All reviewers & AC We thank the reviewers for their thoughtful feedback. The reviewers unanimously agree that the central idea of the work is novel, and reviewer 5 states that the work might have "important implications on hardware design and large-scale optimization". We want to assure the reviewers that we will heed their advice—for example:

1) We will add a substantial section to the appendix detailing our complete experimental setup.
2) We will add more discussion of the major assumptions and motivations underlying deep relative trust.

Multiplicative updates depart from the best practices of deep learning that were refined over many research iterations. Though our early results are promising, we believe that more research effort is needed to improve our new techniques. Sharing this work with the NeurIPS community would help to start this conversation.

***

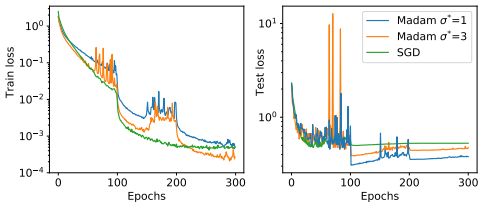
When is deep relative trust satisfied? (R1, R6) For a rigorous derivation of the deep relative trust distance function, please see [1, Theorem 1]. The proof holds for multilayer perceptrons with full rank weight matrices and leaky relu nonlinearity. While deep relative trust is unlikely to hold *exactly* for arbitrary neural networks or compositional functions, we contend that it is a better model of neural networks than its alternatives (e.g. Lipschitz gradients).

[1] Bernstein et al. 2020. *On the distance between two neural networks and the stability of learning.* arXiv:2002.03432.

Comparison to exponential gradient (R1) The exponential gradient algorithm is classically derived via mirror descent with a KL-divergence distance function. This leads to *unbounded* relative updates of the form $\exp(\eta g)$. Our analysis, on the other hand, leads to *bounded* relative updates of the form $\exp(\eta \operatorname{sign} g)$. Bounded and unbounded relative updates have substantially different properties. More broadly, our analysis is tailored to neural networks via the deep relative trust assumption—in contrast, KL-divergence seems to have no obvious connection to neural networks.

Strength of the theory (R1) Our empirical results question the suitability of the modelling assumptions used to prove many existing theory results in deep learning. We therefore believe our work to have strong theoretical significance despite only proving one simple result in this paper. That said, we should mention that Theorem 1 *does* entail convergence to stationary points, provided no weight is initialised to zero. To see this, note that the weights stay non-zero under a multiplicative update, so $g(W) \neq 0 \implies \cos\gamma > 0 \implies$ descent is possible by Theorem 1.

Convergence plots (R5, R6) Though we did not include this in the paper, Madam converges at a very similar speed to Adam and SGD. See the inset figure for a comparison on CIFAR-10 classification. Note that reduced σ^* both stabilises *and* regularises the test loss.



Discussion on weight clipping via σ^* (R5) At the end of training, a learning algorithm can overfit the training set by simply scaling up the learnt weights (amplifying confidence on its current predictions). Since Madam can increase the weights *exponentially*, this effect was particularly unstable and led us to introduce weight clipping. Afterward we discovered that tuning σ^* had a regularizing effect. The best σ^* was fairly consistently between 1 and 3 times the Xavier init scale. We did not experiment thoroughly with alternative regularizers in Madam. The baselines we compare against are heavily tuned—for example, SGD on Imagenet uses a weight decay value of 10^{-4}.

Reviewer 1 miscellanea On a *killer app*: the tuning gains and easy low bit width training may be highly significant.

Reviewer 5 miscellanea We would like to clarify one potential misunderstanding. Madam uses elementwise exponentiation (not matrix exponentiation)—thus the iteration cost of Madam is very similar to Adam. When ported to log number system hardware, Madam should be significantly faster than Adam. Also, since Madam is purely elementwise, applying Madam to DenseNet is simple and does not require defining a notion of a layer. Now let us clarify the phrase "for every subset W_* of weights W"—let's say a network has N weights $W = [W_1, W_2, ..., W_N]$. Now take any subset of the indices, e.g. $S = \{2, 4, 5\}$, and construct a new weight vector W_* based on S—in this case $W_* := [W_2, W_4, W_5]$. Then after a multiplicative update of size η, the relative change in the vector 2-norm of *any* such W_* is η.

Reviewer 6 miscellanea Figure 1 shows test set results. A Hessian with 10^{18} entries comes from assuming a neural network with 10^9 weights (e.g. GPT-2). By O(0.01) we mean the same order of magnitude as 0.01—specifically, one coauthor used $\eta = 0.016$ in some low-precision experiments so that they could decay to the base precision by dividing by 2. Finally, here are the complete FP32 results. The metrics are classification error ($100 -$ accuracy) for the classifers, FID score for the cGAN and perplexity for the transformer.

Dataset	Adam η	Adam train	Adam test	SGD η	SGD train	SGD test	Madam η	Madam train	Madam test
CIFAR-10	0.001	0.01 ± 0.01	6.6 ± 0.2	0.1	0.01 ± 0.01	8.2 ± 1.3	0.01	0.01 ± 0.01	7.8 ± 0.2
CIFAR-100	0.001	0.02 ± 0.01	29.8 ± 0.4	0.1	0.03 ± 0.01	29.1 ± 0.2	0.01	2.35 ± 0.08	30.2 ± 0.1
ImageNet	0.01	19.8 ± 0.2	26.7 ± 0.3	0.1	17.7 ± 0.1	24.1 ± 0.1	0.01	25.4 ± 0.1	28.9 ± 0.1
cGAN	0.0001	23.1 ± 0.8	23.9 ± 0.9	0.01	34 ± 1	34 ± 1	0.01	19 ± 2	19 ± 2
Wikitext-2	0.0001	109.8 ± 0.5	173.4 ± 0.9	1.0	149.9 ± 0.2	169.6 ± 0.6	0.01	126.9 ± 0.2	173.3 ± 0.6