# Momentum Enables Large Batch Training

**Samuel L. Smith** [1] **Erich Elsen** [1] **Soham De** [1]

## Abstract

We provide both a theoretical and an empirical comparison of Stochastic Gradient Descent (SGD) and Stochastic Gradient Descent with Momentum for training deep learning architectures. We find that both methods have near-identical performance when the learning rate $\epsilon \to 0$ (after careful tuning). However Momentum outperforms vanilla SGD when the learning rate is large. Since the noise inherent in stochastic gradients is proportional to the ratio of the learning rate to the batch size, this implies that SGD and Momentum usually have near-identical performance when the batch size is small, while Momentum outperforms SGD when the batch size is large. We thus conclude that Momentum does not typically improve the efficiency of training, measured by the number of epochs, but it can reduce the wall clock time.

## 1. Small learning rate stochastic optimization

We first analyze the behaviour of Stochastic Gradient Descent (SGD) and Heavy-Ball Momentum as the learning rate $\epsilon \to 0$. While this limit is somewhat artificial, it will enable us to make concrete predictions which are often valid in practice. As we will observe empirically in section 3, this analysis is usually sufficient to describe the behaviour of both algorithms when the batch size is small. We will discuss the large learning rate limit, relevant to large batch training, in section 2. The goal of sections 1 and 2 is to help the reader build an intuition for our empirical results. More rigorous analysis of both SGD and Heavy-Ball Momentum can be found in a number of recent papers (Sutskever et al., 2013; Yang et al., 2016; Yuan et al., 2016; Li et al., 2017; Mandt et al., 2017; Smith & Le, 2017; Kidambi et al., 2018; Gadat et al., 2018). The analysis provided in section 1.2 carries over straightforwardly to Nesterov momentum.

[1]DeepMind, London, UK. Correspondence to: Samuel L. Smith <slsmith@google.com>.

Presented at the ICML 2019 Workshop on Theoretical Physics for Deep Learning. 

### 1.1. Stochastic Gradient Descent

The $s^{\text{th}}$ update of SGD is given by,

$$\Delta\omega_s = -\frac{\epsilon}{B}\sum_{i=1}^{B}\frac{dL(y_i, x_i, \omega_s)}{d\omega}, \tag{1}$$

where $(x, y)$ denotes the inputs and labels of a training set of size $N \gg B$, where $B$ is the batch size, and $L(y_i, x_i, \omega)$ is the loss of the $i^{\text{th}}$ training example. For simplicity we assume the indices $i$ are randomly reshuffled between each update, such that training batches are sampled randomly without replacement. We also assume the per example gradients are both smooth and bounded, such that $v \cdot \frac{dL(y_i,x_i,\omega)}{d\omega} < G$ for all $(i, \omega)$ and any normalized vector $v$. For example, this scenario arises for quadratic losses if the distance of the iterates from the minimum is bounded. As discussed in section 2, this assumption is not necessary to derive our main results, but it simplifies the analysis. We defer to Yuan et al. (2016) for a more general treatment. To interpret equation 1, we separate the update into signal and noise,

$$\Delta\omega_s = -\epsilon\left(\frac{dC}{d\omega}\Big|_{\omega_s} + \frac{\delta_s}{\sqrt{B}}\right). \tag{2}$$

The full-batch loss $C(\omega) = \frac{1}{N}\sum_{i=1}^{N}\frac{dL(y_i,x_i,\omega)}{d\omega}$, and the gradient noise $\delta_s = \delta(\omega_s)$. Applying the central limit theorem, we assume $\delta_s$ is an uncorrelated Gaussian noise source with mean $\langle\delta_s\rangle = 0$ and covariance $\langle\delta_s\delta_{s'}^{\top}\rangle = \Sigma(\omega_s)\delta_{ss'}$, where $\delta_{ss'}$ is the Dirac delta function (Mandt et al., 2017; Smith & Le, 2017). We discuss this approximation and derive the $1/\sqrt{B}$ scaling factor in appendix A. We note that this $1/\sqrt{B}$ factor does not hold with batch normalization (Ioffe & Szegedy, 2015), although it does hold with ghost batch norm if the ghost batch size is constant (Hoffer et al., 2017). We introduce the temperature $T = \epsilon/B$ to obtain,

$$\Delta\omega_s = -\epsilon\frac{dC}{d\omega}\Big|_{\omega_s} + \sqrt{\epsilon T}\delta_s. \tag{3}$$

We identify equation 3 as the Euler discretization of a stochastic differential equation (Arnold, 1974)[1], whose timestep $\Delta t = \epsilon$. After $n$ consecutive parameter updates,

$$\Delta\omega' = \sum_{i=s}^{s+n}\Delta\omega_i = -\epsilon\sum_{i=s}^{s+n}\frac{dC}{d\omega}\Big|_{\omega_i} + \sqrt{\epsilon T}\sum_{i=s}^{s+n}\delta_i. \tag{4}$$

[1]Strictly speaking, this interpretation only holds when the covariance matrix $\Sigma(\omega)$ is independent of the parameters.

To simplify equation 4, we assume that the time interval $t = n\epsilon \to 0$. Since $|\omega_{s+n} - \omega_s| \leq n\epsilon G$, this permits us to approximate $\frac{dC}{d\omega}\big|_{\omega_i} = \frac{dC}{d\omega}\big|_{\omega_s}$ and $\Sigma(\omega_i) = \Sigma(\omega_s)$ for all $i \in \{s, s+n\}$. We therefore conclude,

$$\Delta\omega' = -n\epsilon \frac{dC}{d\omega}\Big|_{\omega_s} + \sqrt{n\epsilon T}\xi_s. \tag{5}$$

We introduce a new random variable $\xi_s = \frac{1}{\sqrt{n}} \sum_{i=s}^{s+n} \delta_i$ to describe the cumulative influence of the noise in each update. This random variable will also be Gaussian, with mean $\langle \xi_s \rangle = 0$ and covariance,

$$\langle \xi_s \xi_s^\top \rangle = \frac{1}{n} \sum_{i=s}^{s+n} \sum_{j=s}^{s+n} \delta_i \delta_j^\top \tag{6}$$

$$= \frac{\Sigma(\omega_s)}{n} \sum_{i=s}^{s+n} \sum_{j=s}^{s+n} \delta_{ij} = \Sigma(\omega_s). \tag{7}$$

This confirms that $\xi$ and $\delta$ are random variables from the same distribution. Comparing equations 3 and 5, we conclude that if $n\epsilon \to 0$, then $n$ SGD steps at temperature $T$ with learning rate $\epsilon$ is equivalent to a single SGD step at temperature $T$ with learning rate $\epsilon' = n\epsilon$. Since the temperature $T = \epsilon/B$, to maintain a constant temperature we must scale the batch size $B \propto \epsilon$. This "linear scaling rule" has been predicted by a number of papers and observed empirically on a wide range of models (Krizhevsky, 2014; Balles et al., 2016; Goyal et al., 2017; Smith et al., 2017; Jastrzębski et al., 2017; Chaudhari & Soatto, 2018; McCandlish et al., 2018; Shallue et al., 2018). It implies that the dynamics of small learning rate SGD depends only on the number of training epochs performed and the temperature $T = \epsilon/B$.

In practice we apply equation 3 at a finite step size $\epsilon > 0$, however we will assume that whenever the linear scaling rule holds, SGD is well approximated by the underlying stochastic differential equation. This scaling rule can be used to increase the batch size and reduce the number of parameter updates, without sacrificing final performance and without hyper-parameter tuning (Goyal et al., 2017). However the scaling rule will break when the learning rate is large (McCandlish et al., 2018; Shallue et al., 2018).

Practitioners often introduce a decaying learning rate, $\epsilon_s = \epsilon g(t)$. In this case the time $t(s) = \sum_{i=0}^{s} \epsilon_i$. This does not affect the linear scaling rule, which will hold whenever the initial learning rate $\epsilon$ is sufficiently small (Smith et al., 2017). The use of a decaying learning rate causes the temperature $T = \epsilon g(t)/B$ to fall during training, thus enabling the parameters $\omega$ to concentrate near the minimum (Mandt et al., 2017). As proposed by a number of recent works, one can exploit the linear scaling rule to replace learning rate decay with increasing batch sizes (Friedlander & Schmidt, 2012; Byrd et al., 2012; De et al., 2017; Smith et al., 2017).

### 1.2. Momentum

We demonstrated above that a single SGD step corresponds to the Euler discretization of a stochastic differential equation. Here we will establish a similar analysis, which demonstrates that Heavy-Ball Momentum with small learning rates is equivalent to SGD with an adjusted step size (Yuan et al., 2016). The $s^{\text{th}}$ parameter update is composed of an unnormalized exponential moving average of the gradient,

$$\Delta\omega_s = -\epsilon \left( \sum_{i=0}^{s} m^{s-i} \left( \frac{dC}{d\omega}\Big|_{\omega_i} + \frac{\delta_i}{\sqrt{B}} \right) \right). \tag{8}$$

We introduce the momentum coefficient $m$, which we assume is bounded by a constant $\alpha$: $0 < m \leq \alpha < 1$. We note that $|\omega_i - \omega_s| \leq \epsilon(s-i)G/(1-m)$ (see below), while contributions to the sum in equation 8 vanish when $m^{s-i} < \alpha^{s-i} \ll 1$. Thus if $\epsilon \to 0$ we may approximate,

$$\Delta\omega_s = -\epsilon \left( \sum_{i=0}^{s} m^{s-i} \left( \frac{dC}{d\omega}\Big|_{\omega_s} + \frac{\delta_i}{\sqrt{B}} \right) \right). \tag{9}$$

Intuitively, when $\epsilon \to 0$ and $m$ is fixed below 1, Momentum will take an exponential moving average over noisy gradient estimates at the same location in parameter space. By the time the parameters move, the stale gradients in this average have been forgotten. Evaluating the geometric series,

$$\Delta\omega = -\epsilon_{\text{eff}} \frac{dC}{d\omega}\Big|_{\omega_s} + \sqrt{\epsilon_{\text{eff}} T}\gamma_s. \tag{10}$$

We redefine the temperature $T = \epsilon_{\text{eff}}/B$, and introduce the effective learning rate, $\epsilon_{\text{eff}} = \epsilon/(1-m)$. Note that we have assumed $m^s \ll 1$. When $m^s \sim 1$ the effective learning rate is suppressed, which can cause Momentum to lag behind SGD at the start of training. The Gaussian random variable $\gamma_s$ has mean $\langle \gamma_s \rangle = 0$ and co-variance, $\langle \gamma_q \gamma_w \rangle = (1-m)^2 \sum_{i=0}^{q} \sum_{j=0}^{w} m^{(q+w-i-j)} \langle \delta_i \delta_j \rangle$. Notice that unlike the noise introduced in a single SGD step, the noise present in a single Momentum update is correlated with neighbouring updates. Finally, in order to compare SGD and Momentum we must evaluate the combined effect of $n$ consecutive Momentum updates as $n\epsilon \to 0$,

$$\Delta\omega' = \sum_{i=s}^{s+n} \Delta\omega_{s+i} \tag{11}$$

$$= -n\epsilon_{\text{eff}} \frac{dC}{d\omega}\Big|_{\omega_s} + \sqrt{n\epsilon_{\text{eff}} T}\zeta_s \tag{12}$$

The Gaussian random variable $\zeta_s$ describes the cumulative influence of the noise in each update, with mean $\langle \zeta_s \rangle = 0$ and covariance $\langle \zeta_s \zeta_s^\top \rangle = (1/n) \sum_{q=s}^{s+n} \sum_{w=s}^{s+n} \langle \gamma_q \gamma_w^\top \rangle$. In appendix B we show that when $m^n \ll 1$, $\langle \zeta_s \zeta_s^\top \rangle = \Sigma(\omega_s)$. Thus, comparing equations 5 and 12, we conclude that if the learning rate is sufficiently small, computing $n$ SGD updates

with learning rate $\epsilon_{\text{SGD}}$ and batch size $B$ is equivalent to computing $n$ Momentum updates with effective learning rate $\epsilon_{\text{eff}} = \epsilon_{\text{SGD}}$ and batch size $B$. We therefore anticipate that when the learning rate is sufficiently small, Momentum and SGD will have near-identical performance on both training and test set after the same number of training epochs, so long as their temperatures are equal (Mandt et al., 2017; Smith & Le, 2017). Just like SGD, Momentum will exhibit a linear scaling rule, and we anticipate that the performance of Momentum and SGD will only differ once the learning rate is large enough to break this linear scaling relationship.

## 2. Large learning rate stochastic optimization

In section 1, we established an equivalence between SGD and Momentum, which arises when the effective learning rate is sufficiently small. For SGD, $\epsilon_{\text{eff}} = \epsilon$, while for Momentum $\epsilon_{\text{eff}} = \epsilon/(1-m)$. However, in the deterministic setting Momentum can substantially reduce the number of parameter updates required to reach the minimum (Polyak, 1964; Nesterov, 1983). Crucially, this arises because Momentum updates are often more stable than SGD when the effective learning rate is large (Qian, 1999). It would therefore be useful to estimate the threshold learning rate $\epsilon_{\text{thres}}$, above which linear scaling no longer holds. We predict that Momentum can only outperform SGD when $\epsilon_{\text{eff}} \gtrsim \epsilon_{\text{thres}}$.

A key assumption behind linear scaling states $\frac{dC}{d\omega}\big|_{\omega_{s+n}} \approx \frac{dC}{d\omega}\big|_{\omega_s}$ for $n \geq 1$. In section 1, we bounded the gradient in order to ensure that the absolute change in the gradient vanished as $n\epsilon_{\text{eff}} \to 0$. However in practice we anticipate that it is the relative change in the gradient which matters. We therefore predict that linear scaling breaks down if,

$$\begin{aligned} v \cdot \left( \frac{dC}{d\omega}\Big|_{\omega_{s+1}} - \frac{dC}{d\omega}\Big|_{\omega_s} \right) &\approx \epsilon_{\text{eff}}\, v \cdot \left( \frac{d^2C}{d\omega^2} \frac{dC}{d\omega} \right) \Big|_{\omega_s} \\ &\gtrsim \beta v \cdot \frac{dC}{d\omega}\Big|_{\omega_s}. \end{aligned} \tag{13}$$

for any normalized vector $v$. The scalar $\beta > 0$ defines how much approximation error we are willing to tolerate, but for simplicity we fix $\beta \sim 1$. To analyze equation 13, we restrict ourselves to deterministic updates on a simple quadratic loss, $C(\omega) = \omega^\top H \omega/2$. This allows us to simplify,

$$\epsilon_{\text{eff}}\, v^\top H^2 \omega_s \gtrsim \beta v^\top H \omega_s. \tag{14}$$

This approximation breaks first when $v$ is parallel to $h_{\text{max}}$, the largest eigenvalue of $H$. We thus predict that a difference between Momentum and SGD will arise once $\epsilon_{\text{eff}} \gtrsim \epsilon_{\text{thres}}$, where $\epsilon_{\text{thres}} \sim 1/h_{\text{max}}$. The threshold $\epsilon_{\text{thres}}$ has a simple interpretation; it is the learning rate at which SGD begins to oscillate across the floor of a quadratic valley. We note that $\epsilon_{\text{thres}}$ estimates the largest stable learning rate of SGD in the deterministic case. However in the stochastic setting, the largest stable learning rate is often significantly smaller due to gradient noise. As shown in section 1, in this noisy limit the largest stable learning rate is proportional to the batch size, and we therefore conclude that Momentum will only outperform SGD if the batch size is sufficiently large to achieve stable effective learning rates $\epsilon_{\text{eff}} \gtrsim \epsilon_{\text{thres}}$.

The threshold learning rate is inversely proportional to the largest Hessian eigenvalue, suggesting Momentum may outpeform SGD for smaller learning rates (and smaller batch sizes) when the loss is poorly conditioned. This may explain the popularity of Momentum in the literature (Sutskever et al., 2013), as we find evidence that Momentum outperforms SGD with relatively small batch sizes on early neural networks trained without batch norm (Ioffe & Szegedy, 2015), skip connections (He et al., 2016) and other innovations. However we have not found a popular benchmark for which Momentum outperforms SGD at batch size $B = 1$.

We emphasize that with linear scaling, increasing the batch size does not reduce the computational cost of training, since the number of epochs is constant. However as linear scaling reduces the number of parameter updates required, it can reduce the wall clock time if one has sufficient resources available to parallelize the computation of a large minibatch (Goyal et al., 2017). As models and datasets grow and compute becomes cheaper, wall clock time is an increasingly important factor limiting the rate of progress in the field.

## 3. Experiments

In the main text, we present a limited range of experiments with a 16-4 wide ResNet on CIFAR-10. Additional results with a wider range of models, datasets and momentum coefficients are presented in appendix C. While our theoretical analysis only applies without batch normalization or with ghost batch norm at fixed ghost batch size, we run our experiments with conventional batch norm to emphasize that this often does not affect our main conclusions in practice.

We omit a detailed description of model architecture and refer the reader to the original paper (Zagoruyko & Komodakis, 2016). We apply data augmentation including random crops, flips and reflections, and the $L_2$ regularization parameter was $5 \times 10^{-4}$. We train for 200 epochs using a modified learning rate decay schedule. This schedule holds the learning rate constant for 100 epochs, and then drops the learning rate by a factor of 2 every 10 epochs for the remaining 100 epochs. We found this simple schedule matched the final performance of the original schedule, but was less sensitive to the choice of the initial learning rate. In the main text, the momentum coefficient $m = 0.9$.

We consider small batch training ($B = 32$) in figure 1, and large batch training ($B = 1024$) in figure 2. Considering figure 1, we find that in the small batch limit, SGD and

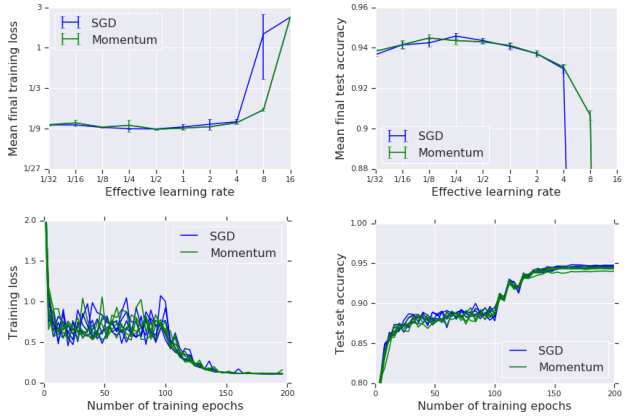


*Figure 1.* A 16-4 wide ResNet on CIFAR-10 with batch size 32. In the top panel we provide the final training loss and final test set accuracy as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. Both methods show near-identical performance when the effective learning rate is small, while momentum outperforms SGD when the learning rate is large. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 1/4$.

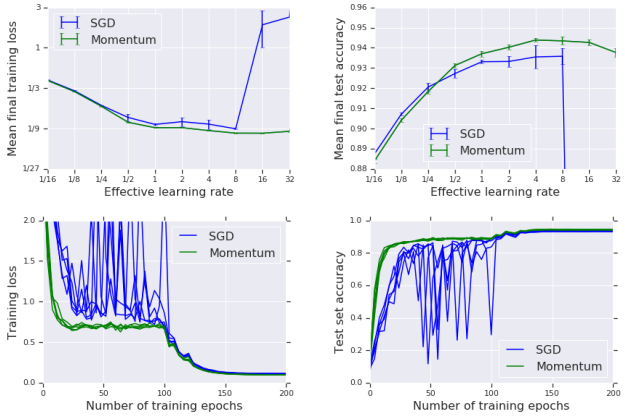


*Figure 2.* A 16-4 wide ResNet on CIFAR-10 with batch size 1024. In the top panel we provide the final training loss and final test accuracy as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. SGD slightly outperforms Momentum for very small effective learning rates, while Momentum outperforms SGD when the learning rate is large. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 8$.

Momentum exhibit near identical performance when the effective learning rate is small, while Momentum outperforms SGD when the effective learning rate is above a critical scale $\epsilon_{\text{thres}} \approx 4$. Crucially however, both the final training loss and the final test set accuracy are optimal for effective learning rates $\epsilon_{opt} \approx 1/4 \ll \epsilon_{\text{thres}}$. Consequently in this small batch limit, there is no advantage to the use of Momentum over SGD. To emphasize this point, in the lower panel of figure 1 we provide the evolution of both the training loss and test set accuracy during training at $\epsilon_{\text{eff}} = 1/4$. One can clearly see that the evolution of SGD and Momentum are near-identical (within noise) when properly tuned.

Considering figure 2, in the large batch limit Momentum slightly outperforms SGD once $\epsilon_{\text{eff}} \gtrsim 1/2$. This gap increases substantially for $\epsilon_{\text{eff}} \gtrsim 8$. We speculate that a small performance gap emerges earlier with large batches because the gradient noise is suppressed (Sutskever et al., 2013). Crucially the optimal learning rate $\epsilon_{opt} \approx 8$, at which value Momentum significantly outperforms SGD on both training and test set, thus confirming that Momentum improves final performance with large training batches. We also provide the learning curves at $\epsilon_{\text{eff}} = 8$, from which it is clear that SGD is initially unstable at this high learning rate (until the learning rate decays), while Momentum is stable throughout. Finally, we observe that the batch size in figure 2 is 32 times larger than that of figure 1, while $\epsilon_{\text{opt}}$ increased by the same factor, consistent with linear scaling. We note that the optimal effective learning rate will also depend on the number of training epochs and the learning rate schedule (Shallue et al., 2018). Our additional experiments in appendix C all exhibit the same qualitative trends shown in figures 1 and 2.

## 4. Related Work

Canonical results, reviewed in Goh (2017), emphasize the superiority of full-batch Momentum over gradient descent on poorly conditioned loss landscapes. Several papers in the continuous setting further emphasize these properties (Su et al., 2014; Wibisono et al., 2016; Shi et al., 2018). Meanwhile, it is also known that SGD with Polyak averaging is asymptotically *optimal* for convex losses (Polyak & Juditsky, 1992). In this paper, we investigate when Momentum starts showing benefits over SGD in minibatch training.

Previous papers have shown that SGD and Momentum are equivalent for small learning rates (after appropriate rescaling), both in the continuous limit (Orr & Leen, 1994; Wiegerinck et al., 1994) and discrete settings (Yuan et al., 2016; Ma & Yarats, 2018). However these results fail to explain why Momentum sometimes outperforms SGD during minibatch training. Kidambi et al. (2018) and Shallue et al. (2018) provide empirical evidence that Momentum enables reduced wall-clock times when the batch size is large. We provide a theoretical analysis of this effect, alongside additional empirical evidence. We also demonstrate a simple tuning strategy to convert between SGD schedules and Momentum schedules when the batch size is small.

A number of recent papers discuss the stochastic differential equation limit of SGD (Li et al., 2017; Jastrzębski et al., 2017; Chaudhari & Soatto, 2018), or observed linear scaling empirically (Krizhevsky, 2014; Goyal et al., 2017; Smith et al., 2017; McCandlish et al., 2018). Some of these derive a corresponding second order stochastic differential equation for Momentum (Mandt et al., 2017; Smith & Le, 2017).

## References


Arnold, L. Stochastic differential equations. *New York*, 1974.

Balles, L., Romero, J., and Hennig, P. Coupling adaptive batch sizes with learning rates. *arXiv preprint arXiv:1612.05086*, 2016.

Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

De, S., Yadav, A., Jacobs, D., and Goldstein, T. Automated inference with adaptive batches. In *Artificial Intelligence and Statistics*, pp. 1504–1513, 2017.

Friedlander, M. P. and Schmidt, M. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Gadat, S., Panloup, F., Saadane, S., et al. Stochastic heavy ball. *Electronic Journal of Statistics*, 12(1):461–529, 2018.

Goh, G. Why momentum really works. *Distill*, 2(4):e6, 2017.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Li, Q., Tai, C., et al. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2101–2110. JMLR. org, 2017.

Ma, J. and Yarats, D. Quasi-hyperbolic momentum and adam for deep learning. *arXiv preprint arXiv:1810.06801*, 2018.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Nesterov, Y. E. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Orr, G. B. and Leen, T. K. Momentum and optimal stochastic search. In *Proceedings of the 1993 Connectionist Models Summer School*, pp. 351–357. Psychology Press, 1994.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Shi, B., Du, S. S., Jordan, M. I., and Su, W. J. Understanding the acceleration phenomenon via high-resolution differential equations. *arXiv preprint arXiv:1810.08907*, 2018.

Smith, S. L. and Le, Q. V. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Su, W., Boyd, S., and Candes, E. A differential equation for modeling nesterovâĂŹs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pp. 2510–2518, 2014.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Wibisono, A., Wilson, A. C., and Jordan, M. I. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47): E7351–E7358, 2016.

Wiegerinck, W., Komoda, A., and Heskes, T. Stochastic dynamics of learning with momentum in neural networks. *Journal of Physics A: Mathematical and General*, 27(13): 4425, 1994.

Yang, T., Lin, Q., and Li, Z. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*, 2016.

Yuan, K., Ying, B., and Sayed, A. H. On the influence of momentum acceleration on online learning. *The Journal of Machine Learning Research*, 17(1):6602–6667, 2016.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

## A. Central limit theorem and minibatch noise

We assumed in the main text that the gradients of individual examples are bounded, such that $v \cdot \frac{dL(y_i,x_i,\omega)}{d\omega} < G$ for all $(i, \omega)$ and any normalized vector $v$. This implies the difference between the minibatch gradient and full batch gradient is also bounded, and consequently the gradient noise $\delta_s$ is guaranteed to be short tailed. Therefore the central limit theorem guarantees that $\delta_s$ will be well approximated by Gaussian noise in the limit $\{N \to \infty, B \to \infty, B \ll N\}$.

While the limits $\{N \to \infty, B \ll N\}$ are usually plausible in practice, the batch size $B$ is often relatively small. However we note that our main conclusions do not in fact require that $\delta_s$ is well described by Gaussian random noise, so long as the combined noise over multiple adjacent updates, $\xi_s = \frac{1}{\sqrt{n}} \sum_{i=s}^{s+n} \delta_i$, is well approximated by a Gaussian, where $n$ denotes the number of adjacent steps. The central limit theorem predicts that $\xi_s$ will be well approximated by a Gaussian so long as $\{N \to \infty, nB \to \infty, B \ll N\}$, and we therefore conclude that $\xi_s$ is likely to be close to Gaussian if we integrate over sufficiently many updates.

We now evaluate the covariance matrix, $\langle \delta_s \delta_s'^{\top} \rangle$. We note,

$$\frac{\delta_s}{\sqrt{B}} = \frac{1}{B} \sum_{i=1}^{B} \frac{dL(y_i, x_i, \omega_s)}{d\omega} - \frac{1}{N} \sum_{i=1}^{N} \frac{dL(y_i, x_i, \omega_s)}{d\omega}. \quad (15)$$

The expected gradient $\langle \frac{dL(y_i,x_i,\omega_s)}{d\omega} \rangle = \frac{dC}{d\omega}\Big|_{\omega_s}$. We assume that the gradients of individual examples are independent,

$$\left\langle \frac{dL(y_i, x_i, \omega_s)}{d\omega} \frac{dL(y_j, x_j, \omega_s)}{d\omega^{\top}} \right\rangle = \left( \frac{dC}{d\omega} \frac{dC}{d\omega^{\top}} \right) \Big|_{\omega_s} + \Sigma(\omega_s) \delta_{ij}. \quad (16)$$

We have introduced the empirical gradient covariance matrix $\Sigma(\omega_s)$. We therefore conclude that $\langle \delta_s \rangle = 0$ and,

$$\frac{1}{B} \langle \delta_s \delta_s'^{\top} \rangle = \left( \sum_{i=1}^{B} \left( \frac{1}{B} - \frac{1}{N} \right)^2 \Sigma(\omega_s) \right) \delta_{ss'} \quad (17)$$

$$= \frac{B}{N^2} \left( 1 - \frac{N}{B} \right)^2 \Sigma(\omega_s) \delta_{ss'}. \quad (18)$$

Recalling that $N \gg B$, we obtain,

$$\langle \delta_s \delta_s'^{\top} \rangle \approx \Sigma(\omega_s) \delta_{ss'}. \quad (19)$$

Thus confirming the $\frac{1}{\sqrt{B}}$ scaling provided in the main text. Re-arranging equation 16, we conclude,

$$\Sigma(\omega_s) = \left\langle \frac{dL(y_i, x_i, \omega_s)}{d\omega} \frac{dL(y_i, x_i, \omega_s)}{d\omega^{\top}} \right\rangle - \left( \frac{dC}{d\omega} \frac{dC}{d\omega^{\top}} \right) \Big|_{\omega_s}$$

We note that the assumption of independent gradients does not hold for batch normalization. We can recover equation 19 if we use ghost batch norm with a fixed ghost batch size.

## B. Noise covariance of Momentum updates

We wish to evaluate,

$$\langle \zeta_s \zeta_s^{\top} \rangle = (1/n) \sum_{q=s}^{s+n} \sum_{w=s}^{s+n} \langle \gamma_q \gamma_w^{\top} \rangle. \quad (20)$$

We recall that,

$$\langle \gamma_q \gamma_w^\top \rangle = (1-m)^2 \sum_{i=0}^{q} \sum_{j=0}^{w} m^{(q+w-i-j)} \langle \delta_i \delta_j^\top \rangle, \quad (21)$$

where $\langle \delta_i \rangle = 0$ and $\langle \delta_i \delta_j^\top \rangle = \Sigma(\omega_i)\delta_{ij}$. Thus,

$$\langle \zeta_s \zeta_s^\top \rangle = \frac{(1-m)^2 \Sigma(\omega_s)}{n} \sum_{q=s}^{s+n} \sum_{i=0}^{q} \sum_{w=s}^{s+n} \sum_{j=0}^{w} m^{(q+w-i-j)} \delta_{ij}. \quad (22)$$

Simplifying,

$$\begin{aligned} \frac{n\langle \zeta_s \zeta_s^\top \rangle}{(1-m)^2 \Sigma(\omega_s)} &= \sum_{q=s}^{s+n} \sum_{i=0}^{q} \sum_{w=\max(s,i)}^{s+n} m^{(q+w-2i)} \\ &= \sum_{q=s}^{s+n} \sum_{i=s}^{q} \sum_{w=i}^{s+n} m^{(q+w-2i)} \\ &+ \sum_{q=s}^{s+n} \sum_{i=0}^{s-1} \sum_{w=s}^{s+n} m^{(q+w-2i)}. \quad (23) \end{aligned}$$

Evaluating the first sum,

$$\begin{aligned} \sum_{q=s}^{s+n} \sum_{i=s}^{q} \sum_{w=i}^{s+n} m^{(q+w-2i)} &= \frac{1}{1-m} \sum_{q=s}^{s+n} \sum_{i=s}^{q} m^{q-i} \\ &= \frac{1}{1-m} \sum_{q=s}^{s+n} \sum_{x=0}^{q-s} m^{x} \\ &= \frac{1}{(1-m)^2} \sum_{q=s}^{s+n} (1-m^{q-s}) \\ &= \frac{1}{(1-m)^2} \left( n - \frac{1}{(1-m)} \right) \\ &= \frac{n}{(1-m)^2}. \quad (24) \end{aligned}$$

Note that we have assumed $m^s \ll 1$ and $m^n \ll 1$, which also implies that $n \gg 1/(1-m)$. The second sum,

$$\begin{aligned} \sum_{q=s}^{s+n} \sum_{i=0}^{s-1} \sum_{w=s}^{s+n} m^{(w+q-2i)} &= \frac{m^{2s}}{(1-m)^2} \sum_{i=0}^{s-1} m^{-2i} \\ &= \frac{m^2}{(1-m)^2(1-m^2)} \quad (25) \end{aligned}$$

Since $m^2/(1-m^2) < 1/(1-m) \ll n$, we conclude,

$$\langle \zeta_s \zeta_s^\top \rangle = \Sigma(\omega_s). \quad (26)$$

This confirms the relationship provided in the main text.

## C. Additional Experimental Results

In the main text we provided a comparison of SGD and Momentum with a 16-4 wide ResNet on CIFAR-10. Here we provide an additional sweep over a range of Momentum coefficients in section C.1, which demonstrates that smaller Momentum coefficients are usually best, unless the effective learning rate is very large. We provide an additional comparison of SGD and Momentum for the same 16-4 wide ResNet on CIFAR-100 in section C.2. We also provide a similar comparison between SGD and Momentum for a fully connected auto-encoder on MNIST (Sutskever et al., 2013) in section C.3, and for an LSTM model on Penn-TreeBank (Zaremba et al., 2014) in section C.4. In all cases we find that SGD and Momentum exhibit near-identical performance when the batch size is small, while Momentum outperforms SGD when the batch size is large. However the threshold batch size, above which Momentum outperforms SGD, is significantly smaller for the fully connected auto-encoder on MNIST than it is for the more recent ResNet architecture on CIFAR. Intriguingly, we note that the popularity of Momentum in the literature in recent years can be traced back to the work of Sutskever et al. (2013), who demonstrated the benefits of Momentum when training fully connected auto-encoders on MNIST at batch size 200.

### C.1. 16-4 wide ResNet on CIFAR-10

Our main results for this architecture and dataset were presented in the main text. However in figure 3, we plot the final training accuracy and the final test accuracy as a function of effective learning rate, for momentum coefficients in the range $[0.9, 0.9999]$ and for both small batch training ($B = 32$) and large batch training ($B = 1024$). When the effective learning rate is small, the smaller momentum coefficients $m \lesssim 0.99$ have similar performance, while very large momentum coefficients show larger training losses and smaller test accuracies. However when the effective

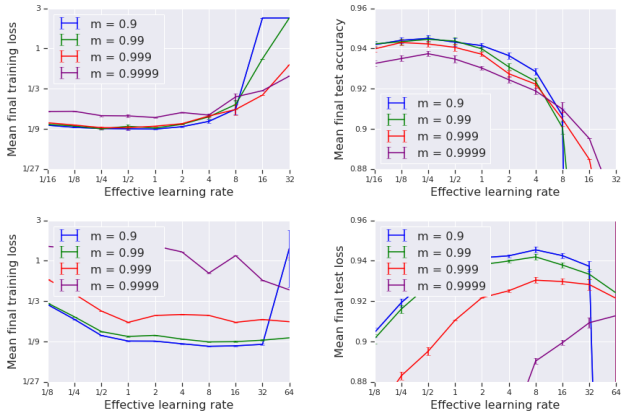


*Figure 3.* A 16-4 wide ResNet on CIFAR-10. In the top panel, we provide the final training loss and test accuracy as a function of the effective learning rate at batch size $B = 32$, for a range of momentum coefficients $m$. In the bottom panel we provide similar plots for $B = 1024$. Small momentum coefficients perform best at small effective learning rates, while larger momentum coefficients are more stable when the effective learning rate is large.

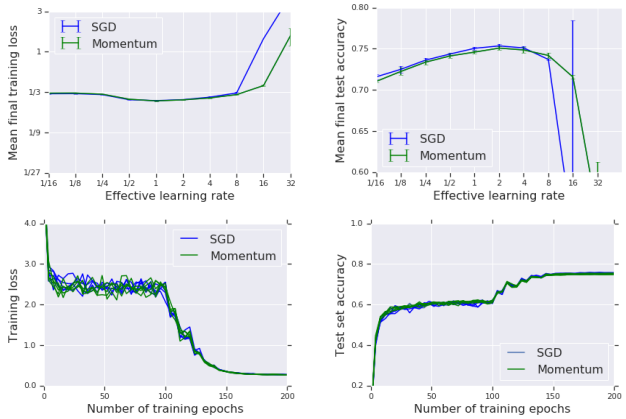


*Figure 4.* A 16-4 wide ResNet on CIFAR-100 with batch size 128. In the top panel we provide the final training loss and test accuracy as a function of the effective learning rate. We perform 5 training runs for each learning rate, and provide the mean and standard deviation. In the bottom panel we plot training loss and test accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 2$.

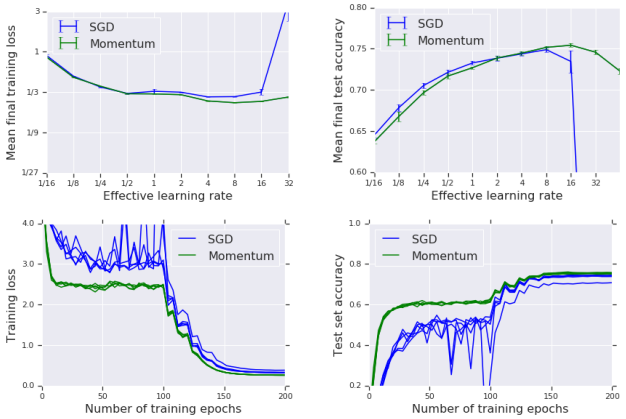


*Figure 5.* A 16-4 wide ResNet on CIFAR-100 with batch size 1024. In the top panel we provide the final training loss and test accuracy as a function of the effective learning rate. We perform 5 training runs for each learning rate, and provide the mean and standard deviation. In the bottom panel we plot training loss and test accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 16$.

learning rate is large, larger momentum coefficients perform best on both the training set and the test set. Consistent with the results provided in the main text, this suggests that if practictioners wish to optimize the efficiency of training, they should use SGD with small batch sizes and tune the learning rate, while if they wish to minimize the wall clock time, they should use Momentum, and it is worthwhile to tune both the learning rate and the momentum coefficient.

### C.2. 16-4 wide ResNet on CIFAR-100

We consider small batch training ($B = 128$) in figure 4 and large batch training ($B = 1024$) in figure 5. The model architecture, data augmentation and learning rate schedule match the implementation for CIFAR-10 described in the main text. Considering figure 4, in the small batch limit SGD and Momentum have near identical performance on the training set when the effective learning rate is small, while SGD slightly outperforms Momentum on the test set. However above a threshold $\epsilon_{\text{thres}} \approx 8$, Momentum outperforms SGD. Crucially, the optimal effective learning rate $\epsilon_{\text{opt}} \approx 2 < \epsilon_{\text{thres}}$, for which SGD slightly outperforms Momentum. We also provide training curves at $\epsilon_{\text{eff}} = 2$.

Considering figure 5, we observe that in the large batch limit Momentum slightly outperforms SGD on the training set once $\epsilon_{\text{eff}} \gtrsim 0.5$, although it does not begin to outperform SGD on the test set until $\epsilon_{\text{eff}} \gtrsim 8$. Crucially, the test set performance is optimal at $\epsilon_{\text{opt}} \approx 16$, at which value Momentum significantly outperforms SGD on both the training set and the test set. We note that the batch size increased by a factor of 8 between figure 4 and figure 5, while the optimal effective learning rate increased by the same factor, consistent with linear scaling. We also provide the training curves at $\epsilon_{\text{eff}} = 16$, from which it is clear that SGD is initially unstable while Momentum is stable throughout.

### C.3. Fully connected auto-encoder on MNIST

We now consider training a fully-connected auto-encoder on the MNIST dataset. Our network architecture is described by the sequence of layer widths $\{784, 1000, 500, 250, 30, 250, 500, 1000, 784\}$, where 784 denotes the input and output dimensions. Because of the bottleneck structure of the model, it is known to be a difficult problem to optimize and has often been used as an optimization benchmark (Sutskever et al., 2013; Martens &

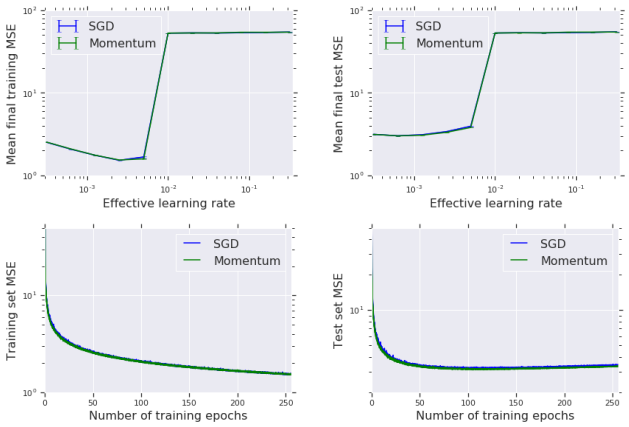


*Figure 6.* A fully connected autoencoder on MNIST with batch size 1. In the top panel we provide the final training mean-squared-error (MSE) and final test MSE as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. Both methods show similar performance. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 2.5 \times 10^{-3}$.

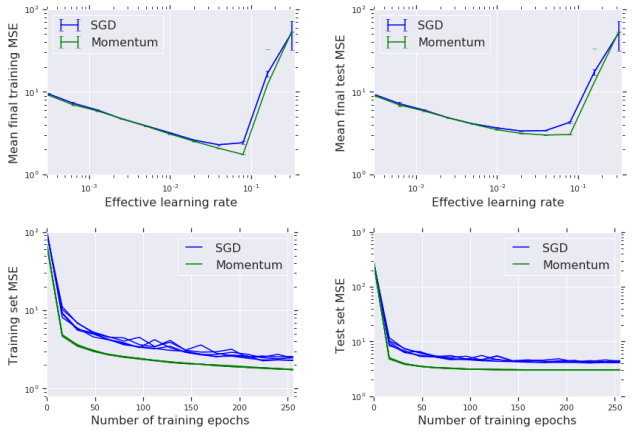


*Figure 7.* A fully connected autoencoder on MNIST with batch size 64. In the top panel we provide the final training mean-squared-error (MSE) and final test MSE as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. Both methods show near-identical performance when the effective learning rate is small, and Momentum outperforms SGD when the learning rate is large. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 0.08$.

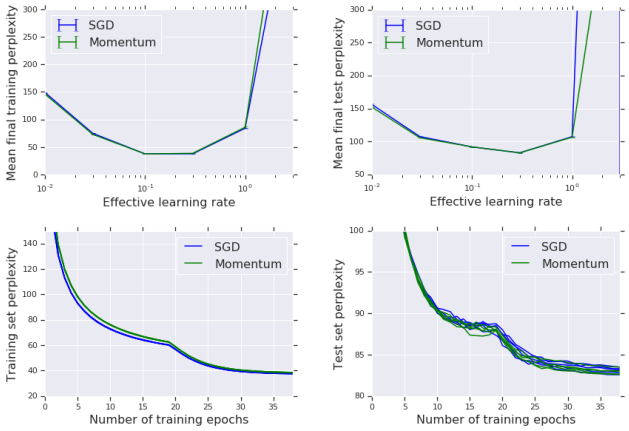


*Figure 8.* A word-level language model on PTB using an LSTM with the standard batch size 1. In the top panel we provide the final training perplexity and final test perplexity as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. Both methods show near-identical performance with small learning rates, although Momentum is more stable at higher learning rates above the optimum. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate, $\epsilon_{\text{eff}} = 0.3$.

Grosse, 2015; Kidambi et al., 2018). The $L_2$ regularization parameter was set at $10^{-5}$. We consider a very small batch ($B = 1$) in figure 6 and a batch size $B = 64$ in figure 7. We use SGD and Momentum with a constant learning rate, as in previous work (Sutskever et al., 2013). We also used a fixed momentum coefficient throughout training. The models are trained for 256 epochs. We tuned over $m = 0.9$ and $0.99$, and found little difference in performance. Thus, we show results for momentum $m = 0.9$ in figures 6 and 7.

Considering figure 6, we see that Momentum and SGD have near-identical performance for this very small batch size. Momentum begins to outperform SGD when the batch size is increased, and we start to see fairly significant benefits when the batch size is 64, as shown in figure 7. In figure 7, we see SGD and Momentum have similar performance when the learning rate is small, while Momentum starts outperforming SGD when the learning rate is large. Note that for this model, we start seeing the benefits of Momentum at a much smaller batch size than in the case of wide ResNets. This is likely due to the poor conditioning of the model.

### C.4. LSTM on Penn TreeBank

We now consider training an LSTM model for word-level language modeling on the Penn TreeBank dataset (PTB). We use the same setting as the medium-sized LSTM model from (Zaremba et al., 2014), which uses two layers with 650 units per layer. The parameters are initialized uniformly in $[-0.05, 0.05]$. We apply gradient clipping at 5, as well as dropout with probability 0.5 on the non-recurrent connections. We train the LSTM for 39 epochs using an unroll step of 35. We use a fixed learning rate for the first 20 epochs, and then decrease it by a factor of 0.8 after each epoch. We found this schedule to perform similarly to the original schedule in (Zaremba et al., 2014) when using the default batch size of 20, and to work better for larger batch sizes. The momentum coefficient was set to $m = 0.9$.

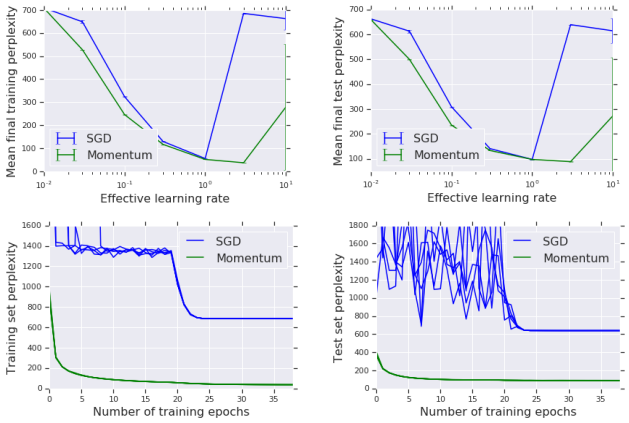


*Figure 9.* A word-level language model on PTB using an LSTM with a large batch size of 256. In the top panel we provide the final training perplexity and final test perplexity as a function of the effective learning rate. For each learning rate we perform 5 training runs with both SGD and Momentum and provide the mean and standard deviation. Unusually, Momentum outperforms SGD at both high and low learning rates, although the performance gap grows considerably for $\epsilon_{\text{eff}} \gtrsim 1$. In the bottom panel we plot the training loss and test set accuracy during training at the optimal effective learning rate (for training with momentum), $\epsilon_{\text{eff}} = 3.0$.

We consider small-batch training ($B = 1$) in figure 8 and large-batch training ($B = 256$) in figure 9. As before, we see near-identical performance for the small-batch case (figure 8), where the learning curves of SGD and Momentum almost completely match. We start seeing significant benefits of Momentum once we start making the batches larger, as shown in figure 9. For the optimal learning rate of Momentum in the large batch case ($\epsilon_{\text{eff}} = 3.0$), we see that SGD becomes unstable and fails to optimize the model properly.