatives, without an L1 AI blockchain network that can provide efficient and performant verified inference, fine tuning, and training on a large LLM in a simple to adopt manner while guaranteeing quality. The blockchains that do exist do not: a) Verify model provenance b) Support auditability or reproducibility of model execution or training c) Support transparent sharding and optimized concurrency for extremely large models d) Provide much in the way of user privacy / anonymity guarantees. e) Provide quality of service guarantees around model quantization and response times. f) Give miners sufficient rewards. g) Allow miners to optimize model delivery by focusing on one model architecture (due to marketplace dynamics fragmenting delivery requirements).

From a community perspective, current L1 blockchains are also deficient in the sense that: a) They provide no censorship resistance, b) They do not further the widely held

community goal of training extremely large non-commercial models in a transparent programmatically specified way for the public good.

Ambient sets out to address the deficiencies just described by building a fundamental pillar of the agentic economy: an AI secured blockchain ten times more efficient than incumbent systems with built-in privacy and censorship resistance that runs a single, huge, highly performant, auditable, and transparent model (and its fine tunes) at low latency by leveraging hyperscale on-chain distributed computing to deliver human-like capabilities to applications on-chain and cross-chain. It also sets out through its design to deliver what miners and the community want: a sink for idle GPU resources with guaranteed market demand and competitive consistent rewards that welcomes all contributors and continuously deploys a portion of network power in service of keeping the network model evergreen via fine tuning and even additional pre-training, and which ultimately supports training an upgrade / replacement to the starting model on-chain.

The key insight driving Ambient is that to deliver a trusted experience on trustless hardware, the blockchain itself must be secured using the core network applications: inference on and training of a huge multimodal large language model. Incorporating AI onto the blockchain enables “provably secure computation” [10] on untrusted hardware, while the incorporation of privacy-preserving technologies into the network design ensures that users can leverage the technology without fear of censorship or reprisal. The potential of a new large language model trained on the blockchain itself with both the training process and the weights being fully open crucially enables the auditability and transparency needed to secure public confidence. In other words, the design of Ambient ensures the integrity of the model and experience of a service competitive with ChatGPT. Ambient relies on the rock solid blockchain primitives established by Solana paired with its own proof of work to deliver on this vision.

[10] https://cloud.google.com/blog/topics/systems/the-fifth-epoch-of-distributed-computing

Technical Design

Ambient pairs a new consensus proof of work algorithm called “proof of logits” (PoL) with the Proof Of History (PoH) architecture described in the Solana white paper. The result is a highly efficient blockchain architecture that guarantees known good output when running a high quality base model.

Ambient Node Architecture

APIs and SDKs	Language SDKs
	RESTful Services
Blockchain Infrastructure	Smart Contracts
	Consensus Processing
	Transaction Services
AI Model Training and Execution Engine	Model Profiler
	Sharding Manager
Network Services	Data Oracles
	Query Processing
	Privacy Primitives

The diagram above summarizes the different architectural components implemented within a given Ambient node. Each node is equipped with a set of network service primitives that ensure anonymized query processing and access to data. A model profiler offers fingerprints used by validators for both inference and training operations. Smart contracts are executed in Ambient’s fork of the SVM. Clients can access all the subsystems of a given node via simple

RESTful services provided as well as via language-specific SDKs. Initial SDK targets would include Javascript, Python, Go, and Rust.

In the following subsections, after first concisely defining some key terms, we will explain the core consensus mechanism of “proof of logits” at a high level, then describe how PoL is integrated with “proof of history”, describe extensions in relation to model training and sharding and then touch briefly on each element of the node architecture.

Technical Note:

The proposed blockchain implementation stands on the shoulders of Solana’s implementation, but replaces its proof of stake with a proof of work in a manner we will describe later. For more information on Solana, please see : https://solana.com/solana-whitepaper.pdf. To simplify the discussion herein, randomness and other core blockchain design issues are assumed to be dealt with according to Solana best practices.

Key Concepts

Logits

In the context of machine learning, especially in neural networks, logits are the raw, unnormalized outputs of the last layer of a model before applying an activation function, such as the softmax function. In a classification model, for example, logits are the inputs to the softmax function which outputs probabilities for each class.

In large language models, logits represent the raw output values that are produced by the model for each token in the vocabulary before any normalization is applied to turn these values into probabilities. Each logit corresponds to a model's prediction of the next word or token's likelihood, given the input sequence.

How an LLM Processes Data

Input Encoding: The computation chain begins with encoding the input text into a format the model can understand, usually as a sequence of tokens or embeddings.

Model Layers: The input then passes through multiple layers of the model (e.g., transformer layers in LLMs). These layers process the input in stages, applying attention mechanisms, and other transformations to capture complex patterns in the data.

Output Layer: At the final stage, the last layer of the model produces logits. These are the raw, unnormalized scores for each token in the model's vocabulary, indicating their likelihood given the context provided by the input sequence.

Normalization (Softmax): To convert the logits into probabilities, a softmax function is applied. This function exponentiates and normalizes the logits, making them sum up to 1, thus turning them into a probability distribution over all possible next tokens.

Selection: Based on these probabilities, the model selects the next token (e.g., through sampling or picking the most likely token). This process repeats for generating sequences token by token until a stop condition is met (e.g., generating an end-of-sentence token or reaching a maximum length).

Logits as Fingerprints

Essentially, you can look at the combination of all logits produced as a unique fingerprint for a model's thinking as it generates every single token in a text response. Conveniently, logits can easily be hashed, reducing this fingerprint to a single number. *Assuming a model was running in a virtual environment that abstracted away certain hardware and software quirks (like the handling of floating point and random numbers), a machine running the same model as another machine would always produce the same logits given the same inputs and it would not be possible to forge logits using another model and remain undetected given a combination of both user and random system queries (An identical effect to that of the virtual machine described can be achieved by generating hardware agnostic representations of the logits themselves, which is the technique that Ambient undertakes)*.

As well, it is also not necessary for a validator node to fully replicate another node's work in order to validate its logit output. This is due to the fact that with access to the input, output and a hash of logits at different stages of the output generation (let us call these stages progress markers), a given validator need only run full inference from one token prior in the output stream up to a given marker in order to generate an equivalent set of logits that can be hashed and compared to the hash that has been provided.

The result is radically reduced computation required for validation as compared to generation, a desirable property that is conceptually similar to what Bitcoin does with hashes, but which, unlike Bitcoin, puts the computation involved to a useful purpose at the same time.

To illustrate how this can be true, let's create a very simplified example using a model that only has a four word output vocabulary. This model's output vocabulary consists of the words:

[Spain, the, in, rain] - *This is the vector containing the vocabulary*
0 1 2 3 - *These are the positions of the vocabulary words in the vector*

Now, suppose we send this model the following query: “Write the first four words of a common typing exercise”

And suppose that the output of the model is:
“the rain in Spain”

And suppose that the model shares a hash of its logits for each step of its processing.
The logits calculated for the first word might look like :
A
[0, 3.7, -1, 2]
0 1 2 3

In this case, the word “the” (vector number 1 in our vocabulary) is by far the most likely output, indicated by the fact that 3.7 is greater than the other numbers. Let’s say the hash of this set of logits is:
b9776d7

We repeat this process for the other tokens, eventually getting the output.

Assume that a validator is attempting to test whether our machine produced “the rain in Spain” using the same model and it has access to the input, the output, and the hash from marker C. All it then has to do is take all of the input up to but not including marker C and run inference to produce the next token.

So, given “Write the first four words of a common typing exercise” and “the rain” as input, if the hash of the next set of logits is not equal to the previously produced set, then a different model is being run and validation fails. Simply inferencing on one token (a tiny cost compared to running the whole model to generate all the response tokens) can thus validate that one model running is the same as another model.

Proof of Logits and Consensus

The “proof of logits” proof of work algorithm exploits the fact that logits are both unique fingerprints and that hashes of the model’s logits during generation can effectively capture the model’s “thinking” at a given point in time (as a model ‘streams’ output).

Under this scheme, the proof of logits hash shall be the hash of the list of hashes of each set of logits antecedent to each outputted token. In other words, for each token n, up until a final token t, the proof of logits hash is:

Hash(Hash(n) … Hash(t))

The proof of logits progress marker hash shall be the hash of the logits generated x tokens into the output, where x falls between n and t inclusive, being:

Hash(n) … Hash(x) *… Hash(t)*

Now we can construct a validation scheme as follows:

- Miner produces text
- Validator picks a random word in text and requests a miner's "state of thinking" at that point in the text (corresponding to the proof of logits progress marker hash)
- Validator runs one token of inference on same word in context on same model to produce its own "state of thinking"
- If states of thinking (as represented by the hashes) match, then output is validated

This proof of work matches the design principle of Bitcoin, in that mining (in our case repeated exercise of the model through, say 4000 tokens of inference) is very costly, but validation is very cheap (requiring only 1 token of inference).[11]

What follows is a section that describes how the proof is utilized in the network design. A bit of background is required to get there. Specifically, it is necessary to describe core Solana mechanics and then contrast with our approach leveraging the proof of logits. If you are unfamiliar with Solana's Proof of History, we recommend reading their whitepaper as a refresher. In Solana:

[11] Note: There are additional layers of security that help to make this scheme undefeatable; we have narrowed our description for simplicity and because the process described represents the dominant term in the validation cost equation.

- Finality is determined through a process called "Tower BFT," which is Solana's version of Practical Byzantine Fault Tolerance (PBFT).
- A supermajority vote is required for finality. This means that more than 2/3 of the stake-weighted validators must agree on the state.
- Specifically, validators vote on the PoH sequence and the resulting state after processing transactions.
- Once a block receives votes from validators representing more than 2/3 of the total stake, it's considered finalized.
- This process happens very quickly in Solana, often within 400ms, leading to near-instant finality.

Importantly:

- The current leader node (elected through PoS) is responsible for processing transactions and incorporating them into the PoH sequence.
- The leader orders transactions, executes them, and produces a new state.
- This new state, along with the PoH sequence, is then broadcast to validators.
- Validators verify the correctness of the PoH sequence and the resulting state.
- If validators agree (by supermajority as mentioned above), the transactions are considered confirmed and added to the blockchain.

We note that:

- It's not just about high-stake nodes agreeing. While stake weight is important, the system requires a supermajority of all stake, not just a few large stakeholders.
- The leader's role is to propose the order and execution of transactions, but finality depends on validator consensus.
- Solana's architecture allows for parallel transaction processing and verification, contributing to its high throughput.

In essence, while the leader proposes the transactions and their order, it's the consensus of the validator set (weighted by stake) that ultimately determines what gets added to the blockchain. This system allows for both speed and security in transaction processing.

Ambient's Approach

In a pure proof of stake system, stake has been abstracted to be purely economic. In a pure proof of work system, stake represents the amount of computation brought to bear in the present moment to solve a problem, and more compute = greater likelihood of success. Pure proof of stake systems are very fast because they are non-blocking, but they suffer from problems with decentralization (Solana was almost taken down, for example, when Hetzner kicked Solana nodes off its network https://www.reddit.com/r/CryptoCurrency/comments/ykaz2z/1000_solana_validators_go_offline_as_hetzner/) because profitable validation requires a huge stake, which tends to inherently limit the number of validators. Pure proof of work systems are slow at processing useful work

because all nodes are trying to solve the same problem, the problem is very hard, and the solution to the problem is blocking.

As mentioned, our design goal then is to make a useful non blocking proof of work system. How can we accomplish this? In order to think about this, let's relax some of the design constraints for proof of work to reflect our concerns, while keeping elements that help facilitate decentralization and maximizing speed a la PoS. Imagine:

- Miners and validators of ML ops are relatively interchangeable (same as existing proof of work)
- Miners and validators of ML ops both get some rewards relative to the amount of compute they expend (same as existing proof of work)
- But… Miners don't have to all work on the same problem, as long as their work is validated by a quorum of randomly selected reputable validators (more on 'reputable' in a moment). This design choice removes the upper bound on the number of ML ops that can be performed by the network, making it effectively infinite
- And … Miners and validators get 'credit' for their contributions of valid proofs of work in the short term (days) and the medium term (months). This credit is an abstraction of 'work' (demonstrated commitment of hardware to the useful functioning of the network) that is effectively used to replace 'stake'

The resulting scheme is:

1. Continuous Proof of Logits (CPoL): Instead of discrete PoW blocks, we have a continuous stream of PoL work. Nodes constantly perform LLM operations, generating logits.
2. Short and medium term validated proofs of work (inference, fine tuning, training) are tracked.
3. Leader Election:
    - Leaders are elected based on a weighted average of their short and medium term validated proofs of work
4. Transaction Processing:
    - Elected leaders process and order transactions continuously, similar to Solana's approach.
    - Leaders propose transaction ordering and state updates.
5. Logit Validation:
    - We implement a parallel, ongoing process of validating logits in the Logit Pool.
    - This validation doesn't block transaction processing or finality voting but can result in retrospective slashing if invalid work is detected.
6. Dynamic Difficulty Adjustment:
    - We adjust the difficulty and requirements of Logit proofs based on network conditions, security needs, and performance targets.
7. Specialized Roles:
    - We allow nodes to specialize in different aspects: some might focus on LLM operations, others on transaction processing, and others on validation.

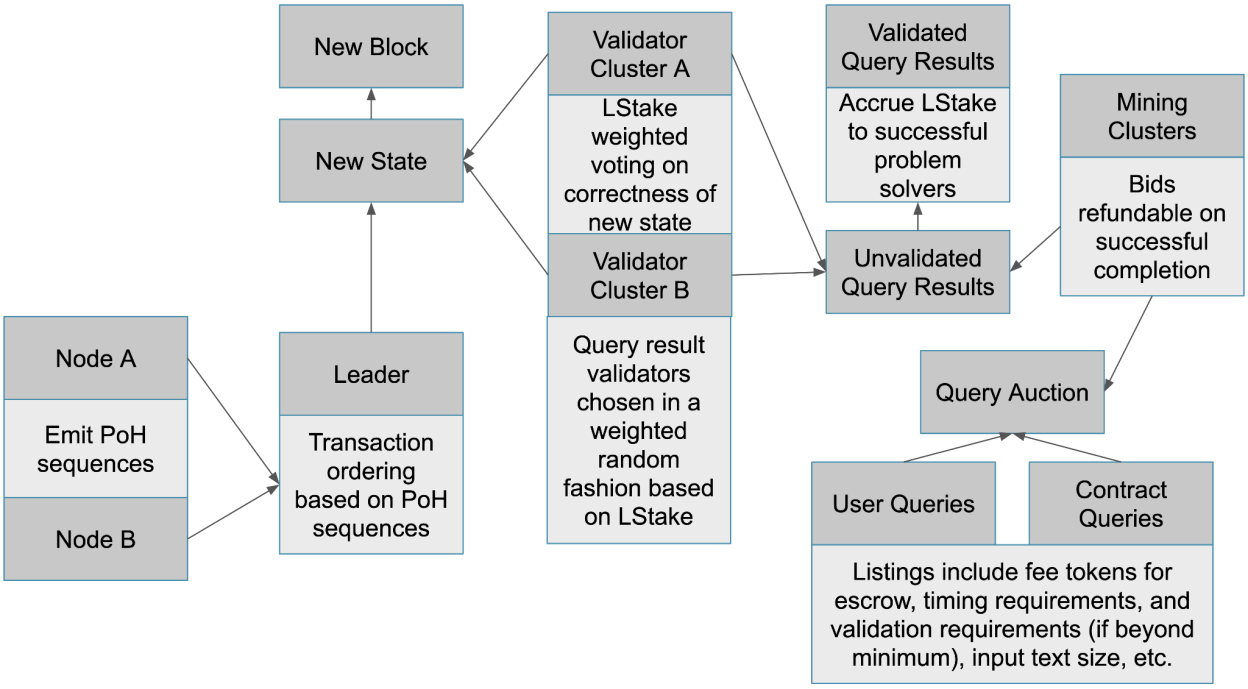
    - This leads to a more efficient division of labor while maintaining decentralization.
8. Incentive Structure:
    - We reward nodes for their contributions to the Logit Pool, successful leader terms, and participation in finality voting. Effectively, nodes get a % of the ‘inflation’ on the network within a given period of time. Solana, for instance, started with an 8% inflation rate which tapers about 15% a year. In a naive sense, if a mining pool solved 25% of the problems on our network for 1 year, it would get ¼ of 8%, or 2% of the inflation generated during the year (ignoring the other reward %s simply to illustrate the concept).

Benefits of our approach:

1. Speed: We maintain fast leader election and transaction processing.
2. Security: We incorporate work-based elements (PoL) into both leader election and finality.
3. Efficiency: We decouple logit validation from transaction processing, thus preventing PoL from becoming a bottleneck.
4. Philosophical: We directly incorporate LLM operations into the consensus mechanism, making it very purposeful.
5. Scalability: The specialization of roles allows for better scaling of different network functions.
6. Decentralization: Our approach addresses centralization issues like the Solana-Hetzner incident by incentivizing a wider distribution of nodes. Since influence is tied to active participation, it's more difficult for a small number of large stakeholders to dominate the network. This encourages a larger, more diverse set of participants, enhancing resilience against single points of failure.

This approach aims to leverage the continuous nature of Ambient's LLM operations while maintaining the high throughput and low latency that are hallmarks of Solana-inspired systems. It provides multiple layers of security (stake, logit contribution, ongoing validation) without sacrificing speed for most transactions.

What follows is a simplified diagram of how this plays out:



Continuous Proof of Logits (cPol) in Action, where LStake = “Logit Stake,” a shorthand for measured relative problem solving contribution.

Note that queries enter the system via an auction contract. Users / agents say: “I would like to have this query served within X minutes, and I am willing to pay Y,” and miners themselves bid on the queries. The miner bids are refunded if the miners complete the query within the specified period of time. Validators for miner results are chosen in a weighted random fashion based on LStake.

Sharded Model Training and Inference

A core part of the vision for Ambient involves the use of state of the art model sharding techniques to make the training and running of extremely large models tractable even for smaller / consumer grade nodes in the system. Currently, while the actual figure has been clouded in some uncertainty, it has been estimated that commercial-grade models are on the order of 8x180 billion parameters, whereas most consumer accessible models are at most 70 billion parameters.

Still, blockchain-based AI execution presents some unique challenges having to do with Internet connection speeds, node reliability and availability, and performance. Our approach to tackling these problems with generally adopt proven lessons from the PETALS architecture (https://browse.arxiv.org/html/2312.08361v1) as well as algorithmic approaches to sparsity derived from Distributed Slide (https://arxiv.org/pdf/2201.12667.pdf).

While a full treatment of these excellent papers is beyond the scope of this litepaper, we would like to highlight the core techniques that would be employed to achieve distributed fault tolerant training and inference at scale. These include:

Utilization of Sparsity: Leveraging sparsity within neural network computations to reduce the computational and communication workload. D-SLIDE does this through the SLIDE algorithm's adaptive dropout and hash-based sparse computations, significantly reducing the data that must be communicated across nodes. PETALS, through efficient data partitioning and selective computations, achieves similar reductions in network traffic. We believe that these techniques can be combined to achieve even better efficiencies. Specifically, D-SLIDE's core reliance on the SLIDE algorithm's extreme adaptive dropout and hash-based neuron selection introduces a unique approach to sparsity that's not incorporated into PETALS. This method drastically cuts down on the computations needed for each training step.

Model Parallelism: The incorporation of model parallelism as a core technique to enable training large models across distributed systems. In D-SLIDE, this is achieved by sharding the model (and corresponding hash tables) across nodes. PETALS similarly shards models but also implements fault-tolerance mechanisms to handle node failures dynamically.

Efficient Distributed Computing under Constraints: D-SLIDE focuses on low-bandwidth, few-core CPU clusters, while PETALS targets a broader range of constraints for GPU clusters

including unreliable devices and geographically dispersed resources. Both frameworks aim to harness underutilized computational resources efficiently and for Ambient's purposes employing these techniques would be key to democratizing mining on the network. In particular, we are interested in using adaptive sparsity from D-SLIDE to limit the amount of data needing synchronization across the network dramatically, alongside load balancing algorithms from PETALS.

Algorithmic over Hardware Acceleration: D-SLIDE emphasizes the importance of smart algorithmic solutions, like SLIDE, over pure hardware acceleration. This philosophy aligns with PETALS', focusing on flexible, algorithm-driven approaches to overcome hardware and network limitations. Our intention would be to merge state-of-the-art techniques to achieve competitive performance for a state-of-the-art model on relatively low spec hardware.

In the time since ChatGPT came out, we have seen a cambrian explosion in techniques to improve both training and inference efficiency driven by the extreme focus on these techniques from the entire ML community. We strongly believe that these techniques have advanced enough for a network with a design like Ambient's to become possible, where it would not have been feasible before.

Privacy Primitives

A core goal of the project would be to make both transactions and inference on the network anonymous and censorship resistant. A fulsome discussion of privacy approaches could easily occupy another whitepaper. We will simply mention that we will be deploying a range of techniques, the simplest of which involves client side personally identifiable data

obfuscation using a small local LLM, and the most complex of which will involve (when we find a performant implementation) fully or somewhat homomorphic encryption. The query auction process itself will obfuscate the origin of queries and the result consumers. Pre training data is not intended to be obfuscated, but generated synthetic data can be used to obfuscate fine tuning data.

Data Oracle Integration: Http and Bittorrent

Due to the extremely high volume of data flowing through this network (particularly during training), and the weaknesses in economic incentive schemes in preserving extremely large amounts of data for long periods of time (which leads to issues with model training reproducibility), this network would directly integrate both HTTP and BitTorrent oracles onto the nodes. Such oracles are widely used in Solana's smart contract system, and we believe that their risk profiles are well understood enough such that a careful and parsimonious implementation of such can be promoted to become a core part of the blockchain design. We make liberal use of IPFS in our auction mechanism as temporary storage.

Smart Contracts

We use the SVM as our base as we think it is a great architecture that has also attracted a fulsome developer community. As mentioned previously, we would have some extensions to the SDK to support calling the on-network foundation model and for agentic tooling.

Network and Core LLM Upgrades

The network would implement a voting system to enable network participants to vote to initiate new training campaigns that were programmatically specified. New model training runs would be able to be proposed fully in code (with reference to web and torrent URLs) and would be interleaved with model inference workloads across the network in a seamless fashion. This topic will be addressed in its own full section in future whitepapers.

Conclusion

In this Litepaper, we have advanced a radically innovative concept for a blockchain that will serve as foundational infrastructure for the agentic economy by making an evergreen human-like censorship-resistant foundation model available to the community with blazing fast verified inference speeds and highly efficient training. We hope you will support us in manifesting major techno-social change and making it Ambient.