DEMON: IMPROVED NEURAL NETWORK TRAINING WITH MOMENTUM DECAY

John Chen[⋆]*, Cameron Wolfe*[⋆]*, Zhao Li*[†]*, Anastasios Kyrillidis*[⋆]

[⋆]Rice University, Houston, Texas
[†]UTHealth, Houston, Texas

ABSTRACT

Momentum is a widely used technique for gradient-based optimizers in deep learning. Here, we propose a decaying momentum (DEMON) hyperparameter rule. We conduct large-scale empirical analysis of momentum decay methods for modern neural network optimization and compare to the most popular learning rate decay schedules. Across 28 relevant combinations of models, epochs, datasets, and optimizers, DEMON achieves Top-1 and Top-3 performance in 39% and 85% of cases, respectively, almost doubling the second-placed cosine learning rate schedule at 17% and 60%, respectively. DEMON consistently outperforms other widely-used schedulers including, but not limited to, the learning rate step schedule, linear schedule, OneCycle schedule, and exponential schedule. Compared with the widely-used learning rate step schedule, DEMON is less sensitive to parameter tuning, which is critical to training neural networks in practice. Results are demonstrated across a variety of settings and architectures, including image classification models, generative models, and language models. DEMON is easy to implement, requires no additional tuning, and incurs almost no extra computational overhead compared to the vanilla counterparts. Code is readily available.

***Index Terms*—** Momentum, Hyperparameter tuning

1. INTRODUCTION

Motivation. Training Deep Neural Networks (DNNs) is expensive (e.g., recent language models cost several million USD to train [1, 2]). For practitioners, even moderate-scale tasks can be prohibitive in time and cost due to hyperparameter tuning, which may lead to multiple iterations of model retraining.

To ease DNN training cost, adaptive gradient-based methods [3, 4, 5, 6, 7] were devised. Currently, SGD with momentum (SGDM) and Adam [6] remain widely-used for training DNNs. Many benchmarks in Computer Vision are achieved using SGDM with a curated learning rate step schedule [8, 9, 10, 11, 12], while Adam variants are popular for training langauge models [13, 1].

Hyperparameters must be set correctly for optimizers to perform well. For example, the value and decay schedule for the learning rate must be carefully selected, where no one setting is known to be superior across domains. Due to large, hyperparameter-based fluctuations in performance, significant opportunity exists in fostering robustness to hyperparameters and making performance more consistent.

Momentum tuning. We propose simple techniques for the momentum parameter that improve empirically model performance and hyperparameter robustness. Momentum, originally designed to accelerate training in directions of low curvature without causing instability in direction of high curvature, is often set as $\beta = 0.9$ (e.g., [14, 6, 15]). *There is no indication that this choice is universally well-behaved [16, 17, 18, 19].*

Table 1: Different schedules ranked according to % of Top-1 (Top-3) finishes in 28 experiments. Top-1 (Top-3) refers to the best (or within the three best) performance for a particular `model-dataset-base optimizer-epoch` setting (e.g., `RN20-CIFAR10-SGDM-300epochs`).

Method	Top-1 Performance	Top-3 Performance
OneCycle Momentum	0%	0%
Linear Momentum	0%	0%
Exp Momentum decay	0%	0%
Cosine Momentum	0%	3.57%
LR Exp decay	7.14%	10.71%
LR Decay on Plateau	7.14%	21.43%
OneCycle	7.14%	25.00%
LR Linear Schedule	10.71%	39.29%
LR Step Schedule	10.71%	53.57%
LR Cosine Schedule	17.86%	60.71%
DEMON	**39.29%**	**85.71%**

This paper. Our contributions can be summarized as follows:

- We propose a new momentum decay rule, dubbed as DEMON, motivated by decaying the total contribution of a gradient to all future updates. We test DEMON and other learning rate/momentum schedules across eight datasets and seven model architectures.
- Across 28 settings, DEMON achieves the highest ratio of Top-1 and Top-3 finishes (i.e., 39% and 85%; see Table 1), nearly doubling those of the second-placed cosine learning rate decay schedule. DEMON outperforms other popular learning rate schedules and all other momentum schedules that were considered; see Table 1.
- In comparison to learning rate step schedules, DEMON is observed to improve hyperparameter robustness.

2. PRELIMINARIES

SGDM. Let $\theta_t \in \mathbb{R}^p$ represent network parameters at iteration t, $\eta \in \mathbb{R}$ the learning rate, and g_t the stochastic gradient w.r.t. θ_t for empirical loss $\mathcal{L}(\cdot)$. SGDM with momentum $\beta \in \mathbb{R}$ is shown below, where $v_t \in \mathbb{R}^p$ accumulates momentum and $\beta = 0$ recovers SGD:

$$\theta_{t+1} = \theta_t + \eta v_t, \quad v_t = \beta v_{t-1} - g_t.$$

Adaptive gradient descent methods. These algorithms utilize current and past gradients to design preconditioning matrices that better approximate the local curvature of $\mathcal{L}(\cdot)$ [3, 5]. Adam [6] uses a decaying average of past (i.e., $\mathcal{E}^g_{t+1} = \beta_1 \cdot \mathcal{E}^g_t + (1-\beta_1)\cdot g_t$) and squared (i.e., $\mathcal{E}^{g\circ g}_{t+1} = \beta_2 \cdot \mathcal{E}^{g\circ g}_t + (1-\beta_2)\cdot (g_t \circ g_t)$) gradients, yielding the recursion below:[1]

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\mathcal{E}^{g\circ g}_{t+1,i}+\varepsilon}} \cdot \mathcal{E}^g_{t+1,i}, \quad \forall t,$$

[1]We eliminate bias correction for simplicity [6].

Algorithm 1 DEMON in SGDM

Parameters: T, step size η, initial mom. β_{init}. $v_0 = \theta_0 = 0$ or random.
for $t = 0, \dots, T$ **do**
$\beta_t = \beta_{\text{init}} \cdot \frac{\left(1-\frac{t}{T}\right)}{(1-\beta_{\text{init}})+\beta_{\text{init}}\left(1-\frac{t}{T}\right)}$
$\theta_{t+1} = \theta_t - \eta g_t + \beta_t v_t$
$v_{t+1} = \beta_t v_t - \eta g_t$
end for

Algorithm 2 DEMON in Adam

Parameters: T, η, initial mom. β_{init}, β_2, $\varepsilon = 10^{-8}$. $v_0 = \theta_0 = 0$ or random.
for $t = 0, \dots, T$ **do**
$\beta_t = \beta_{\text{init}} \cdot \frac{\left(1-\frac{t}{T}\right)}{(1-\beta_{\text{init}})+\beta_{\text{init}}\left(1-\frac{t}{T}\right)}$
$\mathcal{E}^{g\circ g}_{t+1} = \beta_2 \cdot \mathcal{E}^{g\circ g}_t + (1-\beta_2)\cdot(g_t \circ g_t)$
$m_{t,i} = g_{t,1} + \beta_t m_{t-1,i}$
$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\mathcal{E}^{g\circ g}_{t+1,i}+\varepsilon}} \cdot m_{t,i}$
end for

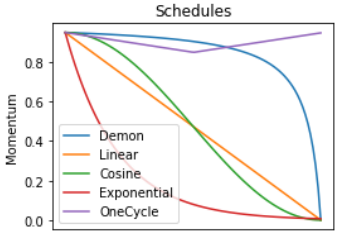


Fig. 1: *Non-linear* DEMON schedule vs. and other schedules, from $\beta_{init} = 0.9$ to 0. x-axis represents iterations.

where usually $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

3. DEMON: DECAYING MOMENTUM ALGORITHM

Our goal here is to present a concrete, effective, and easy-to-use momentum decay procedure. DEMON, motivated by reducing the total contribution of a gradient to future updates, decays momentum as follows:

$$\beta_t = \beta_{\text{init}} \cdot \frac{\left(1-\frac{t}{T}\right)}{(1-\beta_{\text{init}})+\beta_{\text{init}}\left(1-\frac{t}{T}\right)} . \tag{1}$$

Fig. 1 plots DEMON alongside other common schedules.

Interpretation. Assume a fixed momentum parameter $\beta_t \equiv \beta$ (e.g., $\beta = 0.9$). Because $v_0 = 0$ and $v_t = \beta v_{t-1} - g_t$, the SGDM recursion yields:

$$\theta_{t+1} = \theta_t - \eta g_t - \eta\beta g_{t-1} - \eta\beta^2 g_{t-2} + \eta\beta^3 v_{t-2}$$
$$= \dots = \theta_t - \eta g_t - \eta \cdot \sum_{i=1}^{t} \left(\beta^i \cdot g_{t-i}\right).$$

As shown above, *a particular gradient g_t contributes a total of $\eta\sum_i \beta^i$ of its "energy" to all future gradient updates.* For an asymptotically large number of iterations, β contributes to $t-1$ terms. Then, $\sum_{i=1}^{\infty}\beta^i = \beta\sum_{i=0}^{\infty}\beta^i = \beta/(1-\beta)$.

We aim to derive a schedule that decays cumulative momentum to 0. Let β_{init} be the initial β. At step t of T total steps, we design the decay routine such that $\beta/(1-\beta) = (1-t/T)\beta_{\text{init}}/(1-\beta_{\text{init}})$, yielding (1). Although β changes in subsequent iterations, this is a close approximation because $\beta^i\beta^{i+1}\dots\beta^t$ for a particular g^i diminishes much faster than β changes.

Connection to previous algorithms. For $\beta_t = 0.9$ and $\beta_t = 0$ in Algorithm 1 we obtain SGDM and SGD, respectively. For $\beta_1 = 0.9$ in Algorithm 2 (with a slight adjustment of learning rate) we obtain Adam, while for $\beta_1 = 0$ we obtain a non-accumulative AdaGrad.

Efficiency. DEMON (implemented below) causes minimal overhead.

```
p_t = (iters - t) / iters
beta_t = beta1 * (p_t / (1 - beta1 + beta1 * p_t))
```

Convergence analysis. *We provide convergence results for* DEMON *SGDM in the convex setting by bounding auxiliary function sequences* [20]. The full proof is omitted due to lack of space.

Theorem 1. *Assume that f is convex, continuously differentiable, its gradient $\nabla f(\cdot)$ is Lipschitz continuous with constant L, with a decreasing momentum, but constant step size, as in: $\beta_t = \frac{1}{t}\cdot\frac{t+1}{t+2}, \alpha \in \left(0, \frac{2}{3L}\right)$. We consider the SGDM iteration in non-stochastic settings, where: $\theta_{t+1} = \theta_t - \alpha\nabla f(\theta_t) + \beta_t(\theta_t - \theta_{t-1})$. Then, the sequence $\{\theta_t\}_{t=1}^T$ generated by the SGDM iteration, with decreasing momentum, satisfies:*

$$f(\bar{\theta}_T) - f^* \le \frac{\|\theta_1-\theta^\star\|^2}{T}\left(\frac{3}{4}L + \frac{1}{2\alpha}\right),$$

where $\bar{\theta}_T$ is the Cesaro average of the iterates: $\bar{\theta}_T = \frac{1}{T}\sum_{t=1}^T \theta_t$.

For DEMON Adam, we observe it lies within the definition of Generic Adam in [21], and inherits the non-convex results. Namely, to restate the theorem [21]:

Theorem 2. *Assuming $\tau \in \{1, 2, \dots, T\}$ is an index over T iterations, Demon + Adam satisfies:*

$$\mathbb{E}[\|\nabla f(x_\tau)\|_2^{4/3}]^{3/2} \le \frac{C + C'\sum_t \alpha_t\sqrt{\{1-\theta_t\}}}{\alpha_T \cdot T} = O\left(\frac{1}{T}\right),$$

where C, C' are constants, α_t are step sizes, θ_t are the parameters related to the diagonal Hessian approximation of Adam's preconditioner, and $\beta_t < 1$, as is the case for Demon technique.

Practical suggestions. We advocate for decaying momentum from β_{init} at $t = 0$, to 0 at $t = T$ as a general rule, which is the setting we use for all DEMON experiments in this paper.

4. EXPERIMENTS

We perform experiments in many, widely-used settings; see Table 2. Experiments with different total epochs are standalone experiments with independently-tuned hyperparameters.

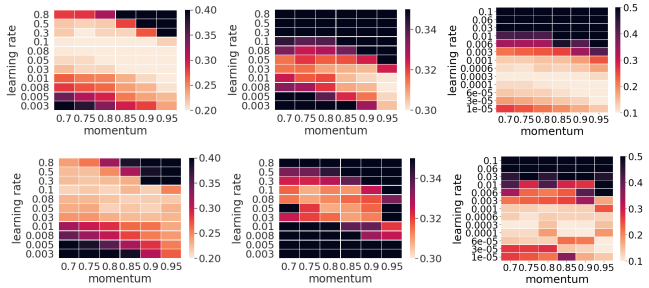
Table 2: Summary of experimental settings.

Experiment short name	Model	Dataset
RN20-CIFAR10	ResNet20	CIFAR10
RN56-TINYIMAGENET	ResNet56	Tiny ImageNet
VGG16-CIFAR100	VGG-16	CIFAR100
WRN-STL10	Wide ResNet 16-8	STL10
LSTM-PTB	LSTM RNN	Penn TreeBank
VAE-MNIST	VAE	MNIST
NCSN-CIFAR10	NCSN	CIFAR10
CAPS-FMNIST	Capsnet	FMNIST
BERT_{BASE}-GLUE	BERT (Pre-trained)	GLUE (9 tasks)

Learning rate and weight decay are tuned in multiples of 3. We test $\beta \in \{0.9, 0.95, 0.97\}$. "Cost" refers to whether other, specialized hyperparameters exist that must be tuned. We consider the following schedules; note we test some of these schedules as both learning rate and momentum rules:

- *No schedule:* Self-explanatory (**1× cost**).

Fig. 2: Left to right: Error rate for `WRN-STL10-DEMONSGDM` (top) and `WRN-STL10-SGDM` (bottom) for 50 epochs, error rate for `VGG16-CIFAR100-DEMONSGDM` (top) and `VGG16-CIFAR100-SGDM` (bottom) for 100 epochs, and error rate for `RN20-CIFAR10-DEMONAdam` (top) and `RN20-CIFAR10-Adam` (bottom) for 100 epochs. Light-colored patches indicate better performance.



- *Step schedule:* Achieves strong results in the literature [22, 11, 23, 24, 10]. We attempt decay schedules including $0.1\times$ at 50% and 75% of total epochs; $0.1\times$ at 25%, 50%, 75%; $0.1\times$ at 33% and 66%; $0.1\times$ at 10%, 25%, 50%, 75% (**4× cost**).
- *Cosine schedule[25]:* Follows the smooth schedule of $\gamma_t = \gamma_{min} + 0.5 \cdot (\gamma_{max} - \gamma_{min})(1 + cos(\pi t/T))$. γ can be the learning rate or the momentum. We consider $\gamma_{min} = 0$ to achieve (**1× cost**).
- *OneCycle[26]:* This scheme roughly follows increasing learning rate linearly from 0.1η to η and back to 0.1η, while simultaneously decreasing momentum linearly from β_{max} to β_{min} and back to β_{max}. For the momentum values, we consider the pairs [(0.95, 0.85), (0.9, 0.85), (0.95, 0.9)]. For the momentum variant, we simply consider the momentum component of OneCycle (**1× cost**).
- *Linear schedule:* Decreases the hyperparameter from the initial value to 0 across epochs (**1× cost**).
- *Exponential schedule[27]:* Follows the smooth schedule of $\gamma_t = \gamma \cdot e^{kt}$. We tune k from a reasonable starting point, $-0.05 \cdot (100/T)$ which is a scaled version of the curve in Figure 1 (i.e., plotted for 100 iterations in multiplies of 2) (**∼4× cost**).
- *Decay on Plateau[27]:* Commonly used in practice where the learning rate is multiplied by a factor if validation loss stops improving for a certain number of epochs (patience). We tune the patience in multiples of 5, and multiply the learning rate by 0.1 (**∼5× cost**).
- DEMON: This paper. The schedule follows Algorithm 1 and decays momentum to 0 at the end of training (**1× cost**).

We apply these schedules to SGDM and Adam, focusing on the performance of different schedules. *The overall tuning budget for* DEMON *SGDM/Adam is generally equal to or less than that of SGDM/Adam with its relevant possible schedules.*

4.1. Decreasing the need for tuning

In Fig 2, we plot DEMON SGDM and DEMON Adam performance relative to SGDM with learning rate step decay (i.e., we use the highest-performing learning rate schedule) and Adam. In all settings, `Demon` has larger bands of light color, indicating better performance for a wider range of hyperparameters. For example, on `WRN-STL10-DEMONSGDM`, SGDM has roughly one configuration per column with $< 22\%$ generalization error, while DEMON SGDM has five. Similarly, Adam performance fluctuates on `RN20-CIFAR10`, while DEMON Adam yields a wide band of high performance across hyperparameters. These results suggest that both DEMON Adam and DEMON SGDM are less sensitive to hyperparameter tuning than their vanilla counterparts.

Fig. 3: Randomly selected CIFAR10 images generated with NCSN. Left: Real CIFAR10 images. Middle: Adam. Right: DEMON Adam.

4.2. Results

For benchmarking purposes, we also include some other baselines where the learning rate and/or momentum are automatically adapted. These include QHAdam [7], QHM [7], AMSGrad [28], AdamW [29], YellowFin [17], AggMo [30]. QHM family is capable of recovering Accelerated SGD, Nesterov Accelerated Gradient, Synthesized Nesterov Variants [31], and others, thus covering more algorithms than those present. *We emphasize that* DEMON *can be combined with any momentum method. We present results with SGDM and Adam due to their wide usage.*

Mainline results. We summarize the results of all relevant settings in Table 1. Out of the 28 relevant settings in this paper, DEMON achieves the highest percentage of Top-1 and Top-3 finishes, with 39% and 85% respectively. Other momentum decay schedules are not competitive, likely due to being overly aggressive. Learning rate step schedule, linear schedule, and cosine schedule perform comparably; however, the different learning rate schedulers do not always yield comparable performance across settings. For example, linear learning rate decay performs exceptionally well in the ResNet settings, but closer to average on other settings. Such results indicate that the decay schedule, for learning rate or momentum, is an additional hyperparameter that must be tuned. Even though Decay On Plateau and Exponential Decay are implemented in most popular frameworks as empirical alternatives, they perform poorly when the total number of epochs are predefined. DEMON is the most consistent across a wide range of settings, with by far the highest Top-3 performance. Due to lack of space, tables or figures are omitted, but the overall performance is reported in Table 1. Below, we bring forward some observations, based on our extensive experiments in Table 3.

Residual Networks (`RN20-XX`, `WRN-XX`, `RN56-XX`). We emphasize that ResNet20 is commonly conflated with the more expressive ResNet18, which achieves different performance. STL-10 and Tiny ImageNet both have higher resolution images in comparison to CIFAR. We used all three datasets in our experiments. Due to limited resources, `RN56-XX` is run to supplement the other settings.

The momentum component of OneCycle, when used in isolation, appears to destabilize training, leading in some cases to performance worse than the vanilla counterpart. The learning rate step schedule has no clear advantage over the learning rate cosine schedule, the learning rate linear schedule, or DEMON. DEMON has strong performance, outperforming methods that automatically tune the momentum.

Non-Residual Convolutional Network (`VGG16-CIFAR100`). We slightly adjust the VGG-16 model [34] for the CIFAR-100 dataset, and results follow previous general trends. DEMON performs the best among the algorithms under consideration.

LSTM (`PTB-LSTM`). We use the official TensorFlow v1 implementation for PTB - LSTM. As a whole, smooth decay methods do not

Table 3: RN20-CIFAR10, WRN-STL10 and RN56-TINYIMAGENET generalization error, VGG16-CIFAR100 generalization error, LSTM-PTB generalization perplexity, VAE-MNIST generalization loss. The number of epochs was predefined before the execution of the algorithms. Red indicates Top-1 performance, bold is Top-3. Adaptive and non-adaptive methods are considered separately.

	ResNet 20	**Wide ResNet 16-8**	**ResNet 56**	**VGG-16**	**LSTM**	**VAE**
SGDM	300 epochs	300 epochs	-	300 epochs	39 epochs	100 epochs
+ LR Step Schedule	**7.32** ± .14	**14.51** ± .26	41.66 ± .10	**27.83** ± .30	**82.02** ± .13	137.70 ± .93
+ LR Cosine Schedule	7.78 ± .14	14.66 ± .25	-	**27.84** ± .12	**83.23** ± .06	**136.69** ± .27
+ OneCycle	8.42 ± .12	19.00 ± .42	-	29.09 ± .12	91.19 ± .01	137.20 ± .06
+ LR Linear Schedule	**7.62** ± .12	**14.58** ± .18	-	28.26 ± .08	98.79 ± .02	141.72 ± .48
AggMo + LR Step	7.62 ± .03	14.49 ± .26	-	28.64 ± .45	83.43 ± .17	**136.56** ± .28
QHM + LR Step	7.67 ± .10	**14.44** ± .23	-	29.01 ± .54	88.42 ± .10	137.97 ± .54
DEMON SGDM	**7.59** ± .12	**14.44** ± .53	40.85 ± .01	**27.69** ± .11	84.84 ± .22	**136.55** ± .64
Adam	11.94 ± .06	18.65 ± .07	50.89 ± .59	31.09 ± .09	116.26 ± .10	134.64 ± .14
+ LR Step Schedule	**8.55** ± .05	18.29 ± .10	-	**27.75** ± .15	**99.69** ± .03	134.14 ± .56
+ LR Cosine Schedule	**8.93** ± .07	19.08 ± .36	-	**28.08** ± .10	**100.56** ± .27	**133.25** ± .26
+ OneCycle	9.03 ± .18	19.03 ± .43	-	29.58 ± .18	117.31 ± .11	**133.27** ± .07
+ LR Linear Schedule	8.89 ± .05	**17.85** ± .15	-	28.65 ± .10	115.24 ± .10	134.00 ± .49
AMSGrad	10.48 ± .12	**18.21** ± .18	-	31.62 ± .12	103.26 ± .17	135.69 ± .03
AdamW	10.50 ± .17	**17.00** ± .41	-	32.22 ± .13	110.72 ± 1.63	134.68 ± .19
QHAdam	13.36 ± .11	18.52 ± .25	-	30.97 ± .10	106.18 ± .32	134.84 ± .08
YellowFin	11.39 ± .16	18.56 ± .33	-	50.18 ± 4.02	109.04 ± .20	351.80 ± 6.68
DEMON Adam	**8.50** ± .12	18.62 ± .41	45.72 ± .31	**27.11** ± .19	**102.07** ± .05	**133.98** ± .40

Table 4: Results of BERT$_{BASE}$-GLUE. Adam + LR Linear Schedule follows the huggingface [32] implementation, and achieves the results in well-known studies [13, 33].

	Score	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
Adam + LR Linear Schedule	79.1	57.4	84.3	89.0	91.4	89.2	69.4	92.7	89.0	49.6
DEMON Adam	**79.7**	58.4	84.2	90.0	90.9	89.0	69.4	92.5	88.8	53.8

Table 5: VGG16-CIFAR100-DEMONSGDM, RN20-CIFAR10-DEMONSGDM generalization error, PTB-LSTM-DEMONSGDM (perplexity) and VAE-MNIST-DEMONSGDM generalization loss. The number of epochs was predefined before the execution.

	RN-20		**VGG-16**		**LSTM**		**VAE**	
	150 epochs	300 epochs	100 epochs	200 epochs	25 epochs	39 epochs	50 epochs	100 epochs
SGD ELR	8.58 ± .08	8.16 ± .15	30.11 ± .28	29.21 ± .32	inf	inf	inf	inf
DEMON SGDM	**7.95** ± .15	**7.59** ± .12	**28.67** ± .11	**27.69** ± .11	**82.66** ± .105	**84.84** ± .22	**138.29** ± .08	**136.55** ± .64

perform comparably to step decay methods on this task. DEMON achieves comparable performance with both SGDM and Adam.

Variational AutoEncoder (VAE-MNIST). VAEs [35] are generative models that pair a generator network with a recognition model that performs approximate inference and can be trained with backprop. General trends follow the ResNet settings.

Noise Conditional Score Network (NCSN-CIFAR10. We train a NCSN [36], a recent generative model, on CIFAR10, for which NCSN achieves a strong inception score. Although vanilla Adam achieves a slightly superior inception score (**8.15** ± .20 vs. 8.07 ± .08), the results in Figure 3 are unnaturally green compared to those produced by DEMON Adam.

BERT (BERT$_{BASE}$-GLUE). We achieve the performance in well-known studies [13, 33] on fine-tuning BERT [13] on the GLUE benchmark [37]. We also only tune the learning rate for DEMON Adam. Running the same seeds, DEMON Adam yields a slight improvement over the baseline (Table 4).

Demon and Effective Learning Rate. The effective learning rate is proposed to approximate SGDM with SGD, where the learning rate is adjusted with a factor of $1/(1-m)$ and m is momentum. However, the results in Tables 5 demonstrate that DEMON cannot be accurately approximated with an effective learning rate adjusted SGD.

5. CONCLUSION

We show the effectiveness of DEMON across a large number of datasets and architectures with both SGDM and Adam. Compared to other schedules, DEMON achieves the largest number of Top-1 and Top-3 finishes. This includes improvements over the popular learning rate step schedule, cosine decay schedule, OneCycle, and many others. DEMON is computationally cheap, understandable, and easy to implement.

6. REFERENCES

[1] Tom B. Brown et al., "Language models are few-shot learners," 2020.

[2] Or Sharir, Barak Peleg, and Yoav Shoham, "The cost of training nlp models: A concise overview," *arXiv preprint arXiv:2004.08900*, 2020.

[3] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[4] Matthew D Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[5] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, pp. 8, 2012.

[6] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[7] Jerry Ma and Denis Yarats, "Quasi-hyperbolic momentum and Adam for deep learning," *arXiv preprint arXiv:1810.06801*, 2018.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in NeurIPS*, 2012, pp. 1097–1105.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on CVPR*, 2016, pp. 770–778.

[10] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[14] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz, "Revisiting distributed synchronous SGD," *arXiv preprint arXiv:1604.00981*, 2016.

[15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[16] Ioannis Mitliagkas, Ce Zhang, Stefan Hadjis, and Christopher Ré, "Asynchrony begets momentum, with an application to deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 997–1004.

[17] Jian Zhang and Ioannis Mitliagkas, "Yellowfin and the art of momentum tuning," *arXiv preprint arXiv:1706.03471*, 2017.

[18] Mehdi Mirza and Simon Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[19] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[20] Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson, "Global convergence of the heavyball method for convex optimization," *arXiv preprint arXiv:1412.7457*, 2014.

[21] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu, "A sufficient condition for convergences of adam and rmsprop," *arxiv preprint arXiv:1811.09358*, 2018.

[22] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu, "Squeeze-and-excitation networks," *arxiv preprint arXiv:1709.01507*, 2017.

[23] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," *CVPR*, 2017.

[24] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang, "Residual attention network for image classification," *CVPR*, 2017.

[25] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017.

[26] Leslie Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[27] Martin Abadi et al., "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on OSDI*, 2016, pp. 265–283.

[28] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar, "On the convergence of Adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[29] Ilya Loshchilov and Frank Hutter, "Fixing weight decay regularization in adam," *arXiv preprint arXiv:1711.05101*, 2017.

[30] James Lucas, Shengyang Sun, Richard Zemel, and Roger Grosse, "Aggregated momentum: Stability through passive damping," *arXiv preprint arXiv:1804.00325*, 2018.

[31] Laurent Lessard, Benjamin Recht, and Andrew Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.

[32] Thomas Wolf et al., "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[33] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[34] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes.," *arXiv preprint arXiv:1312.6114*, 2015.

[36] Yang Song and Stefano Ermon, "Generative modeling by estimating gradients of the data distribution," *arXiv preprint arXiv:1907.05600*, 2019.

[37] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2019, In the Proceedings of ICLR.