

---

# BitTensor: An Intermodel Intelligence Measure

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1 The dominant approach to creating a machine learning system is to collect a dataset  
2 of training examples that demonstrate correct behavior for a desired task, then  
3 train that system to imitate these behaviors [1]. The system’s performance on  
4 that task is then measured against existing systems as benchmark. We argue that  
5 using labeled datasets creates single-subject matter experts that generalize rather  
6 poorly, with self-contained knowledge that cannot be shared with other models. To  
7 reconcile this, we propose BitTensor: a framework in which other models measure  
8 the informational significance of their peers across a network. Each model uses a  
9 digital ledger to negotiate their significance to the network. In doing so, BitTensor  
10 measures intelligence of each model with respect to intelligence itself, rather than  
11 via a projection onto labeled datasets. Additionally, we show how competition for  
12 connectivity in the network can be used to force honest bidding. We first prove  
13 that selecting inter-model scores using gradient descent is a regret-free strategy:  
14 one which generates the best subjective outcome regardless of the behavior of  
15 others. We then empirically show that when nodes apply this strategy, the network  
16 converges to a ranking that correlates with the one found in a fully coordinated and  
17 centralized setting. The result is a fair mechanism for training an internet-wide,  
18 decentralized and incentivized machine learning system. Such a system produces  
19 – on an ongoing basis – a continually hardening and expanding benchmark at the  
20 generalized intersection of the participants.

21 The best in-class machine learning systems today boast high performance and trend-setting bench-  
22 marks at specific tasks for which they are trained. They utilize representations of inputs that are  
23 semantic, disjoint, and general enough to tackle a given task while maintaining their integrity [1, 2].  
24 However, when dealing with unexpected tasks or slight changes in data distribution these models  
25 become brittle and sensitive, making them narrow experts rather than resilient generalists [1]. This is  
26 due to the models’ reliance on training feedback mechanisms that benchmark them against examples  
27 of data rather than direct knowledge [3].

28 Machine learning systems are trained on a dataset of training examples that demonstrate desired  
29 behavior for a given task, and then its intelligence (i.e. performance) is then evaluated on independent  
30 and identically distributed examples [1, 4]. Finally, the system’s performance on that task is measured  
31 against existing systems as a “performance benchmark”. In contrast, if the mechanism is misaligned  
32 to the salient feature being produced, then a large portion of the work is lost as a result. This low  
33 resolution and sensitivity to changes in data is further compounded when considering the following:

- 34 1. Lack of breadth: the narrowness of the domain described by the tasks make it easier for  
35 models to “buy progress” without improved general understanding [4].
- 36 2. Lack of inter-model knowledge transfer: measuring how much one model could improve  
37 another is not possible due to the inherent incompatibility between model architectures.

After training, these models become very good at performing only that desired task. We argue that the measure of intelligence should be – in information-theoretic terms – with respect to *intelligence* itself rather than via a projection onto labeled datasets. Further, the lack of inter-model knowledge transfer ensures that the field is inherently non-collaborative, with the majority of researchers world-wide in essence contending against each other [5]. Ultimately, the only users capable of creating the systems with high performance are those with the means and resources to train the largest models.

We present a novel framework in which a collective of intelligence systems can be used to evaluate their peers. By using intelligence itself as the measure of a model’s intelligence, this allows models to share knowledge with each other and removes the need to re-train models when new information arrives. Finally, this turns the performance benchmark into an intelligence market that rewards models that improve knowledge within it.<sup>1</sup> We design the benchmark to run in a continuous and asynchronous fashion across a peer-to-peer (P2P) network. Since this introduces trustless computation<sup>2</sup> we also dedicate a part of this paper to explaining how the system remains fair when little assurance can be given about the computers that compose it.

The contributions of this paper are thus three-fold:

1. We introduce a P2P based intelligence framework.
2. We show how models can share knowledge to increase each other’s performance and remove the need to re-train models each time new data arrives.
3. We propose turning typical data-based performance benchmarks into an intelligence market, in which models that improve knowledge are rewarded.

The rest of this paper is organized as follows: Section 1 presents the intelligence benchmark and ranking mechanism that models use to evaluate each other. Section 2 considers the scenario where participants are not honestly reporting the significance of their peers, and proposes a method to detect such behavior. Section 3 highlights the experiments performed to verify the function of the framework. Finally, Section 4 presents a summary and future works on this system.

## 1 Methodology

### 1.1 Benchmark

The network is composed by  $n$  unique parameterized functions  $F = f_0, \dots, f_j, \dots, f_n$  where each function is producing an output tensor  $f_i(F(x))$ , a ‘representation’ from an input tensor  $F(x) = [f_0(x) \dots f_n(x)]$  gathered by querying its neighbors. Each function is training asynchronously over a dataset  $D_i = [X, Y]$  such that, given an error function  $\mathcal{Q}_i$ , its expectation over that data  $E_{D_i}$  defines a loss  $\mathcal{L}_i = E_{D_i}[\mathcal{Q}_i(y, f_i(F(x)))]$ . We assume these losses are measured on the same scale and thus our benchmark  $\mathcal{B}$  can be defined by their sum:

$$\mathcal{B} = \sum_i^n \mathcal{L}_i \quad (1)$$

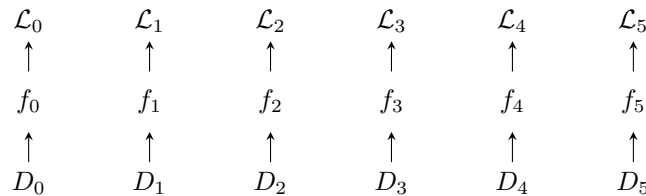


Figure 1:  $n = 5$  parameterized functions with losses  $\mathcal{L}_i$  and datasets  $D_i$ .

<sup>1</sup>“The iron rule of nature is: you get what you reward for. If you want ants to come, you put sugar on the floor.” - Charlie Munger

<sup>2</sup>In decentralized systems, trust is shifted from the individuals to the network itself, “trustless” usually means “minimal trust is required” w.r.t the protocol

Each parameterized function is represented here in its most abstract sense[6] and need only accept the same input type  $x$  and produce the same output dimension to fit within the network. For instance, unicode encoded strings  $x$ , and responses: their semantic representations within some standard dimension. This widened scope ensures participants can be multi-task [7], use completely distinct computing substrates [8] or train on unique datasets. [9].

## 1.2 Ideal Ranking

Our goal in this work is to produce a ranking  $R = [R_i]$  over these functions where the score  $R_i \in R$  represents participant  $i$ 's information-theoretic significance to the benchmark  $\mathcal{B}$ . Following Le Cun and others [10, 11], it is reasonable to analytically define this significance by equating it with the cost of removing each component from the network:

$$R_i \approx \frac{1}{n} \sum_j \sum_{x \in D_j} \Delta F^T(x)_i * H(Q_j(x)) * \Delta F(x)_i \quad (2)$$

$$\Delta F(x)_i = [0, \dots, 0, -f_i(x), 0, \dots, 0]$$

Where the above is derived using a Taylor series (Appendix 6.1) and  $\Delta F(x)_i$  is the perturbation of the inputs when removing function  $f_i$  at the point  $x$ . Note, the linear and higher order terms of the Taylor series have been removed following [11] and the remaining term  $H(Q_i)$  is the hessian of our error function. When the error function  $Q$  is the twice-differentiable cross-entropy, then  $H(Q_i)$  is the Fisher information matrix, and  $R_i \in R$  is measured as relative entropy: reflects each participants informational significance to the network as a whole.

## 1.3 Inter Ranking

It is not possible to compute the ranking score above without access to the parameters of each function in the network. Instead, we use a set of inter-model weights  $W = [w_{ij}]$  where each  $w_{i,j}$  is the score attributed to  $f_j$  from  $f_i$  combined into an  $n \times n$  square matrix.

$$w_{ij} = \sum_{x \in D_i} \Delta F^T(x)_j * H(Q_i(x)) * \Delta F(x)_j \quad (3)$$

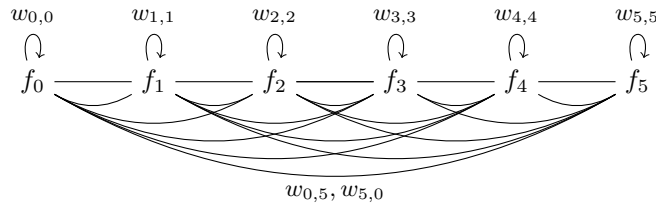


Figure 2: Inter-model contribution weights:  $w_{i,j}$  the score attributed to  $f_j$  from  $f_i$

The weights can be computed on the fly either approximately [11], or by using the full hessian of the error. We store them on a distributed ledger and allow participants to update them by making changes of bounded size:  $W^{t+1} = W^t + \lambda \Delta W$ , where  $\|W_i\|_2 < \epsilon$  at block step  $t$ . We also enforce that the scores in each row sum to 1,  $\|w\|_1 = 1$ . The equivalent ranking  $R$  (2) can then be computed by normalizing column sum of the weight matrix:

$$R = \frac{1}{n} W^T * \mathbb{1} \quad (4)$$

96 The problem is of course is that without system-wide access to the model parameters the computation  
 97 of  $w_{ij}$  (3) is non-auditable. It is reasonable to assume participants will select weights which artificially  
 98 increase their own rank rather than others in the network. Moreover, since the network remains open,  
 99 participants may choose to create many spuriously neighbours and rank themselves higher. The  
 100 remainder of this paper describes our proposal for resolving these issues.

#### 101 1.4 Stake

102 The proposed solution begins by introducing a finite resource  $S = [s_i]$ , a component's 'stake' in the  
 103 system, and an inflation mechanism  $\tau$  which translates the ranking vector  $R$  into additional stake as  
 104 incentive.

$$R = \frac{1}{n} W^T \circ S * \mathbb{1} \quad (5)$$

$$S^{t+1} = S^t + \tau * \frac{R}{||R||_2} \quad (6)$$

105 The  $\circ$  here is the Hadamard product between the  $n \times n$  matrix  $W$  and the  $n \times n$  matrix containing  $S$   
 106 in each column, and  $t$  is the time-step referred to in Section 2.3 (measured in distinct blocks on the  
 107 distributed ledger). By design (5) increases the importance of those with stake,  $s_0 * w_{ij}$ . This serves  
 108 two purposes:

- 109 1. Since it is finite, new computers cannot spuriously create nodes to game the ranking.
- 110 2. The resource provides mechanism power.

111 By providing it to nodes with large rank, this ensures that those with weight must have worked to  
 112 attain it, or indirectly subsidized those who have done so already. A single staked token would be  
 113 enough to bootstrap the process.<sup>3</sup>

---

#### Algorithm 1 Inflation mechanism

---

**Require:**  $S = [n \times 1]$  Stake Vector  
**Require:**  $W = [n \times n]$  Weight Matrix  
**Require:**  $\tau > 0$  inflation rate  
**while** TRUE **do**  
      $W = W + \lambda \Delta W$   
      $R = \frac{1}{n} W^T \circ S * \mathbb{1}$   
      $S = S + \tau * \frac{R}{||R||_2}$   
**end while**

---

#### 114 1.5 Competitive weights

115 While stake provides some protection against malicious actors, it does not ensure weights are set  
 116 accurately. Our solution begins by introducing competition for connectivity within the network.  
 117 Nodes that underweight are punished by having inputs from the network masked to zero (7). To frame  
 118 this market we borrow the continuous differential activation function  $\sigma$  with range  $(0, 1)$ . Under a  
 119 choice of weights  $W_i$  the inputs to component  $i$  are:

$$F_W(x) = [f_0(x) * \sigma(s_i * w_{i,0} - \mu_0), \dots, f_n(x) * \sigma(s_i * w_{i,n} - \mu_n)] \quad (7)$$

---

<sup>3</sup>The propagation of a resource mocks the flow of the brain-derived neurotrophic factor (BDNF). [12]

$$\sigma = \frac{1}{1 + e^{-\frac{x}{T}}} \quad (8)$$

Here, the shift term  $\mu_j$  is the average of the weights in each column  $\mu_j = (\frac{1}{n}) \sum_i s_i * w_{i,j}$ , and the activation function is the temperature scaled sigmoid. Because the allocation mechanism is standard across the network it is possible for each participants to compute both  $\frac{\partial \mathcal{L}_i}{\partial W_i}$  and  $\frac{\partial R_i}{\partial W_i}$ . Computers may augment their usual training framework, for instance, Tensorflow, with the allocation mechanism shown here.

## 1.6 Running the network

The steps to run a network participant are:

- 1 Participant defines its dataset  $D_i$ , loss  $\mathcal{L}_i$  and parameterized function  $f_i$
- 2 At each training iteration the participant broadcasts batches of examples from  $D_i$  to its peers  $[batch\_size, x]$
- 3 Responses  $F(x)$  from the network produce a loss-gradient  $\frac{\partial \mathcal{L}}{\partial F}$  which back-propagates through  $f_i$  and out to the network.
- 4 During 2 and 3 the participant competitively selects the weights for their row  $w_{ij} \in W$ .
- 5 Participants submit changes to the weights  $\Delta W_i$  which, in-turn changes the ranking and induces inflation  $\tau * R$ .
- 6 Participants disconnect and verify the model in a normal manner.

Peers only communicate with computers that hold stake as a consequence of section 2.5. Those that fail to produce value will be pruned naturally as participants learn to differentiate signal from noise.

## 1.7 Extracting knowledge

Inter-node dependence in the network is broken using distillation[6], a compression and knowledge technique in which a smaller model – the student - mimics the behaviour of an ensemble. We employ this technique over the gating ensemble (10) where the student model learns to minimize the cross-entropy (shown below as KL) between the logits produced by the gating network and its predicted distribution. [13]

$$\text{distillation loss} = \text{KL}_D(\text{dist}(x), G(x)) \quad (9)$$

We use the distilled model as proxy to cut recursive calling between each components rather than query farther into the network. If models go offline, their peers can use their distilled versions in-place. Private data can be validated over the distilled models instead of querying the network. Eventually, components can fully disconnect from the network using the distilled inputs to validate and inference the models offline.

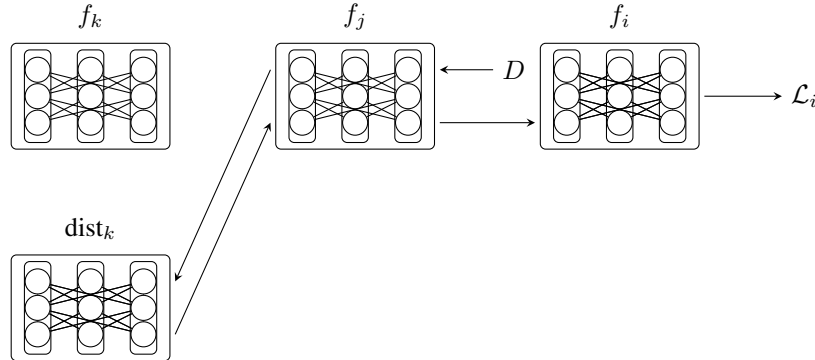


Figure 3: Queries propagate to depth=1 before the distilled model is used.

## 2 Analysis

We consider the scenario where participants are not honestly reporting the significance of their peers. The network is progressing at discrete timesteps  $t$  and it is reasonable to assume that each participant is attempting to maximize their subjective payoff over these steps.

### 2.1 Payoff Model

The staking system in Section 2.4 gives participants incentive to maximize their self-weight  $w_{ii}$  while the competitive connectivity described in Section 2.5 makes it costly for participants to decrease the remaining weights  $w_{ij}$  in their row. Since the row must sum to 1, we have a trade-off in two terms:

1. A utility term attached to the loss  $U(\mathcal{L}(W))$ .
2. The token emission via inflation  $\tau * R_i(W)$ .

Both of these are functions of the weights and, without loss of generality, measured in similar units:

$$P_i(W) = U_i(\mathcal{L}_i(W)) + \tau * R(W)_i \quad (10)$$

It is reasonable to assume payoff maximizing participants will use  $P_i$  as their objective during training. This can be computed using standard tools since both terms  $U(\mathcal{L}_i(W))$  and  $R(W)_i$  are fully continuous and differentiable. The system can be characterized as a competitive gradient descent game where participants are making steps  $\Delta W_i = \frac{\partial P_i}{\partial W_i}$ . In appendix 6.2 we prove that this strategy is regret-free and achieves the best expected payoff in hindsight. This assumption is also generally employed in smooth markets [14]. The iterative descent thus follows:

$$W^{t+1} = W^t + \lambda \Delta W \quad (11)$$

$$\Delta W = [\frac{\partial P_0}{\partial W_0}; \dots; \frac{\partial P_n}{\partial W_n}] \quad (12)$$

### 2.2 Empirical Model

Without access to the running network we evaluate our system using an empirical model. To derive the gradient steps in this model  $\frac{\partial P_i}{\partial W_i}$  we make the following assumptions:

1. The utility functions are continuous-differentiable and can be approximated by their first order derivatives  $\frac{\partial U}{\partial \mathcal{L}} = \alpha$ .
2. The network is converged to a local minimum in the inputs  $\frac{\partial \mathcal{L}}{\partial F} = 0$ .

The first assumption is approximate for small changes of continuous functions and the second assumption is realistic after extended training. In Appendix 6.2 we derive the following gradient step:

$$\frac{\partial P}{\partial W} = \frac{\alpha}{\tau} * \frac{\partial L}{\partial W} + \frac{\partial R}{\partial W} \quad (13)$$

$$\frac{\partial L}{\partial W} = \frac{\partial}{\partial W} [(F_W - F_{W_0})^T * H(\mathcal{L}(F)) * (F_W - F_{W_0})] \quad (14)$$

180 Here  $F_W$  is the masked inputs from Section 2.5.  $(F_W - F_{W_0})$  is the difference in the mask between  
 181 the choice of weights  $W$  and the weights at the minimum  $W_0$  and  $H(\mathcal{L}(F))$  is the  $[n \times n]$  hessian of  
 182 the loss over inputs  $F$ . This formulation is both intuitive and useful: the gradient term  $\frac{\partial L}{\partial W}$  measures  
 183 the change in loss for any choice of weights. Additionally, it also allows us to compute the ranking  
 184 for simulated hessian terms.

185 The remainder of the network is deterministic, and can be described by a choice of  $\theta =$   
 186  $[\alpha, \tau, \lambda, n, \sigma, T, S, W, H]$ . For instance, the secondary term,  $\frac{\partial R}{\partial W}$  can be computed-directly from  
 187 Section 2.4 and only depends on the stake vector  $S$  and weights  $W$ .

### 188 3 Experiments

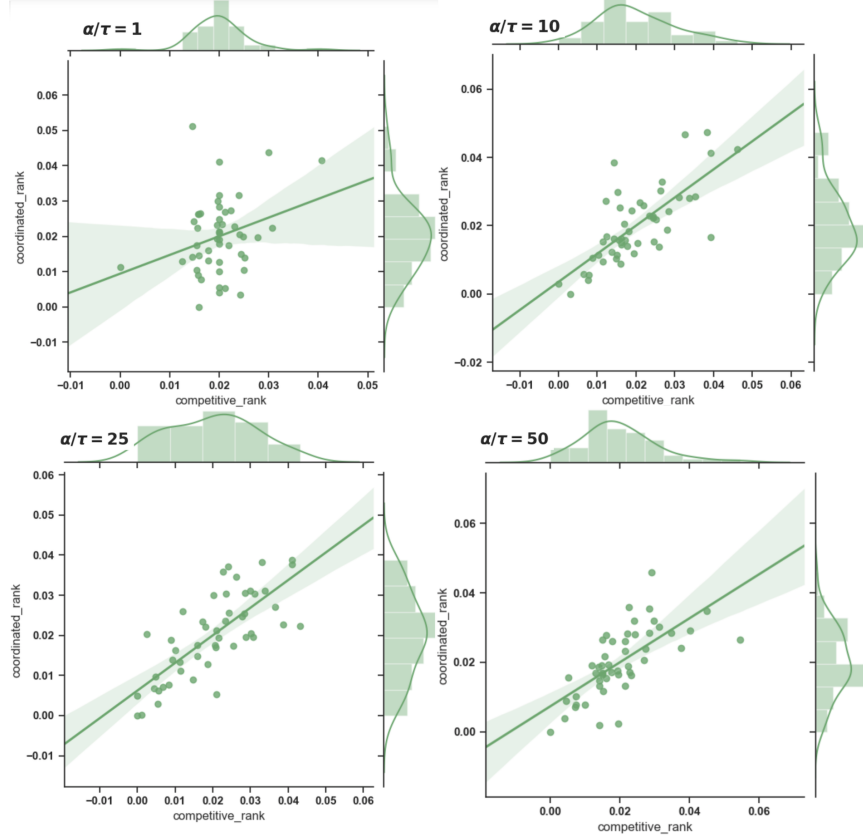


Figure 4: Correlations between the competitive rank and coordinated rank for  $\frac{\alpha}{\tau} \in \{1, 10, 25, 50\}$ . For low values of  $\frac{\alpha}{\tau}$  the weights converge to the identity: the state where peers are fully disconnected.

189 To generate sample statistics from the network, we first select  $[\alpha, \tau, \lambda, S, \sigma, T, n] \in \theta$ , then we  
 190 generate random positive semi-definite Hessians  $H$ , and random uniform initial weights  $W_0$ . For each  
 191 parameterization we discover the competitive ranking by converging the system to a Nash-equilibrium:  
 192 an equilibrium where no individual can vary from their set of weights and stand to gain. [15] To find  
 193 these equilibrium, we use competitive descent strategy described in (14) and compute the gradient  
 194 terms from (15). In each trial we use a learning rate  $\lambda = 0.005$  and stop when the gradient terms are  
 195 bounded by  $\epsilon$  or the steps exceed  $1000 \times n$ . The competitive ranking  $R^*$  at this point follows from  
 196 (4) and can be compared to the idealized score in  $R$  from (2).

197 We are interested in the correlation between  $R^*$  and  $R$  as we vary the ratio  $\alpha$  and  $\tau$ , this ratio is  
 198 explicit in (15) where the fraction relates the two gradient terms. Intuitively the ratio is between the

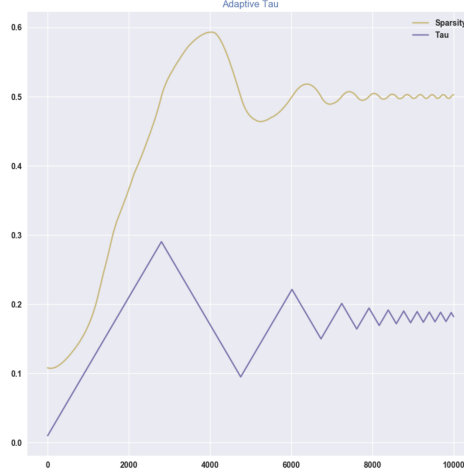


Figure 5: (i) Left: the ratio between the main diagonal and the remainder of the weights. Right: the adaptive  $\tau$  parameter converging onto the target. The weight matrix sparsity is a proxy for the ranking accuracy which we see in Figure 4. As sparsity converges onto the target the ranking is also improving.

value of minimizing the loss and maximizing revenue from inflation. Since this effects the ranking, we show this trade-off for various choices in Figure 4.

Finally, in Figure-5 we implement an adaptive- $\tau$  strategy where the network varies the inflation rate. Initial inflation is zero and then increases until the weights begins to converge towards the main diagonal  $w_{ii} = 1$ . We measure the sparsity  $sparsity = \text{sum}(W_{dg}) / \text{sum}(W)$ , the ratio between the main diagonal and the remaining weights. As sparsity increases we push the market equilibrium by decreasing  $\tau$ . Figure shows this adaptive convergence for  $\alpha = 1$  with a sparsity target of 1.

### 3.1 Discussion

Figure 4 shows the relationship between the idealized rank and the competitive rank as a function of  $\frac{\alpha}{\tau}$ . Figure-4-a,  $\frac{\alpha}{\tau} = 1$  shows the case where this ratio negatively effects the ranking accuracy. Here, all components have set  $w_{ii} = 1$  and the resulting scores for all participants has converged to  $1/n$ . When this occurs, the system could decrease the inflation rate  $\tau$  and push the network towards the high information markets seen in Figure-4-b, c, and d. Figure-5 shows a basic implementation of this where  $\tau$  adapts to the ratio between the row-sums and the main-diagonal. By lowering inflation, it subsequently 'costs less' to connect with peers. Profit maximizing nodes automatically adjust to the change and the system converges back towards an accurate ranking. Those with high ranks will oppose inflation decreased, while those with low ranks will welcome it. The equilibrium found in this meta game will most certainly depend on the number of participants, key to both the ranking accuracy and the market at its core. However, we leave this analysis for a follow up paper.

## 4 Conclusion

We have proposed an inter-model benchmark that can run in a P2P setting outside of a trusted environment. We started with a typical machine learning framework defined by a set of functions with their losses over given datasets, then derived an idealized ranking score. The measure produced an information theoretic score that new participants can improve by learning how to be useful to their peers. However, the system is incomplete without a mechanism that prevents participants from ranking dishonestly. To resolve this, we proposed an incentive scheme and a differential allocation system over the network weights. The system allows the participants to train for connectivity in the graph. Following this, we described how to increase the outward bandwidth in the system using a trainable gating network and how to cut independence between nodes using distillation. Finally we showed how increasing the number of nodes in the system and fixing the inflation mechanism



properly ensured that the resulting rank scores would correlate with those found in a idealized setting. While this is true, stake in the system holds value as a means to drive what the network learns. That benchmark is continually being solved by the participants, compounding what has been learned before and making it available to new learners in the system.

## 5 Broader Impact

In order to provide a balanced perspective, authors are required to include a statement of the potential broader impact of their work, including its ethical aspects and future societal consequences. Authors should take care to discuss both positive and negative outcomes.

## References

- [1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
- [2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [3] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.
- [4] F. Chollet, “On the measure of intelligence,” *arXiv preprint arXiv:1911.01547*, 2019.
- [5] M. Riabinin and A. Gusev, “Learning@home: Crowdsourced training of large neural networks using decentralized mixture-of-experts,” *arXiv preprint arXiv:2002.04013*, 2020.
- [6] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [7] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, “One model to learn them all,” 2017.
- [8] M. A. Nugent and T. W. Molter, “Cortical processing with thermodynamic-ram,” 2014.
- [9] G. Lample and A. Conneau, “Cross-lingual language model pretraining,” 2019.
- [10] Y. LeCun, D. J. S., and S. S. A., “Optimal brain damage,” *Advances in Neural Information Processing Systems 2 (NIPS)*, 1989.
- [11] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “Nisp: Pruning networks using neuron importance score propagation,” 2017.
- [12] B. S and D. UN, “Brain-derived neurotrophic factor and its clinical implications,” *Arch Med Sci*. 2015;11(6):1164–1178. doi:10.5114/aoms.2015.56342, 2015.
- [13] L. C. J. W. T. Sanh, Victor; Debut, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *arXiv preprint arXiv:1910.01108*, 2019.
- [14] D. Balduzzi, W. M. Czarnecki, T. W. Anthony, I. M. Gemp, E. Hughes, J. Z. Leibo, G. Piliouras, and T. Graepel, “Smooth markets: A basic mechanism for organizing gradient-based learners,” 2020.
- [15] P. Dütting, Z. Feng, H. Narasimhan, D. C. Parkes, and S. S. Ravindranath, “Optimal auctions through deep learning,” 2017.

## 6 Appendix

### 6.1 Deriving the idealized ranking.

We approximate the change in the benchmark  $\Delta\mathcal{B}$  at a local minimum under a perturbation  $\Delta F(x)_i = [\dots, -f_i(x), \dots]$  reflecting the removal of the  $i^{th}$  node.

$$\Delta\mathcal{B} = \mathcal{B}(F + \Delta F_i) - \mathcal{B}(F) = \sum_i^n \mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F) \quad (15)$$

$$\mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F) \approx \frac{\partial \mathcal{L}_i}{\partial F} * \Delta F_i + \frac{1}{2} \Delta F_i^T * H(\mathcal{L}_i) * \Delta F_i + O(\Delta F_i^3) \quad (16)$$

(17) follows from the definition of the benchmark  $\mathcal{B}$  and (18) follows from a Taylor series under the perturbation  $\Delta F(x)_i$ . Note that the first term  $\frac{\partial \mathcal{L}_i}{\partial F}$  is zero at the local minimum and the higher order term  $O(\Delta F_i^3)$  can be

271 ignored for sufficiently small perturbations. These assumptions are also made by [10] and [11]. Note that  $\mathcal{L}_i$  is  
 272 an expectation over the dataset  $D_i$ , and all terms are evaluated at a point  $x$  so we have:

$$\Delta \mathcal{B} \approx \sum_i^n \sum_{x \in D_i} \Delta F_i^T(x) * H(\mathcal{Q}_i(x)) * \Delta F_i(x) \quad (17)$$

273 Here the hessian over the error function  $H(\mathcal{Q}_i(x))$  and the summation over the dataset  $\sum_{x \in D_i}$  have been  
 274 appropriately substituted. The constant factor  $\frac{1}{2}$  can be removed and this leaves our result.

## 275 6.2 Deriving the weight convergence game.

### 276 6.3 Theorem

277 For choice of Hessians  $H(\mathcal{L}(F))$  the network convergence-game can be described with the following linear  
 278 relationship between gradient terms:

$$\frac{\partial P}{\partial W} = \frac{\alpha}{\tau} * \frac{\partial L}{\partial W} + \frac{\partial R}{\partial W} \quad (18)$$

279

280 With the gradient of the loss:

$$\frac{\partial L}{\partial W} = \frac{\partial}{\partial W} [(F_W - F_{W_0})^T * H(\mathcal{L}(F)) * (F_W - F_{W_0})] \quad (19)$$

281

### 282 6.4 Setup

283 We analyze the system by characterizing the behaviour of participants via their payoff in two terms:

- 284 1. The utility attached to that participant's loss as a function of their weights  $U(L(W))$ .  $U$  is assumed  
 285 roughly linear for small change in the weight matrix,  $U(\mathcal{L}) = \alpha * \mathcal{L}$ , and  $\frac{\partial U}{\partial \mathcal{L}} = \alpha$ .
- 286 2. The network is converged to a local minimum in the inputs  $\frac{\partial \mathcal{L}}{\partial F} = 0$ .

287 (1) and (2) From the payoff formulation in 6.3 we write:

$$P(W) = \alpha * L(W) + \tau * R(W) \quad (20)$$

288

289 Note, the utility function and emission were measured in similar units and so  $\alpha$  is the *price* of each unit change  
 290 in loss. The analysis just supposes such a score exists, not that it can be computed. Participants are selecting  
 291 their weights by making gradient steps  $\Delta W_i = \frac{\partial P_i}{\partial W_i} = \frac{\partial U_i}{\partial W_i} + \tau * \frac{\partial R_i}{\partial W_i}$  as to maximize their local payoff. For  
 292 brevity we omit the subscript  $i$  for the remainder of the analysis. Consider a Taylor expansion of the loss under a  
 293 change  $\Delta F$  in the inputs.

$$\mathcal{L}(F + \Delta F) = L(F) + \frac{\partial \mathcal{L}}{\partial F} \Delta F + \frac{1}{2} \Delta F * H(\mathcal{L}(F)) * \Delta F + O(\Delta F^3) \quad (21)$$

294 The first linear term  $\frac{\partial \mathcal{L}}{\partial F}$  is zero and the higher order terms are removed for sufficiently small perturbations in  $F$ .  
 295 We then perform a change of variable  $F = F_{W_0}$ , and  $\Delta F = F_{W_1} - F_{W_0}$  where  $W_0$  are the current set of  
 296 weights and  $W_1$  are another choice such that  $F_{W_0}$  and  $F_{W_1}$  are those inputs masked by  $W_0$  and  $W_1$  according  
 297 to (7). Substituting this into (23):

$$\mathcal{L}(F_{W_1}) = L(F_{W_0}) + \frac{1}{2} (F_{W_1} - F_{W_0})^T * H(\mathcal{L}(F)) * (F_{W_1} - F_{W_0}) \quad (22)$$

298 The function  $\mathcal{L}(F_{W_1})$  is simply an approximation of the loss for any choice of weights  $W_1$  given that the  
 299 network has converged under  $W_0$ . Finally, by the  $\alpha$ -linear assumption of the utility we can attain the following:

$$\frac{\partial U}{\partial W} = \alpha * \frac{\partial L}{\partial W} = \alpha \frac{\partial}{\partial W} [(F_W - F_{W_0})^T * H(\mathcal{L}(F)) * (F_W - F_{W_0})] \quad (23)$$

300 Note that we've dropped the subscript  $W_1$ ,  $L(F_{W_0})$  is constant not depending on the choice of weights, and the  
 301 fraction  $\frac{1}{2}$  has been removed. The remaining term  $\frac{\partial R}{\partial W}$  is derivable via the ranking ranking function in Section  
 302 2.3. Finally, dividing both terms by  $\tau$  shows the result:

$$\frac{\partial P}{\partial W} \approx \frac{\alpha}{\tau} * \frac{\partial L}{\partial W} + \frac{\partial R}{\partial W} \quad (24)$$

$$\frac{\partial L}{\partial W} = \frac{\partial}{\partial W} [(F_W - F_{W_0})^T * H(\mathcal{L}(F)) * (F_W - F_{W_0})] \quad (25)$$

303

## 304 6.5 Deriving the ex-post zero-regret step.

305 Consider the system described above. A set of  $n$  nodes are changing the weights in the ranking matrix  $W$   
 306 iteratively using gradient descent with learning rate  $\lambda$ .  $W^{t+1} = W^t + \lambda \Delta W$ . Here, the change of weights is  
 307  $\Delta W = [\Delta w_0, \dots, \Delta w_n]$  where each  $\Delta w_i$  is the weight change pushed by node  $i$ . Each node is attempting to  
 308 competitively maximize it's payoff as a function of the weights  $P_i(W)$ .

### 309 6.5.1 Definition

310 The ex-post regret for a single step is the maximum difference in loss between the chosen step  $\Delta w_i$  and all  
 311 alternative  $\Delta w_i^*$ . The *expected* ex-post regret is this difference in expectation, where the expectation is taken  
 312 over all choices  $\Delta w_j$ 's chosen by other participants [15].

$$\text{rgt}_i = E_{\Delta w_j} [\max_{\Delta w_i^*} [P_i(\Delta w_i^*) - P_i(\Delta w_i)]] \quad (26)$$

### 313 6.5.2 Theorem

314 For sufficiently small  $\lambda$ , the expected ex-post regret for strategy  $\Delta w_i = \frac{\partial P}{\partial w_i}$  is 0.

### 315 6.5.3 Proof

316 Consider Taylor's theorem at the point  $W$  for the payoff function  $P$  under a change in weights  $W^* = W + \lambda \Delta W$ .  
 317 There exists a function  $h(W^*)$  such that in the limit as,  $W^* \rightarrow W$  we have the exact equivalence:

$$P(W^*) = P(W) + \frac{\partial P}{\partial W}(W^* - W) + h(W^*) \quad (27)$$

318 Let  $P(W_*)$  represent the payoff when the weight change of the  $i^{th}$  row is  $\Delta W_i = \frac{\partial P}{\partial W_i}$ , and let  $P(W^*)$  be any  
 319 other choice. By the definition of regret, and Taylors theorem as  $\lambda \rightarrow 0$ , we have:

$$\text{rgt}_i = E_{\Delta w_j} [\max_{\Delta w_i^*} [\frac{\partial P}{\partial W}(W^* - W) - \frac{\partial P}{\partial W}(W_* - W)]] \quad (28)$$

320 This follows by subtracting (29) with choice  $W^*$  and  $W_*$ . Next, substituting  $W^* - W = -\lambda \Delta W$  and expanding  
 321  $\frac{\partial P}{\partial W} \Delta W = [\frac{\partial P}{\partial W_0} * \Delta W_0, \dots, \frac{\partial P}{\partial W_n} * \Delta W_n]$  into the equation above leaves:

$$\begin{aligned} \frac{\partial P}{\partial W}(W^* - W) - \frac{\partial P}{\partial W}(W_* - W) &= \lambda (\frac{\partial P}{\partial W_i} * \Delta W_i^* - \frac{\partial P}{\partial W_i} * \Delta W_{i*}) + \\ &\quad \lambda \sum_{j \neq i}^n (\frac{\partial P}{\partial W_j} * \Delta W_j^* + \frac{\partial P}{\partial W_j} * \Delta W_{j*}) \end{aligned} \quad (29)$$

322 The constant  $\lambda$  can be removed and the second term depends only on weights of other rows  $W_{j \neq i}$ . These can be  
 323 removed under the expectation  $E_{\Delta w_j}$ . He have:

$$\text{rgt}_i = E_{\Delta w_j} [\max_{\Delta w_i^*} [\frac{\partial P}{\partial W_i} * \Delta W_i^* - \frac{\partial P}{\partial W_i} * \Delta W_{i*}]] \quad (30)$$

324 Finally, we use the the fact that for vectors  $a, b$  and angle between them  $\theta$  the magnitude of the dot product is  
 325  $|a||b|\cos\theta$ . This is maximized when the vectors are parallel  $\theta = 0$  and  $\cos(\theta) = 1$ , or  $\Delta W_i = \kappa * \frac{\partial P}{\partial W_i}$  for  
 326 some constant  $\kappa > 0$ . Thus  $P(\Delta W^*)$  is maximize when  $\Delta W_i^* = \kappa * \frac{\partial P}{\partial W_i}$ . Since  $P(\Delta W^*) = P(\Delta W_*)$  in  
 327 the maximum, this proves the point.