CAUTIOUS DRIVER: A ROBUST, ZERO PRE-TUNING LEARNING RATE SCHEDULER

Anonymous authors
Paper under double-blind review

ABSTRACT

Choosing the correct learning rate is one of the most important parts of machine learning. Much work has been done to try to make this process more efficient, but there is still not a fully satisfactory way to do so. This paper introduces "Cautious Driver", a method to tune learning rates requiring no pre-tuning and no expensive overhead, that can achieve close to optimal performance across 9 orders of magnitude in learning rate.

1 INTRODUCTION

Selecting a good learning rate has been called "the single most important hyperamater" in machine learning (Bengio, 2012). If the rate is too low, the model will take too long to converge. If the rate is too high, the model will shoot past optimal parameters, thus failing to improve and often even getting worse. As such, the learning rate has to be carefully tuned for each individual project (Nar & Sastry, 2018)

Attempts have been made to reduce the impact of learning rates, such as developments of the Adadelta (Zeiler, 2012) and Adam (Kingma & Ba, 2014) optimizers. However, these still rely on a global learning rate that must be tuned. There are many ways to try to find the optimal learning rate for a given scenario, including grid search, random search, Bayesian optimization (Onorato, 2024), Hyperband (Li et al., 2018), and Population based methods (Jaderberg et al., 2017).However, these are time consuming, often requiring the model to be run multiple times in order to find the best via trial and error.

More modern approaches attempt to reduce the time needed to tune the learning rate. Two such approaches include AutoLRS and learning rate adaptation with hypergradient descent. However, these methods all leave something to be desired, from adding too much overhead to adding in additional Hyperparameters that must themselves be tuned.

This paper introduces the Cautious Driver; a fully online scheduler that doesn't require any time dedicated specifically to tuning, either before or during the training step, while still delivering comparable results for all learning rates tested between 1e-9 and 1e0. The method is also incredibly simple and easy to implement.

After any epoch, multiply the learning rate by 1.05, unless the model is performing "badly". If the model is performing badly, rollback to the previous best checkpoint, and divide the learning rate by 10^4. Here, performing "badly" means one of the following:

1. The training loss has doubled between epochs.
2. It is time to save a new checkpoint, but fewer than 30

While this method also has hyperparameters that could in theory be tuned, our work has shown that the default hyperparameters work across a broad range of domains, meaning in practice they can be used out of the box as is, much like the beta values in the Adam optimizer.

This paper shows that the method works across 3 very different domains of machine learning: Convolutional Neural Networks, Variational Autoencoders, and Graph Neural Networks. In all three domains, this method had a similar best validation accuracy for any learning rates tested between 1e-9 and 1e0, and always took a similar amount of epochs to reach a high degree of accuracy. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

code and implementation are available at [REMOVED] for computer vision and [REMOVED] for VAE and GNN.

2 Previous Work

2.1 Pre-Tuning

The most common way to tune learning rate is to train the model several times, with several learning rates, and select the best. Ways of doing this include grid searching, Bayesian optimization (Onorato, 2024), Hyperband (Li et al., 2018), and Population based methods (Jaderberg et al., 2017).

A slightly different way to tune this is to use gradient based methods (Do et al., 2007). This method seeks to optimize for hyperparameters by calculating the gradient of loss with respect to Hyperparameters; however, this method still results in large overheads, where "each line search takes 2-4 times the cost of [training a model] once," and "the overall hyperparameter learning procedure takes approximately 20 times the cost of [training a model] once" (Do et al., 2007)

While these methods work, they have several problems. Firstly, training the model several times over is very slow relative to training it once. Secondly, if there are multiple other parameters to optimize, such as number of layers or layer size, adding learning rate to the list only worsens the issue. Furthermore, training several times can lead to overfitting on the validation data, as the best sample may be the best simply due to random chance, rather than genuine understanding of the data.

2.2 Online / Adaptive Methods

Several ways to adjust learning rates without an expensive offline tuning have been developed. While some work better than others, all found methods introduce a new hurdle.

2.2.1 Large Overhead

These methods technically avoid offline tuning of learning rates, but require much more time during the training process for other reasons. One such method is AutoLRS (Jin et al., 2021). Every 20 or so epochs, it adjusts the learning rate, by using Bayesian optimization to choose a new learning rate and an exponential decay model to predict the validation loss. Using exponential decay modelling means this method doesn't have to do the full set of epochs for every possible learning rate. However, it still has to do several for each learning rate it wants to test, meaning that it still often doubles the training time.

Another similar practice is vSGD (Schaul et al., 2013). This method adapts the learning rate using the Hessian diagonal. However, current methods to calculate this diagonal are often just as expensive as a full backpropagation calculation, meaning that "the overhead of vSGD is roughly a factor two" (Schaul et al., 2013).

Cautions Driver avoids the additional overhead: the only time spent adjusting the learning rate is with simple scalar multiplication, and the only additional computations done are rollbacks, which it does on far less than 5% of epochs, meaning very little overhead is added.

2.2.2 New Hyperparameters

The most common pitfall of online learning rate adjustments is that these methods often add a new hyperparameter that is in practice just as sensitive and just as difficult to tune as the learning rate. One of the earlier methods was Stochastic Meta Descent (SMD), which aims to decay learning rate by exponential gradient descent, but this gradient descent relies on a hyperparameter μ, and researchers noticed that SMD "may occasionally diverge, in particular if the meta-learning μ has been chosen too high or too low". (Schraudolph, 1999).

Hypergradient descent follows a similar problem. The algorithm applies gradient descent to the learning rate as well as the model parameters. However, this hypergradient descent of the learning rate still requires a learning rate β, and different datasets in their paper required different β values (Baydin et al., 2018).

The final method of this variety was developed by Plagianakos et al. (1998), which aimed to dynamically adjust the learning rate using standard inner products between gradients and changes in parameters. However, this method was prone to diverging, and so required a max learning rate and, "tuning [the max learning rate] is quite difficult and seems to be problem dependent." (Plagianakos et al., 1998). The authors did show that a standard value could perform better than other methods at the time of this paper's publication on their tested problems, but the methods at the time did not include modern schedulers or optimizers, and no problem more difficult than font recognition was attempted. It is also possible that this would struggle with mini-batching, as it relies on complete, dataset-wide information to calculate the next learning rate.

2.3 Inability to handle mini batching

A scheduler very similar to the one developed by this paper is called "Bold Driver" (Vogl et al., 1988), from which the name "Cautious Driver" was adapted from. Bold Driver is also a loss based adapter, that increases the lr when things are "good", and rolls back and decreases the lr when things are "bad". However, Bold Driver is far more aggressive, increasing if and only if loss decreases, and rolling back and decreasing lr by a factor of 2 for every epoch where loss increases. This method struggles much more in the age of mini-batching, as loss is an inherently noisy signal now; it is not correct to say "loss went up for one epoch, the learning rate must be too high". As such, Bold Driver lacks the robustness of Cautious Driver, as we demonstrate in the experimental section.

Two additional practices that fall into this category include RProp (Riedmiller & Braun, 1993) and Delta-Bar-Delta (Jacobs, 1988) . Both algorithms rely on using gradient information to adjust learning rates, with gradients in the same direction resulting in increased learning rate and changing direction relying on decreasing learning rate. RProp uses just the sign, and DBD also includes the magnitude. However, the introduction of mini-batching led to problems in both RProp and DBD (Song & Yang, 2023) (Sutton, 1992), as gradients can vary wildly from batch to batch, and attempting to update after every batch leads to wild swings in learning rate (Song & Yang, 2023) (Sutton, 1992). The awareness of this problem eventually led to the development of modern optimizers.

2.4 Optimizers

Many optimizers have been developed in recent years, including Adam (Kingma & Ba, 2014), RMSProp (Kurbiel & Khaleghian, 2017), and Adagrad (Duchi et al., 2011). However, this is not directly related to this paper's aims, as optimizers still depend on a good initial learning rate, and more so enhances performance with a good learning rate rather than removing the need to find one. It would be possible to combine the practices of this paper with the use of an Adam optimizer, and indeed we have.

3 Theory

There are an infinitely large number of bad LR's for any given problem. We aim to apply "proof by cases" (Chartrand et al., 2013) here, breaking bad LR's into three distinct cases, and showing that Cautious Driver can handle them all.

The three classes of bad learning rates are:

1. Too low
2. Far too high
3. Slightly too high

For each, Cautious Driver has a plan to recover.

If the learning rate is too low to make much progress, the steady increase in learning rate will push the learning rate up to a healthier place.

Meanwhile, if the lr is far too high, we quickly see "large increase in the norm of the gradient during training" similar to exploding gradients seen in RNNs (Pascanu et al., 2013). This usually happens very quickly at the start of a training regiment, often taking less than 5 epochs at the start to correct.

It's with learning rates that are just slightly too high that this method can struggle. This is a region similar to what Cohen et al. have called the "edge of stability"; when the learning rate is just above two divided by the largest eigenvalue of the Hessian matrix, and "the train loss behaves non-monotonically, yet consistently decreases over long timescales" (Cohen et al., 2022). Here, progress can be very slow, but loss neither explodes nor stalls between epochs consistently enough to trigger the learning rate decrease.

However, Cautious Driver will still recover, because steady increases in learning rate pushes the lr out of the edge of stability region and into the exploding gradients region, causing the lr to decrease into a good range once again.

4 Experimental

We evaluated the Cautious Driver on three distinct domains: Convolutional Neural Networks, variational autoencoders, and Graph Neural Networks. Code was adapted from 'Fully Hyperbolic Convolutional Neural Networks for Computer Vision' (Bdeir et al., 2024) for the Convolutional Neural Network, and from "MAPE-PPI: Towards Effective and Efficient Protein-Protein Interaction Prediction via Microenvironment-Aware Protein Embedding" (Wu et al., 2024) for Variational Autoencoders and Graph Neural Networks.

The convolutional Neural Network uses tiny-imagenet (Le & Yang, 2015) and the Resnet-18 architecture (He et al., 2015). Training was performed on an H100 GPU. The GNN + VAE project aims to predict Protein-Protein interactions from information about the Protein itself as well as its microenvironment, and was trained using Nvidia A100 GPUs. The regiment uses the MAPE-PPI architecture on SHS27k (Chen et al., 2019), a dataset adapted from the STRING dataset (Szklarczyk et al., 2017) Hyperparameters were first explored on the CNN task, and then transferred over to MAPE-PPI where they are shown to generalize well.

All models and schedulers were implemented in PyTorch (Paszke et al., 2019).

4.1 Convolutional Neural Network

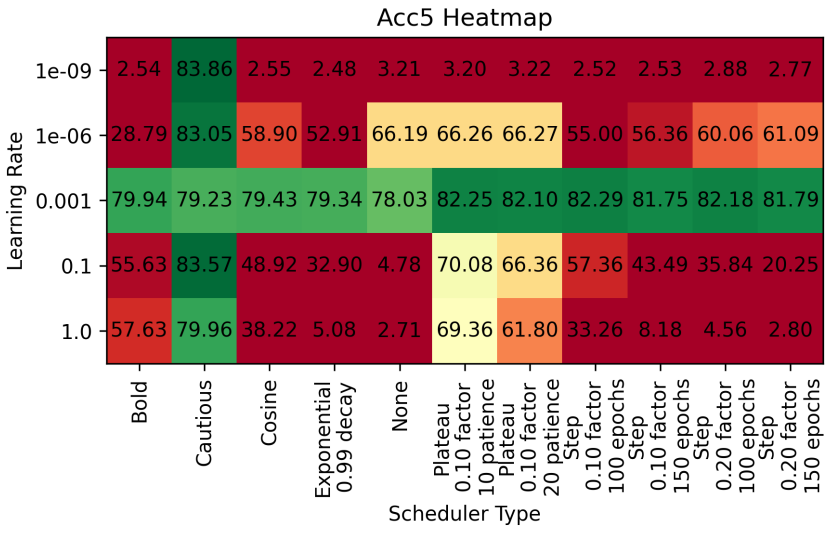


Figure 1: A comparison of Acc5 on Tiny-Imagenet against standard schedulers, showing remarkable ability to achieve a high accuracy across 9 orders of magnitude of learning rates.

Cautious Driver shows remarkable resilience to learning rates in this domain. It is able to achieve results similar to well-tuned standard schedulers across every tested learning rate. Cautious also seems to give the best performance when given learning rates known to be far too low. Other schedulers only seemed to converge within a narrow band. The sensitivity is likely exaggerated

here, however; it's likely that other schedulers would still converge with learning rate 0.0001, and possibly with 0.01. However, even granting this, Cautious clearly shows a much wider range of learning rates it can perform well in.

Note: AutoLRS and vSGD were not included in comparisons, due to lack of widespread use and difficulty of implementation.

4.1.1 CONVERGENCE TIME

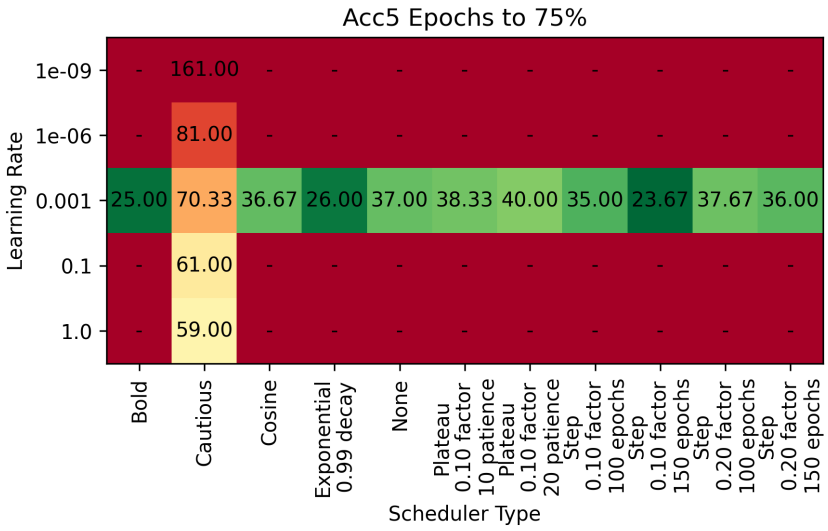


Figure 2: The number of epochs needed to reach 75% accuracy. Dashes mean not all runs reached 75% accuracy.

Unfortunately, this robustness comes at a price: it seems to take more epochs to reach a satisfactory accuracy when compared to existing schedulers given good learning rates. If we treat the 1e-9 learning rate as a stress test outlier, the number of epochs needed to converge is about 2x as many (avg 67.8 vs 33.53). If we include the 1e-9 lr, it requires 2.5 times as many epochs on average (86.47 vs 33.53).

At first glance, this would seem to suggest that this method is no better than other methods with large overhead such as AutoLRS. However, while the additional time to converge isn't ideal, it still represents a practical improvement, since needing 2x as many epochs to converge as the ideal rate will only matter if you know beforehand how many epochs you need. In reality, Cautious will often converge just fine within an allocated budget. For example, if one sets the maximum epoch to 100, and a perfectly tuned scheduler converges in 40, and Cautious in 80, both will have taken roughly the same amount of time in the fixed-max-epochs regiment, while AutoLRS will still have taken twice as long.

Even in situations with early stopping, Cautious will be roughly equal to AutoLRS (assuming that AutoLRS converges in roughly as many epochs as a well-tuned standard scheduler), as Cautious will take roughly twice as many epochs, but AutoLRS effectively spends twice as much time per epoch. Additionally, Cautious will likely be faster across an entire tuning session, as many times a model will fail to converge due to reasons other than the learning rate. In those runs, the full epoch budget will be spent, and Cautious will be 2x faster.

4.1.2 HYPERPARAMETER ROBUSTNESS

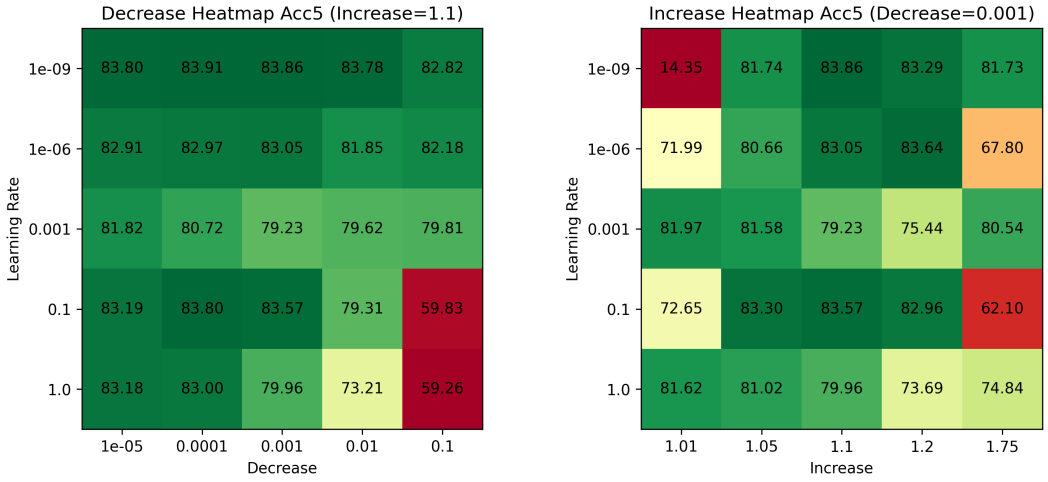


Figure 3: Performance of Cautious across different hyperparameters

The value we decrease by seems to be incredibly robust, achieving strong results across 2 entire orders of magnitude from 10^-3 to 10^-5, and possibly more. The tolerance for the increase seems much more limited; only being robust from 5% increase to a 10% increase. Any more or less results in noticeable degraded performance at certain learning rates.

From this, the best hyperparameters seemed to be a decrease factor of 10^-4 and an increase factor of 1.05. These were the values used when using Cautious Driver in the context of GNNs and VAEs.

4.2 GNN + VAE

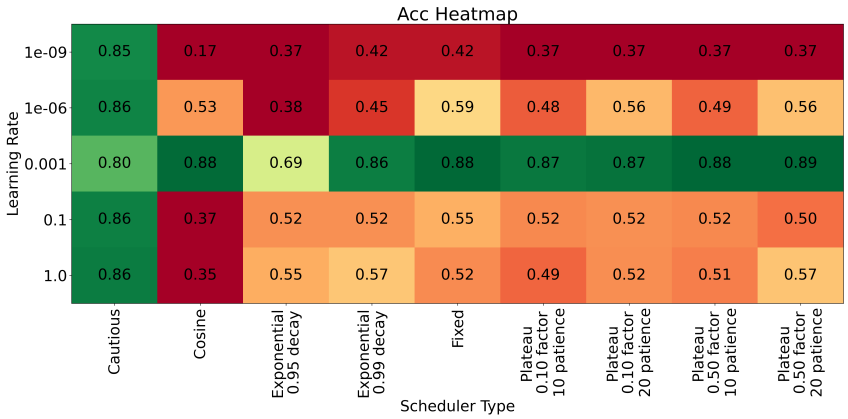


Figure 4: Scheduler Accuracy on MAPE.

Once again, Cautious Driver shows extraordinary resilience across learning rates; achieving at least 80% average accuracy across 9 orders of magnitude. Notably, it seems to perform worst, relative to its own performance at other learning rates, at the one learning rate where every other scheduler performs best relative to their own performance elsewhere. The reason for this peculiarity is unclear. We presume, given that fixed and plateau performed better than exponential 95%, this is just a good LR to be at, and Cautious moved out of it. Even still, the performance there is within the ballpark of what other schedulers achieved (albeit worse than the best of the bunch) at that learning rate, and far better than anything outside the narrow range of good performance for the others.

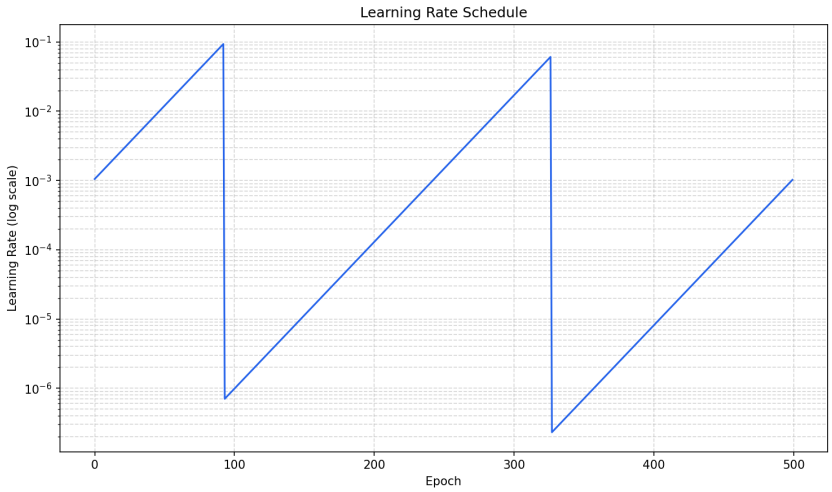


Figure 5: Learning rate vs. Epoch for a run with initial LR 0.001. Learning rate initially rises to the area where other schedulers performed poorly.

4.3 Time to Convergence

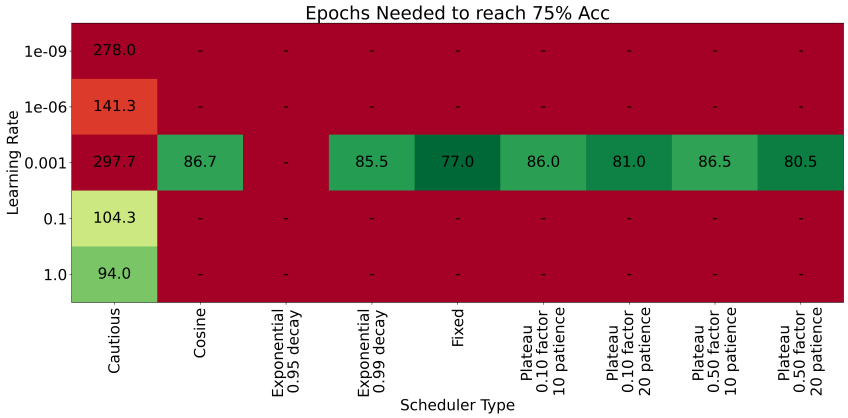


Figure 6: Epochs needed to reach 75% accuracy. Dashes mean not all runs reached 75%.

A similar story appears here as with computer vision. Cautious Driver always converges, but takes about 2x as many epochs on average when compared to a well tuned standard scheduler.

5 Conclusion

In this paper, we introduce Cautious Driver, a scheduler that works with existing optimizers that all but removes the need to tune learning rates. This method does not require any pre-tuning and only introduces trivial overhead. While it does have hyperparameters, experiments show that the standard settings generalize across entirely different domains, meaning they do not need to be tuned in practice. Limitations still exist however, as it unfortunately takes around 2x as many epochs to converge compared to an oracle scheduler. Further work could see if there are ways to tune this method to avoid this penalty, as well as see if the method works in domains not tested in this paper, such as Natural Language Processing and Reinforcement Learning.

AI USE STATEMENT

AI was used to source papers and find existing code bases to test on. For writing, AI helped with transforming all manually written citations into citep commands. AI was also used to ideate over which specific learning rates were worth testing.

ETHICS STATEMENT

The authors report no conflict of interest.

REPRODUCIBILITY STATEMENT

All code is available on GitHub, including scripts to run.

ACKNOWLEDGMENTS

The authors would like to thank Digitial Alliance for providing computing resources.

REFERENCES

A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. *arXiv*, 2018. URL http://arxiv.org/abs/1703.04782.

A. Bdeir, K. Schwethelm, and N. Landwehr. Fully hyperbolic convolutional neural networks for computer vision. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ekz1hN5QNh.

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *arXiv*, 2012. URL https://arxiv.org/abs/1206.5533v2.

G. Chartrand, A. D. Polimeni, and P. Zhang. *Mathematical Proofs: A Transition to Advanced Mathematics*. Pearson Education, 2013.

Muhao Chen, Chelsea J T Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. Multifaceted protein–protein interaction prediction based on siamese residual rcnn. *Bioinformatics*, 35(14):i305–i314, 07 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz328. URL https://doi.org/10.1093/bioinformatics/btz328.

J. M. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv*, 2022. URL http://arxiv.org/abs/2103.00065.

Chuong B. Do, Chuan-sheng Foo, and Andrew Ng. Efficient multiple hyperparameter learning for log-linear models. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/851ddf5058cf22df63d3344ad89919cf-Paper.pdf.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv*, 2015. URL http://arxiv.org/abs/1512.03385.

R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1 (4):295–307, 1988. doi: 10.1016/0893-6080(88)90003-2.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017. URL https://arxiv.org/abs/1711.09846.

Y. Jin, T. Zhou, L. Zhao, Y. Zhu, C. Guo, M. Canini, and A. Krishnamurthy. Autolrs: Automatic learning-rate schedule by bayesian optimization on the fly. *arXiv*, 2021. URL http://arxiv.org/abs/2105.10762.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. URL https://arxiv.org/abs/1412.6980.

T. Kurbiel and S. Khaleghian. Training of deep neural networks based on distance measures using rmsprop. *arXiv*, 2017. URL http://arxiv.org/abs/1708.01911.

Y. Le and X. Yang. Tiny imagenet visual recognition challenge, 2015. URL https://api.semanticscholar.org/CorpusID:16664790.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv*, 2018. URL http://arxiv.org/abs/1603.06560.

Kamil Nar and Shankar Sastry. Step size matters in deep learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e8fd4a8a5bab2b3785d794ab51fef55c-Paper.pdf.

Gabriele Onorato. Bayesian optimization for hyperparameters tuning in neural networks. 2024. URL https://arxiv.org/abs/2410.21886.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv*, 2013. URL http://arxiv.org/abs/1211.5063.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

V. Plagianakos, D. Sotiropoulos, and M. Vrahatis. An improved backpropagation method with adaptive learning rate. Technical report, Department of Mathematics, University of Patras, 1998. URL https://old.math.upatras.gr/~vrahatis/papers/other/PlagianakosSV98d_Technical_Report_NoTR98-02_Dept_Math_UPatras_1998.pdf.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pp. 586–591, San Francisco, March 1993.

T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. *arXiv*, 2013. URL http://arxiv.org/abs/1206.1106.

N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *ICANN99: Ninth International Conference on Artificial Neural Networks*, pp. 569–574, 1999. doi: 10.1049/cp:19991170.

Z. Song and C. Yang. An automatic learning rate schedule algorithm for achieving faster convergence and steeper descent. *arXiv*, 2023. URL http://arxiv.org/abs/2310.11291.

R. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Eighth International Conference on Machine Learning*, pp. 171–176. MIT Press, 1992. URL http://www.incompleteideas.net/papers/sutton-92a-remastered.pdf.

D. Szklarczyk, J. H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N. T. Doncheva, A. Roth, P. Bork, L. J. Jensen, and C. von Mering. The string database in 2017: Quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Research*, 45(D1):D362–D368, 2017.

T P Vogl, J K Mangis, A K Rigler, W T Zink, and D L Alkon. Accelerating the convergence of the back-propagation method. *Biol. Cybern.*, 59(4-5):257–263, September 1988.

L. Wu, Y. Tian, Y. Huang, S. Li, H. Lin, N. V. Chawla, and S. Z. Li. Mape-ppi: Towards effective and efficient protein-protein interaction prediction via microenvironment-aware protein embedding. *arXiv*, 2024. URL https://arxiv.org/abs/2402.14391.

M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv*, 2012. URL http://arxiv.org/abs/1212.5701.