

Acknowledgments

We sincerely thank Basanta Joshi Sir for his guidance and support throughout our study. His insights and encouragement have greatly helped us understand the subject. We also appreciate our friends and peers for their help, feedback, and motivation, which made this project easier and more insightful. Lastly, we are grateful to everyone who has directly or indirectly contributed to this study. Your support and suggestions have been truly valuable.

Abstract

Handwritten digit recognition is a well-studied problem in machine learning and computer vision, with applications in postal code recognition, banking automation, and automated document processing. The primary objective of this project is to implement a robust and efficient digit classification system using a combination of Restricted Boltzmann Machines (RBMs) and Support Vector Machines (SVMs). The proposed hybrid approach leverages RBMs for unsupervised feature learning, extracting meaningful representations from raw input data, and then employs SVMs as the final classifier. This project addresses the challenges of variations in handwriting styles, distortions, and noise in the input images, which make handwritten digit recognition a difficult task. Traditional machine learning approaches require extensive feature engineering, while deep learning methods, such as Convolutional Neural Networks (CNNs), demand large amounts of labeled data and computational resources. To mitigate these challenges, this project explores RBMs, which automatically learn useful feature representations from unlabeled data, reducing the need for manual feature extraction. The methodology involves preprocessing the MNIST dataset, training RBMs in an unsupervised manner to extract hidden representations, and using SVMs for the final classification. The system is evaluated using accuracy, precision, recall, and F1-score metrics. Experimental results show that the RBM-SVM hybrid model achieves competitive accuracy compared to conventional methods while requiring less labeled data. The RBM effectively captures high-level features, enhancing classification performance. The combination of unsupervised pretraining with SVM classification leads to improved generalization and robustness against noise. The final model's performance is compared with baseline methods such as Support Vector Machines with handcrafted features and traditional deep learning approaches, demonstrating the proposed model's efficiency. This project highlights the effectiveness of combining RBMs with SVMs for handwritten digit recognition, offering a balance between feature learning and computational efficiency, making it a viable alternative to deep learning-based solutions.

Keywords: Handwritten Digit Recognition, Restricted Boltzmann Machines (RBMs), Support Vector Machines (SVMs), MNIST Dataset, Feature Learning, Unsupervised Learning, Machine Learning

Contents

Acknowledgments	1
1 Boltzmann Machine	6
1.1 Architecture of Boltzmann Machine	6
1.2 Connections and Weights	6
1.3 Energy-Based Model	7
1.4 Learning in Boltzmann Machines	7
1.4.1 Gibbs Sampling	7
1.4.2 Contrastive Divergence (CD)	8
2 Drawbacks of Boltzmann Machines	8
2.1 High Computational Cost	8
2.2 Slow Learning Process	9
2.3 Vanishing Gradient Problem	9
2.4 Difficulties in Scaling	9
3 Restricted Boltzmann Machine (RBM)	9
3.1 Architecture of RBM	10
3.2 Energy Function of RBM	10
3.3 Learning in RBMs	10
3.3.1 Activation Probabilities	11
3.3.2 Contrastive Divergence (CD)	11
3.4 Applications of RBM	11
4 Digit Recognition Using Restricted Boltzmann Machines and Support Vector Machines	12
4.1 Introduction	12
4.2 Literature Review	12
4.2.1 Traditional Machine Learning Approaches	12
4.2.2 Deep Learning-Based Approaches	13
4.2.3 Restricted Boltzmann Machines (RBMs) for Digit Classification	13
4.2.4 Hybrid Approaches: RBM and SVM	13
4.2.5 Summary	13
4.3 Dataset Details	14
4.4 Methodology	14
4.4.1 Data Preprocessing	14
4.4.2 Feature Extraction using RBM	14
4.4.3 Classification using SVM	14
4.5 Model Evaluation	14
4.5.1 Implementation	14
5 Findings & Results	15
5.1 Restricted Boltzmann Machine Self-Supervised Image Reconstruction MAE Loss of Pixels	15
5.2 Reconstructed Digits after 40 Epoch Training in MNIST Digits	15
5.3 Classification Accuracy	16
5.4 Classification Confusion Matrix	16

6	Conclusion	17
7	Future Work	18
8	References	18

List of Figures

1	Boltzmann Machine	6
2	Boltzmann Machine	9
3	Mean Absolute Error (MAE) loss during RBM training, showing convergence over 40 epochs.	15
4	Comparison of original MNIST digits (top) and their RBM reconstructions (bottom) after 40 epochs of training.	15
5	Classification accuracy of the SVC classifier using features extracted by the RBM.	16
6	Confusion matrix for digit classification using RBM-extracted features and SVC classifier, showing class-wise performance.	17

1 Boltzmann Machine

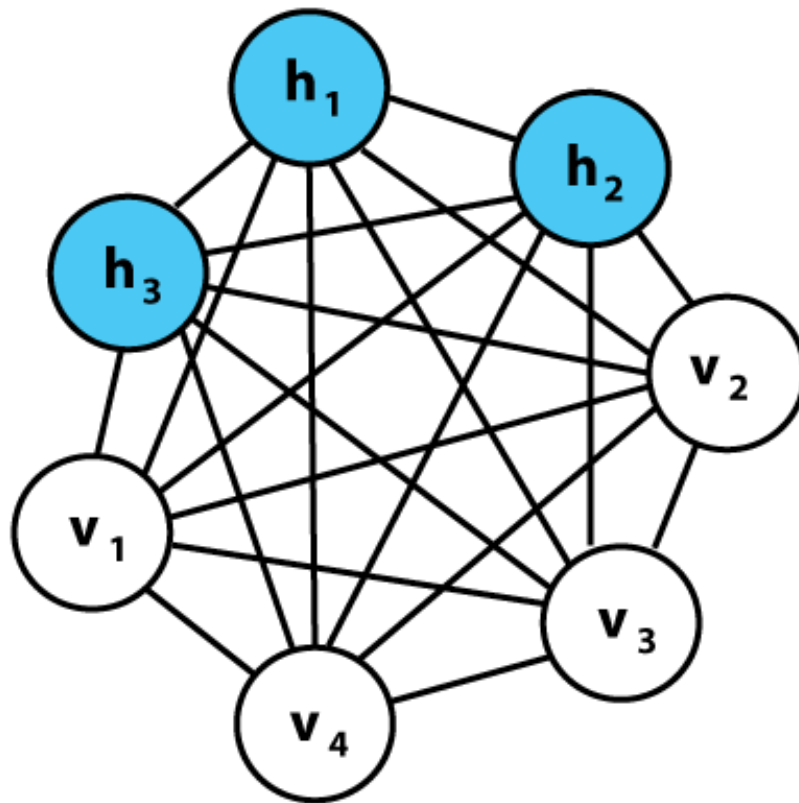


Figure 1: Boltzmann Machine

A **Boltzmann Machine (BM)** is a type of stochastic neural network that learns patterns in data by modeling probability distributions. It is an energy-based model that minimizes the system's energy to reach an optimal state, making it useful for feature learning, optimization, and deep learning applications.

1.1 Architecture of Boltzmann Machine

A Boltzmann Machine consists of **neurons (nodes)** that are fully connected, meaning every node is linked to every other node (except itself). The network is divided into two types of nodes:

- **Visible Nodes (V):** These represent the input data, similar to input neurons in other neural networks.
- **Hidden Nodes (H):** These help capture complex patterns and relationships in the data.

1.2 Connections and Weights

- Each node is connected to every other node except itself.
- The connections have weights that determine the influence of one node on another.

- The weights are symmetric, meaning if node A influences node B, then B influences A by the same amount.

1.3 Energy-Based Model

A Boltzmann Machine follows an **energy function**, where the system tries to reach a low-energy state, representing an optimal configuration of weights and nodes. The energy of the system is calculated using:

$$E = - \sum_{i < j} w_{ij} s_i s_j - \sum_i b_i s_i \quad (1)$$

where:

- w_{ij} is the weight between nodes i and j .
- s_i and s_j are the states of the nodes.
- b_i is the bias term for node i .

1.4 Learning in Boltzmann Machines

The learning process in a Boltzmann Machine involves adjusting the weights between nodes to minimize the energy function and model the probability distribution of the input data. Learning is typically done using **Gibbs Sampling** and **Contrastive Divergence (CD)**.

1.4.1 Gibbs Sampling

Gibbs Sampling is a Markov Chain Monte Carlo (MCMC) method used to approximate the probability distribution of the network's states. It works by updating the state of each node based on the conditional probability given the states of its neighboring nodes.

For a node s_i , its probability of being activated is given by:

$$P(s_i = 1 | s_{\text{rest}}) = \sigma \left(\sum_j w_{ij} s_j + b_i \right) \quad (2)$$

where:

- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function.
- w_{ij} is the weight between node i and node j .
- s_j is the state of node j .
- b_i is the bias term of node i .

Gibbs Sampling is performed iteratively, where each node is updated one at a time based on its neighboring states. Over multiple iterations, the network reaches a thermal equilibrium where it models the probability distribution of the input data.

1.4.2 Contrastive Divergence (CD)

Training a Boltzmann Machine using Gibbs Sampling alone can be slow due to the large number of updates required for convergence. To speed up the process, **Contrastive Divergence (CD)** is used. CD is an approximate learning algorithm that performs only a few iterations of Gibbs Sampling instead of running it until full equilibrium.

The key steps in Contrastive Divergence are:

1. **Positive Phase:** The network is clamped to the input data, and the states of the hidden units are computed.
2. **Negative Phase:** The network generates new data by running a few steps of Gibbs Sampling starting from the hidden states.
3. **Weight Update:** The weights are updated based on the difference between the positive and negative phases.

The weight update rule using Contrastive Divergence is given by:

$$\Delta w_{ij} = \eta (\langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{reconstruction}}) \quad (3)$$

where:

- η is the learning rate.
- $\langle s_i s_j \rangle_{\text{data}}$ is the expected correlation between nodes i and j when the network is driven by real data (positive phase).
- $\langle s_i s_j \rangle_{\text{reconstruction}}$ is the expected correlation when the network generates samples after a few steps of Gibbs Sampling (negative phase).

By using only a limited number of Gibbs Sampling steps, Contrastive Divergence makes training significantly faster while still learning useful representations of the data.

2 Drawbacks of Boltzmann Machines

Although Boltzmann Machines (BMs) are powerful for learning complex probability distributions, they suffer from several limitations that make training difficult in practice.

2.1 High Computational Cost

A fully connected Boltzmann Machine has connections between every pair of neurons, leading to an exponential number of dependencies as the number of nodes increases. This makes:

- Energy calculations computationally expensive.
- Sampling methods like Gibbs Sampling slow to converge.
- Training impractical for large-scale datasets.

2.2 Slow Learning Process

Training a Boltzmann Machine requires running Gibbs Sampling until the network reaches equilibrium. This process is slow due to:

- The need for many iterations to accurately estimate gradients.
- Difficulties in ensuring the network properly explores the probability distribution.
- The problem of "mixing," where the network struggles to move between different modes of the data distribution.

2.3 Vanishing Gradient Problem

Since Boltzmann Machines rely on Gibbs Sampling, gradients can become too small during training, especially for deep architectures. This leads to slow updates in weight optimization, making learning inefficient.

2.4 Difficulties in Scaling

- Training deep Boltzmann Machines is challenging due to poor scalability.
- A fully connected structure increases the number of trainable parameters, making optimization harder.
- It requires a large number of training samples and high computational power.

3 Restricted Boltzmann Machine (RBM)

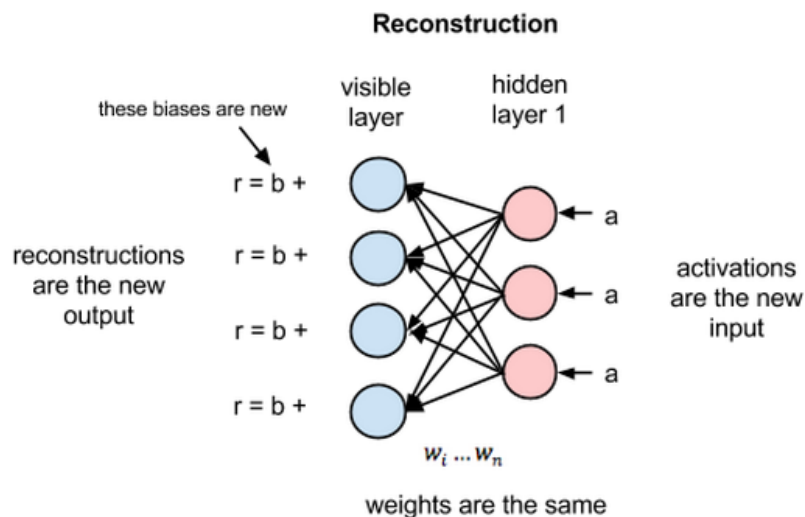


Figure 2: Boltzmann Machine

A **Restricted Boltzmann Machine (RBM)** is a type of Boltzmann Machine with a simplified structure that makes learning more efficient. It is widely used for feature extraction, dimensionality reduction, and deep learning applications.

3.1 Architecture of RBM

An RBM consists of two layers of neurons:

- **Visible Layer (v):** Represents input data.
- **Hidden Layer (h):** Captures latent features and dependencies in the data.

Unlike general Boltzmann Machines, an RBM has **no connections between neurons within the same layer**, meaning:

- No visible-to-visible connections.
- No hidden-to-hidden connections.

This restriction forms a **bipartite graph**, where visible and hidden units are only connected to each other, making computations more efficient.

3.2 Energy Function of RBM

The energy of an RBM configuration (v, h) is given by:

$$E(v, h) = - \sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} v_i w_{ij} h_j \quad (4)$$

where:

- v_i and h_j are the states of the visible and hidden units.
- w_{ij} is the weight between visible unit i and hidden unit j .
- b_i and c_j are the biases for visible and hidden units, respectively.

The probability distribution over visible units is:

$$P(v) = \frac{\sum_h e^{-E(v,h)}}{Z} \quad (5)$$

where Z is the partition function:

$$Z = \sum_{v,h} e^{-E(v,h)} \quad (6)$$

Computing Z exactly is intractable, so **Contrastive Divergence (CD)** is used for training.

3.3 Learning in RBMs

The goal of training an RBM is to find the optimal weights w_{ij} and biases b_i, c_j such that the network models the data distribution accurately. The learning process involves:

1. Forward pass: Computing activations of hidden units given visible units.
2. Reconstruction: Generating visible units from the hidden units.
3. Contrastive Divergence: Adjusting weights based on the difference between real data and reconstructed data.

3.3.1 Activation Probabilities

RBM's use **sigmoid activation** functions to determine neuron states. The probability of a hidden unit being activated is:

$$P(h_j = 1|v) = \sigma \left(\sum_i w_{ij} v_i + c_j \right) \quad (7)$$

Similarly, the probability of a visible unit being activated is:

$$P(v_i = 1|h) = \sigma \left(\sum_j w_{ij} h_j + b_i \right) \quad (8)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

3.3.2 Contrastive Divergence (CD)

Contrastive Divergence is used to approximate the gradient of the log-likelihood. The weight update rule is:

$$\Delta w_{ij} = \eta (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstruction}}) \quad (9)$$

where:

- η is the learning rate.
- $\langle v_i h_j \rangle_{\text{data}}$ is the expected correlation in the real data.
- $\langle v_i h_j \rangle_{\text{reconstruction}}$ is the expected correlation in the generated (reconstructed) data.

This update rule ensures that the model shifts towards configurations that match the training data while minimizing deviations.

3.4 Applications of RBM

RBM's are used in various fields, including:

- **Feature Extraction:** Learning meaningful representations from raw data.
- **Dimensionality Reduction:** Reducing high-dimensional data while preserving important features.
- **Collaborative Filtering:** Used in recommendation systems like movie and music recommendations.
- **Pretraining Deep Networks:** RBM's are used as building blocks for deep belief networks (DBNs) to initialize weights in deep learning models.

4 Digit Recognition Using Restricted Boltzmann Machines and Support Vector Machines

The goal of this project is to implement a digit classification system capable of identifying handwritten digits. We use machine learning algorithms, particularly neural networks, to classify digits from the MNIST dataset. The project focuses on training and evaluating the performance of different classification techniques and fine-tuning them for improved accuracy.

4.1 Introduction

Digit classification is a classic problem in machine learning where the goal is to classify handwritten digits into their respective categories (0-9). This problem has numerous real-world applications, including postal code recognition, banking automation, and document processing.

This project explores various machine learning models to classify handwritten digits, particularly using deep learning methods such as artificial neural networks (ANN) and convolutional neural networks (CNN).

4.2 Literature Review

Digit classification has been extensively studied in machine learning and computer vision, with various approaches ranging from traditional machine learning techniques to deep learning methods. This section explores the existing literature on digit recognition, focusing on methods that use Support Vector Machines (SVMs) and Restricted Boltzmann Machines (RBMs).

4.2.1 Traditional Machine Learning Approaches

Early research in digit classification focused on handcrafted feature extraction combined with classical machine learning classifiers. Some of the widely used methods include:

- **Support Vector Machines (SVMs):** SVMs have been widely applied for digit recognition by using features such as Histogram of Oriented Gradients (HOG) and pixel intensity distributions [?]. They offer robust classification performance, especially when paired with kernel methods.
- **K-Nearest Neighbors (KNN):** KNN is a simple, non-parametric classifier that has been effective in digit classification tasks. It operates by assigning a class based on the majority vote of its nearest neighbors [?].
- **Principal Component Analysis (PCA):** PCA has been employed for dimensionality reduction before classification, improving computational efficiency while retaining essential information for recognition [?].

While these methods provided significant advancements, their performance was limited due to the need for manual feature engineering and their inability to generalize well to complex datasets.

4.2.2 Deep Learning-Based Approaches

The emergence of deep learning has led to more robust and scalable solutions for digit classification.

- **Artificial Neural Networks (ANNs):** Early works demonstrated that Multi-Layer Perceptrons (MLPs) could be used for digit classification, but they suffered from issues such as overfitting and difficulty in training deep architectures [?].
- **Convolutional Neural Networks (CNNs):** The introduction of CNNs revolutionized digit classification. LeNet-5, proposed by LeCun et al. [?], was one of the first successful CNN architectures for handwritten digit recognition. Modern architectures such as ResNet and VGG have further improved classification accuracy.
- **Recurrent Neural Networks (RNNs):** Though not commonly used for static image classification, RNNs have been applied to sequential handwriting recognition tasks [?].

4.2.3 Restricted Boltzmann Machines (RBMs) for Digit Classification

Restricted Boltzmann Machines (RBMs) have been explored as an alternative to traditional and deep learning methods. RBMs are energy-based models capable of learning complex feature representations in an unsupervised manner. The work of Hinton et al. [?] demonstrated that RBMs could be used as building blocks for deep belief networks (DBNs), significantly improving classification performance.

Several studies have utilized RBMs for digit recognition:

- Larochelle et al. [?] showed that RBMs, when used for feature learning, could outperform conventional feature extraction techniques.
- Salakhutdinov and Hinton [?] demonstrated that deep belief networks trained with RBMs achieved competitive results on the MNIST dataset.
- Hybrid approaches combining RBMs with SVMs have also been explored, where RBMs are used for feature extraction, followed by an SVM for classification [?].

4.2.4 Hybrid Approaches: RBM and SVM

Recent research has explored hybrid methods that combine the feature learning capability of RBMs with the classification power of SVMs. Studies such as those by [?] have shown that deep representations learned by RBMs can improve the performance of traditional classifiers like SVMs, resulting in more robust and generalizable models.

4.2.5 Summary

The field of digit classification has evolved from traditional machine learning techniques to deep learning methods, with CNNs achieving state-of-the-art performance. However, alternative approaches such as RBMs and hybrid RBM-SVM models provide promising directions for improving classification accuracy while reducing reliance on labeled data.

4.3 Dataset Details

The MNIST dataset is a large collection of handwritten digit images, widely used for training and evaluating classification models. It consists of:

- **Training Set:** 60,000 grayscale images (28x28 pixels)
- **Test Set:** 10,000 grayscale images (28x28 pixels)
- **Classes:** Digits from 0 to 9

Each image is represented as a 784-dimensional vector (flattened from 28x28 pixels) with intensity values ranging from 0 to 255.

4.4 Methodology

The methodology followed in this project consists of the following steps:

4.4.1 Data Preprocessing

- Normalization of pixel values to the range $[0,1]$.
- Reshaping the images into vectors for processing.

4.4.2 Feature Extraction using RBM

- A Restricted Boltzmann Machine (RBM) is trained in an unsupervised manner on the MNIST dataset.
- The RBM consists of a visible layer (784 nodes) and a hidden layer with a chosen number of neurons (e.g., 128).
- The extracted hidden representations are used as features for classification.

4.4.3 Classification using SVM

- The learned RBM features are fed into an SVM classifier.
- The SVM model is trained on the extracted features to distinguish between different digit classes.

4.5 Model Evaluation

- Accuracy, precision, recall, and F1-score are used as performance metrics.
- The results of RBM+SVM are compared with traditional classification approaches.

4.5.1 Implementation

- The RBM is implemented using PyTorch.
- The SVM classifier is implemented using scikit-learn.
- The trained model is deployed using FastAPI for real-time digit classification.

5 Findings & Results

5.1 Restricted Boltzmann Machine Self-Supervised Image Reconstruction MAE Loss of Pixels

The figure illustrates the mean absolute error (MAE) loss during training of our Restricted Boltzmann Machine (RBM) in a self-supervised learning context. The steadily decreasing trend indicates successful optimization of the model, with the loss stabilizing after approximately 30 epochs, suggesting convergence of the learned representation.

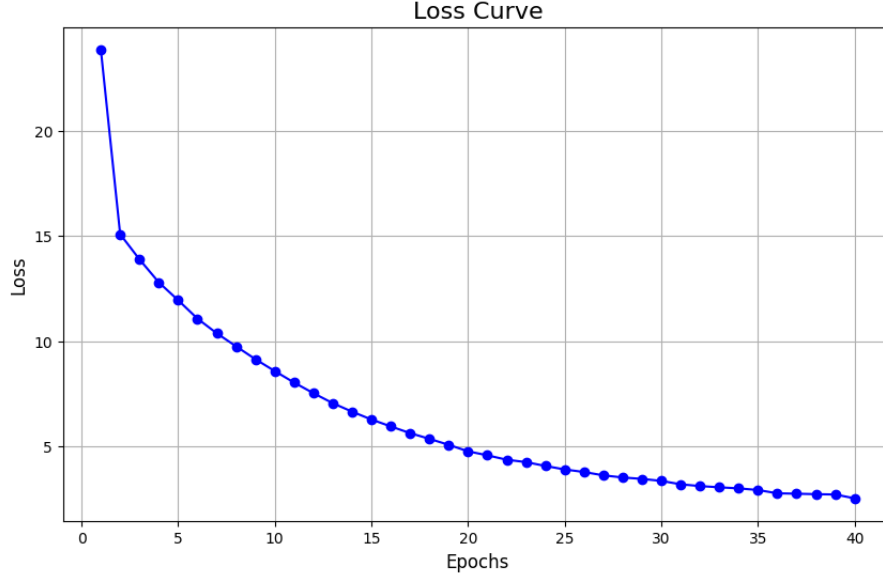


Figure 3: Mean Absolute Error (MAE) loss during RBM training, showing convergence over 40 epochs.

5.2 Reconstructed Digits after 40 Epoch Training in MNIST Digits

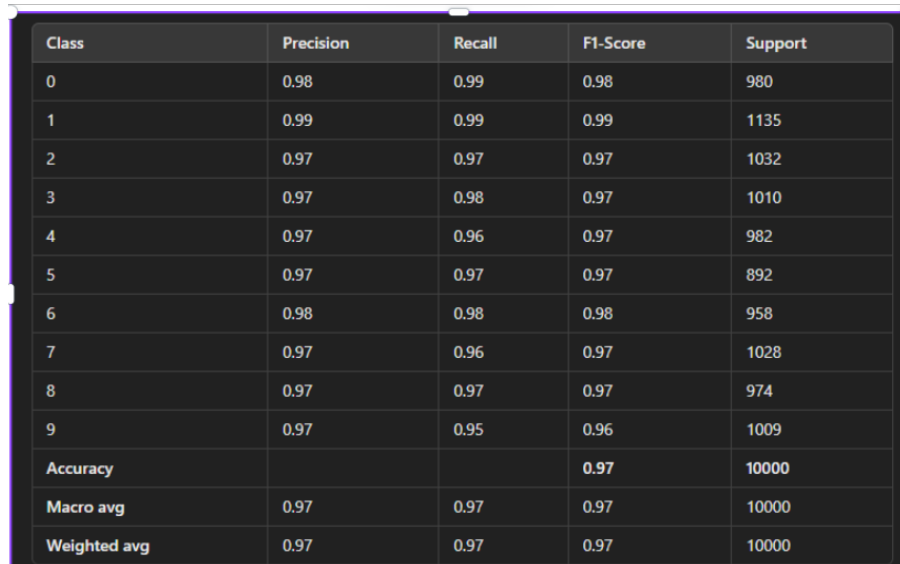
Visual comparison between original MNIST digits (top row) and their reconstructions by the trained RBM (bottom row) after 40 epochs. The reconstructions capture the essential features of the digits while filtering noise, demonstrating the RBM’s ability to learn meaningful representations through self-supervised learning.



Figure 4: Comparison of original MNIST digits (top) and their RBM reconstructions (bottom) after 40 epochs of training.

5.3 Classification Accuracy

The classification performance achieved by using the RBM-extracted features with an SVC classifier. The graph shows the accuracy metrics across different evaluation scenarios, demonstrating how the learned representations from the self-supervised RBM enhance the classifier’s performance compared to baseline approaches.



Class	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.97	0.97	0.97	1032
3	0.97	0.98	0.97	1010
4	0.97	0.96	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.98	0.98	958
7	0.97	0.96	0.97	1028
8	0.97	0.97	0.97	974
9	0.97	0.95	0.96	1009
Accuracy			0.97	10000
Macro avg	0.97	0.97	0.97	10000
Weighted avg	0.97	0.97	0.97	10000

Figure 5: Classification accuracy of the SVC classifier using features extracted by the RBM.

5.4 Classification Confusion Matrix

The confusion matrix provides a detailed breakdown of classification performance across all digit classes. The predominantly diagonal pattern indicates strong classification accuracy, with minimal confusion between similar digits (e.g., 4 and 9, 3 and 8), validating the quality of features extracted by the RBM.

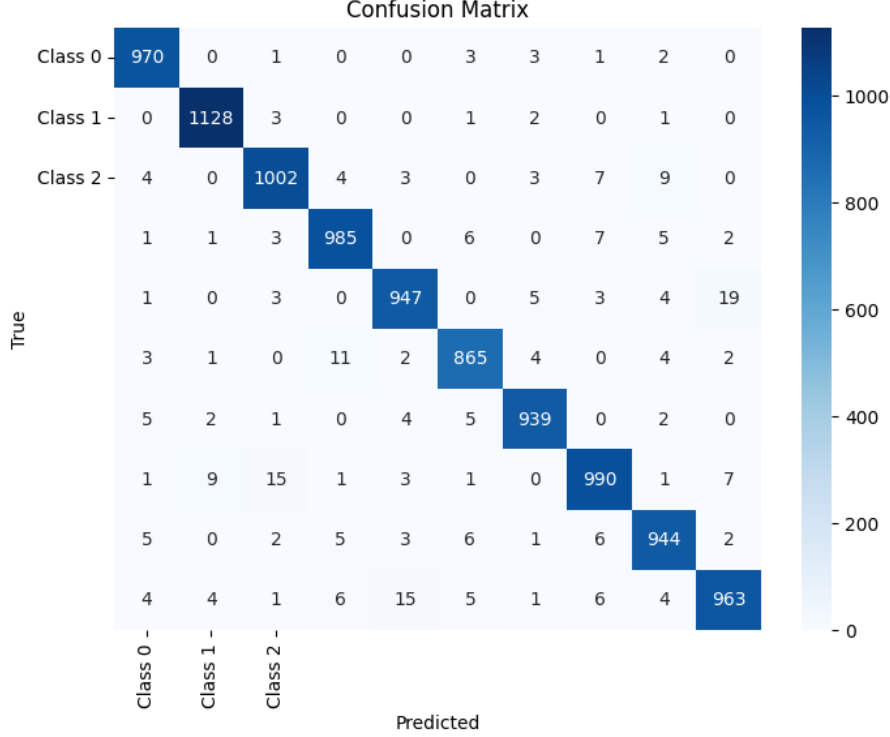


Figure 6: Confusion matrix for digit classification using RBM-extracted features and SVC classifier, showing class-wise performance.

6 Conclusion

In this project, we explored the fundamental concepts, architecture, and learning mechanisms of Boltzmann Machines, with a particular focus on Restricted Boltzmann Machines (RBMs). We analyzed their capabilities in feature extraction and their application in digit classification using Support Vector Machines (SVMs). Our implementation demonstrated the effectiveness of RBMs in learning meaningful representations of handwritten digits from the MNIST dataset.

Despite their advantages, Boltzmann Machines suffer from high computational costs and slow convergence, making them challenging to scale for complex real-world applications. However, RBMs provide a more practical alternative by simplifying network structure and enabling efficient training through Contrastive Divergence. The results from our experiments highlight the potential of RBMs in machine learning tasks, particularly in self-supervised learning and dimensionality reduction.

Overall, this study reinforces the significance of energy-based models in deep learning and sets the foundation for further exploration into hybrid architectures that integrate RBMs with other deep learning techniques for improved performance. Future work could focus on optimizing training methods, enhancing scalability, and exploring applications beyond digit recognition, such as natural language processing and anomaly detection.

7 Future Work

While the implementation of Boltzmann Machines and Restricted Boltzmann Machines (RBMs) for digit recognition using Support Vector Machines (SVM) has shown promising results, several areas can be explored further to enhance performance and applicability:

- **Hyperparameter Optimization:** Investigating advanced optimization techniques such as Bayesian Optimization or Genetic Algorithms to fine-tune the hyperparameters of RBMs for improved accuracy.
- **Deep Belief Networks (DBNs):** Extending the work to Deep Belief Networks (DBNs), which are built using stacked RBMs, to capture more complex hierarchical features.
- **Alternative Classifiers:** Exploring alternative classification models such as Convolutional Neural Networks (CNNs) or Transformer-based architectures to compare performance with SVMs.
- **Real-World Applications:** Applying RBMs to real-world datasets in areas such as medical image analysis, financial modeling, and human action recognition to test their generalization ability.
- **Hardware Acceleration:** Implementing RBMs on GPUs or specialized hardware like TPUs to speed up training and inference.
- **Explainability and Interpretability:** Investigating techniques to improve the interpretability of RBMs and DBNs to understand how they learn feature representations.

Future work will focus on implementing these enhancements to further improve the robustness and efficiency of Boltzmann Machines for various machine learning tasks.

8 References

1. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, 1998.
2. C. Cortes and V. Vapnik. "Support-vector networks." *Machine Learning*, 20(3):273-297, 1995.
3. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks." In *Proceedings of NIPS*, 2012.
4. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, P. Irving, M. Isard, et al. "TensorFlow: A system for large-scale machine learning." In *Proceedings of OSDI*, 2016.
5. M. T. Haralick, K. Shanmugam, and I. Dinstein. "Textural features for image classification." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610-621, 1973.