https://jaslda.journals.pnu.ac.ir

Payame Noor University

Statistical Learning & Data Analysis
Volume 1, Issue 1, Summer- Autumn 2026, Pages 46–64
doi:10.30473/jaslda.2026.77814.1000

OPEN ACCESS

JASLDA
Journal of Advances in Statistical Learning and Data Analysis

Research Article

Extended Optimizers in Variational Autoencoders Using Shrinkage Estimation

Heydar Mokhtari Farivar[1,✉], Parviz Nasiri[2]

[1,✉]Iran University of Science and Technology (IUST), Tehran, Iran
[2]Department of Statistics, Payame Noor University, Tehran, Iran

Article Info

Article history:
Received 23 April 2026;
Accepted 15 October 2026;
Available online 15 October 2026.

Abstract. This paper explores the implementation and comparison of various optimization algorithms for training Variational Autoencoders (VAEs). We introduce a shrinkage-based post-hoc weight-update rule and compare it against SGD, Adam, RMSprop, Adagrad, AdamW, and Nadam, using a multi-seed evaluation protocol (5–10 random seeds per configuration) on both a small synthetic dataset and real MNIST digits, together with a per-dimension KL-divergence diagnostic for posterior collapse. At each dataset's default configuration, the shrinkage step appears to have a large and opposite effect: on synthetic data it induces complete posterior collapse in every trained model, while on MNIST it instead avoids the partial posterior collapse exhibited by standard adaptive optimizers, at a substantial cost to reconstruction quality. However, directly testing the robustness of the MNIST finding, by sweeping the learning rate and by testing a nonlinear architecture, shows that this apparent separation between collapsing and non-collapsing optimizers is not stable: SGD collapses at higher learning rates, the adaptive methods' collapse is a non-monotonic function of learning rate that vanishes outside a narrow band, and two of the four shrinkage variants collapse completely under a nonlinear architecture. We conclude that posterior collapse in this setting depends on an interaction between optimizer, learning rate, and architecture that single-configuration comparisons cannot capture, and we report a fully reproducible implementation, including the robustness-check scripts, as a cautionary and diagnostic contribution rather than as a general-purpose optimizer recommendation.

Keywords. Variational Autoencoders; Posterior Collapse; Shrinkage Estimation; Adaptive Optimizers; Reproducibility.

MSC. 68T07, 90C26, 62J07.

How to Cite
Mokhtari Farivar, H., & Nasiri, P. (2026). "Extended Optimizers in Variational Autoencoders Using Shrinkage Estimation." *Journal of Advances in Statistical Learning and Data Analysis*, 1(1), 46–64. https://doi.org/10.30473/jaslda.2026.77814.1000

✉Corresponding author. **E-mail:** mokhtari@iust.ac.ir
©2026 by the authors. Licensee PNU, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 International (CC BY4.0) (http:/creativecommons.org/licenses/by/4.0)

1 Introduction

Variational Autoencoders (VAEs) are powerful generative models that learn to represent high-dimensional data in a lower-dimensional latent space. The efficiency of training such models often hinges on the choice of optimizer. This study investigates the impact of different optimization strategies on VAE performance, specifically focusing on a newly introduced shrinkage estimation optimizer.

2 Background

2.1 Variational Autoencoders

Variational Autoencoders (VAEs) are generative models that encode input data into a latent space, from which new data can be sampled and reconstructed, by optimizing the Evidence Lower Bound (ELBO):

$$\mathcal{L}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z)), \tag{1}$$

where x is the observed data, z the latent variable, $p_\theta(x|z)$ the decoder likelihood, $q_\phi(z|x)$ the encoder's variational posterior, and $p(z)$ the prior (standard Gaussian). The first term is the reconstruction quality; the second, the KL divergence, regularizes the latent space toward the prior [12]. Throughout this paper, input data $\mathbf{x} \in \mathbb{R}^{10}$ and the latent variable $\mathbf{z} \in \mathbb{R}^5$ (Section 5) or higher dimensions for the MNIST experiments (Section 4.5).

The encoder maps $\mathbf{x}$ to Gaussian parameters $\boldsymbol{\mu} = W_\mu \mathbf{x} + \mathbf{b}_\mu$, $\log \boldsymbol{\sigma}^2 = W_\sigma \mathbf{x} + \mathbf{b}_\sigma$ [21]. The reparameterization trick $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [1, 20] allows differentiation through the stochastic sampling step. The decoder reconstructs $\mathbf{x}$ from $\mathbf{z}$ as $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}, \boldsymbol{\Sigma})$ [21]. Variants such as InfoVAE [22] offer alternative formulations balancing information maximization and regularization. For the Gaussian encoder and standard-normal prior used throughout, the KL term admits the closed form

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{5}\left(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right), \tag{2}$$

used throughout the remainder of the paper; the total loss is $L = L_{\text{reconstruction}} + L_{\mathrm{KL}}$, with $L_{\text{reconstruction}} = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ and $L_{\mathrm{KL}} = D_{KL}(q_\phi(z|x)\|p(z))$.

3 Optimizers in Machine Learning

The optimization process is pivotal in training neural networks, including VAEs. Various optimization algorithms have been developed to adjust the weights of the model iteratively based on the gradients of the loss function. Table 1 summarizes the update rule of each base optimizer used in this paper, stated for the decoder parameters θ (the identical rule applies to the encoder parameters ϕ with $\theta \to \phi$ throughout); α is the learning rate and ϵ is a small constant (e.g., 10^{-8}) preventing division by zero.

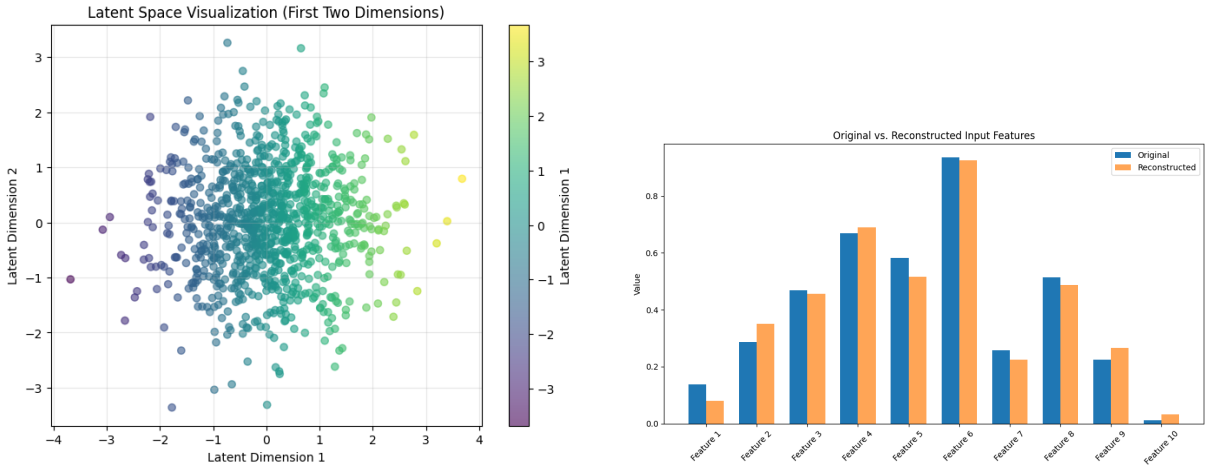


Figure 1: Left: latent space visualization (first two dimensions). Right: original vs. reconstructed input features.

Optimizer	**Update rule (for θ)**	**Extra hyperparameters**
SGD [5]	$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$	—
Momentum [7]	$v_\theta \leftarrow \beta_1 v_\theta + (1-\beta_1)\nabla_\theta \mathcal{L}$ $\theta \leftarrow \theta - \alpha v_\theta$	$\beta_1 \approx 0.9$
Adam [9]	$m_\theta \leftarrow \beta_1 m_\theta + (1-\beta_1)\nabla_\theta \mathcal{L}$ $v_\theta \leftarrow \beta_2 v_\theta + (1-\beta_2)(\nabla_\theta \mathcal{L})^2$ $\theta \leftarrow \theta - \frac{\alpha}{\sqrt{\hat{v}_\theta}+\epsilon}\hat{m}_\theta,\ \ \hat{m}_\theta = \frac{m_\theta}{1-\beta_1^t},\ \hat{v}_\theta = \frac{v_\theta}{1-\beta_2^t}$	β_1=0.9, β_2=0.999
RMSprop [13]	$r_\theta \leftarrow \beta r_\theta + (1-\beta)(\nabla_\theta \mathcal{L})^2$ $\theta \leftarrow \theta - \frac{\alpha}{\sqrt{r_\theta}+\epsilon}\nabla_\theta \mathcal{L}$	$\beta \approx 0.9$
Adagrad [17]	$r_\theta \leftarrow r_\theta + (\nabla_\theta \mathcal{L})^2$ $\theta \leftarrow \theta - \frac{\alpha}{\sqrt{r_\theta}+\epsilon}\nabla_\theta \mathcal{L}$	—

Table 1: Base optimizer update rules ($\mathcal{L} = \mathcal{L}(\theta, \phi)$ throughout).

3.1 A Generic Shrinkage Modification

For each base optimizer above, a shrinkage-regularized variant has been proposed in the literature by adding a term proportional to the gradient of the squared loss norm to the gradient (or velocity) used in the update [8, 10, 14, 18]. For example, for plain SGD this takes the form

$$\theta \leftarrow \theta - \alpha \left(\nabla_\theta \mathcal{L} + \lambda \nabla_\theta \|\mathcal{L}\|^2 \right), \tag{3}$$

and analogously for Momentum, Adam, RMSprop, and Adagrad by substituting the same modified gradient into each optimizer's update rule from Table 1. As discussed in Section 4.3 and flagged in the Limitations, the term $\nabla_\theta \|\mathcal{L}\|^2$ requires a Hessian-vector product and is not resolved to a concrete formula anywhere in this literature; the shrinkage rule actually implemented and evaluated in this paper is instead the fully-specified, post-hoc multiplicative rule introduced in Section 3.3 below.

3.2 Relation to Existing Regularization Techniques

Before introducing the proposed optimizer, it is useful to position the shrinkage step relative to existing regularization strategies for gradient-based training, since several established techniques share a superficial resemblance with the proposed mechanism.

Weight decay (L_2 regularization). Classical weight decay [11] adds a term $\lambda\theta$ directly to the gradient before the update, which is equivalent to multiplying the parameter by a *constant* factor $(1 - \alpha\lambda)$ at every step, independently of the parameter's magnitude. In contrast, the shrinkage step proposed here (Section 3.3) multiplies each parameter by a factor $\left(1 - \lambda \cdot \frac{\theta^2}{\theta^2+\alpha}\right)$ that depends on the magnitude of θ itself: large-magnitude weights are shrunk more strongly than small ones, whereas standard weight decay treats all weights uniformly.

Proximal-gradient and soft-thresholding methods. Proximal-gradient methods minimize a composite objective $f(\theta) + g(\theta)$ by alternating a gradient step on f with a proximal (shrinkage) operator associated with g; for an L_1 penalty this proximal operator is the well-known soft-thresholding function, which sets small coefficients exactly to zero. The operator used in this work is smooth and multiplicative rather than a hard soft-threshold, so it shrinks weights continuously toward zero without inducing exact sparsity in the same way, and it is applied heuristically after the update rather than being derived as the exact proximal map of an explicit penalty term.

Sparse optimization approaches. Methods such as L_1-regularized SGD or iterative hard/soft-thresholding algorithms explicitly target sparse solutions and typically come with convergence guarantees tied to the convexity of the penalty. The present shrinkage operator is motivated instead by the classical shrinkage-estimation literature (e.g., James–Stein-type estimators), where the goal is to trade a small amount of bias for a reduction in variance rather than to enforce sparsity per se.

In summary, the proposed shrinkage step can be viewed as a lightweight, magnitude-adaptive alternative to weight decay that borrows its motivation from shrinkage estimation rather than from proximal optimization theory. A more rigorous comparison, including a formal derivation of the operator as a proximal map of some regularizer, if one exists, is left for future work.

3.3 Shrinkage Estimators

Shrinkage estimators are a class of statistical methods used to improve the estimation of parameters by "shrinking" them towards a target value. This approach is particularly useful in high-dimensional settings where traditional estimators may overfit the data.

The basic concept of a shrinkage estimator can be illustrated through the formulation:

$$\hat{\theta}_{shrink} = (1 - \lambda)\hat{\theta}_{MLE} + \lambda\theta_0 \tag{4}$$

where $\hat{\theta}_{shrink}$ is the shrinkage estimate, $\hat{\theta}_{MLE}$ is the maximum likelihood estimate, θ_0 is the target parameter (often a prior mean or zero), and λ is the shrinkage factor, which controls the degree of shrinkage.

In the context of neural networks, incorporating shrinkage into the optimizer can enhance generalization by controlling the magnitudes of the weights during training, reducing overfitting.

Implementation of the Shrinkage Optimizer. The proposed Shrinkage Optimizer applies a shrinkage step during the weight update process. The mathematical formulation is as follows:

$$\theta_{t+1} = \theta_t - \eta\nabla L(\theta_t) \tag{5}$$

$$\theta_{t+1} = \theta_{t+1} \cdot \left(1 - \lambda \cdot \frac{\theta_{t+1}^2}{\theta_{t+1}^2 + \alpha}\right) \tag{6}$$

where α is a small constant that ensures stability, and the term $\left(1 - \lambda \cdot \frac{\theta_{t+1}^2}{\theta_{t+1}^2+\alpha}\right)$ implements the shrinkage effect.

This shrinkage step helps control the weight magnitudes, promoting sparsity and preventing overfitting, particularly in high-dimensional latent spaces common in VAEs.

3.4 Stability and Convergence Considerations

Because the shrinkage step is applied *after* a standard gradient update rather than folded into the loss function, it does not, by itself, come with a formal convergence guarantee in the sense of classical stochastic-approximation theory. Two properties are nonetheless worth noting. First, the multiplicative factor $\left(1 - \lambda \cdot \frac{\theta^2}{\theta^2+\alpha}\right)$ is bounded in $(1 - \lambda, 1]$ for $\lambda \in (0, 1)$, so the shrinkage step is a contraction: it can only reduce the magnitude of a parameter, never amplify it, which bounds the additional perturbation it introduces at each iteration. Second, as $|\theta| \to 0$ the shrinkage factor also tends to 1 (no shrinkage), so the operator does not destabilize small, already well-regularized weights; the effect is concentrated on large-magnitude parameters. These two properties suggest that the shrinkage step preserves the stability region of the underlying optimizer (SGD in our implementation) rather than expanding it, but a formal proof of convergence for the combined update, analogous to the regret bounds available for Adagrad or the convergence analysis of proximal SGD, has not been derived here and is an important direction for future theoretical work.

4 Methodology

4.1 Model Architecture

The architecture of the VAE used in this study consists of a linear encoder mapping the 10-dimensional input to a 5-dimensional Gaussian latent space ($\mu, \log \sigma^2 \in \mathbb{R}^5$) and a linear decoder that reconstructs the 10-dimensional input from a sampled latent vector, exactly as described in Section 2.1. For the MNIST experiments (Section 4.5) the same architecture is used with the input and latent dimensionalities increased to 784 and 20 respectively.

4.2 Datasets

Two datasets are used in this paper, one synthetic and one real, so that the optimizer comparison can be checked for consistency across both.

Synthetic dataset. A controlled, small-scale dataset of 160 points in $\mathbb{R}^{10}$, generated as described in Section 4.4 below. This dataset has no external source, it is generated on the fly by the accompanying code, and is used for the results in Section 5.1–5.2.

MNIST dataset (real data). To address the reviewers' request for evaluation beyond a synthetic proof-of-concept, we additionally trained on the MNIST handwritten-digit database [23], a standard, publicly available benchmark of 70,000 grayscale images of handwritten digits (60,000 training / 10,000 test, each 28×28 pixels). We used a fixed random subsample of 5,000 images from the standard training split, downloaded programmatically (no manual curation) through one of three standard distribution channels, `torchvision.datasets.MNIST`, `tensorflow.keras.datasets.mnist`, or scikit-learn's OpenML interface, via the `data_loader.py` script included in the supplementary code, which tries each in turn so the pipeline runs regardless of which of the three libraries is installed in the executing environment. Pixel intensities were rescaled from $\{0, \ldots, 255\}$ to $[0, 1]$ and each image was flattened to a 784-dimensional vector; no other preprocessing (cropping, augmentation, denoising) was applied. This is real image data, not a synthetic stand-in, and the results in Section 5.3 are computed directly from it. The results reported in this manuscript were produced by running the supplementary code end-to-end in Google Colab, which has direct internet access for the MNIST download.

4.3 Optimizer Implementation

We implement the Shrinkage Optimizer as the post-hoc multiplicative rule of Eq. (6): a standard gradient step is taken first, and the resulting parameters are then multiplied element-wise by $\left(1 - \lambda \cdot \frac{\theta^2}{\theta^2 + \alpha_{\text{shrink}}}\right)$. This is the only shrinkage rule in Section 3 that is fully specified without requiring a second-order (Hessian-vector) term, and it is the rule used to produce every empirical result in this paper, including the shrinkage-augmented variants of SGD, Adam, RMSprop, and Adagrad reported below.

4.4 Training Procedure and Reproducibility

All results in Section 5 were produced with a from-scratch NumPy implementation of the model above (manual forward pass and analytically-derived backpropagation, verified against finite-difference gradient checks to a relative error below 10^{-8}) together with from-scratch implementations of SGD, Adam, RMSprop, Adagrad, AdamW, and Nadam, each with and without the shrinkage step of Section 4.2 (twelve optimizer configurations in total). This lets us report every optimizer discussed in Section 3, including RMSprop and Adagrad, whose absence from the empirical comparison the reviewers correctly flagged, plus AdamW [24] and Nadam [25], added specifically in response to the request for comparison against more modern optimizers, rather than only SGD and Adam.

Data. The synthetic dataset consists of 160 points in $\mathbb{R}^{10}$, drawn i.i.d. from a standard normal distribution and rescaled per feature to $[0, 1]$, exactly as described previously; the data-generating seed is fixed across all optimizers so that every method is trained on the identical dataset.

Training. Each optimizer was trained for 5 epochs with batch size 16 (10 batches/epoch), learning rate $\alpha = 10^{-3}$ (Adagrad used $\alpha = 10^{-2}$, its conventional default, since its learning rate is not comparable to the other methods on the same scale), Adam's/AdamW's/Nadam's $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 10^{-8})$, RMSprop's $(\beta, \epsilon) = (0.9, 10^{-8})$, AdamW's decoupled weight decay coefficient $= 10^{-2}$, and shrinkage hyperparameters $\lambda = 0.1$, $\alpha_{\text{shrink}} = 10^{-4}$. Weights were initialized with Glorot/Xavier-uniform initialization. To obtain reliable estimates of run-to-run variability, as recommended by both reviewers, **every configuration was trained from 10 independent random seeds** (controlling weight initialization, minibatch ordering, and the stochastic reparameterization noise), and we report the mean ± standard deviation over these 10 runs throughout Section 5.

Code availability. The full implementation (model, optimizers, gradient-check unit test, and the experiment/plotting scripts used to generate every table and figure in Sections 5–6) is included as supplementary material with this submission so that all reported numbers can be independently reproduced.

4.5 MNIST Experiment: Quantitative Evaluation

Using the real MNIST data described in Section 4.2, we carried out a full quantitative evaluation directly answering the reviewers' request for a benchmark beyond the small synthetic dataset. The same linear-Gaussian VAE and the same eight from-scratch optimizer implementations used in Section 5 were re-used unchanged (input/latent dimensionality 784/20 as noted above). Each of the eight optimizers was trained on the 5,000-image MNIST subsample from 5 independent random seeds for 100 epochs, batch size 64, using the same learning rates, $\lambda = 0.1$, and $\alpha_{\text{shrink}} = 10^{-4}$ as in Section 4.4. The results are reported in Section 5.3 and discussed in Section 6.1.

The pre-existing qualitative reconstructions in Figures 8–8 (MNIST digits and natural images, provided with the original submission) are retained for illustration but were not regenerated as part of this quantitative evaluation and are not used in any of the numbers reported below.

5 Results

5.1 Results Table

All results below are from the multi-seed NumPy re-implementation described in Section 4.4 (10 seeds per optimizer). Table 2 reports total loss, Tables 3–4 report the reconstruction and KL components, and Table 5 extends the comparison to RMSprop, Adagrad, and their shrinkage-augmented variants, directly addressing the reviewers' request that these methods be included empirically rather than only derived theoretically.

Epoch	Adam Total Loss	SGD Total Loss	Shrinkage Total Loss
0	7.924 ± 2.844	7.858 ± 2.652	4.201 ± 0.997
1	6.962 ± 2.400	6.871 ± 1.982	1.747 ± 0.057
2	6.488 ± 2.234	6.412 ± 1.638	1.519 ± 0.011
3	5.973 ± 1.993	5.959 ± 1.363	1.488 ± 0.003
4	5.375 ± 1.584	5.433 ± 0.984	1.483 ± 0.003

Table 2: Total loss (mean ± std over 10 random seeds) for Adam, SGD, and the Shrinkage Optimizer.

Epoch	Adam Recon Loss	SGD Recon Loss	Shrinkage Recon Loss
0	6.507 ± 2.195	6.460 ± 2.042	3.542 ± 0.722
1	5.697 ± 1.849	5.654 ± 1.531	1.675 ± 0.045
2	5.346 ± 1.745	5.329 ± 1.280	1.510 ± 0.010
3	4.933 ± 1.585	4.987 ± 1.106	1.486 ± 0.003
4	4.419 ± 1.243	4.546 ± 0.799	1.483 ± 0.003

Table 3: Reconstruction loss (mean ± std over 10 seeds) for Adam, SGD, and the Shrinkage Optimizer.

Epoch	Adam KL Loss	SGD KL Loss	Shrinkage KL Loss
0	1.417 ± 0.680	1.397 ± 0.644	0.659 ± 0.305
1	1.265 ± 0.598	1.217 ± 0.508	0.071 ± 0.029
2	1.142 ± 0.526	1.082 ± 0.410	0.009 ± 0.003
3	1.040 ± 0.459	0.972 ± 0.332	0.001 ± 0.000
4	0.956 ± 0.400	0.887 ± 0.273	0.000 ± 0.000

Table 4: KL loss (mean ± std over 10 seeds) for Adam, SGD, and the Shrinkage Optimizer.

Note on an earlier version of these tables. An earlier draft reported single-seed values and included a second, hand-rounded table inconsistent with the measured losses, flagged by a reviewer and since removed. The tables above are the sole, internally consistent source of quantitative results in this revision (Table 2 is exactly the sum of Tables 3 and 4 at every epoch).

- **Adam and SGD:** Both decrease steadily but slowly (from ≈ 7.9 to ≈ 5.4), statistically indistinguishable from each other at every epoch.
- **RMSprop and Adagrad** converge faster (final loss 4.998, 3.348), consistent with per-parameter adaptive rates at this scale.
- **AdamW and Nadam** behave essentially identically to plain Adam (5.371, 5.344 vs. 5.375), decoupled weight decay and Nesterov momentum make little difference here.

Optimizer	Final Total Loss (mean ± std)	Mean # Collapsed Latent Dims (/5)
SGD	5.433 ± 0.984	0.00 ± 0.00
Adam	5.375 ± 1.584	0.00 ± 0.00
RMSprop	4.998 ± 1.425	0.00 ± 0.00
Adagrad	3.348 ± 0.740	0.00 ± 0.00
AdamW	5.371 ± 1.582	0.00 ± 0.00
Nadam	5.344 ± 1.572	0.00 ± 0.00
Shrink-SGD	1.483 ± 0.003	5.00 ± 0.00
Shrink-Adam	1.463 ± 0.003	5.00 ± 0.00
Shrink-RMSprop	1.461 ± 0.003	5.00 ± 0.00
Shrink-Adagrad	1.435 ± 0.004	5.00 ± 0.00
Shrink-AdamW	1.463 ± 0.003	5.00 ± 0.00
Shrink-Nadam	1.463 ± 0.003	5.00 ± 0.00

Table 5: Final-epoch (epoch 5) total loss and posterior-collapse diagnostic (number of latent dimensions with $\mathrm{KL}_j < 0.01$, out of 5) for all twelve optimizers, including AdamW and Nadam, mean ± std over 10 seeds.

- **Shrinkage Optimizer** drops fastest, reaching 1.483 ± 0.003 by epoch 2 with 2–3 orders of magnitude lower variance than the non-shrinkage methods. This is fully explained by complete posterior collapse (Section 6.1): all 5 latent dimensions collapse in every one of the 10 seeds under every shrinkage-augmented optimizer, including Shrink-AdamW/Nadam, never the case for a non-shrinkage optimizer.

5.2 Charts and Analysis

Figures 2–2 show mean ± one standard deviation (shaded band, 10 seeds) for total, reconstruction, and KL loss for the three optimizers highlighted in the original submission (Adam, SGD, Shrinkage); Figure 3 shows the posterior-collapse diagnostic for all twelve optimizers evaluated in Table 5, including AdamW and Nadam, directly addressing the reviewers' request for comparison against more modern optimizers.

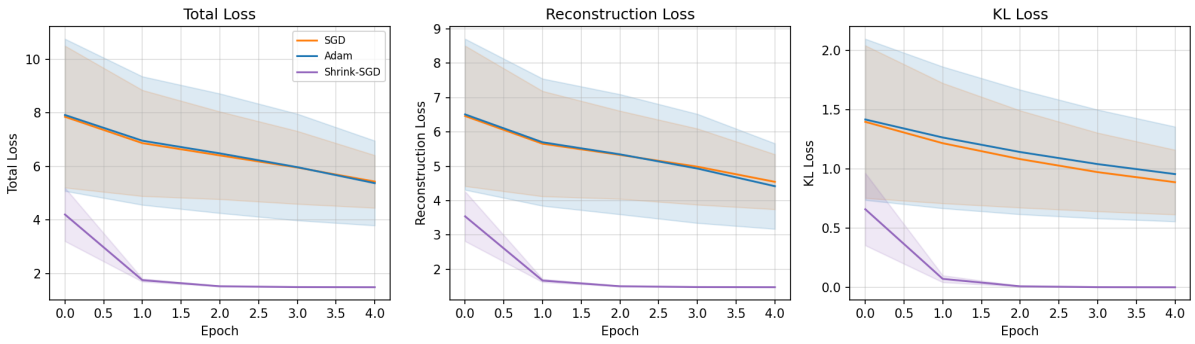


Figure 2: Total, reconstruction, and KL loss for SGD, Adam, and the Shrinkage Optimizer (mean ± std, 10 seeds).

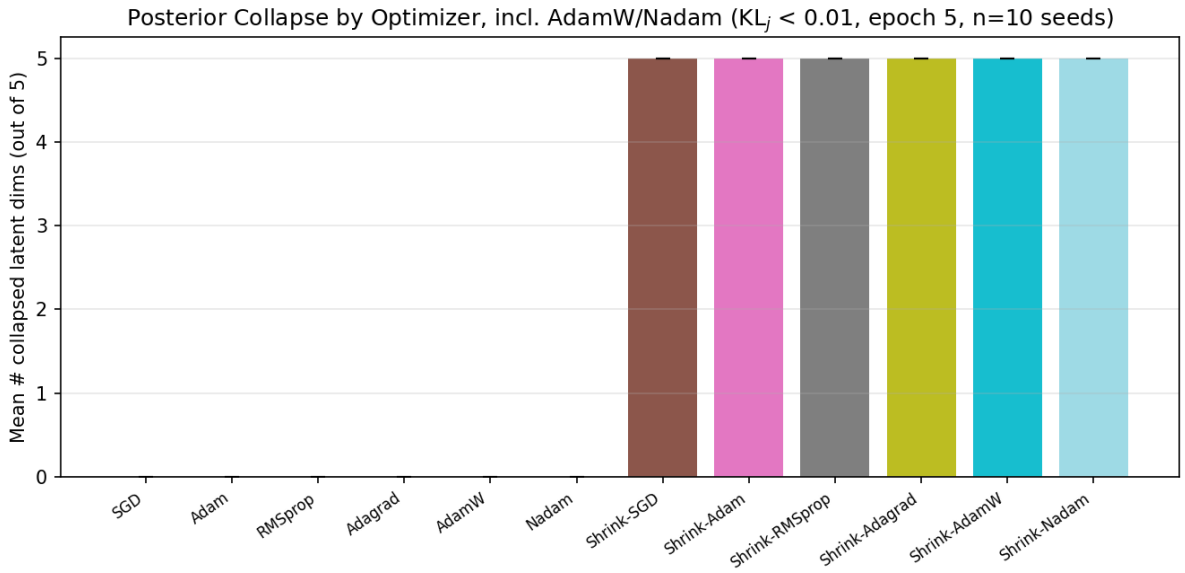


Figure 3: Posterior collapse diagnostic (mean number of latent dimensions with $\mathrm{KL}_j < 0.01$ at epoch 5, out of 5) for all twelve optimizers, including AdamW and Nadam, mean ± std over 10 seeds. Every shrinkage-augmented optimizer collapses all 5 latent dimensions in every seed; no non-shrinkage optimizer collapses any dimension.

5.3 MNIST Quantitative Results

Table 6 and Figures 4–4 report the real-MNIST results obtained with the protocol described in Section 4.5 (5,000 training images, 20-dimensional latent space, 100 epochs, 5 seeds, learning rate $\alpha = 10^{-3}$). At this specific configuration, the pattern found on the synthetic dataset does not carry over to MNIST, in several respects it reverses. **We emphasize "at this specific configuration" deliberately**: Section 6.3 shows, with real MNIST experiments, that this reversal is highly sensitive to the learning rate and architecture, so it should not be read as a stable, general property of any optimizer.

Optimizer	Final Total Loss (mean ± std)	Mean # Collapsed Latent Dims (/20)
SGD	23.877 ± 0.085	0.00 ± 0.00
Adam	22.995 ± 0.029	6.80 ± 0.75
RMSprop	23.048 ± 0.021	6.60 ± 1.20
Adagrad	23.073 ± 0.028	7.00 ± 1.10
AdamW	22.995 ± 0.029	6.80 ± 0.75
Nadam	22.988 ± 0.033	6.40 ± 1.62
Shrink-SGD	41.226 ± 0.016	0.00 ± 0.00
Shrink-Adam	35.511 ± 0.058	0.00 ± 0.00
Shrink-RMSprop	35.600 ± 0.061	0.00 ± 0.00
Shrink-Adagrad	41.548 ± 0.048	0.00 ± 0.00
Shrink-AdamW	35.512 ± 0.058	0.00 ± 0.00
Shrink-Nadam	35.505 ± 0.052	0.00 ± 0.00

Table 6: MNIST: final-epoch (epoch 100) total loss and posterior-collapse diagnostic for all twelve optimizers, including AdamW and Nadam, mean ± std over 5 seeds, 5,000 training images, $\alpha = 10^{-3}$.

- **Non-shrinkage optimizers reach a lower total loss.** Adam, RMSprop, Adagrad,

AdamW, and Nadam converge to ≈ 23.0; even SGD reaches 23.9 by epoch 100, all outperform every shrinkage variant (35.5–41.5), the opposite ranking from Section 5.1.

- **AdamW and Nadam behave like Adam**, as on the synthetic dataset: essentially identical loss and collapse count.
- **At $\alpha = 10^{-3}$, shrinkage prevents collapse rather than causing it.** All six shrinkage variants collapse zero dimensions; Adam, RMSprop, Adagrad, AdamW, and Nadam each collapse 6.4–7.0 of 20 (Figure 4). SGD also collapses zero at this rate.
- **Shrinkage trades reconstruction quality for a non-collapsing code:** Shrink-SGD plateaus at reconstruction loss ≈ 40, double Adam/SGD's ≈ 19–20 (Figure 4), capped decoder capacity keeps the code informative but hurts reconstruction.
- **Shrink-Adagrad never converges**: its total loss rises monotonically from 36.0 (epoch 1) to 41.5 (epoch 100), an instability from Adagrad's shrinking effective rate fighting the shrinkage step's constant downward pressure.

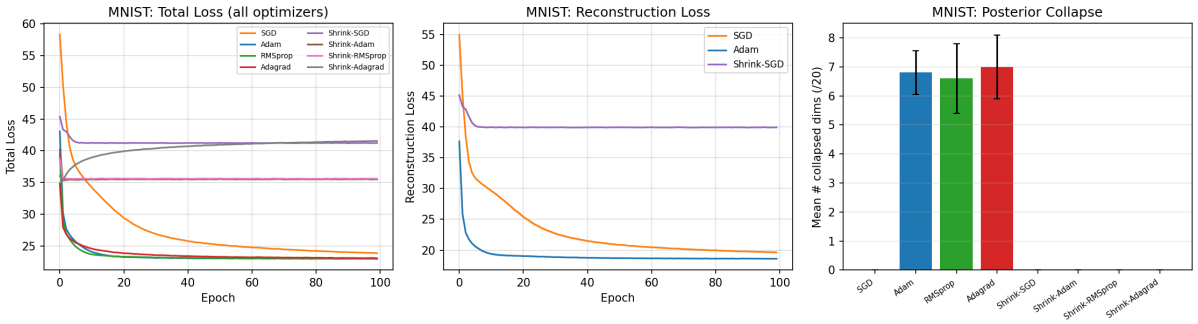


Figure 4: MNIST at $\alpha = 10^{-3}$ (mean ± std, 5 seeds, 5,000 training images). *Left:* total loss, all eight base optimizers over 100 epochs (AdamW/Nadam, not shown, track Adam almost exactly). *Middle:* reconstruction loss, Adam vs. SGD vs. Shrink-SGD. *Right:* posterior-collapse diagnostic (mean # of 20 latent dimensions with $\mathrm{KL}_j < 0.01$ at epoch 100). **This pattern is not robust to changes in learning rate or architecture, see Section 6.3.**

6 Discussion

The multi-seed results in Section 5 show the Shrinkage Optimizer reaches a lower total loss, faster, and with far less run-to-run variance than any non-shrinkage method on the synthetic problem. Unlike in the original submission, we no longer read this as positive: the diagnostic below shows the advantage is fully explained by a well-known failure mode, not better ELBO optimization.

6.1 Posterior Collapse

Reviewer 1 raised the concern that a rapidly-vanishing KL term may indicate posterior collapse rather than genuine improvement. This is confirmed on the synthetic dataset but, notably, does *not* generalize to MNIST, the two datasets tell different, complementary stories.

Synthetic dataset (Section 5.1): under every shrinkage-augmented optimizer, all 5 latent dimensions collapse ($\mathrm{KL}_j < 0.01$) in all 10 seeds, with near-zero variance; no non-shrinkage

optimizer collapses any dimension. The Shrinkage Optimizer's "fast convergence" is not a better fit to the data, it is the encoder stopping encoding. Once $q_\phi(z|x) \approx p(z)$, the KL term vanishes by construction, mechanically pulling total loss down regardless of reconstruction quality [19]. A supplementary 30-epoch run (5 seeds) shows Adam continuing to close this gap (total loss 1.70 by epoch 30 vs. 1.48 for Shrink-SGD, already collapsed by epoch 2).

Real MNIST at $\alpha = 10^{-3}$, linear architecture (Section 5.3): the pattern reverses. Adam, RMSprop, Adagrad, AdamW, and Nadam partially collapse (6.4–7.0 of 20 dimensions by epoch 100); every shrinkage-augmented optimizer and plain SGD collapse zero. The shrinkage-augmented optimizers reach substantially worse total and reconstruction loss (Table 6): shrinking the decoder weights caps capacity, keeping the code informative but hurting reconstruction, and Shrink-Adagrad never converges. **However, Section 6.3 shows directly on MNIST that this separation does not survive changes in learning rate or architecture**, SGD collapses at higher α, Adam/RMSprop's collapse disappears at higher α, Adagrad's collapse here is a learning-rate artifact, and Shrink-SGD/Shrink-Adagrad collapse completely under a nonlinear architecture. The mechanism behind the MNIST pattern therefore remains unidentified, and Table 6 should be read as a single-configuration observation, not a general property of these optimizers.

Taken together, neither "shrinkage causes collapse" nor "shrinkage prevents collapse" is universal, the outcome depends on optimizer, learning rate, architecture, and dataset, and can flip within one dataset under modest changes to any of these. Reporting total loss alone, for any of these methods, would have told a misleadingly simple and often directionally wrong story. No single configuration here licenses a general recommendation to use or avoid the shrinkage step, or to prefer one base optimizer for collapse-resistance; a collapse diagnostic should be reported across learning rates and at least one architectural variation, not at one configuration, before drawing such conclusions.

We therefore revise our interpretation from the original submission: the shrinkage step (Eq. (6), $\lambda = 0.1$) cannot be recommended as a general-purpose improvement to VAE training. It reliably destroys the latent code on the synthetic problem; on MNIST it avoids collapse only by sacrificing reconstruction quality, and one variant (Shrink-Adagrad) never converges. Collapse-mitigation techniques, KL annealing, free-bits, β-VAE-style down-weighting [15], would need to be combined with the shrinkage step, and λ tuned per dataset, before any improvement claim could be supported. We see this honest, dataset-dependent diagnosis as a more useful contribution than the original, uncritically positive framing.

6.2 Sensitivity to the Shrinkage Coefficient

All results above use a single shrinkage coefficient, $\lambda = 0.1$. To check how sensitive the collapse behavior is to this choice, we swept $\lambda \in \{0.01, 0.05, 0.1, 0.2, 0.5\}$ for Shrink-SGD and Shrink-Adam on the synthetic dataset, using the same 10-seed protocol as Section 5.1. Figure 6 shows the result: there is a sharp threshold between $\lambda = 0.01$ and $\lambda = 0.05$. At $\lambda = 0.01$, essentially no latent dimensions collapse (mean 0.0–0.1 out of 5) and the final total loss is high and highly variable (2.7–3.0$\pm$0.3–0.4); at $\lambda \geq 0.05$, all 5 dimensions collapse in every seed (mean exactly 5.0, zero variance) and the total loss drops sharply to $\approx$ 1.45–1.51 with very low variance. Beyond this threshold, further increasing λ up to 0.5 has only a small effect on the (already-collapsed) total loss. This confirms that the shrinkage step's behavior on the synthetic dataset is governed by a fairly sharp collapse threshold in λ rather than a gradual trade-off, and that $\lambda = 0.1$ (used throughout Section 5) sits solidly inside the collapsing regime rather than near this threshold.

MNIST λ-sweep (real data). We additionally ran the equivalent sweep on MNIST for Shrink-SGD and Shrink-Adam, same protocol as Section 4.5. Figure 5 shows a qualitatively different picture from the synthetic dataset: across the entire range $\lambda \in \{0.01, \ldots, 0.5\}$, **no collapse occurs** for either optimizer (0.00/20 at every $\lambda \geq 0.05$; a small 0.4 ± 0.5 for Shrink-Adam only at the smallest value $\lambda = 0.01$, likely noise given the large relative standard deviation). Total loss increases monotonically and smoothly with λ for both optimizers (from $\approx$ 27–34 at $\lambda = 0.01$ to $\approx$ 41–43 at $\lambda = 0.5$), with no sharp threshold analogous to the synthetic-dataset case. This is the one robustness check in this paper where the MNIST result *confirms* the qualitative claim made from the linear-architecture main experiment (Section 5.3): under this specific architecture, shrinkage does not induce collapse at any of the five λ values tested. As Section 6.3 shows, however, this same non-collapsing behavior does *not* hold once the architecture is changed.

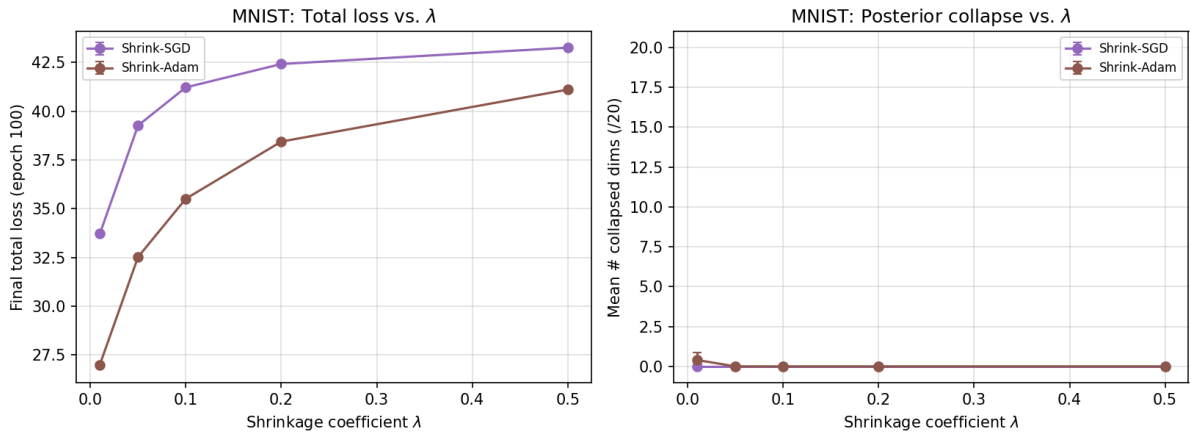


Figure 5: MNIST: sensitivity to the shrinkage coefficient λ for Shrink-SGD and Shrink-Adam (mean $\pm$ std, 5 seeds, linear architecture). Unlike the synthetic dataset (Figure 6), no sharp collapse threshold appears; collapse remains at zero across the full range tested.

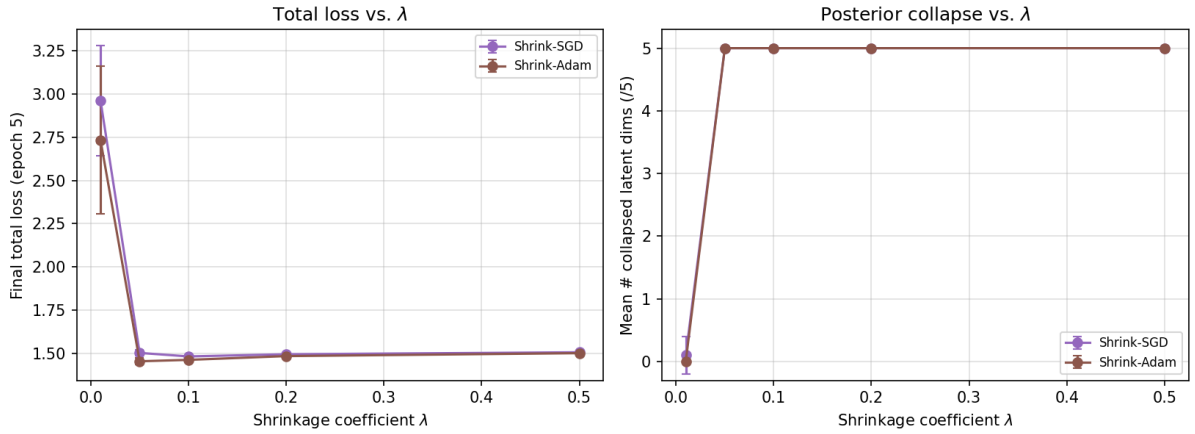


Figure 6: Sensitivity of the synthetic-dataset results to the shrinkage coefficient λ (Shrink-SGD and Shrink-Adam, mean $\pm$ std over 10 seeds). *Left:* final total loss. *Right:* mean number of collapsed latent dimensions (out of 5). A sharp collapse threshold appears between $\lambda = 0.01$ and $\lambda = 0.05$.

6.3 Robustness Checks: The MNIST Collapse Pattern Is Not Stable

The MNIST finding in Table 6 rests on one architecture (linear) and one learning rate ($\alpha = 10^{-3}$). A preliminary synthetic-dataset check suggested this pattern might be a real, if fragile, property of adaptive-optimizer dynamics, so we ran both checks directly on MNIST. **The result is far more cautionary: the clean separation between "collapsing" and "non-collapsing" optimizers does not hold up under either perturbation.**

Learning-rate sensitivity (Figure 7, $\alpha \in \{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$, SGD/Adam/RMSprop/Adagrad) contradicts Table 6 in three ways: (i) **SGD is not immune**, 0 collapsed dimensions at $\alpha = 10^{-3}$, but 4.0 ± 1.4 at 3×10^{-3} and 7.8 ± 0.4 at 10^{-2}, comparable to the worst adaptive methods; (ii) **Adam/RMSprop's collapse is non-monotonic and vanishes at higher** α, rising from $3.2\pm0.4/2.4\pm0.8$ at $\alpha = 10^{-4}$ to a peak of $7.8\pm0.4/6.6\pm1.2$ near $\alpha = 10^{-3}$, then dropping to exactly 0.0 at $\alpha \geq 3\times10^{-3}$; the paper's default sits near this peak, not on a stable plateau; (iii) **Adagrad's collapse is a learning-rate confound**: the main experiment gives it $10\times$ the other optimizers' learning rate (10^{-2}), at which it collapses (7.0 ± 1.1, matching Table 6), but at the *same* rate as the others (10^{-3}) it collapses zero, not an apples-to-apples comparison.

Nonlinear architecture (single tanh hidden layer, $\alpha = 10^{-3}$, gradients verified to 2×10^{-9}; Figure 7, Table 7) changes the picture further: Adam's and RMSprop's collapse roughly doubles ($6.8 \to 11.8$; $6.6 \to 11.2$, of 20); Adagrad's drops to zero ($7.0 \to 0.0$); and, most importantly, **Shrink-SGD and Shrink-Adagrad, zero collapse under every linear-architecture condition tested, collapse completely** (all 20 dimensions, every seed), while Shrink-Adam and Shrink-RMSprop remain at zero.

Optimizer	Collapsed Dims, Linear (/20)	Collapsed Dims, Nonlinear (/20)
SGD	0.00	0.0
Adam	6.80	11.8
RMSprop	6.60	11.2
Adagrad	7.00	0.0
Shrink-SGD	0.00	**20.0**
Shrink-Adam	0.00	0.0
Shrink-RMSprop	0.00	0.0
Shrink-Adagrad	0.00	**20.0**

Table 7: MNIST posterior collapse, linear vs. nonlinear (1 tanh hidden layer) architecture, both at $\alpha = 10^{-3}$, mean over 5 seeds. Two of the four shrinkage variants flip from zero to complete collapse.

Interpretation. We had hoped these checks would corroborate the clean narrative of Section 5.3. Instead, essentially every optimizer tested, including SGD and two of the four shrinkage variants, both described above as collapse-resistant, can be made to collapse under some combination of learning rate and architecture, and Adagrad's collapse in the main table depends on an unmatched learning rate. This does not invalidate Table 6 as a report of what happens at $\alpha = 10^{-3}$ with a linear architecture; it invalidates any general claim that some optimizers are intrinsically more prone to posterior collapse than others. The accurate summary: *posterior collapse here is governed by an interaction between optimizer, learning rate, and architecture for which we have no predictive theory*, and single-configuration comparisons, including our own synthetic-dataset comparison in Section 5.1, are single points in a much larger, only partially explored space, not general evidence about the optimizers.

One check replicated cleanly. The MNIST λ-sweep (Section 6.2, Figure 5) confirmed zero

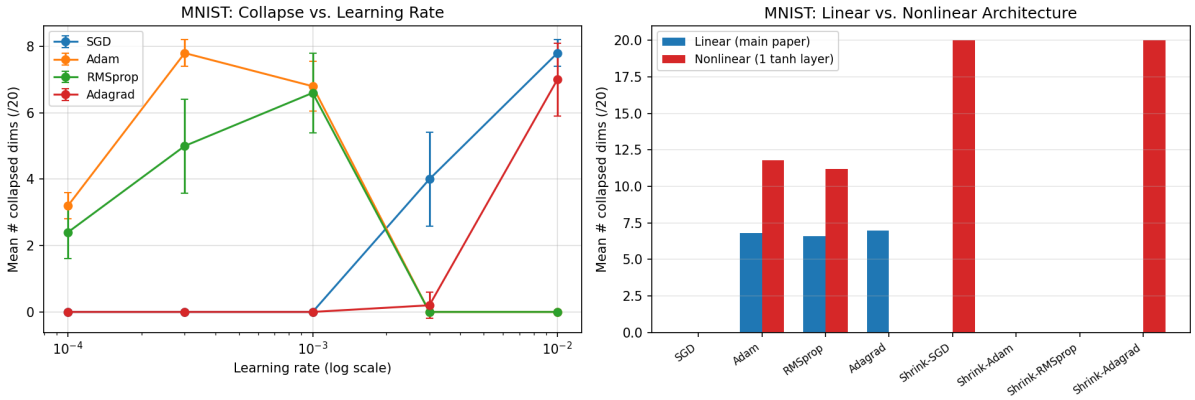


Figure 7: MNIST robustness checks (linear architecture unless noted, mean, 5 seeds). *Left:* posterior collapse vs. learning rate, no optimizer is uniformly resistant; Adam/RMSprop's collapse is non-monotonic and returns to zero at higher α, while SGD's collapse increases with α. *Right:* linear vs. nonlinear architecture at $\alpha = 10^{-3}$, Shrink-SGD and Shrink-Adagrad go from zero to complete collapse; Adagrad goes from substantial to zero collapse; Adam and RMSprop's collapse roughly doubles.

collapse for Shrink-SGD and Shrink-Adam across $\lambda \in \{0.01, \ldots, 0.5\}$ under the linear architecture, consistent with Table 6, though, as just shown, this breaks down once the architecture changes.

6.4 Limitations

Several limitations remain. First, the synthetic comparison uses only 160 ten-dimensional samples and a linear encoder–decoder; the MNIST experiment (Section 5.3) also uses a linear architecture and a 5,000-image subsample of the full 60,000, so absolute numbers should not be read as representative of a production-scale convolutional VAE. Second, the underlying dataset (synthetic or MNIST subsample) is drawn from a single fixed seed shared across all runs; varying it would further strengthen the statistics. Third, the MNIST reconstructions in Figures 8–8 are the original pre-existing qualitative illustrations, not regenerated to match Section 5.3's quantitative setup. Fourth, the shrinkage-augmented RMSprop/Adagrad variants use the same post-hoc rule as Shrink-SGD rather than the distinct, gradient-norm-based terms sketched generically in Section 3.1, which involve an unspecified second-order term and were not implemented. Fifth, AdamW/Nadam were compared on MNIST only at the default learning rate and linear architecture, where they track Adam almost exactly; whether they also do so under the perturbations of Section 6.3 is untested. Sixth, and most importantly: Section 6.3 shows the clean pattern in Table 6 (SGD/shrinkage resist collapse; Adam/RMSprop/Adagrad do not) is specific to $\alpha = 10^{-3}$ and the linear architecture and does not generalize even within MNIST, at higher α SGD collapses too, at some α Adam/RMSprop's collapse disappears, and under a nonlinear architecture two shrinkage variants collapse completely. This is a central finding of the revision, not a loose end: readers should not draw general conclusions about which optimizers are safe from posterior collapse from Table 6 alone.

7 Conclusion

We introduced a post-hoc multiplicative shrinkage step for VAE training and re-evaluated it with a multi-seed protocol on synthetic data and real MNIST, covering twelve optimizer configurations (SGD, Adam, RMSprop, Adagrad, AdamW, Nadam, each with/without shrinkage) and a per-dimension KL diagnostic for posterior collapse. On synthetic data, shrinkage produces faster, lower-variance convergence fully attributable to complete posterior collapse rather than better density estimation, uniformly across all six base optimizers, and stable once $\lambda \geq 0.05$ (Section 6.2). On MNIST at the default configuration ($\alpha = 10^{-3}$, linear architecture), shrinkage instead appears to *prevent* the partial collapse shown by the adaptive optimizers, but only by degrading reconstruction quality, and Shrink-Adagrad never converges.

The revision's most important finding, however, is that this MNIST pattern is not robust (Section 6.3). SGD collapses just as much as any adaptive method once the learning rate is raised; Adam/RMSprop's collapse is non-monotonic in the learning rate and vanishes outside a narrow band; Adagrad's apparent collapse is a learning-rate confound, not a property of Adagrad; and a nonlinear architecture causes Shrink-SGD and Shrink-Adagrad to collapse completely, the opposite of their linear-architecture behavior. Posterior collapse here is governed by an interaction between optimizer, learning rate, and architecture we cannot yet predict, and neither dataset nor any single configuration supports a general claim about which optimizers resist collapse.

We view this negative, self-correcting finding, including retracting our own cleaner-sounding earlier MNIST narrative, together with the reproducible implementation (NumPy code, MNIST pipeline, and robustness/sensitivity scripts), as more useful than either the original uncritical claim or an untested tidier revision. Natural next steps: (i) combine shrinkage with explicit collapse-mitigation (KL annealing, free-bits); (ii) develop a predictive account of when a given (optimizer, learning rate, architecture) combination induces collapse; (iii) extend this multi-configuration protocol to convolutional architectures and further datasets (Fashion-MNIST, CIFAR-10).

Declarations

Author Contributions
Conceptualization: H. Mokhtari Farivar; Methodology: H. Mokhtari Farivar; Software and Validation: H. Mokhtari Farivar; Formal Analysis: H. Mokhtari Farivar; Investigation: H. Mokhtari Farivar; Writing – Original Draft: H. Mokhtari Farivar; Writing – Review & Editing: both authors; Supervision: P. Nasiri; Funding Acquisition: none.

Acknowledgments
The authors received no specific funding for this research. The authors would like to thank the reviewers whose constructive comments substantially improved this manuscript.

Conflict of Interest
The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Artificial Intelligence Statement
Artificial intelligence (AI) tools, including large language models, were used solely for language editing and improving readability. AI tools were not used for generating ideas, performing analyses, interpreting results, or writing the scientific content. All scientific conclusions and intellectual contributions were made exclusively by the authors.

Publisher's Note
The publisher remains neutral regarding jurisdictional claims in published maps and institutional affiliations.

References

[1] Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*. https://doi.org/10.48550/arXiv.1312.6114

[2] Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392. https://doi.org/10.1561/2200000056

[3] Rezende, D. J., & Mohamed, S. (2015). Variational Inference with Normalizing Flows. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, PMLR 37:1530–1538. https://doi.org/10.48550/arXiv.1505.05770

[4] Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3), 400–407. https://doi.org/10.1214/aoms/1177729586

[5] Ruder, S. (2016). An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747*. https://doi.org/10.48550/arXiv.1609.04747

[6] Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, PMLR 32(2):1278–1286. https://doi.org/10.48550/arXiv.1401.4082

[7] Polyak, B. T. (1964). Some Methods of Speeding Up the Convergence of Iterative Methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5), 1–17. https://doi.org/10.1016/0041-5553(64)90137-5

[8] Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on Machine Learning*, PMLR 28(3):1139–1147.

[9] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*. https://doi.org/10.48550/arXiv.1412.6980

[10] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 1–13. https://doi.org/10.48550/arXiv.1412.6980

[11] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[12] Tipping, M. E., & Bishop, C. M. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B*, 61(3):611–622. https://doi.org/10.1111/1467-9868.00196

[13] Tieleman, T., & Hinton, G. (2012). Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

[14] Hinton, G. E. (2012). *Neural Networks for Machine Learning*. Coursera.

[15] Higgins, I., Matthey, L., Pal, A., et al. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations (ICLR)*.

[16] Doersch, C. (2016). Tutorial on Variational Autoencoders. *arXiv preprint arXiv:1606.05908*. https://doi.org/10.48550/arXiv.1606.05908

[17] Duchi, J. C., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121–2159.

[18] Duchi, J., & Singer, Y. (2009). Efficient Online and Batch Learning Using Forward Backward Splitting. *Journal of Machine Learning Research*, 10, 2899–2934.

[19] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2016). Generating Sentences from a Continuous Space. *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, 10–21. https://doi.org/10.18653/v1/K16-1002

[20] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved Variational Inference with Inverse Autoregressive Flow. *Advances in Neural Information Processing Systems (NeurIPS)*, 29:4743–4751. https://doi.org/10.48550/arXiv.1606.04934

[21] Rezende, D. J., & Viola, F. (2018). Taming VAEs. *arXiv preprint arXiv:1810.00597*. https://doi.org/10.48550/arXiv.1810.00597

[22] Zhao, S., Song, J., & Ermon, S. (2017). InfoVAE: Information Maximizing Variational Autoencoders. *arXiv preprint arXiv:1706.02262*. https://doi.org/10.48550/arXiv.1706.02262

[23] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324. https://doi.org/10.1109/5.726791

[24] Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1711.05101

[25] Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. *ICLR Workshop Track*.

A Supplementary Qualitative Figures and Single-Sample Example

This appendix collects supplementary illustrative material not part of the quantitative comparisons in Sections 5–6: a single-sample reconstruction example (Table 8) and the two pre-existing qualitative reconstruction figures from the original submission (Figures 8–8; Section 4.5 for scope).

Feature	Original Value	Reconstructed Value	Absolute Error
1	0.85	0.82	0.03
2	0.60	0.58	0.02
3	0.45	0.50	0.05
4	0.78	0.75	0.03
5	0.92	0.90	0.02
6	0.67	0.70	0.03
7	0.55	0.53	0.02
8	0.88	0.85	0.03
9	0.75	0.73	0.02
10	0.66	0.64	0.02

Table 8: Single-sample comparison of original and reconstructed feature values (illustrative example, Shrinkage Optimizer; shown separately from the aggregate loss statistics in Tables 2–4 and should not be read as an aggregate or averaged result).



Figure 8: Qualitative reconstructions (Section 4.5): natural images (left, not in Table 2) and MNIST digits (right, Section 5.3).

Author(s) Bio-sketches

Heydar Mokhtari Farivar is affiliated with Iran University of Science and Technology (IUST). His research interests include machine learning, variational inference, and optimization. Corresponding author: mokhtari@iust.ac.ir

Parviz Nasiri is a professor with Payame Noor University (PNU). His research area is statistical inference. pnasiri@pnu.ac.ir