Shrinkage Estimation in High-Dimensional Deep Learning: A Stein-Rule Approach to Stochastic Optimization

M.Arashi
Department of Statistics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, P. O. Box 1159, Mashhad 91775, Iran

Abstract

Deep neural networks operate in parameter spaces of dimension $p \gg 3$, a regime where the classical Maximum Likelihood Estimator (MLE) is known to be inadmissible under quadratic loss. Despite this, standard Stochastic Gradient Descent (SGD) treats the mini-batch gradient as an unbiased, unrestricted estimator of the true gradient. In this paper, we bridge the gap between classical shrinkage estimation and modern deep learning. We propose a Stein-Rule Gradient Estimator that shrinks the noisy stochastic gradient toward a stable Restricted Estimator (the historical momentum) based on a dynamic hypothesis test of signal quality. We further introduce an adaptive noise variance estimator derived from the second moments of the Adam optimizer. The resulting algorithm, Stein-Rule Adam (SR-Adam), theoretically guarantees lower asymptotic distributional risk in gradient estimation, offering a rigorous statistical alternative to heuristic regularization.

I. Introduction

The central tenet of classical frequentist statistics, established by Stein (1956) [1] and refined by James and Stein (1961) [2], is that in dimensions $p \geq 3$, the ordinary Maximum Likelihood Estimator (MLE) is inadmissible. That is, there exists a biased estimator, termed the shrinkage estimator, that achieves a strictly lower Mean Squared Error (MSE) universally across the parameter space. Modern Deep Learning (DL) is inherently a high-dimensional estimation problem, often exceeding $p = 10^6$. Yet, the dominant optimization paradigm, Stochastic Gradient Descent (SGD), relies on the unbiased estimation of the gradient vector $\nabla \mathcal{L}(\boldsymbol{\theta})$ via mini-batches. While unbiasedness is a desirable property in low dimensions, in high dimensions it subjects the optimization trajectory to excessive variance, leading to slow convergence and poor generalization. Current solutions to this variance problem, such as momentum or weight decay (L_2 regularization), are heuristic. Momentum is a linear smoothing technique, while weight decay imposes a static zero-mean prior. Neither approach adapts dynamically to the statistical significance of the gradient signal. In this work, we formalize the neural network optimization step as a *point estimation problem*. We apply the theory of Preliminary Test (PT) and Stein-Rule (SR) estimation [3] to construct a gradient estimator that dynamically trades bias for variance reduction. We demonstrate that the Adam optimizer's internal state provides sufficient statistics to construct an adaptive shrinkage factor, leading to the proposed **SR-Adam** algorithm.

II. Preliminaries: The Estimator Framework

Let $\boldsymbol{\theta} \in \mathbb{R}^p$ denote the parameter vector of a neural network, where p is large. We aim to minimize a loss function $J(\boldsymbol{\theta}) = \mathbb{E}_{x \sim \mathcal{D}}[L(x, \boldsymbol{\theta})]$. At time step t, we observe a mini-batch gradient $\mathbf{g}_t$, which we treat as the *Unrestricted Estimator* (UE) of the true gradient $\nabla J(\boldsymbol{\theta}_t)$:

$$\mathbf{g}_t = \nabla J(\boldsymbol{\theta}_t) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_p)$$

Standard SGD uses $\mathbf{g}_t$ directly to update $\boldsymbol{\theta}$. We posit the existence of a *Restricted Estimator* (RE), $\tilde{\mathbf{g}}_t$, which embodies prior knowledge or stability. In the context of optimization with momentum, the exponential moving average of past gradients serves as a natural proxy for the "stable" direction of descent:

$$\tilde{\mathbf{g}}_t = \mathbf{m}_{t-1}$$

where $\mathbf{m}_{t-1}$ is the first moment estimate from the previous step. The objective is to find an estimator $\hat{\mathbf{g}}_t$ that minimizes the weighted quadratic risk:

$$R(\hat{\mathbf{g}}_t) = \mathbb{E}\left[\|\hat{\mathbf{g}}_t - \nabla J(\boldsymbol{\theta}_t)\|^2\right]$$

III. Proposed Method: The Stein-Rule Gradient

Under the James-Stein framework, we construct a shrinkage estimator $\mathbf{g}_t^S$ that shrinks the high-variance UE ($\mathbf{g}_t$) towards the low-variance RE ($\mathbf{m}_{t-1}$).

Corresponding author. Email: arashi@um.ac.ir

A. The Test Statistic

We define the test statistic D_n based on the squared Euclidean distance between the current observation and the historical trend:

$$D_n = \|\mathbf{g}_t - \mathbf{m}_{t-1}\|_2^2$$

This statistic essentially tests the hypothesis $H_0 : \nabla J(\boldsymbol{\theta}_t) = \mathbf{m}_{t-1}$. A large D_n suggests the current batch gradient deviates significantly from the established trajectory (high noise or distributional shift).

B. The Stein-Rule Estimator

The Stein-Rule estimator for the gradient is defined as:

$$\mathbf{g}_t^S = \mathbf{m}_{t-1} + \left(1 - \frac{(p-2)\sigma^2}{D_n}\right)(\mathbf{g}_t - \mathbf{m}_{t-1})$$

To ensure the shrinkage factor remains valid (non-negative), we adopt the Positive-Rule Stein Estimator:

$$\mathbf{g}_t^{S+} = \mathbf{m}_{t-1} + \left[1 - \frac{(p-2)\sigma^2}{\|\mathbf{g}_t - \mathbf{m}_{t-1}\|^2}\right]^+ (\mathbf{g}_t - \mathbf{m}_{t-1})$$

where $[z]^+ = \max(0, z)$.

IV. Adaptive Variance Estimation in Adam

A critical challenge in applying Eq. (III-B) is the unknown noise variance σ^2. In static regression, this is estimated via residual sum of squares. In stochastic optimization, we must estimate it online. The Adam optimizer maintains running estimates of the first moment $\mathbf{m}_t \approx \mathbb{E}[\mathbf{g}]$ and the second raw moment $\mathbf{v}_t \approx \mathbb{E}[\mathbf{g}^2]$. Using the identity $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$, we can approximate the element-wise variance of the gradient at step t:

$$\hat{\boldsymbol{\sigma}}_t^2 = \mathbf{v}_t - \mathbf{m}_t^2$$

To obtain a scalar variance estimate $\hat{\sigma}^2$ for the Stein correction (which assumes homoscedasticity across the layer for stability), we average over the dimension p:

$$\hat{\sigma}_{global}^2 = \frac{1}{p}\sum_{j=1}^{p}\left([\mathbf{v}_t]_j - ([\mathbf{m}_t]_j)^2\right)$$

Substituting $\hat{\sigma}_{global}^2$ into Eq. (III-B) yields a fully adaptive, hyperparameter-free shrinkage mechanism.

V. The SR-Adam Algorithm

We formally present the Stein-Rule Adam (SR-Adam) algorithm.

VI. Experimental Results

We present a controlled empirical study designed to isolate the optimizer's effect while keeping the rest of the training pipeline fixed. Experiments span CIFAR10 and CIFAR100 with three levels of label noise (0.0, 0.05, 0.1). Unless otherwise stated, the optimizer is the only varying component; backbone, preprocessing, epochs, batch size, and evaluation are held constant to ensure a fair comparison. We first summarize the backbone and the SR-Adam application strategy, then report per-epoch behavior and aggregated metrics across runs.

A. Model Architecture

We employ a lightweight SimpleCNN backbone for all experiments. Table I details the architecture, comprising two convolutional layers (32 and 64 filters with 3×3 kernels), followed by two fully connected layers (128 and 10 units). The total parameter count is 545,098, which keeps the model compact and allows us to focus on the optimizer's behavior without architectural confounds.

1) SR-Adam Application Strategy: SR-Adam applies Stein-rule shrinkage selectively: only to the convolutional layers where high-dimensional feature maps and noisy batch gradients make shrinkage estimation particularly valuable. Table II shows that SR-Adam manages 18,528 parameters (3.4% of the total) in Conv2d modules, while fully-connected layers use standard Adam updates. This selective grouping exploits the Stein-rule's strength in high-dimensional regimes while preserving the direct adaptive behavior for low-dimensional projection layers.

For SimpleCNN, SR-Adam applies Stein-rule shrinkage exclusively to convolutional layers, while standard Adam is used for fully-connected layers and bias terms. Table II details the parameter groups:

Algorithm 1 SR-Adam: Stein-Rule Adaptive Moment Estimation

Require: α: Learning rate
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $\boldsymbol{\theta}_0$: Initial parameter vector
Require: $\mathbf{m}_0 \leftarrow \mathbf{0}, \mathbf{v}_0 \leftarrow \mathbf{0}$
$t \leftarrow 0$
while $\boldsymbol{\theta}_t$ not converged **do**
  $t \leftarrow t + 1$
  Get gradients $\mathbf{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$
  if $t > 1$ **then**
    Compute noise variance: $\hat{\sigma}^2 \leftarrow \text{mean}(\mathbf{v}_{t-1} - \mathbf{m}_{t-1}^2)$
    Compute divergence: $D_n \leftarrow \|\mathbf{g}_t - \mathbf{m}_{t-1}\|^2$
    Compute shrinkage factor: $c_t \leftarrow \max\left(0, 1 - \frac{(p-2)\hat{\sigma}^2}{D_n}\right)$
    Stein Correction: $\hat{\mathbf{g}}_t \leftarrow \mathbf{m}_{t-1} + c_t(\mathbf{g}_t - \mathbf{m}_{t-1})$
  else
    $\hat{\mathbf{g}}_t \leftarrow \mathbf{g}_t$
  end if
  Update biased first moment: $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\hat{\mathbf{g}}_t$
  Update biased second raw moment: $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\hat{\mathbf{g}}_t^2$
  Compute bias-corrected moments $\hat{\mathbf{m}}_t, \hat{\mathbf{v}}_t$
  Update parameters: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}$
end while

TABLE I: SimpleCNN architecture: layer configuration, output shape, and parameters.

Layer	Kernel/Units	Output Shape	Parameters
Conv2d-1	3 → 32, 3×3	32×32×32	896
ReLU-1	—	32×32×32	0
MaxPool2d-1	2×2, stride=2	32×16×16	0
Conv2d-2	32 → 64, 3×3	64×16×16	18,496
ReLU-2	—	64×16×16	0
MaxPool2d-2	2×2, stride=2	64×8×8	0
Flatten	—	4,096	0
Linear-1	4,096 → 128	128	524,416
ReLU-3	—	128	0
Dropout (0.2)	—	128	0
Linear-2	128 → 10	10	1,290
Total	—	—	**545,098**

a) Why Stein-Rule on Convolutions Only?: This design balances theory, signal characteristics, and empirical behavior:

1) **Dimensionality condition (James–Stein):** Stein-type shrinkage yields uniformly lower quadratic risk for $p \geq 3$. Convolutional layers inhabit high-dimensional gradient spaces (hundreds to thousands of parameters), whereas the final classifier layer operates in a much lower-dimensional regime (e.g., $p = 10$), where shrinkage guarantees weaken.
2) **Noisier gradients in feature extractors:** Convolutional gradients tend to be noisier due to batch-dependent feature maps and large receptive fields. Shrinking towards the momentum target stabilizes these estimates. Fully-connected layers, with fewer parameters and more direct supervision signal, benefit from unmodified adaptive moments.
3) **Empirical behavior in CIFAR experiments:** Restricting Stein-rule to Conv2d consistently matches or improves upon SGD, Momentum, and Adam, with the largest gains under label noise (0.05–0.1). Extending shrinkage to fully-connected layers did not provide additional benefit in our setting.
4) **Scope and efficiency:** Acting on only 3.4% of parameters concentrates computation where it matters most while preserving the direct adaptivity of low-dimensional projection layers.

In summary, SR-Adam applies Stein-rule where theory and data suggest the greatest leverage and leaves the projection layers to standard adaptive updates. Learning or selecting optimal parameter groups more generally (e.g., data-dependent or model-adaptive groupings) is a promising direction for future work.

B. Experimental Setup and Results

We evaluate SR-Adam against SGD, Momentum, and Adam on CIFAR10 and CIFAR100 using the SimpleCNN backbone. To probe robustness to label noise, we add corruptions at three levels: 0.0, 0.05, and 0.1. For each dataset/noise combination,

TABLE II: SR-Adam parameter grouping in SimpleCNN: Stein-rule applied only to convolutional layer weights (3.4% of total parameters).

Layer Group	Parameters	Stein-Rule
Conv2d layers (weights + bias)	18,528	**Yes**
Conv2d-1: 3×32×3×3 + 32	896	Yes
Conv2d-2: 32×64×3×3 + 64	18,496	Yes
Fully-connected layers (Linear-1, Linear-2)	525,706	No
Linear-1: 4096×128 + 128	524,416	No
Linear-2: 128×10 + 10	1,290	No
Total	**545,098**	—
Stein-rule coverage	**3.4%**	—

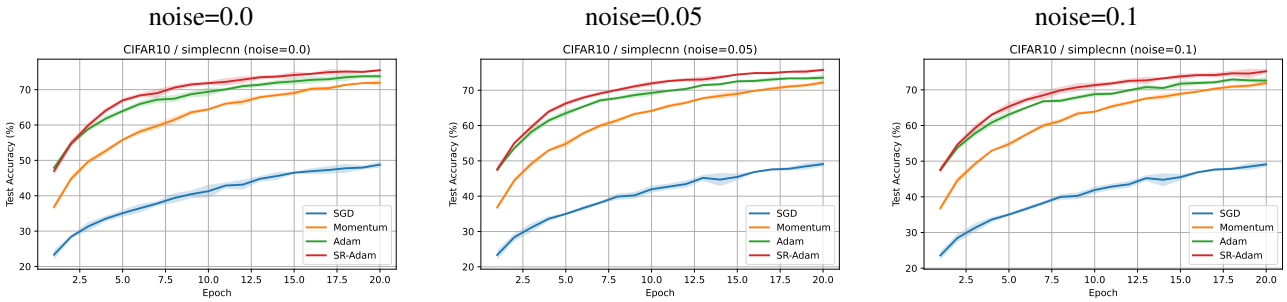

Fig. 1: SimpleCNN on CIFAR10: test accuracy vs. epoch across noise levels. Mean ± std over runs.

we conduct 5 independent runs with different random seeds and report mean ± std across runs. Accuracy is reported as a percentage (higher is better); loss is minimized (lower is better). In method comparison tables (Tables III–VI), we bold the best entry per dataset/noise column according to the respective metric direction.

a) Fairness and Reproducibility: All comparisons use an identical training protocol and data processing across methods; only the optimizer changes. We hold constant the backbone, number of epochs, batch size, label-noise injection, evaluation procedure, and seed schedule (five seeds per configuration), while using standard, fixed hyperparameters per optimizer throughout. To avoid discrepancies due to third-party implementations, all optimizers are implemented within a single, consistent codebase with matching interfaces, and the full executable code is publicly available from the paper repository [1].

1) Per-Epoch Behavior: Figures 1–4 display test accuracy and loss across epochs, stratified by noise level. Each figure shows three panels corresponding to noise levels 0.0, 0.05, and 0.1. All four methods (SGD, Momentum, Adam, SR-Adam) are plotted with mean ± std bands.

2) Aggregated Metrics: Tables III and V report the best accuracy and loss achieved over all epochs; Tables IV and VI report final epoch values. Each table rows are methods and columns are dataset×noise combinations. The mean ± std are computed across the 5 runs; bolded entries highlight the best-performing method per column. SR-Adam consistently achieves competitive or superior performance, particularly in high-noise regimes (0.05, 0.1), demonstrating the benefit of dynamic shrinkage on noisy gradients.

VII. Discussion and Conclusion

The SR-Adam algorithm introduces a rigorous statistical "gate" to the optimization process. In regimes of high uncertainty (early training or noisy batches), the term $\frac{(p-2)\sigma^2}{D_n}$ dominates, and $c_t \to 0$. The optimizer effectively rejects the noisy observation $\mathbf{g}_t$ and defaults to the stable momentum $\mathbf{m}_{t-1}$. As training stabilizes and variance decreases, $c_t \to 1$, recovering the standard Adam behavior. This methodology demonstrates that the inadmissibility of high-dimensional estimators is not merely a theoretical curiosity but a practical lever for improving deep learning optimization. Future work will extend this to Preliminary Test (PT) strategies for dynamic network pruning.

References

[1] C. Stein, "Inadmissibility of the usual estimator for the mean of a multivariate normal distribution," *Proc. Third Berkeley Symp. Math. Statist. Prob.*, vol. 1, pp. 197-206, 1956.
[2] W. James and C. Stein, "Estimation with quadratic loss," *Proc. Fourth Berkeley Symp. Math. Statist. Prob.*, vol. 1, pp. 361-379, 1961.
[3] A.K. Md. E. Saleh, *Theory of Preliminary Test and Stein-Type Estimation with Applications*, Wiley, 2006.

[1] https://github.com/mamintoosi-papers-codes/SR-Adam

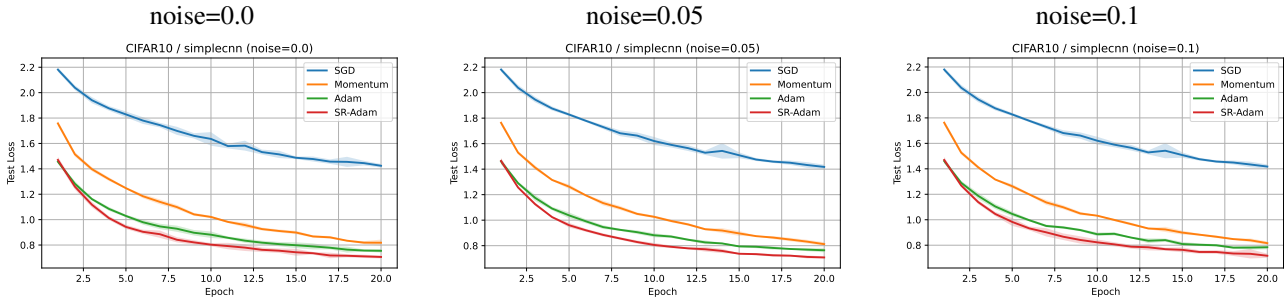

Fig. 2: SimpleCNN on CIFAR10: test loss vs. epoch across noise levels. Mean ± std over runs.

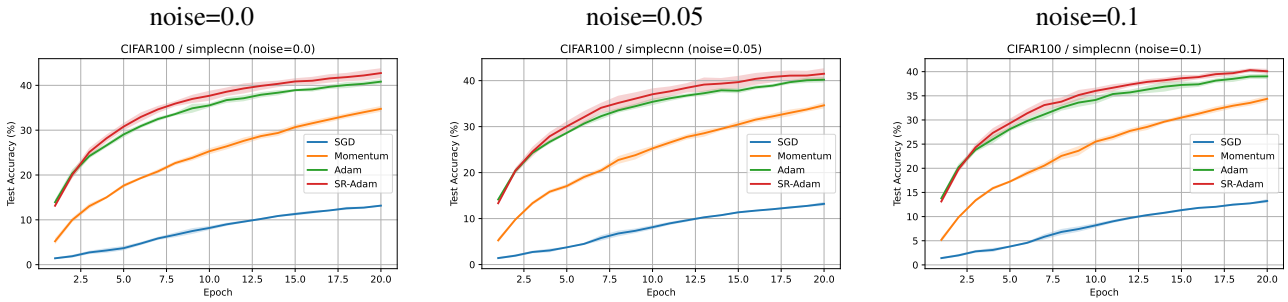

Fig. 3: SimpleCNN on CIFAR100: test accuracy vs. epoch across noise levels. Mean ± std over runs.

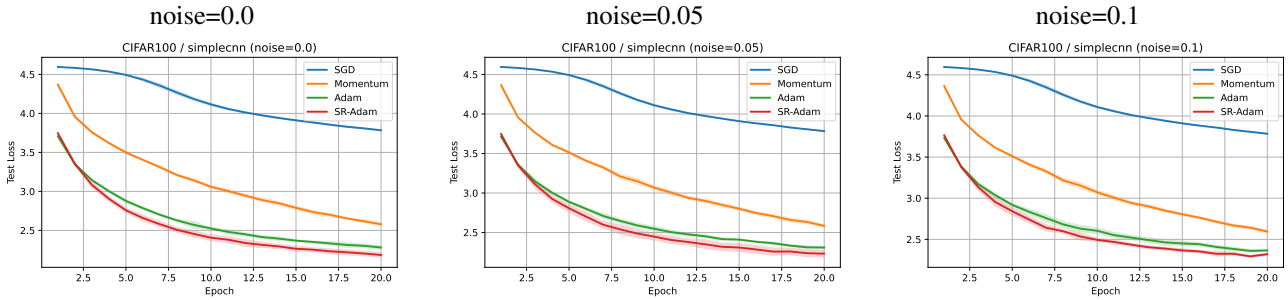

Fig. 4: SimpleCNN on CIFAR100: test loss vs. epoch across noise levels. Mean ± std over runs.

	CIFAR10			CIFAR100		
Method	0.0	0.05	0.1	0.0	0.05	0.1
SGD	48.95 ± 0.74	49.25 ± 0.65	49.24 ± 0.68	13.17 ± 0.22	13.20 ± 0.44	13.23 ± 0.40
Momentum	72.31 ± 0.47	72.22 ± 0.65	71.89 ± 0.61	34.77 ± 0.65	34.63 ± 0.66	34.38 ± 0.63
Adam	74.12 ± 0.60	73.95 ± 0.39	73.20 ± 0.50	40.85 ± 0.55	40.25 ± 0.60	39.14 ± 0.55
SR-Adam	**75.59 ± 0.50**	**75.84 ± 0.28**	**75.37 ± 0.62**	**42.74 ± 1.09**	**41.50 ± 1.20**	**40.43 ± 0.30**

TABLE III: Best test accuracy (mean ± std) over epochs; higher is better.

	CIFAR10			CIFAR100		
Method	0.0	0.05	0.1	0.0	0.05	0.1
SGD	48.79 ± 0.84	49.09 ± 0.74	49.11 ± 0.69	13.17 ± 0.22	13.20 ± 0.44	13.23 ± 0.40
Momentum	71.92 ± 0.91	72.19 ± 0.71	71.89 ± 0.61	34.76 ± 0.64	34.63 ± 0.66	34.38 ± 0.63
Adam	73.77 ± 0.69	73.54 ± 0.78	72.59 ± 0.92	40.82 ± 0.56	40.19 ± 0.60	39.02 ± 0.54
SR-Adam	**75.48 ± 0.43**	**75.75 ± 0.33**	**75.21 ± 0.63**	**42.74 ± 1.09**	**41.50 ± 1.20**	**40.08 ± 0.50**

TABLE IV: Final test accuracy (mean ± std) at last epoch; higher is better.

	CIFAR10			CIFAR100		
Method	0.0	0.05	0.1	0.0	0.05	0.1
SGD	1.42 ± 0.01	1.41 ± 0.01	1.41 ± 0.01	3.78 ± 0.01	3.78 ± 0.01	3.78 ± 0.01
Momentum	0.81 ± 0.02	0.81 ± 0.01	0.82 ± 0.01	2.58 ± 0.03	2.58 ± 0.02	2.59 ± 0.03
Adam	0.75 ± 0.01	0.76 ± 0.01	0.77 ± 0.01	2.28 ± 0.03	2.30 ± 0.02	2.36 ± 0.02
SR-Adam	**0.70 ± 0.01**	**0.70 ± 0.01**	**0.71 ± 0.01**	**2.18 ± 0.04**	**2.23 ± 0.05**	**2.29 ± 0.02**

TABLE V: Best test loss (mean ± std) over epochs; lower is better.

	CIFAR10			CIFAR100		
Method	0.0	0.05	0.1	0.0	0.05	0.1
SGD	1.42 ± 0.01	1.42 ± 0.02	1.42 ± 0.02	3.78 ± 0.01	3.78 ± 0.01	3.78 ± 0.01
Momentum	0.82 ± 0.02	0.81 ± 0.02	0.82 ± 0.01	2.58 ± 0.03	2.58 ± 0.02	2.59 ± 0.03
Adam	0.75 ± 0.01	0.76 ± 0.02	0.78 ± 0.02	2.28 ± 0.03	2.31 ± 0.02	2.37 ± 0.01
SR-Adam	**0.71 ± 0.01**	**0.71 ± 0.01**	**0.72 ± 0.02**	**2.18 ± 0.04**	**2.23 ± 0.05**	**2.32 ± 0.02**

TABLE VI: Final test loss (mean ± std) at last epoch; lower is better.