Stochastic variance reduced gradient methods for training Deep Neural Networks

Alexander Apostolov

École Polytechnique Fédérale de Lausanne

February 1^{st} 2021

Project advised by:
Dr. Tatjana Chavdarova, Supervisor
Prof. Dr. Martin Jaggi, Advising Professor

Empirical study of **stochastic variance reduced gradient methods**[1] on training **deep neural networks** (DNNs).

[1] Herein referred as *Variance Reduced*, for short.

Background

Why variance reduction methods?

- In machine learning, the following optimization is often encountered. Let w denote the model parameters, and let $f_1, f_2, \ldots, f_n$ be a sequence of vector functions $\mathbb{R}^d \mapsto \mathbb{R}$ where each $f_i(\cdot)$ is the loss on a single training data point. The goal is to find a solution to the following **finite-sum** problem:

$$\min_w F(w), \quad F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \,.$$

- **Gradient Descent (GD)** is an iterative method, where at each step $t = 1, \ldots, T$:

$$w^{(t)} = w^{(t-1)} - \eta_t \nabla F\big(w^{(t-1)}\big) \,,$$

  where η_t is the learning rate at iteration t.

- Each iteration requires computation of the whole gradient, this is why **Stochastic Gradient Descent (SGD)**–which decreases the computational cost by $\frac{1}{n}$ per iteration, has been widely adopted:

$$w^{(t)} = w^{(t-1)} - \eta_t \nabla f_{i_t}\big(w^{(t-1)}\big) \,.$$

Why variance reduction methods?

Taking a sub-sample of the full dataset in SGD increases the variance of the gradient estimates, which in turn slows down convergence (in terms of parameter updates).

This creates a trade off between fast per iteration computation and fast convergence.

Variance reduced methods allow keeping fast convergence at fast per iteration speed. We will analyze two methods:

- **Stochastic Variance Reduced Gradient (SVRG)** [Johnson and Zhang, 2013]
- **STOchastic Recursive Mometum (STORM)** [Cutkosky and Orabona, 2019]

SVRG

Each iteration of SVRG is given by:

$$w^{(t)} = w^{(t-1)} - \eta(\nabla f_{i_t}(w^{(t-1)}) - \nabla f_{i_t}(\tilde{w}) + \tilde{\mu})$$

where $\tilde{w}$ is snapshot of w taken when $\tilde{\mu}$ is computed every m iterations, and:

$$\tilde{\mu} = \nabla F(\tilde{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{w})$$

SVRG

Algorithm 1: SVRG Procedure

Input: learning rate η and update frequency m
Initialize $\tilde{w}_0$
for $epoch \leftarrow 1, 2, \ldots$ **do**
    $\tilde{w} \leftarrow \tilde{w}_{epoch-1}$
    $\tilde{\mu} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{w})$
    $w^{(0)} \leftarrow \tilde{w}$
    for $t \leftarrow 1, 2 \ldots, T$ **do**
        Randomly pick $i_t \in \{1, \ldots, n\}$
        $w^{(t)} \leftarrow w^{(t-1)} - \eta(\nabla f_{i_t}(w^{(t-1)}) - \nabla f_{i_t}(\tilde{w}) + \tilde{\mu})$
    end
    Save snapshot $\tilde{w}_{epoch} \leftarrow w_T$
end

SVRG

Notice that:

- $\nabla f_{i_t}(w^{(t-1)}) - \nabla f_{i_t}(\tilde{w}) + \tilde{\mu}$ is an unbiased estimate of $\nabla F(w)$:

$$\mathbb{E}[w^{(t)}|w^{(t-1)}] = w^{(t-1)} - \eta_t \nabla F(w^{(t-1)})$$

- Variance is reduced. When $\tilde{w}$ and $w^{(t)}$ both converge to w_*, then $\tilde{\mu} \to 0$. If $\nabla f_i(\tilde{w}) \to \nabla f_i(w_*)$, then:

$$\nabla f_i(w^{(t-1)}) - \nabla f_i(\tilde{w}) + \tilde{\mu} \to \nabla f_i(w^{(t-1)}) - \nabla f_i(w_*) \to 0$$

STORM

Each iteration of STORM is given by:

$$d^{(t)} = (1-a)d^{(t-1)} + a\nabla f_{i_t}(w^{(t)}) + (1-a)(\nabla f_{i_t}(w^{(t)}) + \nabla f_{i_t}(w^{(t-1)}))$$

$$w^{(t+1)} = w^{(t)} - \eta_t d_t$$

Notice that:

- When $w^{(t)} \approx w^{(t-1)}$, then the last term of the first equality tends to 0.

STORM

Algorithm 2: STORM Procedure

Input: Hyperparameters k, w, c

Initialize w_1

$G_1 \leftarrow |\nabla f_{i_1}(w_1)|$

$d_1 \leftarrow \nabla f_{i_1}(w_1)$

for $t \leftarrow 1, 2, \ldots$ **do**

$\eta_t \leftarrow \frac{k}{(w_t + \sum_{i=1}^{t} G_i^2)^{1/3}}$

$w_{t+1} \leftarrow w_t - \eta_t d_t$

$a_{t+1} \leftarrow c\eta_t^2$

$G_{t+1} \leftarrow |\nabla f_{i_{t+1}}(w_{t+1})|$

$d_{t+1} \leftarrow \nabla f_{i_{t+1}}(w_{t+1}) + (1 - a_{t+1})(d_t - \nabla f_{i_{t+1}}(w_t))$

end

SVRG vs STORM

- SVRG compared to STORM:
  - \+ Difficult to get insight on hyperparameters k, w and c in STORM.
  - \+ More hyperparameters to tune for STORM.
  - − SVRG needs a full gradient computation every m steps (every 5 epochs in non-convex problems).
- Both algorithms need two gradient computations per iteration.
- Both algorithms do not have official PyTorch implementations.

Batch-normalization layers [Ioffe and Szegedy, 2015]

- Layer used to normalize the input of layers of the network with the following formula:

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\mathbb{V}ar[x] + \epsilon}},$$

- No direct access to $\mathbb{E}[x]$ and $\mathbb{V}ar[x]$ as $x's$ are only seen throughout the training,
- Using running estimates for these two values by using average computations with momentum.
- Breaks assumptions for unbiased estimates in variance reduced methods.

MetaInit

Method for weights initialization introduced in 2019 [Dauphin and Schoenholz, 2019].

- Deep Neural networks can be difficult to train, the performance depends heavily on how weights are initialized.
- Hypothesis of MetaInit: start training in regions that look locally linear with minimal second order effects.
- MetaInit uses measures the above with a quantity called gradient quotient.
- The algorithm minimizes this quantity using GD to select the norms of initial weights.

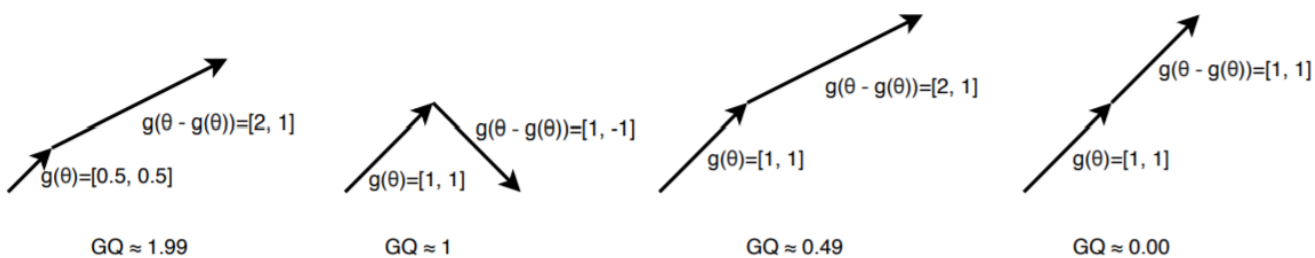


Figure: Gradient quotient for different scenarios of initial gradient $g(\theta)$ and gradient after one update $g(\theta - g(\theta))$.

Project motivation & goals

Goal of this semester project

Empirical study of **stochastic variance reduced gradient methods**[2] on training **deep neural networks** (DNNs).

- Comparison between different variance reduced and stochastic methods on different architectures:
  - Shallow neural network (LeNet),
  - Deep neural network (ResNet18),
  - Deeper neural network (ResNet101).
- Goal: understand better & potentially improve the performance of variance reduction methods for training DNNs.

[2] Herein referred as *Variance Reduced*, for short.

Variance reduced methods on deep neural networks

We want to test if variance reduced methods are still performing better on deep neural networks.
The paper "On the Ineffectiveness of Variance Reduced Optimization for Deep Learning" [Defazio and Bottou, 2019] suggests that this is not the case, but:

- They give **no theoretical insights** why variance reduced methods perform poorly.
- They **use Batch-Norm Layers** in their setup. Batch-Norm layers break the assumption that the updates in SVRG are unbiased in expectation.

Hypothesis we want to test

- Is the use of batch-normalization layers related to the failure of the stochastic variance reduced gradient (SVRG) method on deeper architectures?
- Can initialization methods help give better results to variance reduced methods in a setting without batch-normalization layers, by selecting better initial weights for training?

Experiments & Results

Baseline comparison on shallow NN

We first performed a baseline comparison on MNIST on LeNet for the following algorithms:

- Adam [Kingma and Ba, 2015],
- SGD
- AdaGrad [Duchi et al., 2011],
- SVRG
- STORM

Hyperparameters selected by doing a 4-fold cross-validation.

CV results example

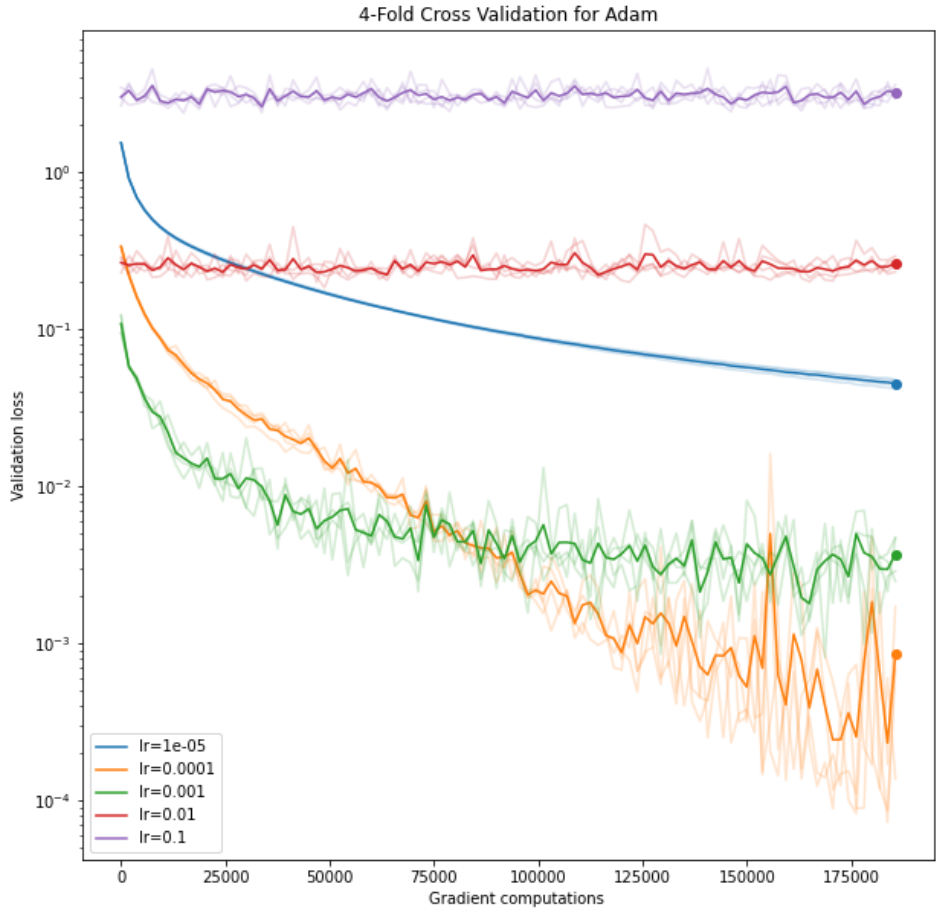


Figure: Cross validation of Adam on LeNet for MNIST

Results on LeNet

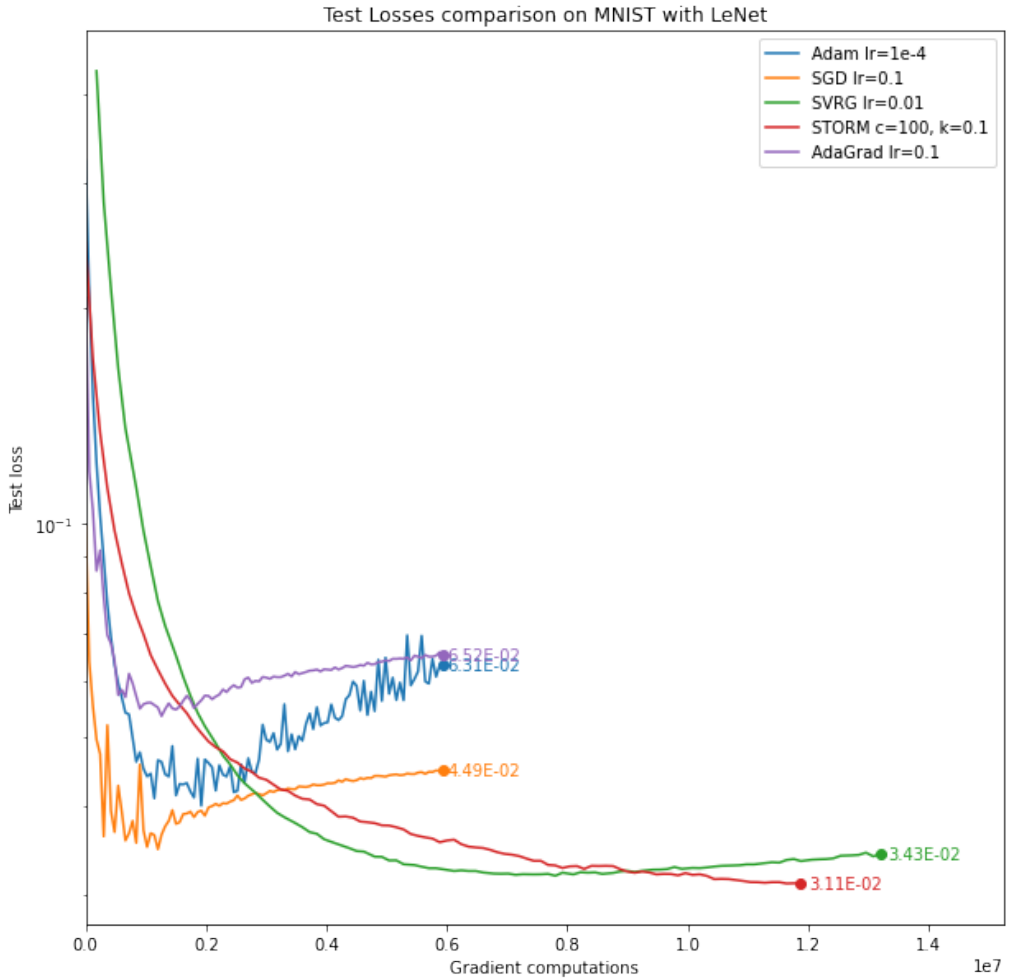
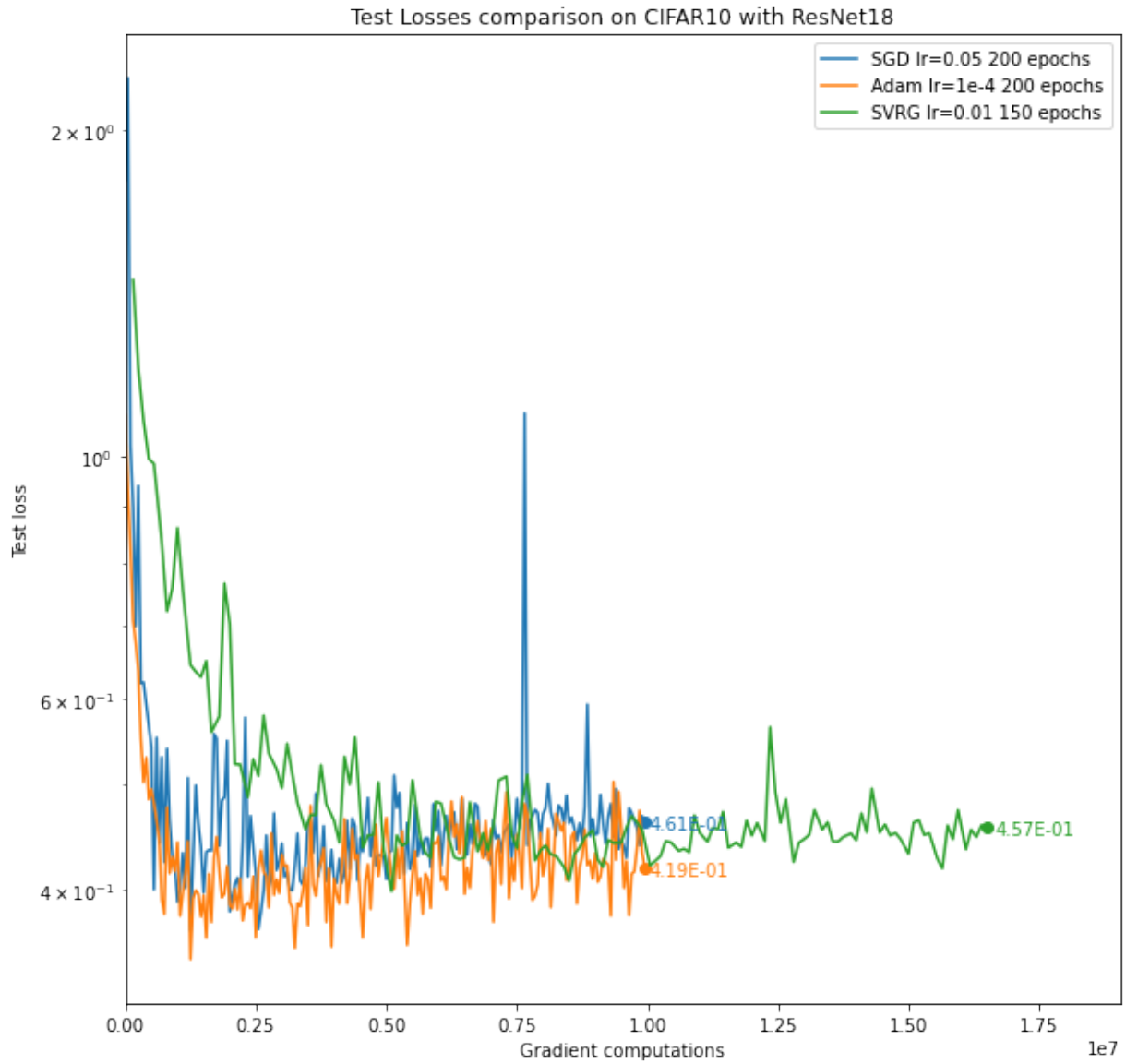


Figure: Test losses on LeNet for MNIST

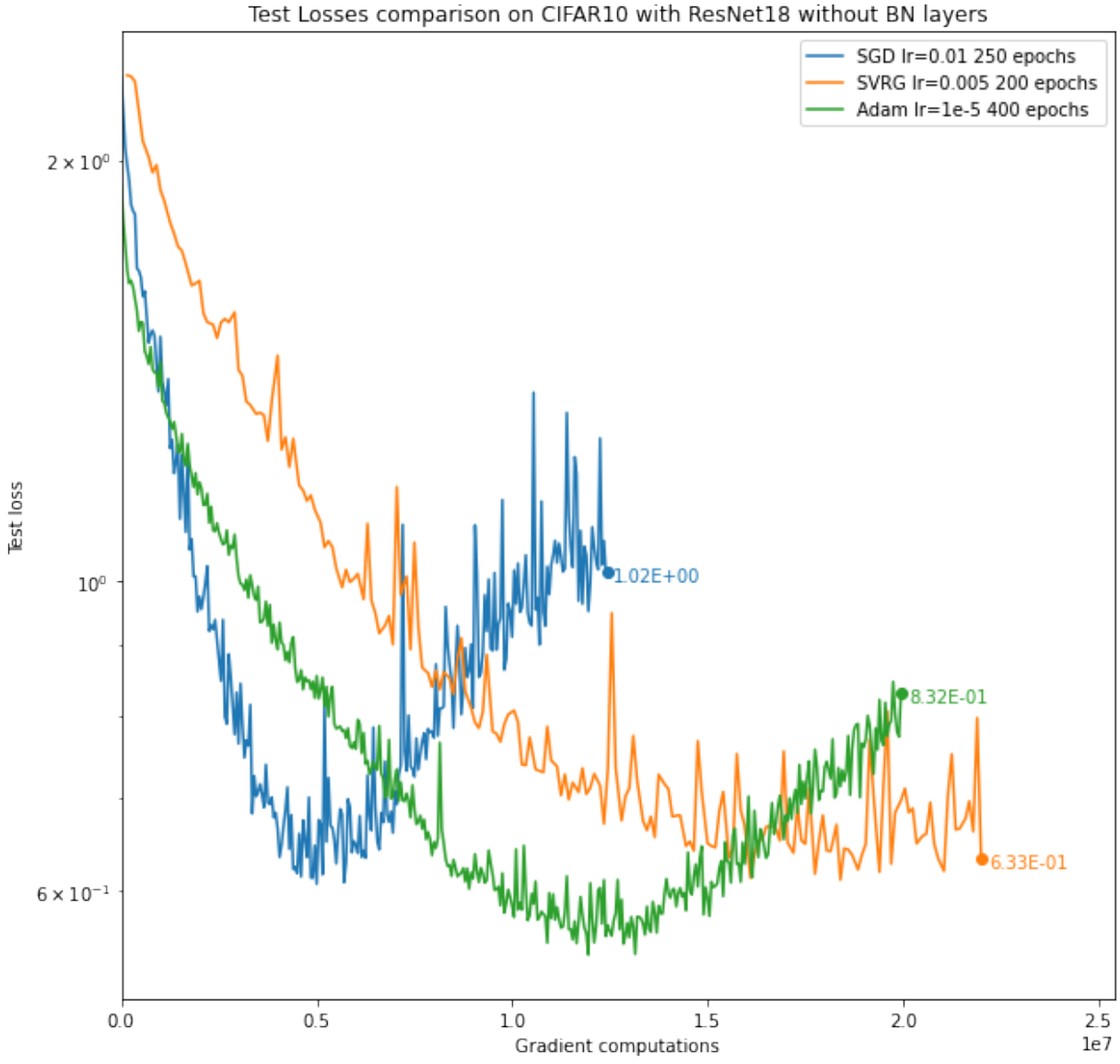
Setup for experiments on deep NN

- ResNet18 for CIFAR10,
- ResNet101 for CIFAR100,
- Both tested with and without batch-normalization layers,
- Hyperparameters selected with validation set.

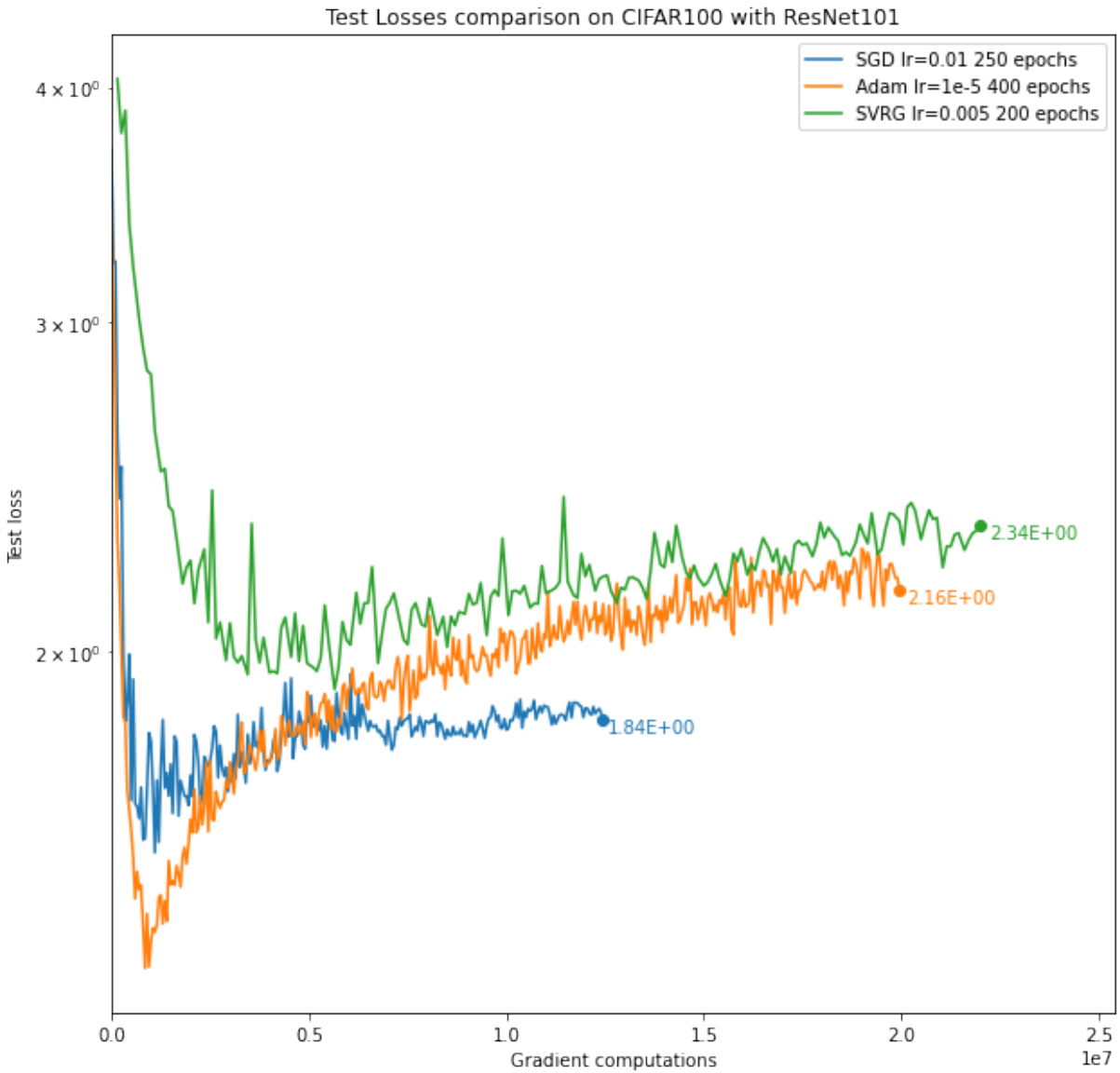
Results ResNet18



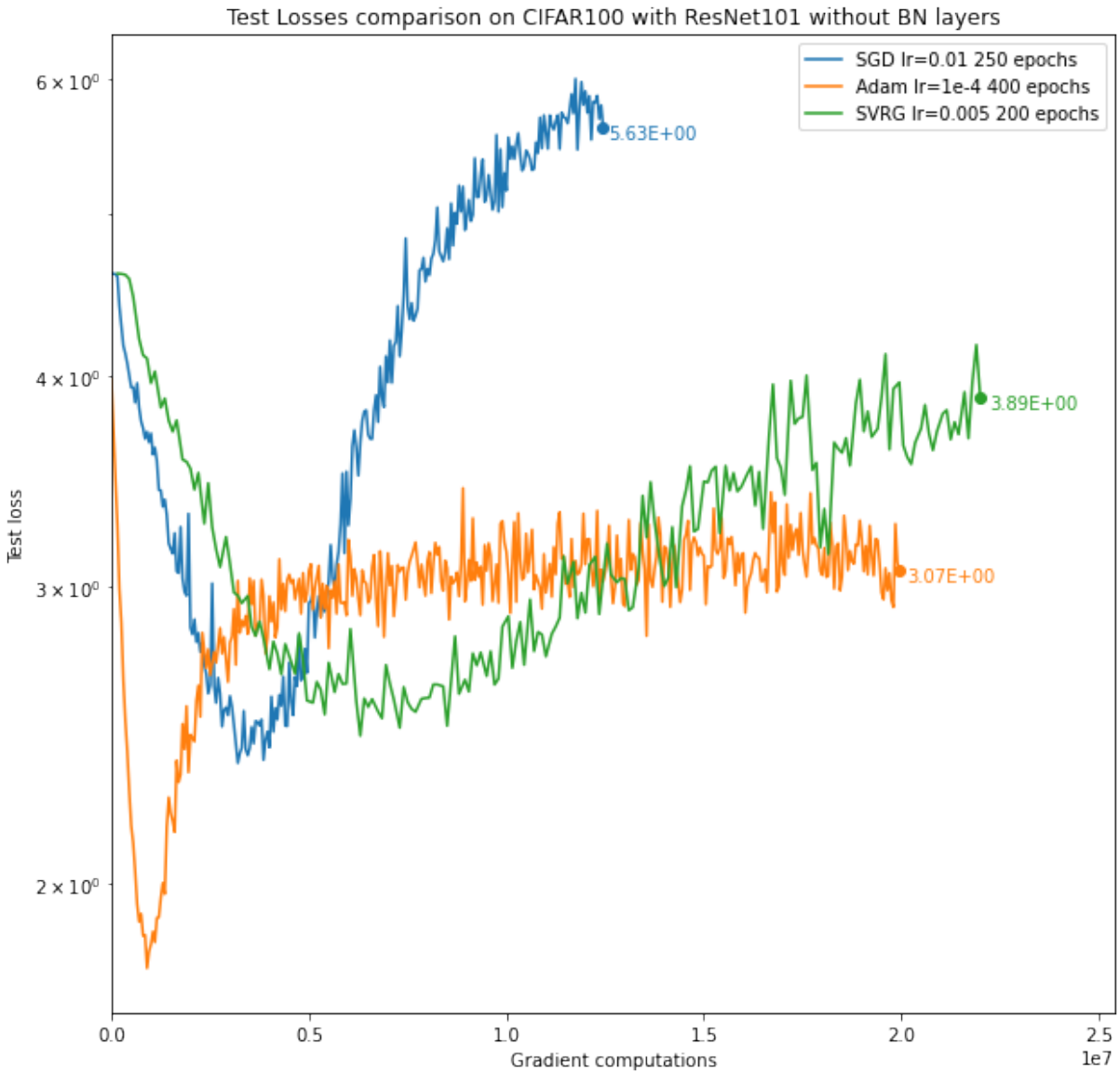
Results ResNet18 without BN



Results ResNet101



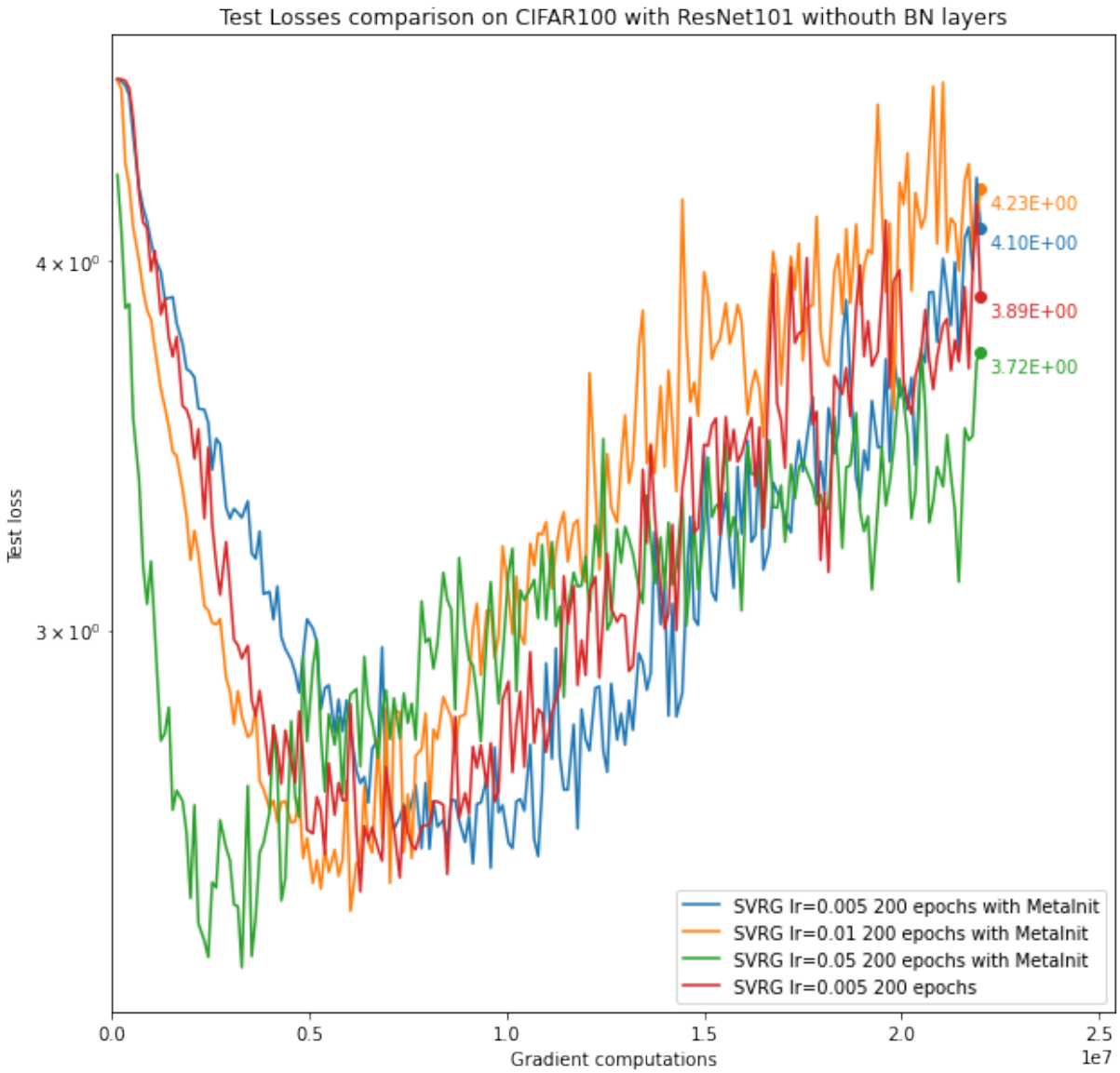
Results ResNet101 without BN



MetaInit setup

- We want to test whether using better weight initialization can help get better results in a setting without batch-normalization layers.
- Compare SVRG on ResNet101 without BN layers with and without MetaInit.

MetaInit results



Conclusions

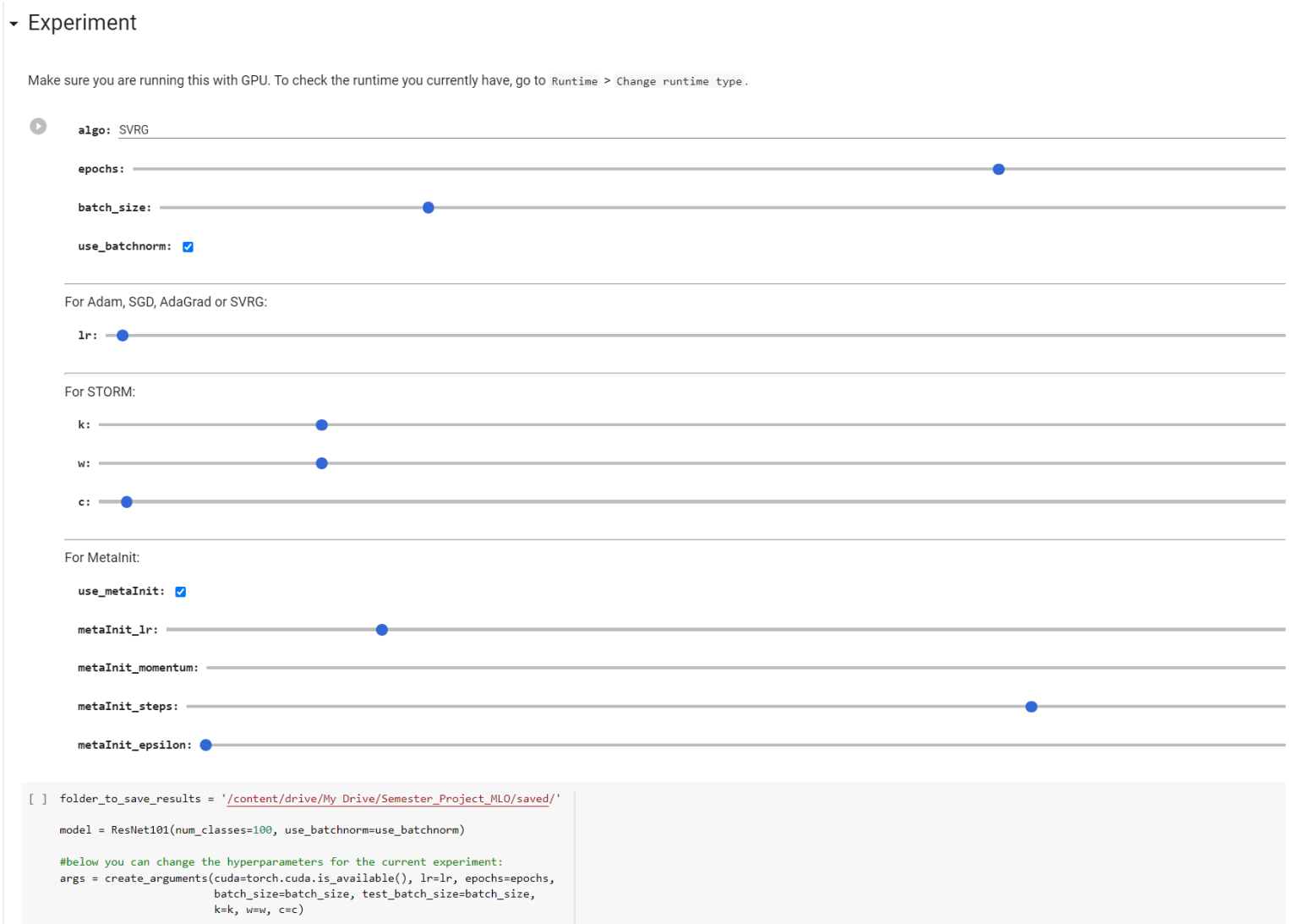
Conclusion

+ Variance reduced methods are performing better than Adam and SGD on smaller architectures,
- VR methods outperformed on common deeper architectures with BN layers,
+ SVRG gets similar or better results than Adam and SGD when BN layers are removed,
- Searching for hyperparameters is harder without BN layers,
- Results without BN layers are not as good as with BN layers,
+ MetaInit helps for using bigger learning rates and get faster initial results.

In summary, we seem to observe that SVRG is not disadvantaged as much as the other algorithms when removing batch-normalization layers from common DNNs but there are other unexplained phenomena that make the results somehow worse for all optimization algorithms.

Available tools

- Code for SVRG and STORM implementation and code for all experiments available in the repo of this project.
- Google Colab Notebook available to recreate experiments:



Future work

- Proper Cross-validation for DNNs (with MetaInit),
- Test different DNNs, datasets and random seeds for more conclusive results.

Thank you
Any questions?

References I

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. *CoRR*, abs/1905.10018, 2019. URL http://arxiv.org/abs/1905.10018.

Yann N Dauphin and Samuel Schoenholz. Metainit: Initializing learning by learning to initialize. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 12645–12657. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/876e8108f87eb61877c6263228b67256-Paper.pdf.

Aaron Defazio and Leon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *NeurIPS*, 2019.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 2011. ISSN 1532-4435.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 315–323. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Additional slides

Setup shallow NN

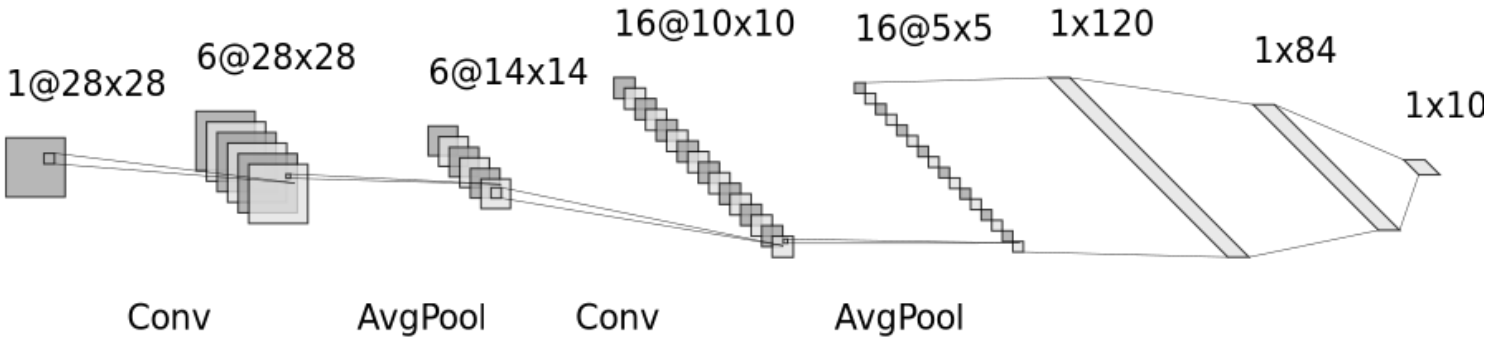


Figure: LeNet network used for classification of MNIST

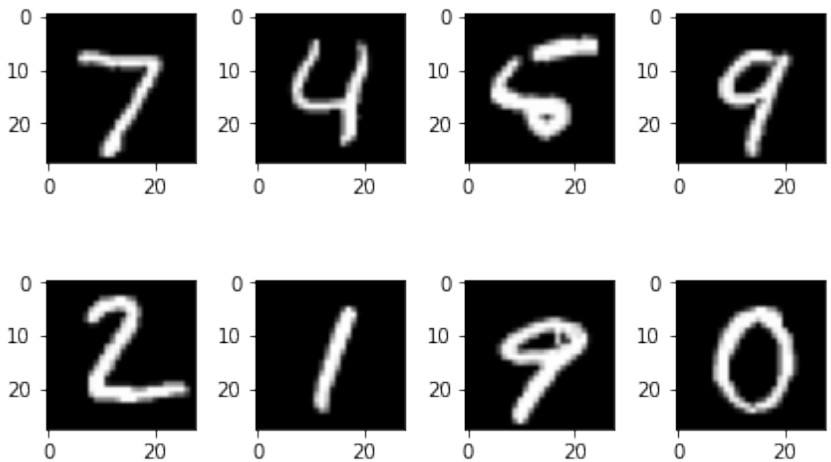


Figure: Example images of MNIST

ResNet

Network for image recognition introduced in 2015 [He et al., 2015].

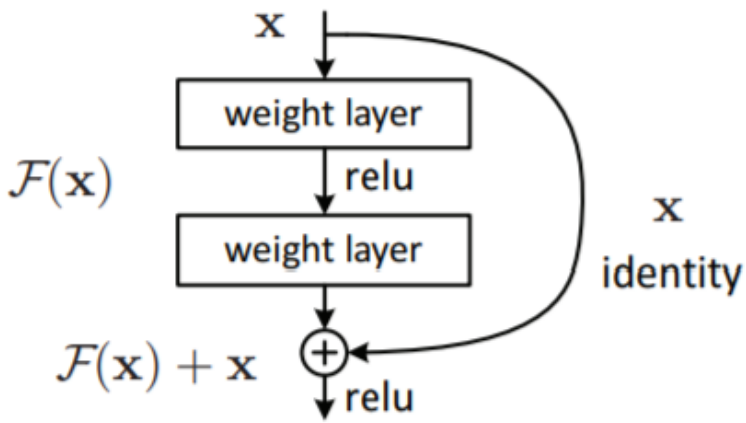
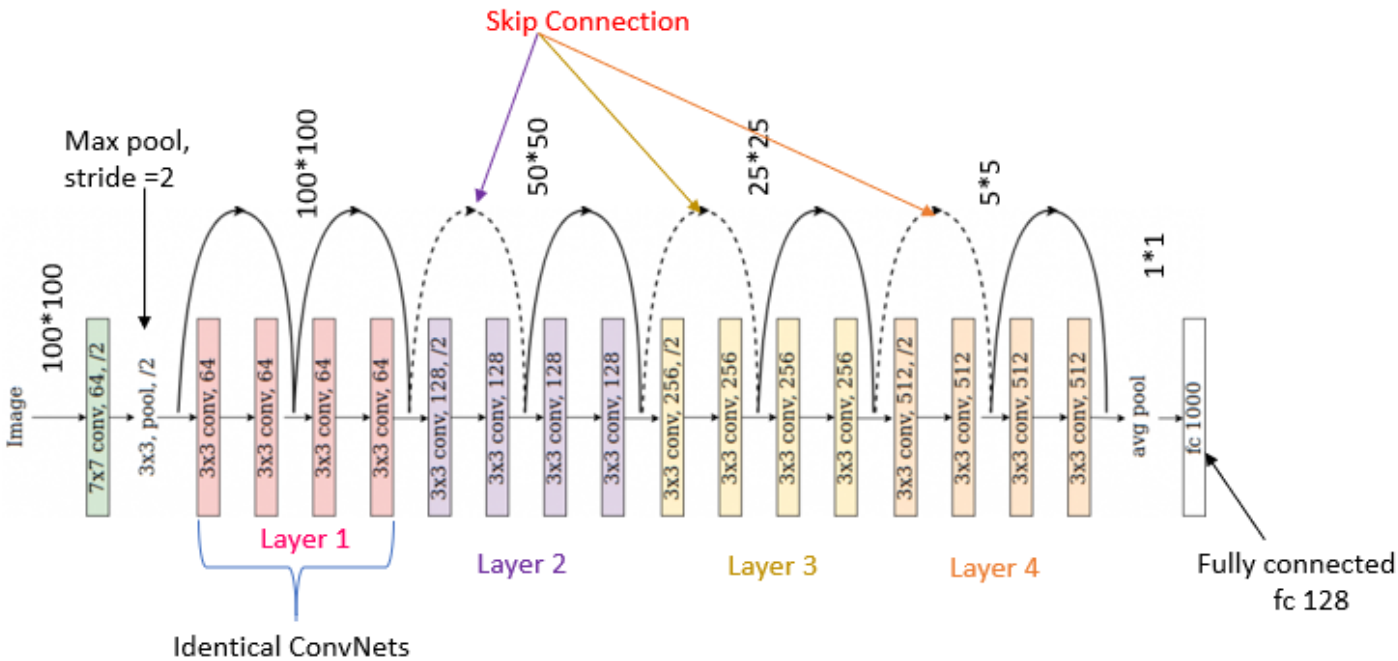


Figure: Residual learning: a building block.

ResNet18



ResNet-18 Architecture

Fruit 360 Input Image size= 100*100 px

Results on LeNet extended

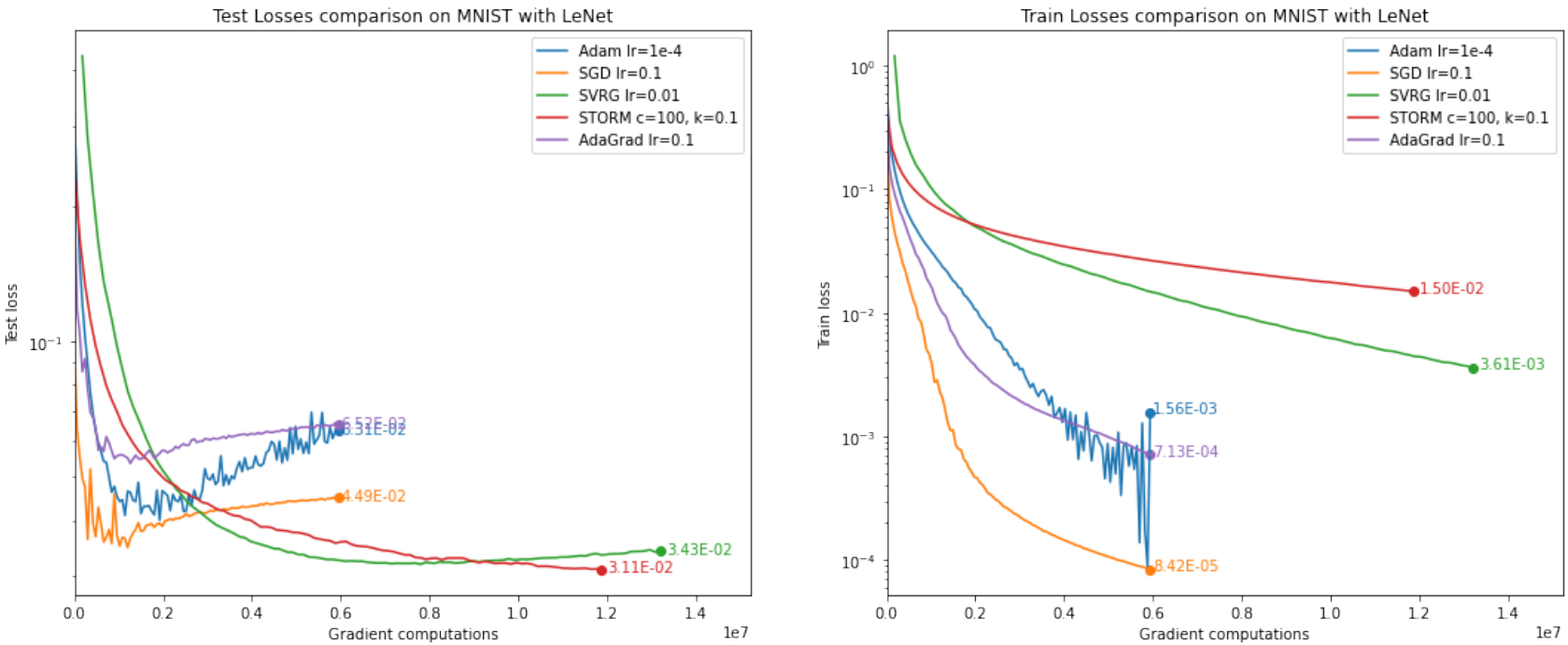
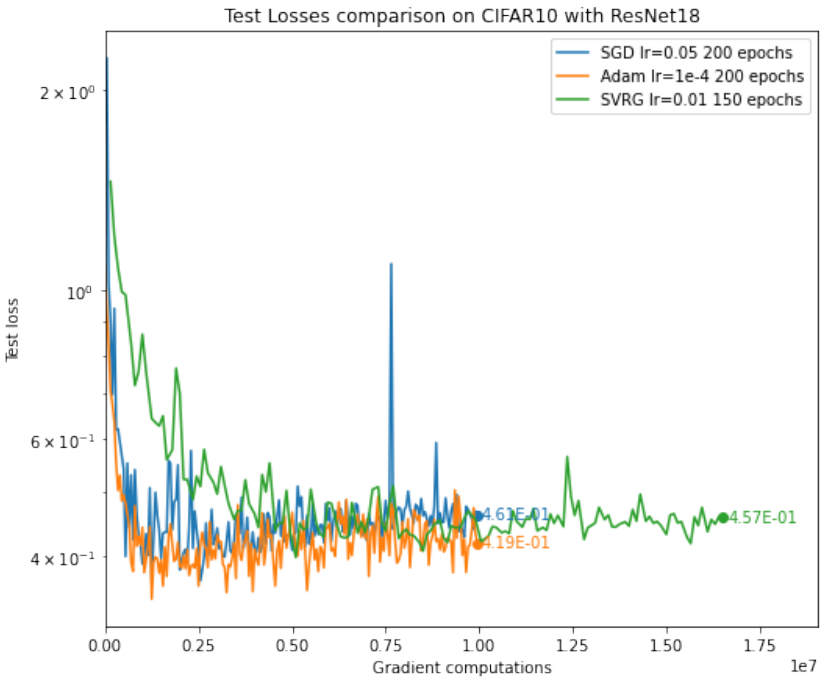


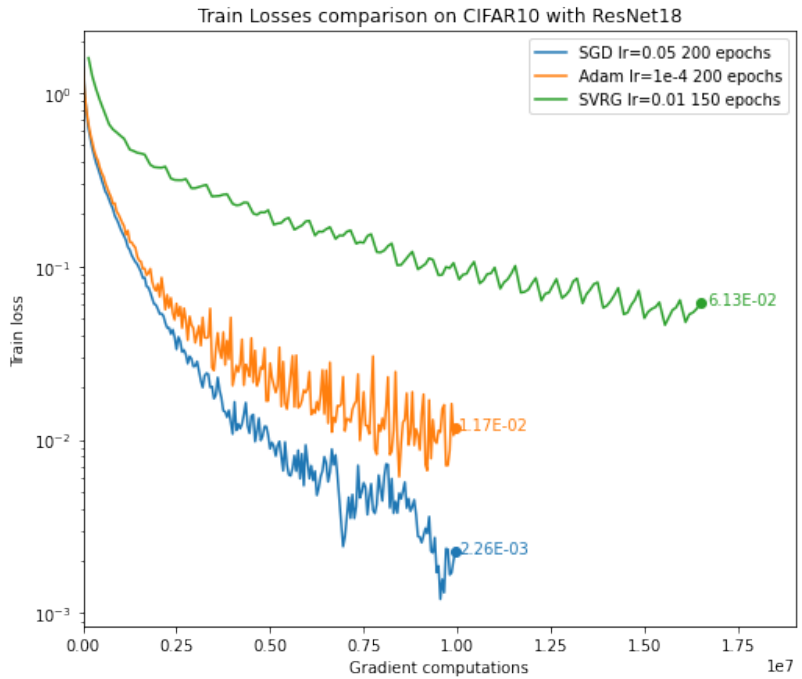
Figure: Test and train losses on LeNet for MNIST

Results on LeNet accuracies

Algorithm	Test accuracy	Train accuracy
Adam	0.9884	0.9999
SGD	0.9902	1.0
SVRG	0.9891	0.9996
STORM	0.9898	0.9964
AdaGrad	0.9859	0.9999

Results ResNet18 extended

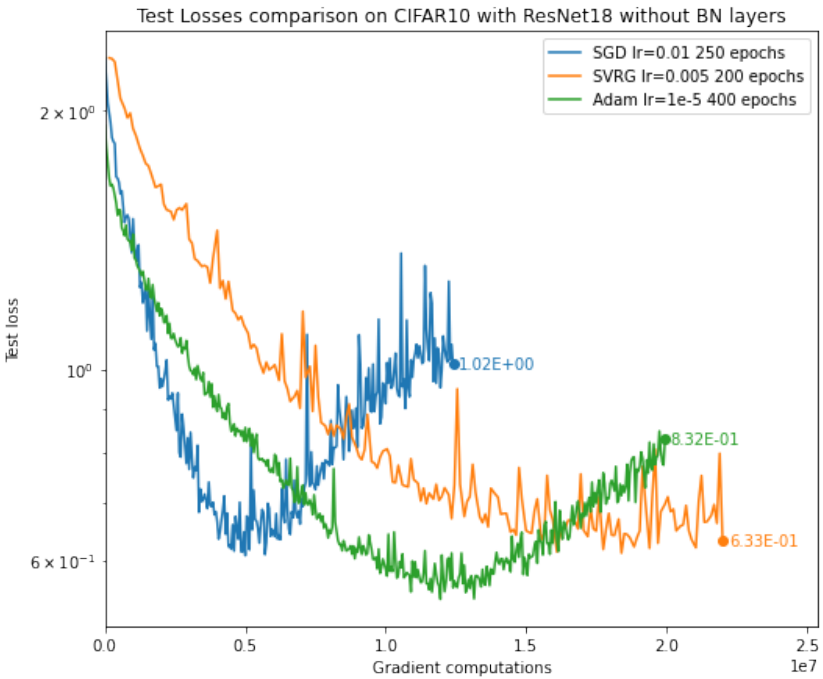


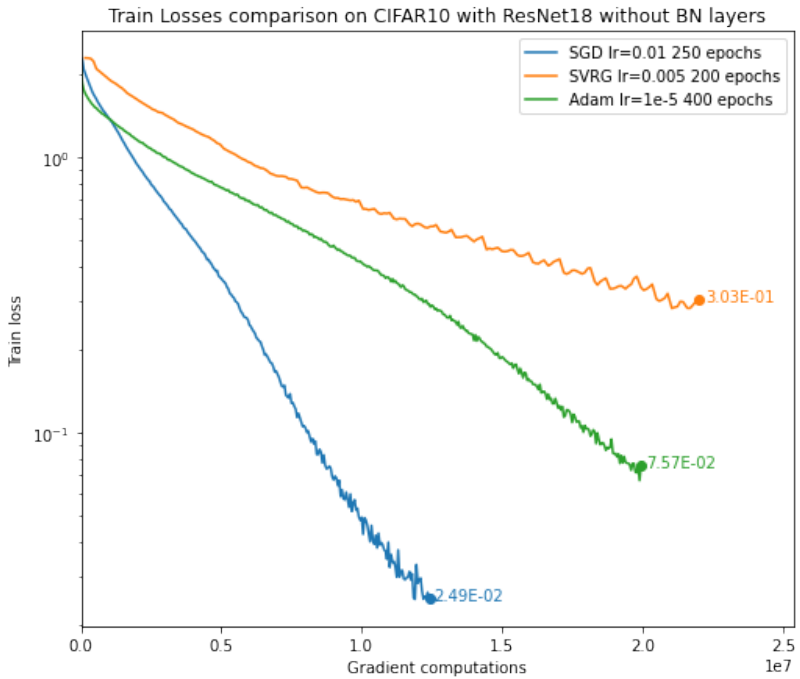


Results on ResNet18 accuracies

Algorithm	Test accuracy	Train accuracy
SGD	0.9221	0.9996
ADAM	0.9184	0.9976
SVRG	0.886	0.97664

Results ResNet18 without BN layers extended

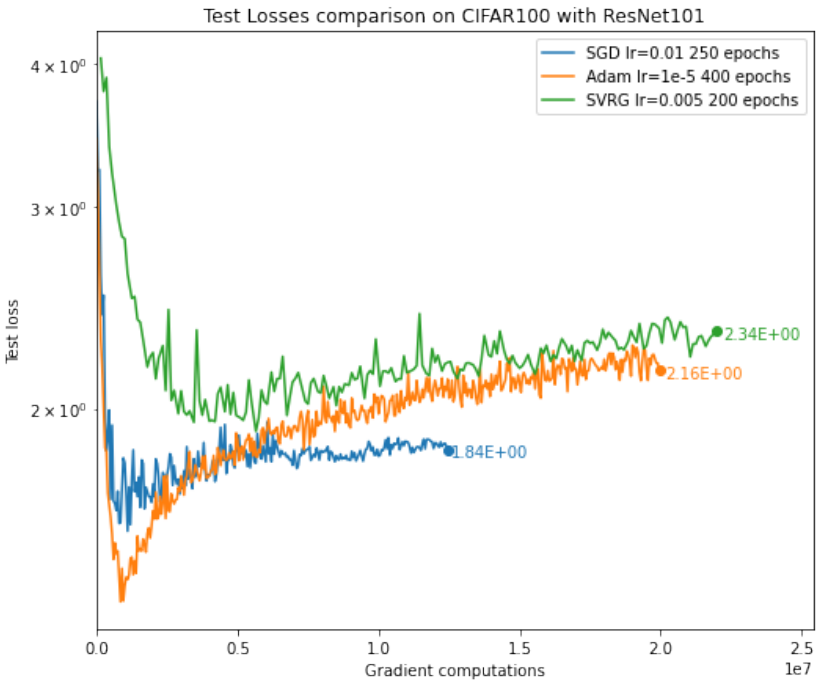


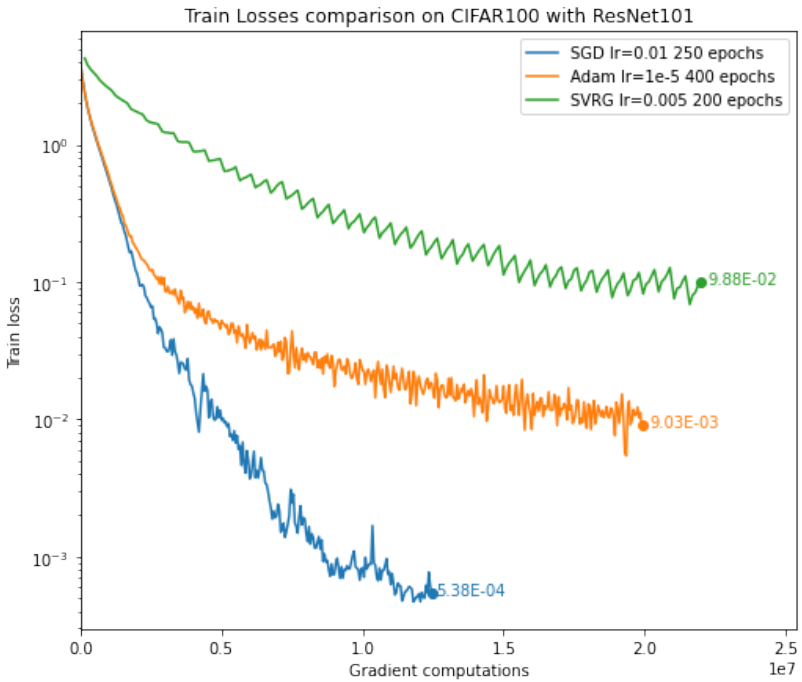


Results on ResNet18 without BN layers accuracies

Algorithm	Test accuracy	Train accuracy
SGD	0.8382	0.9924
ADAM	0.8268	0.9712
SVRG	0.8044	0.8990

Results ResNet101 extended

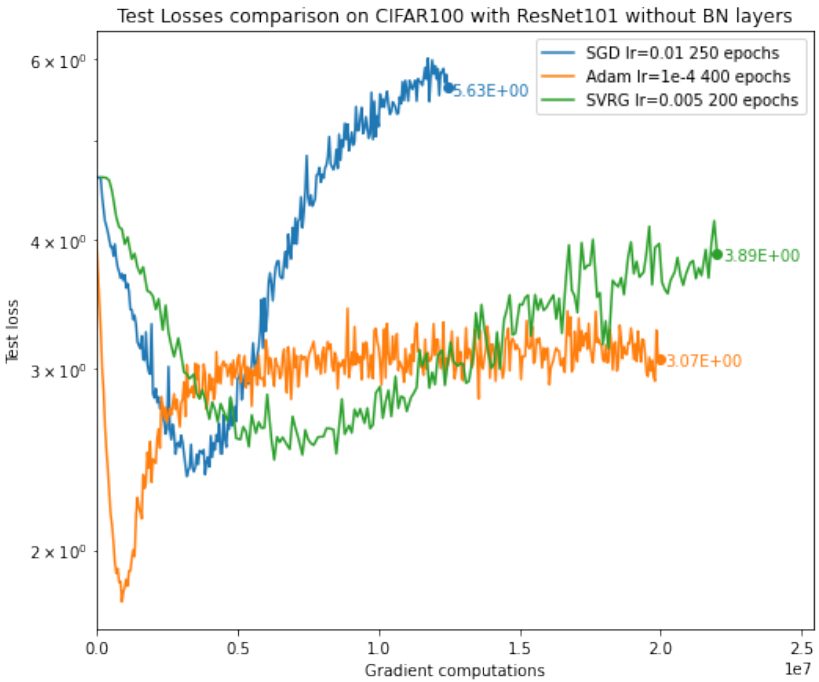


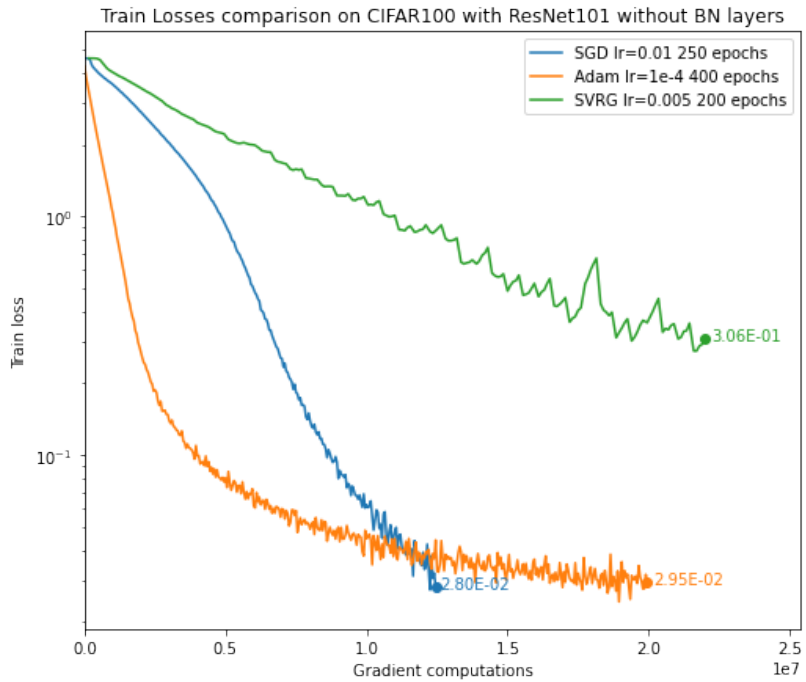


Results on ResNet101 accuracies

Algorithm	Test accuracy	Train accuracy
SGD	0.7239	0.9999
ADAM	0.7073	0.9984
SVRG	0.5947	0.9734

Results ResNet101 without BN layers extended





Results on ResNet101 without BN layers accuracies

Algorithm	Test accuracy	Train accuracy
SGD	0.4771	0.9925
ADAM	0.6055	0.9934
SVRG	0.4441	0.8915