DON'T DECAY THE LEARNING RATE, INCREASE THE BATCH SIZE

Presented by:
Zuber Khan
Debarshi Dasgupta

Introduction

- ➔ It's a common practice to decay learning rate for better optimization/generalization.
- ➔ The same learning curve can be obtained on both training and test sets by increasing the batch size during training instead.
- ➔ It reaches equivalent test accuracies with shorter training time and less parameter updates.
- ➔ ResNet-50 was trained on ImageNet to 76.1% validation accuracy in under 30 minutes.

Introduction

-> We can further reduce the number of parameter updates by increasing the learning rate and scaling the batch size proportionally to it.

-> But when batch size increases the test accuracy falls by a margin.

-> The scale of random fluctuations in the SGD dynamics $g = \epsilon(\frac{N}{B} - 1)$

-> There is an optimum g,thus optimum B which maximizes test accuracy.

-> One can also increase the momentum coefficient and scale B $\propto$ 1/(1 – m) to reduce number of parameter updates, although this slightly reduces the test accuracy.

$$\sum_{i=1}^{\infty} \epsilon_i = \infty,$$

$$\sum_{i=1}^{\infty} \epsilon_i^2 < \infty.$$

->equation 1 ensures that the minimum is reached, no matter how far away the parameters are initialized,
->equation 2 ensures that the learning rate decays sufficiently quickly for converging to the minimum, rather than bouncing around it due to gradient noise.
However equation 2 is valid when batch size is constant.
When batch size is varying sgd is interpreted as the integration of the differential equation shown below

$$\frac{d\omega}{dt} = -\frac{dC}{d\omega} + \eta(t)$$

->C represents the cost function (summed over all training examples),
->ω represents the parameters.
->η(t) represents Gaussian random noise, which models the consequences of estimating the gradient on a mini-batch.

$$g = \epsilon\left(\frac{N}{B} - 1\right)$$

->eps the learning rate, N the training set size and B the batch size.

SIMULATED ANNEALING AND THE GENERALIZATION GAP

->It is now increasingly accepted that small batch training often generalizes better to the test set than large batch training. (**Generalization Gap)**

-> Keskar et al. (2016). Smith & Le (2017) observed an optimal batch size Bopt which maximized the test set accuracy at constant learning rate.They argued that this optimal batch size arises when the noise scale g prop eN/B is also optimal.

->It has been proposed that noise helps SGD escape “sharp minima” which generalize poorly.

-> The initial noisy optimization phase allows to explore a larger fraction of the parameter space without becoming trapped in local minima. Once a promising region of parameter space is located, reduce the noise to fine-tune the parameters.

-> That is why conventional learning rate decay schedules like square roots or exponential decay have become less popular in deep learning in recent years. Increasingly, researchers favor sharper decay schedules like cosine decay or *step-function drops.*

Researchers prefer SGD with momentum to vanilla SGD.

$$g = \frac{\epsilon}{1-m}\left(\frac{N}{B}-1\right) \approx \frac{\epsilon N}{B(1-m)}$$

->This reduced to the noise scale of vanilla SGD when the momentum coefficient m → 0.
->Intuitively, eps(eff) =eps /(1 – m) is the effective learning rate.
->It was proposed to reduce the number of parameter updates required to train a model by increasing the learning rate and momentum coefficient, while simultaneously scaling B ∝ eps/(1 – m).
->It is found that increasing the learning rate and scaling B ∝ eps performs well. However increasing the momentum coefficient while scaling B ∝ 1/(1–m) slightly reduces the test accuracy.

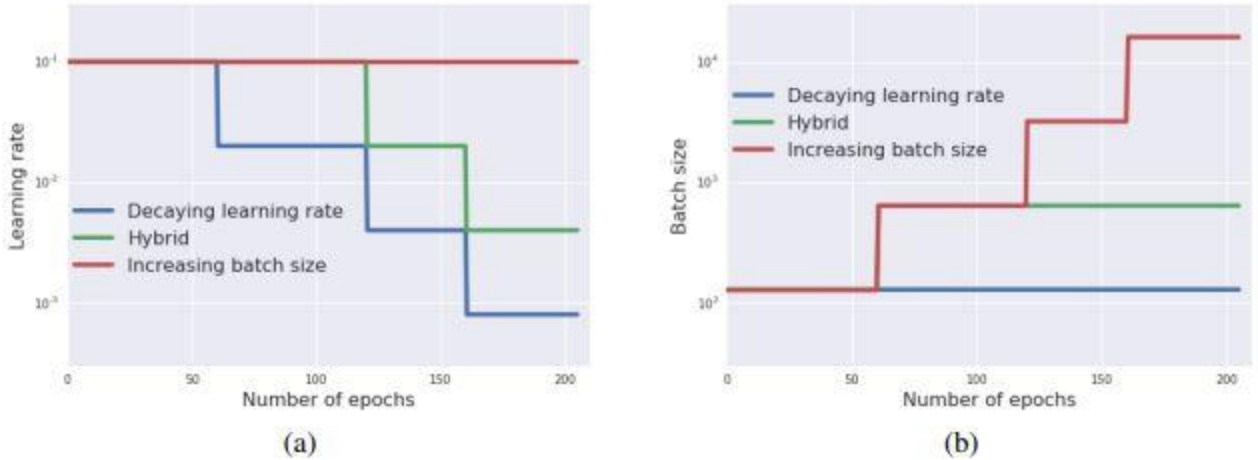
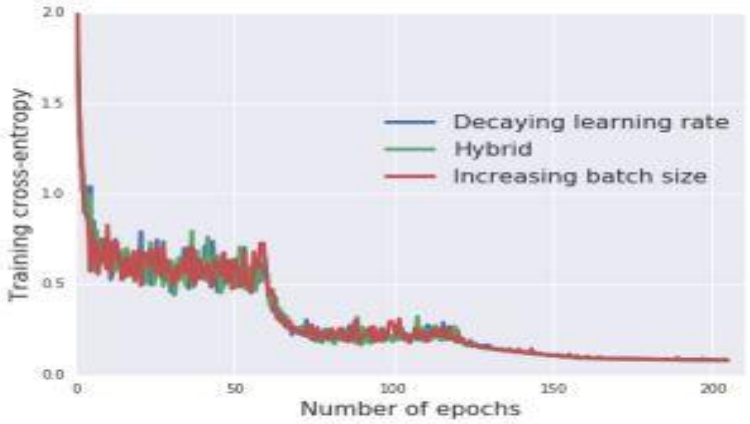
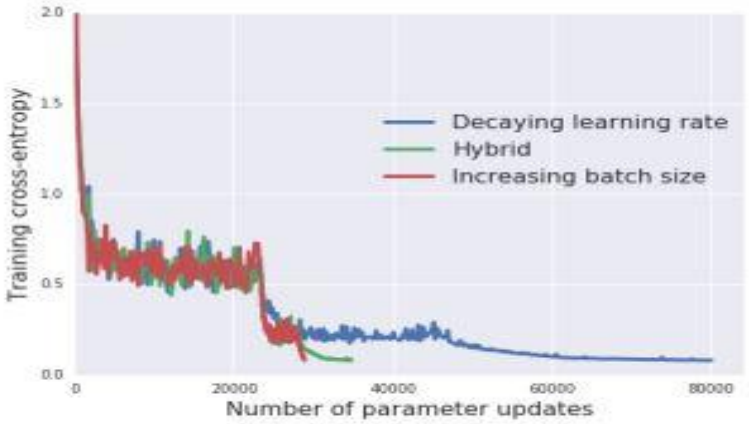


Figure 1: Schedules for the learning rate (a) and batch size (b), as a function of training epochs.

Decaying learning rate: the batch size is constant, while learning rate decays by a factor of 5 at each step.
Hybrid: holds the learning rate constant at the first step, instead increasing the batch size by a factor of 5. After this first step, the batch size is constant and the learning rate decays by a factor of 5 at each subsequent step.
Increasing batch size: learning rate constant throughout training, and batch size increased by a factor of 5 at every step.





The three learning curves are identical, but increasing the batch size reduces the number of parameter updates required.

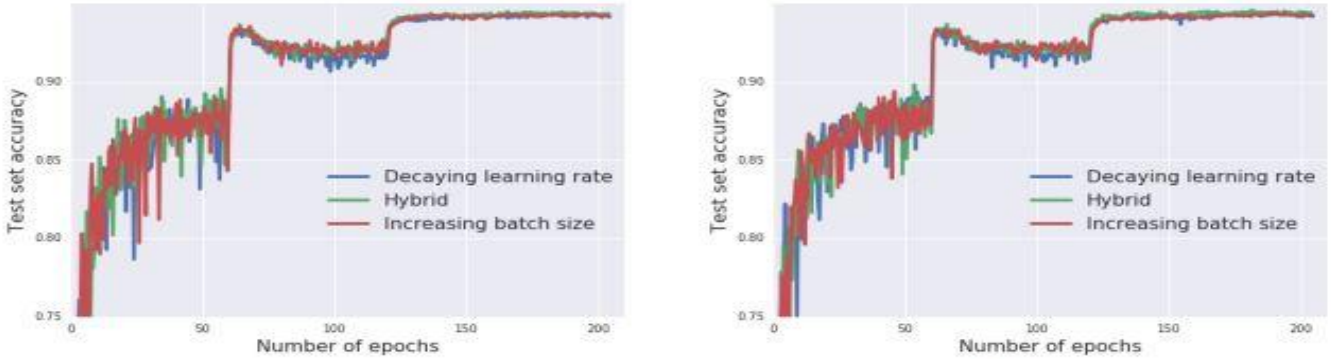


Figure 3: Wide ResNet on CIFAR10. Test accuracy during training, for SGD with momentum (a), and Nesterov momentum (b). In both cases, all three schedules track each other extremely closely.

->Figure 3a shows Test set accuracy, as a function of the number of epochs .
->The three schedules are almost identical.
->It can be concluded that all of the benefits of decaying the learning rate can be achieved by instead increasing the batch size.

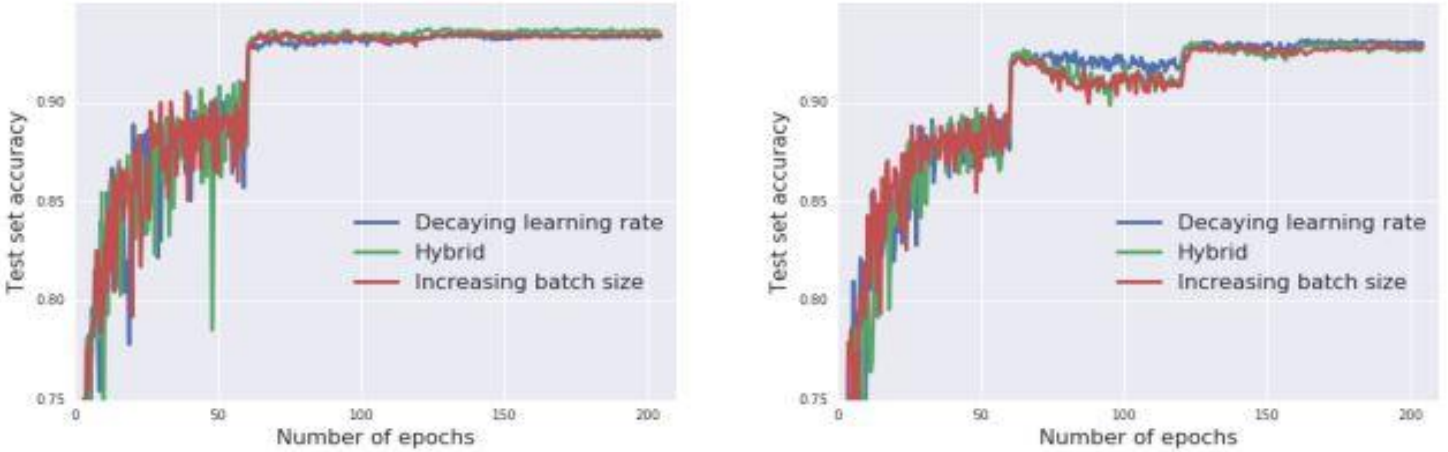


Figure 4: Wide ResNet on CIFAR10. The test set accuracy during training, for vanilla SGD (a) and Adam (b). Once again, all three schedules result in equivalent test set performance.

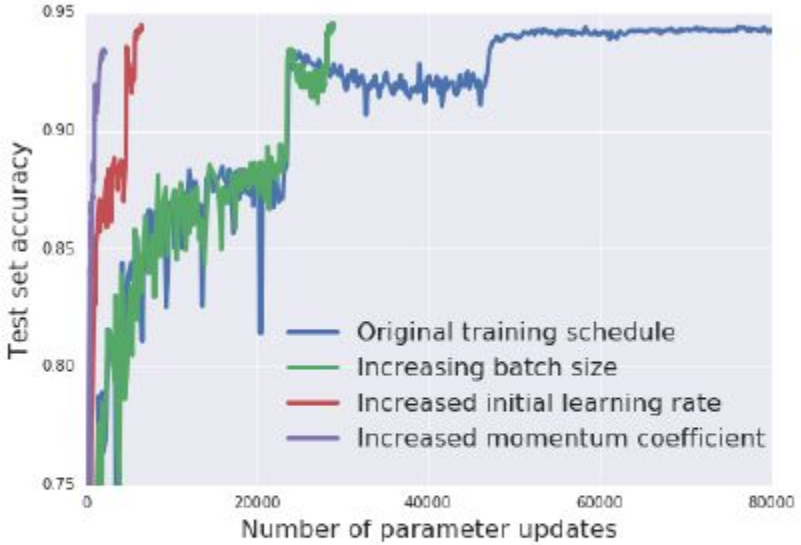


Figure 5: Wide ResNet on CIFAR10. Test accuracy as a function of the number of parameter updates. "Increasing batch size" replaces learning rate decay by batch size increases. "Increased initial learning rate" additionally increases the initial learning rate from 0.1 to 0.5. Finally "Increased momentum coefficient" also increases the momentum coefficient from 0.9 to 0.98.

“**Original training schedule**” using an initial learning rate of 0.1 which decays by a factor of 5 at each step, a momentum coefficient of 0.9, and a batch size of 128.
“**Increasing batch size**” also uses a learning rate of 0.1, initial batch size of 128 and momentum coefficient of 0.9, but the batch size increases by a factor of 5 at each step.
“**Increased initial learning rate**” also uses increasing batch sizes during training, but additionally uses an initial learning rate of 0.5 and an initial batch size of 640.
“**Increased momentum coefficient**” combines increasing batch sizes during training and the increased initial learning rate of 0.5, with an increased momentum coefficient of 0.98, and an initial batch size of 3200

Original training schedule requires ~80000 updates, and reaches a final test accuracy of 94.3%
Increasing batch size requires ~29000 updates, reaching a final accuracy of 94.4%.
Increased initial learning rate requires under 6500 updates, reaching a final accuracy of 94.5%.
Increased momentum coefficient” requires less than 2500 parameter updates, but reaches a lower test accuracy of 93.3%.

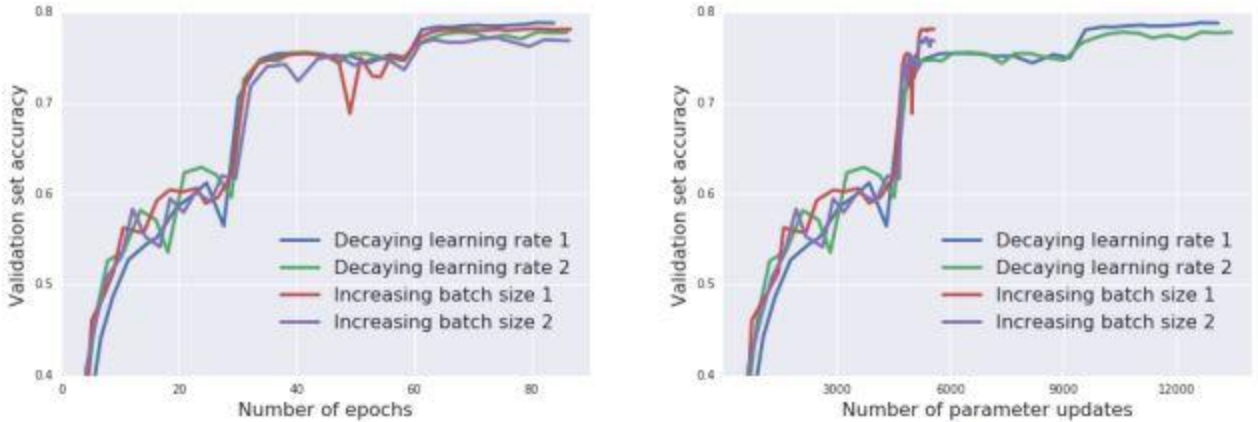


Figure 6: Inception-ResNet-V2 on ImageNet. Increasing the batch size during training achieves similar results to decaying the learning rate, but it reduces the number of parameter updates from just over 14000 to below 6000. We run each experiment twice to illustrate the variance.

TRAINING IMAGENET IN 2500 PARAMETER UPDATES

->The final accuracies of the two “Decaying learning rate” runs are 78.7% and 77.8%, while the final accuracy of the two “Increasing batch size” runs are 78.1% and 76.8%.
->Increasing the batch size reduces the number of parameter updates during training from just over 14000 to below 6000.

“Momentum 0.9” uses an initial batch size of 8192,
“Momentum 0.95” uses an initial batch size of 16384
“Momentum 0.975” used an initial batch size of 32768.
“Momentum 0.9” achieves a final accuracy of 78.8% in just under 6000 updates.
“Momentum 0.95”, achieved final accuracies of 78.1% and 77.8% in under 3500 updates.
“Momentum 0.975” achieved final accuracies of 77.5% and 76.8% in under 2500 updates.

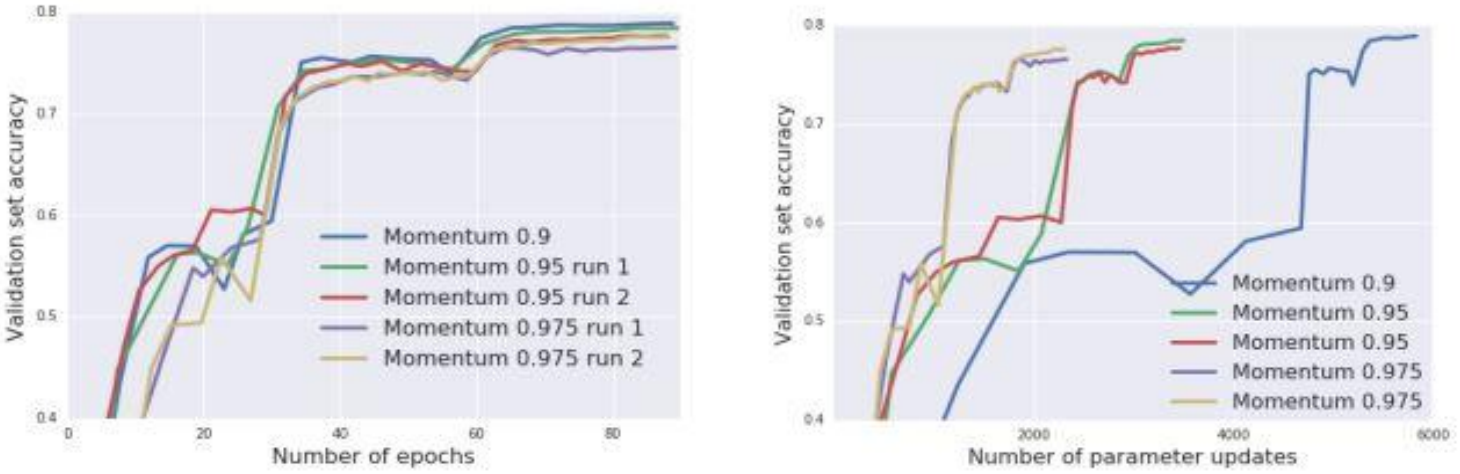


Figure 7: Inception-ResNet-V2 on ImageNet. Increasing the momentum parameter reduces the number of parameter updates required, but it also leads to a small drop in final test accuracy.

TRAINING IMAGENET IN 30 MINUTES

->ResNet-50 was for 90 epochs to 76.1% validation set accuracy in under 45 minutes, utilising batches of 8192 images.
->Next the batch size after the first 30 epochs to 16384 images, and the same validation accuracy of 76.1% was achieved in under 30 minutes
->The last 60 epochs and the first 30 epochs both take just under 15 minutes.

->Doubling the initial learning rate and using batches of 16384 images throughout training achieves a lower validation set accuracy of 75.0% in 22 minutes, demonstrating that increasing the batch size during training is crucial to the performance gains above.

Conclusion

->The benefits of decaying the learning rate can be obtained by instead increasing the batch size during training.
->By this approach the number of parameter updates required to train a model was significantly reduced.
->The batch size B was further increased by increasing the learning rate and momentum parameter m, while scaling $B \propto eps/(1-m)$. Combining these strategies, Inception-ResNet-V2 was trained on ImageNet to 77% validation accuracy in under 2500 parameter updates, using batches of 65536 images.
->Most strikingly, this was achieved without any hyper-parameter tuning, since the scaling rules enables to directly convert existing hyper-parameter choices from the literature for large batch training.

Thank You