Bolstering Stochastic Gradient Descent with Model Building

Özgür Martin

Nesin Matematik Köyü
February 10, 2023

MİMAR SİNAN
GÜZEL SANATLAR
ÜNİVERSİTESİ

A "Large-Scale" Machine Learning Example

The CIFAR-10* dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class.

bird (2)

truck (9)

frog (6)

Image from www.tensorflow.org

*Alex Krizhevsky, Learning multiple layers of features from tiny images, Tech. report, 2009.

A "Large-Scale" Machine Learning Example

$\longrightarrow$ $\{x_i\}_{i=1}^{n}$ with $x_i = \underbrace{(x_{i,1}, \ldots, x_{i,d_x})}_{\text{Pixels}} \in \mathbb{R}^{d_x}$ *

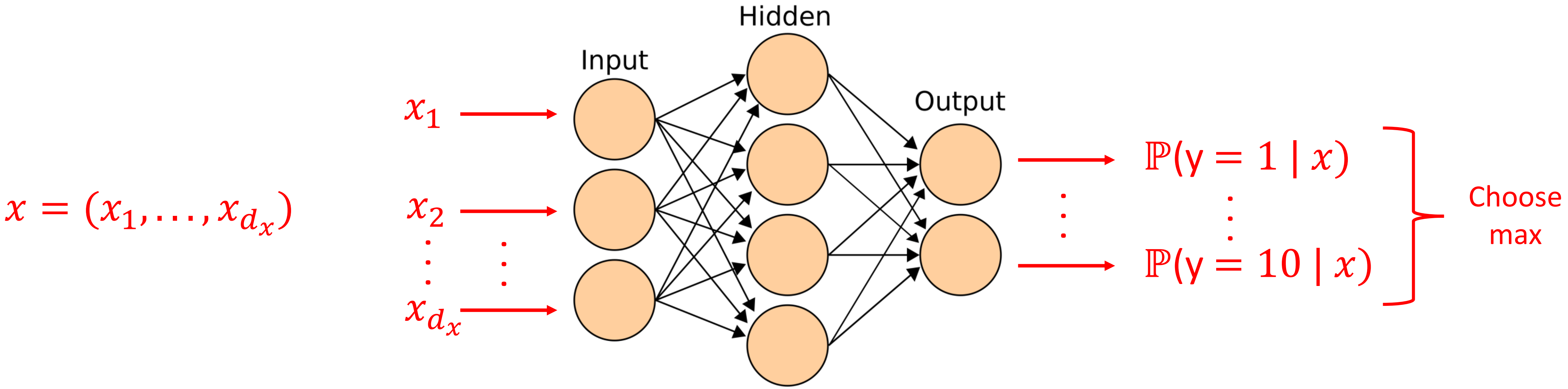
$\underbrace{\{\text{bird, truck, ..., frog}\}}_{\text{Labels}} \longrightarrow \underbrace{\{y_i\}_{i=1}^{n}}_{\text{Target Values}}$ with $y_i \in \{1, \ldots, 10\} \subset \mathbb{R}$

Training points: $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^{d_x} \times \mathbb{R}$ = $\underbrace{\mathcal{X}}_{\text{Input Space}} \times \underbrace{\mathcal{Y}}_{\text{Target Space}}$

* $n = 60000$, $d_x = 32 \times 32 \times 3 = 3072$

A "Large-Scale" Machine Learning Example

Goal. Find $h : \mathcal{X} \to \mathcal{Y}$ s.t. for any given $x \in \mathcal{X}$, $h(x) \approx y$,
where sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from a joint prob. dist. funct. $P(x, y)$.



An example for h: Artificial Neural Network*

*Image from Wikipedia

Optimization Problem

Goal. Find $h : \mathcal{X} \to \mathcal{Y}$ s.t. for any given $x \in \mathcal{X}$, $h(x) \approx y$,
where sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from a joint prob. dist. funct. $P(x, y)$.

Minimize the **expected risk** ***R*(*h*)**:

$$\min_h \left\{ R(h) := \mathbb{P}[h(x) \neq y] = \mathbb{E}[\mathbb{1}(h(x) \neq y)] \right\}.$$

In practice, we minimize the **empirical risk** $\boldsymbol{R_n(h)}$:

$$\min_h \left\{ R_n(h) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x_i) \neq y_i)] \right\}.$$

Here, $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ are independently drawn.

Optimization Problem

Thus, we solve:

$$\min_h \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x_i) \neq y_i))^{*}$$

In order to make this problem easier to solve:

- Choose h from $\mathcal{H} = \{h(.; w) : w \in \mathbb{R}^d\}$
- Use a continuous (smooth) *loss* function $\ell : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ to approximate $\mathbb{1}$

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i)$$

* $\mathbb{1}[A] = 1$ if A is true, and $\mathbb{1}[x] = 0$ otherwise.

As before, choose h from $\mathcal{H} = \{h(.; w) : w \in \mathbb{R}^d\}$.

Choose continuous (smooth) loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

Expected Risk:

$$\min_{w \in \mathbb{R}^d} \left\{ R(w) := \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} \ell(h(x; w), y) dP(x, y) = \mathbb{E}[\ell(h(x; w), y)] \right\}$$

Empirical Risk:

$$\min_{w \in \mathbb{R}^d} \left\{ R_n(w) := \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i) \right\}$$

Minimizing the Empirical Risk

$$\min_{w \in \mathbb{R}^d} \left\{ F(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right\},$$

where $f_i(w) = \ell(h(x_i; w), y_i)$.

So, we have a finite sum optimization problem to solve.

Generalization: Train and Test Data

Sample points: $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^{d_x} \times \mathbb{R}$

$A \subset \{1, \ldots, n\}$

$B = \{1, \ldots, n\}/A$

Training data

$\{(x_i, y_i)\}_{i \in A}$

Test data

$\{(x_i, y_i)\}_{i \in B}$

Solve: $\boxed{\min_w \frac{1}{|A|} \sum_{i \in A} f_i(w)}$ Training Loss Function

Check: $\boxed{\min_w \frac{1}{|B|} \sum_{i \in B} f_i(w)}$ Test Loss Function

SGD Method

Given the training points $\{(x_1, y_1), \dots, (x_n, y_n)\} \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, solve

$$\min_{w \in \mathbb{R}^d} \left\{ f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right\},$$

where f_i corresponds to the loss function evaluated at the training point (x_i, y_i).

Algorithm 2: Stochastic Gradient Descent (SGD)

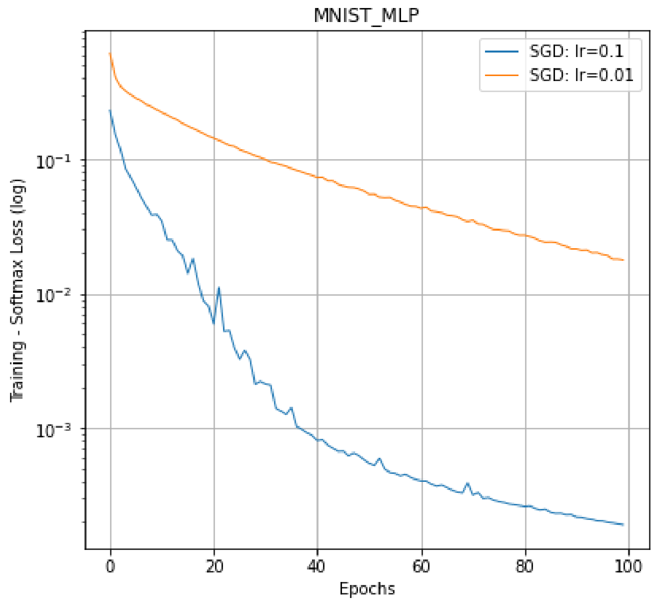
1 **Input:** Initial point w_1, max iteration number T, and the stepsizes $\{\alpha_k\}_{k=1}^{T}$

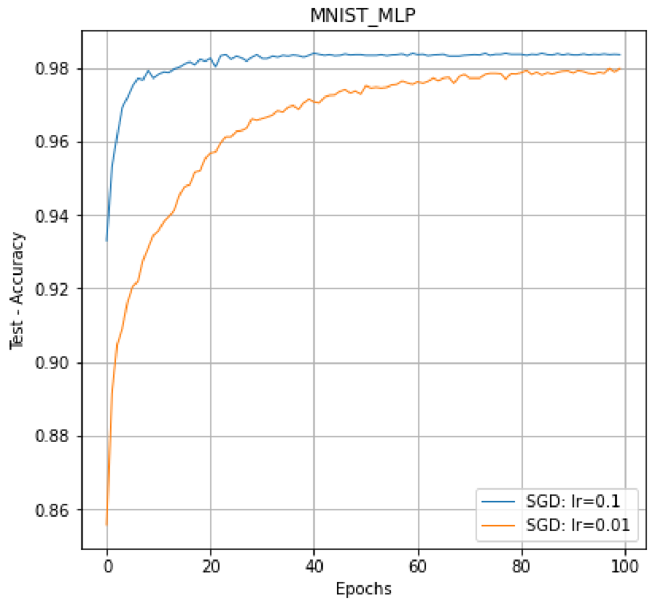
2 **for** $k = 1, \dots, T$ **do**

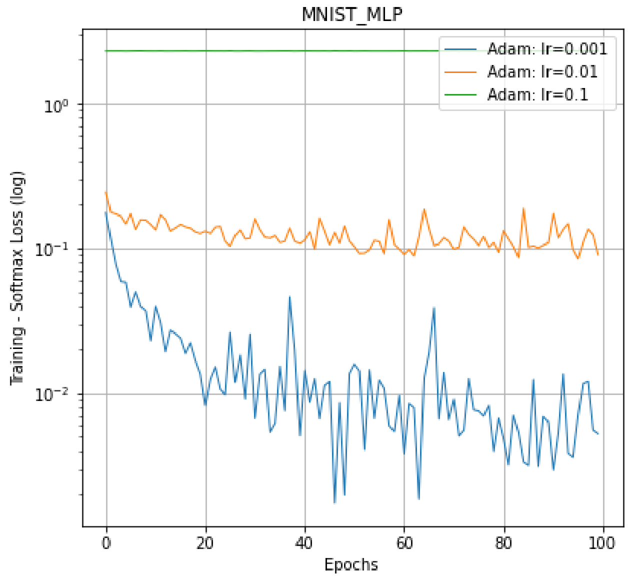
3 Choose $i_k \in \{1, \dots, n\}$ randomly w.r.t. uniform distribution;

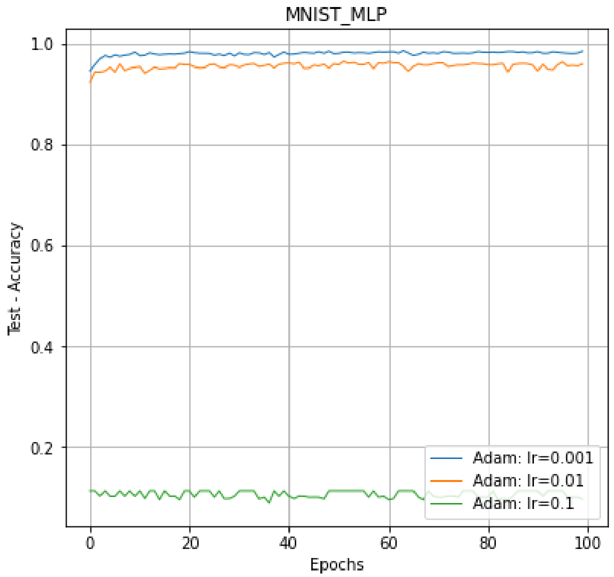
4 $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$

Robustness: SGD, Adam









Stochastic Line Search*

Armijo line search: Given the maximum stepsize α_{max}, find the biggest stepsize $0 < \alpha_k \leq \alpha_{max}$ which satisfies the stochastic Armijo condition

$$f_{i_k}(w_k - \alpha_k \nabla f_{i_k}(w_k)) - f_{i_k} \leq c\ \alpha_k \|\nabla f_{i_k}(w_k)\|^2,$$

where $c > 0$ is a hyper-parameter.

Algorithm 3: Stochastic Line Search (SLS)

Input: Initial point w_1, max iteration number T, maximum stepsize α_{max}, c

for $k = 1, \ldots, T$ **do**

  Choose $i_k \in \{1, \ldots, n\}$ randomly w.r.t. uniform distribution;

  Find α_k with Armijo line search;

  $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$

*Vaswani et al., Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates, NeurIPS 2019.

Stochastic Line Search

Backtracking line search: Start with $\alpha_k = \alpha_{max}$ and repeatedly set $\alpha_k = \alpha_k.\gamma$ until the stochastic Armijo condition is satisfied. Here, $0 < \gamma < 1$ is a hyper-parameter.

Algorithm 4: Stochastic Line Search (SLS)

Input: Initial point w_1, max iteration number T, and maximum stepsize α_{max}, c, γ, N

for $k = 1, \ldots, T$ **do**

- Choose $i_k \in \{1, \ldots, n\}$ randomly w.r.t. uniform distribution;
- Find α_k with Armijo line search and **backtracking**;
- $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$;
- $\alpha_{max} = \alpha_k$;
- **RESET** α_{max} **if** $N \mid k$

Stochastic Line Search with Model Building*

$$\min_{x \in \Re^n} \quad f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

TRIAL STEP $x_k^t = x_k + s_k^t$

SGD $s_k^t = -\alpha_k g_k$

$$f_k^t \leq f_k - c\,\alpha_k \|g_k\|^2$$

$f(x_k^t)$

$f(x_k)$

CONSTANT

✓ $x_{k+1} = x_k^t$

✗ MODEL BUILDING → s_k

$x_{k+1} = x_k + s_k$

*Birbil, Martin, Onay, and Öztoprak, Bolstering Stochastic Gradient Descent with Model Building, 2021.

$$x_k^t = x_k + s_k^t$$

$$s_k^t = -\alpha_k g_k$$

$$f_k^t \leq f_k - c\,\alpha_k \|g_k\|^2$$

✓

$x_{k+1} = x_k^t$

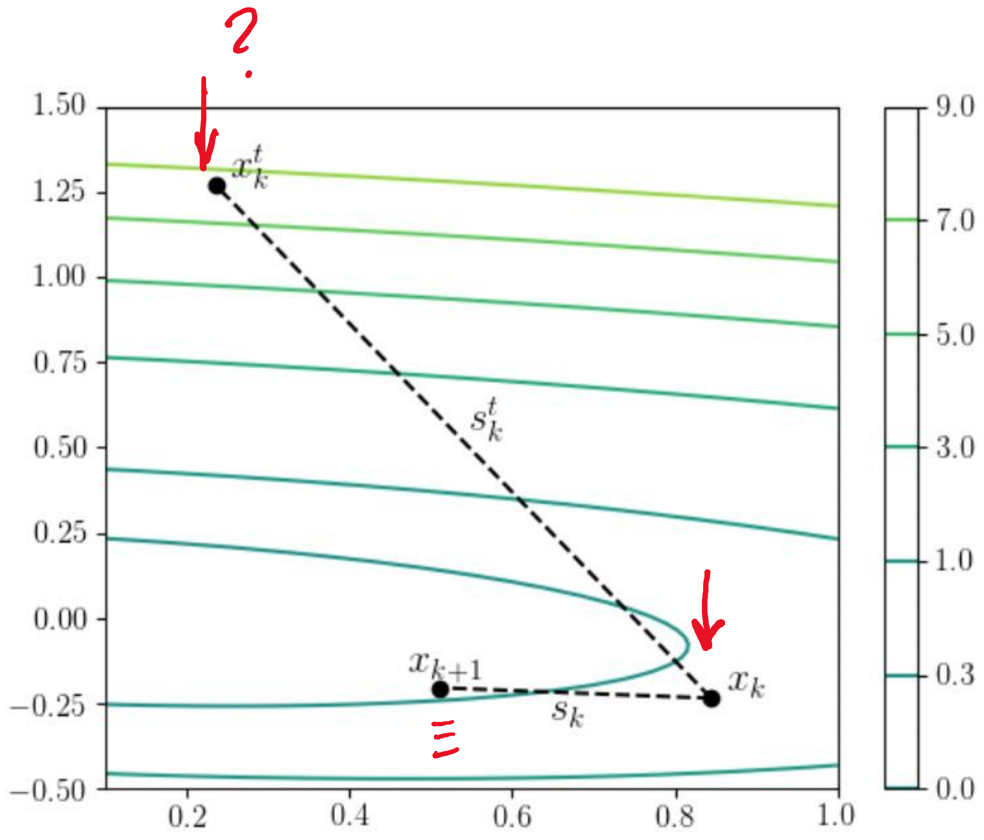
✗

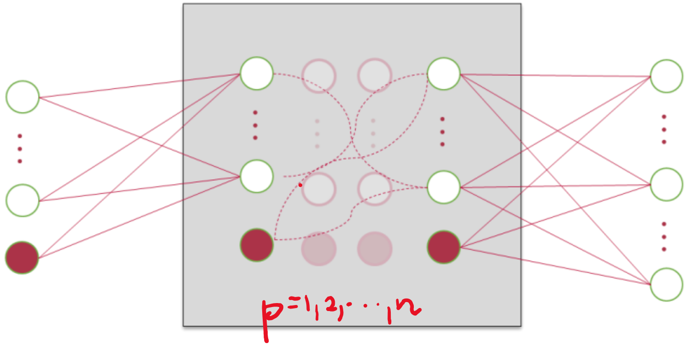
MODEL BUILDING

↓

s_k

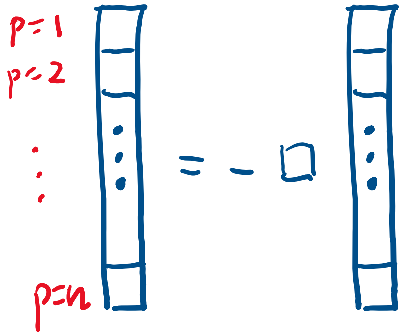
$x_{k+1} = x_k + s_k$

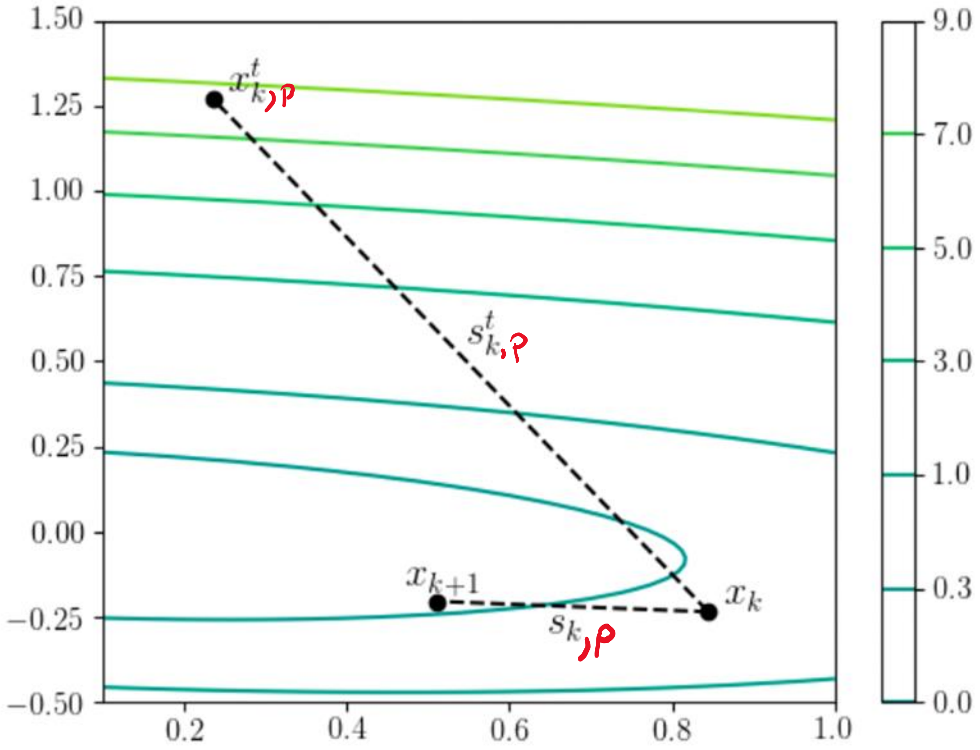




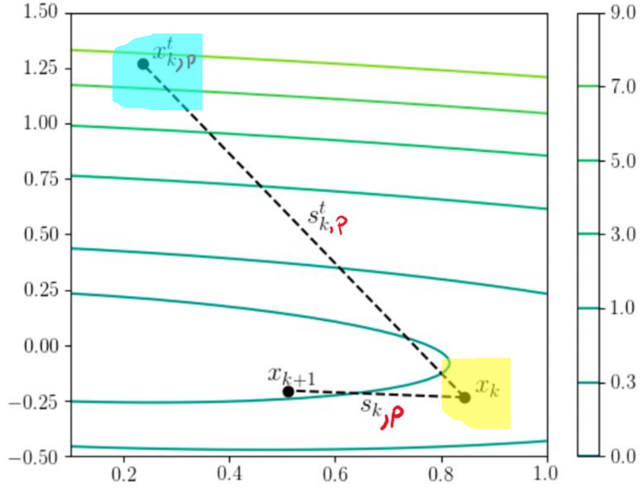
$$x^t_{k,p} = x_{k,p} + s^t_{k,p}$$

$$s^t_{k,p} = -\alpha_k g_{k,p}$$





MODEL BUILDING



$$m^t_{k,p}(s) := \alpha^0_{k,p}(s)\, l^0_{k,p}(s) + \alpha^t_{k,p}(s)\, l^t_{k,p}(s - s^t_{k,p})$$

$$l^0_{k,p}(s) := f_{k,p} + g_{k,p}^\top s$$

$$l^t_{k,p}(s - s^t_{k,p}) := f^t_{k,p} + (g^t_{k,p})^\top (s - s^t_{k,p})$$

$$\alpha^0_{k,p}(s) = \frac{(s - s^t_{k,p})^\top (-s^t_{k,p})}{(-s^t_{k,p})^\top (-s^t_{k,p})}$$

$$\alpha^t_{k,p}(s) = \frac{s^\top s^t_{k,p}}{(s^t_{k,p})^\top s^t_{k,p}}$$

* Figen Öztoprak and Ş İlker Birbil. An alternative globalization strategy for unconstrained optimization. *Optimization*, 67(3):377–392, 2018.

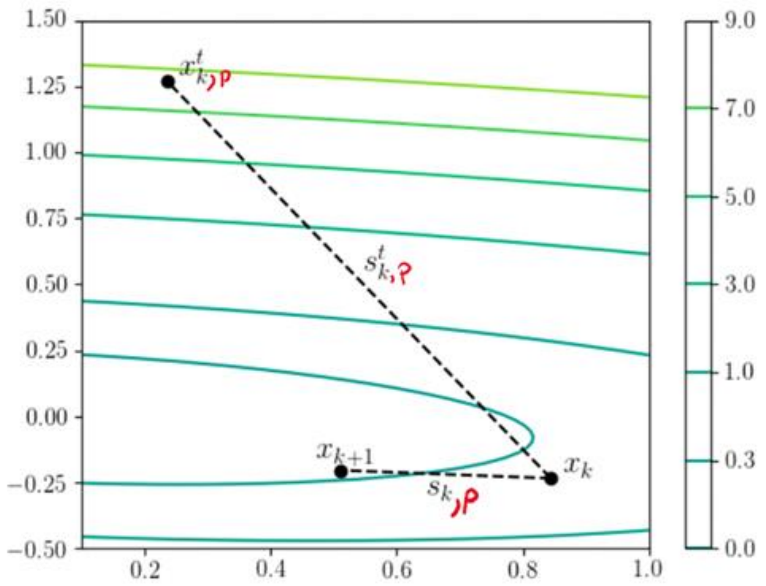
MINIMIZE $\quad m^t_{k,p}(s) := \alpha^0_{k,p}(s) l^0_{k,p}(s) + \alpha^t_{k,p}(s) l^t_{k,p}(s - s^t_{k,p})$

SUBJECT TO $\quad \|s\|^2 + \|s - s^t_{k,p}\|^2 \leq \|s^t_{k,p}\|^2$

$$\downarrow$$

$s_{k,p}$

(ANALYTICAL SOLUTION)



$$s_k^{t+1} = c_g(\sigma) g_k + c_y(\sigma) y_k^t + c_s(\sigma) s_k^t,$$

where

$$c_g(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma}, \quad c_y(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma\theta}[-((y_k^t)^\intercal s_k^t + 2\sigma)((s_k^t)^\intercal g_k) + \|s_k^t\|^2((y_k^t)^\intercal g_k)],$$

$$c_s(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma\theta}[-((y_k^t)^\intercal s_k^t + 2\sigma)((y_k^t)^\intercal g_k) + \|y_k^t\|^2((s_k^t)^\intercal g_k)],$$

with

$$\theta = \left((y_k^t)^\intercal s_k^t + 2\sigma\right)^2 - \|s_k^t\|^2 \|y_k^t\|^2,$$

and

$$\sigma = \tfrac{1}{2}\left(\|s_k^t\|\left(\|y_k^t\| + \frac{1}{\eta}\|g_k\|\right) - (y_k^t)^\intercal s_k^t\right).$$

Algorithm 1: SMB: Stochastic Model Building

Input: $x_1 \in \mathbb{R}^n$, stepsizes $\{\alpha_k\}_{k=1}^{T}$, mini-batch sizes $\{m_k\}_{k=1}^{T}$,$c > 0$, and α_{max} satisfying (8)

for $k = 1, \dots, T$ **do**

  $f_k = f(x_k, \xi_k)$, $g_k = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i})$;

  $s_k^t = -\alpha_k g_k$;

  $x_k^t = x_k + s_k^t$;

  $f_k^t = f(x_k^t, \xi_k)$, $g_k^t = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k^t, \xi_{k,i})$;

  if $f_k^t \leq f_k - c\ \alpha_k \|g_k\|^2$ **then**

    $x_{k+1} = x_k^t$;

  else

    for $p = 1, \dots, r$ **do**

      $y_{k,p} = g_{k,p}^t - g_{k,p}$;

      $s_{k,p} = c_{g,p}(\delta) g_{k,p} + c_{y,p}(\delta) y_{k,p} + c_{s,p}(\delta) s_{k,p}^t$;

    $x_{k+1} = x_k + s_k$ with $s_k = (s_{k,p_1}, \dots, s_{k,p_r})$;

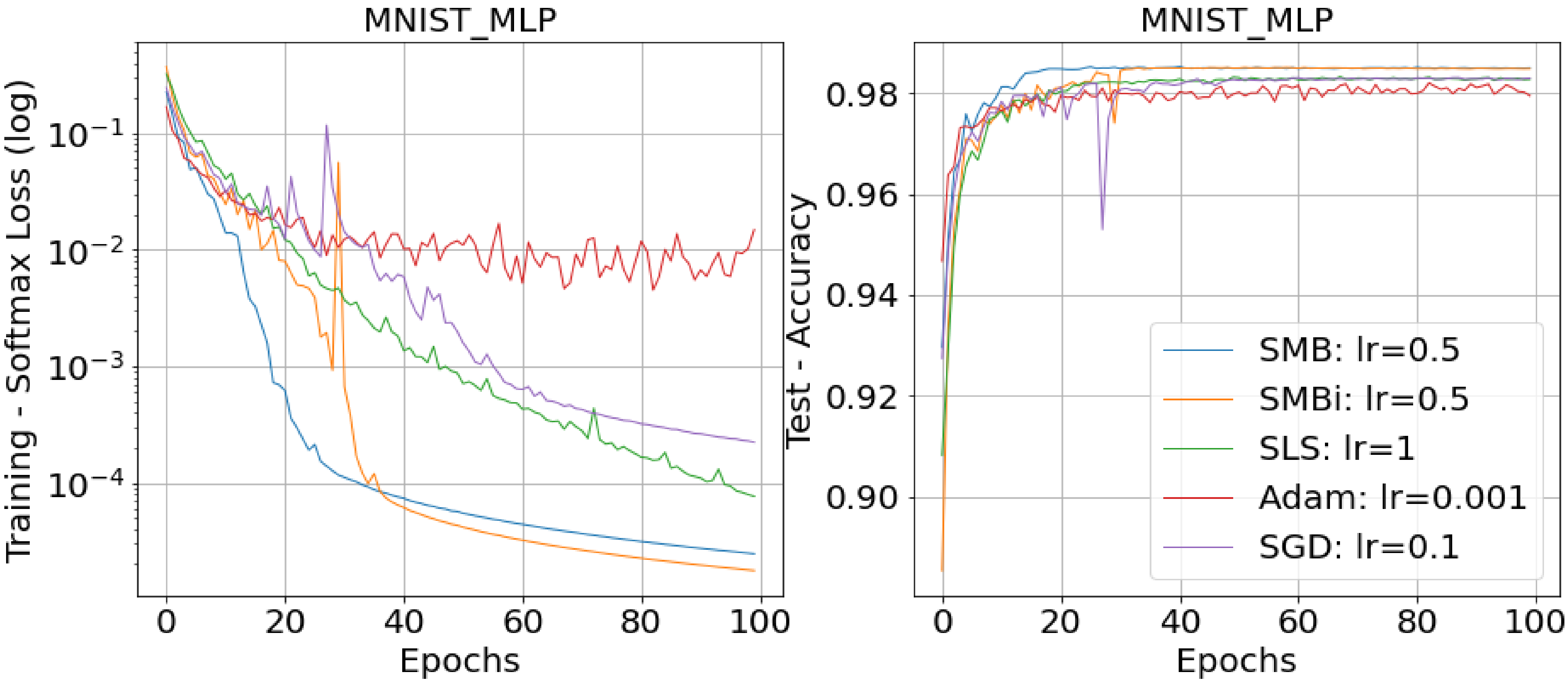
$$x_{k+1} = x_k - \alpha_k H_k g_k, \quad (s_k = -\alpha_k H_k g_k)$$

$$H_k = \begin{bmatrix} H_{k,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & H_{k,n} \end{bmatrix}$$

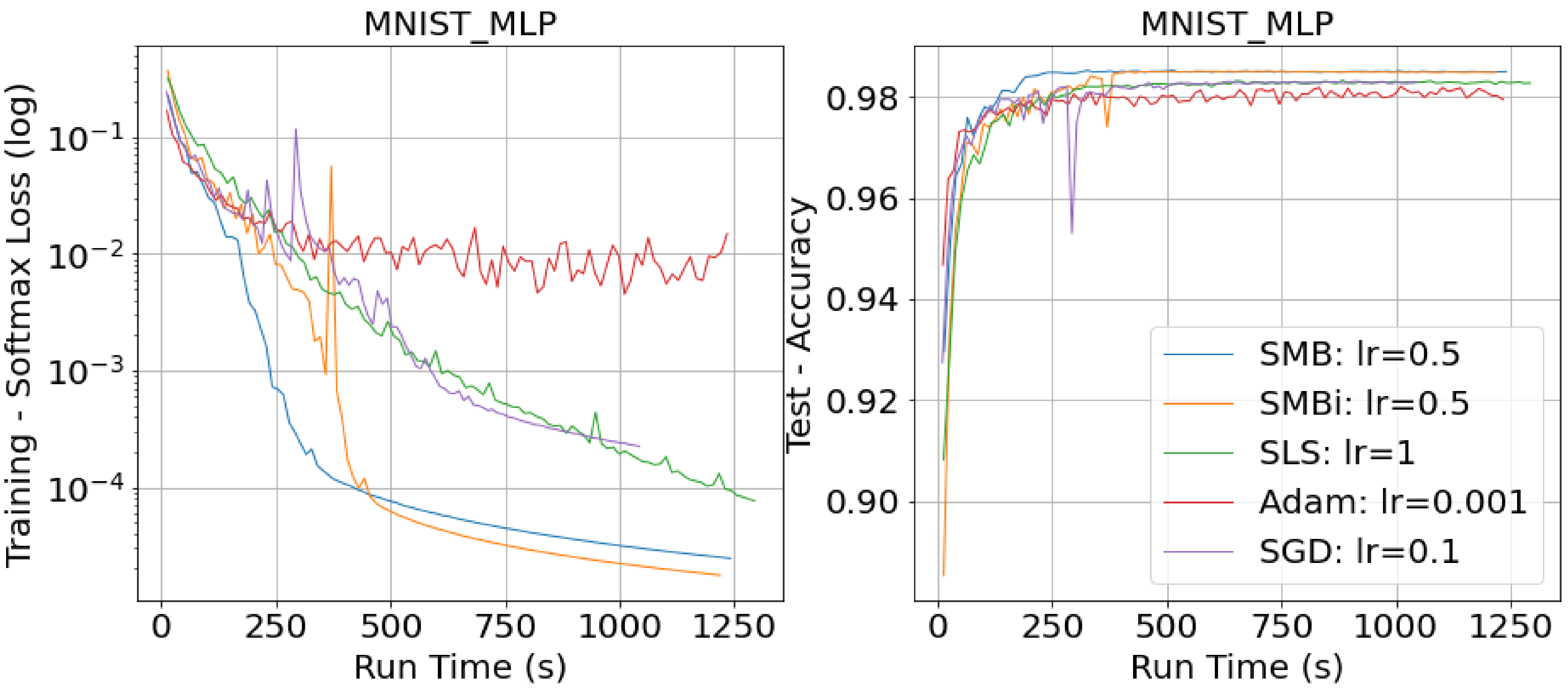
$$H_{k,p} = \frac{\|g_{k,p}\|^2}{\sigma_p \gamma_p} \left[\gamma_p I + \beta_p y_{k,p} g_{k,p}^\top + \|g_{k,p}\|^2 y_{k,p} y_{k,p}^\top + \beta_p g_{k,p} y_{k,p}^\top + \|y_{k,p}\|^2 g_{k,p} g_{k,p}^\top \right],$$

where

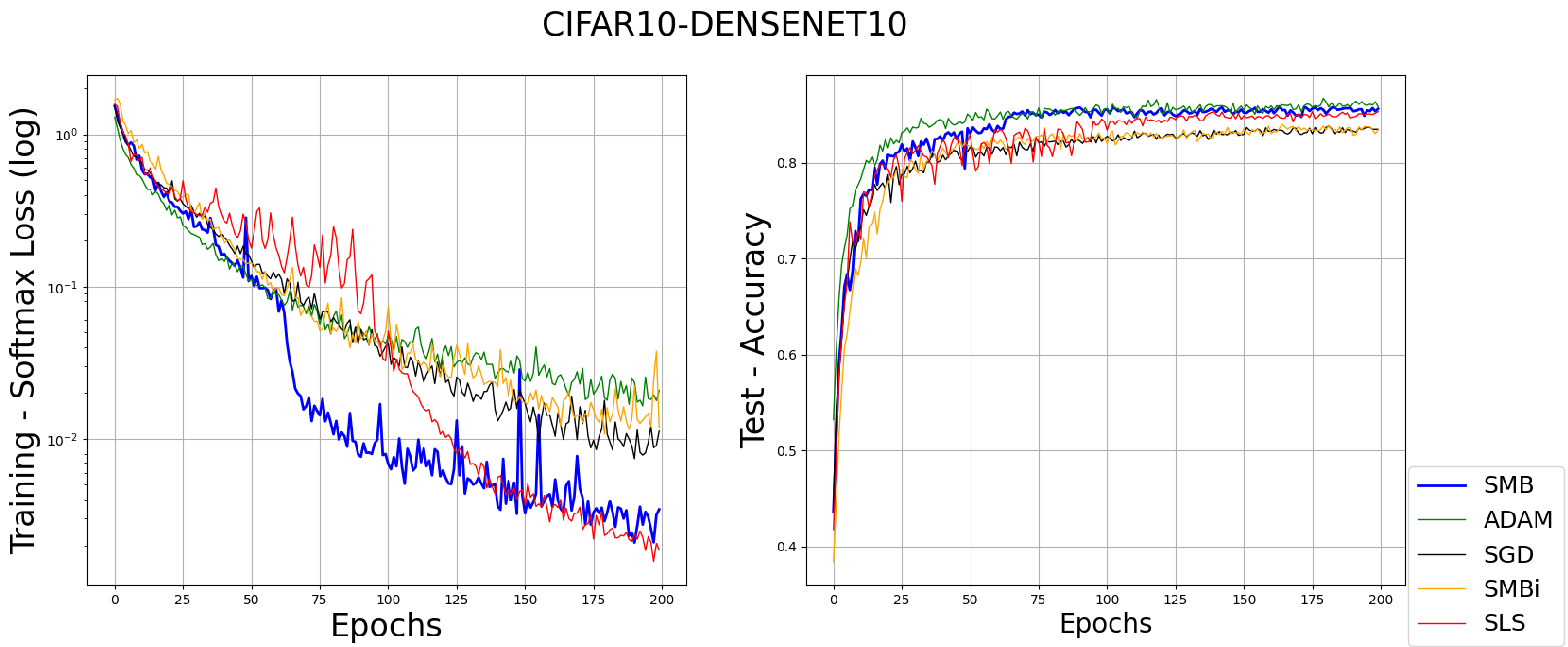
$$\sigma_p = \|g_{k,p}\| \|y_{k,p}\| + \frac{1}{\eta} \|g_{k,p}\|^2 + y_{k,p}^\top g_{k,p}, \;\; \beta_p = \sigma_p - y_{k,p}^\top g_{k,p}, \;\; \text{and } \gamma_p = (\beta_p^2 - \|g_{k,p}\|^2 \|y_{k,p}\|^2).$$

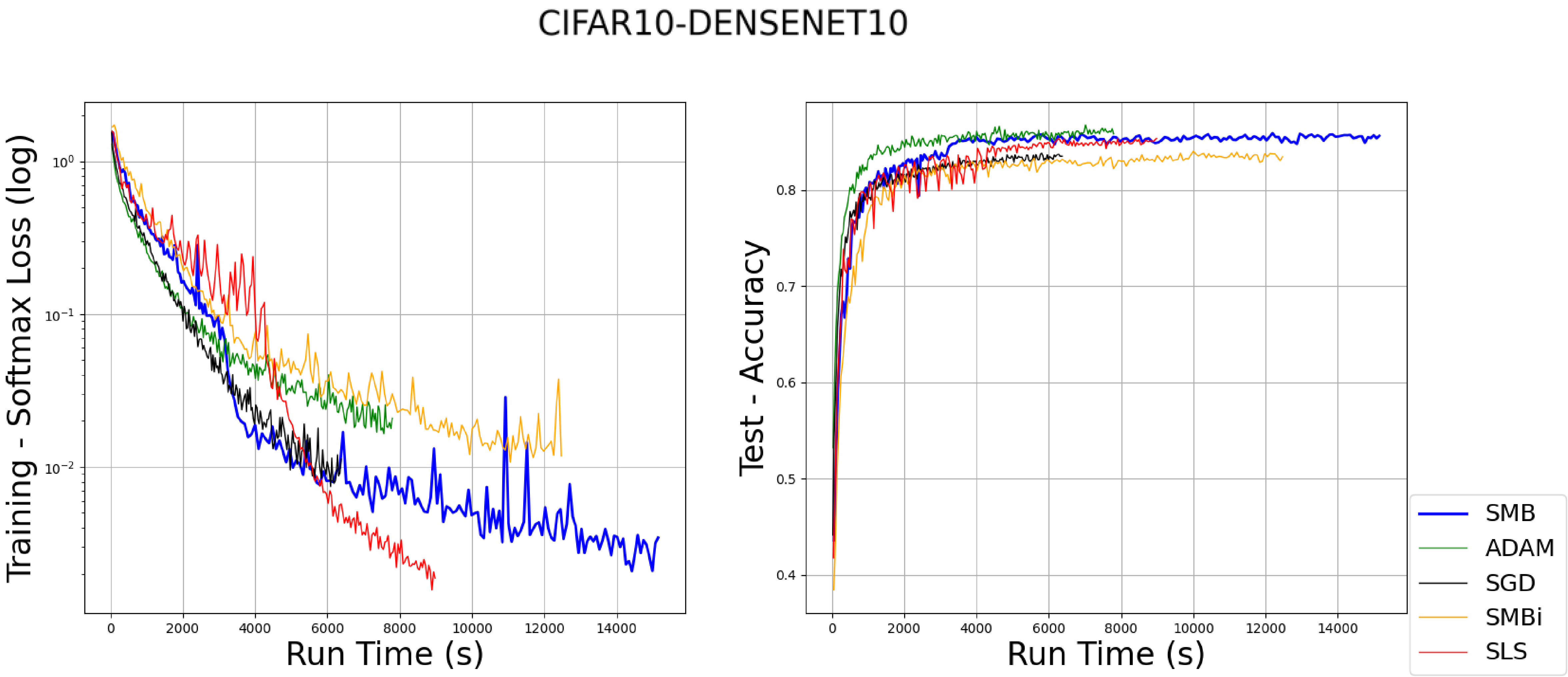


MNIST_MLP
Training - Softmax Loss (log)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
20
40
60
80
100
Epochs
MNIST_MLP
Test - Accuracy
0.98
0.96
0.94
0.92
0.90
SMB: lr=0.5
SMBi: lr=0.5
SLS: lr=1
Adam: lr=0.001
SGD: lr=0.1
0
20
40
60
80
100
Epochs

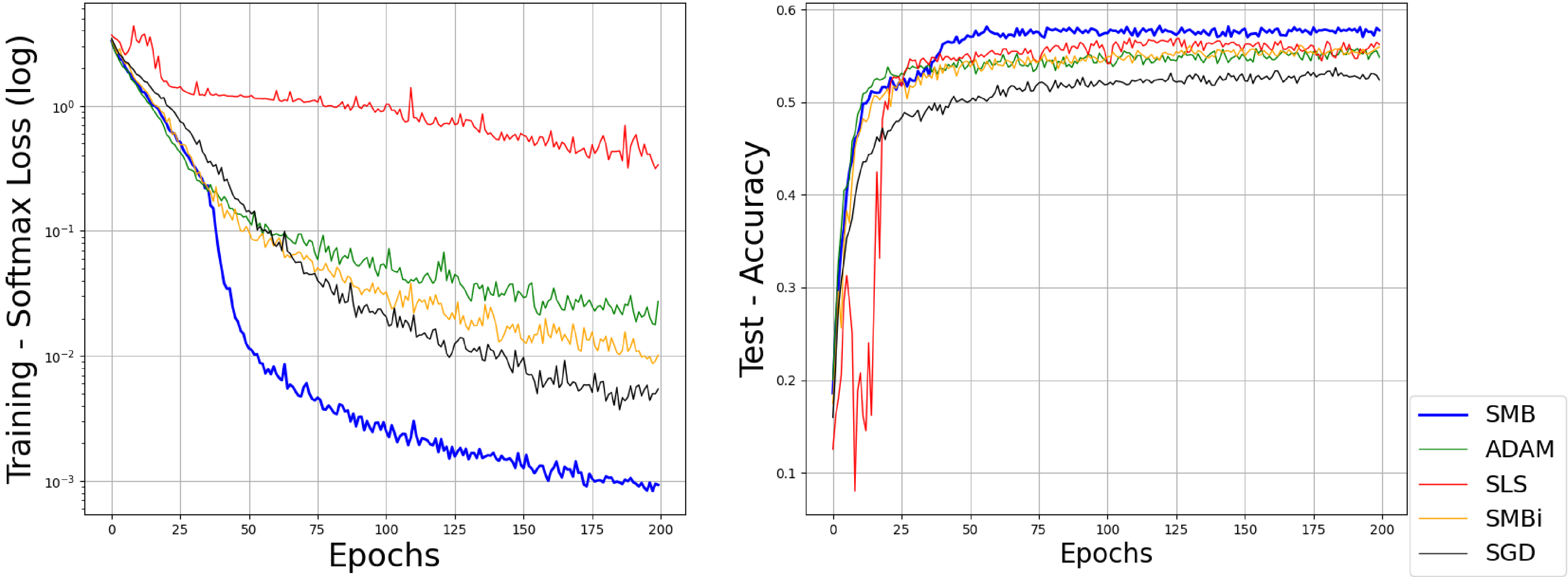


MNIST_MLP
Training - Softmax Loss (log)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
250
500
750
1000
1250
Run Time (s)
MNIST_MLP
Test - Accuracy
0.98
0.96
0.94
0.92
0.90
0
250
500
750
1000
1250
Run Time (s)
SMB: lr=0.5
SMBi: lr=0.5
SLS: lr=1
Adam: lr=0.001
SGD: lr=0.1

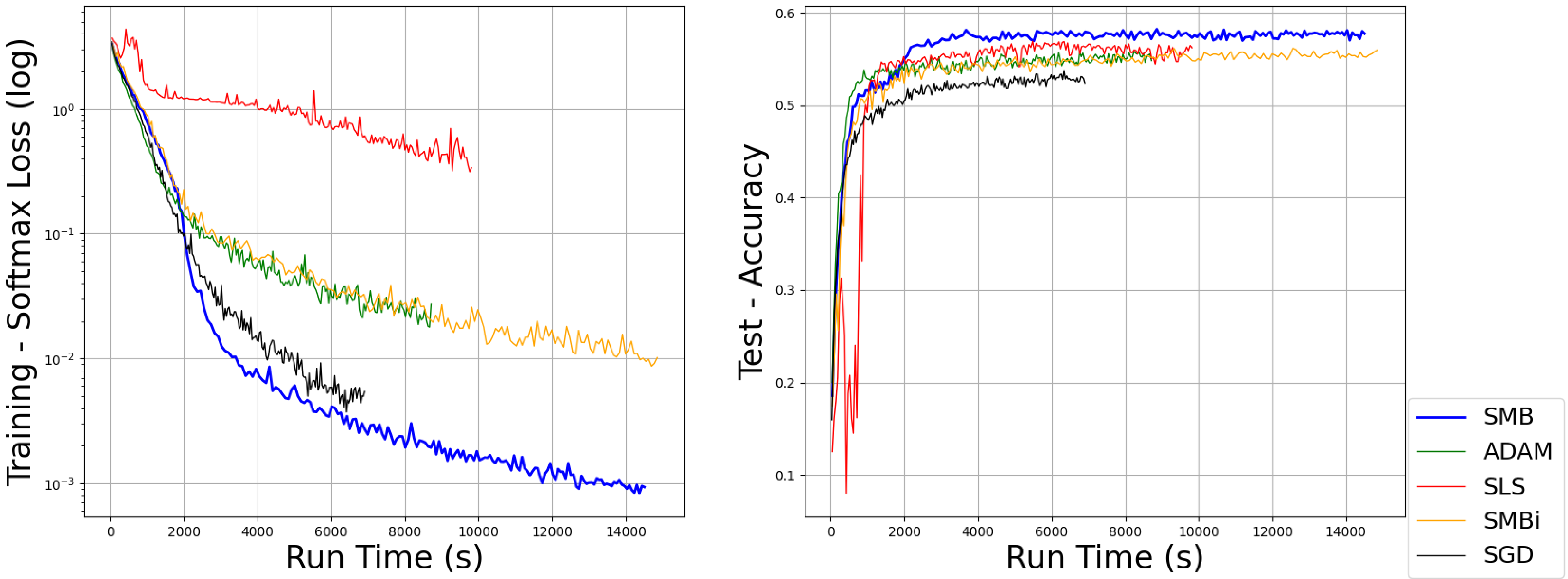

CIFAR10-DENSENET10
Training - Softmax Loss (log)
10^0
10^{-1}
10^{-2}
0
25
50
75
100
125
150
175
200
Epochs
Test - Accuracy
0.4
0.5
0.6
0.7
0.8
0
25
50
75
100
125
150
175
200
Epochs
SMB
ADAM
SGD
SMBi
SLS


CIFAR10-DENSENET10
Training - Softmax Loss (log)
10^0
10^{-1}
10^{-2}
0
2000
4000
6000
8000
10000
12000
14000
Run Time (s)
Test - Accuracy
0.8
0.7
0.6
0.5
0.4
0
2000
4000
6000
8000
10000
12000
14000
Run Time (s)
SMB
ADAM
SGD
SMBi
SLS

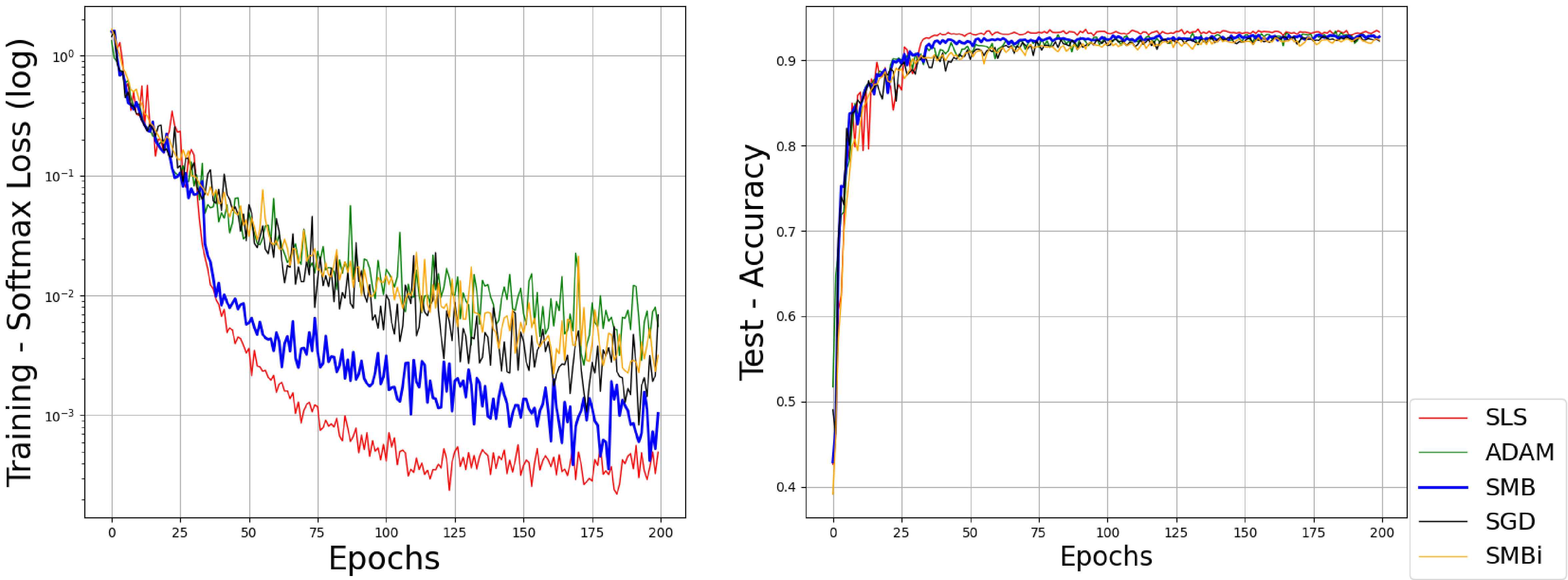
CIFAR100-DENSENET100



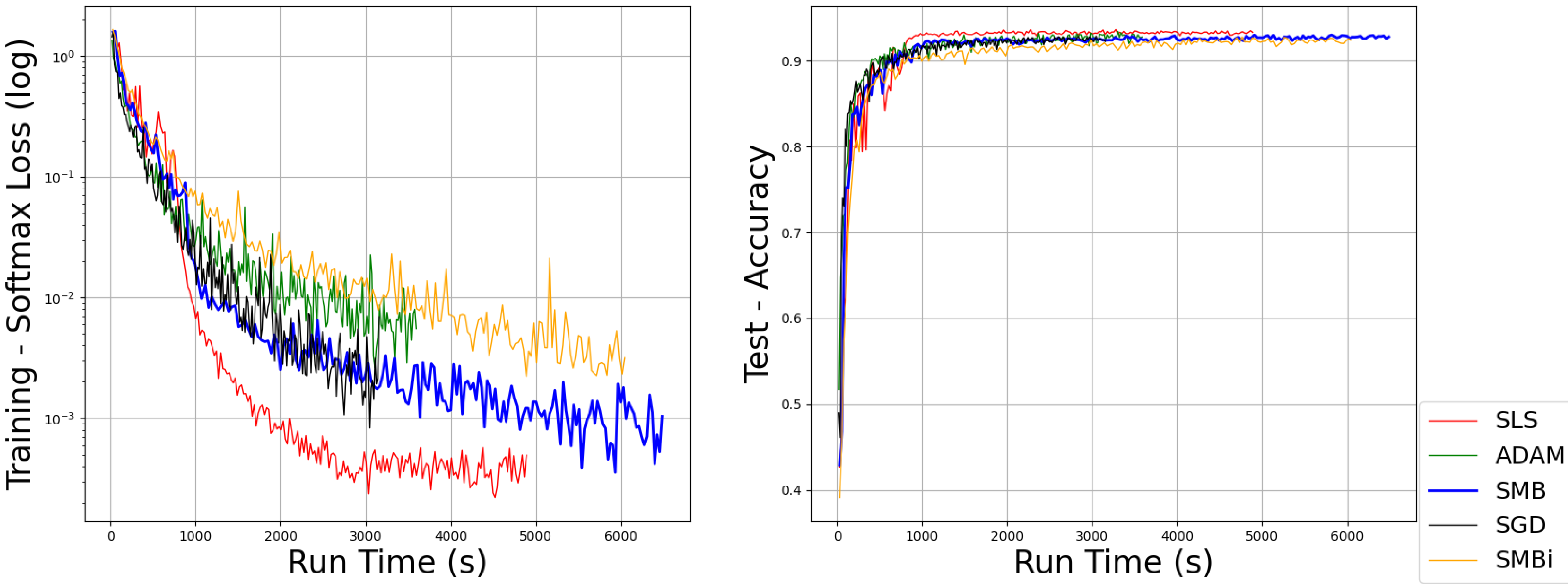
CIFAR100-DENSENET100



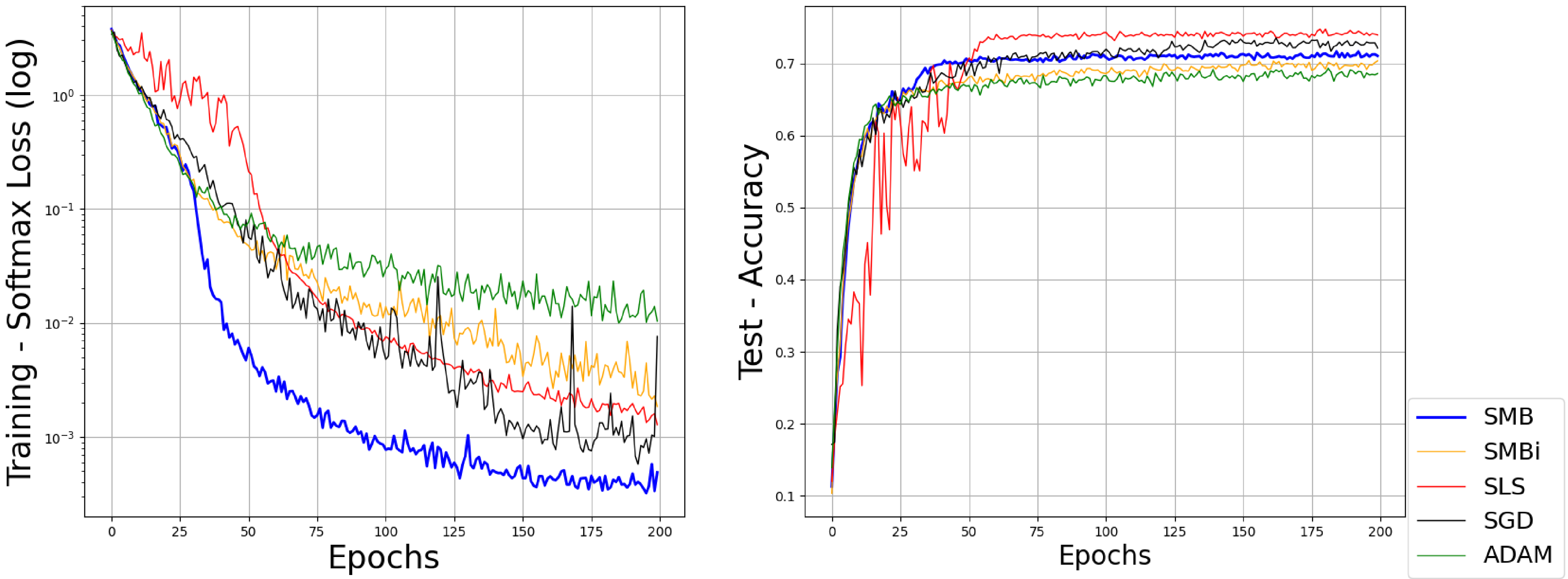
CIFAR10-RESNET34_10



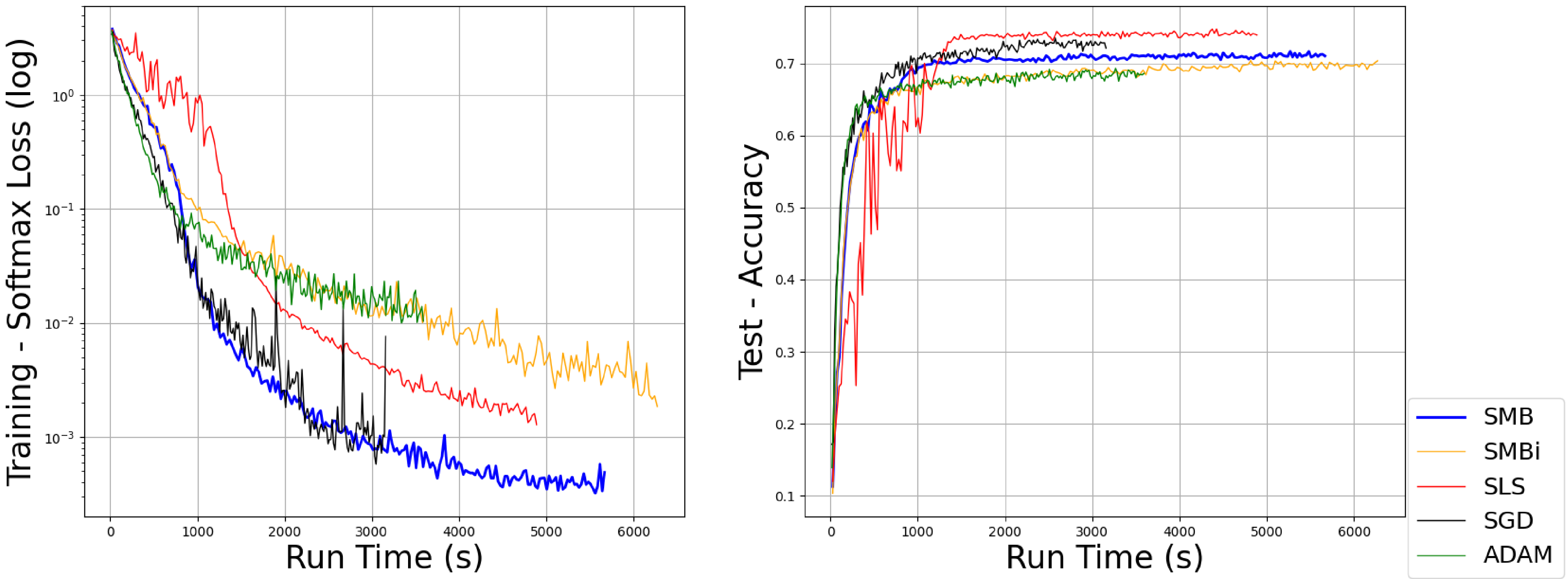
CIFAR10-RESNET34_10

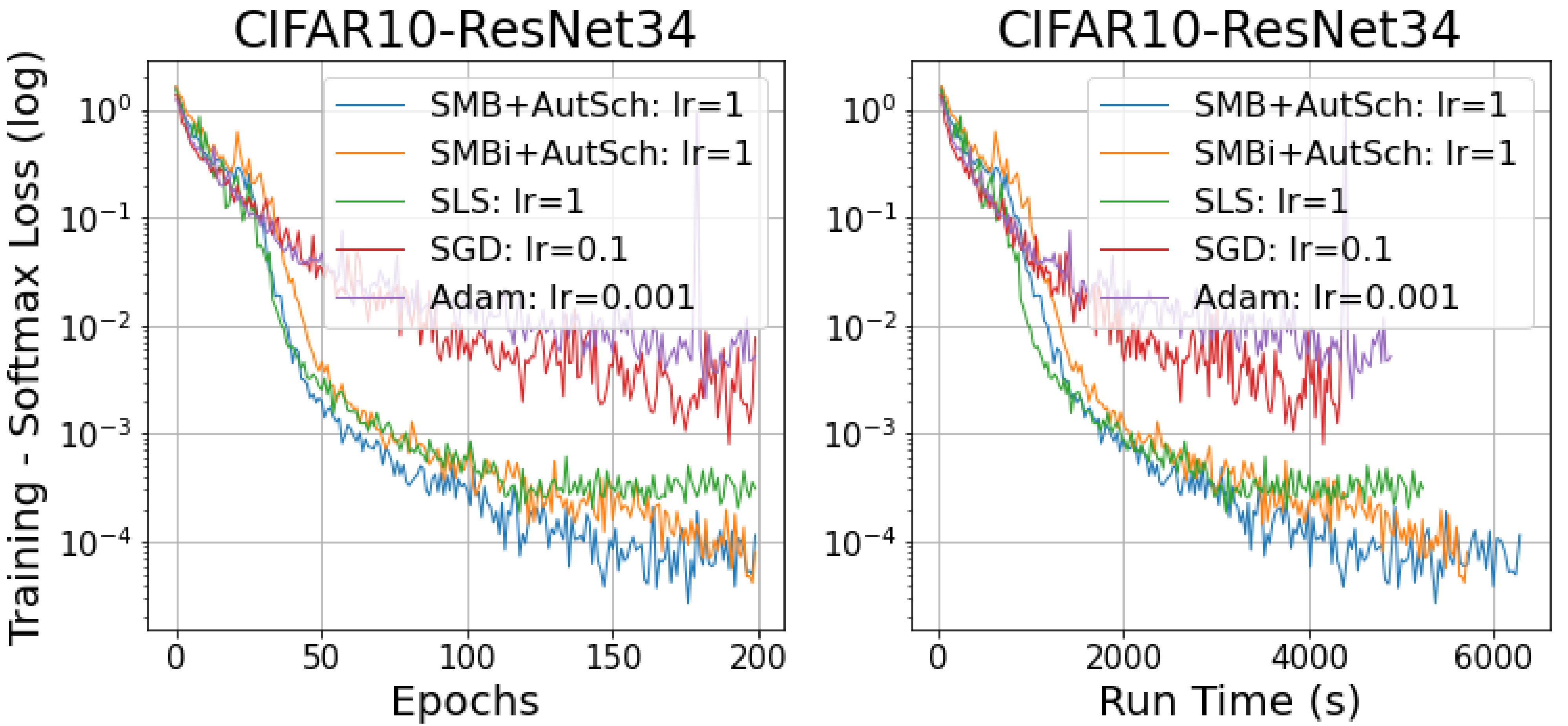


CIFAR100-RESNET34_100

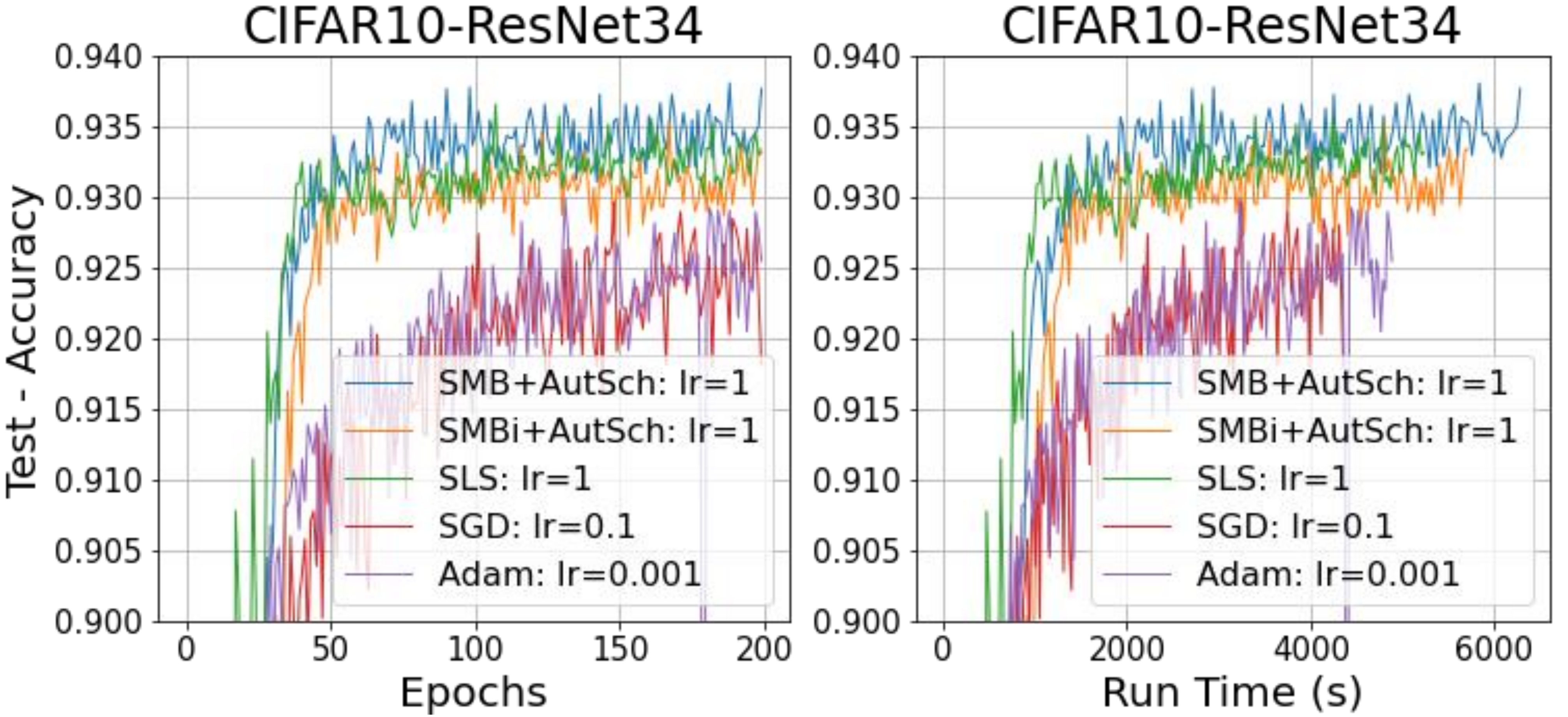


CIFAR100-RESNET34_100

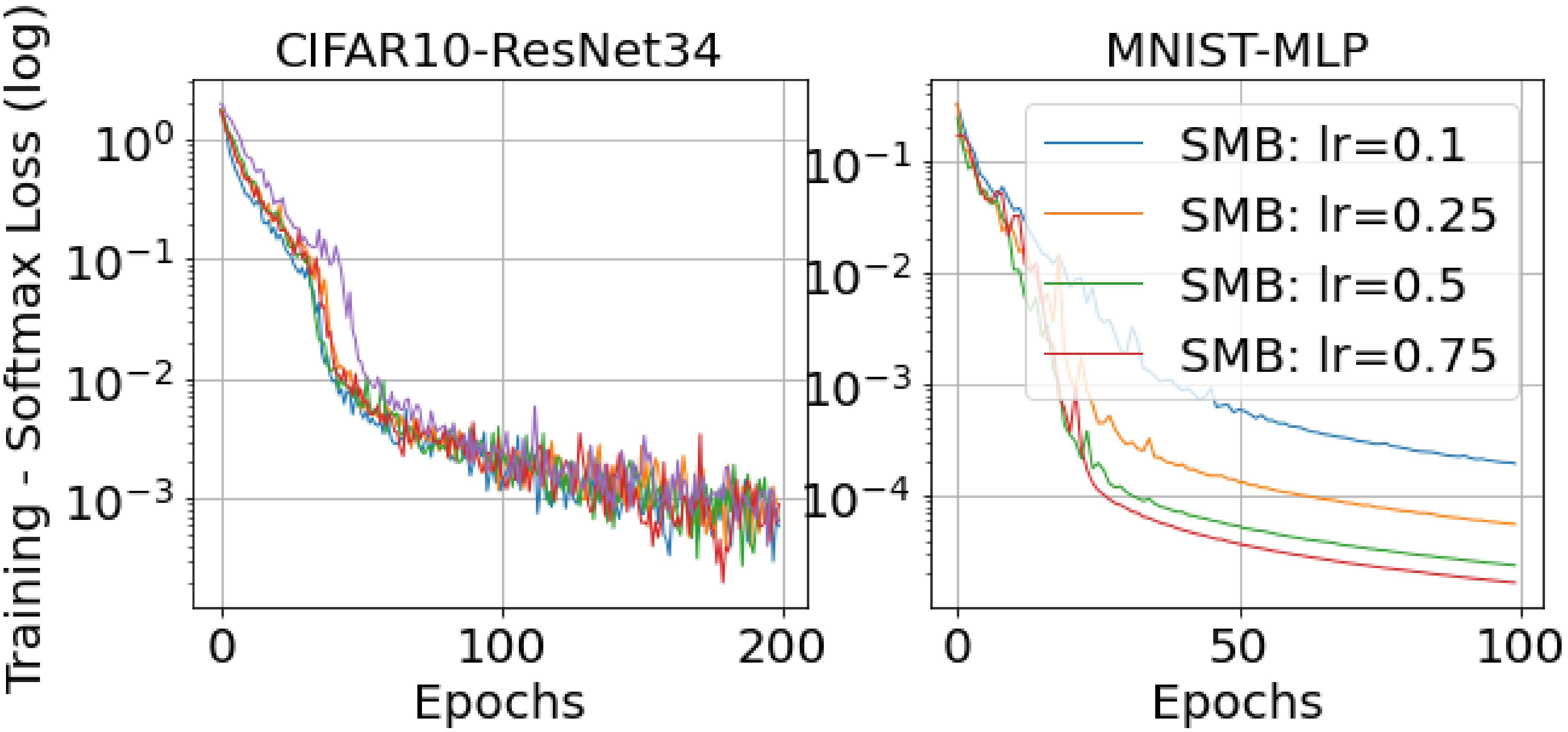




CIFAR10-ResNet34
CIFAR10-ResNet34
Training - Softmax Loss (log)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
50
100
150
200
Epochs
0
2000
4000
6000
Run Time (s)
SMB+AutSch: lr=1
SMBi+AutSch: lr=1
SLS: lr=1
SGD: lr=0.1
Adam: lr=0.001



CIFAR10-ResNet34
CIFAR10-ResNet34
Test - Accuracy
0.940
0.935
0.930
0.925
0.920
0.915
0.910
0.905
0.900
0
50
100
150
200
Epochs
0
2000
4000
6000
Run Time (s)
SMB+AutSch: lr=1
SMBi+AutSch: lr=1
SLS: lr=1
SGD: lr=0.1
Adam: lr=0.001



Training - Softmax Loss (log)
CIFAR10-ResNet34
MNIST-MLP
SMB: lr=0.1
SMB: lr=0.25
SMB: lr=0.5
SMB: lr=0.75
Epochs
Epochs

Paper: https://arxiv.org/abs/2111.07058

Codes: https://github.com/sibirbil/SMB

Thanks!