CYCLICAL LOG ANNEALING AS A LEARNING RATE SCHEDULER

Philip Naveen
University of Virginia
Charlottesville
philipnaveen@email.virginia.edu

ABSTRACT

A learning rate scheduler is a predefined set of instructions for varying search stepsizes during model training processes. This paper introduces a new logarithmic method using harsh restarting of step sizes through stochastic gradient descent. Cyclical log annealing implements the restart pattern more aggressively to maybe allow the usage of more greedy algorithms on the online convex optimization framework. The algorithm was tested on the CIFAR-10 image datasets, and seemed to perform analogously with cosine annealing on large transformer-enhanced residual neural networks. Future experiments would involve testing the scheduler in generative adversarial networks and finding the best parameters for the scheduler with more experiments.

***K*eywords** Deep-Learning, Convolutional Neural Networks

1 Introduction

1.1 Gradient Based Optimization

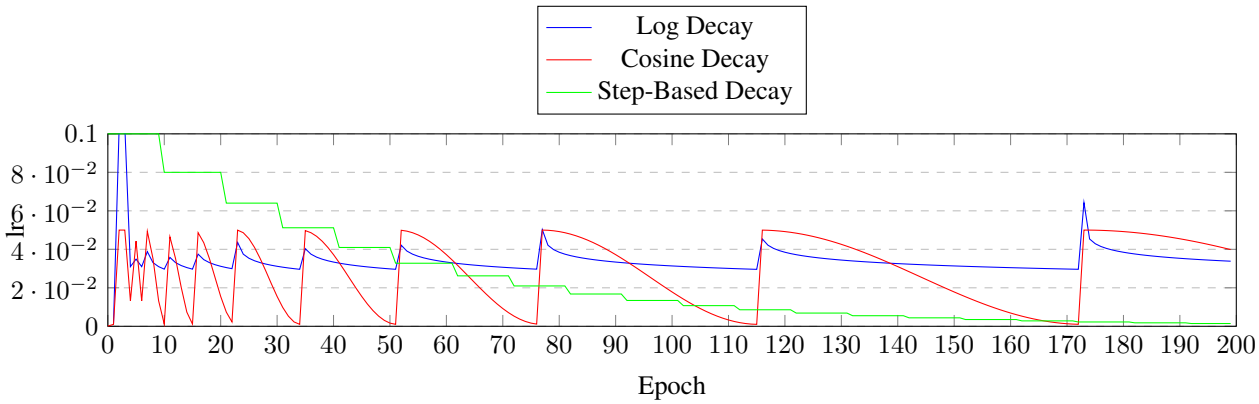

Figure 1: Learning Rate Schedulers; shared parameters are initial decay epochs=1, min decay lr=0.001, restart interval=1, restart interval multiplier=1.5, restart lr=0.05, warmup epochs=1, warmup start lr=0.0001 for annealing schedulers.

[h]

Deep neural networks (DNN) are becoming the staple for a growing majority of classification problems [10, 3], and are gaining use in generative algorithms such as the large language models. They typically perform well on training sets with thousands of parameters, and are able to produce difficult approximations easily. The main caveat to these algorithms is the long training time, which prompts the use of expensive computatioal equipment such as GPUs, TPUs [11]. Algorithms that speed up the training proccess are one of the most lucrative parts of the feild at the moment.

Suppose a model has n parameters, is trained through the process of minimizing the function $f : \mathbb{R}^n \to \mathbb{R}$. Normally, we use iterative stochastic gradient descent at time step t to adjust that parameter vector $\mathbf{x}_t \in \mathbb{R}^n$. We do this by taking gradient information $\nabla f_t(\mathbf{x}_t)$ which we get using some small batch b of datapoints [9, 11].

There is stochastic gradient descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t) \tag{1}$$

and with second order information

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{H}_t^{-1} \nabla f_t(\mathbf{x}_t) \tag{2}$$

that is usually avoided using LBFGS because the difficulties behind stochasticity in $\nabla f_t(\mathbf{x}_t)$. Typically, the use of momentum based optimization and approximating second order gradient information is used to achieve a superior regret bound and avoid the difficulty and unpredictability that comes with conditioning a parameter space for a model [9].

1.2 Restarting Mechan

In any form of optimizing multimodal functions regardless of the variation in divergence for each minimum or maximum, it is common there exists a single minima of interest. In his paper relating to global optimization, Guo et al suggested a multi armed bandit and ranking system. They further use variations of reward and restless algorithms to achieve better convergence results on objective functions with multiple extrema such as the Ackley, Griewank, and Rastrigin functions [5].

This paper takes a different approach more akin to Loshchilov and Hutter by using the idea of shifting the learning rate using a scheduling system to prevent the optimizer from getting stuck in saddle points and local minima [11, 8].

Restarts are usually used to avoid converging to local minima rather than dealing with multimodality. Suppose we have a collection of non-smooth, yet convex optimization problems. About a decade ago, Yang and Lin found that a linear convergence rate is achievable by geometrically decreasing the learning rate throughout training [1].

Suppose we have η^i_{min} and η^i_{max} to define the range of learning rates, so that generally $\eta^i_{min}, \eta^i_{max} \in \mathbb{R}$. The T_{cur} counts for the accumulation of epochs for training a model through a process such as backpropagation. In their paper on warm restarts to improve the convergence of stochastic gradient descent, Loshchilov and Hutter suggested cyclical cosine annealing

$$\eta_t = \eta^i_{min} + \frac{1}{2}\left(\eta^i_{max} - \eta^i_{min}\right)\left(1 + \cos(\frac{T_{cur}}{T_i}\pi)\right) \tag{3}$$

for the purpose of rescheduling the learning rates in convolutional neural networks [11].

This project introduces an alternative approach similar to theirs that uses logarithmic properties instead of those of cosine. We decide to vary the base of the logarithmic expansion using a difference in the current range of learning rates. Therefore,

$$\eta_t = \left|\eta^i_{min} + \frac{1}{2}\left(\eta^i_{max} - \eta^i_{min}\right)\left(1 + \log_{\left(\max\limits_{\eta_j \in \eta^i_{min}}\right)^{-1}}(\frac{T_i}{T_{cur}}\pi)\right)\right| \tag{4}$$

is what we have introduced as a different restart mechanism. We introduced this alternative method with the reasoning that using the difference as a base in an asymptotic function such as a logarithm would help mitigate the range of η without any edge cases such as division by 0.

A higher learning rate can give fast, yet unreliable convergence due to overfitting, but can be mitigated with dropout [10]. Further, a high learning rate without proper batch normalization for prolonged periods during training can cause neurons using ReLU to die [15]. Cosine annealing lowers η_t at roughly the same pace as it increased it previously. This approach instead increases η_t almost asymptotically for a select few T, and then anneals it at a logarithmic pace. The purpose of this was to rapidly diverge from a solution in a local minima in the span of a few T, and then reduce η_t such that gradient descent can locate the global minima at a reasonable convergence rate such that there is no rapid overfitting from a prolonged high η_t.

2 Experiments

2.1 CIFAR-10 Classification

To test the learning rate schedulers, we replicated a pretrained model on a benchmark dataset CIFAR-10 [3]. The model of 63.5 million parameters trained using stochastic gradient descent. The loss of interest was categorical crossentropy to

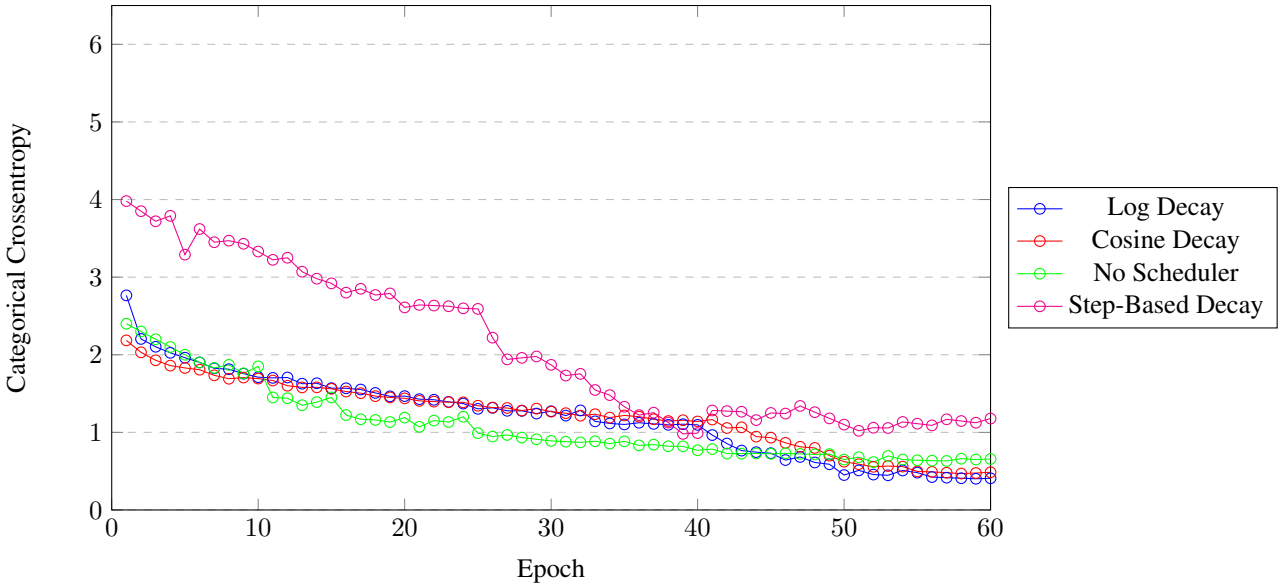

Figure 2: The Effect of Learning Rate Schedulers on CIFAR-10 Image Classification using ResNet34

ensure the model had some ambiguity in the output layer such that each output neuron is not strictly mututally exclusive. Each image was preprocessed using the pipeline created by Zagoruyko and Komodakis such that we augment each image using horizontal flips and randomized crops for 4x4 patches of pixels on the 32x32 image data. We also add pixel by mean, ZCA whitening, and global contrast normalization. We also use contrast stretching equalization [10, 11].

SGD was used instead of Adam because it is less adaptable, and thus less compatable with a constant learning rate, as the gradient distribution can get distorted faster. The initial learning rate was set to η_0 = 0.0001, weight decay to 0.0005, dampening to 0, momentum to 0.9 and minibatch size to 128 [11]. We used three different schedulers and a control variable of no scheduler to compare. We used init decay epochs of 10, minimum decay for η_t of 0.001, restart interval of 10, restart interval multiplier of 2.0, restart η_t of 0.1, warmup epochs of 5, warmup start η_t of 0.01. We trained ResNet34 [7] using the three schedulers for a total accumulation of 60 epochs due to constraints with GPU access.

In figure 3 the results of this is shown. These are the averages of each epoch, which is why the model's loss spikes for the restarts are not apparent, as the spikes sometimes stalled depending on the images in a particular set of batches in an epoch, for a set of epochs $T_n, T_{n+1}...T_{n+10}$ between restarts we assigned. Thus, the highs and lows of the restarts cancelled out for each period for both cosine and log decay. The graph depicts the loss reduction of a large convolutional neural network when using learning rate schedulers through a budget of 60 epochs. When observing the graph, two things are immediately apparent:

1 For dealing with categorical crossentropy loss, it seems that the loss is rather stable Poisson or deserialization loss when using learning rate schedulers, likely due to abstaining from using softmax, hardmax, or argmax in favor of a sigmoid dense layer for outputs.

2 Of the three schedulers and the control, it seems that log annealing and cosine annealing perform roughly the same for the majority of the training. However, we note that log annealing performs worse in the beginning and better at the end. This is because a high η_t can result in higher variability in $\frac{\partial J}{\partial w_{j,i}^2}$ from inheritly more extreme predictions from less exposure to training data.

Considering these observations, there is a strong case that log annealing is at the very least capable of the same performance and exhibits similar properties to cosine annealing.

2.2 Transformer CIFAR-10 Classification

Suppose we introduce a transformer into a larger residual model. For this series of experiments, the model was over 23 million parameters and included one multi-head attention transformer before the 48 convolution layers [12]. Note that we added a mean pooling layer after the transformer, and concatenated it to the initial ResNet50.

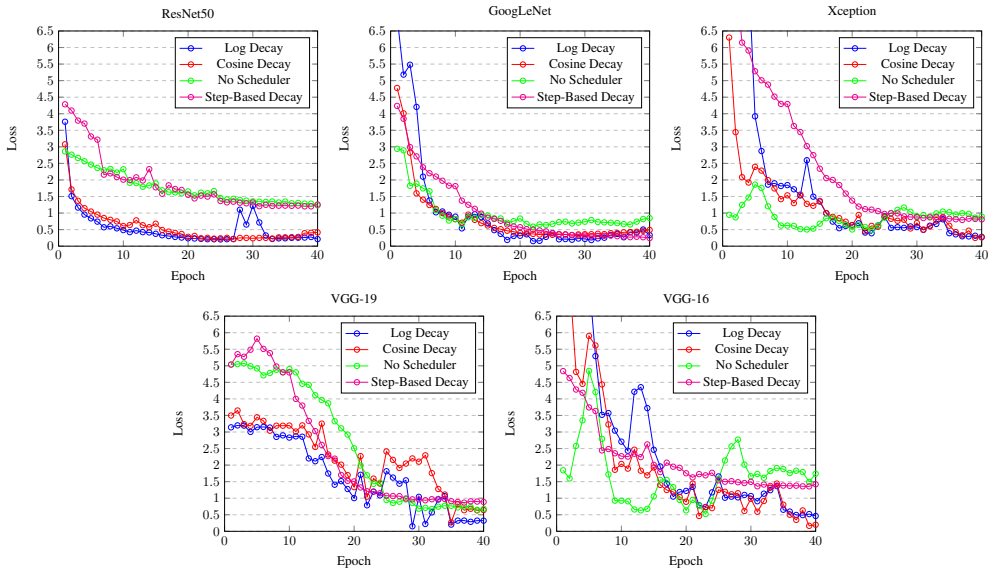

Figure 3: The Effect of Learning Rate Schedulers on CIFAR-10 Image Classification using Transformers

Then we trained this ensemble model on CIFAR-10, using a test train split, with a 32 batch size. Once again, we augment each image using horizontal flips and randomized crops for 4x4 patches of pixels on the 32x32 image data. We also add pixel by mean, ZCA whitening, and global contrast normalization. We also use contrast stretching equalization. We used Adam to maximize convergence speed and to replicate a real-life setting on which a network this size would be trained [9, 10, 12]. We did not use grid search or Bayesian optimization for hyperparameterization. We trained the model for 40 epochs, with a restart for cosine and log decay every 10 epochs. We used no warm up period and basically had no actual annealing effect. We repeated this process using the same transformer for GoogLeNet, Xception, VGG-19 and VGG-16, which are roughly 4, 23, 144, 138 million parameters respectively [14, 2, 13]. The loss was sparse categorical crossentropy, as each class in CIFAR-10 is mutually exclusive to one another [3].

The results of the training loop described above are shown in figure 3. At a first glance, three observations from these graphs can be drawn:

1 A steady progression in loss is apparent in the smaller models under 50 million parameters. In the larger models, there is some substantial evidence provided by the VGG ensembles, that the training loss does not stabilize even with the intervention with a restart on the 10th epochs.

2 While yes, log decay does provide a decent loss reduction, indicating the convergence of the predicted to actual minima on a weight bias field, its initial loss is quite high, even in respect to other schedulers like cosine decay. By observing the behavior of the restarts themselves for both, log annealing has a high initial loss for the first restart, but significantly less than cosine annealing for the subsequent ones. Examples of this are depicted for $T = 20, 21, 22, 23, 24$ on VGG-19 and for $T = 10, 11, 12, 13, 14, 15, 16$ on Xception.

3 Between log decay and cosine decay, apart from VGG-16, where cosine decay is a clear winner, for the rest of the models, between the two annealing schedulers, the ending loss is nearly the same. However, when examining the path that the models took to achieve the end sparse categorical crossentropy at $T = 40$, we see that log decay is advantageous relative to no scheduler, or a scheduler without restarts.

Across 40 epochs, it seems that indeed cosine annealing was effective in reducing loss in a large transformer-based convolutional neural network, as the past studies also concluded. Log decay exhibits similar properties, and also yielded decent results.

3 Discussion

From our two experiments, there is evidence suggesting that logarithmically varying the stepsize of provides a reasonable convergence for algorithms that use a limited budget on the online convex optimization framework, particularly SGD and Adam [6].

Contrary to cosine annealing, log annealing employs a harsher restarting mechanism, more akin to a spike rather than a wave [11]. However, it can still be geometrically annealed. In theory, log annealing would likely benefit more from an extended warmup period such that it could benefit late stage training of the residual networks. The exploding gradients could then therefore be neglected via the use of rapidly expanding regret values constrained to a small number of batches passed via backpropagation [9].

The initial purpose of using a restart scheduler all together is to get the best anytime performance of a model. It should be noted that though these experiments used the same configurations for the experiments across any geometrically annealed learning rate scheduler, those may not be the best settings. Log annealing in particular has a much higher potential for variability in the schedule because of the more aggressive asymptotic type increases in loss. Log annealing generally is more suceptable to prolonged high η_t values. Lowering η^i_{max} is not a workarounnd here, as the learning rate spikes rely on that high max value, so those would be effected rather than the middle range, which we are targeting. However, tweaking other parameters such as minimum decay learning rate, restart intervals, restart interval multipliers, and the restart learning rate itself. These stepsize spikes provide a more temporary effect than the longer restarts from cosine annealing, which could show some potential for more consistent convergence for a neural network [11]. This means that much more greedy stochastic solvers than Adam or SGD may have better use cases in deep learning, as they will not go overkill with a prolonged period of increased learning rates, but could in theory pair nicely to escape a local minimum on the weight bias fields.

In the event that a particular restart spike is too harsh and causes an irreversible jump in loss, a workaround could be creating a breakpoint in the training loop in the event of an extreme increase in loss. For example, the condition in our first experiment was to hault training IFF $T_t < T_n$ and

$$-\sum_{i}^{C} t_i \log(f(s)_i) > 4.5 \tag{5}$$

such that

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{6}$$

where C is the number of output neurons in the multilayer perceptron; note that we included a buffer such that the training would never immediately from a high initial loss [10, 3, 4]. In our case $T_t = 2$. Though this hault condition was in place, it was never used, which can be seen in figure 2. After the stop, the model's weights could be saved and retrained using an adjusted version of the scheduler [11].

Our results from the second experiment showed that when using a transformer enhanced residual neural network, our learning rate scheduler can achieve better convergence anytime. It is best to mention here that a flaw in our experiments was the low amount of epochs spent training the models relative to other researchers, who often trained for five times the amount of epochs we did. This was simply due to time constraints and hardware constraints. In particular, the VGG models are both over 120 million parameters, and almost certainly would show more accurate and reproducible results with longer periods of training [13]. Further work with these transformer models should be done, to ensure that the effect of geometrically annealing the learning rate scheduler would benefit these models as well.

4 Conclusions

This short paper investigates the possibility of using a logarithmic learning rate scheduler. It further goes into the surface level properties of residual neural networks and transformer models when training on this schedule. It was found that there are quite a few possibilities that come from a relatively minute change in the way η_t is varied through training.

Models using this scheduler achieved similar, and sometimes better results than cosine decay and step-based decay, albeit those results were short term relative to the experiments performed by Loshchilov and Hutter. Our experiments were conducted on CIFAR-10.

Future work could extend to a wide variety of tests when varying the parameters of the scheduler to find the best combination of how extreme the restarts are, and how frequent they occur. Log annealing comprises a larger range of η_t than cosine annealing, however it crosses that range significantly w.r.t. T. Other future work would involve testing the scheduler on generative adversarial networks, as the results could be reproduced using the two player minimax setting [4].

While the experiments show some promising results, this manuscript is merely an arXiv preprint at the moment, and there lacks the extensive testing to make broad claims on log annealing's use case, and where it outshines other learning rate schedulers, as that is an inherently ambiguous and difficult question to answer.

5 acknowledgements

I would like to thank Srinivasan Kannan and Zaid Contractor for suggesting some small edits and formatting changes the paper. Special thanks also to Professor Shangtong Zhang for endorsing the preprint.

References

[1] "A Stochastic Gradient Method with Linear Convergence Rate for a Class of Non-smooth Non-strongly Convex Optimizations". In: *CoRR* abs/1510.01444 (2015). Withdrawn. arXiv: 1510.01444. URL: http://arxiv.org/abs/1510.01444.

[2] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV].

[3] Raveen Doon, Tarun Kumar Rawat, and Shweta Gautam. "Cifar-10 Classification using Deep Convolutional Neural Network". In: *2018 IEEE Punecon* (2018), pp. 1–5. URL: https://api.semanticscholar.org/CorpusID:195736921.

[4] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Neural Information Processing Systems*. 2014. URL: https://api.semanticscholar.org/CorpusID:261560300.

[5] Phillip Guo and Michael C. Fu. "Bandit-Based Multi-Start Strategies for Global Continuous Optimization". In: *2022 Winter Simulation Conference (WSC)* (2022), pp. 3194–3205. URL: https://api.semanticscholar.org/CorpusID:259100800.

[6] Elad Hazan. "Introduction to Online Convex Optimization". In: *Found. Trends Optim.* 2 (2016), pp. 157–325. URL: https://api.semanticscholar.org/CorpusID:30482768.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[8] Chiheon Kim et al. "Automated Learning Rate Scheduler for Large-batch Training". In: *ArXiv* abs/2107.05855 (2021). URL: https://api.semanticscholar.org/CorpusID:235828729.

[9] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60 (2012), pp. 84–90. URL: https://api.semanticscholar.org/CorpusID:195908774.

[11] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *arXiv: Learning* (2016). URL: https://api.semanticscholar.org/CorpusID:14337532.

[12] Yulin Peng. "Images Classification Integrating Transformer with Convolutional Neural Network". In: *Advances in Engineering Technology Research* (2023). URL: https://api.semanticscholar.org/CorpusID:260526953.

[13] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[14] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[15] Bing Xu et al. "Empirical Evaluation of Rectified Activations in Convolutional Network". In: *CoRR* abs/1505.00853 (2015). arXiv: 1505.00853. URL: http://arxiv.org/abs/1505.00853.