NEB: PROOF OF WORK BY TRAINING SHARES

Whitepaper - NEB 0.1.0 - testnet

Source: docs/whitepaper.md of github.com/SilasEigen/neb
SHA-256 of the source: 8ea9e293da0c3363b15514597af1c738056aa8b7f4966a0b518a55ec0cb20f1c
In this PDF the tables of the Markdown source are written as lists.

CONTENTS

Abstract

1. Introduction

2. Related work

3. The attempt and the ticket
   3.1 Several models and convolutional networks

4. Shares and the model step

5. Verification by replay

6. Security model
   6.1 Assumptions
   6.2 Bounded influence of one step
   6.3 Cost of choosing a share
   6.4 Measured adversaries

7. Results
   7.1 Training
   7.2 The one-block delay
   7.3 Several pools
   7.4 Convolutional networks
   7.5 The public test network

8. Implementation and network

9. Assumptions and scope

10. Conclusion

References

ABSTRACT

NEB is a proof-of-work cryptocurrency whose mining attempts are training steps of small neural networks that the chain owns. An attempt takes a nonce, derives 16 samples of a dataset fixed by the chain, runs one training step of a network with 4-bit weights in exact integer arithmetic, and hashes the digest of the resulting gradients. That hash is the ticket of the attempt: below the block target it makes a block, below a target 32 times easier it makes a share. The clipped gradients of the shares of a round are summed into one bounded update of the model, applied one block later. Every node rebuilds the model by replaying the shares listed in the blocks, so the model is a function of the chain, and a block is verified by re-execution, with no proof system. On a network of one hidden layer an attempt costs about 0.5 ms of one CPU core; in simulation that network reaches 96.0% test accuracy on MNIST, and 95.7% with the one-block delay of the protocol, and on the node software 10 pools reach 89.9% after 300 blocks, the accuracy of a single pool. The public test network trains two models on scientific data, transit signals of the Kepler mission and retinal scans, which the chain moved to convolutional networks at a block height; trained by independent miners, they reached 87.0% and 67.8% about 830 blocks later.

1. INTRODUCTION

In a hash-based proof of work, the computation of the miners has no other use. Replacing the puzzle with a useful computation raises a known difficulty: when the miner chooses the inputs, nothing forces them to be useful, and the chain cannot check that the output is worth anything. NEB keeps the security of the chain in a ticket lottery, as in a hash proof of work, and makes the training of a model a by-product that the miner cannot choose away, because the chain fixes the model and the data.

The contribution of NEB lies in the protocol-level integration of deterministic neural network training with proof of work. It combines three elements, each defined by the protocol:

1. **The attempt is one deterministic integer training step.** An attempt selects 16 samples of the chain's dataset and runs forward pass, error and backward pass of the chain's network in exact integer arithmetic (inputs in 0..7, 4-bit weights, int32 sums, rounding shifts and clipping). Every correct implementation gives the same bits on any hardware, so any node can run an attempt again and obtain the same result (section 3).
2. **The digest of the gradient is part of the proof.** The clipped gradients are folded into a 32-byte digest, and the ticket is BLAKE3(digest || key). The lottery is that of a hash, and under the assumption of section 6.1 a miner cannot produce a winning ticket without computing the gradient it commits to. One computation gives the ticket and a usable gradient.

3. **The model evolves by bounded aggregation of validated shares, with a one-block delay.** A share is bound to the parent of its header, not to the header, so every attempt of every miner on that parent counts, whichever header wins. A block lists the share records of its parent's round; every node evaluates them on the model of the grandparent, clips each contribution, sums them and applies one bounded step to the model of the parent (sections 4 and 5).

As a result, the miner chooses no input except the nonce; shares come from the same attempts as blocks and cost no extra work; a node verifies by re-execution, with no zero-knowledge proof and no trusted hardware; and the influence of one contribution on one step is bounded (Theorem 1).

The normative texts are spec/shares.md (the attempt, the model step and how the chain carries shares), spec/nebhash-v2.md (the convolutional networks) and spec/node.md (blocks, transactions, chain rules and the network). Sections 3 to 5 summarise them on the network of one hidden layer, on which the experiments of sections 6.4 and 7.1 to 7.3 were made; section 3.1 gives the convolutional networks, to which the same rules apply layer by layer. Section 6 gives the security model, section 7 the main results and section 8 the implementation. The detail of every experiment is in docs/experiments.md.

2. RELATED WORK

Useful proof of work. Primecoin searches for chains of primes tied to the block hash (King, 2013). Ball et al. (2017) build proofs of work whose hardness follows from worst-case problems under stated complexity assumptions. Komargodski and Weinstein (2025) construct a proof of useful work for arbitrary matrix multiplication; there the miner chooses the matrices, and Basu (2026) reports that Pearl, a network mining on matrix products, accepts random matrices. The systematisation of Dikshit et al. (2025) observes that the useful part of such designs does not add to the security budget, and Pass (2026) models their economics. The same holds for NEB: its security comes from the ticket lottery, and the model does not add to it.

Training as the work. In Lihu et al. (2020) clients submit tasks and miners train models for them; Coin.AI (Baldominos and Saez, 2019) and the Proof of Learning of Lan et al. (2020) select the winner by a threshold or a score on a trained model; PoGO (Orlicki, 2025) proves steps of gradient descent with quantised gradients and Merkle proofs. In these designs the task or the data come from a client or from published data and the winner is chosen by a score. In NEB the chain fixes data and model, each nonce is one training step, and the shares of every miner reach the model. Proof of learning in the sense of Jia et al. (2021), where one prover logs its training for a verifier, is a different problem with known attacks (Fang et al., 2023).

Verification. NEB verifies by re-execution, which requires a deterministic computation. Integer-only training exists (Wang et al., 2020); for floating point, bit-level reproducibility across hardware is the subject of RepDL (Xie et al., 2025) and of Verde (Arun et al., 2025). Succinct proofs keep verification cheap for any computation (Parno et al., 2013; Ben-Sasson et al., 2018), including inference (Kang et al., 2022) and training (Abbaszadeh et al., 2024), at the cost of the prover's work and of a proof system in the trusted base. Interactive schemes such as TrueBit (Teutsch and Reitwiessner, 2019) and Verde delegate the check to challengers. In NEB every node checks every block.

Delayed, coarse and untrusted gradients. Stale gradients have convergence analyses (Lian et al., 2015; Dutta et al., 2018), including with compression and error feedback (Stich and Karimireddy, 2019). Error feedback keeps what a coarse update cannot express (Seide et al., 2014; Stich et al., 2018; Karimireddy et al., 2019); the bounded residual of NEB plays that role. Clipping each contribution bounds its influence, as in differentially private SGD (Abadi et al., 2016). Robust aggregation assumes a server that applies a median, a trimmed mean or a sign vote to the gradients of workers (Yin et al., 2018; Bernstein et al., 2018, 2019); federated and open collaborative training distribute the data (McMahan et al., 2017; Diskin et al., 2021). NEB has no server: every node applies the same clipped sum to the shares listed in the chain.

Shares in the chain. Pools count weak solutions outside the chain to split rewards (Rosenfeld, 2011). FruitChains (Pass and Shi, 2016), StrongChain (Szalachowski et al., 2019), Bobtail (Bissias and Levine, 2020) and P2Pool record weak solutions in or beside the chain for fair rewards, lower variance or decentralised payment. A NEB share is bound to the parent of its round, like a fruit, so that it does not depend on which block wins; its purpose is to carry a gradient.

Each ingredient has antecedents. Among the works above, none combines an attempt that is a full training step whose gradient digest is the ticket, a model that is a function of the nonces listed in the chain, and an integer step that makes the replay exact on any hardware. This reading is not a systematic survey.

3. THE ATTEMPT AND THE TICKET

The chain owns a dataset of N samples, in one file pinned by its BLAKE3 hash in the network parameters (spec/shares.md section 11.6), and a model with 4-bit signed weights. This section describes the network of one hidden layer, with W1 (128 x 784) and W2 (10 x 128), on the 60,000 MNIST training images with pixels quantised to 0..7. The initial model is derived from the genesis block id.

An attempt has a 32-byte key. In the proof of work it is the BLAKE3 hash of an 80-byte block header, so the miner changes the key through the nonce field of the header, the extranonce of the coinbase or the time. The steps of an attempt are:

1. **Samples.** The first 64 bytes of ChaCha8 keyed by the key give 16 words u_j; sample j is floor(u_j * N / 2^32).
2. **Forward pass and error.** With rshift(x, s) = floor((x + 2^(s-1)) / 2^s) and clip(v, lo, hi) = min(max(v, lo), hi), for the inputs X (16 x 784, values 0 to 7) and the targets T (+7 for the true class, -8 for the others):

   ```
   y1 = rshift(X W1^T, S1)          a1 = clip(y1, -8, 7)
   y2 = clip(rshift(a1 W2^T, S2), -8, 7)
   E2 = clip(y2 - T, -8, 7)         G2 = E2^T a1
   E1 = clip(rshift(E2 W2, SB), -8, 7) * [ -8 <= y1 <= 7 ]      G1 = E1^T X
   ```

   All sums are exact in int32. Every entry of G1 and G2 is then clipped to [-CLIP, CLIP].
3. **Digest.** The clipped gradients are folded into eight 32-bit lanes.
4. **Ticket.** ticket = BLAKE3(digest || key).

The constants are S1 = 5, S2 = 5, SB = 3, CLIP = 256 and 16 samples per attempt (spec/shares.md sections 2 and 5). One attempt costs about 3.3 million multiply-accumulates: 16 x 784 x 128 for the forward pass of layer 1, the same for G1, and about 20,000 each for the products of layer 2 and the error propagation.

The ticket depends on the weights through the gradient, so it is the ticket of the attempt on a given model state. The block of a chain whose tip is P is an attempt on the model M(P) whose key is the hash of its own header; it is valid when its ticket is below the block target. A header alone cannot be checked, because M(P) depends on the bodies of the ancestors. The difficulty rule is ASERT (spec/node.md section 6.2).

3.1 Several models and convolutional networks

A network may hold K models, each with its own dataset in one set file pinned by its hash. A block of height h works on the model of slot h mod K, and its shares train that model (spec/shares.md sections 11.2 and 11.6). A slot moves to another dataset or another network at a height fixed in the network parameters, on the same chain: its model starts again from an initial state derived from that height, and the blocks, the share lists and the coins stay as they are (spec/shares.md section 11.7).

The network of protocol 3 has two models on public scientific data. On the test network both slots moved, at blocks 8800 and 8801, from the network of one hidden layer to small convolutional networks (spec/nebhash-v2.md):

- **data** — slot 0, transit signals: Kepler DR24 threshold-crossing events (NASA Exoplanet Archive): planet candidate or false positive; slot 1, retinal scans: OCT scans of OCTMNIST at 64 x 64 (MedMNIST+): normal or abnormal
- **input** — slot 0, transit signals: a global view of 583 values and a local view of 201 values of the folded light curve; slot 1, retinal scans: 4,096 values
- **layers** — slot 0, transit signals: two 1D convolutions on each view, then two linear layers; slot 1, retinal scans: three 2D convolutions, then two linear layers
- **weights** — slot 0, transit signals: 26,000; slot 1, retinal scans: 22,280
- **forward multiply-accumulates per sample** — slot 0, transit signals: 180,800; slot 1, retinal scans: 680,000
- **one attempt, C reference, one CPU core** — slot 0, transit signals: about 2 ms; slot 1, retinal scans: about 7 ms

A layer is a convolution or a linear map with weights in -8..7 and no bias, followed by a rounding shift, a clip to -8..7 and, for some layers, a max pooling. The key, the 16 samples, the clip of the gradients, the digest, the ticket and the model step of section 4 are those of the network above, applied layer by layer with one update shift per layer.

4. SHARES AND THE MODEL STEP

Shares. An attempt whose ticket is below the share target is a share. The share target is the block target times 2^SHARE_RATIO_LOG2 = 32, so a round of attempts yields about 32 shares for each block. The attempts are those that look for a block: one that meets the share target is kept as a share, and one that also meets the block target is a block.

The model step. For each layer l, the clipped gradients of the shares of a step are summed into GS_l, and with SU_l the update shift of the layer (SU1 = 15, SU2 = 12) the state (W_l, R_l) changes as follows (spec/shares.md section 6):

```
R_l  <- R_l + GS_l
D_l   = clip(trunc(R_l / 2^SU_l), -1, 1)
R_l  <- R_l - D_l * 2^SU_l
R_l  <- clip(R_l, -2^SU_l, 2^SU_l)
W_l  <- clip(W_l - D_l, -8, 7)
```

The residual R_l carries the small gradients from one step to the next, which a plain rounding would lose (error feedback). It is bounded by 2^SU_l, so a weight moves by at most one unit per step. The sums are exact in int32 for up to 8,388,479 shares, and a block lists at most 256.

State. The state (W1, W2, R1, R2) is a pure function of the genesis block id, the dataset and the share records listed in the blocks. A share is a record of 40 bytes, body_root (32) || LE32(time) || LE32(nonce). Its share header is the 80-byte header with the parent, the bits of the round and the fields of the record, and its key is the BLAKE3 hash of that header (spec/shares.md section 11.2).

Delay of one block. Let B be a block with parent P and grandparent PP. The shares of the round of P are the attempts of any miner on any header whose parent is PP (the header of P, competing blocks, and the headers that other pools build with their own coinbase) whose ticket on M(PP) meets the share target. B lists their records; their gradients are evaluated on M(PP), summed, and applied by one step to M(P), which gives M(B). A share trains the model one block after it was found, with a gradient one step old. The update shifts are chosen for this delay (section 7.2). Because a share does not depend on the header that wins the round, a node that keeps the records its miners find and relays them to its peers lists the shares of all the pools that reach it (spec/node.md section 8.2).

5. VERIFICATION BY REPLAY

A node verifies a block by replaying its attempts; nothing else about the work is trusted. One evaluation takes about 0.5 ms of one CPU core in the C reference and gives the ticket and the clipped gradient at once. For a block B with parent P, a node:

1. checks the header and the shape of the body and of the share list (at most 256 records, strictly increasing);
2. evaluates the ticket of B on M(P) with the key BLAKE3(header), and rejects it above the block target;
3. evaluates the ticket of every listed share record on M(PP), rejects any above the share target of P, and sums the clipped gradients;
4. applies the model step to M(P) and keeps M(B).

A block costs 1 + k evaluations: 33 for the expected 32 shares and at most 257, that is about 17 ms and at most 130 ms of one core. Cheap checks come first and evaluations last, and each peer has a budget of ticket evaluations per second (spec/node.md section 8.1), so a block that costs its sender nothing cannot cost the receiver more than that budget.

A node keeps the dataset file (47,100,008 bytes) and the states of the last 101 levels of its best chain and of the side branches it has checked. A state holds the packed weights (50,816 bytes) and the residuals (406,528 bytes). A missing state is rebuilt by replaying the blocks from genesis; a reorganisation of at most 60 blocks finds its states among the kept ones. A miner does not hold the residuals: the work service sends it the packed weights of M(P) and the header to roll.

With the convolutional networks of section 3.1 an evaluation takes about 2 ms (transit signals) or 7 ms (retinal scans), and the shares that a block lists were mined on the model of the block before it. A block with the expected 32 shares costs about 70 ms of one core when its shares are transit attempts and about 230 ms when they are retinal attempts, and at most 1.8 s with 256 shares. The packed weights take 13,000 and 11,140 bytes.

6. SECURITY MODEL

6.1 Assumptions

NEB inherits the model of a hash proof of work: the security of the chain comes from the ticket lottery and the cumulative work of the best chain, under an honest majority of the hash rate. Its specific assumption concerns the attempt.

Assumption A (no shortcut). Model BLAKE3 as a random oracle. Computing the digest of an attempt for a given nonce and model needs a forward and a backward pass, with no strategy that costs much less.

Under Assumption A an adversary that makes q evaluations finds a ticket below a target tau with probability at most q * tau / 2^256, as in a hash proof of work, and the chance that an attempt wins is the same for every miner whatever data it works on. Assumption A plays the role that the absence of shortcuts in the hash function plays for Bitcoin. It is an assumption, stated here so that it can be examined: no proof excludes a cheaper way to obtain the digest, and no review outside the project has examined it yet (section 9).

6.2 Bounded influence of one step

Theorem 1 (one-step influence). Fix the state (W, R) and two lists of n shares that differ in at most k shares, which the adversary may choose after any number of nonce trials. Then for every weight, the steps D' and D'' satisfy |D' - D''| <= min(2, ceil(2kC / 2^SU)), with C = CLIP.

Proof. Each share that differs changes an entry of the sum by at most 2C, since both contributions lie in [-C, C], so the sums differ by at most 2kC and the residuals R + G by the same amount. For a >= b, trunc(a) - trunc(b) <= ceil(a - b): if both are non-negative this is floor(a) - floor(b) <= ceil(a - b), if both are negative it is the same bound for ceil, and if a >= 0 > b the difference is floor(a) - ceil(b) <= a - b. Dividing by the power of two 2^SU keeps the bound, so |trunc(R'/2^SU) - trunc(R''/2^SU)| <= ceil(2kC / 2^SU). The clamp to [-1, 1] does not increase a difference and bounds it by 2. The final clip to [-8, 7] does not increase it either. QED.

With C = 256 and k = 8 of 32 shares, 2kC = 4096: at most 1 per weight and per step in layer 1 (SU1 = 15) and at most 1 in layer 2 (SU2 = 12), whatever the adversary tries. A producer can choose the whole list of its block; the bound is then 1 level in layer 1 (up to 64 shares) and 2 in layer 2. The theorem holds for each layer of the convolutional networks with the update shift of that layer, 12 to 16 (spec/nebhash-v2.md section 3): with k = 8 the bound is 1 in every layer.

Theorem 2 (replay). The model after T steps is a function of the initial model, the dataset and the sets of share records of the T steps. It does not depend on the order of the shares inside a step, or on the machine.

Proof. Each contribution is a deterministic function of the weights, the dataset and the nonce, computed in exact integer arithmetic. The sums are integers with |sum| <= nC < 2^31, so they do not overflow, and integer addition is associative and commutative. The residual update and the clips are deterministic functions of those sums. QED.

6.3 Cost of choosing a share

A candidate share is not a free choice of a nonce: it has to meet the share target, which takes about 2^15 attempts at the genesis target (2^20 for a block). An adversary that picks among M valid shares for each of its shares spends M / 32 block-equivalents of work on each. Gradients cannot be crafted either: an attempt draws real samples of the chain's dataset, so no input is chosen by the miner.

6.4 Measured adversaries

Theorem 1 bounds one step from a given state. Over many steps the honest gradients depend on the weights, so the effect of an adversary is measured (docs/experiments.md, sections 2 and 5):

- **Grinding against the gradient.** An attacker with 8 of the 32 shares of every step, picking the worst of 128 nonces for each, lowers the accuracy after 300 steps by 0.1 point, within run-to-run noise; with 16 of 32 shares and 64 nonces each, by 0.9 point.
- **Targeted poisoning.** An attacker steering half of the shares toward one class with 4,096 nonces each lowers the accuracy by 1.7 points after 300 steps; in the chain each candidate would cost a valid share.
- **Deviating pools on the node software.** One pool among 10 that withholds its shares, or reports only the quarter that push the model most toward one class, leaves the predicted share of that class at 9.8 to 10.2% (class 3 is 10.2% of the samples) and the accuracy within the spread of honest runs. The largest effect is a censoring producer with a third of the attempts: blocks list 13 shares on average instead of 30, and the accuracy trails by about 3 points after 200 blocks and 1.8 points after 300. The training slows down and shows no bias.

These adversaries are fixed strategies, measured on the network of one hidden layer. No proof bounds an adaptive adversary over many steps, and selfish mining and the economics of pools are outside this analysis (section 9).

7. RESULTS

The results of sections 7.1 to 7.4 come from simulations (tools/) and from measurements on the node software with simulated miners on one machine. Methods, commands, all tables and their raw outputs are in docs/experiments.md. Section 7.5 reports the public network with independent miners.

7.1 Training

The simulation trains the 784-128-10 network on MNIST with the model step of section 4 and evaluates it on the 10,000 test images. After 9,600, 32,000 and 96,000 share nonces it reaches 90.2%, 93.2% and 96.0% test accuracy. Clipping the gradients at CLIP = 256 costs nothing measurable (90.2% after 300 steps with and without clipping).

7.2 The one-block delay

The update shifts are chosen for the delay of the protocol. Final test accuracy after 96,000 shares, 32 shares per block, mean [min-max] over 4 seeds, with SU1 = 15 and SU2 = 12:

- **Delay (blocks): Accuracy** — 0: 95.9 [95.8-96.0]; 1: 95.7 [95.7-95.8]; 2: 95.4 [95.2-95.7]; 4: 93.7 [91.7-94.4]

The chain's case is one block: 95.7%, 0.2 point below no delay. Faster shifts (SU1 = 13, SU2 = 10) reach 97.3% without delay but fall to 61.2% with one block of delay, which is why the protocol uses the slower step.

7.3 Several pools

The model learns from the shares that blocks list. Binding a share to the parent of its round, not to the winning header, is what lets the work of every pool reach the model. With a share bound to the winning header, only the pool that built that header could list the round's shares, and the model would receive sum_i f_i^2 (1 + 31 f_i) shares per block for pool shares f_i: 0.41 with 10 equal pools.

The bench runs nodes of this repository joined over TCP, each with a simulated miner using the work service, with blocks every 2 s and the testnet's ratio of 32 between share and block targets. The nodes list 29.8 to 32.0 shares per block with 1, 2, 5, 10 and 20 equal pools. Accuracy after 300 blocks on the first 10,000 samples of the network dataset:

- **pools: 1** — share bound to the winning header: -; share bound to the parent (this protocol): 89.7%
- **pools: 10** — share bound to the winning header: 36.2%; share bound to the parent (this protocol): 89.9%
- **pools: 20** — share bound to the winning header: 24.4%; share bound to the parent (this protocol): 89.4%

With 50 or 100 ms of delay on every link, blocks still list 30 to 35 shares. On Fashion-MNIST, with constants chosen for MNIST, 10 pools and one pool reach 82.0% and 80.7% after 1,000 blocks.

7.4 Convolutional networks

A simulation of the step rule on the two datasets of section 3.1, with 32 shares of 16 samples per step, compares the network of one hidden layer, the convolutional network under the same rule, and the same convolutional network trained with ordinary floating-point training (spec/nebhash-v2.md section 7):

- **transit signals (2,333 test signals)** — one hidden layer, chain rule: 88.3% at 2,000 steps; convolutional, chain rule: 90.8% at 3,000 steps; convolutional, float training: 91.6%
- **retinal scans, normal or abnormal (500 test scans, balanced)** — one hidden layer, chain rule: about 78%; convolutional, chain rule: 81 to 83% from 5,000 to 8,000 steps; convolutional, float training: 92.0%

Each figure comes from one run, and runs differ by about 1 point on the transit signals and 2 points on the retinal scans. On the transit signals the integer rule with 4-bit weights is within 1 point of float training; on the retinal scans it is about 10 points below.

7.5 The public test network

The test network of protocol 3 started on November 7, 2026 with independent miners and pools. The figures below are read from its chain at block 9,644 (November 14) and from the models that a node rebuilds from it; the accuracies are measured on samples kept out of the dataset file.

- **block spacing** — measured: 58.9 s on average, for a target of 60 s
- **addresses that mined at least one block** — measured: 76 (a pool counts as one)
- **shares listed per block** — measured: 20.6 on average, median 12, for 32 expected; 4.7% of the blocks list none
- **transit signals, one hidden layer (blocks 0 to 8,799)** — at its first block: 14.1%; measured later: 89.0% at block 8,798
- **retinal scans with 4 labels, one hidden layer (blocks 0 to 8,800)** — at its first block: 9.2%; measured later: 53.6% at block 8,799, 56.0% at best
- **transit signals, convolutional (from block 8,800)** — at its first block: 42.8%; measured later: 87.0% at block 9,632
- **retinal scans, normal or abnormal, convolutional (from block 8,801)** — at its first block: 47.8%; measured later: 67.8% at block 9,632, 70.0% at best

The models learn from the shares of miners that do not coordinate, and every node that replays the chain obtains the same weights. Blocks list fewer shares than the 32 expected, so a step of the chain carries less than a step of the simulations of sections 7.1 to 7.4; the convolutional models had made about 420 steps each at block 9,632.

8. IMPLEMENTATION AND NETWORK

The repository holds the C reference of the attempt and of the model step (pow/ref), an independent Python oracle written from the specification (pow/oracle), and a Rust workspace with the node, the wallet and the CPU and GPU miners (node/). The node evaluates tickets with the C reference only. The miners may use optimised engines (AVX-512 on the CPU, tensor-core kernels on NVIDIA GPUs) that compute the same tickets faster; they are off by default, self-tested against the reference, and every nonce they report is evaluated again with the reference before it is sent (docs/mining.md). On the measured processor the sparse AVX-512 engine is 30 to 46 times faster than the reference built with MSVC, on all threads. For the convolutional networks the CPU miner runs the reference code, and the GPU miner has a CUDA kernel that is self-tested against the reference at start.

The implementation is checked bit for bit: the model step of the node agrees with an independent 64-bit step on all 77,312,098,320 entries of the domain of the consensus, the AVX-512 engines agree with the reference on 1,673,728 evaluations, and recorded chains give the same final model hashes in the node and in the Python oracle (docs/experiments.md, section 7).

The network has a block every 60 s on average (ASERT with a half-life of 600 s), 20 coins per block with a halving every 450,000 blocks, and a UTXO money layer with Ed25519 signatures and BLAKE3 hashing, with no virtual machine. Of each reward, 5% goes to the project in a second coinbase output that the rules make mandatory (spec/node.md section 6.3); the miner gets the rest and the fees.

The main network starts with the two convolutional networks at block 0, each from an untrained model. Its genesis block pays each lock a tenth of the balance it held on each of the two test networks, as coinbase outputs that mature after 100 blocks; the lists are rebuilt from the published chains (docs/carry-over.md). Nodes find each other through public relays and then through the addresses their peers reached (spec/node.md section 8.4). Parameters and operation are in docs/mainnet.md, and those of the test network in docs/testnet.md.

9. ASSUMPTIONS AND SCOPE

1. **Assumption A.** Like every proof of work, NEB rests on an assumption about its work: no strategy produces a winning ticket for less than an evaluation of the attempt. What supports it today is the construction (the ticket hashes a digest of every entry of the gradient) and the checks of section 8. It has no proof and no external review.
2. **Many steps.** Theorem 1 bounds one step, and section 6.4 measures fixed strategies over 300 steps. An adaptive adversary over many blocks, one that optimises the loss of the model with look-ahead, selfish mining and the economics of pools are not covered by a proof or a measurement.
3. **Incentives.** A share is not paid by the protocol; the number that reaches the model depends on the nodes that relay and list them, and section 7.5 gives the number measured on the public network. Studies of paid variants are in docs/experiments.md, section 6.

4. **Forks.** The model of a block is a function of its own chain, so a reorganisation replaces the model with that of the new chain, and a node accepts none deeper than 60 blocks. The effect of orphaned blocks on learning was not measured apart from the curves of section 7.5.
5. **Usefulness.** The models are small classifiers on public data, and their usefulness depends on that choice. The security of the chain and the reward do not depend on it. A slot moves to another dataset or a larger network at a height fixed in the network parameters (section 3.1), with update constants tuned for it.
6. **Hardware.** NEB does not claim ASIC resistance: an attempt is millions of small integer multiply-accumulates and one hash, and dedicated hardware may beat a GPU. Energy use and the economic value of the coins were not measured.

10. CONCLUSION

NEB makes the training of a small neural network the work of a proof-of-work chain. Each attempt is one deterministic integer training step, the digest of its gradient is the ticket, and the shares listed in the blocks drive a bounded update that every node replays, so the model is a function of the chain. Theorem 1 bounds what one contribution can do to one step. In simulation the network reaches 96.0% on MNIST and 95.7% with the delay of the protocol; on the node software, 10 pools train the model as well as one. On the public test network, independent miners trained two models on scientific data, and the chain moved both to convolutional networks at a block height without a new genesis. What remains outside this document is a bound over many steps, incentives to list and relay shares, and an external review of the protocol and of Assumption A.

REFERENCES

- J. O'Connor, J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn. BLAKE3: one function, fast everywhere. Specification, 2020.
- D. J. Bernstein. ChaCha, a variant of Salsa20. 2008.
- D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, B.-Y. Yang. High-speed high-security signatures. Journal of Cryptographic Engineering 2, 2012 (Ed25519).
- M. Lundeberg. The aserti3-2d difficulty adjustment algorithm (absolutely scheduled exponentially rising targets).
- Y. LeCun, C. Cortes, C. J. C. Burges. The MNIST database of handwritten digits.
- NASA Exoplanet Archive. Kepler Q1-Q17 DR24 threshold-crossing events table.
- C. J. Shallue, A. Vanderburg. Identifying exoplanets with deep learning: a five-planet resonant chain around Kepler-80 and an eighth planet around Kepler-90. The Astronomical Journal 155(2), 2018 (the global and local views of a folded light curve).
- J. Yang, R. Shi, D. Wei, Z. Liu et al. MedMNIST v2: a large-scale lightweight benchmark for 2D and 3D biomedical image classification. Scientific Data 10, 41, 2023.
- D. S. Kermany, M. Goldbaum, W. Cai et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell 172(5), 2018 (the OCT scans).

- H. Xiao, K. Rasul, R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.
- F. Seide, H. Fu, J. Droppo, G. Li, D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. Interspeech 2014.
- S. U. Stich, J.-B. Cordonnier, M. Jaggi. Sparsified SGD with memory. NeurIPS 2018.
- S. King. Primecoin: cryptocurrency with prime number proof-of-work. 2013.
- M. Ball, A. Rosen, M. Sabin, P. N. Vasudevan. Proofs of useful work. IACR ePrint 2017/203, 2017; and Proofs of work from worst-case assumptions, CRYPTO 2018.
- I. Komargodski, O. Weinstein. Proofs of useful work from arbitrary matrix multiplication. arXiv:2504.09971, 2025.
- A. Basu. The usefulness gap in proof-of-useful-work: an empirical study of Pearl's cuPOW protocol. arXiv:2606.04819, 2026.
- P. Dikshit, A. Emami, J. Sedlmeir, G. Fridgen. SoK: is proof-of-useful-work really useful? IACR ePrint 2025/1814, 2025.
- R. Pass. The economics of proof-of-useful-work. arXiv:2606.06700, 2026.
- A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics et al. A proof of useful work for artificial intelligence on the blockchain. arXiv:2001.09244, 2020.
- A. Baldominos, Y. Saez. Coin.AI: a proof-of-useful-work scheme for blockchain-based distributed deep learning. Entropy 21(8), 2019 (arXiv:1903.09800).
- Y. Lan, Y. Liu, B. Li. Proof of Learning (PoLe): empowering machine learning with consensus building on blockchains. arXiv:2007.15145, 2020.
- J. I. Orlicki. PoGO: a scalable proof of useful work via quantized gradient descent and Merkle proofs. arXiv:2504.07540, 2025.
- H. Jia, M. Yaghini, C. A. Choquette-Choo, N. Dullerud et al. Proof-of-learning: definitions and practice. IEEE Symposium on Security and Privacy 2021 (arXiv:2103.05633).
- C. Fang, H. Jia, A. Thudi, M. Yaghini et al. Proof-of-learning is currently more broken than you think. IEEE European Symposium on Security and Privacy 2023.
- M. Wang, S. Rasoulinezhad, P. H. W. Leong, H. K.-H. So. NITI: training integer neural networks using integer-only arithmetic. arXiv:2009.13108, 2020.
- P. Xie, X. Zhang, S. Chen. RepDL: bit-level reproducible deep learning training and inference. arXiv:2510.09180, 2025.
- A. Arun, A. St. Arnaud, A. Titov, B. Wilcox, V. Kolobaric, M. Brinkmann, O. Ersoy, B. Fielding, J. Bonneau. Verde: verification via refereed delegation for machine learning programs. arXiv:2502.19405, 2025.
- B. Parno, C. Gentry, J. Howell, M. Raykova. Pinocchio: nearly practical verifiable computation. IEEE Symposium on Security and Privacy 2013 (IACR ePrint 2013/279).
- E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. IACR ePrint 2018/046, 2018.
- D. Kang, T. Hashimoto, I. Stoica, Y. Sun. Scaling up trustless DNN inference with zero-knowledge proofs. arXiv:2210.08674, 2022.
- K. Abbaszadeh, C. Pappas, J. Katz, D. Papadopoulos. Zero-knowledge proofs of training for deep neural networks. IACR ePrint 2024/162, 2024.
- J. Teutsch, C. Reitwiessner. A scalable verification solution for blockchains. arXiv:1908.04756, 2019.

- X. Lian, Y. Huang, Y. Li, J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. NeurIPS 2015 (arXiv:1506.08272).
- S. Dutta, G. Joshi, S. Ghosh, P. Dube et al. Slow and stale gradients can win the race: error-runtime trade-offs in distributed SGD. arXiv:1803.01113, 2018.
- S. U. Stich, S. P. Karimireddy. The error-feedback framework: better rates for SGD with delayed gradients and compressed communication. arXiv:1909.05350, 2019.
- S. P. Karimireddy, Q. Rebjock, S. U. Stich, M. Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. ICML 2019 (arXiv:1901.09847).
- J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, A. Anandkumar. signSGD: compressed optimisation for non-convex problems. ICML 2018 (arXiv:1802.04434).
- J. Bernstein, J. Zhao, K. Azizzadenesheli, A. Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. ICLR 2019 (arXiv:1810.05291).
- D. Yin, Y. Chen, K. Ramchandran, P. Bartlett. Byzantine-robust distributed learning: towards optimal statistical rates. ICML 2018 (arXiv:1803.01498).
- H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. AISTATS 2017 (arXiv:1602.05629).
- M. Diskin, A. Bukhtiyarov, M. Ryabinin, L. Saulnier et al. Distributed deep learning in open collaborations. arXiv:2106.10207, 2021.
- M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan et al. Deep learning with differential privacy. ACM Conference on Computer and Communications Security 2016.
- M. Rosenfeld. Analysis of Bitcoin pooled mining reward systems. arXiv:1112.4980, 2011.
- R. Pass, E. Shi. FruitChains: a fair blockchain. IACR ePrint 2016/916, 2016 (PODC 2017).
- P. Szalachowski, D. Reijsbergen, I. Homoliak, S. Sun. StrongChain: transparent and collaborative proof-of-work consensus. arXiv:1905.09655, 2019.
- G. Bissias, B. N. Levine. Bobtail: improved blockchain security with low-variance mining. NDSS 2020.
- Bitcoin Wiki. P2Pool. en.bitcoin.it/wiki/P2Pool.