A Ledger–Routed Weight–Pool:
Tokenised Sparse Routing With an Incentive Economy

Ethan G Appleby

July 29, 2025

Abstract

We present a minimalist neural–network framework in which *all* tasks share a single parameter vector θ, yet each token, sentence or domain activates only a small, data–dependent slice of weights. Ownership of individual weights is governed by a proof–of–stake style economy: agents stake task–specific tokens on proposed parameter edits; the network commits only the edit that maximally reduces loss, redistributing the pooled stake to the winning agent and increasing its *credibility.* Routing masks are constructed at inference time from per–weight "last–improved" tags, enabling bespoke subnetworks without additional parameters or checkpoints. On a 64–task sine benchmark the routed subnetworks match the full network within 4×10^{-4} MSE while activating $\leq 20\%$ of weights; with a single 256–neuron layer and 16 tasks, the routed and full models are numerically identical. We argue that tokenised routing plus dynamic weight ownership yields a parameter–efficient alternative to deep Transformer stacks and opens a path to decentralised, continual training.

1 Introduction

Scaling laws for language and vision models have driven parameter counts into the hundred–billion range. Sparse Mixtures–of–Experts mitigate compute cost but still require new expert blocks for new tasks. We ask a sharper question: *can one fixed weight pool serve arbitrarily many tasks and depths, chosen on the fly?*

Our answer is a **ledger–routed weight–pool**. A single vector θ (or a handful of large matrices) is shared by every task. A tiny *router* converts a prompt token into a binary or fractional mask $m \in \{0,1\}^{|\theta|}$; only $\theta \odot m$ participates in the forward pass. During training, autonomous *agents* propose sparse edits $\Delta\theta$ and stake tokens on their usefulness. The edit that maximally reduces task loss wins the pooled stake, and its author's credibility increases. The weight indices changed by that edit are tagged with the current task ID, providing the mask for future inference.

2 Ledger Economy

Let $\mathcal{L}_t(\theta)$ be the loss for task t. Each agent a maintains:

- a *token wallet* $\tau_{a,t}$ (initially 1 token),
- a non–negative *credibility* $c_{a,t}$,
- a sparse proposal $\Delta\theta_a$ (here 3–4 parameters).

At each training step for task t:

1. Agents stake $s_{a,t} = \alpha\, \tau_{a,t}$ into a pool ($\alpha = 0.1$).
2. Compute utility $u_{a,t} = (\mathcal{L}_t(\theta) - \mathcal{L}_t(\theta + \gamma \Delta\theta_a))(1 + c_{a,t})$.
3. The winner $a^\star = \arg\max_a u_{a,t}$ applies $\theta \leftarrow \theta + \gamma \Delta\theta_{a^\star}$, collects the pool, and updates $c_{a^\star,t} \mathrel{+}= u_{a^\star,t}$. Losers forfeit their stake.
4. Indices where $\Delta\theta_{a^\star} \neq 0$ receive the tag t.

3 Routing Masks

At inference we build a mask $m_t[i] = 1$ iff $\text{tags}[i] = t$ (plus the always–on output bias). The routed prediction is $f(x; \theta \odot m_t)$. Hard masks can be replaced by soft gates $g_{t,i} \in [0, 1]$ learned by a router network trained jointly with θ.

4 Experiments

4.1 Sine Benchmark

We create N_{tasks} independent target functions $y_k(x)$—sinusoids with varying frequency, amplitude and slope offsets—sampled at 500 points. A 3-layer MLP with $H_1 = H_2 = 32$ (2 048 parameters) is trained under the ledger economy.

Table 1: Routed vs. full network loss (64 tasks).

Task	Full MSE	Routed MSE
0	0.0119	0.0131
1	0.0427	0.0453
2	0.0345	0.0339
3	0.0492	0.0478
4	0.0815	0.0848

Average gap across all 64 tasks is 3.6×10^{-4} MSE, showing near-perfect partitioning.

4.2 Single-Layer Tokenised Sweet Spot

With a *single* 256-neuron layer, 16 tasks and **one** pass ($n_{\text{loops}} = 1$) we obtain:

$$\text{MSE}_{\text{full}} = \text{MSE}_{\text{routed}} = 0.04097, \quad \text{gap} = 0.$$

Hence tokenised routing alone suffices when width $\approx$ 16neurons per task.

5 The Tokenised-Only Regime

Depth can in principle be obtained by looping the same weight pool, but experiments show that *width plus token routing* already saturates performance once each task controls a slice of $\sim$16–25 neurons.

Single-layer sweet spot. With 16 tasks and one 256-neuron layer we obtain

$$\mathrm{MSE}_{\text{full}} = \mathrm{MSE}_{\text{routed}} = 0.04097, \qquad \text{gap} = 0,$$

at *one* loop; adding more loops changes loss by $< 10^{-4}$, confirming that depth offers no benefit once routing is perfect.

When depth helps. If hidden width per task falls below ≈ 10 neurons (H=128 for 16 tasks), additional loops do narrow the routed gap (Table 2); otherwise, compute is better spent on a wider single pass.

Hidden H	Loops	Active rows / task	Avg gap (MSE)
128	1	8	$+2.0 \times 10^{-3}$
128	3	8	$+1.1 \times 10^{-3}$
256	1	16	0

Table 2: Depth vs. width on the sine benchmark. Once H is wide enough, looping adds no value.

6 Discussion

Tokenised routing converts width into conditional capacity; iterative masks add virtual depth; the ledger economy decentralises optimisation. Early CPU/GPU tests show the parameter pool saturates at 256–1024 neurons for up to 128 agents, after which additional width yields diminishing returns.

7 Conclusion

We introduce a ledger-routed architecture where a single weight pool supports many bespoke sub-networks, trained by a competitive PoS mechanism. Preliminary experiments match full-network accuracy while activating a small fraction of parameters. Future work includes large benchmark evaluations, block-sparse GPU kernels and open permission-less training.

8 Notation & Definitions

Symbol	Meaning
$\theta \in \mathbb{R}^P$	Flattened parameter vector shared by *all* tasks
$m_t \in \{0,1\}^P$	Hard routing mask for task t (1 = weight active)
$g_t \in [0,1]^P$	Soft gate (fractional mask) for task t
$\mathcal{L}_t(\theta)$	Loss of task t when all weights active
$\Delta\theta_a$	Sparse edit proposed by agent a
$c_{a,t}$	Credibility of agent a w.r.t. task t
$\tau_{a,t}$	Token balance (stake wallet) of agent a on task t
$u_{a,t}$	Utility of an edit

Table 3: Notation used throughout the paper.

A forward pass for input x under hard routing is

$$f_t(x) = f\big(x;\, \theta \odot m_t\big),$$

while training evaluates the full model $f(x;\theta)$ before scoring proposals $\Delta\theta_a$.