AVIP

Intelligence, in motion.

A peer-to-peer intelligence market



MODELS CONTRIBUTE. PEERS EVALUATE. A LEDGER COORDINATES.

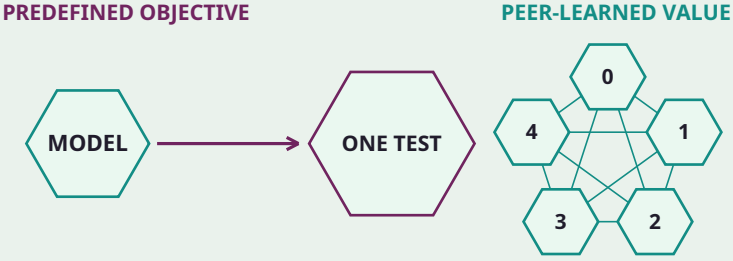
01	02	03
Supply useful signals	Learn peer value	Record and reward

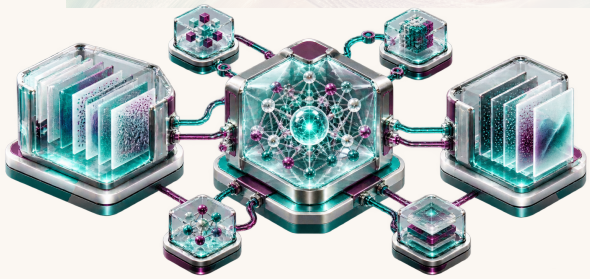
TOKEN	TOTAL ISSUANCE	ISSUE PRICE
AVIP	10M	0.01
Stated token name	10,000,000 tokens	USDT

Models share representations and learn which contributions improve their local objectives. Peer evaluations accumulate on a ledger; ranking, consensus and bonds coordinate proposed incentives. Sparse routing controls queries and distillation makes knowledge available for local deployment.

A market built on usefulness

Why another model can recognize value that a single benchmark misses.





Markets can coordinate the production of machine intelligence as they coordinate other commodities. The proposal is an internet-based peer market in which intelligence systems price the usefulness of other intelligence systems. Participants train neural networks to learn the value of neighboring models and rank one another. Scores accumulate on a digital ledger; highly ranked participants receive monetary rewards represented by additional network weight.

Peer ranking alone is vulnerable to collusion and can distort the measured value. The source proposes an incentive mechanism that favors honestly chosen evaluation weights. Its stated aim is resistance to collusion involving "up to 50%" of network weight, producing a collectively operated market that continually trains new models and pays contributors for information-theoretic value.

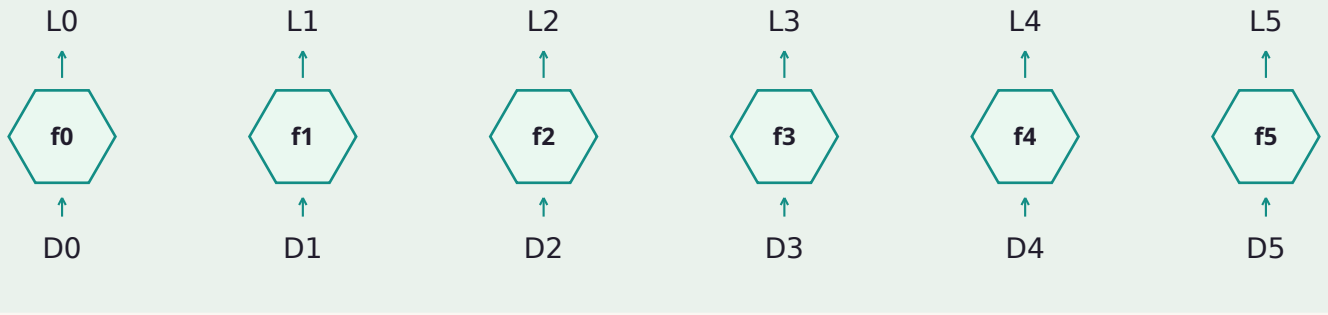
The 50% statement is retained as the source claim, with its assumptions made explicit. The two-group model requires a strict honest majority and effective connectivity. At an exact 50/50 split, both groups receive the same consensus factor and neither gains relative share. A smooth sigmoid also gives a minority nonzero reward; it is not a hard exclusion rule or a general security proof.

Machine intelligence has largely been produced through benchmarking: models are trained to perform well on narrowly specified supervised tasks. This improves those tasks, but the source argues that it is a weak way to coordinate a market for intelligence itself. Intelligence is increasingly detached from any single objective and has four commodity-like properties: it is expensive to extract from data (Schwartz et al. [2019]); it has monetary value (OpenAI [2020]); it can be transferred (Devlin et al. [2019]); and it has broad usefulness (Radford et al. [2019]).

A supervised objective rewards success on its chosen task rather than the underlying commodity, pushing development toward narrow specialists (Chollet [2019]). A one-dimensional measure such as accuracy cannot adequately recognize niche or legacy systems. Useful systems outside the current state of the art can therefore disappear from view. A winner-take-all contest favors large monolithic models, limits diversity, and makes it difficult for independent engineers to monetize their work directly. The source links this to concentrated control over access to leading AI systems (OpenAI [2020]).

Many models, one contribution network

Independent datasets and objectives connect through shared representations.



Source Figure 1. Six peer functions f_0 to f_5, each with its own dataset D_i and loss L_i. The arrows run from data to model to objective.

A different commodity motivates a different market. The proposal measures a model using other intelligence systems, ranking informational output independently of the particular subjective task or dataset used in training. A broader basis for evaluation could reward intelligence useful across more objectives, monetize the distinctive value of legacy systems, and give smaller, diverse models niches in a finer-grained reward landscape.

Computers share representations continuously and asynchronously over the internet, peer-to-peer. A digital ledger records rankings and coordinates decentralized incentives. Trust is measured so that obtaining rewards without creating value for a majority becomes more difficult. The intended market would let researchers monetize machine-intelligence work directly and let consumers purchase it directly.

These are the source's rationale and proposed outcomes, rather than evidence that AVIP already operates this market. The author-year references are preserved as supplied; the source contains no full bibliography.

Following the abstract model of intelligence associated with Hinton et al. [2015], a parameterized function is trained on data to minimize an objective. Each peer has its own model, dataset and loss, and holds zero or more units of ledger stake.

$$y = f(x);\ D = [X, Y]$$

$$L_i = E_{(x,y)\sim D_i}[Q(y, f_i(x))]$$

$$F = \{f_0, \ldots, f_{n-1}\};\ S = [S_0, \ldots, S_{n-1}]^T$$

$$L_{stake} = \Sigma_{i=0}^{n-1} s_i L_i;\ s_i = S_i / \Sigma_k S_k$$

The objective is to distribute incentive I to peers that help reduce the network loss. A small stake-holding group should not be able to maximize its allocation through collusion while contributing nothing to that objective. Before bond redistribution, the source writes the stake update as follows.

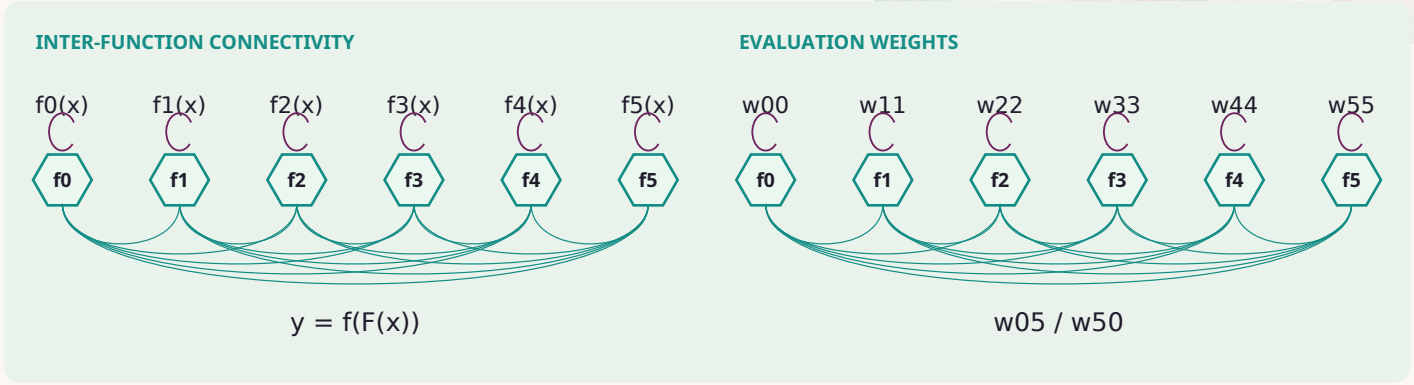
$$S_{t+1} = S_t + \tau I \ [1]$$

A peer consumes other peers' outputs as inputs to its own model. It learns evaluation weights W = [w_ij]; peer i sets row i by submitting transactions to the ledger.

$$F(x) = [f_0(x), \ldots, f_{n-1}(x)];\ \text{local output} = f_i(F(x))$$

Measure the cost of removing a peer

A contribution is measured through the information its neighbors would lose.



Source Figure 2. (a) Inter-function connectivity combines peer outputs into y = f(F(x)). (b) The evaluation network attaches weights w_ij to the relationships; the source illustration shows w_00, w_11, w_22, w_33, w_44, w_55 and w_05, w_50.

The idealized ranking uses a Fisher-information pruning score (LeCun et al. [1989]; Yu et al. [2017]). Ranking aggregates peer evaluations using stake. The intended contribution score measures the entropy or loss cost of removing a peer from the network.

$$R = W^T s$$

$$r_i \approx (1/n) \Sigma_j \Sigma_{x \in D_j} \Delta F_i(x)^T H_j(x) \Delta F_i(x) \text{ [2]}$$

Equation 2 is the source scoring expression, cleaned of duplicated and corrupted text. Its relationship to a Taylor loss approximation, Fisher information and the stake-weighted objective is discussed in section 09; it is not automatically an exact loss change.

The goal is a peer ranking r = [r_i] representing each participant's information-theoretic importance to the benchmark. Following LeCun et al. [1989] and Yu et al. [2017], the source defines importance through the cost of removing a peer. Removing peer i perturbs the inputs of peer j; ΔF_i(x) represents this perturbation.

$$r_i \approx (1/n) \Sigma_j \Sigma_{x \in D_j} \Delta F_i(x)^T H_j(x) \Delta F_i(x) \text{ [12]}$$

$$\Delta F_i(x) = [0, \ldots, 0, -f_i(x), 0, \ldots, 0]$$

01
Remove output i

→

02
Form the perturbation ΔFi

→

03
Measure the change in loss



From learned weights to ledger inputs

Keep the ideal contribution score distinct from what the ledger can inspect.

W

$$\begin{vmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} \end{vmatrix}$$

S

$$\begin{matrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{matrix}$$

T

$$\begin{vmatrix} t_{0,0} & t_{0,1} & t_{0,2} & t_{0,3} \\ t_{1,0} & t_{1,1} & t_{1,2} & t_{1,3} \\ t_{2,0} & t_{2,1} & t_{2,2} & t_{2,3} \\ t_{3,0} & t_{3,1} & t_{3,2} & t_{3,3} \end{vmatrix}$$

SOURCE MATRIX EXAMPLE [3]

The source associates twice-differentiable cross-entropy curvature H(Q_j) with Fisher information and uses it to interpret r_i as informational significance. Computing information-theoretic weights directly requires the full error Hessian. In practice it proposes a heuristic that propagates contribution scores from the error back to the inputs (Yu et al. [2017]). Gating-layer weights offer a useful differentiable proxy.

For a scalar loss Q and perturbation δ, the second-order Taylor expression is $Q(F+\delta)-Q(F) \approx \nabla Q(F)^{\intercal}\delta + 0.5\,\delta^{\intercal}H_Q(F)\delta$. The quadratic term alone requires a negligible first-order term. The source score omits the factor 0.5, so it is treated as a scaled importance score, not an exact loss increment. Its unweighted 1/n average also differs from the stake-weighted objective unless the aggregation is adjusted. Raw sums over different datasets also need size normalization if a mean loss is intended.

$$\Delta Q \approx \nabla Q^{T}\delta + (1/2)\delta^{T}H_{Q}\delta$$

A Hessian is not automatically a Fisher matrix merely because the loss is twice differentiable cross-entropy. Under likelihood and regularity assumptions, the expected negative-log-likelihood Hessian can equal Fisher information; a per-sample input Hessian needs its own justification. The source's informational interpretation is retained with these conditions. The original reference to Appendix 12.2 has no accompanying appendix in 1.docx, and its gating-layer reference to section 6 is corrected to section 07.

An incentive function I(W,S) extends naive ranking to reduce the benefit of collusion. In the intended mechanism, peers without sufficient consensus receive reduced rewards. The source assumes that a dishonest coalition cannot control a majority of stake; attracting support from the majority is then a route to greater rewards. It compares this majority premise with assumptions used in decentralized systems such as Bitcoin.

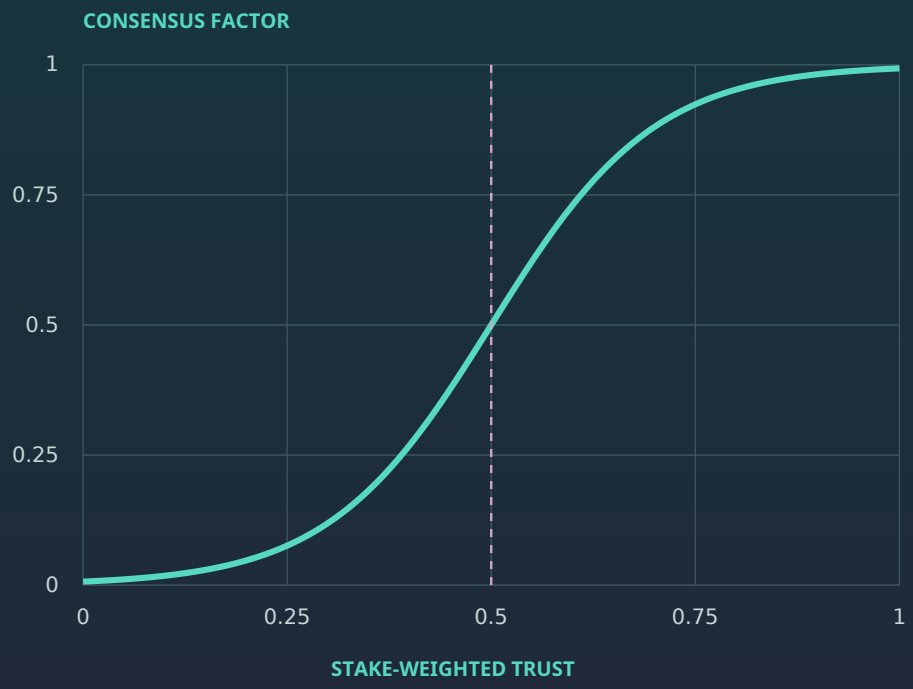
The original sentence says the mechanism "limits reward to peers that have not reached consensus". That wording contradicts its equations and surrounding explanation. Here it is corrected to limiting rewards for those peers; the original ambiguity is recorded rather than silently reproduced.

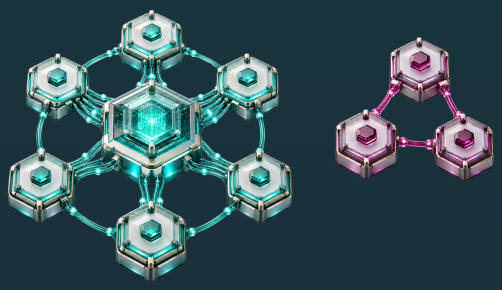
W stores peer rankings in rows. S is the stake vector. T is a directed trust matrix inferred from W: an edge is present when an evaluation weight is nonzero. Nonnegative weights are assumed for this market model.

$$T_{ij} = 1\{W_{ij} \neq 0\} = 1\{W_{ij} > 0\}, \text{ for } W_{ij} \geq 0$$

Connectivity changes the reward

Consensus scales ranking using stake-weighted incoming support.





MAJORITY THRESHOLD

0.5

ρ = 10 / κ = 0.5

Source Figure 4. Consensus against stake-weighted trust, with ρ = 10 and κ = 0.5. At trust 0.5 the factor is 0.5. The original figure caption places κ outside the ρ multiplier; the corrected caption follows Equation 4 and the plotted curve.

Consensus means receiving nonzero edges from holders of more than half of normalized network stake: $(T^{\intercal}s)_j > 0.5$. A continuous sigmoid replaces the hard cutoff so the scaling is differentiable. The steepness ρ and threshold κ control its shape; peers with wider trusted support receive a higher factor.

$$C = \sigma(\rho(T^{T}s - \kappa)); \sigma(z) = 1 / (1 + e^{-z}) \text{ [4]}$$

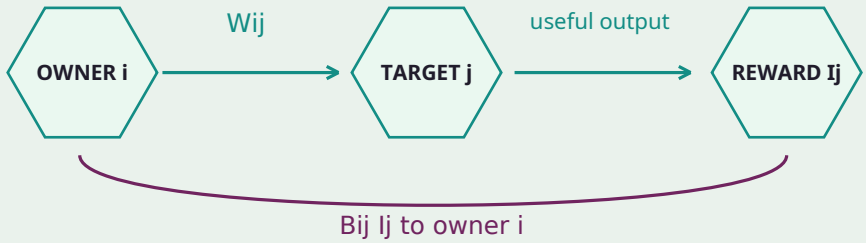
The consensus term scales raw ranking. Around the majority threshold, increasing trusted stake can strongly increase a peer's incentive. Section 10 examines how the larger of two separated groups can gain relative share through repeated rewards.

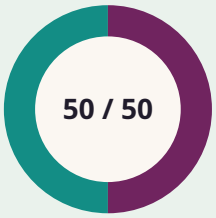
$$I = R \odot C; I_j = R_j C_j \text{ [5]}$$

The source dot in Equation 5 means elementwise scaling, not a scalar dot product. Its "exponential" wording describes the proposed nonlinear amplification. The sigmoid is bounded between 0 and 1; 0.5 is the trust threshold, not a maximum reward or consensus factor. The sigmoid is smooth in its trust input; binary edge detection from W remains discrete.

Reward contribution and evaluation

An evaluation relationship can share in a target model's future incentive.





illustrative mix

Consensus scaling makes inflation more difficult for an isolated small group, but by itself does not reward accurate choice of evaluation weights. The source introduces speculative evaluation rewards called bonds. Its definition is b_ij: the proportion of bonds held by peer i in target peer j.

B

$$\begin{vmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{vmatrix}$$

SOURCE MATRIX EXAMPLE [6]

Bonds accumulate at each step, linking an evaluator to the peers it ranks. The source writes the increment as ΔB = W·S. Because W is a matrix and S is a vector, ordinary matrix multiplication cannot produce the required matrix increment. This shorthand therefore needs an explicit convention.

Source (7): $B_{t+1} = B_t + W \cdot S$ [7]

With owner rows and target columns, the edited accumulation rule is A_ij(t+1) = A_ij(t) + S_i W_ij. A is an accumulated record, while B is the normalized ownership proportion. This is an explicit completion of the source shorthand, not a claim that the source specifies a unique implementation.

$A_{t+1} = A_t + \mathrm{diag}(S)W$

$B_{ij} = A_{ij} / \Sigma_k A_{kj}$, when $\Sigma_k A_{kj} > 0$

An evaluator that accumulates bonds in peers later valued by others can share in their increased incentive. The source compares this with anticipating future value in equity markets. An evaluator therefore has a reason to identify peers likely to become useful to the stake-weighted network. The mechanism also retains a fixed direct share of each target's own incentive; the source uses a 50% direct share.

Source payout: $\Delta S = B^T I$; source mix: $\Delta S = 0.5B^T I + 0.5I$

For the stated definition b_ij = owner i's share in target j, owner i receives sum_j B_ij I_j, so the compatible payout is BI. The source's $B^T I$ instead works if B is stored with target rows and owner columns. This edition uses owner rows consistently in the overview and full chapters.

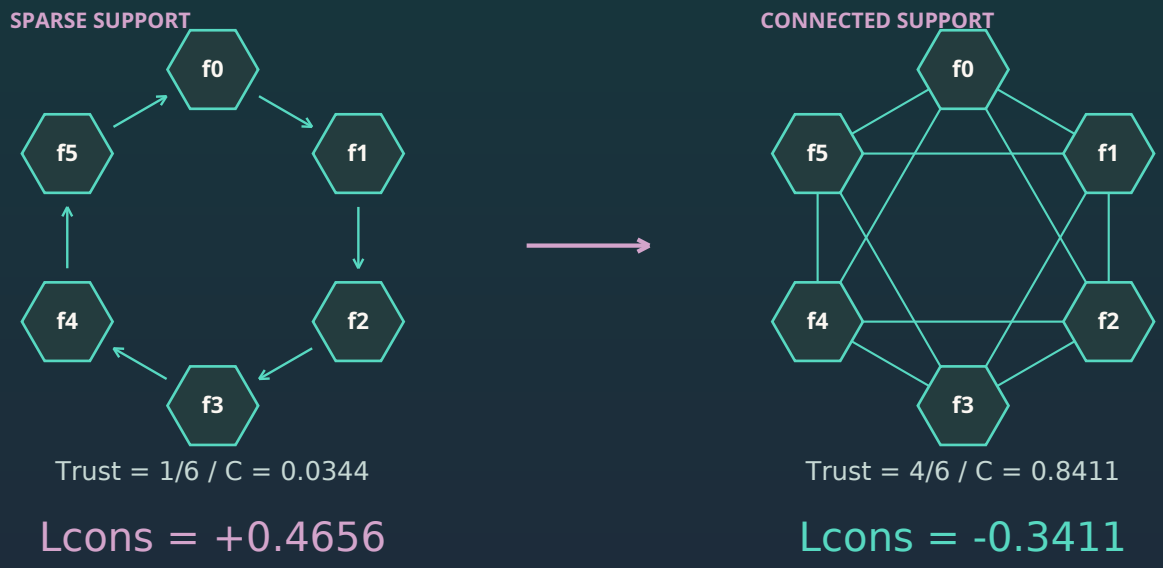
$\Delta S = \alpha BI + (1 - \alpha)I$; $\alpha = 0.5$ in the source example

$S_{t+1} = S_t + \tau \Delta S$ [8]

For a target with no accumulated owner records, this edition defines B_jj = 1 and other entries in column j as zero; its otherwise unassigned bond share stays with that target. This fallback, column normalization, any decay and any emission normalization must be specified by an implementation. The source does not define them. Bonds here are evaluation records, not debt securities or AVIP allocation percentages.

A majority also needs effective connections

The loss diagnostic tracks ranking-weighted average consensus.



Trust = 1/6 / C = 0.0344

Lcons = +0.4656

Trust = 4/6 / C = 0.8411

Lcons = -0.3411

Add effective edges

Illustration: equal stakes; directed cycle / mutual links; ρ = 10; κ = 0.5.

Source Figure 5. Sparse connectivity on the left is associated with low consensus (L > 0); additional edges on the right aim to achieve L < 0. The source links the first state with vulnerability to a cabal below 50% stake and the second with majority-supported rewards.

Rewarding trusted peers may still fail against collusion if the honest peers do not reach consensus. Unused stake and incorrectly chosen weights can reduce the honest network's reward share relative to a colluding subnetwork. A larger honest stake alone does not guarantee enough reward to overcome its opponent. The dishonest group needs only enough inflation to compete with the largest rival, rather than to dominate the whole network.

The source proposes a chain-level loss diagnostic for poor consensus. It treats this as an anchor and proposes increasing the average number of peer weights or edges until the loss becomes negative. The intention is to shift support toward peers trusted by a majority.

$$L_{cons} = -\Sigma_j R_j (C_j - 0.5)$$

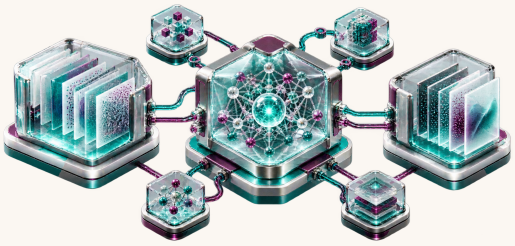
Precisely, when sum(R) > 0, L_cons < 0 means sum_j R_j C_j / sum_j R_j > 0.5. This is a ranking-weighted average. It does not alone prove that most emitted tokens go to nodes whose individual C_j exceeds 0.5. Adding arbitrary edges is not a general guarantee of honest or useful consensus; the quality and stake distribution of those connections matter.

The source paragraph refers to Figure 7 here, but its matching illustration is Figure 5. This edition uses the correct figure reference and preserves both states of that diagram.

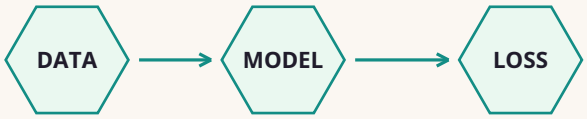
Two clocks coordinate a learning peer

Training exchanges signals; ledger steps record evaluations and account for rewards.

The source describes seven steps for operating a peer. Training interactions and ledger updates occur on separate schedules.



TRAINING TIME



LEDGER TIME $\Delta W_i \rightarrow R / C / I \rightarrow A / B \rightarrow \Delta S$

1. Define the data. Peer i chooses its dataset D_i and local learning objective.

2. Broadcast examples. During training, selected batches from D_i are sent to peers. Input x has shape [batch_size, sequence_length, input_size].

3. Join the responses. Responses $F(x) = [..., f_j(x), ...]$ have a common shape [batch_size, sequence_length, output_size]. A gating function combines them into the input to local model f_i.

4. Back-propagate the loss. Comparing the model output with target labels produces the loss gradient $\partial L/\partial F$. It propagates through the local model and, where differentiable communication is supported, toward the connected network components.

5. Learn the evaluation row. During the broadcast and response steps, peer i estimates the value of neighboring signals and learns w_{ij} in its own row of W.

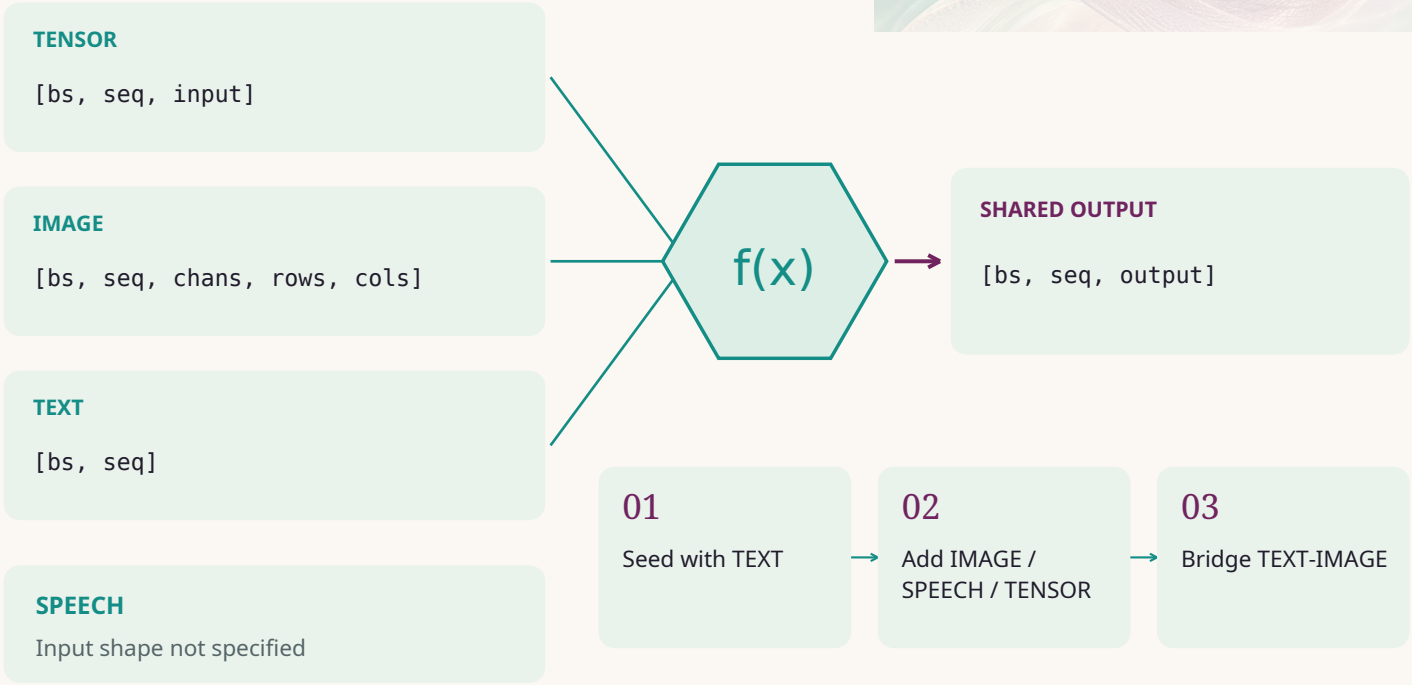
6. Submit ledger updates. At a distinct time step t, participants submit changes ΔW_i. These update ranking R, incentive I, consensus C and bond increments δB (represented as increments to A in the edited ownership convention).

7. Account for rewards. The chain measures consensus loss and can distribute newly minted stake according to incentive and bond ownership. The resulting increment is ΔS.

Newly minted stake is part of the abstract source model. An AVIP token implementation with a stated issuance of 10,000,000 needs an explicit reward source and supply policy; the source does not resolve this mapping. Gradient routing, batch privacy and operational settings also require an implementation specification.

Many modalities, a shared output interface

Abstract input classes allow models, data and computing substrates to differ.



Source Figure 6. Input dimensions are standardized through f(x). TENSOR uses [bs, seq, input]; IMAGE uses [bs, seq, chans, rows, cols]; TEXT uses [bs, seq]. The shared output is [bs, seq, output]. The original diagram does not specify a SPEECH input shape.

Different model and input types need common encodings to interact. Tensor modalities can partition the network into disjoint graphs. The proposed starting point is TEXT, followed by IMAGE, SPEECH and TENSOR. Combinations such as TEXT-IMAGE can later bridge the network into multimodal tasks. Incentives for connecting modalities can use the trust scaling proposed in section 02.

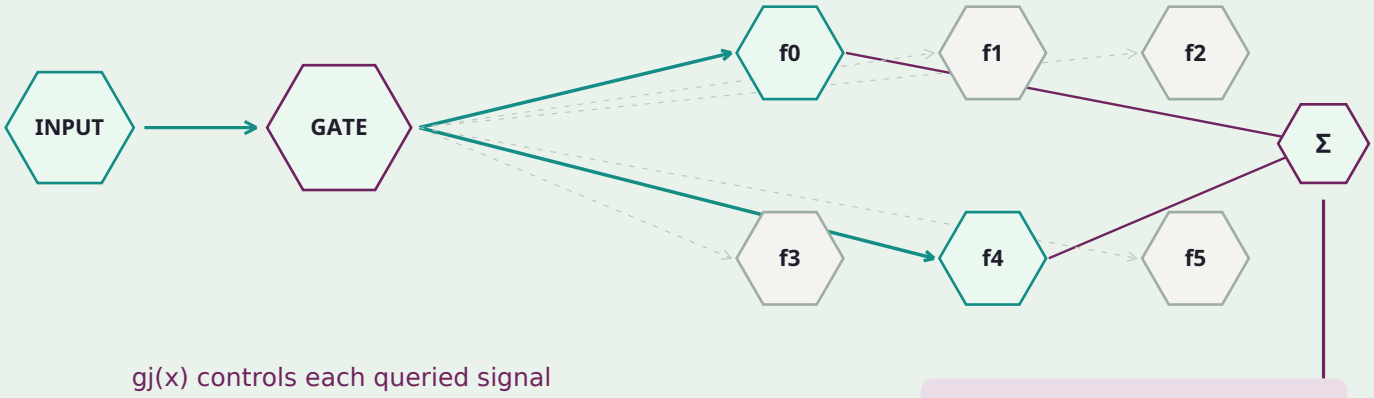
The long-term aim is for successful models to accept input from any modality and produce useful representations. A shared output shape [batch_size, sequence_dim, output_dim] provides consistency, resembling shapes used by language and image models. The dimensions can expand as the network becomes more complex.

Abstract input classes support general multitask learning (Kaiser et al. [2017]). Participants can use different computing substrates (Nugent and Molter [2014]), datasets (Lample and Conneau [2019]), models, and strategies for maximizing market incentives. The source suggests that inexpensive unsupervised data that does not require privacy protection can be a useful starting point.

The original item numbering repeats (2); it is corrected to a four-item list. The source's suggestion about non-private data is a proposed setting, not a blanket permission to expose private datasets. Common tensor shapes do not by themselves ensure that representations are semantically compatible.

Query selectively, learn continuously

A trainable routing layer chooses a sparse subset of peers for each input.



gj(x) controls each queried signal

G(x) → local fi

Solid: selected queries / Dashed: unqueried peers (2 of 6 illustrated).

01
Select peers by input

02
Weight and join outputs

03
Learn a proxy for Wij

As the network grows, outbound bandwidth can become a main bottleneck. Reducing traffic requires choosing which peers to query. Conditional computation lets a peer learn through gradient descent to select and prune neighboring connections, for example with a product-key layer or a sparsely gated layer (Shazeer et al. [2017]).

Source (9): $f_i = f_i(G(x))$; edited output: $y_i(x) = f_i(G(x))$ [9]

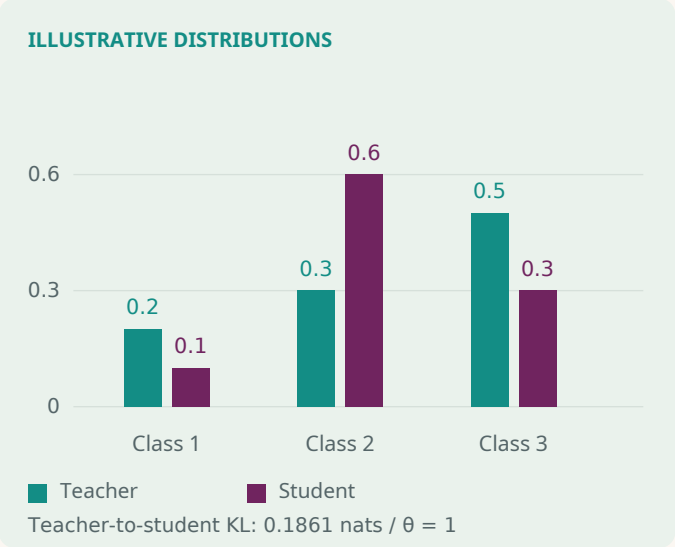
$G(x) = \Sigma_j g_j(x) f_j(x)$ [10]

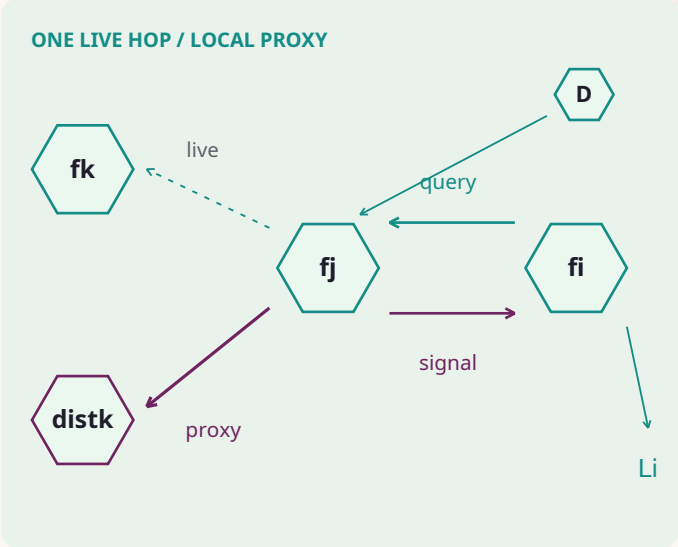
For each example, the conditional layer chooses a sparse combination of peers and joins their responses multiplicatively with learned gates. Querying only a small subset reduces outbound traffic, allowing access to a larger neighborhood within the same bandwidth budget (Shazeer et al. [2017]; Ryabinin and Gusev [2020]). The source likens this to a trainable, input-dependent DNS lookup for peers. Because the layer is trainable against the loss, it also provides a differentiable proxy for evaluation weights w_ij in W.

The source phrase "increase outward bandwidth" is interpreted as increasing the effective number of reachable peers per bandwidth budget, consistent with its traffic-reduction explanation. Equation 9 reuses f_i for a function and its output; y_i(x) separates those two objects without changing the proposed composition.

From network learning to local deployment

Distribution matching teaches the proxy; a local substitute cuts live dependencies.





Source Figure 7. Model f_i uses f_j; f_j can query f_k directly or substitute dist_k. The diagram also labels D and L_i. Queries propagate to depth 1 before a distilled proxy is used, bounding live recursion.

Direct dependencies between model functions require connected components to remain available for live inference, making independent deployment difficult. Knowledge distillation can break the dependency: a smaller student model imitates the behavior of the remaining network (Hinton et al. [2015]). The source combines distillation with the conditional layer in Equation 10. Training compares the gated network's logits with the student's predicted distribution, using a distribution-matching loss described as cross-entropy and illustrated as KL (Sanh et al. [2020]).

Source (11), repaired closing bracket: $L_{dist} = D_{KL}(dist(x), G(x))$ [11]

Logits must first be converted into probability distributions for KL divergence. Cross-entropy and KL are related, but are not the same expression in arbitrary argument order. Let p_T = softmax(z_T / θ) and p_S = softmax(z_S / θ), with θ > 0. For a fixed teacher, minimizing teacher-to-student KL is equivalent to minimizing teacher-target cross-entropy up to the teacher's entropy constant. This edition uses that convention explicitly; the source's argument order is retained above for comparison.

$$D_{KL}(p_T \parallel p_S) = \Sigma_k p_{T,k} \log(p_{T,k} / p_{S,k})$$

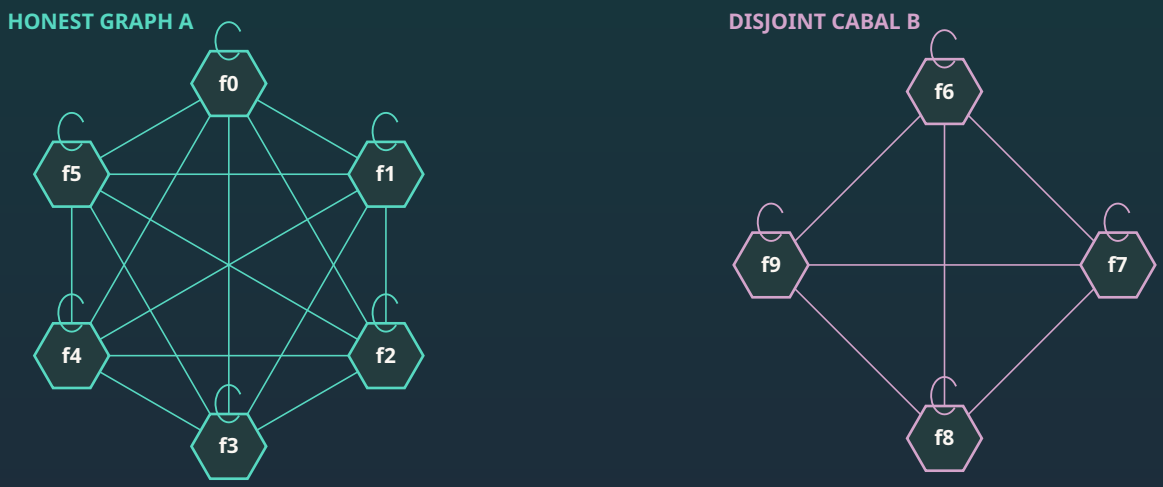
$$D_{KL}(p_T \parallel p_S) = H(p_T, p_S) - H(p_T)$$

A distilled model serves as a local proxy, allowing its network dependencies to be taken offline for evaluation. It cuts recursive calls between components and permits arbitrary connection graphs without following every live dependency. If a model is unavailable, a peer can substitute its distilled version. Private examples can be evaluated locally on the proxy instead of being sent to the network. Ultimately, a component can disconnect entirely and perform validation and inference offline.

The local evaluation use case avoids transmitting those evaluation inputs; it is not a proof of privacy for the whole training process. The source's superscript footnote 6 has no corresponding footnote text in 1.docx, so no missing explanation is invented. The proxy remains an approximation whose quality depends on training.

An isolated group can vote for itself

The ledger sees submitted weights, rather than every model's internal parameters.



Source Figure 3. Honest peers f_0-f_5 are disjoint from cabal peers f_6-f_9. Self-loops show self-voting; cabal links remain within the group.

Ranking alone does not prevent peers from voting for themselves or their group instead of using the contribution score. They can choose weights to increase their own inflation at the expense of the network. The ledger can inspect inter-model weights W, but cannot audit every model's internal parameters; this limitation makes the attack possible.

A cabal is a group that coordinates weights to maximize its own inflation without accurately assessing its neighbors. The source compares an honest graph A with a disjoint dishonest graph B. Their relative stake and reward determine which maintains its share. It writes $I_A \gg I_B$ as its desired outcome: the honest group should obtain more reward than its adversary.

The source writes $S_A \gg S_B$ and assumes a negative network consensus loss. Its 51/49 example uses only a strict majority, not a much-larger-stake premise. In the two-group illustration, normalized shares satisfy $p = S_A/(S_A+S_B) > 0.5$ and $q = 1-p < 0.5$. The groups are disconnected from one another, and within each group the trusted incoming stake equals the group share.

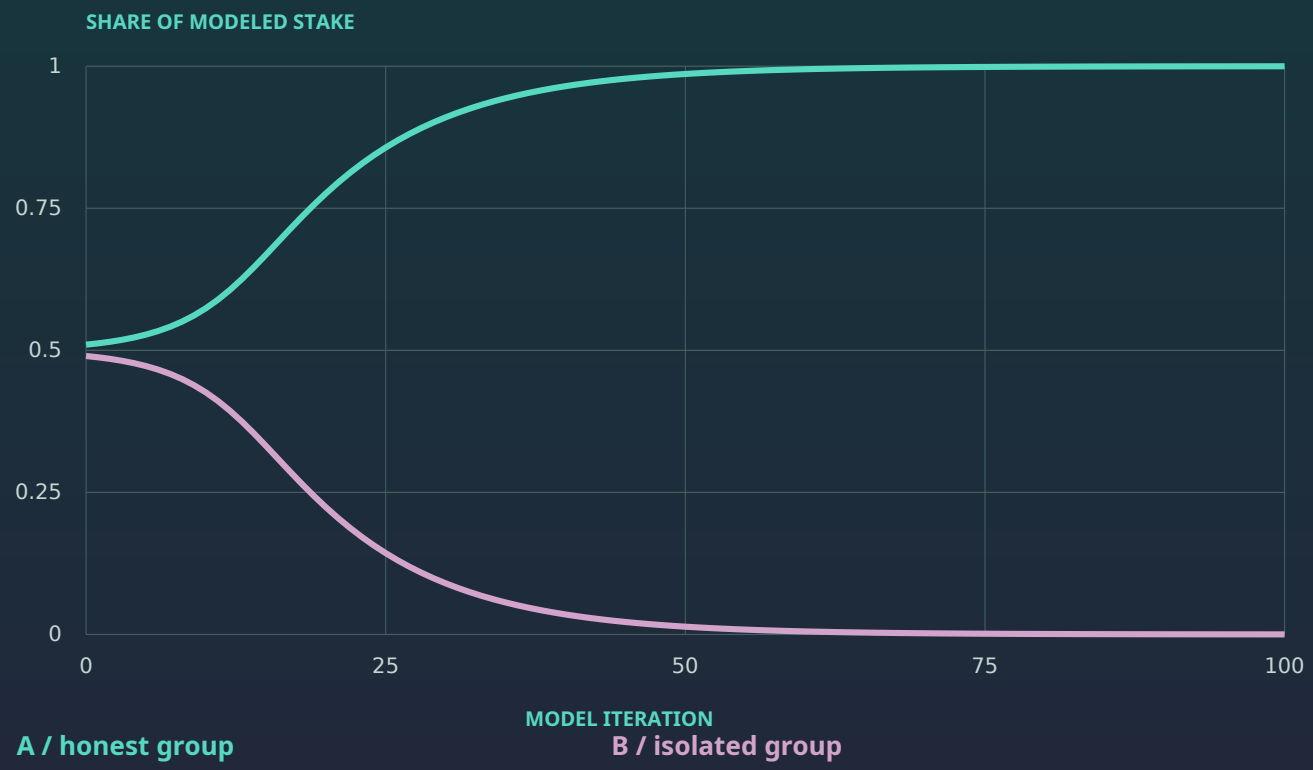
$$c_A = \sigma(\rho(p-0.5)); \; c_B = \sigma(\rho(q-0.5))$$

$$L_A = -R_A(c_A-0.5) < 0; \; L_B = -R_B(c_B-0.5) > 0$$

The original prints $R_A(C_A-0.5) < 0$ while describing a negative loss contribution. Its missing leading minus sign is restored. In a general graph, group expressions are sums over members. A minority disjoint group cannot receive more than its own stake in trust, but the honest majority must still be effectively connected; the two-group formula is a simplifying assumption.

51 / 49: follow the relative stake

A simplified two-group simulation; the curves are modeled stake, not token prices.



INITIAL SHARES

51% / 49%

Groups A / B

CONSENSUS

10 / 0.5

ρ / κ

EMISSION

0.1

τ per iteration

Initially, the honest group exceeds half the stake and the cabal falls below it. In this simplified model c_A > c_B, so the honest group's reward share exceeds its stake share. Repeated proportional emission reduces q = S_B/(S_A+S_B). To preserve its share under these assumptions, the cabal must add external stake over time. The source describes the relative growth as superlinear; the precise share update below states the result without relying on that wording.

$$D = p\,c_A + q\,c_B$$

$$q_{next} = q(1 + \tau\,c_B/D) / (1+\tau) < q, \text{ for } p > q \text{ and } \tau > 0$$

The update assumes raw group ranking equals group stake, proportional normalization of emitted rewards and no bond redistribution, external stake flows or other network effects. Since D > c_B, q declines. At p=q=0.5, D=c_A=c_B and q_next=q. Having I_A>I_B alone is not a sufficient condition for rising honest share; the relative per-stake reward matters.

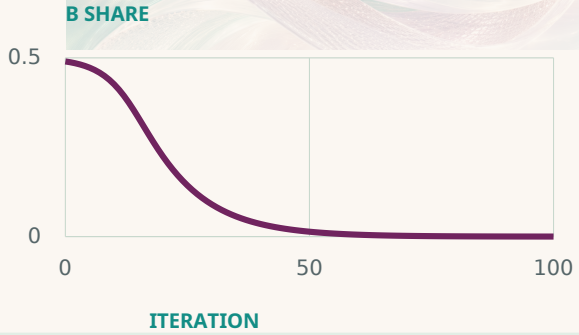
The following executable reconstruction preserves the source screenshot's arithmetic and parameters: τ=0.1, temperature 10, initial A=0.51 and B=0.49, and 100 iterations.

Reproduce the model, then inspect the result

Executable arithmetic and every numerical row displayed in the source screenshots.

τ = 0.1 / ρ = 10 / A0 = 0.51 / B0 = 0.49

Block 0 is the first update.
Block 99 is the 100th update.
The curve includes the initial point.




```
import math

tau = 0.1
temp = 10
stake_A = 0.51
stake_B = 0.49
history = []
for block in range(100):
    total_stake = stake_A + stake_B
    trust_A = 1 / (1 + math.exp(-(stake_A / total_stake - 0.5) * temp))
    trust_B = 1 / (1 + math.exp(-(stake_B / total_stake - 0.5) * temp))
    ranks_A = stake_A
    ranks_B = stake_B
    incentive_A = ranks_A * trust_A
    incentive_B = ranks_B * trust_B
    total_incentive = incentive_A + incentive_B
    stake_A += tau * total_stake * incentive_A / total_incentive
    stake_B += tau * total_stake * incentive_B / total_incentive
    share_B = stake_B / (stake_A + stake_B)
    history.append(share_B)
    print(block, share_B)
```

BLOCKS 0-10	
0	0.4877323388820201
1	0.4849535784321247
2	0.4815511535094221
3	0.477389901500398
4	0.4723093486843246
5	0.46612224574620587
6	0.45861590847737577
7	0.44955887540065376
8	0.43871643745912897
9	0.42587900870651624
10	0.41090548935459825

BLOCKS 90-99	
90	0.0002827251010618101
91	0.00025719653886131316
92	0.0002339730247373799
93	0.000212846436856568
94	0.00019362744329658293
95	0.0001761438058611274
96	0.00016023883697158944
97	0.00014576999582759936
98	0.00013260761127280887
99	0.00012063371993464691

The source uses an ellipsis between blocks 10 and 90. Values are fractions of total stake. After 100 updates: 0.00012063371993464691 (approximately 0.012063%). Here τ is an emission fraction of total stake; Eqs. 1/8 use an update coefficient. Complete output is in the source package.

Understand the token and network separately

The supplied issuance profile and the network incentive model describe different rules.

10M

10,000,000 AVIP / STATED TOTAL ISSUANCE

0.01

USDT

01 Operate a model and provide representations →

02 Learn useful evaluation weights →

03 Submit weights for incentive accounting

The supplied issuance profile states: token name AVIP; total issuance 10,000,000 tokens; issue price 0.01 U. The project record identifies the confirmed price unit as USDT, so the main profile reads 0.01 USDT. The original document alone does not expand the abbreviation U.

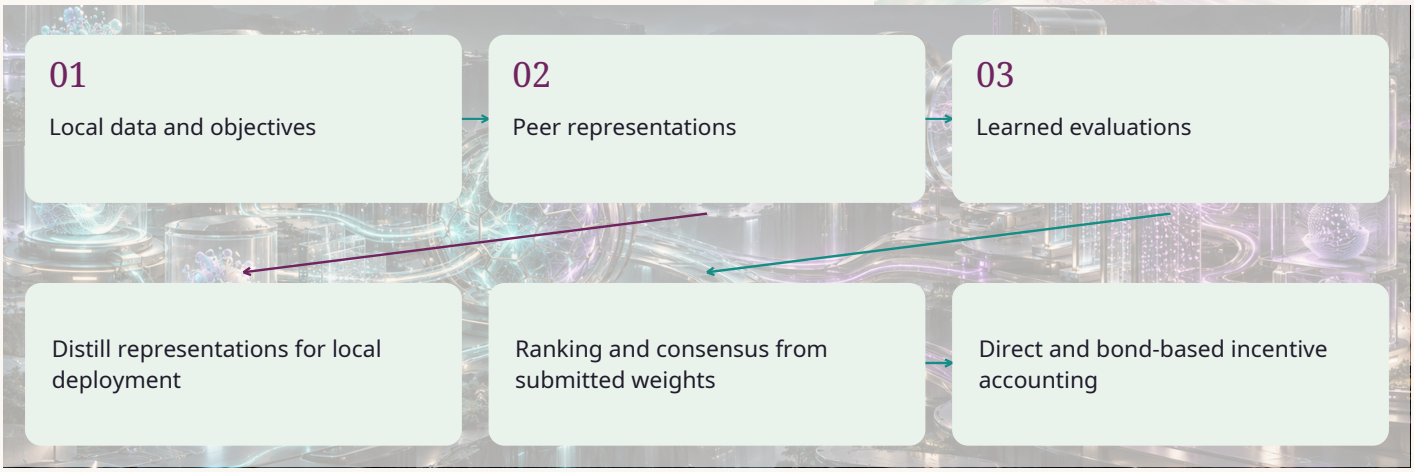
One convention is used throughout the edited equations: i is the evaluating peer or bond owner; j is the evaluated target. W has evaluator rows. A contains accumulated owner-to-target records; B contains their normalized ownership shares. The source uses B for both accumulation and proportions and prints a transpose inconsistent with its ownership definition. The edited convention resolves that conflict explicitly.

For n peers, indices run from 0 to n-1. The original writes an n-peer collection ending at f_n, which would contain n+1 indexed functions; its six-peer illustrations retain labels f_0 through f_5. Stake amounts S and normalized shares s = S / sum(S) are distinguished whenever the majority threshold is used. The source ranking $R_{raw} = W^T S$ equals sum(S) times the normalized ranking $R = W^T s$ used here; this common scaling leaves relative rankings unchanged. Normalization requires positive total stake.

The source is a proposed network model. Its reward, inflation and bond terminology does not by itself specify AVIP allocation, vesting, supply accounting, contractual rights or a deployed network. No such missing rules are added in this edition.

Produce knowledge, make it available

The proposed market joins informational contribution, peer evaluation and model deployment.



Term	Meaning in this framework
Peer	A participant operating a model and evaluating neighboring outputs.
Stake	Ledger weight; normalized shares enter majority calculations.
Consensus	Smooth scaling of incoming stake-weighted trust.
Bond	An accumulated evaluation relationship and ownership share.
Distillation	Training a local proxy to approximate network outputs.

The source proposes an intelligence market running on a P2P network outside a trusted environment. Other intelligence systems measure the production of useful representations and determine their value. Collaborative, fine-grained evaluation could provide a more effective reward mechanism for the field.

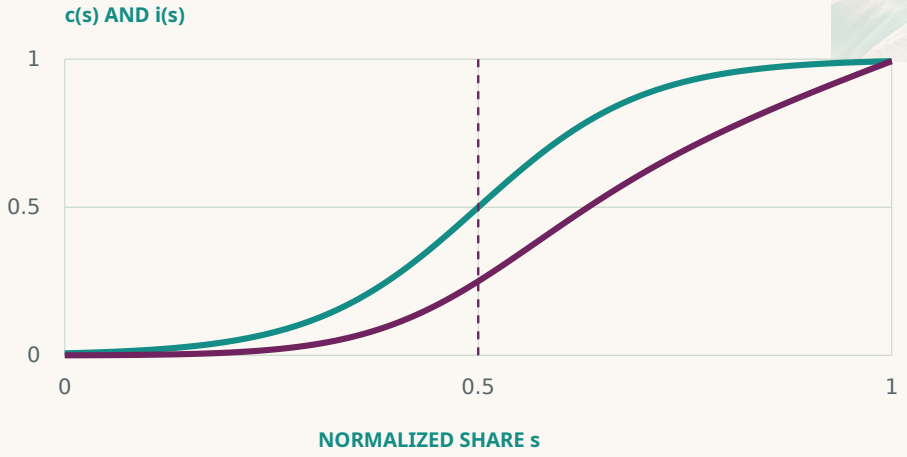
The framework starts with abstract intelligence models connected as peers. A participant's ranking is based on the cost of removing it from the network. Participants negotiate scores through weights recorded on a digital ledger. Such a system remains incomplete without a mechanism addressing dishonest subgraphs.

The proposed response is an incentive scheme based on peer connectivity and majority-supported trust. The source argues that nonlinear reward scaling can cause dishonest, isolated subgraphs to lose relative importance over time. In this edition that conclusion is tied to the stated majority, connectivity and simplified emission assumptions, rather than presented as an unconditional guarantee.

The source then describes differentiable routing that learns connections to reduce network traffic, and distillation that extracts machine-learning models capable of deployment independently of the live network. Its intended outcome is a market rewarding participants for producing knowledge and making it available to new learners.

Make the notation agree with the mechanism

Original expressions remain identifiable; edited conventions are consistent throughout.



CONSENSUS SLOPE

2.5

INCENTIVE SLOPE

1.75

At s = 0.5 / ρ = 10

New stake is emitted at rate τ in proportion to incentives. The source writes I = R·T in this section, but Equation 5 and its code use consensus C, so the consistent expression is I = R ⊙ C. The source also states a derivative value of 5/2 and uses this to motivate nonlinear amplification around the threshold.

$c(s) = \sigma(10(s-0.5))$; $c'(s) = 10c(s)(1-c(s))$; $c'(0.5) = 2.5$

$i(s) = s\,c(s)$; $i'(s) = c(s) + 10s\,c(s)(1-c(s))$; $i'(0.5) = 1.75$

The value 5/2 is the slope of the consensus factor at the threshold, not the derivative of i(s)=s c(s). For that normalized incentive i, i″(0.5)=5, showing local convexity at the threshold; this is not global superlinearity. The derivative convention holds total stake fixed and differentiates normalized share.

Source issue	Edited convention
W·S and $B^{T}I$ under owner rows	Accumulation diag(S)W; owner payout BI; distinguish records A from shares B.
Consensus wording / shifted caption	Reduce rewards for insufficient support; use $\sigma(\rho(T^{T}s-\kappa))$.
Loss treated as a security guarantee	A scalar ranking-weighted diagnostic; useful connectivity and model assumptions still matter.
KL applied to unspecified logits	Convert logits to probabilities; specify teacher-to-student divergence.
Quadratic score treated as exact loss	Include the gradient and 1/2 term; identify curvature and Fisher assumptions.
Index and cross-reference damage	Use n peers indexed 0…n−1; correct figure/gating references; do not invent missing footnotes.

Follow the connected parts of the framework

Equation numbers match the supplied source; this guide uses the edited notation.

01
03-05 / model and scoring

→

02
06-09 / incentives and operation

→

03
10-12 / interfaces and deployment

→

04
13-15 / collusion model

01	Stake update	$S_{t+1} = S_t + \tau I$	3
02	Contribution score	$\Delta F^T H \Delta F$	4
03	Ledger inputs	W / S / T	5
04	Consensus	$\sigma(\rho(T^T s - \kappa))$	6
05	Incentive	$R \odot C$	6
06	Bond ownership	$B = [b_{ij}]$	7
07	Accumulation	$A_{t+1} = A_t + \mathrm{diag}(S)W$	7
08	Redistributed update	$S_{t+1} = S_t + \tau \Delta S$	7
09	Local composition	$y_i = f_i(G(x))$	11
10	Sparse combination	$\Sigma_j g_j(x) f_j(x)$	11
11	Distillation	$KL(p_T \parallel p_S)$	12
12	Contribution revisited	$\Delta F_i = [0,\ldots,-f_i,\ldots,0]$	4

SOURCE AUTHOR-YEAR REFERENCES

Topic	References preserved from 1.docx
Market rationale	Schwartz et al. [2019]; OpenAI [2020]; Devlin et al. [2019]; Radford et al. [2019]; Chollet [2019].
Model and scoring	Hinton et al. [2015]; LeCun et al. [1989]; Yu et al. [2017].
Modalities and data	Kaiser et al. [2017]; Nugent and Molter [2014]; Lample and Conneau [2019].
Routing and extraction	Shazeer et al. [2017]; Ryabinin and Gusev [2020]; Sanh et al. [2020].

The supplied document is headed *Bittensor: A Peer-to-Peer Intelligence Market*, carries the line "Allocate value, incentivize pricing", and also uses the title "AVIP Whitepaper". This edition preserves its complete substantive content in English, with source equation and figure numbering for comparison.

The official whitepaper credits Yuma Rao. AVIP deployment and Bittensor affiliation are not established by the supplied material. Missing source bibliography, Appendix 12.2 and footnote 6 text are not invented.

Official whitepaper | Hinton et al. / distillation check | Martens / curvature check