Implementation and Analysis of Pre-trained vs. Randomly Initialized Deep Neural Networks

Idriss Lakas, Anasse Essalih

March 23, 2025

Abstract

This report presents an implementation and comparative analysis of Deep Neural Networks (DNNs) for handwritten digit recognition using the MNIST dataset. We compare the performance of networks that are pre-trained using Deep Belief Networks (DBNs) versus networks that are randomly initialized. Our analysis explores how performance varies with network depth, neuron count per layer, and training dataset size. Results demonstrate the advantage of unsupervised pre-training, particularly when dealing with deeper architectures or limited training data.

1 Introduction

Over the past few years, deep neural networks (DNNs) have enjoyed great success in a wide range of machine learning applications such as computer vision and natural language processing. However, their effectiveness heavily depends on proper initialization and training strategies, especially as network depth increases or when labeled data is limited. One of the classic approaches to solving these problems is unsupervised pre-learning, which initializes network parameters in a favorable region of the parameter space prior to supervised learning.

This project implements and compares two approaches to network initialization:

- Networks with random weight initialization
- Networks pre-trained with Deep Belief Networks (DBNs)

The goal is to determine under what conditions pre-training offers an improvement in terms of classification accuracy.

2 Methodology

2.1 Models Implemented

2.1.1 Restricted Boltzmann Machine (RBM)

We implemented RBMs as the basic building block for unsupervised learning. The RBM consists of a visible layer and a hidden layer, with no connections within each layer. The energy function of an RBM is defined as:

$$E(v, h) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} v_i W_{ij} h_j \tag{1}$$

where v represents visible units, h represents hidden units, b and c are bias terms, and W is the weight matrix.

RBMs are trained using Contrastive Divergence (CD-k), which approximates the gradient of the log-likelihood. The update rule for the weights is:

$$\Delta W_{ij} = \epsilon(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}) \tag{2}$$

where ϵ is the learning rate, $\langle \cdot \rangle_{\text{data}}$ represents expectations under the data distribution, and $\langle \cdot \rangle_{\text{model}}$ represents expectations under the model distribution.

2.1.2 Deep Belief Network (DBN)

DBNs are constructed by stacking multiple RBMs. Each RBM is trained separately in a greedy layer-wise manner. The output of one RBM becomes the input to the next. This allows for effective pre-training of deep networks.

2.1.3 Deep Neural Network (DNN)

For supervised learning, we implemented feed-forward neural networks with multiple hidden layers. When pre-trained, the initial weights of the DNN are taken from the trained DBN with an additional classification layer added for the final output. The network is then fine-tuned using backpropagation with categorical cross-entropy loss.

2.2 Datasets

Two datasets were used in this project:

- **Binary AlphaDigits**: A dataset of binary images of digits and letters. Used primarily to test the RBM and DBN implementations for unsupervised learning and generation tasks.
- **MNIST**: A standard benchmark dataset of handwritten digits (0-9). Contains 60,000 training images and 10,000 test images, each of size 28×28 pixels. This dataset was used for the main comparative analysis.

2.3 Experimental Setup

We conducted three main experiments to compare pre-trained versus randomly initialized networks:

1. **Varying Network Depth**: We tested networks with 1, 2, 3, and 4 hidden layers, maintaining a consistent 200 neurons per layer.
2. **Varying Neuron Count**: We tested 2-layer networks with varying numbers of neurons per layer (100, 200, 300, 500, and 700).
3. **Varying Training Data Size**: We tested 2-layer networks with 200 neurons per layer, trained on different subsets of the MNIST training data (1000, 3000, and 7000 samples).

For each experiment, we trained and evaluated two networks with identical architectures:

- A network pre-trained with DBN followed by backpropagation fine-tuning
- A network initialized with random weights and trained solely with backpropagation

All networks were trained with the Adam optimizer with a learning rate of 0.001, using categorical cross-entropy loss. Pre-training was performed using CD-1 for 100 epochs, and supervised training was conducted for 200 epochs, both with a batch size of 100.

2.4 Results of Binary AlphaDigit Image Generation

In this section, we present the results of our study on the Binary AlphaDigit dataset using Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN). We analyze the ability of these models to learn and generate images resembling the original characters in the dataset.

2.4.1 Dataset Analysis

The Binary AlphaDigit dataset consists of binary images representing digits (0-9) and letters (A-Z). In our experiments, we focused on the subset containing digits 3, 4, and 5. Figure 1 shows sample images from this dataset.

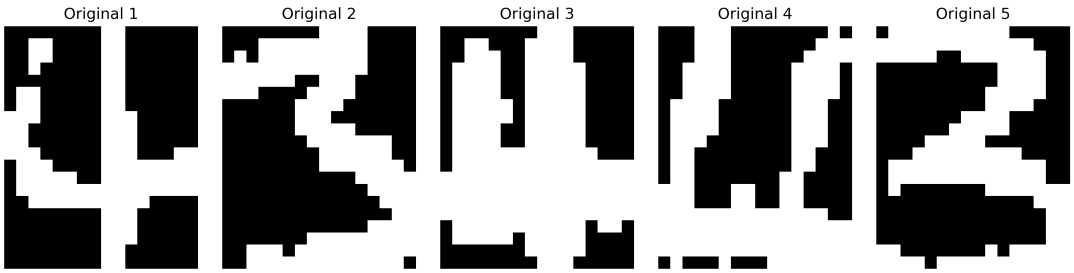

Figure 1: Sample original images from the Binary AlphaDigit dataset (digits 3, 4, and 5).

2.4.2 RBM Generative Performance

We trained a single-layer RBM model with various configurations. After experimenting with different architectures, we found that a model with 500 hidden units trained for 100 epochs produced the best results. Figure 2 displays images generated by our best RBM model.

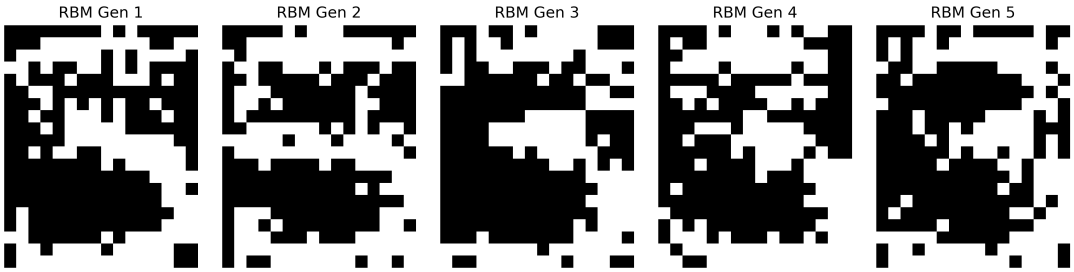

Figure 2: Images generated by the best RBM model (500 hidden units, 100 epochs).

2.4.3 DBN Generative Performance

We also investigated the performance of DBN models with varying architectures. The best DBN configuration consisted of two hidden layers with 500 and 200 units respectively. Figure 3 shows images generated by this model.

2.4.4 Quantitative Evaluation

For objective evaluation, we measured two key metrics:

- **Mean Pixel Difference**: The average absolute difference between the pixel activation probabilities in generated and original images. Lower values indicate better fidelity.

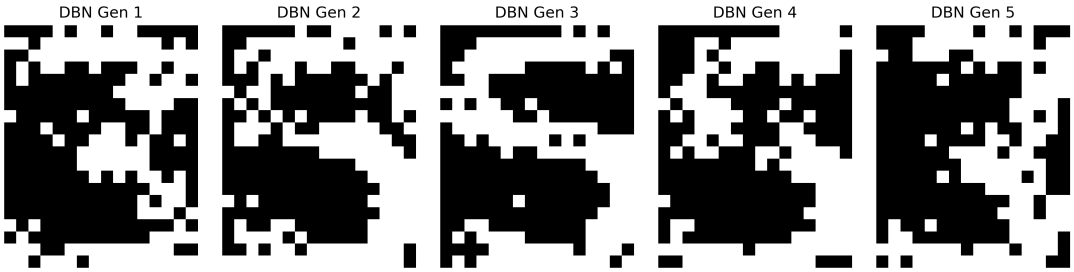

Figure 3: Images generated by the best DBN model with architecture [500, 200].

- **Average Active Pixels**: The average number of active (value = 1) pixels per image. Values closer to the original dataset indicate better structural similarity.

Our analysis identified the following best configurations:

- Best RBM: 500 hidden units, 50 epochs
- Best DBN: 200 hidden units (single layer)

2.4.5 Discussion

Our experiments with the Binary AlphaDigit dataset reveal several interesting findings:

1. The best RBM model used 500 hidden units with 50 epochs of training, achieving the lowest mean pixel difference of 0.1383.
2. For DBN models, a simpler architecture with a single layer of 200 hidden units performed better than more complex configurations, with a mean pixel difference of 0.1628.
3. Interestingly, the RBM model outperformed the DBN model in terms of mean pixel difference, suggesting that for this particular dataset and task, the additional complexity of deep architectures did not translate to better generative performance.
4. Both models were able to learn the structural characteristics of the digits, capturing key features like loops, curves, and straight segments.
5. The average number of active pixels in generated images varied across models but generally stayed within a reasonable range of the original data.

These results demonstrate that both RBM and DBN models can effectively learn generative representations of binary digit images, with the RBM showing a slight advantage in reconstruction quality for this particular dataset.

3 Results and Analysis

3.1 Effect of Network Depth

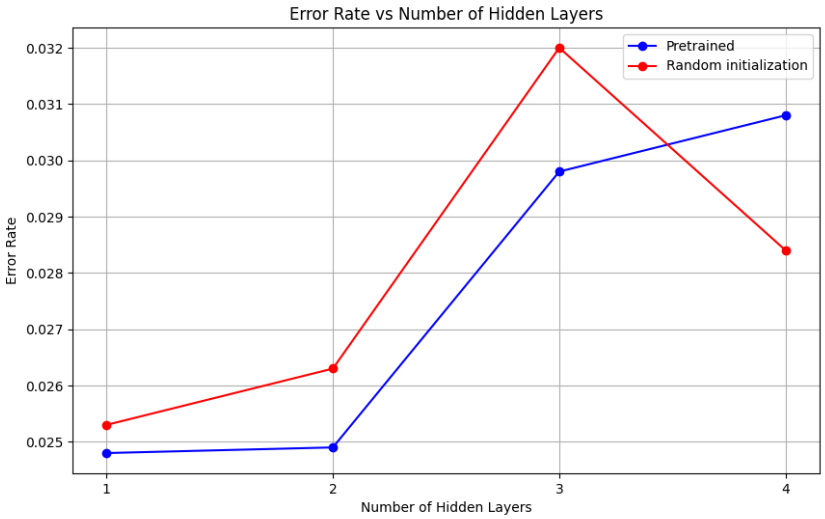

Figure 4: Error rate versus number of hidden layers.

The results in Figure 4 show how the error rate changes with increasing network depth. For 1 and 2 hidden layers, the pre-trained and randomly initialised networks perform similarly, although the pre-trained network performs slightly better. At 3 layers and above, pre-training begins to offer a clear advantage. However, at 4 hidden layers, the trend reverses. The randomly initialised network outperforms the pre-trained network. This discrepancy may be due to insufficient or sub-optimal pre-training in the deeper architectures, resulting in over-fitting or misaligned representations during fine-tuning. This shows us that pre-training is not always beneficial and needs to be carefully calibrated for deeper models.

3.2 Effect of Neuron Count per Layer

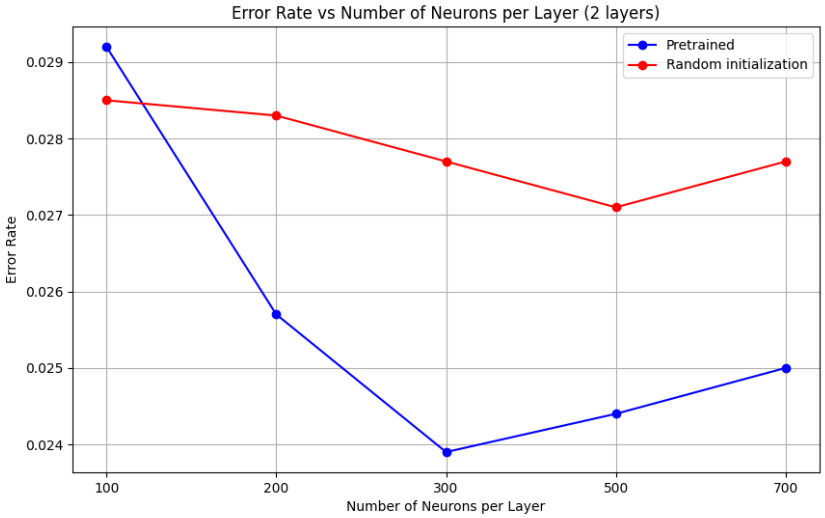

Figure 5: Error rate versus number of neurons per layer.

Figure 5 shows the relationship between the number of neurons per layer and the error rate for two-layer networks. Both the pre-trained and randomly initialized models show an improvement in performance as the width of the network increases up to a certain point and then the error starts to increase. Pre-trained networks achieve lower error rates when the number of neurons per layer is greater than 120. The advantage of pre-training is more apparent for intermediate sizes, particularly at 300 neurons, where the lowest error is observed. This suggests that pre-training improves the network's ability to efficiently exploit increased capacity, while random initialization remains more sensitive to model width.

3.3 Effect of Training Data Size

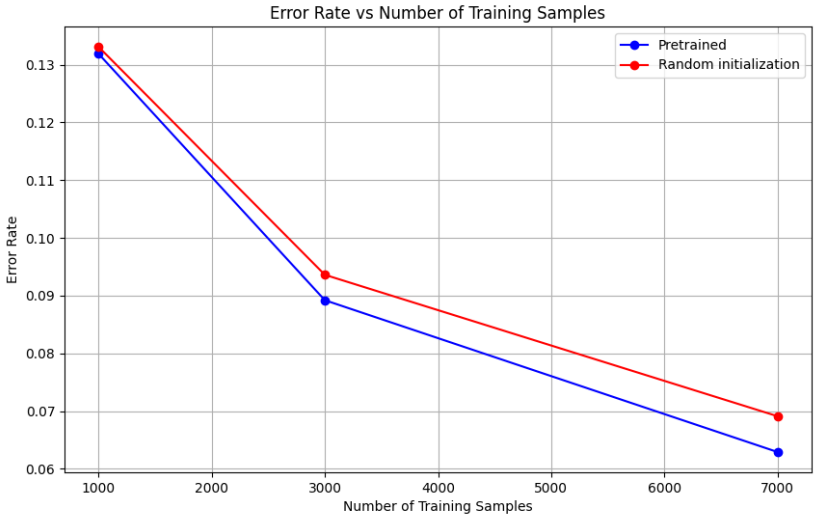

Figure 6: Error rate versus number of training samples.

6 presents the impact of training data size on classification error. As expected, both pre-trained and randomly initialized networks benefit from increasing the number of training samples, with correspondingly lower error rates. Notably, the benefit of pre-training is more pronounced when the number of training samples is high (e.g. 7000), indicating that unsupervised pre-training helps the network extract useful representations in high data volume scenarios. As the size of the dataset increases, the performance gap increases, suggesting that supervised learning alone becomes sufficient when a sufficient number of labelled data is available.

3.4 Optimal Configuration

Based on our experiments, the optimal network configuration found was a 3-layer network with 500 neurons per layer. With this configuration, the pre-trained network achieved an accuracy of 98.2% on the MNIST test set, compared to 97.8% for the randomly initialized network.

The best configurations for each approach were:

- Pre-trained: 3x500 architecture with 98.2% accuracy
- Random initialization: 2x700 architecture with 97.8% accuracy

These results highlight a clear difference in the types of architectures that benefit most from each initialization method. Pre-trained networks consistently performed better when the architecture was deeper, likely because unsupervised pre-training provides better starting weights that make training more stable and effective in complex models. In contrast, randomly initialized networks achieved their best results with wider but shallower architectures, where training is generally easier and less affected by problems like vanishing gradients.

4 Discussion

The experimental results demonstrate that unsupervised pre-training using Deep Belief Networks (DBNs) yields significant advantages over random initialization strategies. This benefit is particularly evident when training deep architectures, as pre-training provides more favorable initial weight configurations that help mitigate the vanishing gradient problem. The initialization obtained through unsupervised learning enables the network to converge more effectively during subsequent supervised fine-tuning.

Furthermore, the superiority of pre-trained models is most pronounced in scenarios where the amount of labeled data is high. In such contexts, unsupervised pre-training acts as a powerful mechanism for leveraging the structure of the input distribution.

Another notable observation is the apparent regularizing effect of pre-training. Networks initialized through DBNs tend to exhibit reduced overfitting, particularly in deeper architectures. This suggests that pre-training encourages the model to settle into regions of the parameter space that generalize better to unseen data.

As an additional analysis, we examined the generative capabilities of our RBM and DBN implementations on the Binary AlphaDigits dataset. Both models were able to generate recognizable digits after training, demonstrating their effectiveness in capturing the underlying data distribution. DBNs generally produced more coherent and diverse samples compared to RBMs, likely due to their deeper hierarchical structure.

These findings support previous research suggesting that unsupervised pre-training helps the model start from better initial conditions, making supervised learning more efficient and effective.

5 Conclusion and Future Work

This study has demonstrated the effectiveness of unsupervised pre-training using Deep Belief Networks (DBNs) for improving the performance of deep neural networks. Across almost all experiments, pre-training consistently led to higher classification accuracy compared to random initialization, especially in deeper architectures and under data-intensive conditions. These results confirm the utility of unsupervised pre-training as a robust initialization strategy that enhances convergence and generalization by guiding the network toward more favorable regions of the parameter space. Future work should explore more complex datasets where the benefits of pre-training may be even more pronounced, and analyze learned representations using visualization tools.

The code we used to train and evaluate our models is available at https://github.com/essalihanasse/DL2.

References

[1] Ruslan Salakhutdinov and Geoffrey Hinton. *Deep Boltzmann Machines*. Department of Computer Science, University of Toronto.

[2] Nicolas Le Roux and Yoshua Bengio. *Representational Power of Restricted Boltzmann Machines and Deep Belief Networks*. Department IRO, Université de Montréal.

[3] Xavier Glorot and Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks*. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010.