OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Connections between Schedule-Free optimizers and Accelerated SGD variants

Depen Morwani DMORWANI@G.HARVARD.EDU
Nikhil Vyas NIKHIL@G.HARVARD.EDU
Hanlin Zhang HANLINZHANG@G.HARVARD.EDU
Sham Kakade SHAM@SEAS.HARVARD.EDU
Harvard University

Abstract

Recent advancements in deep learning optimization have introduced new algorithms, such as Schedule-Free optimizers, AdEMAMix and Lion which modify traditional momentum mechanisms. In a separate line of work, theoretical acceleration of stochastic gradient descent (SGD) in noise-dominated regime has been achieved by decoupling the momentum coefficient from the current gradient's weight. In this paper, we establish direct connections between these two lines of work. Specifically, we demonstrate that Schedule-Free SGD is equivalent to performing accelerated SGD followed by weight averaging. Moreover, optimizers like Lion, Schedule-Free AdamW, and AdEMAMix can be interpreted as combining preconditioning with accelerated SGD methods. While some of these connections have been partially recognized in prior literature, the full extent of the relationship between these new optimizers and accelerated SGD variants has not been previously explored.

We substantiate our theoretical findings with preliminary experiments on a 150-million-parameter decoder-only transformer model trained on 15 billion tokens using a small batch size of 32k tokens, ensuring a noise-dominated regime. Our results indicate that (1) methods based on accelerated SGD exhibit slightly improved performance in such settings and (2) Schedule-Free AdamW performs comparably to an accelerated SGD-based AdamW optimizer.

1. Introduction

Recently, many new optimizers have been proposed for deep learning such as Lion [1], ScheduleFreeSGD/AdamW [3], and AdEMAMix [10]. While these optimizers are proposed with different motivations, all of them can be understood as changing the form of momentum used in the optimization.

A different line of theoretical work has been focused on accelerating gradient descent in the presence of noise. Although the standard or Nesterov momentum is sufficient to accelerate deterministic gradient descent, it is not sufficient to accelerate SGD [5, 7]. This has led to proposals about modifying the momentum schemes to achieve acceleration in the presence of noise [4, 5, 7, 12]. All of these accelerated SGD methods proposed in the literature are equivalent to decoupling the momentum coefficient from the weight on the current gradient in the optimizer update.

Our main contribution is to establish direct connection between the ideas developed by these two lines of work. In fact, we show that ScheduleFree SGD is exactly equivalent to doing accelarated SGD followed by weight averaging. Approaches such as Lion, ScheduleFree AdamW and AdEMAMix are equivalent to combining preconditioning with the accelerated SGD approaches. We note that some of these connections have been noticed in the literature [2], but to the best of

our knowledge, the connection of these new optimizers to accelerated SGD has not been observed before.

We follow up with experiments on a 150m decoder-only transformer model trained on 15B tokens and a small batch size of 32k tokens to ensure that we are in a noise-dominated regime. As expected from our theoretical results, the performance of schedule-free AdamW is very close to that of accelerated SGD based AdamW (Algorithm 2). We also show that accelerated methods enjoy slightly improved performance at such small batch sizes.

Our main contributions are stated below:

1. We show precise connections between recently proposed optimizers such as Schedule-Free SGD, AdEMAMix and accelerated SGD.
2. We provide experimental results on a 150m decoder only transformer with small batch size, comparing AdamW, Schedule-Free AdamW and accelerated AdamW.

2. Related Work

We will cover the literature on accelerated SGD variants and optimizers which are directly relevant for our work.

Jain et al. [5] introduced an accelerated SGD variant, that provided improved convergence for least-squares problem. Kidambi et al. [6] simplified the update of this variant and also provably showed that momentum does not provide acceleration in this case. Gupta et al. [4], Liu and Belkin [7], Vaswani et al. [12] extend these results to the general cases of convex and strongly convex functions under different assumptions.

Various optimizers have been proposed over years which resemble the accelerated SGD variants described above. Lucas et al. [8] use a weighted sum of multiple momentum terms with different coefficients for the final update. Ma and Yarats [9] propose an optimizer directly inspired by the theoretical work of Jain et al. [5]. Chen et al. [1] is a recent optimizer discovered using genetic search algorithm, which, akin to the previous accelerated SGD variants, place different mass on the gradient in the update as compared to the momentum coefficient. Pagliardini et al. [10] also mixes two different momentum scales in the final update.

3. Background

3.1. Momentum

Momentum is a well-known technique for accelerating convergence of gradient descent in deterministic settings. Momentum update with a momentum coefficient β for weights w_t is given by

$$m_t = \beta m_{t-1} + \nabla f(w_t); w_t = w_{t-1} - \eta m_t$$

3.2. Weight Averaging

Weight averaging is a known-technique generally used in the stochastic optimization literature, for reducing the noise present in the iterates. Instead of returning the final iterate w_T, a weighted average $\bar{w}_T$ of the iterates is returned with weights denoted by γ_t.

$$\bar{w}_T = (1 - \gamma_T)\bar{w}_{T-1} + \gamma_T w_T$$

All weight averaging used in this paper will use coefficients γ_t of the form $\gamma_t \approx 1 - 1/(\delta t)$ for some constant $0 \leq \delta \leq 1$.

3.3. Accelerated SGD

In this section we provide a generic structure into which many accelerated SGD methods fall into:

$$m_t = \beta_{\mathrm{a},t} m_{t-1} + g_t \qquad w_{t+1} = w_t - \eta_{\mathrm{a},t} m_t - \alpha_{\mathrm{a},t} g_t \tag{1}$$

where $\beta_{\mathrm{a},t}, \alpha_{\mathrm{a},t}, \eta_{\mathrm{a},t}$ are (possibly time dependent) scalars and g_t is the stochastic gradient evaluated on w_t. We will use the subscript 'a' to denote the coefficients that fit this exact accelerated SGD form. We begin by noting that it we set $\alpha_{\mathrm{a},t} = 0$ we recover standard SGD with momentum. Further, it is easy to see and has also been noticed in previous work that for many of the algorithms proposed in the accelerated SGD literature such as those in the works of Gupta et al. [4], Jain et al. [5], Liu and Belkin [7], Vaswani et al. [12] fall directly into this format.

3.4. Schedule-free SGD

Schedule-free SGD [3] is a recently proposed constant learning rate optimizer aimed at removing the need for scheduling. Following the notations as used in Defazio et al. [3], the updates are given by

$$\begin{aligned} y_t &= (1 - \beta) z_t + \beta x_t \\ z_{t+1} &= z_t - \gamma g(y_t) \\ x_{t+1} &= (1 - c_{t+1}) x_t + c_{t+1} z_{t+1} \end{aligned}$$

Here, y_t denotes the current weights of the model (note that the gradient is evaluated on y_t), while x_t are the weights that will be used for evaluation.

We will begin with writing the update in terms of y_t and m_t, which we define as

$$m_{t+1} = \frac{x_t - z_{t+1}}{\gamma} \tag{2}$$

We can further simplify m_{t+1} as following

$$\begin{aligned} m_t &= \frac{x_t - z_{t+1}}{\gamma} && (3) \\ &= \frac{x_t + \gamma g_t - z_t}{\gamma} && (4) \\ &= \frac{x_t - z_t + \gamma g_t}{\gamma} && (5) \\ &= \frac{(1 - c_t)(x_{t-1} - z_t) + \gamma g_t}{\gamma} && (6) \\ &= (1 - c_t) m_{t-1} + g_t && (7) \end{aligned}$$

Hence m_t follows the momentum update in Equation (1) with $\beta_{\text{a},t} = 1 - c_t$. Now given m_t let us consider the update of y_t.

$$\begin{aligned} y_{t+1} &= (1-\beta)z_{t+1} + \beta x_{t+1} && (8)\\ &= (1-\beta)\left(z_t - \gamma g_t\right) + \beta\left((1-c_{t+1})x_t + c_{t+1}z_{t+1}\right) && (9)\\ &= (1-\beta)z_t + \beta x_t - (1-\beta)\gamma g_t + \beta c_{t+1}(z_{t+1} - x_t) && (10)\\ &= y_t - \gamma[\beta c_{t+1} m_t + (1-\beta)g_t] && (11) \end{aligned}$$

Hence y_t follows weight update in Equation (1) with $\eta_{\text{a},t} = \gamma\beta c_{t+1}$ and $\alpha_{\text{a},t} = \gamma(1-\beta)$ and $w_t = y_t$. Hence at this point ScheduleFree SGD exactly follows accelerated SGD. A complication arises when we consider the fact that for evaluation ScheduleFree SGD uses x_t. Let us now try to understand the dynamics of x_t:

$$\begin{aligned} x_{t+1} &= (1 - ct + 1)x_t + c_{t+1}z_{t+1}\\ x_{t+1} &= (1-c_{t+1})x_t + c_2\left(\frac{y_{t+1} - \beta x_{t+1}}{1-\beta}\right)\\ x_{t+1}\left(1 - \beta + c_{t+1}\beta\right) &= (1-c_{t+1})(1-\beta)x_t + c_{t+1}y_{t+1}\\ x_{t+1} &= \frac{(1-c_{t+1})(1-\beta)x_t + c_{t+1}y_{t+1}}{(1-c_{t+1})(1-\beta) + c_{t+1}} \end{aligned}$$

hence x_t is just a weight averaged version of y_t.

After the last equation above we can keep recursively expanding x_t to get that x_t is an exponential average of y_t for a constant c_t. So overall ScheduleFree SGD can be understood as a accelarated SGD followed by weight averaging.

This shows that the benefits of ScheduleFree SGD are directly attributable to its two components:

1. Better performance than standard SGD with momentum can be attributed to accelerated SGD.
2. Using a constant learning rate without any scheduling can be attributed to weight averaging (to be precise, tailed weight averaging, look Section 3.4.2)

We note that there are two benefits that are unique to the ScheduleFree SGD/Adam: 1. It does not require extra space to do weight averaging and 2. We do not have an extra hyperparameter: weight averaging coefficient. Both of these benefit occur since it ties the coefficients for momentum and weight averaging.

3.4.1. $\beta = 0.0$

As noted in [3], at $\beta = 1$, schedule free SGD reduces to normal momentum but with momentum $\beta_{\text{a},t}$ scaling up as $1 - 1/t$. Since $c_t = 1/t$ it does weight averaging from the start.

3.4.2. $\beta = 0.9$

The default value for β is ScheduleFree SGD is .9. For $\beta = 0.9$:

- As c_t scales down as $1/t$, therefore momentum keeps scaling up as $1 - 1/t$.
- The ratio of the weight on the current gradient to momentum is fixed to $(1 - \beta)/(\beta c_{t+1}) \approx 0.11$ (since $c_{t+1} \approx 1$ is just a bias correction for momentum).
- Weight averaging is done approximately over the last $100/10 = 10\%$ of the iterates as the weight on y_{t+1} in weight averaging is $\gamma_{2,t+1}/(1 - \beta + \gamma_{2,t+1}\beta) \approx 10/t$.

3.5. Lion

We note that Lion is exactly equivalent to accelerated SGD followed by the sign operation.

3.6. AdEMAMix

A recent optimizer AdEMAMix [10] also seems to have an updated step similar to that of accelerated SGD based AdamW with one difference instead of using a linear combination of the current gradient and the momentum as accelerated SGD does it instead maintains two momentums with different coefficients and takes there linear combination. The algorithm for the optimizer is stated in Section 3.6. In Pagliardini et al. [10], the authors observed comparable performance of $\beta_1 = 0.0$ and $\beta_1 = 0.9$ at small batch sizes. At $\beta_1 = 0.0$, AdEMAMix precisely falls within the accelerated SGD framework, combined with preconditioning.

Algorithm 1 Single step of AdEMAMix optimizer. Differences with AdamW are in blue.

Input: Data distribution $\mathcal{D}$. Initial model parameters $\theta^{(0)}$. Number of iterations T. Learning rate η. ϵ a small constant. AdamW parameters: β_1, β_2. AdEMAMix parameters β_3, α. Warmup parameter T_{α,β_3}, note that we usually set it to T. β_{start} is usually set to β_1.
Optional: use schedulers $\eta^{(t)}$, $\beta_3^{(t)} \leftarrow f_{\beta_3}(t, \beta_3, \beta_{\text{start}}, T_{\alpha,\beta_3})$ and $\alpha^{(t)} \leftarrow f_\alpha(t, \alpha, T_{\alpha,\beta_3})$
Sample batch: $x \sim \mathcal{D}$
Compute gradient: $g^{(t)} \leftarrow \nabla_\theta \mathcal{L}_{\theta^{(t-1)}}(x)$
Update the fast EMA m_1: $m_1^{(t)} \leftarrow \beta_1 m_1^{(t-1)} + (1 - \beta_1) g^{(t)}$
Update the slow EMA m_2: $m_2^{(t)} \leftarrow \beta_3^{(t)} m_2^{(t-1)} + (1 - \beta_3^{(t)}) g^{(t)}$
Update the second moment estimate: $\nu^{(t)} \leftarrow \beta_2 \nu^{(t-1)} + (1 - \beta_2) \left(g^{(t)}\right)^2$
Update parameters: $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta^{(t)} \left(\frac{\hat{m}_1^{(t)} + \alpha^{(t)} m_2^{(t)}}{\sqrt{\hat{\nu}^{(t)}} + \epsilon} \right)$

4. Experiments

In this section, we provide experiments on a 150m decoder-only transformer model for language modeling task on C4 dataset. We train it with a sequence length of 1024 and batch size of 32, for 15B tokens ($\approx 5\times$ Chinchilla) to ensure we are in a noise dominated regime. We compare the following algorithms:

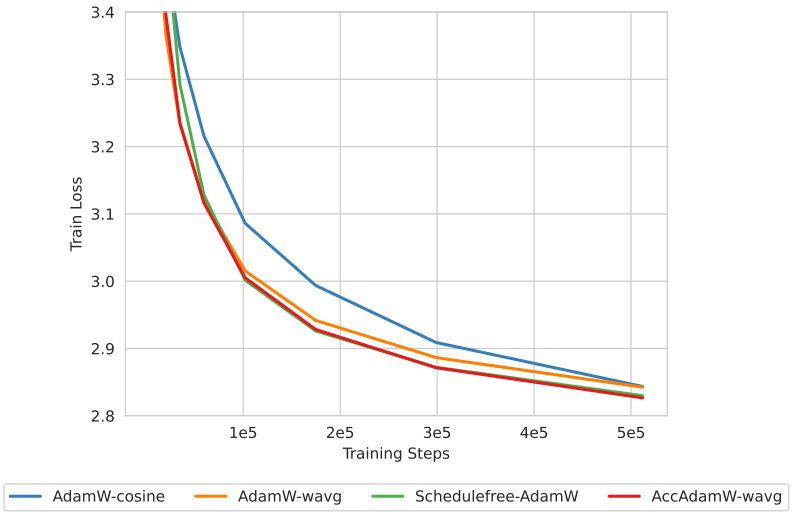

Figure 1: Comparison of the best runs of AdamW with cosine decay, AdamW with weight averaging, Schedule free AdamW and accelerated AdamW with weight averaging for language modeling task on a decode-only 150m transformer model.

Algorithm 2 Single step of accelerated SGD based Adam with weight averaging. For simplicity we ignore the initialization, other boundary effects such as bias correction, and weight decay. Hyperparameters: Learning rate η, betas $= (\beta_1, \beta_2, \beta_3)$, weight averaging coefficient δ, and epsilon ϵ.

Sample batch B_t.
$g \leftarrow -\nabla_w \phi_{B_t}(w_t)$
$v \leftarrow \beta_2 v + (1-\beta_2)(g \odot g)$
$N \leftarrow \frac{\beta_3 m + (1-\beta_3) g}{\sqrt{v}+\epsilon}$
$w \leftarrow w - \eta N$
$m \leftarrow \beta_1 m + (1-\beta_1) g$
$c = \max(1 - 1/t, 1 - 1/(\delta t))$
$w_{\text{avg}} \leftarrow c w_{\text{avg}} + (1-c) w$

1. Standard AdamW with cosine decay
2. Standard AdamW with weight averaging
3. Schedule-free AdamW
4. Accelerated AdamW with weight averaging (Algorithm 2)

Hyperparameter sweeps for the algorithms are provided in Appendix A.

As shown in Figure 1, schedule-free AdamW and accelerated AdamW with tailed weight averaging perform comparably supporting our claims. We can also see that they outperform both AdamW with cosine decay and AdamW with tailed weight averaging.

5. Conclusion

In this work, we showed the equivalence between the recently proposed schedule-free SGD optimizer and accelerated SGD with weight averaging. We also empirically demonstrated that accelerated SGD demonstrates improved performance as compared to SGD with momentum for training a 150m decoder-only transformer model on language modeling task.

References

[1] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

[2] Aaron Defazio. Momentum via primal averaging: Theoretical insights and learning rate schedules for non-convex optimization, 2021. URL https://arxiv.org/abs/2010.00406.

[3] Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *CoRR*, abs/2405.15682, 2024. doi: 10.48550/ARXIV.2405.15682. URL https://doi.org/10.48550/arXiv.2405.15682.

[4] Kanan Gupta, Jonathan Siegel, and Stephan Wojtowytsch. Achieving acceleration despite very noisy gradients, 2023. URL https://arxiv.org/abs/2302.05515.

[5] Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Accelerating stochastic gradient descent for least squares regression. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 545–604. PMLR, 06–09 Jul 2018. URL https://proceedings.mlr.press/v75/jain18a.html.

[6] Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham M. Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJTutzbA-.

[7] Chaoyue Liu and Mikhail Belkin. Accelerating sgd with momentum for over-parameterized learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1gixp4FPH.

[8] James Lucas, Shengyang Sun, Richard Zemel, and Roger Grosse. Aggregated momentum: Stability through passive damping. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Syxt5oC5YQ.

[9] Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and adam for deep learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1fUpoR5FQ.

[10] Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. 2024. URL https://arxiv.org/abs/2409.03137.

[11] Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *CoRR*, abs/2406.19146, 2024. doi: 10.48550/ARXIV.2406.19146. URL https://doi.org/10.48550/arXiv.2406.19146.

[12] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1195–1204. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/vaswani19a.html.

Appendix A. Hyperparameters

1. AdamW with cosine decay - 51.2k warmup - learning rate in [3.16e-4, 1e-3, 3.16e-3], β_1 in [0.9, 0.95], β_2 in [0.99, 0.999, 0.99968, 0.9999]. The optimal values of β_1 and β_2 were .9 and .999 respectively matching the default values. We note that for larger batch sizes it is common to use $\beta_2 = .95$, the benfit of higher β_2 at smaller batch sizes has also been observed by Porian et al. [11].

2. AdamW with cosine decay - 10k warmup - learning rate in [3.16e-4, 1e-3, 3.16e-3], $\beta_1 = 0.9$, $\beta_2 = 0.999$ i.e. we fix β_1, β_2 to be the optimal values from the previous sweep. This performed worse that warmup of 51.2k steps.

3. AdamW constant fraction weight averaging: - learning rate in [3.16e-4, 1e-3, 3.16e-3], $\beta_1 = 0.9$, β_2 in [0.99, 0.997, 0.999, 0.9997], δ in [0.05, 0.1, 0.2].

4. AdamW with cosine decay and weight averaging - learning rate in [3.16e-4, 1e-3, 3.16e-3], $\beta_1 = 0.9$, $\beta_2 = 0.999$, δ in [0.025, 0.05, 0.1].

5. Accelerated SGD based AdamW with cosine decay - learning rate in [3.16e-4, 1e-3, 3.16e-3], β_1 in [0.999, 0.99968, 0.9999], β_2 in [0.99, 0.9968, 0.999], $\beta_3 = 0.9$

6. Accelerated SGD based AdamW with constant learning rate and weight averaging - learning rate in [3.16e-4, 1e-3, 3.16e-3], β_1 in [0.99684, 0.999], β_2 in [0.999], $\beta_3 = 0.9$, δ in [0.05, 0.1]

7. Accelerated SGD based AdamW with cosine decay and weight average - learning rate in [3.16e-4, 1e-3, 3.16e-3], β_1 in [0.99684, 0.999], $\beta_2 = 0.999$, δ in [0.05, 0.1], $\beta_3 = 0.9$

8. Schedulefree AdamW with constant learning rate - learning rate in [3.16e-4, 1e-3, 3.16e-3, 1e-2], β_1 in [0.8, 0.9, 0.95], $\beta_2 = 0.999$

9. Schedulefree AdamW with cosine decay - [3.16e-4, 1e-3, 3.16e-3, 1e-2], β_1 in [0.8, 0.9, 0.95], $\beta_2 = 0.999$

Appendix B. Equivalence of previous acceleration methods

The general accelerated SGD form is provided in Equation (1). In this section, we will show that all the methods in the works Gupta et al. [4], Jain et al. [5], Liu and Belkin [7], Vaswani et al. [12] fall within this form.

B.1. AGNES

The update for Gupta et al. [4] is given below:

$$x'_n = x_n + \alpha v_n \qquad x_{n+1} = x'_n - \eta g'_n \qquad v_{n+1} = \rho_n(v_n - g'_n)$$

where g'_n is stochastic gradient evaluated on x'_n and the final function is evaluated on x_n. The above equations can be rewritten as

$$x'_{n+1} = x'_n - \eta g'_n + \alpha v_{n+1} \qquad -\frac{v_{n+1}}{\rho_n} = \rho_{n-1}\left(-\frac{v_n}{\rho_{n-1}}\right) + g'_n$$

Thus x'_{n+1} follows update equation of the form of Equation (1).

B.2. ASGD

The update for Jain et al. [5] is given by:

$$y_{j-1} = \alpha x_{j-1} + (1-\alpha)v_{j-1} \qquad x_j = y_{j-1} - \delta g_{j-1} \qquad z_{j-1} = \beta y_{j-1} + (1-\beta)v_{j-1} \qquad v_j = z_{j-1} - \gamma g_{j-1}$$

where g_{j-1} represents the stochastic gradient evaluated on y_{j-1} and the function is evaluated on the tail averaged x.

The update equations above can be rewritten as:

$$y_j = y_{j-1} - \alpha\delta g_{j-1} - (1-\alpha)[y_{j-1} - v_j] \qquad \frac{y_{j-1} - v_j}{\gamma - (1-\beta)\alpha\delta} = (1-\beta)\alpha\frac{y_{j-2} - v_{j-1}}{\gamma - (1-\beta)\alpha\delta} + g_{j-1}$$

The update equations above follow the form of Equation (1).

B.3. MaSS

The update for Liu and Belkin [7] is given by:

$$w_{t+1} = u_t - \eta_1 g_t \qquad u_{t+1} = (1+\gamma)w_{t+1} - \gamma w_t + \eta_2 g_t$$

where g_t is the stochastic gradient evaluated on u_t and the function is evaluated on w_t. These equations can be rewritten as:

$$u_{t+1} = u_t - \gamma(w_t - u_t) - [\eta_1(1+\gamma) - \eta_2]g_t \qquad \frac{w_t - u_t}{\eta_1\gamma - \eta_2} = \gamma\frac{w_{t-1} - u_{t-1}}{\eta_1\gamma - \eta_2} + g_t$$

The update equations above follow the form of Equation (1).

B.4. SGD with Nesterov Acceleration

The update for Vaswani et al. [12] is given by:

$$w_{k+1} = \zeta_k - \eta g_k \qquad \zeta_k = \alpha_k v_k + (1-\alpha_k)w_k \qquad v_{k+1} = \beta_k v_k + (1-\beta_k)\zeta_k - \gamma_k\eta g_k$$

These equations can be rewritten as:

$$\zeta_{k+1} = \zeta_k - \eta g_k + \alpha_{k+1}[v_{k+1} - w_{k+1}]; \qquad v_{k+1} - w_{k+1} = \beta_k(1-\alpha_k)[v_k - w_k] - \eta(\gamma_k - 1)g_k$$