Adding Two Losses Hurts the Smaller One: A Loss-Ratio Law for Shared Networks

Vuk Rosić

https://github.com/vukrosic/two-losses-one-network

Preprint, November 2026

Abstract

Many networks are trained on a sum of two losses: a driving model predicts fast cars and slow pedestrians, a robot model predicts a heavy base and a light arm, and a mixture-of-experts language model adds a router balancing loss to its next-token loss. We show that the term with the smaller loss is learned worse than it would be by its own network with the same total number of weights, and that one number predicts how much worse: the loss ratio Λ, each term's loss measured with its own network. A single power law in Λ fits 519 shared networks across four different settings, including one with no physics at all. The cause is the sum itself: weighting a loss acts exactly like changing that term's units. For a rank-limited linear network we derive the harm in closed form and show that it does not depend on network size. Training longer does not remove the harm, and a larger network removes it only once separate networks stop improving with size. Dividing each loss by its own size removes it.

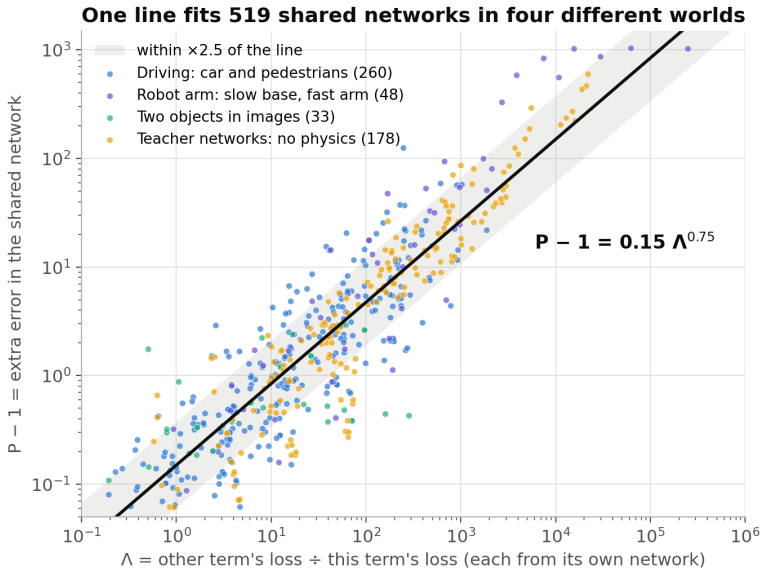


Figure 1: One law across four settings. Each point is one shared network; P is the smaller term's error in the shared network divided by its error in its own network of the same total size. The line is fitted in driving and applied to the other settings without refitting; the band marks a factor of 2.5. Points with $\Lambda < 1$ are terms whose own loss was larger than the other's.

1 Introduction

Adding losses is the default way to train one network on several outputs. When the terms differ in size, because they are measured in different units or because one target moves more than the other, the larger term supplies most of the gradient (Figure 2a). It is widely known that this can crowd out the smaller term [12, 13], and normalization methods exist to prevent it [9, 22]. What has been missing is a quantitative rule: given two terms, how much worse is the smaller one learned, and do training length or network size change the answer?

We measure this harm as P, the smaller term's error in the shared network divided by its error in a network of the same total size split into one half per term. We find that P is predicted by one number that can be measured before choosing an architecture, the loss ratio Λ. Our contributions are:

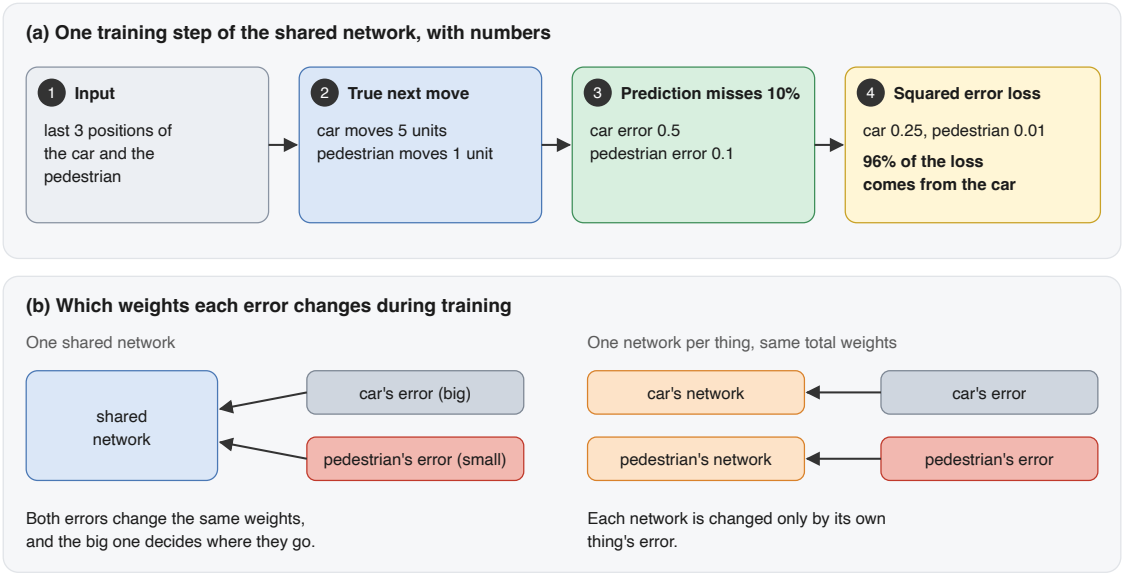


Figure 2: The mechanism. (a) In one training step of a shared driving model, the car contributes 96% of the loss and therefore most of the gradient. (b) A shared network is updated by the sum of both losses; separate networks of the same total size are each updated only by their own loss.

- **A law.** $P - 1 \approx 0.15\,\Lambda^{0.75}$ across 519 shared networks in four settings, including one with no physics at all (Figure 1, Section 4).
- **The cause.** A loss weight a^2 has the same effect as measuring a term in units a times larger; a hand split of the weights shows the shared network acts as if it gave the smaller term 10% of the weights, half of what the best split gives it (Section 5).
- **Theory.** At the optimum of any network, raising one term's weight can only raise the other term's loss. In a rank-limited linear network the harm has a closed form in which network size cancels (Section 6).
- **When training length and size help.** The shared network learns the smaller term a fixed factor slower, so longer training leaves P unchanged; size lowers P only once separate networks stop improving with size (Section 7).
- **A fix and a measurement trap.** Dividing each loss by its own size keeps $P \approx 1$ at every gap; a constant learning rate inflates the measured harm up to 3× (Section 8).

2 Related Work

Sums of losses in practice. Driving models predict all agents in a scene with one loss [11, 17, 18, 20], world models predict everything in a scene together [8, 10], and robot policies predict several joints at once [3]. Switch Transformer adds a router balancing loss with weight 0.01 [6], and DeepSeek-V3 adds a multi-token prediction loss with weight 0.3, then 0.1 [4]. None of these works compares the smaller term against its own network.

The effect and the fix are known. Khakhar and Buckman [13] show that under squared error a target at a larger scale swamps a smaller one, and replace the loss with a histogram loss; they do not compare against per-target networks of equal size or give a law for the size of the harm. PopArt normalizes value targets during training [22] and stops high-reward games from dominating a shared agent [9]; this is the same idea as our fix. Task-weighting methods rebalance losses or gradients during training [2, 12, 15, 23] and report per-task results, but no rule for how large the harm is.

Shared models acting like smaller ones. In multilingual translation, a model trained on two languages acts like a single-language model with a fraction of the weights, set by the data share and independent of model size [7]. Interference is strongest when the model is small relative to the data [21], and mixed-modality scaling laws add a competition term [1]. Our hand split measures the same kind of fraction, set by the loss ratio instead of the data share.

Learning order. Gradient descent learns large directions of a target before small ones in deep linear networks [14, 19],

and scaling can be modeled as a capacity-limited network learning the most valuable pieces first [16]. Our rank-k result builds on the Eckart–Young theorem [5].

3 Setup

Settings. Each setting has two terms whose losses differ by a controllable gap.

- **Driving.** A car and a pedestrian on a road, each described by position and velocity. The car accelerates, steers and brakes for the pedestrian; the pedestrian crosses, turns back and stops near the car. By default the car moves 5.2 times farther per step; changing its units sets the gap.
- **Robot arm.** A heavy base that moves slowly and an arm that swings fast.
- **Two objects in images.** An image with a robot we steer and a second agent with its own behavior (standing still, circling, chasing, fleeing or switching between them). A frozen encoder maps each object to 16 numbers, which the network predicts.
- **Teacher networks.** Random inputs; each target is the output of a fixed random two-layer network, and one target is scaled by the gap a. Fresh samples every step. This setting has no physics or motion, so the law cannot depend on them.

Networks and training. MLPs of depth 1 to 4, GRUs and transformers, over network sizes spanning more than two orders of magnitude. Adam with learning rate 10^{-3}, batch 256 and cosine decay to zero unless stated. Errors are measured on held-out data; in driving, relative to the error of guessing that each agent keeps its current velocity.

Measurements. For every setting and size we train one shared network on the sum of both losses, and one network of the same total size split into two halves, one per term. P is the smaller-loss term's error in the shared network divided by its error in the split network. Λ is the other term's loss divided by this term's loss, each from its own half. Runs are deterministic; neighboring settings differ by about ± 0.15 in P, which we take as the noise level.

4 The Loss Ratio Predicts the Harm

We fit the law in driving and score it without refitting in the other three settings. The driving fit is $P - 1 = 0.149\,\Lambda^{0.77}$ over 308 points ($R^2 = 0.80$). Rounded to $P - 1 = 0.15\,\Lambda^{0.75}$, it places 377 of 519 points within a factor of 2.5 (Figure 1) and holds in 31 of 35 settings. Median measured-to-predicted ratios are $\times 1.00$ in driving (260 points), $\times 1.69$ for the robot arm (48), $\times 0.91$ for two objects (33) and $\times 0.95$ for teacher networks (178). A separate fit to teacher networks alone gives $0.147\,\Lambda^{0.74}$, nearly identical to the driving fit. The four settings that miss all fall below the line.

5 The Cause Is the Sum

A loss weight equals a unit change. If one term is measured in units a times larger and the last layer is linear, its loss is exactly a^2 times larger. We trained networks both ways. Weighting a term by a^2 and rescaling its units by a give the same harm to the other term in both settings tested (Table 1). Units therefore act as a hidden loss weight.

The shared network acts like an uneven split. In the two-object setting we split the weights unevenly by hand (Table 2). The shared network's error on the smaller term (18.3%) matches a split that gives it only 10% of the weights. The split that minimizes the total loss gives it 20%, so the shared network is also worse than the best split on the total loss (0.0096 against 0.0075).

Table 1: P for the smaller term when the larger term's loss is multiplied by a^2 versus when its units are multiplied by a.

a^2	1/4	4	16	64
Driving, loss $\times a^2$	1.06	2.22	4.25	7.03
Driving, units $\times a$	1.18	1.81	3.39	6.49
Two objects, loss $\times a^2$	1.30	2.29	3.66	–
Two objects, units $\times a$	1.31	2.43	3.78	–

Table 2: Hand-split networks in the two-object setting, with the chasing agent as the smaller term.

Weights given to the smaller term	Larger-term error	Smaller-term error	Total loss
50%	6.2%	10.7%	0.0091
30%	4.5%	13.2%	0.0079
20%	3.8%	14.5%	**0.0075**
10%	3.5%	19.0%	0.0083
Shared network	4.9%	18.3%	0.0096

6 Theory

Result 1. *For any model, data and losses L_A, L_B, let θ_w minimize $L_A + w\,L_B$. If w increases from w_1 to w_2, then $L_B(\theta_w)$ cannot increase and $L_A(\theta_w)$ cannot decrease.*

Proof. Optimality gives $L_A(\theta_1) + w_1 L_B(\theta_1) \le L_A(\theta_2) + w_1 L_B(\theta_2)$ and $L_A(\theta_2) + w_2 L_B(\theta_2) \le L_A(\theta_1) + w_2 L_B(\theta_1)$. Adding them gives $(w_2 - w_1)\big(L_B(\theta_2) - L_B(\theta_1)\big) \le 0$. Substituting back into the first inequality gives $L_A(\theta_1) \le L_A(\theta_2)$. □

Combined with the unit identity of Section 5, the term measured in larger numbers can only gain at the other's expense. Result 1 does not say how much; that requires a model of capacity.

Result 2. *Consider a linear network with k hidden units, and let the i-th direction of each term's target carry loss $c\,i^{-\beta}$ with $\beta > 1$, with constants $c_A > c_B$. Then the harm to term B is*

$$P = \left(\frac{1 + \Lambda^{1/\beta}}{2}\right)^{\beta - 1}, \qquad \Lambda = c_A / c_B. \tag{1}$$

Sketch. The best rank-k fit keeps the k directions with the most loss [5], i.e. every direction above a common threshold τ. Term B keeps $n_B = (c_B/\tau)^{1/\beta}$ directions, so $n_B = k/(1 + \Lambda^{1/\beta})$. The remaining error after n directions is about $c\,n^{1-\beta}/(\beta - 1)$. Separate networks keep $k/2$ directions each, so their loss ratio is $c_A/c_B = \Lambda$. Dividing B's error at n_B by its error at $k/2$ gives Eq. (1). □

Three consequences follow. The harm grows more slowly than Λ: for large Λ, $P \propto \Lambda^{(\beta-1)/\beta}$, an exponent below 1, consistent with the measured 0.75. Network size cancels: k does not appear, so doubling the network doubles both shares and leaves P unchanged as long as both terms still have useful directions left. Normalizing each loss sets $\Lambda = 1$ and $P = 1$. An exact top-k selection over 200,000 directions per term matches Eq. (1) to within 2% for $\Lambda \le 100$ at $k = 400$ (e.g. $\Lambda = 100$, $\beta = 2$: 5.50 measured, 5.50 predicted). For $\beta = 2$ the theory lies close to the measured law over $\Lambda = 10$ to 100 (Figure 3).

7 Training Length and Network Size

Training longer does not help. Doubling and quadrupling training leaves P unchanged within noise in driving and the two-object setting (Table 3). The shared network follows the smaller term's own learning curve slowed by a fixed factor F ($F = 1.2$ at gap 5.2 and 2.3 at gap 21; Figure 4). If own error falls as $S^{-\alpha}$ with steps S, then $P = F^{\alpha}$ at every step. The exception is a term that reaches its error floor: the robot base's P falls from 2.75 to 1.67.

Table 3: P for the smaller term at three training lengths.

Setting	6k steps	12k steps	24k steps
Driving, gap 5.3	1.30	1.33	1.41
Driving, gap 10.5	1.81	1.81	1.72
Driving, gap 21	3.39	3.04	3.08
Two objects, units ×4	3.78	3.82	3.70
Robot arm, base at its error floor	2.75	2.20	1.67

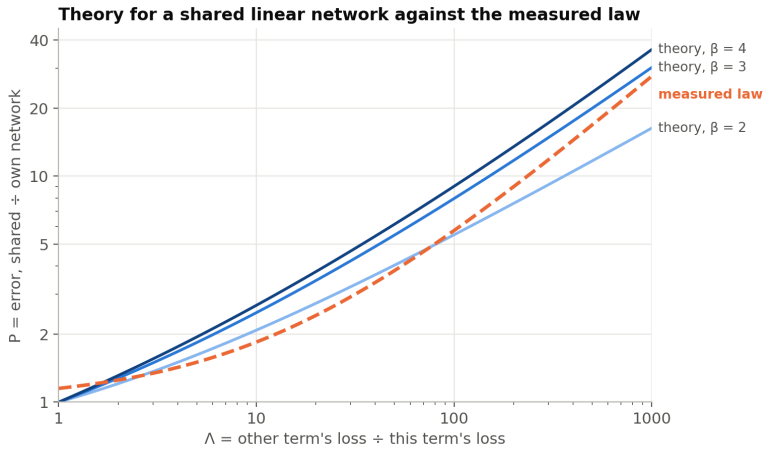


Figure 3: Eq. (1) for three values of β against the measured law. The theory assumes a linear network; the measured law comes from trained nonlinear networks.

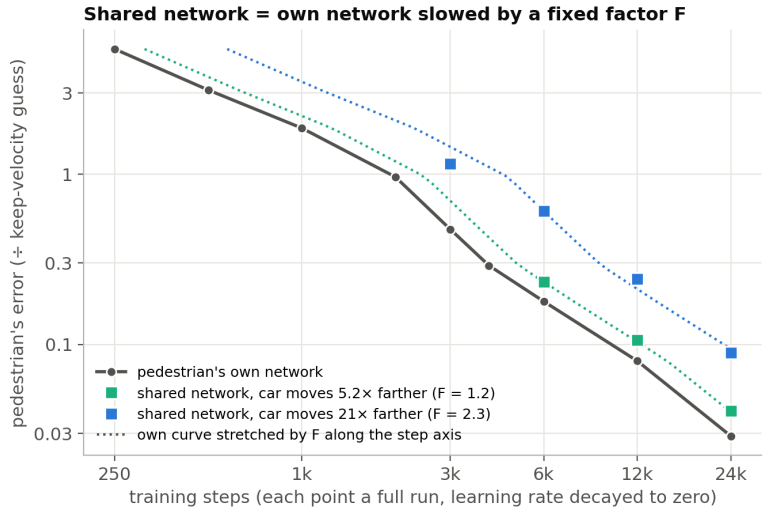


Figure 4: The shared network's error on the pedestrian (squares) lies on the pedestrian's own learning curve stretched by a fixed factor F along the step axis (dotted).

A larger network helps only when separate networks stop improving with size. Eq. (1) predicts that size cancels while both terms still have useful directions to learn. In teacher networks with gap 8, we change one knob, training length, to switch between the two regimes (Figure 5). At 300 steps the separate networks barely improve with size and P falls from 1.94 to 1.21 as the network grows 100×. At 3,000 steps the separate networks keep improving with size and P rises from 1.78 to 4.10. The same switch appears in the two-object setting: with a switching agent, own error improves with size ($N^{-0.46}$ to $N^{-0.62}$); with a chasing agent it does not ($N^{+0.07}$), and adding random pushes restores the size gain ($N^{-0.29}$). For models that are still improving with scale, size should not be expected to remove the harm.

8 The Fix, and a Measurement Trap

Normalizing each loss. In driving we divided each term's loss by its own typical size and repeated the full grid of gaps and sizes. P stays flat across gaps from 2.6 to 42, and $P - 1$ exceeds 0.1 in only 1 of 25 runs (Figure 6). Normalizing per output number instead of per term breaks training, because numbers that rarely change receive very large weights. Separate networks per term also remove the harm, provided each network is large enough and sees every input its term depends on.

A constant learning rate inflates the harm. At the same 6,000 steps in driving, the pedestrian's P is 1.81 with a

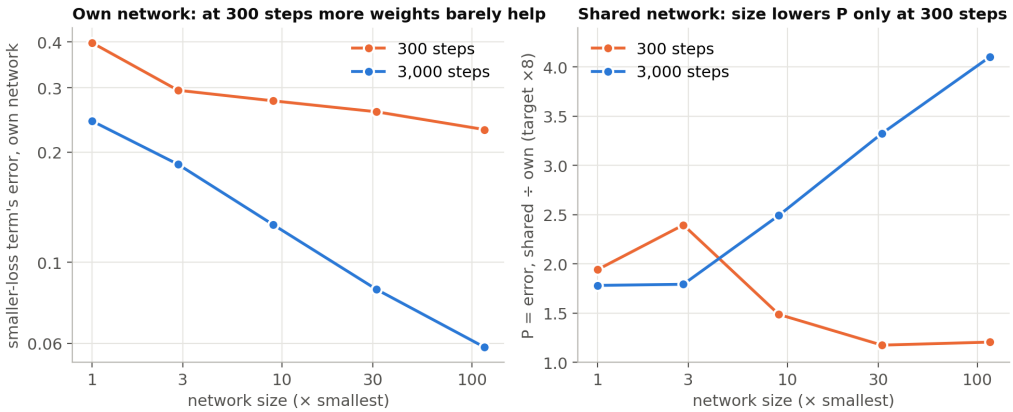


Figure 5: Teacher networks, gap 8. Left: the smaller term's own error against network size. Right: P against network size. Size lowers P only in the regime where separate networks stop gaining from size.

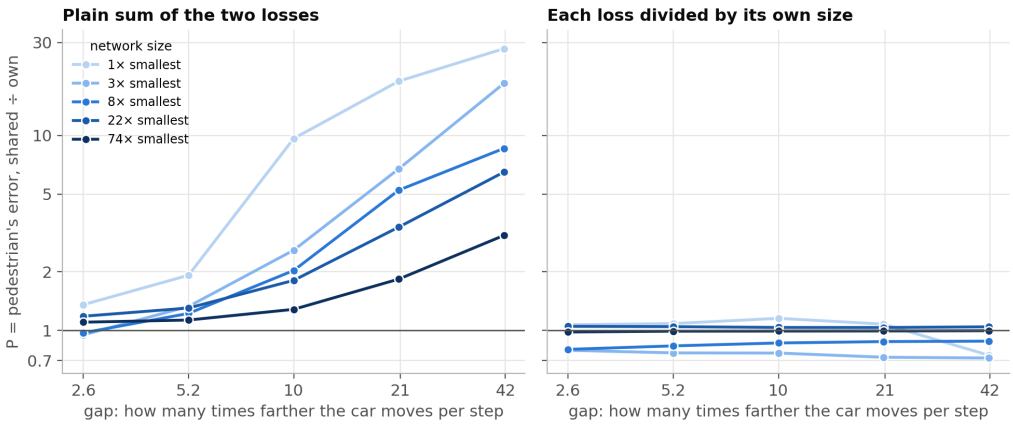


Figure 6: Driving. Left: with the plain sum, the pedestrian's P grows with the gap at every network size. Right: with each term's loss divided by its own size, P stays near 1 at every gap and size; values below 1 mean sharing helps the pedestrian.

constant learning rate, against 1.30 with decay to zero, 1.19 with a 4× larger batch and 1.23 with weight averaging over the last quarter of training (Figure 7). For the robot arm the constant rate gives 2.4 and 4.1 against 1.06 and 1.34 with decay. Any method that removes gradient noise lowers P, so comparisons between shared and split designs should decay the learning rate first. The two-object setting is an exception (1.9 against 2.0).

9 Recommendations

- **Estimate the harm before training.** Measure each term's loss with its own network; $P \approx 1 + 0.15\,\Lambda^{0.75}$ estimates how much the smaller term will lose.
- **Normalize per term.** When terms have different units or typical sizes, divide each loss by its own size, or give each term its own network.
- **Do not rely on longer training or larger models.** Neither removes the harm while the smaller term is still learning and separate networks still gain from size.
- **Report each term separately, with a decayed learning rate.** The total loss hides the smaller term, and a constant learning rate exaggerates the gap between designs.

The same sum appears in driving world models (cars against pedestrians), robot policies (large joints against grippers), and mixture-of-experts language models (next-token loss against a weighted balancing loss). The rule predicts that in each case the smaller term is learned worse than its loss weight alone suggests, unless each term's loss is normalized.

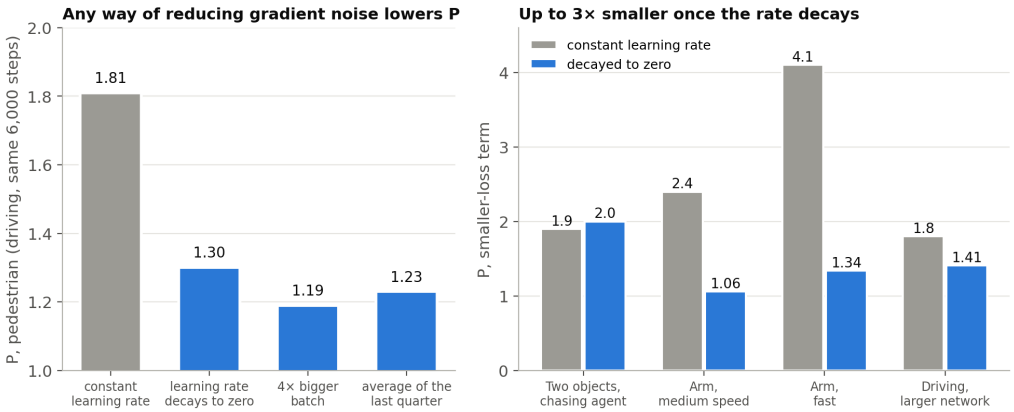


Figure 7: Left: four ways of training driving for 6,000 steps. Right: constant against decayed learning rate in three settings.

10 Limitations

- **Controlled data.** All four settings use generated data, where each term's own-network loss can be measured exactly. Tests on public driving trajectories and mixture-of-experts language models are the next step.
- **Fitted constants.** The constants 0.15 and 0.75 are fitted; depth, optimizer and network type shift the constant (depth 4 raises it about 2.4× in driving).
- **No universal size threshold.** The size rule switches in three direct tests, but the point at which separate networks stop gaining from size differs between settings.
- **Prediction, not decisions.** We have not measured whether worse prediction of the smaller term leads to worse downstream decisions, for example in a planner.

11 Conclusion

When one network is trained on a sum of two losses, the smaller term loses, and the loss ratio Λ predicts by how much: $P - 1 \approx 0.15\,\Lambda^{0.75}$ across four settings and 519 networks. The cause is the sum itself, since a loss weight is equivalent to a change of units. A rank-limited linear model reproduces the shape of the law and shows that network size cancels; experiments confirm that neither longer training nor larger networks remove the harm while the smaller term is still learning. Dividing each loss by its own size removes it. Code and every run are available at https://github.com/vukrosic/two-losses-one-network.

References

[1] Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. Scaling laws for generative mixed-modal language models. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2301.03728.

[2] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, 2018. arXiv:1711.02257.

[3] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. arXiv:2303.04137.

[4] DeepSeek-AI. DeepSeek-V3 technical report, 2024. arXiv:2412.19437.

[5] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[6] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23, 2022. arXiv:2101.03961.

[7] Patrick Fernandes, Behrooz Ghorbani, Xavier Garcia, Markus Freitag, and Orhan Firat. Scaling laws for multilingual neural machine translation. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2302.09650.

[8] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023. arXiv:2301.04104.

[9] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with PopArt. In *AAAI Conference on Artificial Intelligence*, 2019. arXiv:1809.04474.

[10] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving, 2023. arXiv:2309.17080.

[11] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong

Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. arXiv:2212.10156.

[12] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. arXiv:1705.07115.

[13] Adam Khakhar and Jacob Buckman. Neural regression for scale-varying targets, 2022. arXiv:2211.07447.

[14] Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1809.10374.

[15] Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, Shu Liu, and James T. Kwok. Dual-balancing for multi-task learning, 2023. arXiv:2308.12029.

[16] Eric J. Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.13506.

[17] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple and efficient attention networks, 2022. arXiv:2207.05844.

[18] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Ben Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified architecture for predicting multiple agent trajectories. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.08417.

[19] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. arXiv:1312.6120.

[20] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. MotionLM: Multi-agent motion forecasting as language modeling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. arXiv:2309.16534.

[21] Uri Shaham, Maha Elbayad, Vedanuj Goswami, Omer Levy, and Shruti Bhosale. Causes and cures for interference in multilingual translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. arXiv:2212.07530.

[22] Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. arXiv:1602.07714.

[23] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2001.06782.