COGITO: a neural network inside one transaction

COGITO is a Solana-compatible chain we run ourselves. Not Solana mainnet. A fork we run. Its validators carry the vector instructions of the SIMD-0714 draft's prototype, and with them a small neural network runs inside a single transaction. Every figure below is a measurement we recorded on named hardware with the command that produced it in our source tree (How to reproduce says which of those commands the public repository can rerun); none is copied from elsewhere.

WHAT RUNS TODAY

The node is Agave v4.5.0-alpha.4 (upstream commit ccddd3d), built unchanged except for six patch files: the SBPF virtual machine is replaced by the SIMD-0714 prototype (temporalxyz/sbpf, branch rodata-pair, commit 585f893, the sbpf version this Agave release pins), the vector instructions are switched on, the aligned memory mapping they need is forced on, the deploy environment no longer registers the vector syscall twice, two lines follow an API change in the prototype, and the smallest ledger a validator may keep is lowered from 50 million to 100,000 shreds (0006-ledger-floor.patch), so each validator's ledger stays inside a fixed disk share. Four validators of this build run as one private network in three emulated locations, two in Frankfurt, one in New York and one in Tokyo, with one-way delays of 40 to 63 ms between them (the distances between those cities), each holding an equal stake. Every validator is ours, all run on one server, and every one runs on a CPU with AVX-512. Transactions reach the network through one RPC gateway with per-address limits.

A mind is an Int8 network with 256 inputs, 128 hidden units and 16 outputs: h = clamp(relu(W1 x) >> 8, 0, 255), y = W2 h. Today's compilers emit neither the new vector instructions nor the prototype's SBPF v3 layout, so each mind is machine code and an ELF file written by a small generator. Every mind exists twice with the same weights: the vector program, and a scalar twin that uses only today's instruction set. A thought is one transaction whose instruction data is the 256 input bytes; the program logs its 16 outputs as one Program data: line, and the stock Solana CLI reads that line and the compute units (CU, Solana's measure of work) with solana confirm -v <signature> --url https://rpc.cogito.run.

The digit mind is this network trained on MNIST, the public handwritten-digit set, scaled to 16 x 16 grayscale, trained on a CPU with the programs' exact integer arithmetic in the forward pass. Its answer is the largest of outputs 0 to 9. The chain's logged outputs equal the outputs computed off chain from the same weights: we check this on a fixed set of twelve inputs for both programs (26 of 26 transactions equal, including the benchmark pair), and a wrong input length is refused by the program.

MEASURED

Host for every row: one virtual machine with 28 virtual CPUs and 62 GB of memory on an AMD Ryzen 9 9950X (a 16-core processor with AVX-512). Chain rows: our network of 4 validators with equal stake in 3 emulated locations (two in Frankfurt, one in New York, one in Tokyo), the build above, the prototype's CU schedule. Wall-time rows: the prototype VM

with our network's loader configuration, p50 of 2,000 JIT runs or 50 interpreted runs per sample. Every row is p50 of the record's runs.

FIGURE	VALUE	CONDITIONS
mind-digit-vector-program-cu	1,700 CU	digit mind, vector program, one MNIST test digit, includes 264 CU for logging 16 outputs
mind-digit-scalar-program-cu	349,958 CU	digit mind, scalar twin, same input and weights (206x the vector program)
mind-digit-vector-tx-cu	1,700 CU	whole transaction of a vector thought, no ComputeBudget instruction
mind-digit-vector-site-tx-cu	1,850 CU	the same thought as the site sends it, with setComputeUnitLimit(10,000) (150 CU)
mind-digit-scalar-tx-cu	350,108 CU	whole transaction of the scalar twin, with setComputeUnitLimit (150 CU)
mind-transfer-tx-cu	150 CU	comparison: a plain System Program transfer on the same network
mind-digit-vector-jit-us	0.822 µs	digit mind, vector program, JIT
mind-digit-scalar-jit-us	196.6 µs	digit mind, scalar twin, JIT (239x)
mind-digit-vector-interp-us	66.6 µs	digit mind, vector program, interpreter
mind-digit-scalar-interp-us	724.1 µs	digit mind, scalar twin, interpreter (about 11x)
mind-digit-accuracy	98.01 % correct	MNIST test set, 10,000 digits never used in training, exact integer arithmetic of the programs
mind-largest-vector-hidden	19,920 hidden units	widest vector mind 256 -> H -> 16 that deploys and runs in one transaction; bounded by the 10 MiB program account, not by CU (5,418,240 multiply-adds)
mind-largest-vector-program-cu	221,032 CU	that mind's thought, with random weights; outputs equal the reference
mind-largest-scalar-hidden	512 hidden units	widest scalar twin that runs in one transaction; bounded by the 1,400,000 CU limit (139,264 multiply-adds)
mind-largest-scalar-program-cu	1,398,524 CU	that scalar mind's thought
net-slot-pace	4.99 slots/s	the network's slot pace through the gateway over 10 s, 10 samples, on the network as it runs (p99 5.09)
net-confirm-fra	249 ms	a transfer from send to confirmed, client in the Frankfurt location, gateway in Frankfurt (p99 310 ms)

FIGURE	VALUE	CONDITIONS
net-confirm-nyc	468.5 ms	the same, client in the New York location (p99 581 ms)
net-confirm-tyo	489 ms	the same, client in the Tokyo location (p99 763 ms)
net-load-sustained	1,200 visitors	simulated visitors through the gateway, each sending one vector thought every 8 s and reading the slot every 2 s; the last level (doubling from 75, 60 s each) with at least 95 % confirmed and p99 at most 5 s. At 2,400 the gateway ran out of memory first; the validators kept confirming
mind-trial-vector-program-cu	1,531 CU	the benchmark program: the same 256-128-16 shape with fixed pseudo-random Int8 weights and a fixed input built into the program, logging outputs 0 to 3 and the sum of all 16 in one line; it measures cost, not accuracy (the shape of the network in the SIMD-0714 proposal's table), reproduced on our network
mind-trial-scalar-program-cu	349,771 CU	the benchmark's scalar program (228x the vector program)
mind-trial-vector-tx-cu	1,681 CU	the benchmark's vector transaction, with setComputeUnitLimit
mind-trial-scalar-tx-cu	349,921 CU	the benchmark's scalar transaction, with setComputeUnitLimit
mind-trial-vector-jit-us	0.762 µs	the benchmark's vector program, JIT
mind-trial-scalar-jit-us	200.1 µs	the benchmark's scalar program, JIT (263x)
mind-trial-vector-interp-us	65.2 µs	the benchmark's vector program, interpreter
mind-trial-scalar-interp-us	719.7 µs	the benchmark's scalar program, interpreter (about 11x)

The SIMD-0714 document reports, for an Int8 MLP 256→128→16, 490,048 CU scalar against 2,109 CU vector (232x) and 361 µs against 1.17 µs under the JIT (308x), measured on an AMD EPYC 9534. That is another machine and other kernels; it is shown beside ours, not as the same conditions. Our ratios are of the same order: 206x to 228x in CU and 239x to 263x in JIT time.

CU is the prototype's price list, not time. Under the JIT, the digit mind's vector program and its scalar twin both take about 0.57 ns per CU of their own instructions, so on this host the 1-CU price of a vector instruction is consistent with what it costs to run. Under the interpreter it is not: there the gain falls to about 11x.

HOW TO REPRODUCE

Every figure is a rig.py record: the command below is run several times and its last line is the figure. The public repository holds the generator, the reference, the digit mind's weights, the tests and our six patch files: with it alone you can rebuild both programs of the digit mind byte for byte, check their outputs against the reference, and read a recent thought on our network (within the history window under What does not work yet) with the stock CLI. The benchmark programs come from a generator that is not public. The VM bench (chain/minds/vm/bench.sh) is in the public repository; rig.py, chain/minds/measure_one.py, chain/minds/measure-largest.sh and the network scripts under chain/ops/ are in our source tree only.

- Chain CU: MINDS_RPC=https://rpc.cogito.run python3 rig.py run --name mind-digit-vector-program-cu --runs 3 -- python3 chain/minds/measure_one.py think digit-vector program (the same with digit-scalar, and trial-vector, trial-scalar for the benchmark program, and program or tx; measure_one.py transfer for the transfer).
- Wall time: python3 rig.py run --name mind-digit-vector-jit-us --runs 5 -- bash chain/minds/vm/bench.sh <sbpf work dir> chain/minds/build/digit-vector.so chain/minds/fixtures/inputs.bin jit (interp for the interpreter; chain/minds/vm/build.sh prepares the prototype crate with the harness).
- Accuracy: python3 rig.py run --name mind-digit-accuracy --runs 3 -- python3 chain/minds/digit/accuracy.py chain/minds/data chain/minds/digit/weights.bin (chain/minds/fetch_mnist.sh fetches MNIST and checks its SHA-256 sums).
- The programs: python3 chain/minds/gen_mind.py elf-vector chain/minds/digit/weights.bin digit-vector.so and elf-scalar for the twin; python3 chain/minds/reference.py <weights> <input> prints the outputs and the exact Program data: line the chain must log.
- Largest minds: MINDS_RPC=https://rpc.cogito.run bash chain/minds/measure-largest.sh searches the widest hidden layer that deploys and runs in one transaction, vector and scalar, with random weights, and records its CU.
- With the stock CLI, against our public RPC, for a transaction the node still holds (see the history window below): solana confirm -v <signature> --url https://rpc.cogito.run prints the mind's log lines, the consumed N of M compute units line and Compute Units Consumed. Only the URL differs from a mainnet query. With web3.js: getTransaction(signature, { maxSupportedTransactionVersion: 0 }), meta.logMessages and meta.computeUnitsConsumed.
- Network rows: rig.py run --name net-slot-pace --runs 10 -- bash chain/ops/measure/slot-pace.sh https://rpc.cogito.run 10 reads the slot twice 10 s apart through the gateway (our records ran on that same gateway from the network's host, not through the public front); net-confirm-<location> runs node chain/ops/measure/confirm.cjs https://rpc.cogito.run <payer> from a client inside that location, timing a transfer from send to confirmed by polling its status over HTTP. net-load-sustained runs bash chain/ops/load/ladder.sh <project> sustained, the load ladder whose levels and failures are written out on the status page.
- The node: Agave at tag v4.5.0-alpha.4 with our six patch files applied, building the validator and the fork's own solana CLI (needed only to deploy SBPF v3 programs).

SOURCES

- SIMD-0714 "SIMD", solana-foundation/solana-improvement-documents pull request 714: document created 2026-10-28, status Draft, pull request opened 2026-10-30 and open. Authors: cavey, frank, ben, Philip Taffet (Jump), Dean Little (Blueshift), Joe Caulfield (Anza). We are not among them and the proposal is not adopted: we ran its prototype as it is and measured it.
- Its prototype: temporalxyz/sbpf, branch rodata-pair, commit 585f893 (on anza-xyz/sbpf 9f674b4, version 0.26.0).
- Our upstream: anza-xyz/agave, tag v4.5.0-alpha.4 (commit ccddd3d).
- Objections from the proposal's review, each with our answer. ripatel-fd (2026-11-04): "The idea of vector instructions makes sense to me." and "The performance claims are basically unsubstantiated". Our answer is the table above: measured with conditions, in the JIT and in the interpreter, with the commands. nagisa (2026-10-30): "Are we now absolutely sure solana is x86 only and x86 forevermore?" and that the design "is also hugely incompatible with efficient compilation and especially interpreted execution". Our interpreter rows agree: about 11x instead of over 200x. drubin-fd (2026-11-04): "this SIMD should absolutely not include so many instructions". We run the prototype unchanged and do not amend the proposal; our minds use eight of its operations, and the benchmark program nine.
- MNIST handwritten digits (LeCun, Cortes, Burges), training and test sets.
- Solana fees and the per-transaction compute limit of 1,400,000 CU: solana.com/docs/core/fees.
- Another road, for contrast: proving inference that ran elsewhere, as zkLLM (arXiv 2404.16109), opML (arXiv 2401.17555) and TOPLOC (arXiv 2501.16007) do. They make a model's answer checkable without running it on chain; COGITO runs the network on chain, which only works for networks this small.

WHAT DOES NOT WORK YET

- Every validator needs AVX-512. The prototype's loader refuses to start without it, so this network cannot accept validators on other hardware.
- The prototype's CU prices hold under the JIT on this host. In the interpreter a vector operation costs far more time per CU than an ordinary instruction.
- Minds are small: one fixed shape, 256 inputs, 128 hidden units, 16 outputs, Int8, no biases. The widest that fits one transaction is about 5.4 million multiply-adds, and it is bounded by program size; a deeper network or a language model does not fit. The SIMD-0714 proposal is a draft with open objections; it may change or never be adopted.
- No compiler emits the vector instructions: minds are written by our generator, not built from Rust.
- The stock Solana CLI 3.0 reads what the node still holds but cannot deploy an SBPF v3 program to it; deploys use the fork's own CLI on our side.
- Public load is bounded by our one RPC gateway: it held 1,200 simulated visitors and ran out of memory at 2,400 (its cap was then 256 MB; it is now 1 GB, not yet tested at 2,400), before the validators showed any limit. The load came from the same server, not from the internet.

- Transaction history is short. To keep the network inside a fixed disk budget, every validator keeps only its newest ledger: the main RPC node 4,955,731 shreds, the other three 1,321,528 each. At about 96 shreds per quiet slot and 5 slots a second that is about 3 hours of transactions on the main RPC node and about 45 minutes on the second RPC node, which answers while the main one restarts. This is computed from those limits, not yet measured over a full cycle, and busier slots shorten it. An older signature answers "not found".
- The network survives one quiet validator of four (three of four hold 75 % of the stake); two quiet validators stop it.
- All validators are ours and run on one server, so the network has no independent validators. A wallet's gas here is test gas from a faucet, with no value. $COGITO, the token on Solana mainnet, becomes this chain's gas and stake at the gas switch; until then the chain runs on test gas from a faucet.
- The digit mind reads drawings less reliably than MNIST digits, because a drawing is not written the way MNIST's digits were. The 98.01 % figure is for the MNIST test set only.