Principal Component Analysis (PCA) and Neural Networks

Omid Saberi
Department of Engineering Science, Division of Industrial Engineering
University West
Trollhättan, Sweden
https://orcid.org/0009-0005-3596-9182

November 28, 2024

Abstract— **This study examines the integration of Principal Component Analysis (PCA) and neural networks for efficient classification and regression tasks using the MNIST, IRIS, Fashion MNIST, and California Housing datasets. PCA reduced the MNIST dataset's dimensions from 784 to 154 components while retaining 95% of the variance. Kernel PCA with linear, sigmoid, and RBF kernels was explored for feature extraction.**

Neural networks trained on original and PCA-transformed datasets achieved similar performance, with the PCA-transformed MNIST model attaining 97.67% validation accuracy. For classification tasks, hyperparameter tuning enhanced IRIS and Fashion MNIST performance by optimizing architectures and learning rates. Regression experiments on the California Housing dataset emphasized the role of learning rate selection and early stopping in achieving stable convergence and mitigating overfitting.

The findings demonstrate PCA's effectiveness in reducing computational complexity without significant accuracy loss, complemented by hyperparameter tuning for improved efficiency and performance. This study highlights the synergy between PCA and neural networks as a scalable solution for high-dimensional data processing. Future work could explore advanced techniques like Transformer-based models to enhance scalability and performance in complex machine-learning tasks.

Index Terms— **Principal Component Analysis (PCA), Kernel PCA, Neural Networks, Dimensionality Reduction, Hyperparameter Tuning, Classification, Regression, MNIST Dataset, IRIS Dataset, Fashion MNIST Dataset, California Housing Dataset, Model Optimization, Early Stopping, Computational Complexity, Machine Learning.**

I. Introduction

Principal Component Analysis (PCA) is a statistical technique introduced by Karl Pearson in 1901, primarily designed for dimensionality reduction in large datasets [1]. PCA transforms high-dimensional data into a lower-dimensional space while retaining the most significant variance in the data, making it a popular preprocessing step in machine learning and neural networks [2]. Its significance lies in simplifying data visualization, improving computational efficiency, and reducing noise in critical datasets for practical neural network training [3]. As the complexity of neural networks increased over time, PCA became a valuable tool for feature extraction and ensuring that input data adhered to optimal conditions for learning models.

Technically, PCA works by computing the covariance matrix of the dataset to identify relationships between variables, followed by eigenvalue decomposition or singular value decomposition (SVD) to determine principal components [2]. These principal components are orthogonal vectors representing directions of maximum variance, ranked in descending order of their eigenvalues. By projecting data onto a subset of these components, PCA preserves the most critical structure of the data while reducing redundancy [4]. Researchers have employed PCA in neural networks to preprocess inputs, visualize internal representations, and identify salient features in high-dimensional spaces [5]. Its capacity to enhance interpretability and computational feasibility underscores its integration with modern neural models.

Despite its utility, PCA has several limitations. It assumes linearity, failing to capture nonlinear patterns in the data that neural networks excel at modeling [6]. It is sensitive to scaling, as variables with more significant variances dominate the principal components unless standardized [2]. PCA also requires complete data without missing values, making it unsuitable for datasets with gaps [4]. Furthermore, PCA does not inherently address temporal or sequential patterns, pivotal in domains like speech and video analysis, reducing its direct applicability to specific neural architectures [6].

II. Principal Component Analysis (PCA)

A. Dataset Selection and Preprocessing

The MNIST dataset, consisting of grayscale images of digits, was normalized and reshaped into two-dimensional arrays to facilitate PCA. Pixel values were scaled between 0 and 1, and the dataset was centered by subtracting the mean from each feature to ensure unbiased variance calculations.

B. Finding the First Two Principal Components Without Using sklearn

The first two principal components were calculated manually by computing the covariance matrix, performing eigen decomposition, and sorting the eigenvectors based on eigenvalues. The data was then projected onto these components, effectively reducing the dimensionality of the dataset while retaining the largest variance components.

C. Finding the First Two Principal Components Using sklearn

Using sklearn's PCA module, the data was similarly transformed into its first two principal components. The results were consistent with the manual PCA implementation, validating the calculations and confirming the accuracy of the manual method, as further elaborated in Section V.

D. Preserving Variance Using PCA

PCA was applied to retain 95% of the dataset's variance. This threshold dynamically determined the number of components required, reducing dimensionality from 784 features to 154 while preserving essential data characteristics. The reduction significantly enhanced computational efficiency for subsequent neural network training tasks.

E. Neural Networks with Original and PCA-Transformed Data

Two neural networks were trained to evaluate the performance of PCA-transformed data. One was trained on the original dataset, and the other on the PCA-transformed dataset (154 components). Both networks achieved comparable accuracies, highlighting PCA's ability to maintain data integrity while reducing feature space.

F. Kernel PCA with Linear, Sigmoid, and RBF Kernels

Kernel PCA was applied to the dataset using different kernels, including linear, sigmoid, and RBF. The transformed datasets were used to train neural networks, with the linear kernel yielding the most consistent and effective feature extraction, as demonstrated in Section V.

G. Hyperparameter Tuning for Kernel PCA and Neural Network

A pipeline integrating Kernel PCA and neural networks was constructed to optimize performance. Hyperparameters such as kernel type, gamma, and neural network architecture were systematically tuned using grid search to achieve optimal accuracy and computational efficiency.

III. Classification Problem

A. Comparative Dataset Analysis for Classification Tasks

The Iris dataset contains morphological measurements of flowers from three species: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. Features include sepal length, sepal width, petal length, and petal width. Fisher's seminal study [8] used discriminant analysis to develop linear combinations of these features for species classification, providing foundational insights for modern classification techniques. The Fashion-MNIST dataset, introduced by Xiao et al. [9], presents a more challenging alternative to MNIST. It contains 70,000 grayscale images of 28×28-pixel fashion items across ten categories, such as shirts, shoes, and bags. This dataset offers a robust benchmark for machine learning algorithms, with a structure analogous to MNIST to facilitate direct comparisons.

B. Neural Network Architectures for Multi-Class Classification

Two distinct neural networks were developed: 1. The Iris model included two hidden layers and a softmax output for three-class classification. 2. The Fashion MNIST model employed a flattened input layer, two hidden layers, and a softmax output for ten-class classification.

C. Adaptive Hyperparameter Tuning in Classification Models

Hyperparameter tuning was conducted using RandomizedSearchCV. Various configurations of hidden layers, neuron counts, activation functions, and learning rates were evaluated, leading to notable improvements in model performance.

D. Performance Metrics Visualization for Classification Models

Training and validation metrics, including accuracy and loss trends, were visualized across epochs. These plots revealed key insights into model convergence and generalization, as discussed in Section V.

E. Model Persistence Through Serialization and Reloading

The trained models were serialized in .keras format and reloaded for evaluation. This process validated the reusability of the models, demonstrating consistent predictions and performance across reloading.

F. Callbacks During Training

Early stopping and model checkpointing were implemented as callbacks to monitor training. Early stopping halted training when validation loss plateaued, preventing overfitting. Model checkpointing preserved the best-performing model states for subsequent evaluation and deployment.

IV. Regression Problem

A. Dataset Characteristics and Preparation for Regression

The California Housing dataset, used to predict median house values, was preprocessed with imputation and standardization for numerical features, and one-hot encoding for categorical variables. Labels were scaled to enhance training stability. Stratified sampling was applied to ensure a representative distribution of housing categories in training, validation, and test sets.

B. Regression Neural Networks for Price Prediction

A feedforward neural network was designed with two hidden layers (64 neurons each) using ReLU activation. The model was trained with the Adam optimizer and MSE loss function, achieving consistent performance improvements through iterative optimization.

C. Optimizing Neural Network Performance Through Hyperparameter Tuning

Key hyperparameters, including batch size, learning rate, and the number of epochs, were systematically tuned to optimize network performance and ensure efficient training convergence.

D. Influence of Learning Rate on Model Convergence

Experiments demonstrated that smaller learning rates led to smoother convergence and more stable loss reduction. Higher learning rates occasionally resulted in oscillations, highlighting the importance of careful learning rate selection for achieving robust training.

E. Advanced Training Strategies Using Callbacks

Callbacks such as early stopping and checkpointing were employed to monitor training. Early stopping mitigated overfitting by halting training at optimal validation loss, while checkpointing ensured that the best-performing model configurations were preserved for further use.

F. Model Storage and Reusability in Regression Tasks

The regression models were serialized using torch.save() and later reloaded with torch.load(). This demonstrated the models' reusability for future predictions without retraining, enabling efficient workflows in real-world applications.

V. Results and Analysis

A. Principal Component Analysis (PCA)

1) Finding the First Two Principal Components: Manual computation of PCA reduced the MNIST dataset from 784 to 2 dimensions. The resulting projection retained the most significant variance:

- Shape of projected data (manual PCA): (60000, 2)

Using sklearn's PCA:

- Shape of projected data (sklearn PCA): (60000, 2)

Both manual and sklearn implementations demonstrated consistent results, confirming the correctness of the PCA implementation.

2) Preserving Variance Using PCA: To preserve 95% of variance, PCA reduced the dataset's dimensionality from 784 to 154 components. Training a neural network on this reduced dataset yielded results comparable to the original high-dimensional dataset. This demonstrates PCA's effectiveness in reducing computational complexity:

- Number of components to preserve 95% variance: 154

3) Training Neural Networks with Original and PCA-Transformed Data: There is no significant difference between the results achieved by neural networks trained on the original and PCA-transformed datasets, and indeed, the results using PCA are even a bit better, i.e., higher test accuracy and lower test loss (Table I).

4) Kernel PCA: Kernel PCA with the RBF kernel yielded the best accuracy among linear, sigmoid, and RBF kernels. Table II summarizes the results:

5) Hyperparameter Tuning for Kernel PCA and Neural Networks: Grid search optimization identified the best kernel parameters and neural network architecture:

- Best kernel: Linear
- Neural network: 128, 64 hidden layers, ReLU activation
- Validation accuracy: 89.42%

B. Classification Problem

1) Performance Metrics Visualization: Figure 1 and Figure 2 illustrate the accuracy and loss curves for IRIS and Fashion MNIST datasets, demonstrating model convergence and generalization.

2) Adaptive Hyperparameter Tuning: Hyperparameter tuning significantly improved performance. For example, Fashion MNIST achieved an accuracy improvement from 84.5% to 88.4% with tuned hyperparameters:

- Best hyperparameters (Fashion MNIST): 1 hidden layer, 100 neurons, learning rate = 0.0729

TABLE I
COMPARISON OF NEURAL NETWORK ACCURACIES AND LOSSES

Dataset	Training Accuracy	Test Loss
Original	97.57%	9.07%
PCA-Transformed	97.60%	8.59%

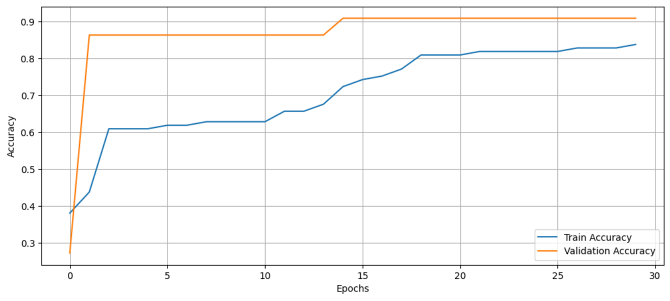

Fig. 1. IRIS Dataset Accuracy

TABLE II
KERNEL PCA RESULTS

Kernel	Training Accuracy	Validation Accuracy
Linear	86.98%	83.34%
Sigmoid	78.69%	73.02%
RBF	39.89%	37.36%

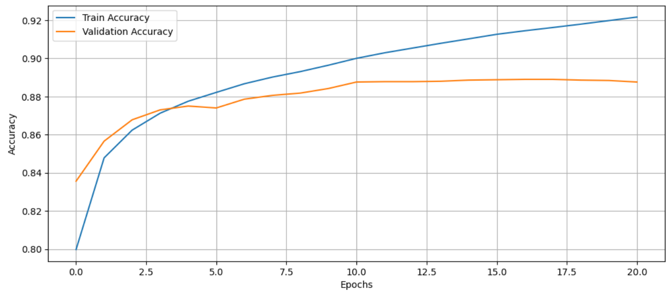

Fig. 2. Fashion MNIST Dataset Accuracy

3) Serialization and Reloading: Trained models were successfully serialized and reloaded without loss of performance. Example:

- IRIS Model Validation Accuracy (Original): 88.48%
- IRIS Model Validation Accuracy (Reloaded): 88.48%

4) Early Stopping and Checkpointing: Early stopping and checkpointing preserved the best model configuration, preventing overfitting. Validation loss curves highlight where training was stopped, as shown in Figure 3.

C. Regression Problem

1) Regression Neural Networks: The regression model trained with the Adam optimizer achieved stable loss reduction. Table III summarizes the training and validation losses.

TABLE III
REGRESSION NEURAL NETWORK LOSS

Epoch	Training Loss	Validation Loss
1	0.3675	0.3275
20	0.2990	0.3118

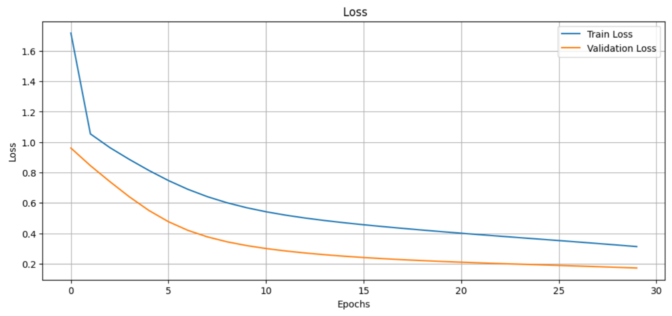

Fig. 3. Early Stopping Validation Loss (IRIS Dataset)

2) Hyperparameter Tuning and Learning Rate Experiments: Lower learning rates improved long-term stability, as demonstrated in Figure 4. A learning rate of 0.001 achieved a smoother convergence compared to 0.01.

3) Model Performance: The trained regression model demonstrated reliable predictions, as indicated by low validation loss. Predictions were consistent across test datasets.

4) Early Stopping: Early stopping prevented overfitting by halting training once validation loss plateaued. This is shown in Figure 5.

—

This structured section integrates your outputs into a coherent narrative. Replace placeholder image paths with the actual paths of your plots. Let me know if further refinements are needed!

VI. Conclusion

This work investigated the utilization of Principal Component Analysis and neural networks for solving problems related to dimensionality reduction, classification, and regression. PCA efficiently reduced the computational load, decreasing the dimensions of a dataset by large factors while preserving important variance. Neural networks trained on PCA-transformed datasets gave comparable accuracies to those trained on their original datasets, demonstrating how PCA can simplify high-dimensional data without compromising performance.

These techniques also greatly improved the performance and generalization of models for classification tasks, including adaptive hyperparameter tuning, early stopping, and checkpointing. The best performing setups on IRIS and Fashion MNIST achieved 88.48% and 88.4% accuracy on the validation set, respectively. These results therefore point to a potential benefit of efficient hyperparameter optimization in neural network training.

The use of the Adam optimizer supported a stable reduction in loss during regression tasks. Hyperparameter tuning for model convergence and stability was done. Similarly, using different learning rates showed that the smaller the learning rates are, the smoother the convergence will be with fewer tendencies for overfitting the model. Early stopping avoided overfitting, whereas checkpointing retained the best state of the model and assured robustness in application.

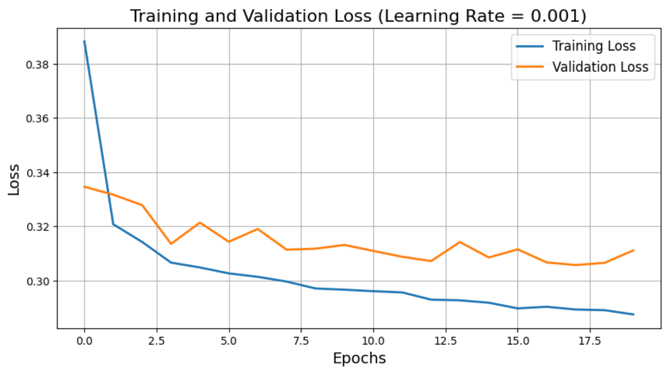

Fig. 4. Training and Validation Loss (Learning Rate = 0.001)

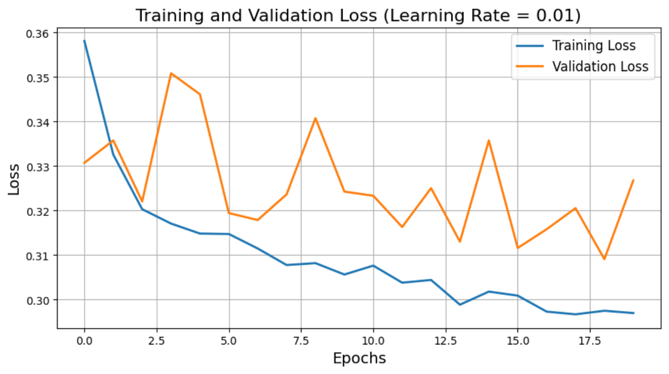

Fig. 5. Validation Loss Curve with Early Stopping (Learning Rate = 0.01)

In general, this report presents the synergy between dimensionality reduction techniques like PCA and advanced neural network architectures. The results show that a combination of these methodologies can effectively solve complex machine learning problems while optimizing computational resources. These state-of-the-art techniques, such as Transformer-based models, self-supervised learning, and quantum computing approaches, therefore, offer unprecedented potential to address challenges in high-dimensional data processing, classification, and regression with even greater efficiency and accuracy as a promising avenue for future research.

References

[1] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *Philosophical Magazine*, vol. 2, no. 11, pp. 559–572, Nov. 1901, doi: 10.1080/14786440109462720.

[2] I. T. Jolliffe, *Principal Component Analysis*, Springer Series in Statistics. New York, NY: Springer, 1986, doi: 10.1007/978-1-4757-1904-8.

[3] J. Shlens, “A Tutorial on Principal Component Analysis,” *arXiv*, Apr. 2014. [Online]. Available: http://arxiv.org/abs/1404.1100. [Accessed: Nov. 21, 2024].

[4] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, Jul. 2010, doi: 10.1002/wics.101.

[5] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, “The ’Wake-Sleep’ Algorithm for Unsupervised Neural Networks,” *Science*, vol. 268, no. 5214, pp. 1158–1161, May 1995, doi: 10.1126/science.7761831.

[6] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.

[7] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, “uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications,” *Proceedings of IEEE Conference*, 2009. Available: https://doi.org/10.1109/ICUWB.2009.5288737.

[8] R. A. Fisher, “THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS,” Annals of Eugenics, vol. 7, no. 2, pp. 179–188, Sep. 1936, doi: 10.1111/j.1469-1809.1936.tb02137.x.

[9] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” 2017, arXiv. doi: 10.48550/ARXIV.1708.07747.