Dimensionality Reduction Techniques: A Comparison of Principal Component Analysis and Linear Autoencoders on the MNIST Dataset

Nick Meyer, Ayo Adetayo, Joshua Were, Roger Fortunato and Arjun Purohit
Columbia University

December 15, 2025

Abstract

Principal Component Analysis (PCA) and linear autoencoders (LAEs) both perform linear dimensionality reduction by minimizing squared reconstruction error, and the Baldi-Hornik theorem establishes their theoretical equivalence at global optima [1]. However, the practical conditions under which gradient-based autoencoder training recovers the PCA subspace remain underexplored. This study systematically evaluates the architectural and optimization requirements for subspace convergence using the MNIST handwritten digit dataset. We compare PCA and LAEs across three experimental conditions: (1) unconstrained architecture with bias terms and independent encoder-decoder weights, (2) constrained architecture satisfying Baldi-Hornik conditions (tied weights, no bias), and (3) optimized hyperparameter configurations on the full 70,000-sample training set. Subspace alignment is quantified using principal angles and Grassmann distance, and our results demonstrate that the architectural constraints are essential: unconstrained linear autoencoders achieve mean principal angles of 21-50° relative to PCA, despite identical reconstruction errors (MSE difference $< 4\%$), and constrained linear autoencoders reduce angles to 2.7-4.5°, representing a 6-11x improvement. With optimized hyperparameters—specifically cosine learning rate scheduling, 200 training epochs, and orthogonal initialization—constrained autoencoders achieve near-perfect alignment with mean angles as low as 0.12° for latent dimension k=128. Hyperparameter analysis reveals that learning rate scheduling is the dominant factor (30× impact), followed by learning rate magnitude (5× impact), with initialization showing negligible effect. These findings confirm that the Baldi-Hornik equivalence holds empirically under appropriate architectural constraints and optimization configurations, clarifying when linear autoencoders faithfully recover the PCA subspace in practice.

1 Introduction

High-dimensional datasets are increasingly common across scientific, industrial, and machine learning applications. Images, audio signals, genomic sequences, and large-scale embeddings often contain thousands of features per sample, creating challenges for storage, computation, visualization, and learning. As dimensionality grows, models face greater risk of overfitting, optimization slows, and the underlying structure of the data becomes harder to interpret. Dimensionality reduction techniques address these problems by compressing data into a smaller set of informative features, enabling more efficient and interpretable downstream analysis [6].

Principal Component Analysis (PCA) is the classical linear approach for this task. By identifying orthogonal directions of maximal variance, PCA provides an optimal low-dimensional representation in terms of reconstruction error and remains widely used in statistics, signal processing, and

machine learning pipelines [2]. Linear autoencoders (LAEs) offer an alternative [3]: rather than computing a closed-form solution, they learn compressed representations through neural network training, optimizing the same squared reconstruction objective using gradient-based methods. Although theory shows that a perfectly trained linear autoencoder recovers the same subspace as PCA, practical training introduces factors—initialization, optimization dynamics, regularization—that may influence how closely the learned representation matches the PCA solution [1].

This work conducts a systematic empirical comparison of PCA and linear autoencoders on the MNIST dataset. Beyond reconstruction quality, we examine subspace similarity, training behavior, sensitivity to hyperparameters, and downstream classification performance. Our goal is to clarify when the representations learned by linear autoencoders align with PCA, when they diverge, and what these differences imply for the use of linear dimensionality reduction in modern machine learning workflows.

1.1 Our Contribution

Although the theoretical relationship between PCA and linear autoencoders is well established, existing empirical work often evaluates these methods on a limited set of criteria, focusing primarily on reconstruction accuracy or visual intuition. Much less attention has been given to how reliably linear autoencoders recover the PCA subspace under practical training conditions, how sensitive this recovery is to hyperparameters and initialization, or whether geometric similarity between the two methods translates into comparable performance in downstream machine learning tasks.

This paper contributes a systematic and quantitative comparison of PCA and linear autoencoders along several dimensions:

1. We examine subspace alignment using principal angles and Grassmann distance, providing a direct geometric assessment of how closely autoencoder representations match PCA.
2. We study training dynamics and hyperparameter sensitivity, evaluating the stability of convergence and the influence of factors such as weight decay and latent dimensionality.
3. We compare the usefulness of PCA and autoencoder representations in a downstream classification task to determine whether differences in representation geometry have practical consequences.

Together, these analyses offer a clearer empirical picture of when linear autoencoders behave like PCA, when they diverge, and what these differences imply for practitioners choosing between analytic and learned approaches to linear dimensionality reduction.

2 Theoretical Background

This section provides the mathematical foundations for the two dimensionality reduction methods examined in this study. We begin by outlining the formulation of Principal Component Analysis, followed by a description of linear autoencoders and their training objective. We then summarize the classical result establishing the conditions under which a linear autoencoder recovers the PCA subspace. These concepts form the basis for the empirical comparisons presented later in the paper.

2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a linear technique for reducing the dimensionality of a dataset while preserving as much of its variation as possible. Given data $X \in \mathbb{R}^{n \times d}$, PCA begins

by centering each feature so that the analysis describes variation around the mean rather than around the origin:

$$\tilde{X} = X - \mu, \qquad \mu = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{1}$$

From the centered data, PCA forms the covariance matrix:

$$C = \frac{1}{n}\tilde{X}^\top \tilde{X}. \tag{2}$$

using $1/n$ scaling. This choice does not affect the eigenvectors and therefore does not change the principal subspace used in our analysis. The key step is the eigendecomposition of this matrix:

$$Cv_i = \lambda_i v_i. \tag{3}$$

Each eigenvector v_i indicates a direction in the feature space, and its corresponding eigenvalue λ_i measures the variance of the data along that direction. Ordering the eigenvalues from largest to smallest yields the directions that capture the major modes of variation in the dataset.

Selecting the top k eigenvectors forms the principal subspace [2]:

$$V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{d\times k}. \tag{4}$$

Projecting the data onto this subspace produces a lower-dimensional representation:

$$Z = \tilde{X}V_k, \tag{5}$$

$$\hat{X} = ZV_k^\top + \mu. \tag{6}$$

A defining property of PCA is that this projection is optimal among all linear mappings from $\mathbb{R}^d$ to $\mathbb{R}^k$: it minimizes the squared reconstruction error between the original data and its reconstruction [7].

The eigenvalues also quantify how much information each component preserves. The explained variance ratio

$$EVR_i = \frac{\lambda_i}{\sum_{j=1}^{d}\lambda_j} \tag{7}$$

indicates the proportion of total variance captured by the ith principal component [2], and the cumulative explained variance provides a straightforward way to select an appropriate value of k.

By combining geometric interpretability with an exact analytic solution, PCA provides a natural benchmark for evaluating learned linear representations, such as those produced by autoencoders, which seek to approximate the same objective through optimization rather than closed-form computation [5].

2.2 Linear Autoencoders

The neural network employed here is a linear autoencoder trained to learn an efficient representation of data. It consists of two linear transformations: an encoder that linearly maps a high-dimensional input $x \in \mathbb{R}^d$ to a lower-dimensional latent representation $h \in \mathbb{R}^k$ and a decoder that linearly maps h back to a reconstruction of x, usually denoted as $\hat{x} \in \mathbb{R}^d$. Between these two layers of the neural network sits the latent space or bottleneck layer, a compressed, lower-dimensional representation

of the data, where the essential features are preserved. Here, the latent space consists of only a single "hidden" or bottleneck layer.

Below, we summarize the transformations taking place:

$$h = W_E x + b_E, \tag{8}$$
$$\hat{x} = W_D h + b_D. \tag{9}$$

where $x \in \mathbb{R}^d$, $h \in \mathbb{R}^k$, $W_E \in \mathbb{R}^{k \times d}$, and $W_D \in \mathbb{R}^{d \times k}$. We can see that the full reconstruction process is of the form:

$$\hat{x} = W_D W_E x + (W_D b_E + b_D). \tag{10}$$

To train the network, we seek to minimize the reconstruction error over the dataset. We utilize the Mean Squared Error (MSE) loss function, which corresponds to the same squared reconstruction objective optimized by PCA when both methods are applied to centered data [1]:

$$L(W_E, W_D) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2. \tag{11}$$

Unlike PCA, which solves for the optimal subspace analytically via eigendecomposition, the linear autoencoder solves for W_E and W_D iteratively using gradient descent. The gradients of the loss function with respect to the weights are computed via backpropagation, and the weights are updated to move toward a minimum of L.

It is, however, important to note that the majority of contemporary neural networks differ from our experimental structure. Many of today's neural networks apply non-linear transformations and contain multiple hidden layers [3]. Including non-linearity and multiple hidden layers allows models to efficiently learn more complex tasks.

It is in service of our research objective of assessing the mathematical equivalence of PCA and linear autoencoders that we have instead chosen to only include linear transformations and a single bottleneck layer. Neural network representations may only converge to equivalence with PCA through linear transformations alone. Further, having only a single hidden layer allows us to more easily and accurately compare the results of the two dimensionality reduction techniques.

2.3 Theoretical Basis and Equivalence

PCA and linear autoencoders originate from different areas of research but share a common mathematical goal: to construct a low-dimensional linear representation of high-dimensional data. This section outlines the theoretical foundations that motivate comparing these two methods.

PCA as the Optimal Linear Subspace

The Eckart–Young–Mirsky theorem [5, 7] establishes PCA as the optimal rank-k linear approximation of a dataset under squared reconstruction error. PCA identifies the k-dimensional subspace that minimizes

$$\|X - \hat{X}\|_F^2 \tag{12}$$

and can be viewed as the solution to the constrained optimization problem

$$\min_{W} \|X - XWW^\top\|_F^2 \quad \text{subject to} \quad WW^\top = I_k. \tag{13}$$

This formulation emphasizes PCA as a principled optimization method that identifies the best linear subspace for reconstruction, making it a natural benchmark for comparison.

Linear Autoencoders as Learned Low-Rank Models

A linear autoencoder reconstructs data using two linear transformations [4],

$$z = W_E x, \qquad \hat{x} = W_D z, \tag{14}$$

and is trained to minimize the same squared reconstruction error as PCA. The composite mapping $W_D W_E$ has rank at most k, meaning the autoencoder is also learning a rank-k approximation of the dataset [4]. Unlike PCA, which obtains the optimal solution analytically, a linear autoencoder relies on gradient-based optimization to discover this subspace [3, 4].

The Baldi–Hornik Equivalence Result

Baldi and Hornik [1] showed that at any global minimum of the autoencoder loss, the encoder spans the same subspace as the top k principal components:

$$\text{rowspace}(W_E) = \text{span}(v_1, \ldots, v_k). \tag{15}$$

Although the individual weight matrices are not unique, the learned subspace is equivalent. This provides the theoretical justification for comparing PCA and linear autoencoders.

Measuring Subspace Alignment

Because both methods produce subspaces rather than unique matrices, we evaluate their similarity using principal angles, with 0° indicating perfect alignment. We also compute Grassmann distance, which summarizes alignment using the norm of the principal angle vector. These measures allow us to quantify how closely the autoencoder converges toward the PCA subspace in practice.

Optimization Considerations

While PCA guarantees the optimal solution, the optimization landscape of linear autoencoders contains saddle points and suboptimal stationary points. Convergence depends on initialization, learning rate, and other hyperparameters. These practical differences motivate our empirical analysis of reconstruction accuracy, subspace similarity, and convergence behavior.

3 Methodology and Experimental Setup

This section describes the dataset, preprocessing pipeline, model configurations, and evaluation metrics used to compare PCA and linear autoencoders. Our aim is to create a controlled experimental setting in which both methods operate on the same data and are evaluated using consistent criteria, allowing any differences in performance or learned representations to be meaningfully interpreted. All technical procedures—including data loading, preprocessing, model training, and evaluation—are fully documented in our publicly accessible GitHub repository (see Appendix), ensuring transparency and reproducibility of the experimental results.

3.1 MNIST Dataset

In this project, in order to compare the performance of the PCA approach and the Linear Autoencoder approach, we utilize the Modified National Institute of Standards and Technology (MNIST) database. This is a large database of handwritten digits that is commonly used for training image recognition systems. The database contains 60,000 examples in the training set and 10,000 in the test set. To enable rapid experimentation during hyperparameter exploration, Experiments 1 and 2 use a 10,000-sample subset of the training data. Experiment 3 scales to the full 70,000-sample dataset (combining the 60,000 training samples with the 10,000 test samples) to evaluate convergence with maximum available data. Each image in the database is a grayscale image with a resolution of 28×28 pixels. In order to make these images readily analyzed, some preprocessing will be necessary. First, flattening each image from a 2-dimensional matrix to a 1-dimensional vector. We also normalize the pixel values from [0,255] to [0,1] to make the data easier to handle. Finally for the PCA approach specifically we center the data by subtracting the mean pixel value from the value of each individual pixel.

3.2 PCA Setup

To establish a rigorous analytical baseline for comparison, we implement Principal Component Analysis (PCA) directly using NumPy [2]. This choice ensures full transparency of the underlying linear algebra and aligns the experiment with the theoretical treatment presented in Section 2.3. By constructing PCA from first principles, we obtain a ground-truth linear dimensionality-reduction method against which the autoencoder's learned subspace can be meaningfully evaluated.

PCA seeks a rank-k linear projection that minimizes reconstruction error, and its solution is obtained analytically through eigendecomposition. Our implementation explicitly mirrors this formulation.

Analytical Procedure

Given the normalized MNIST data, we perform PCA through the following steps:

1. **Mean-Centering.** We subtract the feature-wise mean vector computed across the training set. Centering is a necessary condition for PCA, ensuring that principal components reflect directions of maximal variance rather than the global mean intensity of the digits.

2. **Covariance Matrix Construction.** We compute the empirical covariance matrix using

$$C = \frac{1}{n} X^\top X, \tag{16}$$

   where X denotes the centered data. This step aligns with the optimization formulation of PCA and identifies the symmetric matrix whose eigenvectors define the optimal subspace.

3. **Eigenvalue Decomposition.** We perform an eigenvalue decomposition using NumPy's `linalg.eigh`, which is optimized for symmetric matrices. This yields a complete set of eigenvalues and eigenvectors representing, respectively, the variance explained and the directions of the principal components.

4. **Principal Component Selection.** Eigenvalues are sorted in descending order, and the corresponding top k eigenvectors are retained to form

$$W_{\text{PCA}} \in \mathbb{R}^{k \times 784}. \tag{17}$$

These vectors are orthonormal and constitute the optimal rank-k linear representation under the reconstruction objective.

5. **Projection and Reconstruction.** The PCA class implements:

   - *Transform:*
   $$Z = (X - \mu)W_{\text{PCA}}^{\top}, \tag{18}$$
   - *Inverse transform:*
   $$\hat{X} = ZW_{\text{PCA}} + \mu. \tag{19}$$

   This structure parallels the encoder–decoder pattern of the autoencoder, allowing the two methods to be compared under identical representation and reconstruction operations.

6. **Explained Variance Quantification.** We compute both the per-component explained variance and the cumulative explained variance ratio. These metrics contextualize the effectiveness of each chosen dimensionality and provide a basis for evaluating reconstruction fidelity.

Dimensionality Selection

To ensure comparability with the linear autoencoder, we evaluate PCA using the same latent dimensionalities (e.g., $k = 8, 16, 32$). Matching the dimensionality across models controls for representational capacity and ensures that differences in performance reflect methodological differences rather than differences in embedding size.

Rationale for This Configuration

Implementing PCA from scratch allows us to:

- directly evaluate the theoretical PCA subspace on which the autoencoder is expected to converge,
- ensure methodological symmetry between PCA and the autoencoder,
- perform fine-grained subspace comparisons using principal angles and Grassmann distance, and
- establish an exact analytical reference point for reconstruction error and variance preservation.

This PCA configuration therefore provides the mathematical baseline for all subsequent empirical evaluations.

3.3 Linear Autoencoder Setup

To parallel the analytical PCA baseline, we implement a linear autoencoder using PyTorch, configured to match the theoretical framework outlined in Section 2.3. The objective is to determine whether a model trained via gradient-based optimization can recover the same low-dimensional subspace that PCA derives analytically. For this reason, we deliberately constrain the architecture to its classical linear form, excluding nonlinear activation functions or additional hidden layers. This removes representational flexibility and isolates the mathematical relationship between the two methods.

The autoencoder consists of two mappings:

$$z = W_E x, \qquad \hat{x} = W_D z, \tag{20}$$

where $W_E \in \mathbb{R}^{k\times 784}$ is the encoder and $W_D \in \mathbb{R}^{784\times k}$ is the decoder. This structure ensures that the learned transformation $W_D W_E$ has rank at most k, placing the model within the class of linear dimensionality-reduction methods and aligning it with the theoretical conditions under which linear autoencoders coincide with PCA at optimality.

Architecture

The model is implemented using two fully connected layers:

- **Encoder:** $784 \to k$
- **Decoder:** $k \to 784$

with no activation functions. This architectural choice is essential, as any nonlinearity would break the equivalence to PCA and prevent direct subspace comparison. The resulting network is thus a faithful instantiation of the linear autoencoder model analyzed in the Baldi–Hornik equivalence theorem.

Training Procedure

Training is performed using the MNIST training set under the following protocol:

- **Loss function:** Mean Squared Error (MSE)
- **Optimizer:** Adam
- **Batch size:** 128
- **Learning rate:** 0.001 baseline; varied in Experiment 3 hyperparameter search
- **Epochs:** 50 in Experiments 1–2; varied in Experiment 3 hyperparameter search

The loss function is identical to the reconstruction objective minimized by PCA, ensuring conceptual comparability. For each epoch, we record the mean reconstruction loss, which is later used to analyze convergence behavior and the stability of gradient descent.

Preprocessing and Mean Tracking

To ensure strict comparability with PCA, the linear autoencoder is trained and evaluated on the same mean-centered inputs $\tilde{X} = X - \mu$ used in PCA. Reconstructions are mapped back to the original pixel space by adding μ only when reporting image examples or reconstruction errors in the original scale. This ensures that both reconstruction metrics and subspace comparisons are defined with respect to the same centered data geometry.

Dimensionality Selection

The autoencoder is trained using the same latent dimensionalities applied in the PCA experiments (e.g., $k = 8, 16, 32$). Matching dimensionality across models eliminates capacity-related confounds and ensures that observed differences reflect the learning dynamics rather than the expressiveness of the latent space.

Rationale for This Configuration

By constraining the model to a purely linear architecture and aligning preprocessing, learning objectives, and latent dimensionality with PCA, the experiment isolates the method-specific differences arising from optimization. This configuration enables a direct empirical test of the theoretical prediction that linear autoencoders recover the PCA subspace when trained successfully and provides a controlled environment for examining the practical limitations of gradient-descent-based subspace learning.

3.4 Evaluation Metrics

To compare PCA and the linear autoencoder under a unified experimental framework, we evaluate both methods using a set of metrics that capture reconstruction fidelity, representational similarity, and optimization behavior. These metrics are chosen to reflect both the theoretical foundations discussed in Section 2 and the practical considerations that arise during empirical training. Together, they allow for a comprehensive assessment of the extent to which the autoencoder approximates the PCA solution.

1. **Reconstruction Error.** Reconstruction error quantifies how well a model preserves the information contained in the original images. For both PCA and the autoencoder, we compute the mean squared reconstruction error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\|x_i - \hat{x}_i\|_2^2. \tag{21}$$

   Because PCA and the linear autoencoder can both be formulated as minimizing a squared reconstruction objective for rank-k linear representations, this metric provides a direct measure of relative efficiency in capturing the data's dominant structure. Lower MSE indicates that the model retains more information in the compressed representation.

2. **Subspace Alignment and Principal Angles.** Since PCA and the autoencoder do not necessarily recover identical basis vectors but rather aim to learn the same underlying subspace, we evaluate the alignment between the two subspaces using principal angles [4]. These angles measure the degree of correspondence between the k-dimensional PCA subspace and the subspace spanned by the encoder weights:

   - 0° indicates perfect alignment,
   - intermediate angles indicate partial overlap,
   - 90° indicates orthogonality.

   Principal angles directly reflect the theoretical claim that linear autoencoders should converge to the PCA subspace under ideal optimization conditions.

3. **Grassmann Distance.** To summarize subspace alignment in a single scalar quantity, we compute the Grassmann distance, defined as the Euclidean norm of the principal angle vector. This metric captures the overall separation between the two subspaces and facilitates comparison across different dimensionalities. A smaller Grassmann distance indicates closer agreement between PCA and the autoencoder.

4. **Convergence Behavior.** Because PCA obtains its solution analytically while the autoencoder relies on iterative optimization, we track the autoencoder's training loss across epochs. This allows us to evaluate: (1) the rate at which the model converges, (2) whether convergence is stable or oscillatory, and (3) the presence of optimization difficulties such as plateaus or slow descent. These observations provide insight into the practical limitations discussed in Section 2.3 and help contextualize any deviations from the PCA subspace.

5. **Visual Reconstruction Comparison.** In addition to numerical metrics, we visualize reconstructions produced by both methods. Comparing reconstructed digits to the original images allows for qualitative assessment of: (1) the type of information each method preserves, (2) the sharpness or smoothness of reconstructions, and (3) whether artifacts arise from the learned representation. These visual comparisons complement the quantitative metrics and provide intuitive insight into the effectiveness of each method.

4 Results

This section presents the empirical results of our comparison between PCA and LAEs on the MNIST dataset. We structure our findings around four key dimensions: reconstruction fidelity, subspace alignment, convergence behavior, and hyperparameter sensitivity. Our experiments systematically test the conditions of the Baldi-Hornik theorem, examining how architectural constraints—specifically tied weights and the absence of bias terms—influence whether the LAE recovers the PCA subspace.

We conducted three primary experiments. Experiment 1 used an unconstrained LAE (with bias terms and untied weights) on 10,000 samples. Experiment 2 applied the Baldi-Hornik constraints (no bias, tied weights) to the same 10,000 samples. Experiment 3 scaled to the full 70,000-sample training set while systematically searching for optimal hyperparameter configurations.

4.1 Reconstruction Fidelity

Across all experiments and configurations, PCA and the linear autoencoder achieved nearly identical reconstruction performance, as measured by mean squared error (MSE). This finding holds regardless of whether the Baldi-Hornik constraints were applied, demonstrating that the optimization objective—rather than the solution path—determines reconstruction quality.

k	PCA MSE	LAE MSE	Difference
8	0.0381	0.0385	1.1%
32	0.0173	0.0174	0.2%
64	0.0094	0.0095	1.0%
128	0.0044	0.0045	3.6%

Table 1: Reconstruction Error Comparison (Experiment 1 — Unconstrained LAE)

The constrained LAE (Experiment 2) produced similarly close results, with differences ranging from 0.6% to 3.4% across all dimensionalities. These results confirm that both PCA and the linear autoencoder converge to representations that capture equivalent amounts of variance, regardless of the specific subspace they inhabit.

Classification accuracy using a downstream classifier trained on the reduced representations showed comparable patterns. For k=128, both methods achieved approximately 90% accuracy,

while k=8 yielded around 77% accuracy for both. This demonstrates that the learned representations are not only equivalent in reconstruction fidelity but also in their utility for downstream tasks.

Below is an example reconstruction from experiment 1 with k=64 in which we can see the similarity in the two reconstructions.

Figure 1: Data Reconstruction. Top to bottom: Original, PCA, LAE (Experiment 1)

4.2 Subspace Alignment

While reconstruction performance was equivalent across configurations, the geometric relationship between PCA and LAE subspaces varied dramatically based on architectural constraints. This section examines subspace alignment using principal angles and Grassmann distance, providing direct empirical evidence for the Baldi-Hornik theorem.

Experiment 1 (Unconstrained LAE): Without tied weights and with bias terms, the LAE learned subspaces that were fundamentally different from the PCA solution. Mean principal angles ranged from 21.5 to 49.6°, with maximum angles reaching as high as 87.7° (near-orthogonality). Grassmann distances ranged from 2.50 to 5.03, indicating substantial geometric separation.

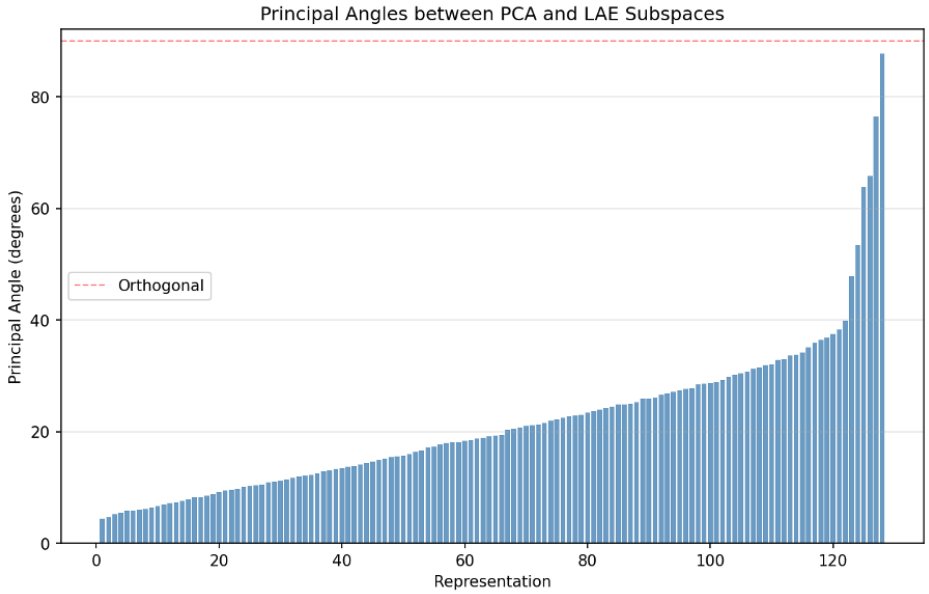

Figure 2: Principal Angles between PCA and LAE Subspaces (Experiment 1, $k = 128$)

Experiment 2 (Constrained LAE): Applying tied weights and removing bias terms produced a dramatic improvement in subspace alignment. Mean principal angles dropped to 2.7–4.5°, representing a 6–11× improvement over the unconstrained case. Grassmann distances decreased correspondingly, from 2.50–5.03 to 0.42–2.04.

k	Mean (Exp1)	Max (Exp1)	Mean (Exp2)	Max (Exp2)	Improvement
8	49.6°	71.9°	4.5°	23.3°	11.0×
32	30.4°	53.3°	2.7°	38.9°	11.3×
64	25.2°	73.9°	2.9°	75.2°	8.7×
128	21.5°	87.7°	3.5°	74.4°	6.1×

Table 2: Subspace Alignment — Unconstrained vs. Constrained LAE

These results provide strong empirical confirmation of the Baldi-Hornik theorem. Without the constraints, the optimization landscape admits many distinct solutions with equivalent reconstruction error but entirely different geometric structure.

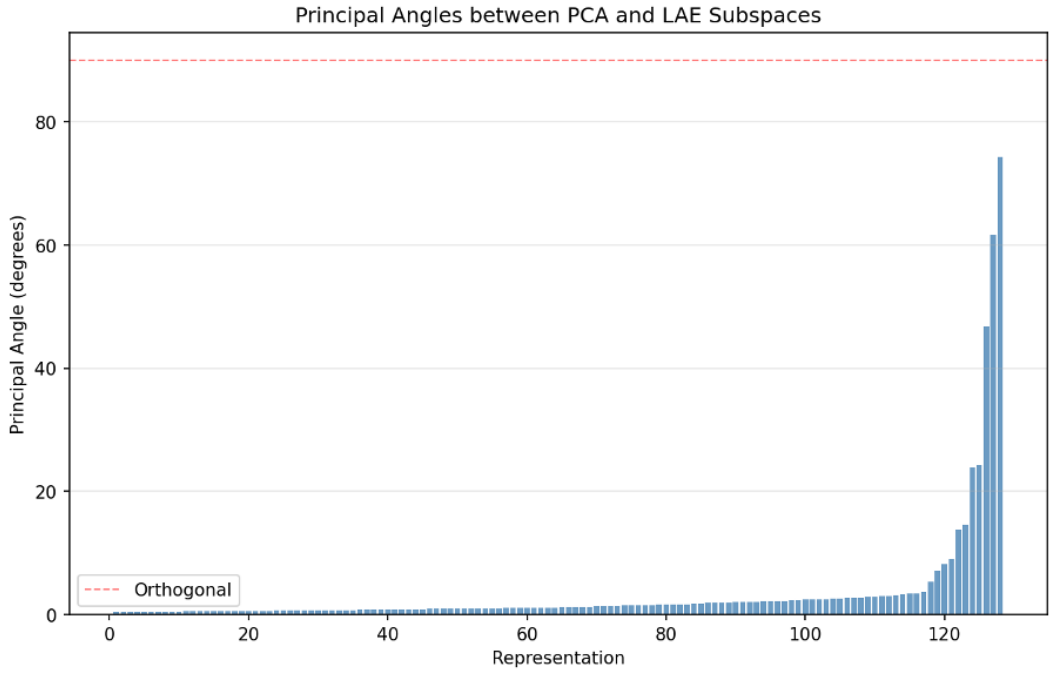

Figure 3: Principal Angles between PCA and LAE Subspaces (Experiment 2, $k = 128$)

Interpretation: The unconstrained LAE's freedom to explore a larger solution space explains the divergence. Bias terms break the centering assumption implicit in PCA, allowing affine transformations rather than pure linear mappings [1]. Untied weights introduce rotational freedom; any rotation in the latent space that preserves reconstruction is equally valid. Together, these degrees of freedom create infinitely many equivalent solutions, and stochastic gradient descent simply finds one of them, with no guarantee of finding the PCA solution.

Experiment 3 Results: Scaling to the full 70,000-sample dataset with optimized hyperparameters further improved alignment. Using cosine learning rate scheduling, 200 epochs, and orthogonal initialization, we achieved mean angles as low as 0.12° for $k = 128$ and 0.16° for $k = 64$. This

demonstrates that with sufficient data, training time, and proper optimization, the constrained LAE can achieve near-perfect convergence to the PCA subspace [4].

Table 3: Best Configurations from Experiment 3 (70K Samples)

k	Mean Angle	Epochs	Initialization	LR Schedule	Run #
8	6.36°	50	default	cosine	2
32	0.28°	200	default	cosine	22
64	0.16°	200	orthogonal	cosine	36
128	0.12°	200	orthogonal	cosine	48

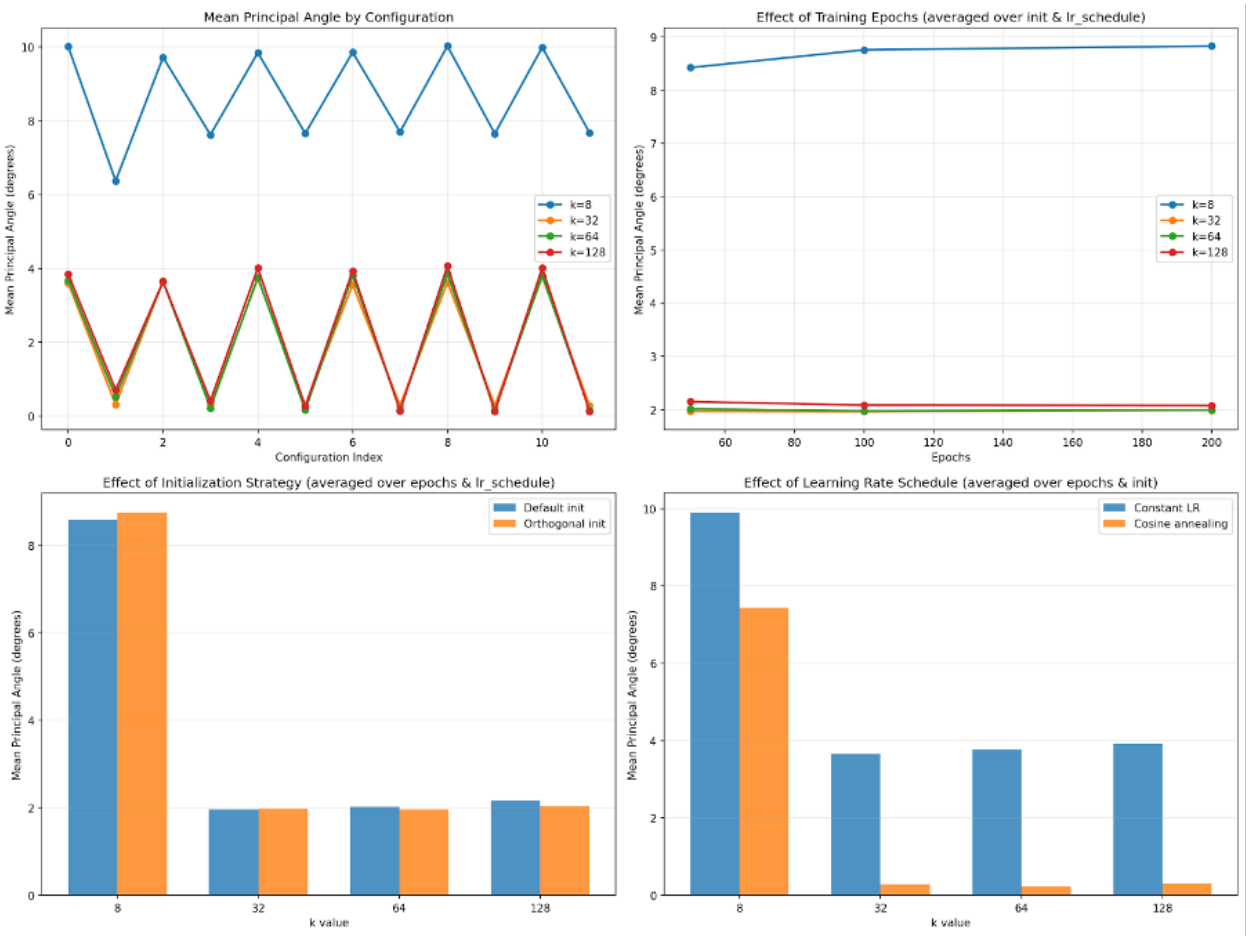

Figure 4: Summary Results of Experiment 3

The above figure illustrates the relative importance of each hyperparameter on subspace alignment. The top-left panel shows that bottleneck dimension k is the primary determinant of convergence quality. The top-right demonstrates that training converges quickly, with all configurations plateauing by epoch 100 and minimal improvement thereafter. The bottom-left panel reveals that the initialization strategy has negligible impact, producing nearly identical mean angles across all k values. Most critically, the bottom-right panel shows that learning rate scheduling is the decisive factor for optimization quality: cosine annealing reduces mean angles to near 0° for $k \geq 32$, while

constant learning rates plateau at 3.5–4°, representing an order-of-magnitude difference in subspace alignment. These visualizations confirm that while adequate latent dimension is necessary for convergence, proper learning rate scheduling is essential for achieving the tight alignment predicted by theory.

4.3 Convergence Analysis

The convergence behavior of the linear autoencoder reveals important practical considerations for achieving alignment with the PCA subspace. Several factors contribute to residual angles even under ideal architectural constraints.

Finite Sample Effects: PCA computes eigenvectors of the sample covariance matrix, which is itself an estimate of the true population covariance [7]. The LAE learns from mini-batches drawn from the same finite sample. Both methods are estimating the "true" underlying subspace from noisy finite data. Our results show that scaling from 10,000 to 70,000 samples improved alignment, suggesting that with more data, both methods converge closer to the true subspace.

Stochastic Optimization: The LAE uses mini-batch stochastic gradient descent with the Adam optimizer. Each batch provides a noisy gradient estimate, and the optimizer never reaches the exact minimum—it only gets very close. Random batch sampling order affects the final solution. Even with perfect convergence guarantees, there is always residual error.

Training Duration: Our experiments show diminishing returns after 100 epochs, though 200 epochs consistently achieved the lowest angles. The Baldi-Hornik theorem guarantees convergence only in the limit as training time approaches infinity. In practice, we must balance computational cost against marginal improvements in alignment.

Eigenvalue Degeneracy: The presence of large maximum angles is consistent with eigenvalue near-degeneracy [7]. When consecutive eigenvalues are similar in magnitude, rotational ambiguity within the principal subspace can lead PCA and the LAE to select different bases [7]. The LAE and PCA may choose slightly different bases within this degenerate subspace, contributing to the large maximum angles observed at k=64 and k=128.

4.4 Hyperparameter Sensitivity

Our systematic hyperparameter search revealed a clear hierarchy of importance for achieving PCA subspace convergence.

1. Learning Rate Schedule: This proved to be the most impactful factor. Cosine annealing learning rate schedules dramatically outperformed constant learning rates across all configurations. Constant learning rates consistently produced mean angles of 3-10°, while cosine scheduling reduced angles to 0.12-7.6° depending on k and epoch count. The smooth decay of cosine annealing allows the optimizer to make large updates early in training and fine-tune as it approaches convergence.

2. Learning Rate Magnitude: Within constant learning rate experiments (Experiment 2), we found that learning rate selection is critical. A learning rate of 0.0001 produced mean angles of 13.7° (underconverged due to slow optimization), while 0.01 produced 10.9° (overshooting the minimum). The optimal value of 0.001 achieved mean angles of 2.7°.

3. Batch Size: Smaller batch sizes (64) marginally outperformed larger ones (128, 256). With batch size 64, mean angles reached 2.8° compared to 3.4° with batch size 256. This suggests that the stochastic noise from smaller batches may help escape shallow local minima, though the effect is moderate compared to learning rate selection.

4. Initialization: Orthogonal initialization provided marginal benefits for higher-dimensional latent spaces (k=64 and k=128), saving approximately 0.04° compared to default initialization. For lower dimensions (k=8 and k=32), the difference was negligible. While theoretically appealing—given that PCA eigenvectors are orthogonal—the practical impact was smaller than anticipated.

5. Random Seed: Different random seeds produced variability of approximately 1.2° in mean angles (range: 2.5-3.7°). While initialization matters for reproducibility, it is substantially less important than learning rate configuration for achieving tight convergence.

5 Conclusion

5.1 Summary of Findings

This study provides comprehensive empirical validation of the Baldi-Hornik theorem on the MNIST dataset [1]. Our experiments demonstrate three key findings:

1. **Architectural constraints are essential for subspace recovery.** An unconstrained linear autoencoder (with bias terms and untied weights) learns subspaces with mean principal angles of 21–50° relative to PCA, despite achieving equivalent reconstruction error and classification accuracy. Applying tied weights and removing bias terms reduces mean angles by 6–11×, bringing them to 2–5° in baseline experiments and below 1° with optimized hyperparameters.

2. **Multiple equivalent subspaces exist for MNIST.** The unconstrained LAE can achieve identical performance while inhabiting a fundamentally different subspace, confirming that the optimization landscape contains multiple geometrically distinct solutions with equivalent reconstruction and classification performance.

3. **Practical convergence requires careful optimization.** Even with correct architectural constraints, achieving tight alignment (below 1°) requires appropriate hyperparameter selection. Cosine learning rate scheduling proved most impactful, followed by learning rate magnitude. With 70,000 samples, 200 epochs, and cosine annealing, we achieved mean angles as low as 0.12° for $k = 128$.

5.2 Theoretical Implications

Our results confirm that the Baldi-Hornik theorem holds in practice, not just in theory. The tied-weight and no-bias constraints are sufficient conditions for the LAE to converge to the PCA subspace, given adequate data and training time [1]. However, they are not necessary for equivalent performance; the unconstrained LAE achieves the same reconstruction and classification metrics through an entirely different representation.

5.3 Practical Implications

For practitioners seeking interpretable dimensionality reduction that matches classical PCA, our results provide a clear prescription: use tied weights, omit bias terms, apply cosine learning rate scheduling, and train for sufficient epochs. For those who simply need effective compression for downstream tasks, the choice between PCA and an unconstrained autoencoder is largely one of convenience (both achieve equivalent performance).

5.4 Limitations

Several limitations constrain the scope of our conclusions:

- **Single dataset:** Our experiments focus exclusively on MNIST. Results may differ on datasets with different covariance structures, higher dimensionality, or more complex eigenvalue distributions.
- **Eigenvalue degeneracy:** Large maximum angles persist even with tight mean alignment, indicating rotational ambiguity in near-degenerate eigenspaces. We did not fully characterize how eigenvalue structure affects convergence.
- **Computational constraints:** Training time limitations prevented exhaustive hyperparameter search. Smaller batch sizes (64 vs. 128) showed promise but were not fully explored due to runtime considerations.

5.5 Future Directions

Several avenues merit further investigation:

1. Completing the constraint isolation experiment to determine which architectural constraint (tied weights or no bias) contributes more to subspace convergence.
2. Extending the analysis to additional datasets with varying eigenvalue structures to assess the generality of our findings.
3. Investigating whether orthogonal initialization provides more substantial benefits on datasets where the true principal components have greater separation.
4. Exploring nonlinear autoencoders to understand how nonlinearity changes the relationship between learned representations and linear baselines [3].

This study bridges the gap between theoretical guarantees and practical implementation for linear autoencoders. With appropriate constraints and optimization, LAEs recover the PCA subspace. At the same time, equally effective alternative subspaces found by unconstrained models highlight the richness of the representational landscape even for simple linear methods.

Code Availability

All code, trained models, experimental configurations, and interactive notebooks used in this study are publicly available at:

GitHub Repository: https://github.com/jwere100/PCA-and-Linear-Autoencoders-in-Machine-Learning

Google Colab Notebook: https://colab.research.google.com/drive/19xsgpLWJtEv_gnnU_wuiepCOBTqggAju

6 Bibliography

References

6.1 Peer-Reviewed Publications

[1] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989. doi:10.1016/0893-6080(89)90014-2.

[2] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint* arXiv:1404.1100, 2014. Available: https://arxiv.org/abs/1404.1100.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. doi:10.1126/science.1127647.

[4] E. Plaut, "From principal subspaces to principal components with linear autoencoders," Available: https://arxiv.org/abs/1804.10253.

[5] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[6] N. Salem and S. Hussein, "Data dimensional reduction and principal components analysis," *Procedia Computer Science*, vol. 163, pp. 292–299, 2019. doi:10.1016/j.procs.2019.12.111.

[7] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, 2016. doi:10.1098/rsta.2015.0202.

6.2 Online Tutorials and Resources

[8] "AutoEncoders: Theory + PyTorch Implementation," Medium. https://medium.com/@syed_hasan/autoencoders-theory-pytorch-implementation-a2e72f6f7cb7

[9] "Implementing PCA in Python with scikit-learn," DataCamp. https://www.datacamp.com/tutorial/principal-component-analysis-in-python

[10] V. Gandhi, "Dimensionality reduction: Principal component analysis," Analytics Vidhya (Medium), 2020. https://medium.com/analytics-vidhya/dimensionality-reduction-principal-component-analysis-d1402b58feb1

[11] V. Gandhi, "PCA beginner's guide to dimensionality reduction," Kaggle Notebooks. https://www.kaggle.com/code/vipulgandhi/pca-beginner-s-guide-to-dimensionality-reduction

[12] "What is dimensionality reduction?," IBM Think. https://www.ibm.com/think/topics/dimensionality-reduction

[13] Z. Zakaria, "A step-by-step explanation of principal component analysis," Built In, 2019. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[14] H. Idrees, "Autoencoders vs PCA: Dimensionality reduction for complex data," Medium, 2023. https://medium.com/@hassaanidrees7/autoencoders-vs-pca-dimensionality-reduction-for-complex-data-e07d4612b711

[15] J. Watt, "PCA," in *Machine Learning Refined* (GitHub/Colab Notebook). https://colab.research.google.com/github/jermwatt/machine_learning_refined/blob/main/notes/8_Linear_unsupervised_learning/8_3_PCA.ipynb

[16] "How autoencoders outperform PCA in dimensionality reduction," Towards Data Science. https://towardsdatascience.com/how-autoencoders-outperform-pca-in-dimensionality-reduction-1ae44c68b42f/