Learning Rate Optimization for Online Deep Learning

Anonymous submission

Abstract

Efficient training via gradient-based optimization techniques is an essential building block to the success of deep learning. Extensive research on the impact and the effective estimation of an appropriate learning rate has partly enabled these techniques. Despite the proliferation of data streams generated by IoT devices, digital platforms, etc., previous research has been primarily focused on batch learning, which assumes that all training data is available a priori. However, characteristics such as the gradual emergence and non-stationarity of data pose additional challenges. Therefore, the findings on batch learning may not be applicable to deep learning in streaming environments. In this work, we seek to address this knowledge gap by (i) evaluating and comparing typical learning rate schedules and optimizers, (ii) exploring adaptations of these techniques, and (iii) providing insights into effective learning rate tuning in the context of stream-based deep learning.

Introduction

Deep learning models have demonstrated exceptional performance in various domains, with the choice of optimizer playing a crucial role in achieving outstanding results. In the context of batch learning, where all data is available simultaneously, extensive research has been conducted to explore different optimizer choices and optimization techniques for deep learning architectures. Numerous methods have emerged to effectively update the weights of these architectures. However, the investigation of optimizer choices in online learning, where models must adapt to evolving data streams, remains relatively limited.

This paper aims to bridge this knowledge gap by investigating how the choice of optimizer changes when transitioning from batch learning to online learning scenarios. Specifically, we address the following research questions:

- How does the choice for the optimizer change from batch to online learning?
- What are practical choices for gradient-based online training of deep architectures in online learning?
- Are adaptive optimization methods better suited in Online Deep Learning?

LR tun-
graphic
?

For the first research question, we explore how the selection of an optimizer differs when moving from the traditional batch learning setting to the dynamic online learning scenario. We examine the suitability of various optimizer choices in online learning and their impact on model performance.

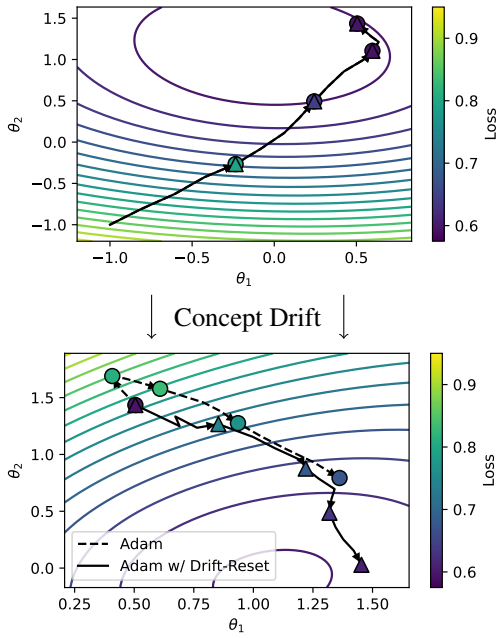

Figure 1: Parameter trajectory of Adam (Kingma and Ba 2017) with or without drift adaptation on synthetic data stream with abrupt concept drift. Marker colors depict the expected prequential loss over the last 16 data instances.

The second research question investigates practical choices for gradient-based online training of deep architectures. We analyze different optimization techniques and explore their effectiveness in adapting to evolving data streams while maintaining model performance. The third research question focuses on the performance of adaptive methods in online deep learning scenarios. These methods dynamically adjust the learning rate based on gradient characteristics, allowing models to adapt more effectively to changing data patterns. We compare the performance of adaptive methods against other optimization approaches to determine their suitability for online deep learning tasks. Through our

in-depth analysis and experimentation, we aim to enhance our understanding of optimizer choices in online deep learning. By shedding light on the impact of optimizers, learning rates, and batch sizes, and comparing the effectiveness of adaptive methods, we aim to enable researchers and practitioners to make informed decisions when selecting optimization techniques for real-time learning tasks.

Learning Rate Scheduling

In the following, we will explain the theoretical background of first-order stochastic optimization and the differences between its application in traditional batch learning and online learning in terms of impact of the learning rate and its optimization.

First-order stochastic optimization techniques like

First order gradient-based optimization approaches like stochastic gradient descent and its derivatives aim to iteratively minimize the error of a DL model using stochastic gradients of a loss function $\mathcal{L}()$ at each step t. We denote the gradient of the prediction error for data samples $(X_t, y_t) \sim p_t$ with respect to model-parameters θ as

$$g_t = \nabla_\theta \mathcal{L}(y_t, f(X_t; \theta)), \tag{1}$$

where $\mathcal{L}$ represents a loss function. Based on these gradients, SGD yields parameter values

$$\theta_t = \theta_0 - \sum_{i=0}^{t} \eta_t \cdot g_t, \tag{2}$$

where η_t denotes the learning rate at timestep t. The task of optimizing the learning rate in a batch learning setting can then be defined as

$$\begin{aligned} &\min_{\eta_0,\dots,\eta_T} \sum_{i=1}^{V} \mathcal{L}(y_i, f(X_i; \theta_T)) \\ &\text{s.t.} \qquad X_i, y_i \sim p^{(v)} \qquad \forall i \in 1, \dots, V, \end{aligned} \tag{3}$$

where $p^{(v)}$ is a distribution of held-out validation data and T the total number of training steps.

The primary trade-off to consider with respect to the choice of η is that increasing the learning rate speeds up convergence, but at the same time also increases stochasticity and therefore leads to the divergence of the training criterion beyond a certain threshold. (Bengio 2012). In fact, Smith and Le (2018), found that when modelling SGD as a stochastic differential equation, the "noise scale" is directly tied to η (Smith and Le 2018). In biological terms, increasing the learning rate increases plasticity, whereas decreasing it increases stability.

To ensure fast convergence at the start of training, while mitigating jumping around potential minima at later stages it is common to use a decaying schedule starting with a large learning rate that decreases over time. An additional benefit of this approach is that of potentially better generalization, since larger learning rates can help skipping over sharp minima with poor generalization (Hochreiter and Schmidhuber 1997; Chaudhari et al. 2017). Some have likened this procedure to simulated annealing, which shifts its focus from exploration at high temperatures to exploitation once temperatures have sufficiently decreased (Smith et al. 2018).

Commonly used forms of decay is exponential decay, where η_t is calculated as

$$\eta_t = \eta_0 \cdot \gamma^t, \tag{4}$$

with $\gamma < 1$, and stepwise decay, which for a regular interval between steps of length s is given as

$$\eta_t = \eta_0 \cdot \gamma^{\lfloor t/s \rfloor}. \tag{5}$$

Other popular options include cyclic learning rate schedules which oscillate η between two values over a predefined interval. For a basic triangular cycle, the learning rate calculates as

$$\eta_t = \eta_0 + \frac{\hat{\eta} - \eta_0}{2s} \cdot \min_i \{|t - i \cdot s|\}, \tag{6}$$

with $\hat{\eta}$ being the learning rate at the middle of each cycle of length s. Some studies (Smith 2017; Smith and Topin 2018) have found cyclic schedules to significantly speed up the convergence of neural networks even when compared to adaptive techniques like Adam in some cases (Kingma and Ba 2017). While there are many other learning rate schedules, we focus on the use of the three aforementioned schedules within data streaming applications in this work. For a comprehensive overview and detailed analysis on learning rate policies, we refer to Wu et al. (2019).

In contrast to conventional batch learning, the impact of the learning rate in stream-based deep learning is a lesser studied issue. According to Bifet et al. (2010) a machine learning model operating in such an environment must be able to

- **R1:** process a single instance at a time,
- **R2:** process each instance in a limited amount of time,
- **R3:** use a limited amount of memory,
- **R4:** predict at any time,
- **R5:** adapt to changes in the data distribution.

These requirements give rise to the *prequential* scheme of evaluating machine learning models (Bifet et al. 2010), in which each instance (X_t, y_t) in the data stream is first used to test and then to train the model ensuring that testing is done exclusively on unseen data. Training in such a scenario can be accurately modeled as an *online convex optimization* (OCO) problem (Shalev-Shwartz 2011; Hazan 2016), where the optimizer suffers a loss $\mathcal{L}_t(\theta_t) = \mathcal{L}(y_t, f(X_t; \theta_t))$ and produces updated parameters θ_{t+1} at each iteration of the optimization process.

The task of finding an optimal learning rate schedule in this setting, can be formulated as

$$\begin{aligned} &\min_{\eta_0,\dots,\eta_T} \sum_{t=0}^{T} \mathcal{L}(y_t, f(X_t; \theta_t)) \\ &\text{s.t.} \qquad X_t, y_t \sim p_t \qquad \forall t \in 1, \dots, T. \end{aligned} \tag{7}$$

Compared to Problem (3), Problem (7) features some key differences. Due to Requirement , the goal is to minimize the total sum of losses incurred over all timesteps of the prequential evaluation process, instead of minimizing the only

the validation loss for the final parameters θ_T. This means that not only the loss achieved by the final parameters θ_T, but the loss suffered at every timestep of the stream contributes equally to the objective. Therefore, speed of convergence is of larger importance in the streaming setting, whereas the performance of the final parameters θ_T has relatively little impact. Since memory is limited (Requirement), it is also not possible to continue training on previously observed data as long as $\mathcal{L}$ decreases, which puts an even greater emphasis on quick adaptation. At the same time, a larger learning rate causing temporary loss increases, due to skipping over local minima can be suboptimal with respect to Problem 7 even if it eventually yields a lower loss.

In OCO literature it is well known that under the assumption of convexity of the loss function w.r.t. θ, the worst-case optimal fixed learning rate is

$$\eta^* = \frac{\theta^* - \theta_0}{\sqrt{\sum_{t=0}^{T} ||g_t||^2}}, \tag{8}$$

Another difference to conventional batch learning is that the distribution p_t of the data stream might, and in practice most likely will, be subjected to change in the form of *concept drift*[1] over time. Under such circumstances, the optimal parameter values θ^* move throughout the progression of the stream requiring the model parameters to adapt. To enhance the model's ability to do so, it appears intuitive, to increase the learning rate whenever distributional change occurs.

Based on this notion, Kuncheva and Plumpton (2008) introduced an adaptive schedule that uses the predictive losses as an indicator for concept drift. Their approach updates the learning rate using

$$\eta_{t+1} = \eta_t^{1+\bar{\mathcal{L}}_{t-M}-\bar{\mathcal{L}}_t}, \tag{9}$$

where $\bar{\mathcal{L}}_t$ is the running mean of M previous losses. By doing so, the authors aim to achieve higher stability, when data is stationary and losses decline and higher adaptability, when data is drifting and losses rise. While this approach seems intuitively sound, for an initial learning rate $\eta_0 \leq 1$ it bears a high risk of increasing up to a value of 1, since increases in loss caused by an excessive learning rate would lead to a feedback loop. Furthermore, loss plateaus that could be avoided by lowering η would instead cause η to remain stable, diminishing performance.

To offer the same potential benefits as Kuncheva and Plumpton (2008) approach while addressing its fundamental issues, we propose a simple adaptation to decaying learning rate schedules that resets η to its original value if a concept drift has been detected. An exponential schedule modified with our approach therefore yield learning rates

$$\eta_t = \eta_0 \cdot \gamma^{t-t_d}, \tag{10}$$

where t_d marks the timestep in which drift was last detected. As a result, feedback-loops are avoided assuming η_0 is small enough to not cause divergence and η_t can also decay in the presence of loss plateaus.

[1]We use concept drift as an umbrella term for any form of distributional shift.

For the purpose of drift detection we apply ADWIN (Bifet and Gavaldà 2007) to the prequential losses. To avoid mistakenly detecting drops in loss as concept drifts, we use a one-tailed ADWIN variant that tests only for increases.

Our approach is similar to some *forgetting mechanisms* (Gama et al. 2014) commonly employed in conventional non-deep online learning, which improve model plasticity by partly (Bifet and Gavaldà 2009) or resetting the current model's parameters to their initial values. However, we hypothesize that this approach is not well suited for deep learning purposes because, under the assumption of convexity, it requires that the newly initiated parameters be closer to the optimal parameters θ^* than the current parameters to be beneficial. For all but the most severe drifts, this seems highly unlikely. Nevertheless, we experimentally compare our approach with this mechanism in Section .

A limitation of our learning rate resetting technique can be seen in the fact, that it is insensitive to drifts that are not significant enough to be detected by ADWIN. To address this, we develop a *soft resetting* adaptation approach that only partly resets the learning rate based on an estimate of the drift probability $\hat{p}$ by using

$$\eta_{t+1} = (\eta_t + \alpha \cdot \hat{p}(\eta_0 - \eta_t)) \cdot \gamma, \tag{11}$$

where $\alpha \in (0, 1]$ is a hyperparameter. We obtain $\hat{p}$ by performing a Kolmogorov-Smirnoff test on two time shifted rolling windows of prediction losses as is also done by the *KSWIN* drift detector (Raab, Heusinger, and Schleif 2020). With this confidence estimate we aim to achieve smaller steps that depend on the severity of drift and therefore cause more granular adaptation.

Concept drift also complicates the tuning of η, since even if data is available beforehand drift would eventually cause the stream to diverge from the distribution of data used for tuning. This effect, combined with the previously described differences in the evaluation scheme can cause conventional learning rate tuning to produce unsuitable results for stream-based learning. We therefore propose a slightly different online learning specific tuning approach, that aims to approximately solve Problem 7.

To emulate the targeted data stream we continually draw samples with replacement from the tuning data in a bootstrapping procedure instead of training on all data for multiple epochs. By doing so we aim to increase data variability, and therefore the resemblance to an actual data stream with random distributional shifts. We then optimize η with respect to the mean prequential performance over the emulated stream instead of the performance on a validation set. For this purpose we use a basic grid-search as is customary in batch learning (Defazio and Mishchenko 2023). We provide a detailed experimental evaluation of our approach in Section .

Adaptive Optimizers

While determining the learning rate through a separate tuning phase with parameter searches like grid- or random-search is still the de facto standard in deep learning (Defazio and Mishchenko 2023), this approach causes significant computational overhead.

To reduce this overhead, several previous works have developed *adaptive optimizers*, which adjust the learning rate based on additional information about the loss landscape obtained from previous gradients at each optimization step, increasing the robustness with respect to the step size (Duchi, Hazan, and Singer 2011).

One of the earlier optimizers in this category is *AdaGrad* (Duchi, Hazan, and Singer 2011), which divides the learning rate by the square root of the uncentered total sum of squares over all previous gradients, for each model parameter resulting in a parameter specific learning rate. Unlike a single global value, parameter specific learning rates therefore not only influence the length, but also the direction of update steps, in case of AdaGrad by shifting updates in the direction of smaller gradients (Wu, Ward, and Bottou 2020).

Among several other approaches like AdaDelta (Zeiler 2012, see e.g.) and RMSProp (Tieleman and Hinton 2012), Kingma and Ba (2017) subsequently introduced Adam as an extension of AdaGrad, that additionally takes a momentum term of past gradients into account (Sutskever et al. 2013, see) to speed up the convergence for parameters with consistent gradients.

While adaptive approaches such as AdaGrad and Adam have been shown to reduce the dependence on the learning rate, they often times still require manual tuning (Wu, Ward, and Bottou 2020). A problem that parameter-free variants of SGD aim to solve by estimating the optimal step size online as training progresses, thus eliminating the learning rate altogether.

For instance, Schaul, Zhang, and LeCun (2013) proposed *vSGD*, which, like Adam, uses first and second order moments of the gradients as well as local curvature information to estimate η (Schaul, Zhang, and LeCun 2013). The authors obtain the latter by estimating positive diagonal entries of the Hessian with respect to the parameters through a backpropagation formula (Schaul, Zhang, and LeCun 2013). Even though Schaul, Zhang, and LeCun (2013) demonstrate *vSGD's* robustness to non-stationary data distributions, it has, to the best of our knowledge, not been widely adopted in the online learning space. Due to the lack of publicly available implementations of the non-trivial algorithm, we have not been able to evaluate vSGD at the time of writing.

not eval- vSGD e its old there is mplemen- n. Is that l?

Instead of using curvature information for adapting η, the *COCOB* algorithm proposed by Orabona and Tommasi (2017) models parameter optimization as a gambling problem, in which the goal is to maximize the rewards obtained from betting on each gradient. The model parameters are then computed based on the rewards accumulated over all previous timesteps (Orabona and Tommasi 2017).

-Hypergradient Descent (Baydin et al. 2018): optimizes the learning rate of stochastic optimizers like SGD using a meta-gradient descent procedure. -WNGrad (Wu, Ward, and Bottou 2020): adapts the dynamic update of AdaGrad to a single learning rate. -A -Mechanic (Cutkosky, Defazio, and Mehta 2023): can wrap around any first order algorithm, removing the need of tuning η. Uses a base online convex optimization algorithm as well as a meta OCO algorithm to optimize the learning rate with respect to the theoretical upper convergence bound of SGD -DoG (Ivgi, Hinder, and Carmon 2023): also optimizes theoretical upper bound by estimating $||\theta_0 - \theta^*||$ as $\max_{i<t} ||\theta_0 - \theta_i||$ -D-Adaptation (Defazio and Mishchenko 2023): can modify popular optimizers by estimating D with weighted dual averaging (Duchi, Agarwal, and Wainwright 2012)

Furthermore, several studies developed parameter-free optimizers for specific areas of application such as time series forecasting (Miyaguchi and Kajino 2019; Fekri et al. 2021; Zhang 2021), federated learning (Canonaco et al. 2021) and recommender systems (Ferreira Jose, Enembreck, and Paul Barddal 2020). Due to our focus for the present work being general data stream applications, we did not further investigate these techniques.

Despite the fact that parameter-free stochastic optimization techniques are inherently well-suited for the highly non-stationary streaming data (Schaul, Zhang, and LeCun 2013) and in some cases even developed based on online convex optimization, their application on this kind of data has rarely been investigated. This raises the question, whether they are suitable for stream-based learning (ii).

Optimizer	Runtime	Space	Param. specific	LR free
DAdapt	$\mathcal{O}(6D)$	$\mathcal{O}(2D)$	✗	✓
DoG	$\mathcal{O}(5D)$	$\mathcal{O}(1D)$	✗	✓
Mechanic	$\mathcal{O}(10D)$	$\mathcal{O}(1D)$	✗	✓
WNGrad	$\mathcal{O}(2D)$	$\mathcal{O}(0)$	✗	✓
SGDHD	$\mathcal{O}(2D)$	$\mathcal{O}(1D)$	✗	✓
COCOB	$\mathcal{O}(14D)$	$\mathcal{O}(4D)$	✓	✓
Adam	$\mathcal{O}(12D)$	$\mathcal{O}(2D)$	✓	✗
vSGD	$\mathcal{O}(21D)^2$	$\mathcal{O}(4D)$	✓	✓
AdaGrad	$\mathcal{O}(5D)$	$\mathcal{O}(1D)$	✓	✗

Table 1: Overview of additional time- and space-complexity of adaptive first-order optimizers compared to basic SGD. Values are given in big O notation with respect to the number of model parameters D. We do not list convergence guarantees because the guarantees given in the original papers of different optimizers are based on different assumptions and are rarely compatible with streaming applications.

Experiments

We ran prequential evaluations using basic SGD with variable batch sizes and learning rates for synthetic data streams with and without incremental concept drift, the results of which are displayed in **??**. For static data, the average prequential accuracy over the entire stream gradually improves when moving up from an inadequately low learning rate until a certain point where training begins to diverge and performance consequently crashes. Based on our results, there seems to be an inverse relationship between batch size and both the optimal learning rate and the optimal accuracy, with

[1]Complexity for feed-forward neural networks. Since *vSGD* requires additional backpropagation steps, its complexity is architecture dependent.

[2]We used the first 100k from a total of 581k examples only.

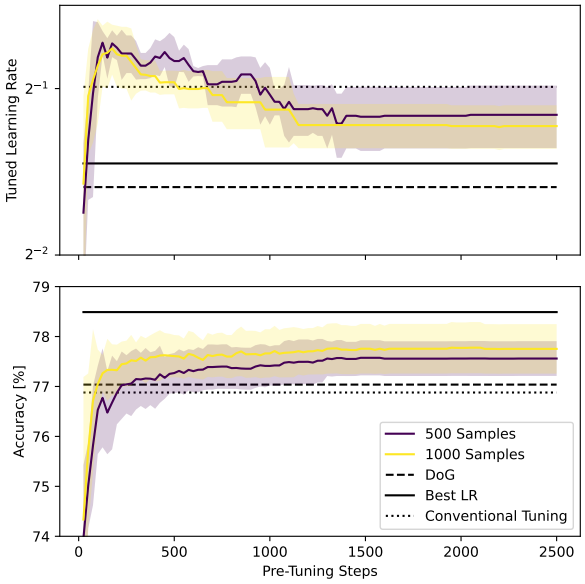

Figure 2: Pre-tuned LR (LR that maximizes accuracy on pre-tuning data) and resulting accuracy on data streams when using SGD and an exponential learning rate schedule with 500 or 1000 separate tuning samples. Results are averaged over all real-world datasets. The shaded area represents the 1σ-interval.

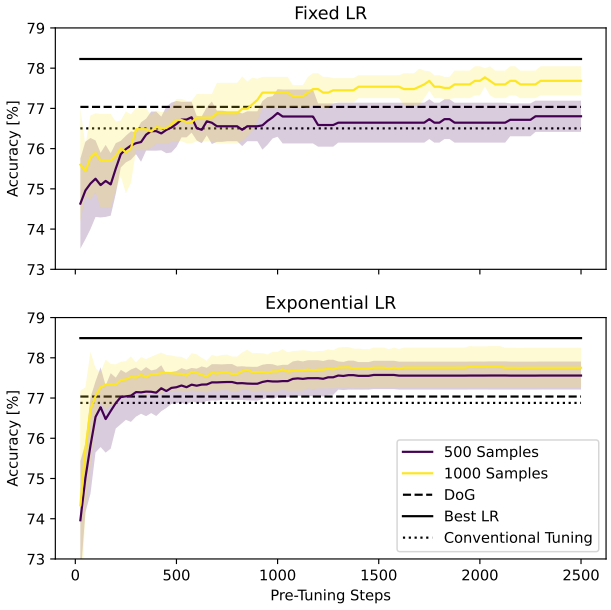

Figure 3: Accuracy achieved by pre-tuning on 500 or 1000 samples when using SGD with a fixed LR schedule (top) or an exponential schedule (bottom), averaged over all real-world datasets. The shaded area represents the 1σ-interval.

Type	Data Stream	Samples	Features	Classes
Synth.	RBF abrupt	20000	20	5
	RBF incremental	20000	20	5
Real	Insects abrupt	52848	33	6
	Insects incremental	57018	33	6
	Insects incr.-grad.	24150	33	6
	Covertype[3]	100000	54	7
	Electricity	45312	8	2

Table 2: Datasets used for experimental evaluations.

larger batch sizes seemingly increasing the risk of divergence.

(12)

This effect is much stronger in the presence of concept drift as the results for RBF Incremental show.

¡-could be explained by the fact that the presence of concept drift exacerbates the gradient stochasticity caused by the delay between observation and learning of samples.

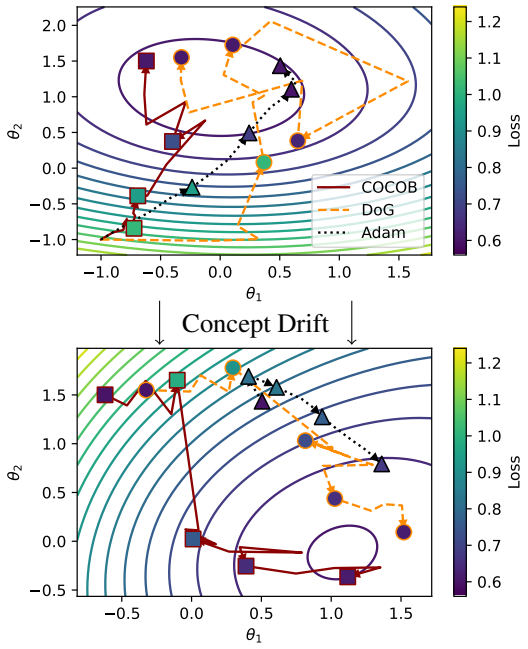

Figure 4: Parameter trajectory of COCOB (Orabona and Tommasi 2017), DoG (Ivgi, Hinder, and Carmon 2023) and Adam (Kingma and Ba 2017) on synthetic data stream with abrupt concept drift. Marker colors depict the expected prequential loss over the last 16 data instances.

Conclusion

References

Baydin, A. G.; Cornish, R.; Rubio, D. M.; Schmidt, M.; and Wood, F. 2018. Online Learning Rate Adaptation with Hypergradient Descent. In *ICLR Proceedings*.

Bengio, Y. 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures. arxiv:1206.5533.

Bifet, A.; and Gavaldà, R. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-630-6 978-1-61197-277-1.

Bifet, A.; and Gavaldà, R. 2009. Adaptive Learning from Evolving Data Streams. In Adams, N. M.; Robardet, C.; Siebes, A.; and Boulicaut, J.-F., eds., *Advances in Intelligent Data Analysis VIII*, Lecture Notes in Computer Science, 249–260. Berlin, Heidelberg: Springer. ISBN 978-3-642-03915-7.

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2010. MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11.

Canonaco, G.; Bergamasco, A.; Mongelluzzo, A.; and Roveri, M. 2021. Adaptive Federated Learning in Presence of Concept Drift. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. arxiv:1611.01838.

Cutkosky, A.; Defazio, A.; and Mehta, H. 2023. Mechanic: A Learning Rate Tuner. arxiv:2306.00144.

Defazio, A.; and Mishchenko, K. 2023. Learning-Rate-Free Learning by D-Adaptation. arxiv:2301.07733.

Duchi, J. C.; Agarwal, A.; and Wainwright, M. J. 2012. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. *IEEE Transactions on Automatic Control*, 57(3): 592–606.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159.

Fekri, M. N.; Patel, H.; Grolinger, K.; and Sharma, V. 2021. Deep Learning for Load Forecasting with Smart Meter Data: Online Adaptive Recurrent Neural Network. *Applied Energy*, 282: 116177.

Ferreira Jose, E.; Enembreck, F.; and Paul Barddal, J. 2020. ADADRIFT: An Adaptive Learning Technique for Long-history Stream-based Recommender Systems. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2593–2600. Toronto, ON, Canada: IEEE. ISBN 978-1-72818-526-2.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A Survey on Concept Drift Adaptation. *ACM Computing Surveys*, 46(4): 1–37.

Hazan, E. 2016. Introduction to Online Convex Optimization. *Foundations and Trends® in Optimization*, 2(3-4): 157–325.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat Minima. *Neural Computation*, 9(1): 1–42.

Ivgi, M.; Hinder, O.; and Carmon, Y. 2023. DoG Is SGD's Best Friend: A Parameter-Free Dynamic Step Size Schedule. arxiv:2302.12022.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arxiv:1412.6980.

Kuncheva, L. I.; and Plumpton, C. O. 2008. Adaptive Learning Rate for Online Linear Discriminant Classifiers. In Da Vitoria Lobo, N.; Kasparis, T.; Roli, F.; Kwok, J. T.; Georgiopoulos, M.; Anagnostopoulos, G. C.; and Loog, M., eds., *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342, 510–519. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-89688-3 978-3-540-89689-0.

Miyaguchi, K.; and Kajino, H. 2019. Cogra: Concept-Drift-Aware Stochastic Gradient Descent for Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 4594–4601.

Orabona, F.; and Tommasi, T. 2017. Training Deep Networks without Learning Rates Through Coin Betting. In *NIPS*.

Raab, C.; Heusinger, M.; and Schleif, F.-M. 2020. Reactive Soft Prototype Computing for Concept Drift Streams. *Neurocomputing*, 416: 340–351.

Schaul, T.; Zhang, S.; and LeCun, Y. 2013. No More Pesky Learning Rates. arxiv:1206.1106.

Shalev-Shwartz, S. 2011. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, 4(2): 107–194.

Smith, L. N. 2017. Cyclical Learning Rates for Training Neural Networks. arxiv:1506.01186.

Smith, L. N.; and Topin, N. 2018. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. arxiv:1708.07120.

Smith, S. L.; Kindermans, P.-J.; Ying, C.; and Le, Q. V. 2018. Don't Decay the Learning Rate, Increase the Batch Size. arxiv:1711.00489.

Smith, S. L.; and Le, Q. V. 2018. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. arxiv:1710.06451.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on Machine Learning*, 1139–1147. PMLR.

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. In *COURSERA: Neural Networks for Machine Learning*. Coursera.

Wu, X.; Ward, R.; and Bottou, L. 2020. WNGrad: Learn the Learning Rate in Gradient Descent. arxiv:1803.02865.

Wu, Y.; Liu, L.; Bae, J.; Chow, K.-H.; Iyengar, A.; Pu, C.; Wei, W.; Yu, L.; and Zhang, Q. 2019. Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks. arxiv:1908.06477.

Zeiler, M. D. 2012. ADADELTA: An Adaptive Learning Rate Method. arxiv:1212.5701.

Zhang, W. 2021. POLA: Online Time Series Prediction by Adaptive Learning Rates. arxiv:2102.08907.