facebook Artificial Intelligence Research

ON THE INEFFECTIVENESS OF VARIANCE REDUCED OPTIMIZATION FOR DEEP LEARNING

Aaron Defazio
Léon Bottou

The application of stochastic variance reduction to optimization has shown remarkable recent theoretical and practical success. The applicability of these techniques to the hard non-convex optimization problems encountered during training of modern deep neural networks is an open problem. We show that naive application of the SVRG technique and related approaches fail, and explore why.

Complications in practice

Data Augmentation

In order to achieve state-of-the-art results in most domains, data augmentation is essential. When applying standard SVRG using gradient recomputation, the use of random transforms can destroy the prospects of any variance reduction if different transforms are used for a data-point during the snapshot pass compared to the following steps. Using a different transform is unfortunately the most natural implementation when using standard libraries as the transform is applied automatically as part of the data-pipeline. We propose the use of transform locking, where the transform used during the snapshot pass is cached and reused during the following epoch/s.

For SVRG with transform locking, the variance of the step is initially zero at the very beginning of the epoch, increasing over the course of the epoch. This is the behavior expected of SVRG on finite sum problems. In contrast, without transform locking the variance is non-zero at the beginning of the epoch, and uniformly worse.

The handling of data augmentation in finite-sum methods has been previously considered for the MISO method, which is one of the family of gradient table methods (as with the storage variant of SVRG). The stored gradients are updated with an exponential moving average instead of overwriting, which averages over multiple past transformed-data-point gradients.

Batch Normalization

Batch normalization is another technique that breaks the finite-sum structure assumption. In batch normalization, mean and variance statistics are calculated within a mini-batch, for the activations of each layer (typically before application of a nonlinearity). These statistics are used to normalize the activations. The finite sum structure no longer applies since the loss on a datapoint depends on the statistics of the mini-batch it is sampled in.

The interaction of BN with SVRG depends on if storage or recomputation of gradients is used. When recomputation is used naively, catastrophic divergence occurs in standard frameworks. The problem is a subtle interaction with the internal computation of running means and variances, for use at test time.

In order to apply batch normalization at test time, where data may not be mini-batched or may not have the same distribution as training data, it is necessary to store mean and variance information at training time for later use. The standard approach is to keep track of a exponential moving average of the mean and variances computed at each training step. For instance, PyTorch by default will update the moving average using the mini-batch mean.

During test time, the network is switched to evaluation mode using model.eval(), and the stored running mean and variances are then used instead of the internal mini-batch statistics for normalization. The complication with SVRG is that during training the gradient evaluations occur both at the current iterate and the snapshot iterate. If the network is in train mode for both, the EMA will average over activation statistics between two different points, resulting in **poor results and divergence.**

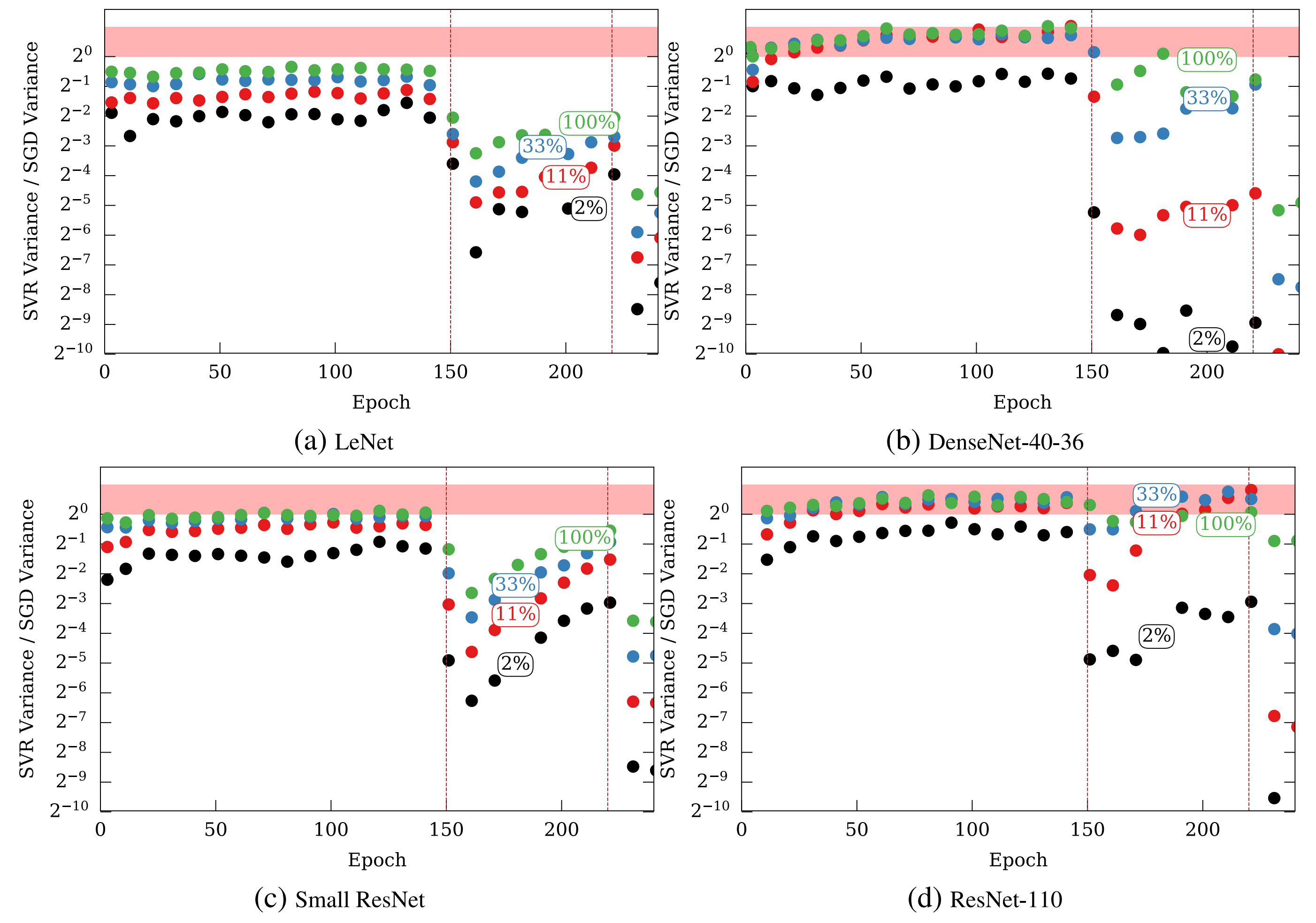
Switching the network to evaluation mode mid-step is the obvious solution, however computing the gradient using the two different sets of normalizations results in additional introduced variance. We recommend a BN reset approach, where the normalization statistics are temporarily stored before the w gradient evaluation, and the stored statistics are used to undo the updated statistics by overwriting afterwards. This avoids having to modify the batch normalization library code. It is important to use train mode during the snapshot pass as well, so that the mini-batch statistics match between the two evaluations.

To illustrate the degree of variance reduction achieved by SVRG on practical problems, we directly computed the variance of the SVRG gradient estimate, comparing it to the variance of the stochastic gradient used by SGD. To minimize noise the variance was estimated using a pass over the full dataset, although some noise remains due to the use of data augmentation.

Ratios below one indicate that variance reduction is occurring, whereas ratios around two indicate that the control variate is uncorrelated with the stochastic gradient, leading to an increase in variance. For SVRG to be effective we need a ratio below 1/3 to offset the additional computational costs of the method. We plot the variance ratio at multiple points within each epoch as it changes significantly during each epoch.

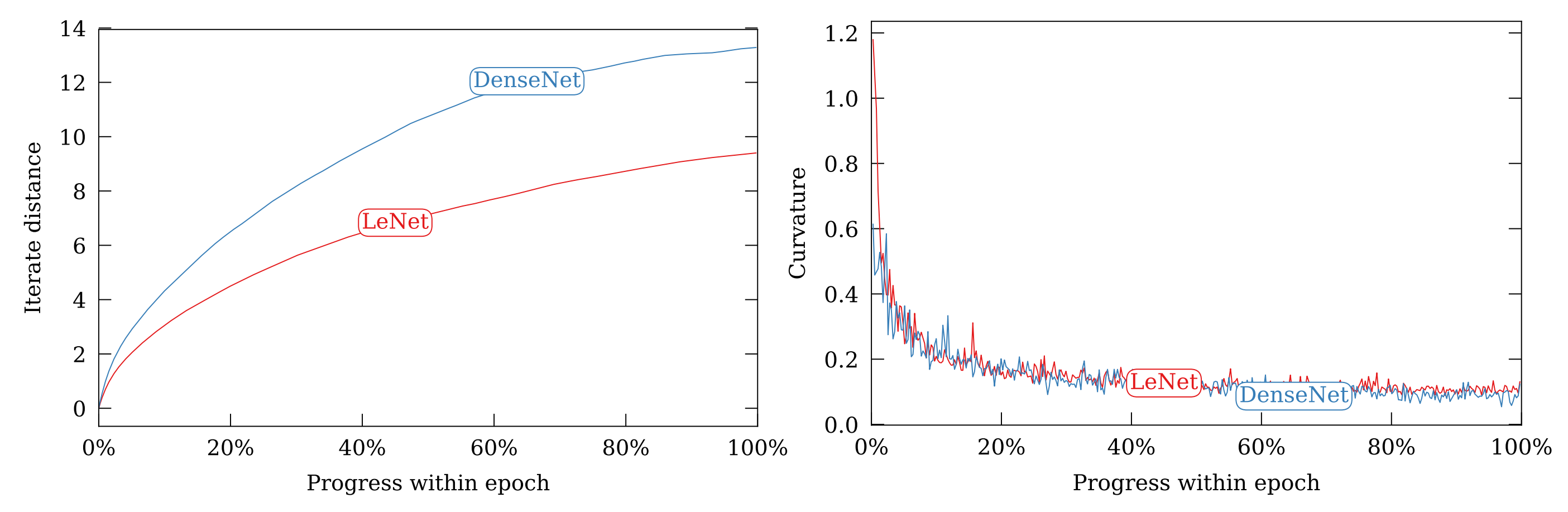
To highlight differences introduced by model complexity, we compared four models:

1. The classical LeNet-5 model [lenet], modified to use batch-norm and ReLUs, with approximately 62 thousand parameters[1].
2. A ResNet-18 model [resnet], scaled down to match the model size of the LeNet model by halving the number of feature planes at each layer. It has approximately 69 thousand parameters.
3. A ResNet-110 model with 1.7m parameters, as used by [resnet].
4. A wide DenseNet model [densenet] with growth rate 36 and depth 40. It has approximately 1.5 million parameters and achieves below 5% test error.

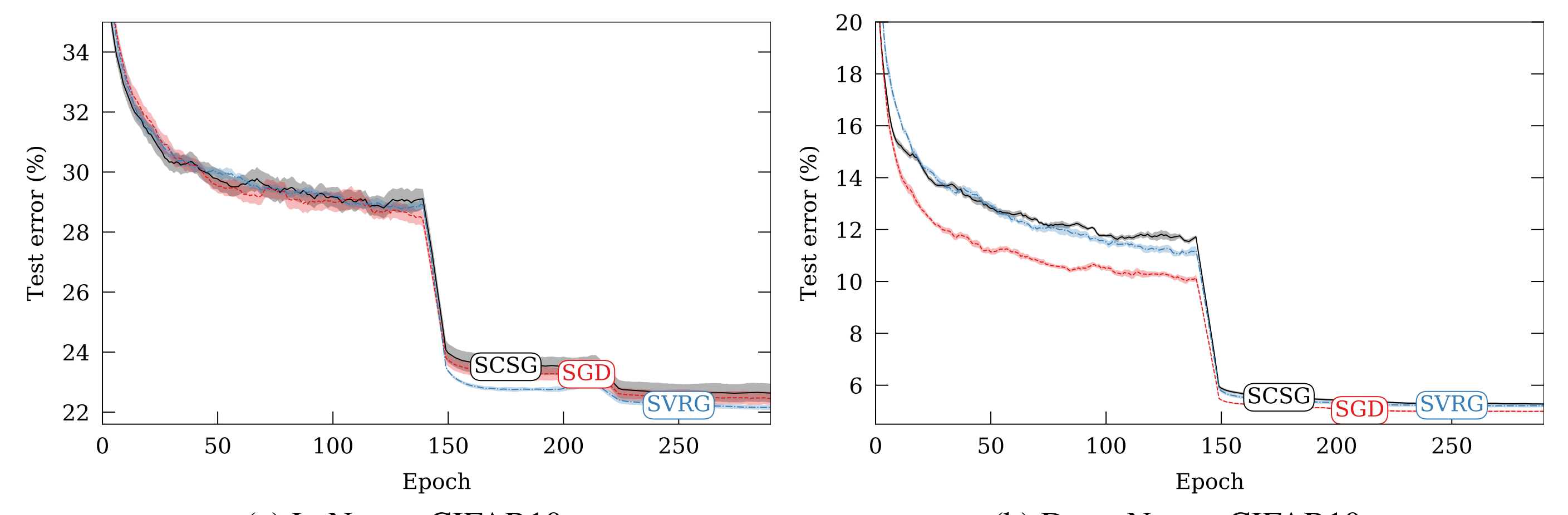


(a) LeNet (b) DenseNet-40-36 (c) Small ResNet (d) ResNet-110

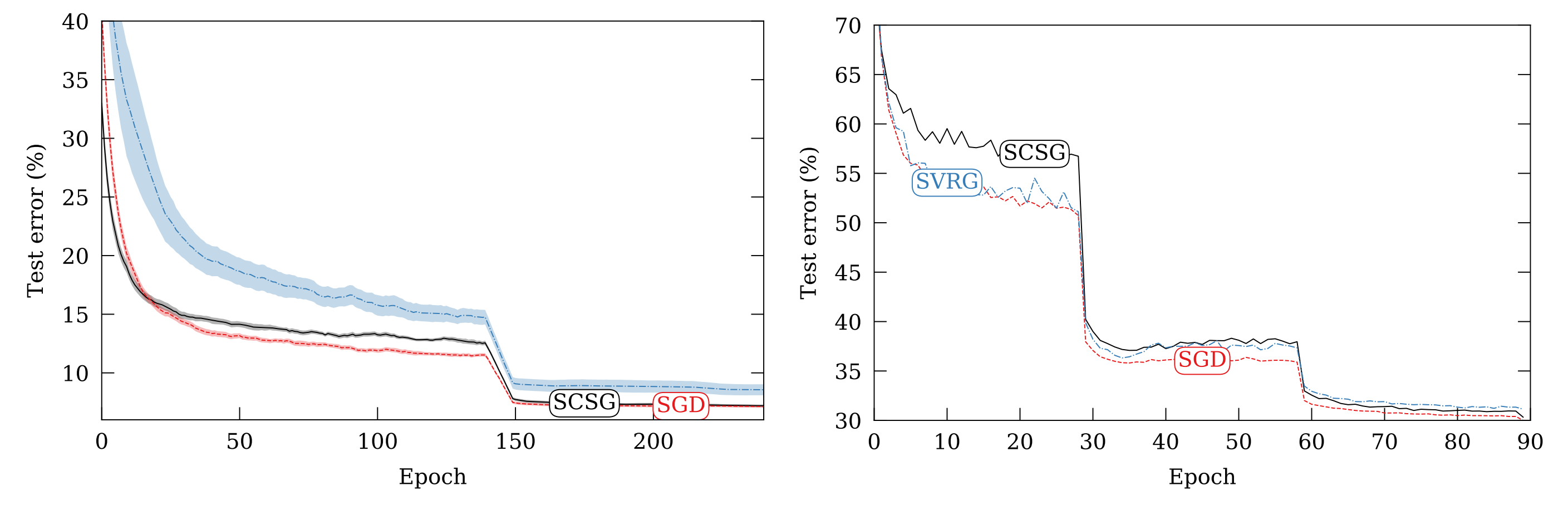
The SVRG to SGD gradient variance ratio during a run of SVRG. The shaded region indicates a variance increase, where the SVRG variance is worse than the SGD baseline. Dotted lines indicate when the step size was reduced. The variance ratio is shown at different points within each epoch, so that the 2% dots (for instance) indicate the variance at 1,000 data-points into the 50,000 datapoints consisting of the epoch. Multiple percentages within the same run are shown at equally spaced epochs. ***SVRG fails*** *to show a variance reduction for the majority of each epoch when applied to* ***modern high-capacity networks,*** *whereas some variance reduction is seem for smaller networks.*



Distance moved from the snapshot point (left), and curvature (right) relative to the snapshot point, at epoch 50. Higher capacity networks result in faster movement through parameter space, but similar curvature as given by the Lipschitz smoothness constant.



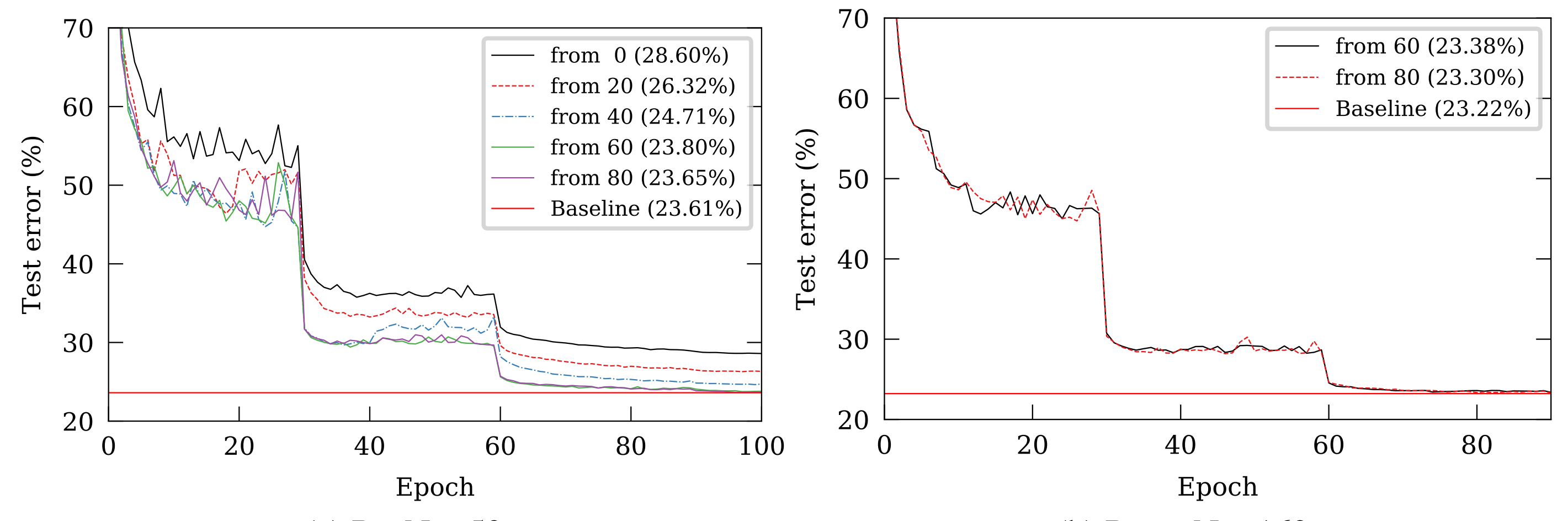
(a) LeNet on CIFAR10 (b) DenseNet on CIFAR10



(c) ResNet-110 on CIFAR10 (d) ResNet-18 on ImageNet

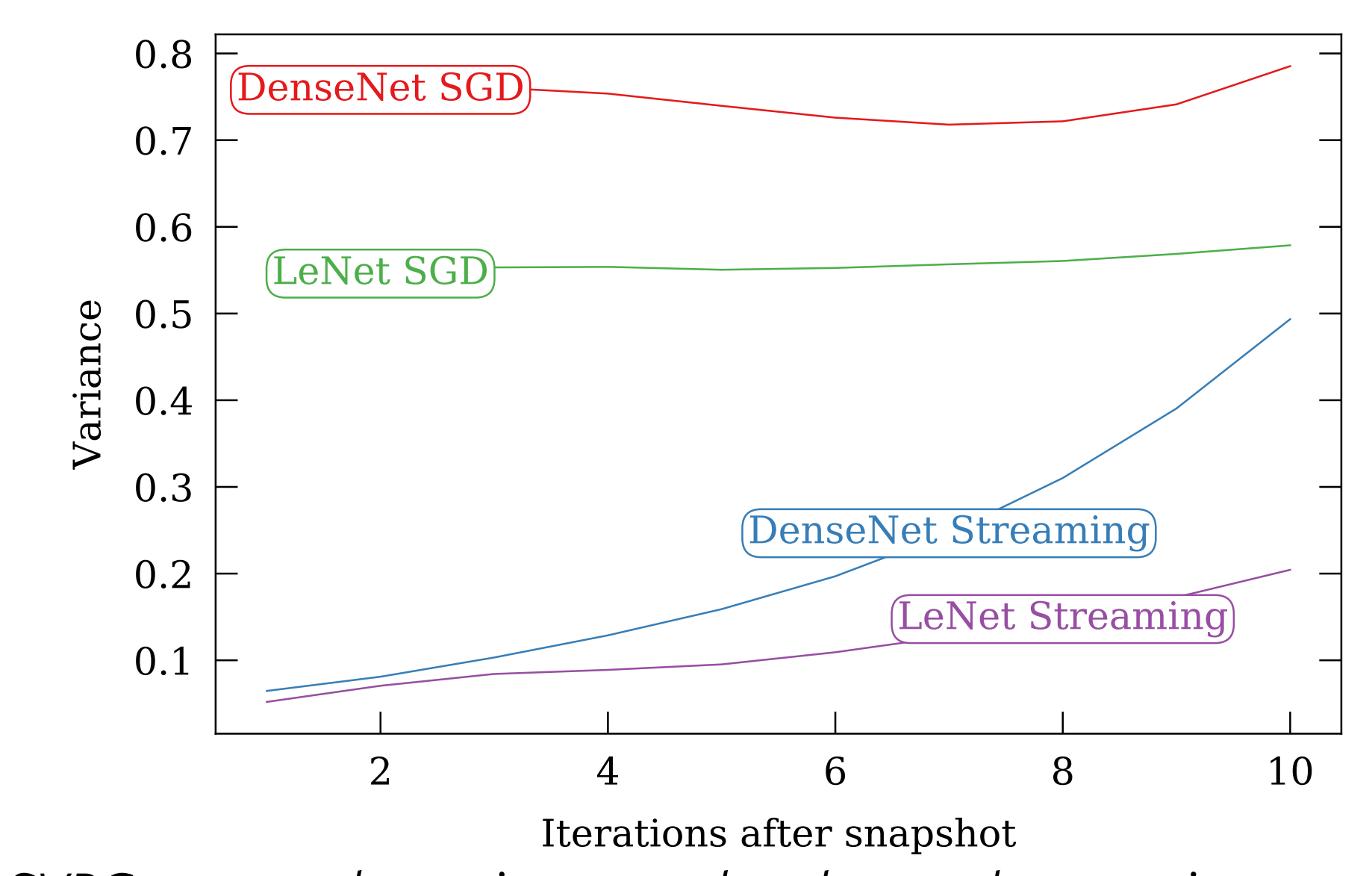
Test error comparison between SGD, SVRG and SCSG. For the CIFAR10 comparison a moving average (window size 10) of 10 runs is shown with 1 SE overlay, as results varied significantly between runs.

As we have shown that SVRG appears to only introduce a benefit late in training, we performed experiments where we turned on SVRG after a fixed number of epochs into training. Using the standard ResNet-50 architecture on ImageNet, we considered training using SVRG with momentum from epoch 0, 20, 40, 60 or 80, with SGD with momentum used in prior epochs. Figure fig:fine-tuning shows that the fine-tuning process did not lead to improved test accuracy at any interval compared to the SGD only baseline. For further validation we evaluated a DenseNet-169 model, which we only fine-tuned from 60 and 80 epochs out to a total of 90 epochs, due to the much slower model training. This model also showed no improvement from the fine-tuning procedure.



(a) ResNet-50 (b) DenseNet-169

Fine-tuning on ImageNet with SVRG



Streaming SVRG approaches using mega-batch snapshots, variance rapidly increases after the mega-batch, quickly approaching the non-variance reduced baseline for higher capacity models