6.883 Science of Deep Learning – Spring 2018 February 26, 2018

Lecture 6: Towards Understanding SGD

Lecturer: Constantinos Daskalakis *Scribes: Colin Grambow, Matt Groh, Madhav Kumar*
(Revised by Andrew Ilyas and Manolis Zampetakis)

1 Stochastic Gradient Descent and Generalization

A currently held belief in the field of non-convex optimization is that popular optimization methods in deep learning happen to choose local minima that achieve very small *generalization error*. Empirically, stochastic gradient descent (SGD) often results in solutions that generalize relatively well and even over-trained networks often still result in good performance. On the other hand, it has been observed that sufficiently large neural networks can easily fit random noise perfectly [1]. This implies that commonly used generalization bounds, such as VC-dimension and Rademacher complexity (see Lecture 3), are not sufficient theoretical tools to explain the generalization behavior of neural networks that we observe in practice. In this lecture, we will try to investigate some abstract properties of local minima that generalize well and we will present some attempts towards an explanation of the belief that SGD chooses this type of local minima and hence can benefit generalization.

2 Flatness of Local Minima

In the past lectures, we introduced the concept of flatness and its relation to SGD. We mentioned that one of the reasons for the superior performance of SGD is that it tends to converge to flat minima, which are schematically illustrated in Figure 1. The figure clearly demonstrates why flat minima, which are characterized by a low curvature, tend to generalize well: A small shift in the objective function, which is assumed to be representative of the differences between testing and training data, does not strongly affect the value of the objective function at the flat minimum, whereas the value at the sharp minimum changes drastically. Fewer bits are required for the description of a flat minimum, which implies that these generalize better [2].

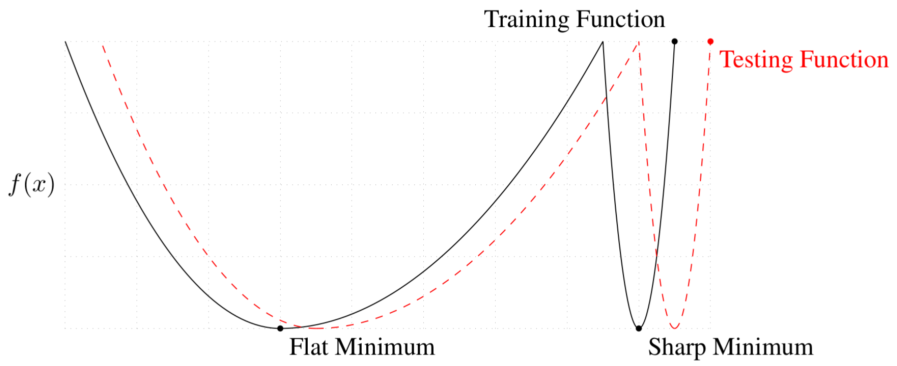


Figure 1: An illustration of flat and sharp minima [3].

It should be noted that flatness is not an easy notion to measure, especially in the high-dimensional spaces that neural networks live in, and the one-dimensional representation in Figure 1 certainly does not capture the complete notion of flatness. Another possible definition relates to the Hessian of the objective function. At a local minimum, the Hessian is positive definite, so a flat region would correspond to the space in which the eigenvalues of the Hessian are small. Essentially, a small determinant of the Hessian can be associated with flatness.

However, recent research has shown that the notion of flatness is intricately tied to the chosen parameterization of the model and cannot be used generally to explain generalization [4]. For example, in ReLU networks, the model can be reparameterized to yield sharp minima that generalize equally well as the flat ones. All of this implies that flatness is only a sufficient condition for generalization and that it cannot be examined separately from the algorithm used for the training of the network.

3 Batch Size vs. Flatness of Minima vs. Generalization

The iteration of the SGD algorithm with a mini-batch of size n is given by

$$w_{t+1} = w_t - \frac{\eta}{n} \sum_{x \in \mathcal{B}} \nabla \ell(x; w_t), \tag{1}$$

where w_t is the vector of *parameters* at steps t, $\ell(x; w_t)$ is the value of the *loss function* for the input x under the weights w_t, η is the *learning rate*. The set of samples in the mini-batch, $\mathcal{B}$, is a random subset of all samples. Changing the size of $\mathcal{B}$ strongly affects the flatness of the minimum solutions discovered by SGD and, in turn, their generalization performance. This was investigated in detail by Keskar *et al.* and will be reviewed here briefly [3].

Remark 1 (Details on the architecture of deep nets used in this lecture). All of the experiments conducted by Keskar *et al.* were done using *Adam* [5], a variant of SGD. In the following, we investigate their results for two different network architectures, F_2 and C_1. F_2 uses a 360-dimensional input layer, 7 ReLU layers with 512 units each and batch-normalization, and an output layer with 1973 units and softmax activation (Figure 2). F_2 is trained on the TIMIT data set [6], which is a speech recognition data set. C_1 is very similar to the often-used AlexNet architecture [7]. It first uses 2 sets of layers, each with 64 5×5 filters with stride 2 followed by max pooling, then 2 fully-connected layers with 384 and 192 units, respectively, and an output layer with 10 units (Figure 3). All units are ReLU and all layers use batch normalization. A dropout of 0.5 is used for the two fully-connected layers. C_1 is trained on the CIFAR-10 data set [8].

3.1 Superior generalization of small batch solutions

It is empirically observed that smaller batch sizes lead to flatter minima and to better generalization. The non-rigorous reasoning for this is that for small batches, the gradient approximations are more noisy, which allow the algorithm to escape from sharp minima and thus make it more likely to remain in flat regions of the objective. Not yet considering flatness, there is a clear gap in accuracy when comparing small batch (SB) and large batch (LB) solutions as shown in Figure 4. Evidently, the SB leads to better training and test accuracy, which implies that training is more successful with small batches (e.g., we avoid undesired regions of the objective function) and that the resulting solution generalizes better.

3.2 Flatness of small batch solutions

The results in Figure 4 already indicate that better generalization is obtained for smaller batches, but it is not yet clear if the flatness of the obtained minima are responsible for this. Denoting the SB solution by x_s^* and the LB solution by x_ℓ^*, the model solutions can be smoothly interpolated in a parametric plot shown in Figure 5 using the expression $f(\alpha x_\ell^* + (1-\alpha)x_s^*)$, where f is the loss and α is the interpolation parameter. The basin of attraction around $\alpha = 0$, which corresponds to the SB solution, is clearly much larger for both train and test data. When moving beyond the $\alpha = 1$ solution, accuracy decreases very rapidly. Consistent results were also obtained with a modified interpolation given by $f(\sin(\alpha\pi/2)x_\ell^* + \cos(\alpha\pi/2)x_s^*)$. Figure 5 is useful for confirming intuition about flatness and batch size, but there is no guarantee that the chosen measure of flatness is representative of the high-dimensional space of the network.

3.3 Effect of Varying Batch Size

As shown in Figure 7, the testing accuracy decreases after a certain point with increasing batch sizes while the sharpness of the solution minimum increases up to a certain point, after which it remains

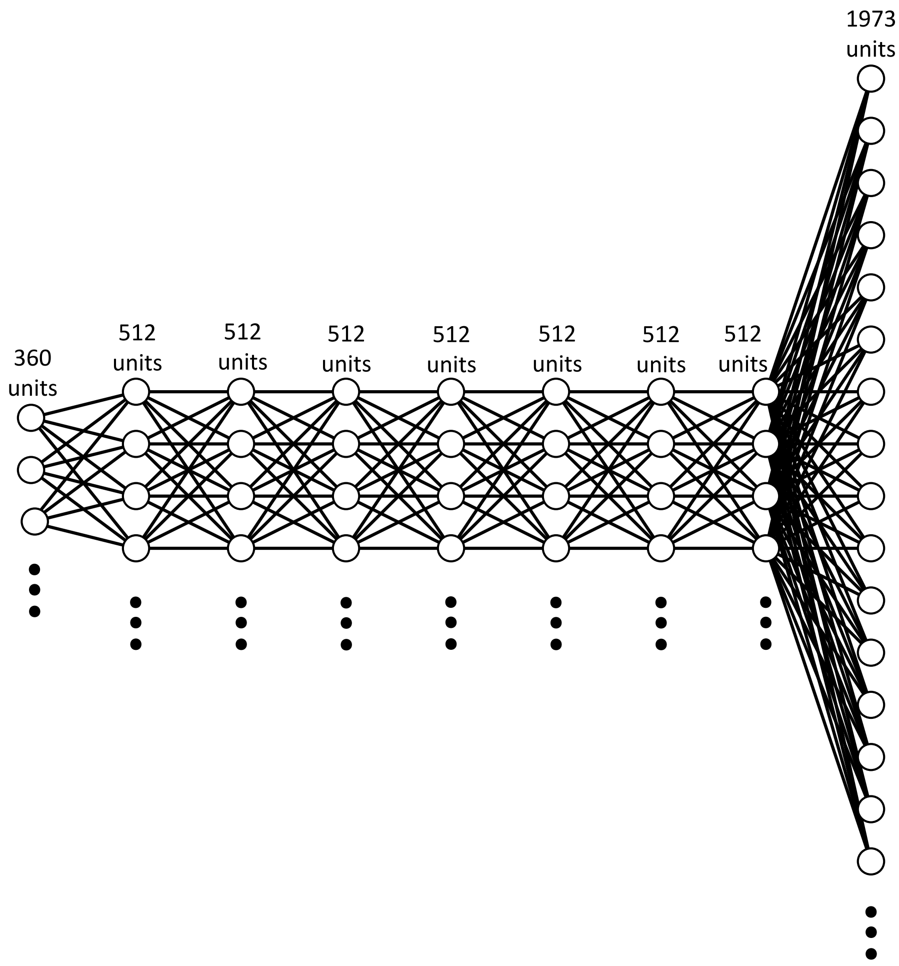


Figure 2: A schematic of network F_2. For more details see Remark 1.

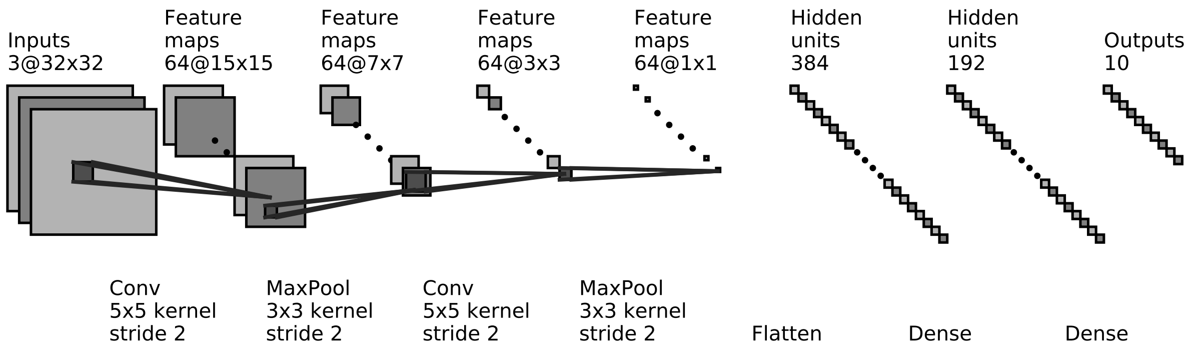


Figure 3: A schematic of network C_1. For more details see Remark 1.

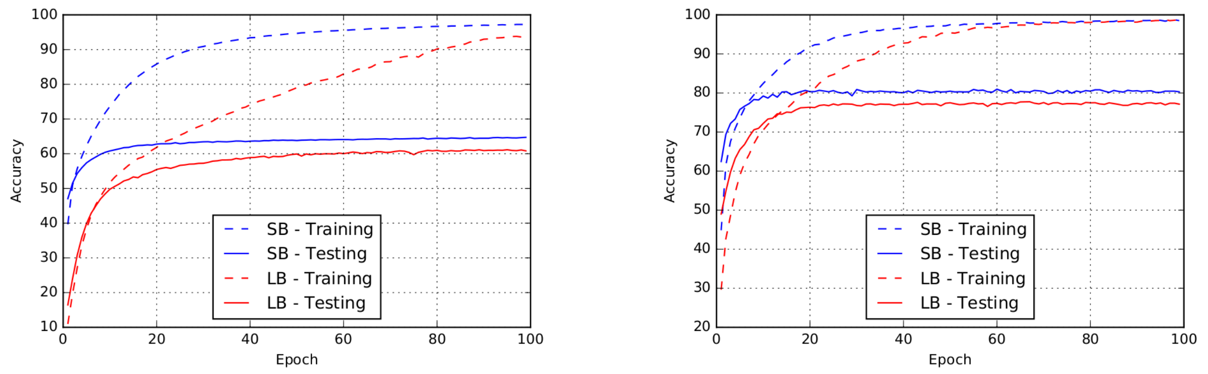


Figure 4: Accuracy gap between small and large batches for two different networks showing that small batch solutions clearly result in superior performance as measured by accuracy on the test set [3]. Left: Variation of accuracy with epoch for network F_2. Right: Network C_1. See Remark 1 for a description of these networks.

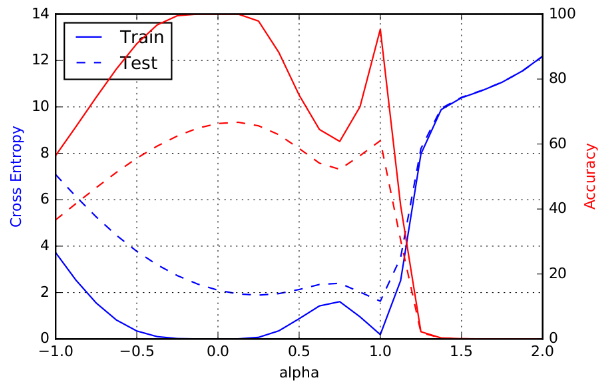


Figure 5: Interpolated loss for network F_2 ($\alpha = 0$ corresponds to small batch and $\alpha = 1$ to large batch) [3]. The left axis corresponds to the cross-entropy loss function used in the optimization and the right axis corresponds to the accuracy obtained on the data. Evidently, the region around the small batch solution is much flatter than the region around the large batch solution, which implies that the small batch solution will have superior generalization performance. See Remark 1 for a description of the network F_2.

relatively constant but quite noisy. Keskar *et al.* [3] argue that there exists a threshold batch size after which the model performance deteriorates, with the value of the threshold correspond to the point after which testing accuracy drops off significantly. Sharpness in this context can be formally defined as follows.

Definition 2. *Given $x \in \mathbb{R}^n$, $\epsilon > 0$, $A \in \mathbb{R}^{n \times p}$, the $(\mathcal{C}_\epsilon, A)$-sharpness of f at x is defined as:*

$$\phi_{x,f}(\epsilon, A) := \frac{(\max_{y \in \mathcal{C}_\epsilon} f(x + Ay)) - f(x)}{1 + f(x)} \cdot 100 \tag{2}$$

$\mathcal{C}_\epsilon$ is a box around the solution:

$$\mathcal{C}_\epsilon = \{z \in \mathbb{R}^p : -\epsilon(|(A^+x)_i| + 1) \leq z_i \leq \epsilon(|(A^+x)_i| + 1) \quad \forall i \in \{1, 2, \cdots, p\}\} \tag{3}$$

A^+ is the pseudo-inverse of a matrix $A \in \mathbb{R}^{n \times p}$, which is an $n \times p$ matrix with randomly generated columns. ϵ determines the size of the box.

Defined in this way, the sharpness measure should be relatively invariant to dimension and sparsity. To demonstrate this invariance, we examine four functions displayed in Figure 6 and we present the sharness value at each of their minima in Table 1. Based on the results in Figure 7, Keskar *et al.* suggest that there exists a threshold batch size after which generalization is poor as indicated by the increase in sharpness and decrease in testing accuracy. However, it is very likely that this threshold depends on other parameters of the algorithm, such as the learning rate.

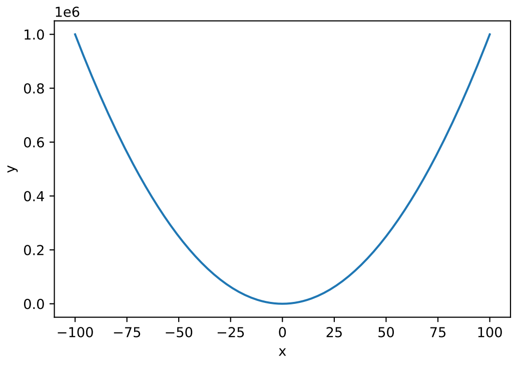
Table 1: Sharpness ($\phi_{x,f}(\epsilon, A)$) for functions with flat and sharp minima as calculated by Definition 2. As can be seen for the 2D functions, Definition 2 treats minima as sharp, even if one of the directions is relatively flat. In fact, if ϵ is sufficiently large, the second 2D function will be sharper because the slope is much greater in the y direction after a certain point.

Dimensions	A	ϵ	Function Type	Fig.	$\phi_{x,f}(\epsilon, A)$ at Min.
1	Identity	0.0010	Sharp	6a	1.0e-02
1	Identity	0.0010	Flat	6b	1.0e-22
1	Identity	0.0005	Sharp	6a	2.5e-03
1	Identity	0.0005	Flat	6b	3.9e-25
1	Random	0.0010	Sharp	6a	2.8e-03
1	Random	0.0010	Flat	6b	6.5e-25
1	Random	0.0005	Sharp	6a	7.1e-04
1	Random	0.0005	Flat	6b	2.5e-27
2	Identity	0.0010	Sharp in X & Y	6c	2.0e-02
2	Identity	0.0010	Sharp in X, Flat in Y	6d	1.0e-02
2	Identity	0.0005	Sharp in X & Y	6c	5.0e-03
2	Identity	0.0005	Sharp in X, Flat in Y	6d	2.5e-03
2	Random	0.0010	Sharp in X & Y	6c	2.1e-02
2	Random	0.0010	Sharp in X, Flat in Y	6d	1.9e-02
2	Random	0.0005	Sharp in X & Y	6c	5.3e-03
2	Random	0.0005	Sharp in X, Flat in Y	6d	4.8e-03

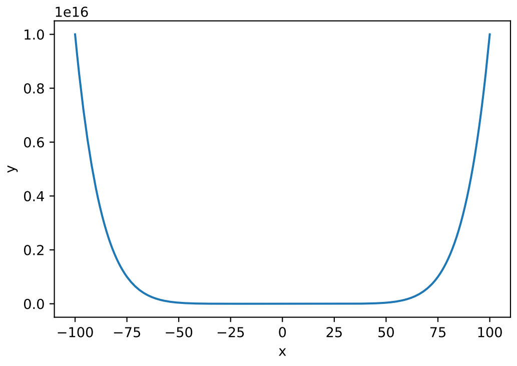
Unfortunately, large batch sizes parallelize much better than smaller ones, so it would be ideal if large batch sizes could be used while retaining sufficient model accuracy. In the next section, we investigate how this can be achieved by tuning the learning rate.

4 Batch Size vs. Learning vs. Training Time vs. Generalization

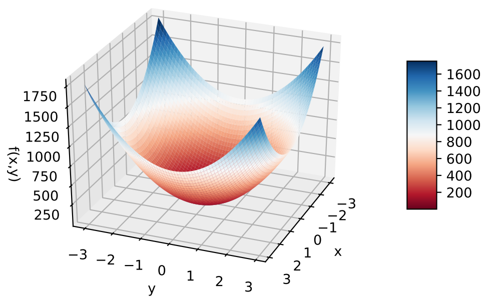
Deeper neural networks and bigger data sets result in longer model training times, which can slow research and development. Larger mini-batch sizes can train quicker because a single step with mini-



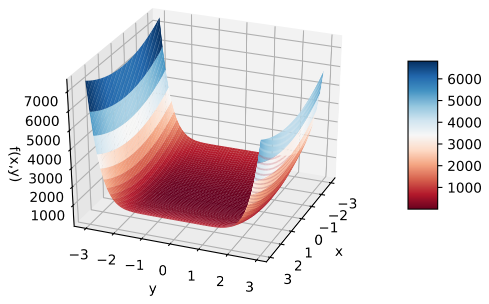
(a) $f(x) = 100x^2$



(b) $f(x) = x^8$



(c) $f(x, y) = 100(x^2 + y^2)$



(d) $f(x, y) = 100x^2 + y^8$

Figure 6: Functions with flat and sharp minima. The first row corresponds to one-dimensional functions, $f(x) = 100x^2$ with a relatively sharp minimum and $f(x) = x^8$ with a relatively flat minimum. Both functions have minima at 0. The second row corresponds to two-dimensional functions, $f(x, y) = 100(x^2 + y^2)$ with a relatively sharp minimum in both x and y and $f(x, y) = 100x^2 + y^8$ with a relatively sharp minimum in x and a relatively flat minimum in y. Both functions have minima at (0,0).

batch of size kn is faster than k steps with mini-batch of size n. Distributed synchronous SGD offers a potential solution to the slow training problem by dividing SGD mini-batches over a pool of parallel GPUs [9]. In order to speed up training, the per-GPU workload must be large, which requires growth in SGD mini-batch sizes. In order to use large mini-batches in place of small mini-batches while maintaining training and generalization accuracy, we can employ a linear scaling rule of thumb to the learning rate [9].

Linear Scaling Rule*: when the mini-batch size is multiplied by k, multiply the learning rate by k.*

This linear scaling rule means we can scale to multiple GPUs without reducing the per-GPU workload or model accuracy. Consider a network at iteration t with weights w_t, and a sequence of k mini-batches $\mathcal{B}_j$ for $0 \leq j < k$, each of size n. Then, we can compare the effect of executing k SGD iterations with small mini-batches $\mathcal{B}_j$ and learning rate η vs. a single iteration with a large mini-batch $\bigcup_j \mathcal{B}_j$ of size kn and learning rate $\hat{\eta}$. Then, after k iterations of SGD with learning rate η and a mini-batch size of n we have:

$$w_{t+k} = w_t - \eta \frac{1}{n} \sum_{j<k} \sum_{x \in \mathcal{B}_j} \nabla \ell(x; w_{t+j}) \tag{4}$$

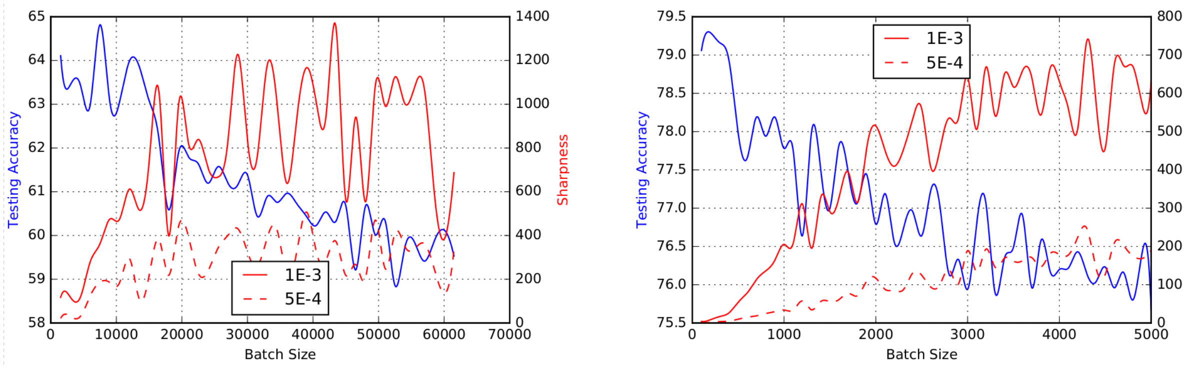


Figure 7: Testing accuracy and sharpness vs. batch size. Sharpness is shown for two values of ϵ [3]. The batch size on the x-axis was used for training each network for 100 epochs. The left y-axis corresponds to the testing accuracy at the final iterate and the right y-axis to the sharpness at that point as evaluated by Definition 2. Left: Network F_2. Right: Network C_1. See Remark 1 for a description of the networks.

where $\ell(x; w_{t+j})$ is the loss function for the input x under the weights w_{t+j}. Likewise, after a single step with a large mini-batch $\bigcup_j \mathcal{B}_j$ of size kn and learning rate $\hat{\eta}$, we have:

$$\hat{w}_{t+1} = w_t - \hat{\eta}\frac{1}{kn}\sum_{j<k}\sum_{x\in\mathcal{B}_j} \nabla\ell(x; w_t) \tag{5}$$

The key intuition here is that these two updates can be similar if we set $\hat{\eta} = kn$. Evidently, this is only the case if $\nabla\ell(x; w_t) \approx \nabla\ell(x; w_{t+j})$, which holds approximately true if the parameters do not change too drastically. This also motivates a warm start on the learning rates: Starting with a small learning rate and gradually increasing it ensures that the parameter changes are not too drastic.

Figures 8 and 9 present the results of SGD across mini-batch size; larger mini-batches up to 8k images have both the same out of sample accuracy and training curves as smaller mini-batches [9]. This empirical finding suggests that the large mini-batch approximation may be valid in large-scale, real-world data.

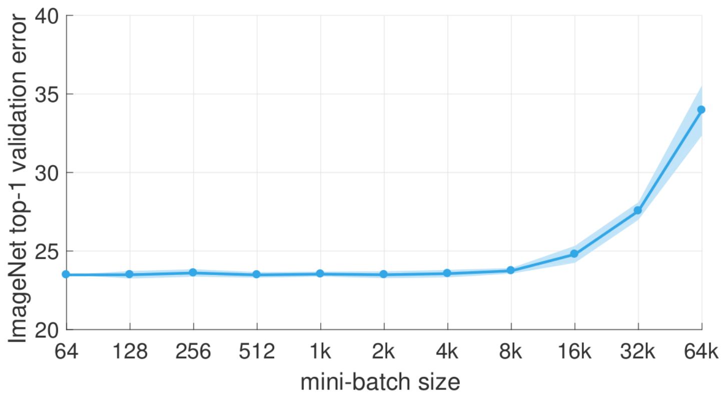


Figure 8: ImageNet top-1 validation error vs. mini-batch size [9]. Accuracy of models is invariant to mini-batch size up to 8k (8192) images, which allowed Goyal *et al.* 2017 to train an accurate 8k mini-batch ResNet-50 model on the ImageNet dataset in 1 hour using 256 GPUs.

In order to ensure that small and large mini-batch models produce similar results, it is important to avoid drastic changes to hyperparameters when changing the mini-batch size. In particular, Goyal *et al.* offer 4 cautionary remarks: (1) Scaling the cross-entropy loss is not equivalent to scaling the learning rate; (2) Apply momentum correction after changing learning rate if using R-CNN; (3) Normalize the

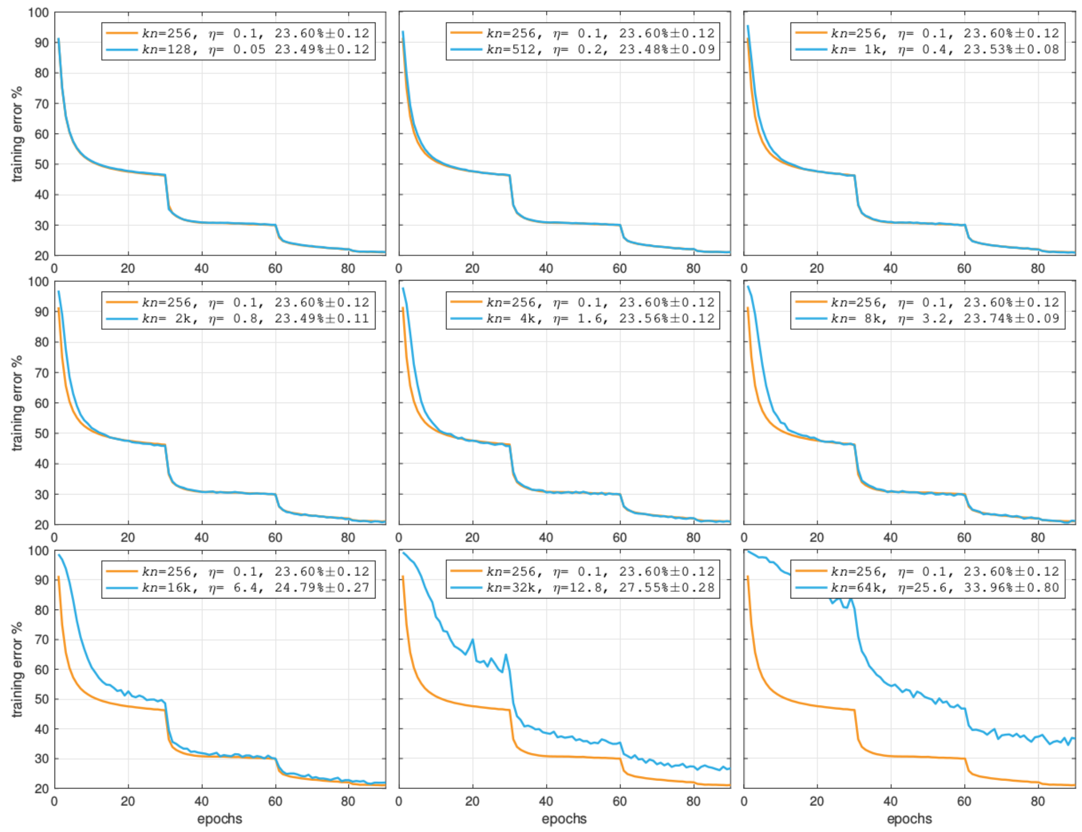


Figure 9: Training error vs. mini-batch size. Training curves for 256 mini-batches closely match larger mini-batches up through 8k mini-batches. Validation error, mini-batch size kn, and reference learning rate η are shown in the legend. It is important to note that these models are trained with gradual warmup where the learning rate starts at η and increases linearly such that it reaches $\hat{\eta} = kn$ after 5 epochs.

per-worker loss by total mini-batch size kn, not per-worker size n; (4) Use a single random shuffling of the training data (per epoch) that is divided amongst all k workers. In the next section, we will explore potential theoretical explanations for these results.

5 Using Stochastic Differential Equations

5.1 True loss vs. Mini-batch loss

Goyal *et al.* hint towards the possibility of an optimal batch size that can help speed-up training while maintaining test set accuracy. Smith *et al.* show that this is indeed the case by approximating SGD as a continuous time Stochastic Differential Equation (SDE). They derive a “noise scale” parameter that quantifies the underlying random fluctuations in the dynamics of the SGD.

To understand this process, consider the “true” loss function, i.e., the loss calculated on the entire dataset:

$$\ell(w) = \sum_{i=1}^{N} \ell(x_i, y_i; w) \tag{6}$$

Further, consider the loss for a mini-batch $\mathcal{B}$ of size B:

$$\hat{\ell}(w) = \frac{N}{B}\sum_{i\in\mathcal{B}} \ell(x_i, y_i; w) \tag{7}$$

With equations (6) and (7), the gradient update of SGD can be written as a discrete-time process as shown below [10]:

$$\Delta_w = -\frac{\eta}{N}\Big(\nabla_w \ell + (\nabla_w \hat{\ell} - \nabla_w \ell)\Big) \tag{8}$$

where

- η is the learning rate,
- N is the training set size,
- $\nabla_w \ell$ is the true gradient at w_t,
- $\nabla_w \hat{\ell}$ is the noisy gradient approximation at w_t,
- $\nabla_w \hat{\ell} - \nabla_w \ell$ is the fluctuation about the true gradient.

We note that each mini-batch is chosen with replacement to facilitate the theoretical analysis. The mini-batch loss has the following properties:

$$\mathbb{E}\left[\hat{\ell}(w)\right] = \ell(w) \tag{9}$$

$$\mathbb{E}\left[\nabla_w \hat{\ell}(w)\right] = \mathbb{E}\left[\nabla_w \ell(w)\right] \tag{10}$$

Assuming the fluctuation around the true loss, labeled as ψ, to be Gaussian random noise, it can be defined as:

$$\psi = \nabla_w \hat{\ell} - \nabla_w \ell \tag{11}$$

The credibility of this approximation rests on the crucial assumption of the central limit theorem (CLT), according to which the random error term will tend towards Gaussian noise as $N \to \infty$, $B \to \infty$, and $B \ll N$. Smith *et al.* note that while B and N are both finite, CLT is fairly robust in practice. In the analysis below, we continue with this critical, but questionable, assumption.

Given equation (10), the expected value of the fluctuation is $\mathbb{E}[\psi] = 0$ and the variance is

$$\mathrm{Cov}(\psi) = \mathbb{E}\left[\psi\psi^T\right] \tag{12}$$

If we re-write ψ as

$$\psi = \frac{N}{B}\left[\sum_{i\in\mathcal{B}}\Big(\nabla_w \ell(x_i, y_i; w) - \frac{1}{N}\sum_{k}\nabla_w \ell(x_i, y_i; w)\Big)\right] \tag{13}$$

the $\mathrm{Cov}(\psi)$ term is given by

$$\mathrm{Cov}(\psi) = N \cdot \left(\frac{N}{B} - 1\right) \cdot \mathbf{F}(w) \tag{14}$$

where $\mathbf{F}(w)$ is a matrix of the gradient covariances, which are a function of the current parameter values. It measures the variation between the gradient calculated at a random point with weights w and the true gradient. Under the assumption, $N \gg B$, we can approximate the variance as:

$$\mathrm{Cov}(\psi) \approx \frac{N^2}{B} \cdot \mathbf{F}(w) \tag{15}$$

Hence, we can summarize the fluctuations as a Gaussian variable, such that

$$\psi \sim \mathcal{N}\Big(0, \frac{N^2}{B}\mathbf{F}(w)\Big) \tag{16}$$

5.2 Re-casting as an SDE

We can re-cast the gradient update from equation (8) as a discrete update of a stochastic differential equation:

$$\nabla_t(w) = -\nabla_w \ell + \varepsilon(t) \tag{17}$$

where t is a continuous variable, $\varepsilon(t)$ is white noise with $\mathbb{E}[\varepsilon(t)] = 0$ and $\mathbb{E}[\varepsilon(t)\varepsilon(t')^T] = g \cdot \mathbf{F}(w) \cdot \delta(t - t')$. It is the constant g that controls the random fluctuations of the underlying dynamics. The noise is also assumed to be uncorrelated at two different time steps.

The gradient update can be computed by integrating the continuous process over N steps as shown below:

$$\Delta w = \int_0^{\eta/N} \nabla_t w \; dt \tag{18}$$

$$= \int_0^{\eta/N} \nabla_w \ell \; dt + \int_0^{\eta/N} \varepsilon(t) \; dt \tag{19}$$

In the first integral of the above equation, w is not going to change much and hence can be approximated as $-\eta/N$. The second integral is the integral of white noise. We can write its variance as:

$$\mathbb{E}\left[\left(\int_0^{\eta/N} \varepsilon(t) \; dt\right)\left(\int_0^{\eta/N} \varepsilon(t') \; dt'\right)\right] \tag{20}$$

$$= \int_0^{\eta/N} dt \int_0^{\eta/N} dt' \;\; \mathbb{E}\left[\varepsilon(t)\varepsilon(t')^T\right] \tag{21}$$

Using the variance of the error term defined in equation (18), we can write

$$= g \cdot \mathbf{F}(w) \cdot \int_0^{\eta/N} dt \int_0^{\eta/N} dt' \;\; \delta(t - t') \tag{22}$$

$$= \frac{\eta}{N} \cdot g \cdot \mathbf{F}(w) \tag{23}$$

Equating the variance obtained here to the one obtained in equation (14), we obtain the "noise scale", g, as:

$$g = \eta \cdot \left(\frac{N}{B} - 1\right) \approx \eta \frac{N}{B} \tag{24}$$

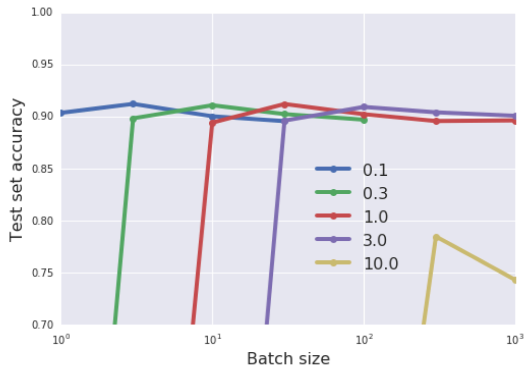
We can see from equation (24) that the "noise scale" falls as the batch size, B, increases. Hence, given a set of fixed parameters for the network, there must be an optimal batch size, B_{opt}, as claimed by Smith *et al.*. The authors run some experiments to provide evidence for this claim. We present their experiments 10a and 11a and the evidence for optimal batch size in figures 10b and 11b.

5.3 Isotropic gradient noise

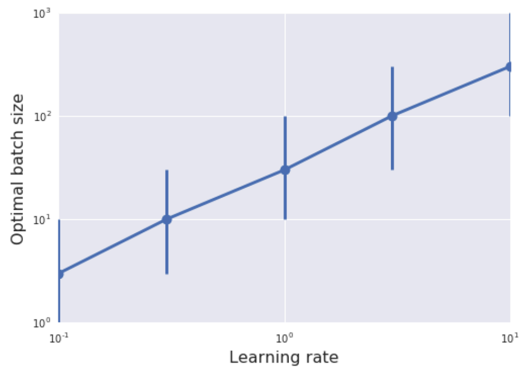
Jastrzebski *et al.* 2017 also approximate SGD as a continuous SDE but go one step further and make a stronger assumption that w is isotropic. More specifically, they assume that $\mathbf{F}(w) = \sigma^2 \mathbf{I}$. This helps them get to the stationary equilibrium of the underlying stochastic process. The equilibrium distribution of the SDE is given by

$$P(w) = P_0 \exp\left(\frac{-2\ell(w)}{\frac{\eta}{B}\sigma^2}\right) \tag{25}$$

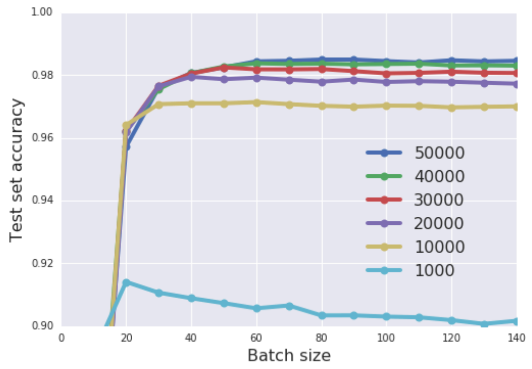
where P_0 is a normalization constant.



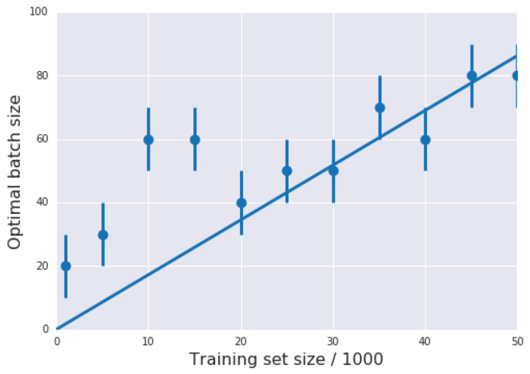
(a) Accuracy of the model on the test set as a function of batch size. Different lines represent different learning rates. As we increase the learning rate, the performance peaks with larger batch sizes. However, the overall performance falls once $\eta \gtrsim 3$.



(b) There is a linear relationship between optimal batch size and the learning rate.



(a) Accuracy of the model on the test set as a function of batch size. Different lines represent different training set sizes. The model's performance peaks at larger batch sizes as we increase the training set size. Note: Each curve is averaged over five experiments to reduce noise.



(b) With, $N \gtrsim 20000$, the optimal batch size is proportional to the size of the training data.

Further, suppose that $\ell(w)$ has well separated minima. Consider one of these minima and call it A. Let the loss at A be ℓ_A and the Hessian at A be H_A. We know that H_A is positive semi-definite. Additionally, the smaller the determinant of H_A, the flatter the minima would be. The (unnormalized) probability of ending up in the bowl of such a minima A is proportional to:

$$P(A) \propto \frac{1}{\sqrt{\det(H_A)}} \exp\left(\frac{-\ell(w)}{\frac{\eta}{B}\sigma^2}\right) \tag{26}$$

Both of the last two expressions depend on the ratio of the learning rate to the mini-batch size, i.e., η/B.

References

[1] Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning requires rethinking generalization. arXiv: 1611.03530 (2016).

[2] Hochreiter, S. & Schmidhuber, J. Flat Minima. *Neural Comput.* **9,** 1–42 (1997).

[3] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima* in *ICLR* (2016). arXiv: 1609.04836.

[4] Dinh, L., Pascanu, R., Bengio, S. & Bengio, Y. Sharp Minima Can Generalize For Deep Nets. arXiv: 1703.04933 (2017).

[5] Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. arXiv: 1412.6980 (2014).

[6] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., Dahlgren, N. L. & Zue, V. *TIMIT Acoustic-Phonetic Continuous Speech Corpus* in *Linguistic Data Consortium, Philadelphia* **33** (1993).

[7] Krizhevsky, A., Sutskever, I. & Hinton, G. E. in *Advances in Neural Information Processing Systems 25* (eds Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q.) 1097–1105 (Curran Associates, Inc., 2012). http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[8] Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images (2009).

[9] Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. & He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv: 1706.02677 (2017).

[10] Smith, S. L., Kindermans, P. & Le, Q. V. Don't Decay the Learning Rate, Increase the Batch Size. arXiv: 1711.00489 (2017).

[11] Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y. & Storkey, A. Three Factors Influencing Minima in SGD. arXiv: 1711.04623 (2017).