Adam through a Second-Order Lens

Ross M. Clarke RMC78@CAM.AC.UK
Baiyu Su BS683@CAM.AC.UK
José Miguel Hernández-Lobato JMH233@CAM.AC.UK
Department of Engineering, University of Cambridge

Abstract

Research into optimisation for deep learning is characterised by a tension between the computational efficiency of first-order, gradient-based methods (such as SGD and Adam) and the theoretical efficiency of second-order, curvature-based methods (such as quasi-Newton methods and K-FAC). We seek to combine the benefits of both approaches into a single computationally-efficient algorithm. Noting that second-order methods often depend on stabilising heuristics (such as Levenberg-Marquardt damping), we propose *AdamQLR*: an optimiser combining damping and learning rate selection techniques from K-FAC [39] with the update directions proposed by Adam, inspired by considering Adam through a second-order lens. We evaluate AdamQLR on a range of regression and classification tasks at various scales, achieving competitive generalisation performance vs runtime.

1. Introduction

Most frequently seen in the ML literature are *first-order* optimisers such as SGD, Adam [27] and their variants, with some exploratory studies on *second-order* algorithms such as quasi-Newton methods and K-FAC [39]. Broadly speaking, second-order algorithms aim to make more principled individual updates, which in turn are more computationally costly than those employed by first-order methods. Combined with a generally more complicated implementation, second-order methods have not yet proven preferable to first-order approaches for most practitioners [2].

In part, this is a stability issue — inaccurate curvature approximations may cause a second-order optimiser to take extremely large, destabilising update steps. Many approaches thus depend on additional heuristics, such as curvature damping. It is then natural to ask if these heuristics might play a more fundamental role than the curvature information, and similarly improve first-order techniques.

In this paper, we propose a damped automatic learning rate strategy: applying K-FAC's damping and learning rate selection techniques to Adam. The result is an efficient, scalable algorithm which competes strongly and robustly with commonly-used optimisers. After a review of related work in Section 2, we present the development of our algorithm in Section 3. We then justify our claims by experiment in Section 4 before Section 5 concludes. Our main contributions are as follows:

- To our knowledge, we present the first use of damping and second-order approximations to select learning rates in Adam
- We propose a variation of damping based on Adam's internal curvature estimates which, when applied to Adam's update proposals, outperforms classical damping from e.g. K-FAC
- We show our method competes with methods using tuned learning rates, while exhibiting robustness to hyperparameters

2. Related Work

SGD and Adam [27] are arguably the most popular optimisers in ML. Adam belongs to a class of *adaptive* first-order methods including Adagrad [14, 41] and RMSprop [54]. Balles and Hennig [4] demonstrate that Adam essentially scales gradient signs by their variance. Zhang et al. [59] show that Adam can be seen as a form of natural gradient mean field variational inference, whose mode-fitting behaviour is known to underestimate variance, corresponding to overestimating curvature in an optimisation task (see e.g. Figure 1.3 in Turner and Sahani [56]). Zhang et al. [60] use a noisy quadratic model to ablate over Adam's components. We seek to exploit the useful curvature information ignored by these methods' use of diagonal approximations or heuristics.

Second-order optimisers are seen more often in the optimisation literature than in practical ML. *Quasi-Newton* methods [45] are inspired by a Taylor series truncated at quadratic order, with a Hessian matrix characterising curvature. Martens [38] use the Hessian-vector product trick [48] to work implicitly with the exact Hessian. Other work modifies the Hessian to avoid degeneracies — a particular concern in saddle point-dense high-dimensional spaces [12, 47]. SHAMPOO [22] learns a factorised set of preconditioning matrices. However, in non-convex, non-quadratic ML problems, the raw Hessian may be badly misleading, leading to divergence.

For probabilistic models, the Fisher information matrix gives rise to the natural gradient family of methods [1]. The Fisher matrix characterises curvature in KL-divergence space between the predicted and ground truth probability distributions. Factorized Natural Gradient [21] approximates the Fisher using a Gaussian graphical model, while the Kronecker-Factored Approximate Curvature (K-FAC) method (Martens and Grosse [39] after an idea by Heskes [25]) imposes a block-diagonal approximation to the Fisher and represents each block by a Kronecker product. Extensions to K-FAC include EKFAC [18], which learns the approximate Fisher in an eigenvalue-aligned basis. K-BFGS [20] applies a similar factorisation strategy to the Hessian matrix, retaining theoretical guarantees from the classical BFGS optimiser [8, 15, 19, 52]. Although K-FAC can be applied in distributed settings, this is somewhat complex [46], and the use of Fisher curvature requires new expressions to be calculated for different loss functions. We also find K-FAC suffers a substantial overfitting risk.

Original efforts to dynamically adapt first-order learning rates imposed fixed learning rate schedules [11, 33, 36, 53, 58], but recent developments involve more dynamic adaptations by hypergradients [9, 13, 16, 35, 42], online Bayesian optimisation [26], or explicitly constructing an optimisation framework around the unique characteristics of deep neural networks [5]. Zhang et al. [60] and Kwatra et al. [30] adopt a similar quadratic model methodology to our work, but the latter compute a finite-difference approximation to this model rather than using the exact curvature information as we do, and introduces additional exploration/exploitation hyperparameters. Niu et al. [44] uses a parallel approach to ours to incorporate momentum into L-BFGS [34].

3. AdamQLR

Let $f(\boldsymbol{\theta})$ be the loss function (which we seek to minimise) of some network parameterised by $\boldsymbol{\theta}$. Adopting ML convention, we express *second-order* optimisers in the form $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \mathbf{C}^{-1}\mathbf{u}(\mathrm{g})$, where $\mathbf{C}$ is some curvature matrix (often a damped Hessian, Fisher or Gauss-Newton matrix).

Practical second-order methods for ML are necessarily approximate, as the curvature $\mathbf{C}$ is otherwise intractably large. Further engineering is then required to mitigate the impact of approximate curvature and non-convexity of f. For example, K-FAC [39] is motivated by a particular Kronecker factorisation of a block-diagonal $\mathbf{C}$, but then applies a raft of corrections and adaptive heuristics

Algorithm 1 Adam [27]

```
$\mathbf{m}_0, \mathbf{v}_0 \leftarrow \mathbf{0}$
for $t = 1, 2, \cdots$ until $\boldsymbol{\theta}$ converged do
    $\mathbf{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_{t-1})$
    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$
    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)(\mathbf{g}_t \odot \mathbf{g}_t)$
    $\widehat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\beta_1^t}$
    $\widehat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1-\beta_2^t}$
    $\mathbf{d}_t \leftarrow \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t}+\epsilon}$


    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \mathbf{d}_t$
end for
```

Algorithm 2 AdamQLR

```
$\mathbf{m}_0, \mathbf{v}_0 \leftarrow \mathbf{0}$
for $t = 1, 2, \cdots$ until $\boldsymbol{\theta}$ converged do
    $\mathbf{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_{t-1})$
    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$
    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)(\mathbf{g}_t \odot \mathbf{g}_t)$
    $\widehat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\beta_1^t}$
    $\widehat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1-\beta_2^t}$
    $\mathbf{d}_t \leftarrow \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t}+\epsilon}$
    Update learning rate $\alpha$ according to (3)
    Update damping $\lambda$ according to (2)
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \mathbf{d}_t$
end for
```

(including multiple periodically-updated damping/factorised Tikhonov regularisation terms, momentum, weight decay, exponential moving averages of curvature statistics and approximate exchange of expectations and Kronecker products). These additions are seemingly integral to K-FAC's success.

A natural question is then whether accepting first-order methods' inaccurate curvature models and applying second-order stability techniques would blend the computational efficiency and optimisation accuracy of each. Our proposal is thus to adapt Adam using techniques from K-FAC.

3.1. Adam Revisited

Algorithm 1 restates the Adam optimisation algorithm from Kingma and Ba [27] applied to f, with some minor notational changes. Our proposed algorithm derives from our anecdotal observation that Adam often makes good choices of update direction $\mathbf{d}_t = \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t}+\epsilon}$.

As we detail in Appendix C, the $\frac{1}{\sqrt{\widehat{\mathbf{v}}_t}+\epsilon}$ term in Algorithm 1 effectively performs a curvature transformation on the averaged gradient $\widehat{\mathbf{m}}_t$ using an approximate empirical Fisher matrix. This has widely-known limitations compared to using the true Fisher [29], and the square root is motivated only by a desire to be "conservative" [27]. Indeed, Adam is very similar to one construction of natural gradient mean-field variational inference [59], a technique known to prioritise locally fitting modes of the target probability distribution [56]. Underestimating global variance in this way corresponds to overestimating local curvature in optimisation, justifying Adam's conservative estimate.

3.2. Adopting Heuristics from K-FAC

K-FAC [39] features three stabilising heuristics: Levenberg-Marquardt damping, and learning rate and momentum selection according to a local second-order model. Since Adam already implements a momentum correction in $\widehat{\mathbf{m}}_t$, we consider only the first two techniques.

Levenberg-Marquardt damping [31, 37, 51] replaces the curvature matrix $\mathbf{C}$ with the damped matrix $\mathbf{C} + \lambda\mathbf{I}$, and can variously be interpreted as approximating a trust region, enforcing positive definiteness of $\mathbf{C}$, preventing large updates in low-curvature directions and interpolating between gradient descent and full second-order updates. In effect, it imposes a 'minimum curvature' on the objective to avoid issues from near-zero eigenvalues of $\mathbf{C}$.

Let $M(\boldsymbol{\theta})$ be an approximate model of f around $\boldsymbol{\theta}_{t-1}$, defined by a truncated Taylor series:

$$M(\boldsymbol{\theta}) = f(\boldsymbol{\theta}_{t-1}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1})^{\mathsf{T}}\mathbf{g}_t + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1})^{\mathsf{T}}(\mathbf{C} + \lambda\mathbf{I})(\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}). \tag{1}$$

The damping parameter λ is adapted by comparing the change in objective value predicted by the model $(M(\boldsymbol{\theta}_t) - M(\boldsymbol{\theta}_{t-1}))$ to the actual observed change $(f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_{t-1}))$. This adjustment quantifies the model's reliability by a reduction ratio ρ, incorporating stepping factors[1] $\omega_{\text{dec}}, \omega_{\text{inc}}$:

$$\rho = \frac{f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_{t-1})}{M(\boldsymbol{\theta}_t) - M(\boldsymbol{\theta}_{t-1})}; \qquad \lambda \leftarrow \begin{cases} \omega_{\text{dec}}\lambda & \text{if } \rho > \frac{3}{4} \\ \lambda & \text{if } \frac{1}{4} \leq \rho \leq \frac{3}{4} \\ \omega_{\text{inc}}\lambda & \text{if } \rho < \frac{1}{4} \end{cases}. \tag{2}$$

Once an update direction $\mathbf{d}_t$ has been chosen, a learning rate α is selected according to the second-order model M. Specifically, we minimise $M(\boldsymbol{\theta}_{t-1} - \alpha\mathbf{d}_t)$ with respect to α, which yields

$$\alpha = \frac{\mathbf{g}_t^{\mathsf{T}}\mathbf{d}_t}{\mathbf{d}_t^{\mathsf{T}}(\mathbf{C} + \lambda\mathbf{I})\mathbf{d}_t}. \tag{3}$$

A minor rearrangement shows the large matrix $\mathbf{C}$ only appears in products with vectors. The Jacobian-vector product trick [48], efficient Fisher decompositions [39] and similar techniques compute these matrix-vector products using only one additional forward pass per product with $\mathbf{C}$. In practice, the information value of these calculations outweighs the additional computational cost.

3.3. Extending Adam

Incorporating K-FAC's damping and learning rate selection strategies into Adam yields Algorithm 2. We name this family of algorithms *AdamQLR*, where *QLR* indicates an optimiser-agnostic quadratic-model learning rate selection logic, which may be applied more broadly (e.g. to SGD).

For $\mathbf{C}$, we use the (true) Fisher matrix throughout, inspired by its connection with Adam's $\widehat{\mathbf{v}}_t$ buffer (see Appendix C.3), its use at the heart of K-FAC and its positive semi-definite guarantee. Particularly large choices of α affected AdamQLR's training stability, and the problem worsened with larger models, as these increase the prevalence of low-curvature regions of the space which induce very large update sizes. We found this issue was most effectively mitigated by clipping the learning rate to some maximum α_{max}, and that larger batch sizes tended to improve our curvature estimates, leading to better performance despite the higher cost of each forward pass.

With these choices made, note that the only remaining hyperparameters are β_1, β_2 and ϵ (from Adam) and an initial damping value λ_0. As it is common for Adam's hyperparameters to be fixed at the default values suggested by Kingma and Ba [27], and we show λ and α_{max} to be sufficiently insensitive that a default value can be recommended (Appendix B.2), we claim that AdamQLR is suitable for use without explicit hyperparameter tuning, and justify that claim in Section 4.

Compared to Adam, the additional forward passes required to compute $M(\boldsymbol{\theta}_t)$ and $(\mathbf{C} + \lambda\mathbf{I})\mathbf{d}_t$ turn out not to impede performance in our experimental results, though we note a careful implementation would amortise the former cost. Our only significant additional memory cost is storing the vector $(\mathbf{C} + \lambda\mathbf{I})\mathbf{d}_t$, making our approximate memory footprint four times that of SGD (as opposed to Adam's footprint of three times SGD).

1. In the most general form we allow separate decrease and increase factors, but in practice we will often choose $\omega_{\text{dec}} = \frac{1}{\omega_{\text{inc}}}$ for simplicity. We also require $0 < \omega_{\text{dec}} < 1 < \omega_{\text{inc}}$.

4. Experiments

We examine the training and test performance of AdamQLR in a variety of settings:

UCI Energy [55] on an MLP with one hidden layer of 50 units; 4 000 epochs
UCI Protein [49] on an MLP with one hidden layer of 100 units; 200 epochs
Fashion-MNIST [57] on an MLP with one hidden layer of 50 units; 10 epochs
SVHN [43] on a ResNet-18 [23]; 10 epochs
CIFAR-10 [28] on a ResNet-18 [23]; 72 epochs

We also demonstrate preliminary scalability to ImageNet in Appendix B.1.9. On UCI datasets we generate random splits using the same sizes as Gal and Ghahramani [17] and use MSE loss; otherwise, we separate the standard test set, choose 1/6 (Fashion-MNIST and SVHN) or 1/10 (CIFAR-10) of the remaining data to form a validation set, and use cross-entropy loss. Our complete code is available at `https://github.com/rmclarke/AdamThroughASecondOrderLens`. We compare:

SGD Minimal Classical mini-batched stochastic gradient descent, with tuned learning rate
SGD Full *SGD Minimal* with additional tuned momentum and weight decay
Adam [27] with tuned learning rate and fixed defaults for other hyperparameters
K-FAC [6, 39] with tuned initial damping
AdamQLR (Tuned) Algorithm 2, using Fisher curvature for $\mathbf{C}$. We tune initial damping, damping adjustment factors $\omega_{\text{dec}}, \omega_{\text{inc}}$ and learning rate clipping
AdamQLR (Untuned) *AdamQLR* with fixed batch size 3 200, initial damping 0.001, $\omega_{\text{dec}} = \frac{1}{\omega_{\text{inc}}} = 0.5$ and learning rate clipping 0.1 (justified by Appendix B.2)

Except for *AdamQLR (Untuned)*, we also tune the batch size. All hyperparameter tuning uses ASHA [32] over 200 random initialisations, where we target a fixed number of training epochs, subject to a maximum runtime of 15 minutes (only reached for CIFAR-10; see Appendix B.1.10 for experiments using runtime as the primary constraint). For our loss evolution figures, we perform 50 runs using each of the best hyperparameters found (measured by final validation loss), then plot the mean and standard deviation of the median trends of each of 50 bootstrap samples of the results. Following Botev and Martens [6], where damping is present we clip it to ensure $\lambda \geq 10^{-8}$.

Our most notable result is that *AdamQLR (Untuned)*, despite receiving no tuning budget and using the same hyperparameters for each dataset, competes very strongly with the tuned implementation of *Adam*. In general, the QLR-computed learning rates accelerate initial convergence, while damping provides some defence against overfitting (at the cost of increased final training loss). We also replicate Martens et al. [40]'s obervation of *K-FAC*'s susceptibility to overfitting, decreasing its appeal for the training task. Throughout, *Adam* and *SGD (Full)* remain strong contenders — perhaps unsurprising given their ubiquity in ML. We observed substantially different training dynamics between ResNet and Transformer models, an investigation of which we leave to future work.

5. Conclusion

We propose AdamQLR, an extension to Adam incorporating K-FAC's learning rate selection and adaptive damping strategies. Our algorithm reduces the overfitting seen in other techniques, and our suggested default hyperparameters compete with tuned optimisers. That AdamQLR competes so strongly with K-FAC, despite representing an algorithmic 'midpoint' between Adam and K-FAC, suggests an interesting direction for future work.

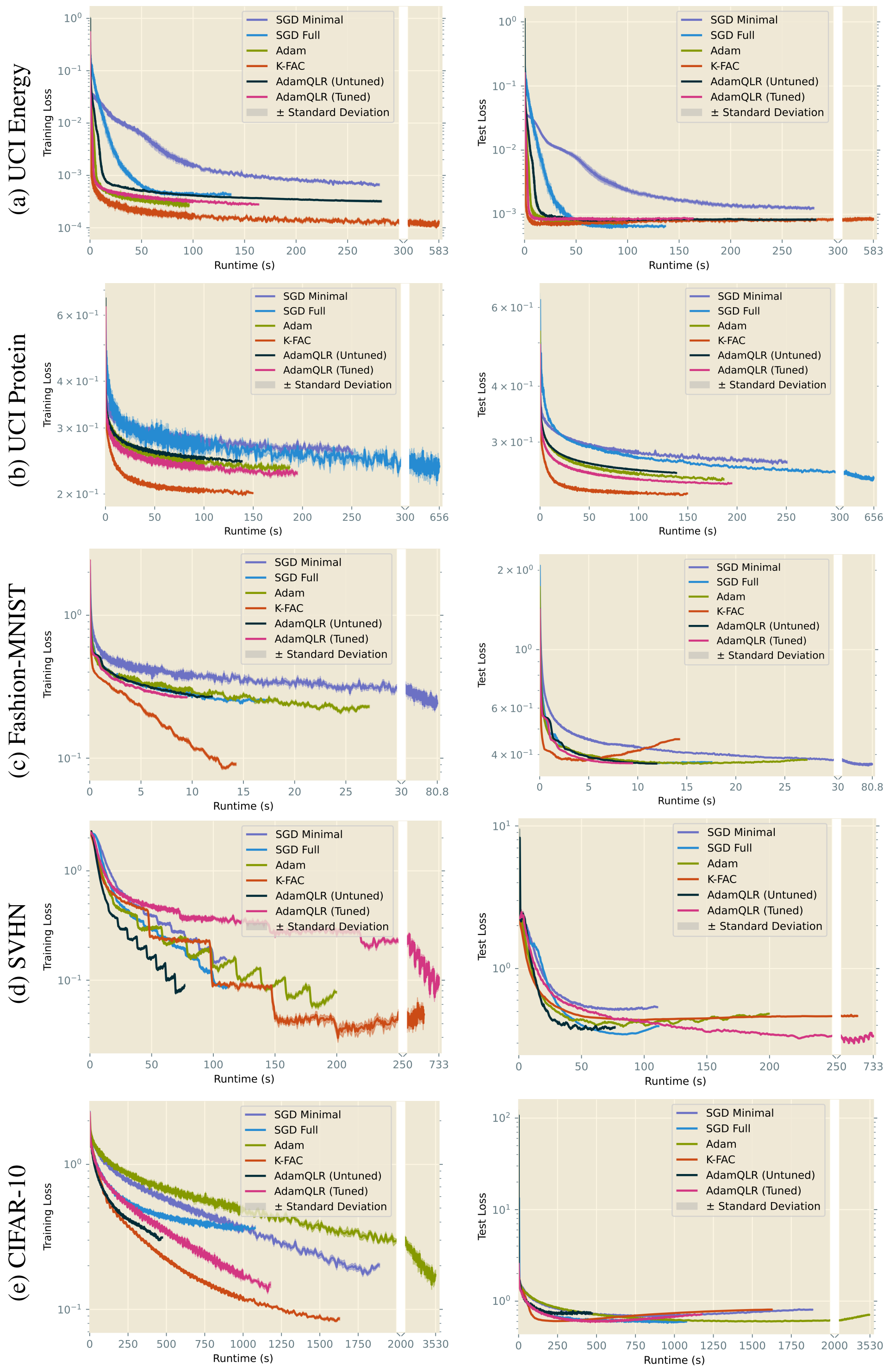

Figure 1: Median training (left) and test (right) loss trajectories, bootstrap-sampled over 50 repetitions per algorithm. Hyperparameters chosen by ASHA over 200 initialisations. Note changes of scale on the time axes. See also results on accuracy metrics and learning rate evolutions in Figures 3 and 4.

Acknowledgements

We acknowledge computation provided by the CSD3 operated by the University of Cambridge Research Computing Service (`www.csd3.cam.ac.uk`), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Countil (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council (`www.dirac.ac.uk`).

Ross Clarke acknowledges funding from the Engineering and Physical Sciences Research Council (project reference 2107369, grant EP/S515334/1).

References

[1] Shun-ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2): 251–276, February 1998.

[2] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable Second Order Optimization for Deep Learning, March 2021. arXiv:2002.09018 [cs, math, stat].

[3] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[4] Lukas Balles and Philipp Hennig. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In *Proceedings of the 35th International Conference on Machine Learning*, pages 404–413. PMLR, July 2018. ISSN: 2640-3498.

[5] Jeremy Bernstein, Chris Mingard, Kevin Huang, Navid Azizan, and Yisong Yue. Automatic Gradient Descent: Deep Learning without Hyperparameters, April 2023. arXiv:2304.05187 [cs, math, stat].

[6] Aleksandar Botev and James Martens. KFAC-JAX, 2022.

[7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[8] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, March 1970.

[9] Ross M. Clarke, Elre Talea Oldewage, and José Miguel Hernández-Lobato. Scalable One-Pass Optimisation of High-Dimensional Weight-Update Hyperparameters by Implicit Differentiation. In *The Tenth International Conference on Learning Representations, {ICLR} 2022, Virtual Event, April 25-29, 2022*, 2022.

[10] George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking Neural Network Training Algorithms, June 2023. arXiv:2306.07179 [cs, stat].

[11] Christian Darken and John Moody. Note on Learning Rate Schedules for Stochastic Optimization. In *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann, 1990.

[12] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[13] Michele Donini, Luca Franceschi, Paolo Frasconi, Massimiliano Pontil, and Orchid Majumder. MARTHE: Scheduling the Learning Rate Via Online Hypergradients. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, {IJCAI-20}*, volume 3, pages 2119–2125, July 2020. ISSN: 1045-0823.

[14] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[15] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3): 317–322, January 1970.

[16] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and Reverse Gradient-Based Hyperparameter Optimization. In *International Conference on Machine Learning*, pages 1165–1173, July 2017.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, June 2016. ISSN: 1938-7228 Section: Machine Learning.

[18] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[19] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[20] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical Quasi-Newton Methods for Training Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 2386–2396. Curran Associates, Inc., 2020.

[21] Roger Grosse and Ruslan Salakhudinov. Scaling up Natural Gradient by Sparsely Factorizing the Inverse Fisher Matrix. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2304–2313. PMLR, June 2015. ISSN: 1938-7228.

[22] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned Stochastic Tensor Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1842–1850. PMLR, July 2018. ISSN: 2640-3498.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919.

[24] Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020.

[25] Tom Heskes. On "Natural" Learning and Pruning in Multilayered Perceptrons. *Neural Computation*, 12(4):881–901, April 2000. Conference Name: Neural Computation.

[26] Yuchen Jin, Tianyi Zhou, Liangyu Zhao, Yibo Zhu, Chuanxiong Guo, Marco Canini, and Arvind Krishnamurthy. AutoLRS: Automatic Learning-Rate Schedule by Bayesian Optimization on the Fly. In *International Conference on Learning Representations*, January 2023.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[28] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, University of Toronto, April 2009.

[29] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[30] Nipun Kwatra, V. Thejas, Nikhil Iyer, Ramachandran Ramjee, and Muthian Sivathanu. AutoLR: A Method for Automatic Tuning of Learning Rate. *Submission to ICLR 2020*, May 2023.

[31] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[32] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A System for Massively Parallel Hyperparameter Tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, March 2020.

[33] Zhiyuan Li and Sanjeev Arora. An Exponential Learning Rate Schedule for Deep Learning. In *8th International Conference on Learning Representations, {ICLR} 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, December 2019.

[34] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[35] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, June 2020. ISSN: 2640-3498.

[36] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[37] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963. Publisher: Society for Industrial and Applied Mathematics.

[38] James Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 735–742, Madison, WI, USA, June 2010. Omnipress.

[39] James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *International Conference on Machine Learning*, pages 2408–2417, June 2015.

[40] James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored Curvature Approximations for Recurrent Neural Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[41] H. Brendan McMahan and Matthew Streeter. Adaptive Bound Optimization for Online Convex Optimization, July 2010. arXiv:1002.4908 [cs].

[42] Paul Micaelli and Amos Storkey. Non-greedy Gradient-based Hyperparameter Optimization Over Long Horizons. *arXiv:2007.07869 [cs, stat]*, July 2020. arXiv: 2007.07869.

[43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[44] Yue Niu, Zalan Fabian, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr. mL-BFGS: A Momentum-based L-BFGS for Distributed Large-Scale Neural Network Optimization. *Transactions on Machine Learning Research*, July 2023.

[45] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[46] Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12359–12367, 2019.

[47] Razvan Pascanu and Yoshua Bengio. Revisiting Natural Gradient for Deep Networks, February 2014. arXiv:1301.3584 [cs].

[48] Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1): 147–160, January 1994.

[49] Prashant Singh Rana. UCI Machine Learning Repository: Physicochemical Properties of Protein Tertiary Structure Data Set, March 2013.

[50] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, January 1960.

[51] Sam Roweis. Levenberg-Marquardt Optimization. Technical report, New York University, 1996.

[52] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[53] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't Decay the Learning Rate, Increase the Batch Size. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[54] Tijmen Tieleman and Geoffrey Hinton. Neural Networks for Machine Learning: Lecture 6. *Coursera*, 2012.

[55] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49: 560–567, June 2012.

[56] R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian time series models*, pages 109–130. Cambridge University Press, 2011. Section: 5.

[57] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, September 2017. arXiv: 1708.07747.

[58] Zhen Xu, Andrew M. Dai, Jonas Kemp, and Luke Metz. Learning an Adaptive Learning Rate Schedule. *arXiv:1909.09712 [cs, stat]*, September 2019. arXiv: 1909.09712.

[59] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy Natural Gradient as Variational Inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5852–5861. PMLR, July 2018. ISSN: 2640-3498.

[60] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which Algorithmic Choices Matter at Which Batch Sizes? Insights From a Noisy Quadratic Model. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Appendix A. Notes

A.1. Ethics Statement

Our work proposes a general optimisation algorithm for neural networks, so is unlikely to influence a particular societal problem. However, increasing the effectiveness of optimisation methods makes it easier for both benevolent and malevolent actors to develop systems aligned with their goals, so this class of risk is unavoidable. Of additional concern is that the typical setting of seeking to optimise a test metric by minimising a training metric fundamentally misaligns our algorithms with our objectives, and that misalignment may cause unexpected downstream consequences if poorly understood by the model developer. Finally, it would be naïve to presume any one optimisation algorithm is a panacea for all settings, and any errant belief in this vein may cause promising research directions to be incorrectly dismissed if a supposedly ‘universal’ optimiser happens to perform poorly on it.

A.2. Reproducibility Statement

We describe our algorithm fully in Section 3, provide full source code to the reviewers and will publish this code to the community after deanonymisation. The descriptions in this paper describe all the modifications we make to Adam and provide an complete intuitive summary of our contribution, while the source code allows any fine detail of our implementation or experiments to be inspected.

A.3. Limitations

While we have evaluated our algorithm on a range of datasets and models, we have necessarily left many important settings untested. Thus, even though we expect our method to generalise well to other settings, we should recognise that it has likely not yet been tested in those settings. In particular, the learning rate selection strategy used by K-FAC and our work assumes the optimisation space is approximately convex and quadratic, which will not generally be true of machine learning problems — this motivates our use of damping to defend against particularly ill-posed updates. With sufficient damping, we effectively define a ‘trust region’ beyond which the surface can be non-quadratic without harming our method. Further, since Adam is known not to perform well in certain (poorly-understood) circumstances [4], we might expect AdamQLR to have difficulty with the same class of problems.

A.4. Hyperparameter Search Space

We use similar hyperparameter search spaces (with unused hyperparameters removed) for each dataset and algorithm combination. These are detailed in Table 1.

A.5. Chosen Hyperparameters

The best hyperparameters selected by ASHA for each setting considered in this work are indicated in Table 2.

A.6. Compute Used

Our experiments were performed on one of the two sets of hardware shown in Table 3. All runtime comparisons were performed on like-for-like hardware. We make use of GPU acceleration throughout,

Table 1: Hyperparameter search spaces for Section 4

Hyperparameter	Search Range
Batch Size	Uniform in $\{50, 100, 200, 400, 800, 1\,600, 3\,200\}$
Learning Rate α	*SGD*: Logarithmic in $[10^{-6}, 10^{-1}]$ *Adam*: Logarithmic in $[10^{-6}, 1]$
Learning Rate Clipping α_{max}	Logarithmic in $[10^{-4}, 10]$
Momentum	Logarithmic in $[10^{-4}, 0.3]$, subtracted from 1
Weight Decay	Logarithmic in $[10^{-10}, 1]$
Initial Damping λ_0	Logarithmic in $[10^{-8}, 1]$
Damping Decrease Factor ω_{dec}	Logarithmic in $[0.5, 1.0]$
Damping Increase Factor ω_{inc}	Logarithmic in $[1.0, 4.0]$

Table 2: Optimal hyperparameters used to produce the results of Section B.1.3

Dataset	Algorithm	Batch Size	Learning Rate	Learning Rate Clipping	Momentum	Weight Decay	Initial Damping	Damping Decrease Factor	Damping Increase Factor
Rosenbrock	GD Minimal	—	—	—	—	—	—	—	—
	GD Full	—	—	—	—	—	—	—	—
	Adam	—	9.8848×10^{-2}	—	—	—	—	—	—
	AdamQLR (Tuned, Hessian)	—	—	6.098	—	—	3.0270×10^{-6}	0.9	2.1
	AdamQLR (Untuned)	—	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
UCI Energy	SGD Minimal	100	9.8838×10^{-2}	—	—	—	—	—	—
	SGD Full	400	6.9156×10^{-2}	—	0.9962	1.2866×10^{-4}	—	—	—
	Adam	800	2.9913×10^{-2}	—	—	—	—	—	—
	K-FAC	50	—	—	—	—	1.0047×10^{-2}	—	—
	AdamQLR (Untuned)	3200	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
	AdamQLR (Tuned)	400	—	2.843	—	—	8.7094×10^{-2}	0.5	2.3
UCI Protein	SGD Minimal	400	7.0021×10^{-2}	—	—	—	—	—	—
	SGD Full	100	2.1694×10^{-4}	—	0.9970	1.5361×10^{-8}	—	—	—
	Adam	800	5.4189×10^{-3}	—	—	—	—	—	—
	K-FAC	3200	—	—	—	—	2.1064×10^{-1}	—	—
	AdamQLR (Untuned)	3200	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
	AdamQLR (Tuned)	800	—	0.141	—	—	1.5054×10^{-4}	0.5	1.9
Fashion-MNIST	SGD Minimal	100	8.0075×10^{-2}	—	—	—	—	—	—
	SGD Full	800	5.8068×10^{-2}	—	0.9289	1.6522×10^{-8}	—	—	—
	Adam	400	2.5634×10^{-3}	—	—	—	—	—	—
	K-FAC	3200	—	—	—	—	1.9224×10^{-1}	—	—
	AdamQLR (Tuned, Hessian)	3200	—	0.269	—	—	2.5420×10^{-5}	1.0	2.8
	AdamQLR (Untuned)	3200	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
	AdamQLR (Undamped)	3200	—	0.149	—	—	—	—	—
	AdamQLR (Tuned)	3200	—	0.219	—	—	4.9595×10^{-3}	0.6	1.3
CIFAR-10	SGD Minimal	200	3.4672×10^{-2}	—	—	—	—	—	—
	SGD Full	400	3.8337×10^{-2}	—	0.9203	8.7353×10^{-4}	—	—	—
	Adam	100	2.0380×10^{-4}	—	—	—	—	—	—
	K-FAC	1600	—	—	—	—	9.0326×10^{-1}	—	—
	AdamQLR (Tuned, Hessian)	200	—	0.001	—	—	2.1848×10^{-4}	0.5	2.1
	AdamQLR (Untuned)	3200	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
	AdamQLR (Undamped)	200	—	0.001	—	—	—	—	—
	AdamQLR (Tuned)	400	—	0.001	—	—	7.1607×10^{-6}	0.5	1.2
SVHN	SGD Minimal	1600	3.8629×10^{-2}	—	—	—	—	—	—
	SGD Full	1600	6.0953×10^{-3}	—	0.9862	8.6112×10^{-7}	—	—	—
	Adam	800	4.1027×10^{-4}	—	—	—	—	—	—
	K-FAC	800	—	—	—	—	6.4013×10^{-1}	—	—
	AdamQLR (Untuned)	3200	—	0.100	—	—	1.0000×10^{-3}	0.5	2.0
	AdamQLR (Tuned)	200	—	0.001	—	—	2.6287×10^{-8}	0.7	1.3

Table 3: System configurations used to run our experiments.

Type	CPU	GPU (NVIDIA)	Python	JAX	CUDA	cuDNN
Consumer Desktop	Intel Core i7-3930K	RTX 2080GTX	3.10.11	0.3.25	11.4	8.05
Local Cluster	Intel Core i9-10900X	RTX 2080GTX	3.10.11	0.3.25	11.8	8.05

Table 4: Licences under which we use datasets in this work

Dataset	Licence	Source	Input	Output	Total Size
UCI Energy	Creative Commons Attribution 4.0 International (CC BY 4.0)	Tsanas and Xifara [55]; Gal and Ghahramani [17]	8-Vector	Scalar	692
UCI Protein	None specified	Rana [49]; Gal and Ghahramani [17]	9-Vector	Scalar	45 730
Fashion-MNIST	MIT	Xiao et al. [57]	28×28 Image	Class (from 10)	60 000
CIFAR-10	None specified	Krizhevsky [28]	32×32 Image	Class (from 10)	60 000
SVHN	None specified	Netzer et al. [43]	32×32 Image	Class (from 10)	99 289

with the JAX [7], Haiku [24] and KFAC-JAX [6] libraries, along with various related components of the DeepMind JAX Ecosystem [3].

Producing experimental data for every plot in this paper required approximately 228.3 GPU-hours on the Local Cluster and 9.5 GPU-hours on the Consumer Desktop. This accounts for performing multiple trials in parallel on the same GPU where capacity exists and for hyperparameter search, but excludes development, debugging and unit testing time, which would substantially increase these figures.

A.7. Datasets

The datasets we use are all standard in the ML literature; we outline their usage conditions in Table 4.

Appendix B. Additional Experiments

B.1. Algorithm Comparisons

In this Section, we provide some additional viewpoints into our main results of Section 4.

B.1.1. UCI ENERGY

UCI Energy provides a low-dimensional regression task on a small dataset, which is amenable to hosting long experiments to explore convergence behaviour. We consider 4 000 epochs of training and plot bootstrap-sampled median training and test loss trends in Figure 1a.

Our principal benchmarks fall much as we would expect: *SGD Minimal* makes respectable, if sluggish, progress during optimisation, but is outclassed by the more rapid initial convergence of *SGD Full* and *Adam*. Both these latter methods achieve strong test performance on this small-scale problem, with *SGD Full* outperforming all other methods. Despite making rapid initial progress, *K-FAC* quickly begins overfitting, reaching a final test loss similar to the *AdamQLR* methods.

Generally, *AdamQLR (Tuned)* competes comparably with its vanilla baseline. The QLR computed learning rates accelerate initial progress, while the addition of damping provides some defence against overfitting, at the cost of a higher final training loss. Note also that *AdamQLR*'s substantially lower variation indicates a robustness beyond that seen in other methods — the *Untuned* variation performs very competitively considering its competition has undergone hyperparameter tuning.

B.1.2. UCI PROTEIN

UCI Protein is another low-dimensional regression task, but with far more data points, allowing for a computationally-efficient study of a larger dataset. We show 200 epochs of training in Figure 1b.

Here we see greater distinction between the generalisation performance of each algorithm. *SGD Full* achieves a slight improvement over *SGD Minimal*, but still lags behind the other methods. *K-FAC* is now clearly the best-performing algorithm, as might perhaps be expected since it computes the most granular curvature approximation when choosing an update direction. However, we still see meaningful benefit from the *AdamQLR (Tuned)* algorithm, which now comfortably outperforms *Adam*. We observe *AdamQLR (Tuned)*'s automatic learning rate selection is capable of outperforming methods which require a sensitive explicit choice of learning rate — the *Untuned* variant is clearly superior to tuned *SGD* on this task and is only slightly worse than a tuned *Adam*.

B.1.3. FASHION-MNIST

Fashion-MNIST provides a first foray into higher-dimensional data, but at a scale still approachable by MLP models. Using a 10-epoch training window, we plot bootstrapped loss evolutions in Figure 1c.

At this slightly larger experimental scale, the benefits of our proposed algorithm become more apparent. Despite achieving the best final training loss of any method, *K-FAC* significantly overfits even before reaching other algorithms' final training losses. While this is a recognised issue with K-FAC [40], and the fundamental idea of minimising a test loss by optimising a training loss frustrates the application of naïvely-powerful optimisers, the impact is to make *K-FAC* undesirable in this application. *SGD Full*, *Adam* and *AdamQLR* all perform very similarly, generalising better than *K-FAC* and overfitting to a far lesser degree. *AdamQLR* is the most performant algorithm by a very small margin. We emphasise that the number of training epochs was chosen arbitrarily based on existing work, so the flattening-out of *AdamQLR*'s test loss at later times indicates robustness, not preferential treatment. We note again the strong performance of *AdamQLR (Untuned)*.

B.1.4. SVHN

With SVHN, we progress to a full-colour image dataset and a substantially larger-scale ResNet-18 model, which we tune for 10 epochs and present in Figure 1d. The periodicity in these loss evolutions corresponds to individual epochs, and is simply an artifact of training.

On this more realistically-scaled problem, we achieve substantial gains over *Adam*. *SGD Minimal* fulfils its expected role as a mediocre baseline, but *SGD Full* performs admirably in this setting, matching the other algorithms' initial rate of convergence in both training and test losses, and achieving the lowest test loss of any method. However, it then overfits, while other methods reach similar test losses more stably. *K-FAC* again fails to generalise its impressively low training losses, instead becoming stuck at a test loss almost ten times larger than its final training loss.

We see particuarly strong performance from the Adam-based methods. While *Adam* itself overfits before matching its competitors' test performance, *AdamQLR* reaches impressively low test losses and remains more stable there. Even though *SGD Full* transiently achieves better performance, *AdamQLR* is a more promising candidate for general application, as it achieves similar losses with greater robustness and meaningfully reduced hyperparameter tuning effort. Finally, the *Untuned* variant performs impressively at both training- and test-time, reinforcing its efficiency and utility.

B.1.5. CIFAR-10

Finally, in a simulation of larger-scale learning, we train a ResNet-18 on CIFAR-10 over 72 epochs. Here we include conventional data augmentation of 4-pixel padding, random cropping and random left-right flipping, and display our results in Figure 1e.

Adam is now slower to converge in both training and test loss, suggesting this could be an ill-suited setting in which Adam can be expected to underperform [4]. Otherwise, increasingly intricate algorithms make progressively faster progress at training-time, even if the generalisation performances are all very similar. The latter effect may reflect inherent issues in the training-test learning paradigm as well as the performance of any particular optimiser.

B.1.6. ROSENBROCK FUNCTION

The Rosenbrock Function [50] provides a visualisable low-dimensional test bed for optimisation algorithms, containing substantial non-linear correlations between its inputs and anisotropic curvature. We consider 200 optimisation steps, using $\mathcal{N}(\mathbf{0}, \mathbf{I})$-sampled initial (x, y) values during hyperparameter tuning, and plot trajectories from the fixed starting point $(1, -1)$ as our test case in Figure 2. As there is no probabilistic model, we cannot apply K-FAC in this setting, so omit it. For the same reason, in this section only, we use Hessian curvature in *AdamQLR*, and use gradient descent (*GD*) in place of *SGD*. Since there is no separate validation set, we tune hyperparameters on the same objective function as is used for 'training'.

Here, *GD Minimal* makes good initial progress into the central 'valley', but its learning rate is too small to continue along the valley floor. *GD Full*'s hyperparameters cause it to bounce unstably around the optimisation space. Because SGD cannot adapt to different gradient magnitudes, it must select conservative step sizes to avoid diverging when initialised away from the optimum — an effect particularly pronounced in *GD Minimal*, where there is no momentum buffer. *Adam*'s adaptive buffers allow it to target the valley more directly, eventually making slow progress along the valley floor, but it takes time to learn the new dynamics in the latter regime, and we see it initially 'overshoot' the valley.

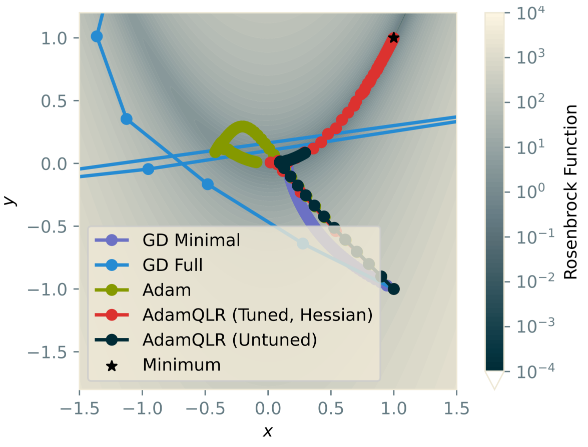

Figure 2: Optimisation trajectories over 200 steps from a fixed initial point on the Rosenbrock Function. Hyperparameter tuning used 200 standard-normal random initial points.

By contrast, *AdamQLR (Tuned)* reaches the valley floor efficiently, then shows an appealing understanding of the objective function geometry, tracking along the valley for substantial distances. SGD-based methods tend to take small, cautious steps along the floor, producing steady but slow convergence, while the Adam-based methods are able to take larger steps, making faster progress. *AdamQLR (Untuned)*'s learning rate clipping

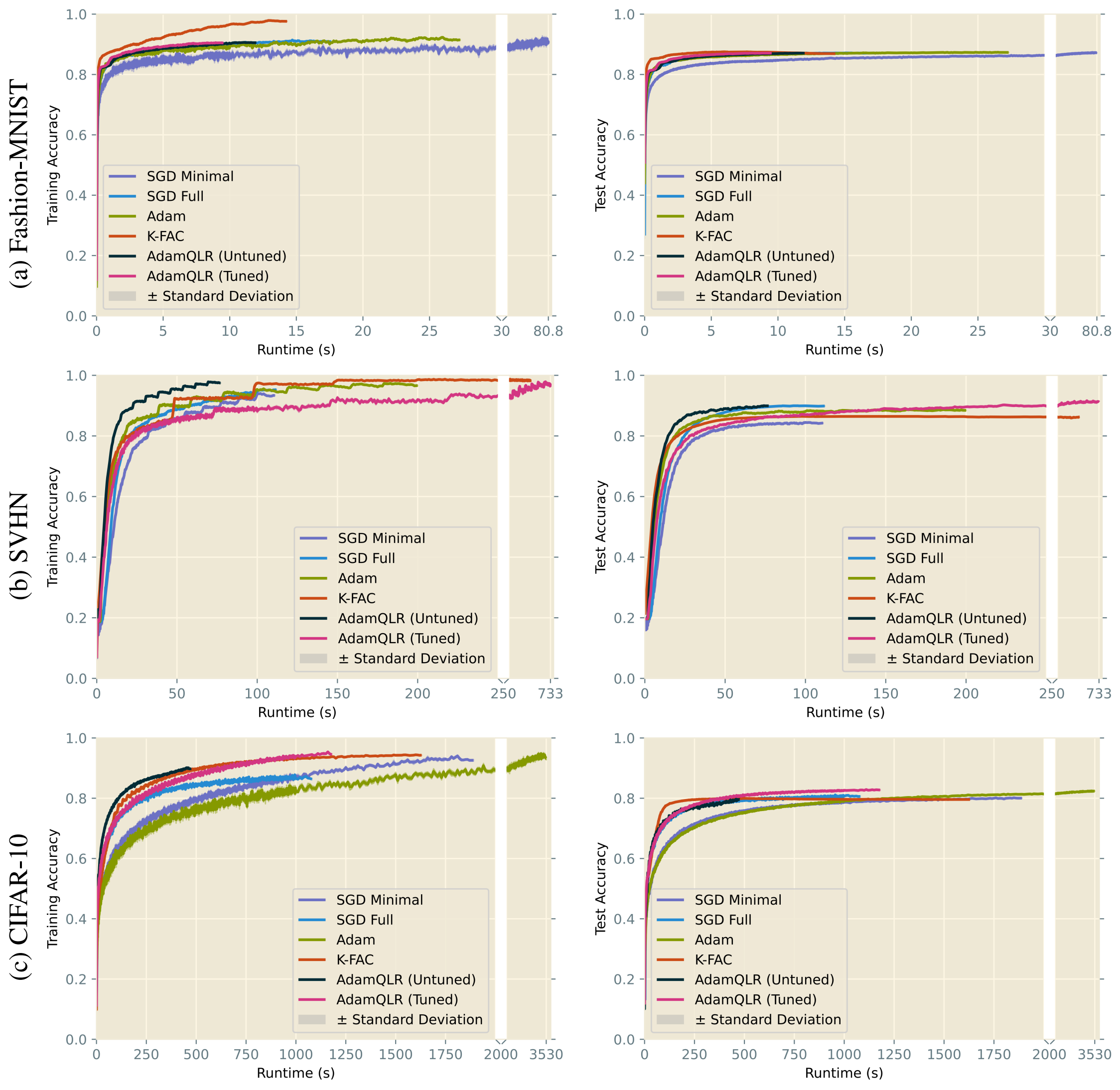

Figure 3: Median training (left) and test (right) accuracy trajectories, bootstrap-sampled over 50 repetitions per algorithm. Hyperparameters chosen by ASHA over 200 initialisations. Note changes of scale on the time axes.

threshold, being chosen for neural network applications, is too small here, but it also makes efficient progress into the valley and quickly adapts to the changing dynamics without overshooting. While this relatively simple function is not representative of the more complicated spaces of machine learning model parameters, our strategy displays a promising understanding of its correlated curvature.

B.1.7. FASHION-MNIST, SVHN AND CIFAR-10 ACCURACY

In Figure 1, we plotted experimental results in terms of the loss metric used during training. For Fashion-MNIST, SVHN and CIFAR-10, we also plot classification accuracy in metrics in Figure 3. These illustrate broadly the same patterns as we discussed in the main body of the paper.

B.1.8. LEARNING RATE EVOLUTION

In Figure 4, we plot the trajectories of average learning rates selected by AdamQLR and K-FAC against the fixed values used in SGD and Adam.

Learning rate schedules are widely known to be important in certain training problems, particularly at larger scales, so it is unsurprising that various algorithms' sense of the 'optimal' learning rate varies over time. For the most part, the chosen schedules give an approximately exponential decay in learning rate, interestingly excluding the warm-up behaviour commonly specified in manually-designed schedules. In UCI Energy and UCI Protein, we observe a resemblance between the fixed learning rates chosen by SGD and Adam and the typical values selected by AdamQLR and K-FAC, but this connection is much less clear in larger datasets, suggesting this scheduling behaviour becomes more important as problems grow in scale.

Curiously, although *AdamQLR (Tuned)* is able to choose a learning rate clipping value, it only seems to use this to completely disable its adaptive approach — as in SVHN and CIFAR — by setting the threshold lower than the learning rates our QLR strategy would otherwise select. This suggests automatic learning rate selection may not be as useful a tool as we might intuitively think, with a fixed value imposing helpful stability on training. Further, it is interesting to note that *AdamQLR (Untuned)* chooses *growing* learning rates on SVHN and CIFAR-10 which differ dramatically from those of other methods, yet achieves similar results in loss and accuracy space. In summation, these results suggest we might do well to explore other approaches to improving machine learning optimisers, beyond focussing on learning rates.

B.1.9. IMAGENET

In our explorations, while AdamQLR demonstrated competitive performance with smaller network architectures, its efficacy waned when scaling to larger models, specifically with a ResNet-50 [23] applied to the ImageNet classification task. We adopt the model and accuracy-time evaluation strategy from Dahl et al. [10] to shed light on these discrepancies, using untuned *Adam* and *AdamQLR* baselines alongside their 'SGD + Heavy ball momentum' setting (which we call *SGD-ImageNet*.

From our preliminary plots of training and test accuracy over time in Figure 5, at the initial phase, the performance hierarchy stands as *Adam* > *AdamQLR* > *SGD-ImageNet*. However, as training progresses, *AdamQLR* plateaus at a training accuracy of around 70% and a test accuracy of around 50%. Unlike *Adam* and *SGD-ImageNet*, which continue their ascent, our method stagnates, unable to further optimise.

A primary reason for this stagnation is the non-convergence of the learning rate in the later stages of training. The algorithm, designed to compute an optimal learning rate for every update step, fails to decrease this rate as training advances, resulting in the persistent large learning rates. This phenomenon suggests that the Levenberg-Marquardt rule, which we employed for damping updates, failed to adjust its damping values for the tail-end of the training process. Future iterations of our algorithm might benefit from a more adaptive damping update mechanism to ensure smoother learning rate annealing. Another intrinsic challenge with our approach is the computation of the optimal learning rate at each step, which requires one evaluation of Fisher vector product per step. For expansive models like ResNet-50, this operation introduces a non-trivial computational overhead. The marginal gains in performance, as observed in our experiments, do not sufficiently offset the increased computational costs for these larger models.

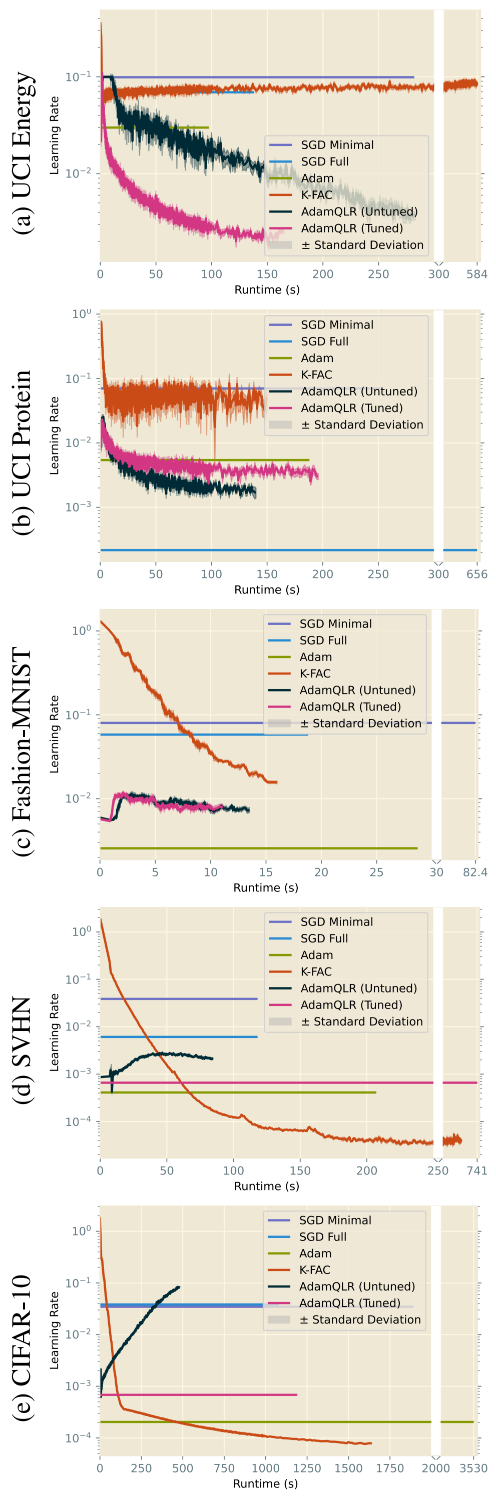

Figure 4: Median learning rate trajectories, bootstrap-sampled over 50 repetitions per algorithm. Hyperparameters chosen by ASHA over 200 initialisations. Note changes of scale on the time axes.

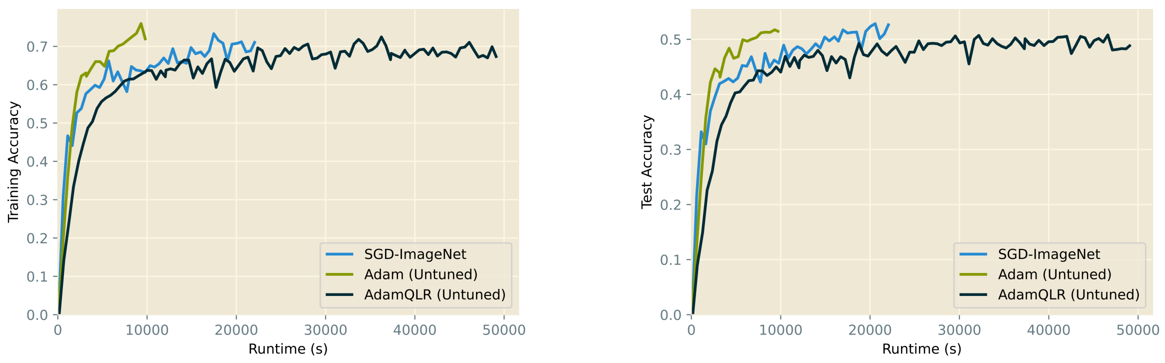

Figure 5: Training (left) and test (right) accuracy vs total training time with ResNet-50 on ImageNet

While our method, AdamQLR, introduces promising improvements for certain scenarios, its application to larger networks, like ResNet-50 on ImageNet, surfaces limitations that warrant further research and refinement. We believe that addressing these highlighted challenges can pave the way for a more universally robust optimisation strategy.

B.1.10. Fixed-Runtime Comparisons

Our main results in Section 4 impose a primary constraint of a fixed number of epochs, with a secondary constraint of a runtime limit. To develop additional context on AdamQLR's performance, we repeat these experiments without the primary number-of-epochs constraint, such that our hyperparameter tuning directly optimises for the best loss attained after the 15 minute runtime limit, and the algorithms are evaluated on the same metric. Figure 6 shows partial results where we optimised for final validation loss, while Figure 7 shows partial results where the hyperparameters were optimised to minimise final *training* loss. This latter setting allows us to compare the naïve power of each optimiser to optimise the given objective in isolation.

These results display an interesting tendency for *K-FAC* to wildly diverge in the later phases of training on Fashion-MNIST, an effect which *AdamQLR* is largely able to avoid. Broadly speaking, *AdamQLR* gives competitive generalisation performance on UCI Energy and UCI Protein in Figure 6, with a more pronounced overfitting behaviour on larger datasets. However, on CIFAR-10 *AdamQLR (Tuned)* achieves the strongest generalisation, and even on SVHN its performance is competitive. We additionally see an effective demonstration of *AdamQLR*'s optimisation power in Figure 7 — although training performance on Fashion-MNIST again lags behind *Adam* in this setting, larger datasets achieve particularly strong training loss evolutions.

B.2. Sensitivity Studies

To justify our configurations and further demonstrate the utility of our algorithm, we conduct a range of sensitivity experiments for *AdamQLR (Tuned)* trained on Fashion-MNIST under the same conditions as in Section B.1.3. All hyperparameters except for the one under investigation are fixed at the best values found for ASHA in those experiments. Again, our plots show the averages of median trends of bootstrap-sampled sets of 50 repetitions for each configuration considered.

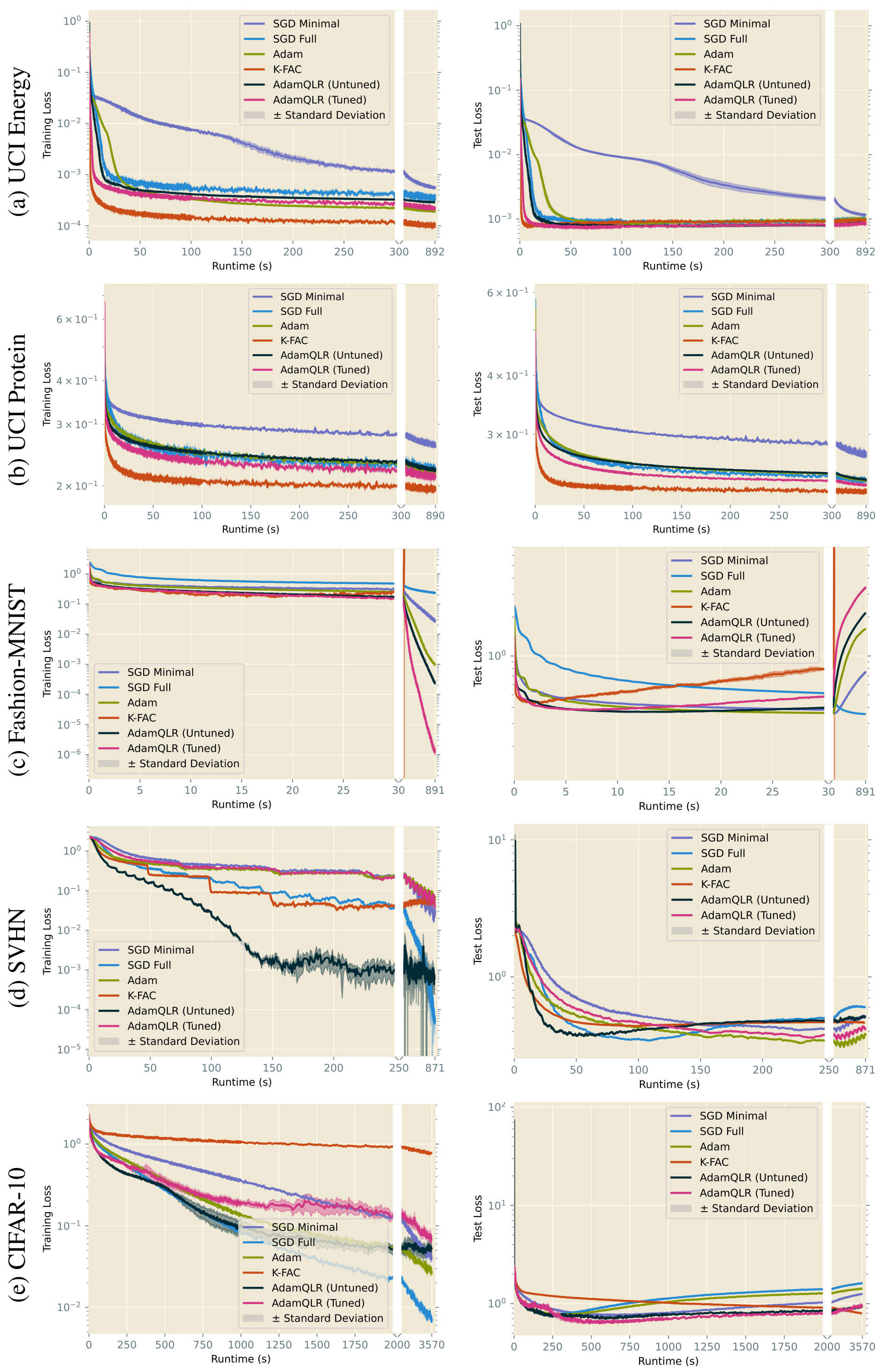

Figure 6: Median training (left) and test (right) loss trajectories, bootstrap-sampled over 50 repetitions per algorithm. Hyperparameters chosen by ASHA over 200 initialisations to minimise validation loss after a fixed runtime of 15 minutes. Note changes of scale on the time axes.

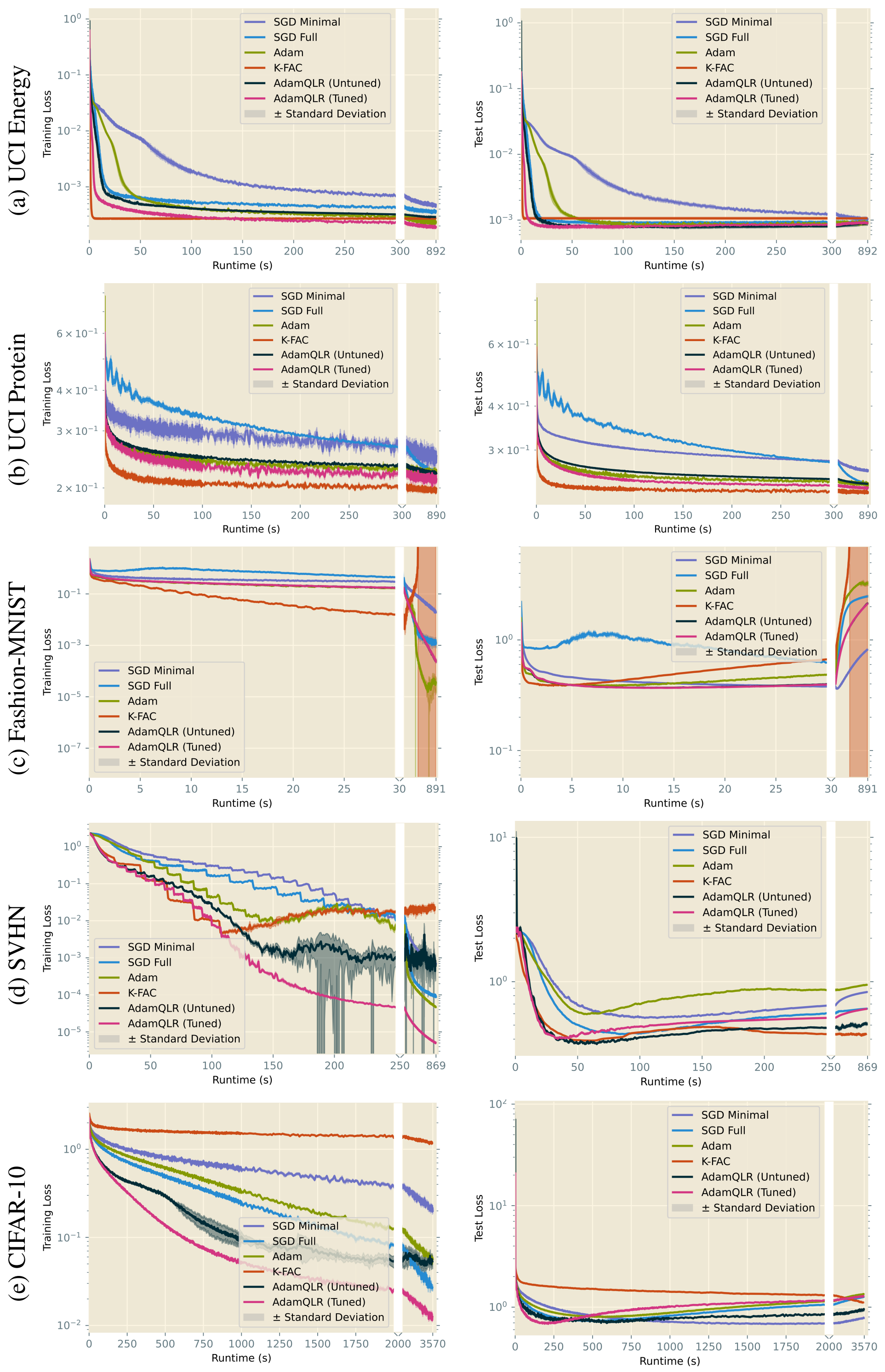

Figure 7: Median training (left) and test (right) loss trajectories, bootstrap-sampled over 50 repetitions per algorithm. Hyperparameters chosen by ASHA over 200 initialisations to minimise *training* loss after a fixed runtime of 15 minutes, characterising the naïve power of each algorithm. Note changes of scale on the time axes.

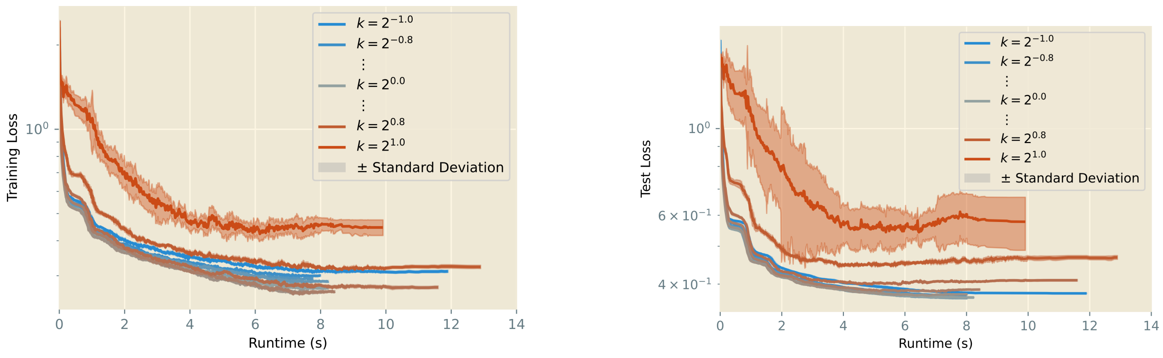

Figure 8: Ablation studies over learning rate, which is scaled by a variety of constant factors k for our Fashion-MNIST trial from Section B.1.3.

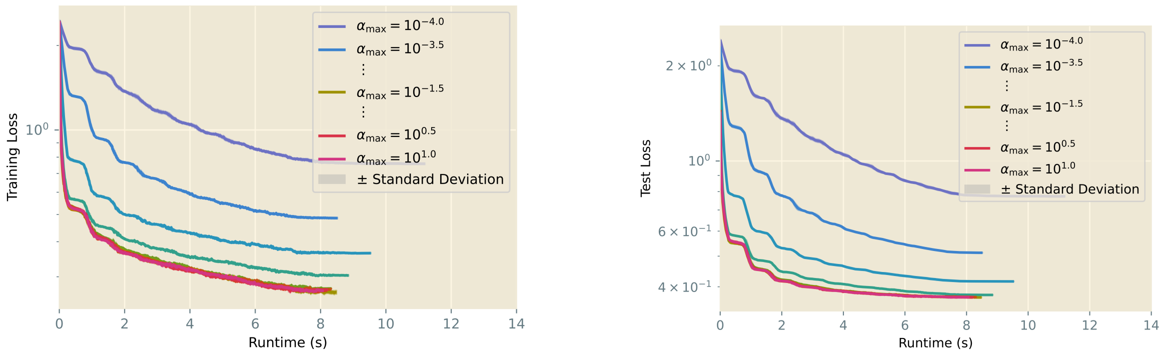

Figure 9: Ablation studies over learning rate clipping α_{max} for our Fashion-MNIST trial from Section B.1.3.

B.2.1. LEARNING RATE RESCALING

Firstly, we analyse the accuracy of our learning rate selection strategy by executing our algorithm as normal, but setting $\alpha \leftarrow k\alpha$ for each k in $\{2^{-1.0}, 2^{-0.8}, 2^{-0.6}, \cdots, 2^{1.0}\}$. This scaling is performed *after* any clipping has taken place. In effect, we investigate the potential for systemic bias in our learning rate selection by asking if our results would improve with a constant scaling factor on those learning rates.

Our results in Figure 8 show the $k = 2^{1.0}$ case exhibiting large variance due to unstable runs, while the best training losses are obtained for k slightly larger than unity. This makes sense given our use of damping: if stability can be achieved without damping for any given update, then the damping will serve only to downsize our proposed update step, so we should expect the best results to be obtained by slightly increasing it again. However, test loss appears generally less sensitive to k, with the lowest value obtained for $k = 1$: this would also be expected under damping, since we would hope the damping would increase generalisation performance. In aggregate, these results confirm our approach accurately selects the correct learning rate to use for any given optimisation step.

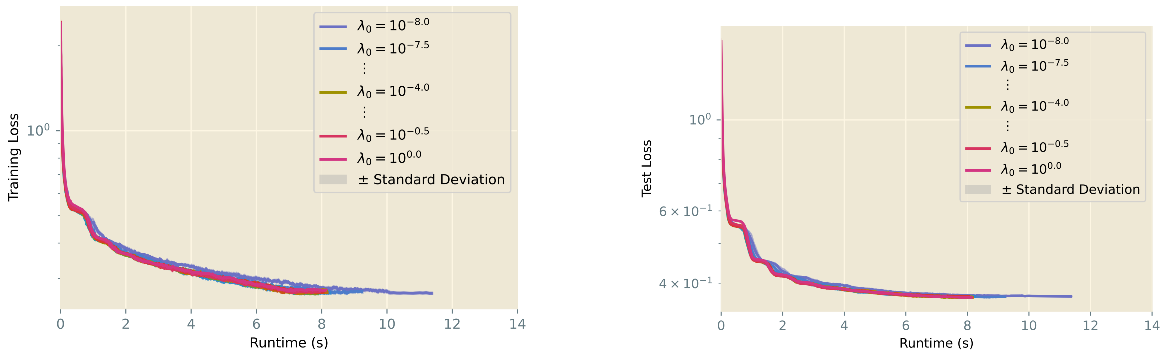

Figure 10: Ablation studies over initial damping value λ_0 for our Fashion-MNIST trial from Section B.1.3.

B.2.2. LEARNING RATE CLIPPING

We continue by considering the learning rate clipping threshold α_{max}, selecting values in $\{10^{-4.0}, 10^{-3.5}, 10^{-3.0}, \cdots, 10^{0.0}\}$ and plotting our results in Figure 9.

On Fashion-MNIST, we see a clear preference for a higher learning rate clipping threshold, corresponding to less aggressive clipping, with the effect shrinking after a point as the threshold becomes larger than any learning rate selected by AdamQLR. This makes sense — we introduce learning rate clipping to mitigate the effects of unstably large learning rates, and if these do not arise, we will only harm performance by clipping learning rates. Fashion-MNIST training proceeded successfully without clipping, so this hyperparameter is only of particular importance in larger problems where it is a more vital component of a stable training algorithm. However, it is reassuring to confirm that a sufficiently high learning rate clipping threshold will not drastically harm performance on otherwise stable problems.

B.2.3. INITIAL DAMPING

Next, we consider the initial value λ_0 assigned to our Levenberg-Marquardt damping term λ, testing values in $\{10^{-8.0}, 10^{-7.5}, 10^{-7.0}, \cdots, 10^{0.0}\}$. Here, we seek to quantify the trade-off between damping's stabilising effect and its tendency to worsen training loss. Figure 10 presents our results.

With the exception of the very smallest values, we see our performance is largely insensitive to λ_0. This matches our empirical observation that damping becomes most important for larger-scale problems than our Fashion-MNIST setting, and thus has minimal effect here. However, given its substantial importance in these more complex experiments, it is reassuring that the inclusion of damping does not dramatically worsen performance when its influence is not required.

B.2.4. BATCH SIZE

In Figure 11, we consider each batch size available to ASHA in Section B.1.3 ($\{50, 100, 200, 400, 800, 1\,600, 3\,200\}$) to investigate the effect of this hyperparameter on our algorithm.

Since the optimal batch size selected by ASHA for *AdamQLR* was generally large (3 200 in this case), it is perhaps unsurprising that we see divergence from smaller batches. This also matches our intuition: unlike classical first-order methods, *AdamQLR* uses each batch to (implicitly) construct a full curvature matrix for the optimisation surface, which magnifies the importance of having a

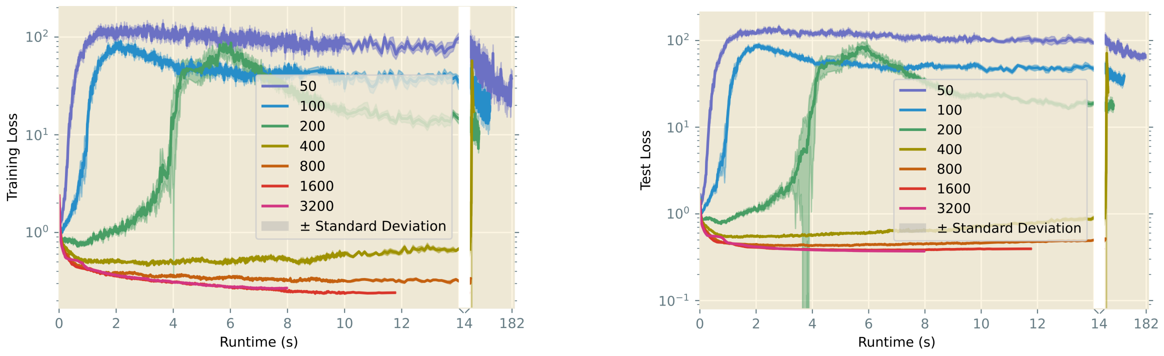

Figure 11: Ablation studies over batch size for our Fashion-MNIST trial from Section B.1.3.

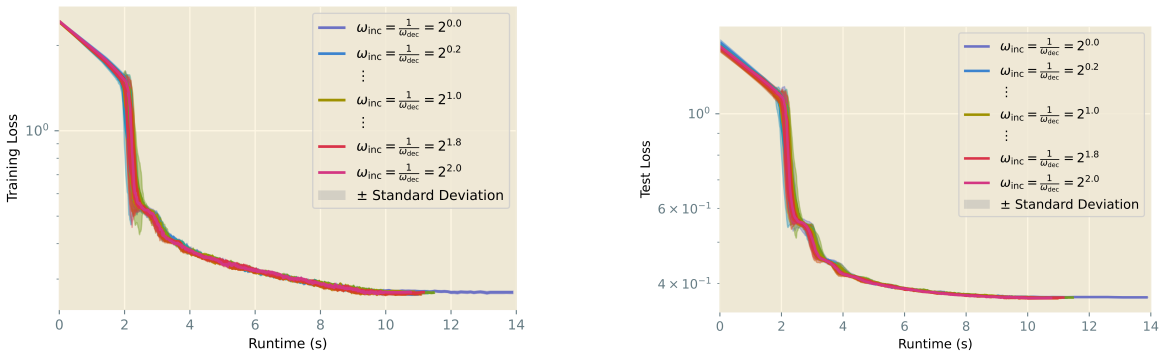

Figure 12: Ablation studies over damping stepping factor for our Fashion-MNIST trial from Section B.1.3.

low-bias sample of the training data. Empirically, we found the computational benefits of fewer batches outweighed the increased cost of computing each batch, so this preference for larger batch sizes aligns with our desire to minimise runtime. Thus, our results show a clear trend that larger batch sizes give greater training and generalisation performance.

B.2.5. DAMPING STEPPING FACTOR

Finally, we explore the effect of different stepping factors by setting ω_{inc} to values in $\{2^{0.0}, 2^{0.2}, 2^{0.4}, \cdots, 2^{2.0}\}$, then choosing a symmetric $\omega_{\text{dec}} = \frac{1}{\omega_{\text{inc}}}$. Our results are plotted in Figure 12.

Similarly to learning rate clipping, the impact of different damping stepping factors only becomes most apparent when damping plays a key role in stabilising the optimiser, which does not happen in this Fashion-MNIST test case. However, the plots match our subjective observation that the behaviour at the very start of training is critical to defining the optimisation trajectory, with a high variance at around 2 s of runtime indicating an increased sensitivity here. Moreover, the results reinforce our intuition that the exact factor by which the damping λ is modified is not crucially important, so long as AdamQLR is capable of making rapid adjustments over successive optimisation iterations when this becomes necessary.

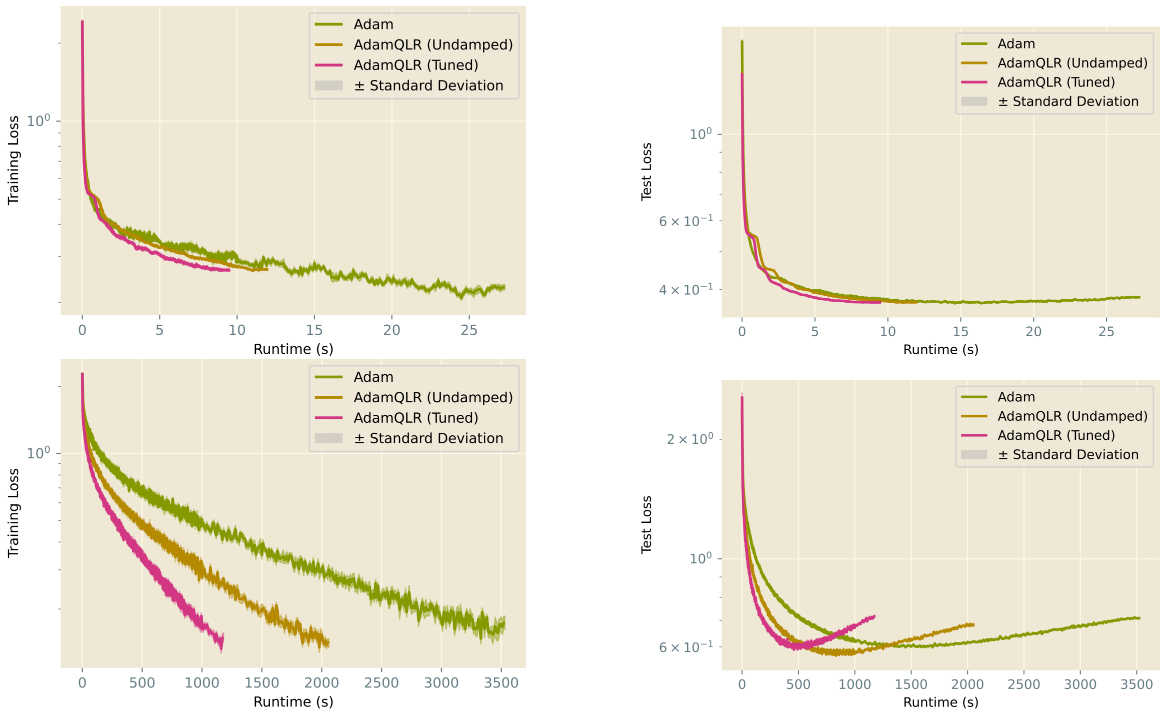

Figure 13: Evolution of Levelberg-Marquardt damping, as measured by Training (left) and Test (right) loss on Fashion-MNIST (top) and CIFAR-10 (bottom)

B.3. Ablation Studies

In addition to the algorithms plotted in Section 4, we conduct additional experiments to study the impact of different components of *AdamQLR* on its overall performance. Specifically, we examine the effects of Levenberg-Marquardt damping and the choice of curvature matrix used to construct our quadratic model. We use the same experimental configuration as in Section 4, including hyperparameter tuning with ASHA, and plot bootstrapped average trends over 50 repetitions of the best hyperparameters found.

B.3.1. LEVENBERG-MARQUARDT DAMPING

Appropriate damping is viewed as a necessity in many second-order algorithms in order to defend against degenerate parameter updates, and Figure 13 examines its inclusion in *AdamQLR*. We consider vanilla *Adam* alongside two versions of *AdamQLR*: one which includes damping, and another which excludes it, and perform hyperparameter optimisation as before on each algorithm.

On Fashion-MNIST, we see minimal effect from the inclusion of damping, as the problem does not suffer greatly from degenerate parameter updates. Thus, especially when the internal model of objective space performs well and damping is pushed to very low values, the damping makes a proportionally very small difference to the updates we take. As such, while we do benefit slightly from damping here, the advantage is very slight.

On CIFAR-10, however, we see more dramatic differences from the inclusion of damping, though we note the difference in horizontal scale is likely due to different optimal batch sizes chosen

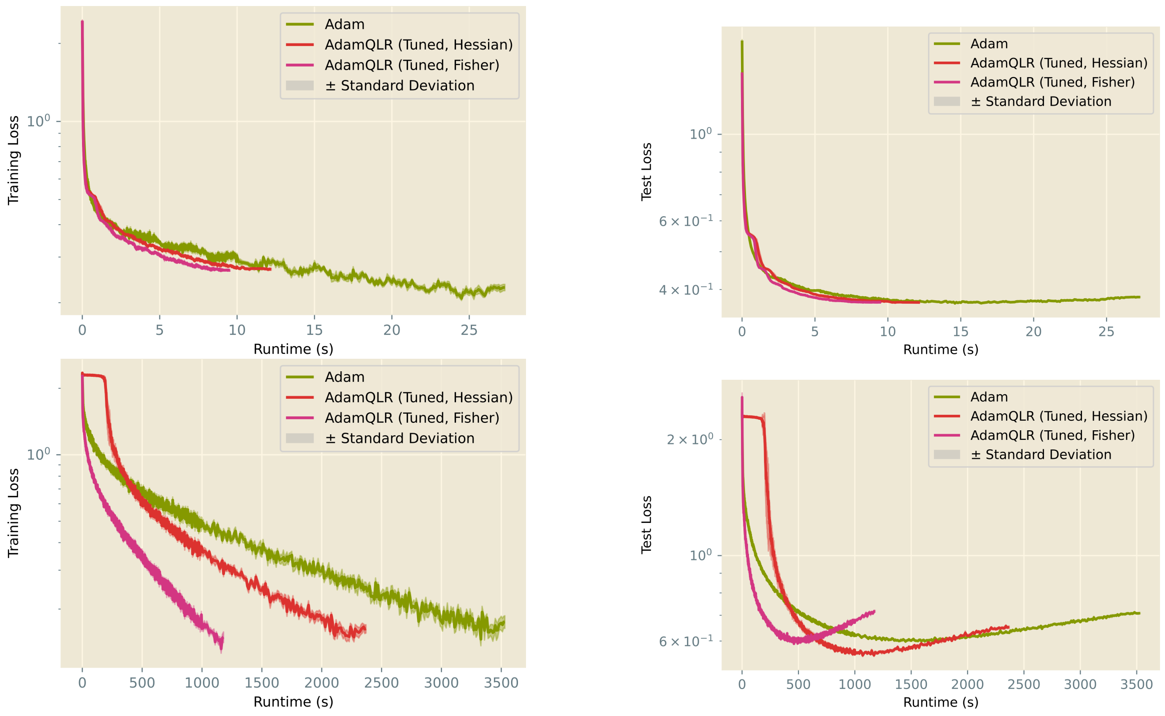

Figure 14: Evaluation of the choice of curvature matrix for the learning rate and damping calculations in *AdamQLR*

by ASHA. Adjusting for this factor of two, we see very little difference between undamped and damped AdamQLR. This result is surprising — since the model is larger and is substantially more overparameterised than in the Fashion-MNIST case, there are likely to be more parameters to which the output of our network is insensitive, corresponding to low-curvature directions of optimisation space. These low-curvature directions correspond to small eigenvalues of the curvature matrix, so a naïve curvature-based approach would take very large steps in these directions. Because the problem is inherently non-convex and non-quadratic, such large steps would not be well-motivated, and we would suffer a resulting penalty in our rapidly-excursing loss. However, during our development of AdamQLR, we observed damping to play an important role in avoiding the destabilisation of training. Further, damping clearly stabilises the algorithm enough here to allow for more aggressive optimisation over time; with all this in mind, we retain damping in our default AdamQLR approach.

B.3.2. CURVATURE MATRIX

As discussed in Appendix C, there is good reason to motivate both the Hessian and the Fisher matrices as curvatures to use to select the learning rate α at each update step. To explore their relative merits, we consider two versions of *AdamQLR*: one which uses Hessian curvature to compute a learning rate and update damping, and another which uses Fisher curvature for the same purposes. The performance of hyperparameter-optimised versions of each setting is compared alongside vanilla *Adam* in Figure 14.

On Fashion-MNIST, we see a slight advantage for Fisher curvature compared to the Hessian curvature, both of which stick generalise very slightly better than vanilla *Adam*. Curiously, the

CIFAR-10 results show the Hessian-based *AdamQLR* technique to make slow progress at the very beginning of training, then proceed similarly to the Fisher version. Again, we note that different optimal batch sizes are likely responsible for most of the horizontal scaling difference. The similarity of these results, combined with the subjectively greater stability of the Fisher version of *AdamQLR* in our development process, justify our use of the Fisher curvature as the default in our algorithm. While Fisher-vector products are more intricate than Hessian-vector products, requiring a rederived component for each loss function, a relatively small number of different loss functions see regular use in practice, so we accept this additional burden.

Appendix C. Curvature Matrices: Hessian and Fisher

In this section we discuss in more detail the two main candidates for the curvature matrix $\mathbf{C}$ in our algorithm. Recall from Section 3 that throughout we consider an arbitrary function $f(\boldsymbol{\theta})$ representing the loss function of some network parameterised by $\mathtt{theta}$.

C.1. Hessian Matrix

In this setting, the Hessian curvature matrix follows naturally from the definition of the objective function. A first derivative with respect to $\boldsymbol{\theta}$ yields the gradient vector $\mathbf{g} = (\nabla_{\boldsymbol{\theta}} f)(\boldsymbol{\theta})$, and repeating the derivative yields the Hessian $\mathbf{H} = (\nabla_{\boldsymbol{\theta}}(\nabla_{\boldsymbol{\theta}} f)^\mathsf{T})(\boldsymbol{\theta})$.

C.2. Fisher Information Matrix

To draw a connection with the Fisher matrix, we must restate our problem in a probabilistic form. We shall separate the loss function from the neural network, naming the latter $\mathbf{w}_{\boldsymbol{\theta}}(\cdot)$, and consider input-output data pairs $(\mathbf{x}, \mathbf{y})$. Let the input data have some ground truth distribution $p(\mathbf{x})$, and suppose we choose to interpret the output of the network as a probabilistic relationship, such that $\mathbf{w}_{\boldsymbol{\theta}}(\mathbf{x}) = \log p(\mathbf{y}|\mathbf{x})$.

For this model $\mathbf{w}$, the *Fisher Information Matrix* (FIM, or "the Fisher") is defined as:

$$\mathbf{F} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x})} \left[\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \boldsymbol{\theta}} \frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \boldsymbol{\theta}}^\mathsf{T} \right]. \tag{4}$$

In its exact form, the Fisher bears many favourable properties for use in optimisation: it is positive semi-definite by construction (so represents a convex space), it is amenable to efficient computation in the form of a matrix=vector product, and provides a parameterisation-independent view of the problem (as in the Natural Gradient Descent [1] family of methods).

Since $\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \boldsymbol{\theta}}$ is the Jacobian of the network output $\mathbf{w}_{\boldsymbol{\theta}}$ with respect to the parameters $\boldsymbol{\theta}$, the outer product of derivatives is readily available as part of our standard training regime. Although $p(\mathbf{x})$ is unknown, in the mini-batched training setting it is commonly approximated by the empirical distribution $\widehat{p}(\mathbf{x}$ implied by our training dataset. It is important to stress that the expectation of $\mathbf{y}$ is taken with respect to the output distribution of the network, *not* with respect to any ground-truth or empirical distribution $\widehat{p}(\mathbf{y}|\mathbf{x})$ given by the training data. However, some previous work uses the latter distribution as an approximation, resulting in the *empirical* Fisher matrix, which is known to be inferior to the true Fisher.

C.3. Adam and Fisher Matrix

While Adam is described by its authors as representing an approximation to the Fisher matrix [27], we seek here to make the connection more explicit.

The matrix computed inside the expectation of Equation 4 has as its diagonal the elementwise square of $\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \boldsymbol{\theta}}$. This is connected to the quantity $\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_{t-1})$ computed by Adam; by the chain rule, $\mathbf{g}_t$ is precisely the product of $\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \boldsymbol{\theta}}$ and the derivative of the loss function with respect to the model output. Neglecting the effect of the latter allows us to view Adam's second-moment buffer $\widehat{\mathbf{v}}_t$ as an approximation to the diagonal of the outer product in Equation 4.

Further, because $\mathbf{g}_t$ is averaged over a mini-batch of input data, we are automatically taking approximate expectations over $\widehat{p}(\mathbf{x})$ and $\widehat{p}(\mathbf{y}|\mathbf{x})$. The approximation arises because the underlying Fisher matrix is not constant, so the contributions from each mini-batch relate to different underlying curvatures. However, the argument motivates the idea that Adam develops an approximation to the diagonal of the empirical Fisher matrix in its buffer $\widehat{\mathbf{v}}_t$.

From this perspective, Adam's elementwise division by the reciprocal of $\widehat{\mathbf{v}}_t$ is simply multiplication by the inverse (approximate) empirical Fisher, and we may interpret ϵ as a fixed damping term. This picture is slightly corrupted by the square root of $\widehat{\mathbf{v}}_t$ being the quantity actually used by Adam; this operation brings the eigenvalues of the approximate empirical Fisher closer to one, in particular increasing problematic near-zero eigenvalues to more stable values, thus justifying Kingma and Ba's statement that the square root permits more "conservative" preconditioning.