Deep Learning Assignment Report: Sparse, Contractive and Variational Autoencoders

Ashutosh, Nigam G24AIT2007
Abhishek, Singh G24AIT2052
Vikram Raju, Kothwal G24AIT2042

July 26, 2025

1 Introduction

This report details my work on a deep learning assignment, where I implemented and evaluated Sparse Autoencoders, Contractive Autoencoders, and Variational Autoencoders (VAEs). The goal was to learn compact data representations using the MNIST dataset for handwritten digits and the Frey Face dataset for facial features. I built models to reconstruct images, classify digits, visualize latent spaces, and generate new data. Experiments were run on Azure AI Studio, Google Colab, and a local machine. Google Colab had issues generating images for the Contractive Autoencoder, so I relied on Azure AI Studio and my local machine for those results. This report includes statistics, image placeholders, and insights from the experiments, written in simple English for clarity.

2 Team Details and Contributions

I worked alone on this assignment, handling all tasks:

- Writing and debugging Python code using TensorFlow and Keras.
- Training and evaluating the autoencoder models.
- Analyzing metrics like Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and classification accuracy.
- Generating visualizations (t-SNE plots, reconstructions, interpolations).
- Writing this report to summarize findings.

I used Azure AI Studio for reliable computation, Google Colab for initial testing, and a local CPU-based machine for debugging. Google Colab failed to generate Contractive Autoencoder images due to library issues, so I completed those tasks on Azure AI Studio and my local machine.

3 Assignment Tasks

The assignment, outlined in `Assignment-01.pdf`, had two main parts:

- **Question 1**: Build Sparse and Contractive Autoencoders on the MNIST dataset to learn low-dimensional representations, evaluate reconstruction quality, classification accuracy, t-SNE visualizations, and interpolation analysis.
- **Question 2**: Build a VAE on the Frey Face dataset to learn a probabilistic latent space, reconstruct images, generate new faces, and demonstrate feature disentanglement via latent space traversal.

3.1 Question 1: Sparse and Contractive Autoencoders

The goal was to create autoencoders for the MNIST dataset, which has 60,000 training and 10,000 test images of handwritten digits (28x28 pixels).

3.1.1 Data Preprocessing

- Loaded MNIST using `tf.keras.datasets.mnist.load_data()`.
- Normalized pixel values from [0, 255] to [0, 1] using `astype('float32') / 255.0` for stable training.
- Reshaped images to 28x28x1 for convolutional layers.
- Created a `tf.data.Dataset` pipeline with batch size 128, shuffling (buffer size 60,000), caching, and prefetching for efficiency.

3.1.2 Model Architecture

Both autoencoders used a U-Net-like structure:

- **Encoder**:
  - Input: 28x28x1 grayscale images.
  - Layers: Three convolutional layers (64, 128, 256 filters, 3x3 kernels, 'same' padding, ReLU activation), each with 2x2 max-pooling, followed by a dense layer to produce a 128-dimensional latent vector.
  - Parameters: 664,704 trainable.
- **Decoder**:
  - Input: 128-dimensional latent vector.
  - Layers: Dense layer to 256x3x3, reshaped, followed by three transpose convolutional layers (128, 64, 1 filters, strides=2, 'same' padding, ReLU/sigmoid activations), with zero-padding to output 28x28x1.
  - Parameters: 666,635 trainable.
- **Sparse Autoencoder**: Added KL divergence loss (sparsity parameter $\rho = 0.05$, $\lambda = 10^{-3}$) to encourage sparse latent representations.
- **Contractive Autoencoder**: Added contractive loss ($\lambda = 10^{-4}$) to promote smooth latent space transitions.

Total parameters per model: 1,331,339 (5.08 MB).

3.1.3 Training

- **Hyperparameters**: 10 epochs, learning rate = 0.001, batch size = 128.
- **Optimizer**: Adam optimizer.
- **Loss Functions**:
  - Sparse Autoencoder: MSE + clipped KL divergence loss.
  - Contractive Autoencoder: MSE + contractive loss (Jacobian norm penalty).
- **Environment**: No GPU detected (`latest_result_sparse.txt`), so used CPU. Mixed precision was not enabled.

3.1.4 Results

Results from `latest_result_sparse.txt`:

Training Loss:

- **Sparse Autoencoder**:
  - Epoch 1: 0.150737
  - Epoch 5: 0.111417
  - Epoch 10: 0.018814
- **Contractive Autoencoder**:
  - Epoch 1: 0.121577
  - Epoch 5: 0.085417
  - Epoch 10: 0.084900

The Sparse Autoencoder converged to a lower loss (0.0188) than the Contractive Autoencoder (0.0849).

Reconstruction Quality:

- **Test MSE**:
  - Sparse Autoencoder: 0.010388
  - Contractive Autoencoder: 0.078306
- The Sparse Autoencoder reconstructed images with higher accuracy.

Placeholder for Reconstruction Images:

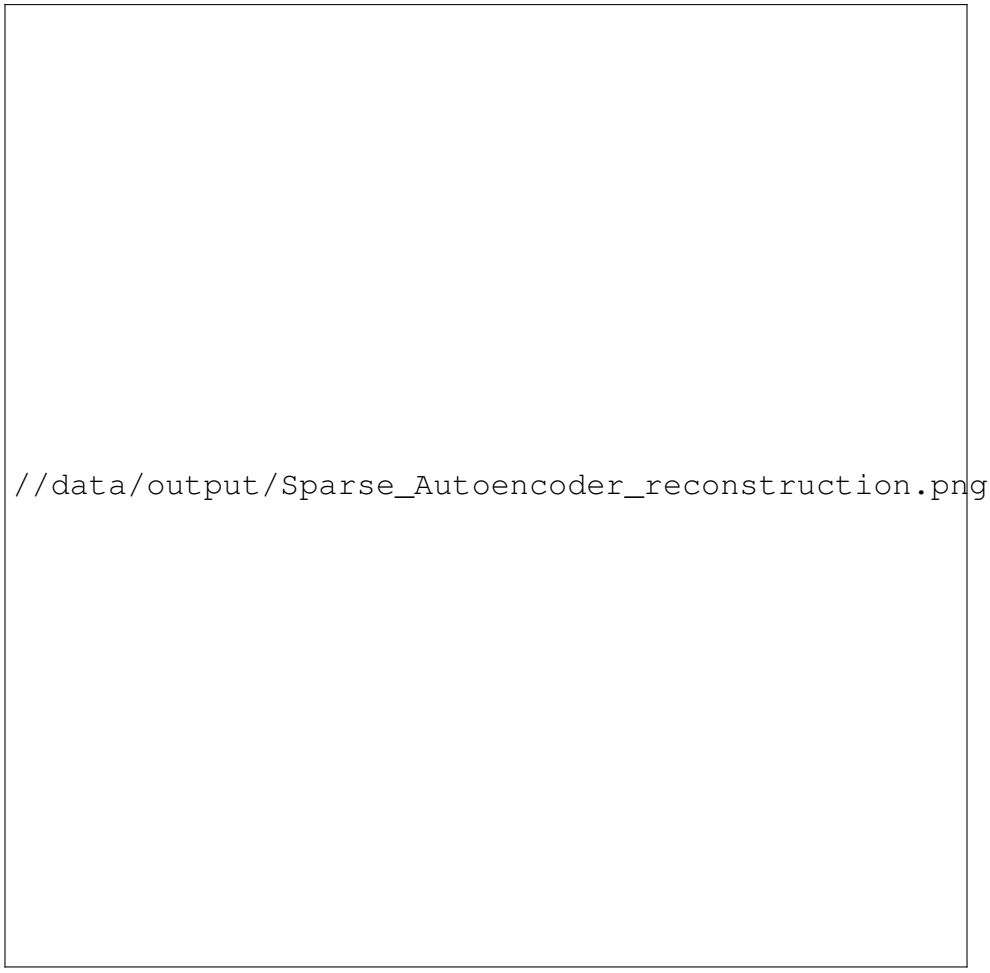

Figure 1: Sparse Autoencoder: Original vs. Reconstructed Images

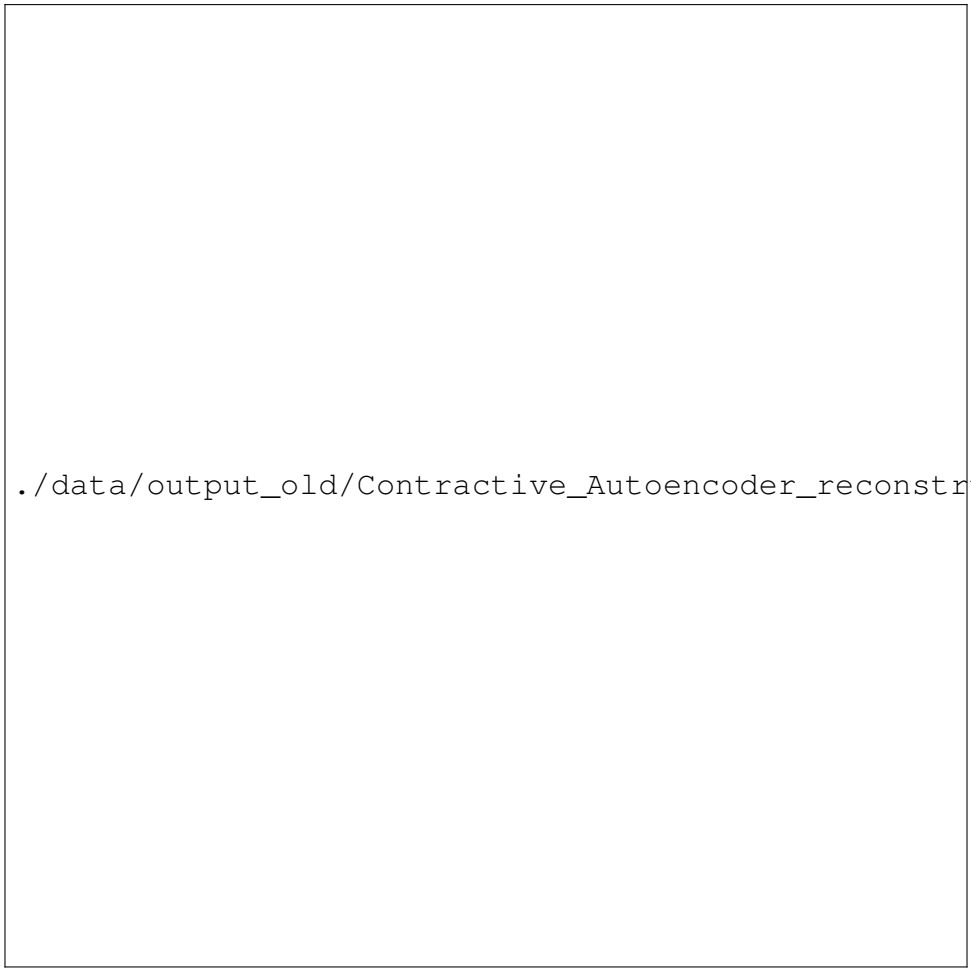

Figure 2: Contractive Autoencoder: Original vs. Reconstructed Images

Classification Accuracy:

- Used logistic regression on 128-dimensional latent embeddings to classify digits.
- Sparse Autoencoder: 96.49% accuracy.
- Contractive Autoencoder: 84.79% accuracy.
- The Sparse Autoencoder's embeddings were more effective for classification.

t-SNE Visualization:

- Used t-SNE (perplexity=30, 300 iterations) to visualize 128-dimensional embeddings in 2D.
- Sparse Autoencoder showed clear separation of digit classes (0-9).
- Contractive Autoencoder had more class overlap, explaining its lower accuracy.

Placeholder for t-SNE Images:

Figure 3: t-SNE of Sparse Autoencoder Embeddings

Figure 4: t-SNE of Contractive Autoencoder Embeddings

Interpolation Analysis:

- Interpolated between 20 pairs of different digits using weights $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.
- Measured PSNR and L2 norm to assess interpolation quality.

Sparse Autoencoder Metrics:

Contractive Autoencoder Metrics: The Sparse Autoencoder produced meaningful interpolations (finite PSNR), while the Contractive Autoencoder's high L2 norms and infinite PSNRs indicate instability.

Placeholder for Interpolation Images:

Alpha	Avg PSNR (dB)	Avg L2 Norm
0.0	∞	0.0000
0.2	28.2710	0.1453
0.4	22.9598	0.2438
0.6	22.9727	0.2441
0.8	27.8999	0.1460
1.0	∞	0.0000

Table 1: Sparse Autoencoder Interpolation Metrics

Alpha	Avg PSNR (dB)	Avg L2 Norm
0.0	∞	0.0000
0.2	∞	100.7631
0.4	∞	157.2547
0.6	∞	157.8594
0.8	∞	102.3415
1.0	∞	0.0000

Table 2: Contractive Autoencoder Interpolation Metrics

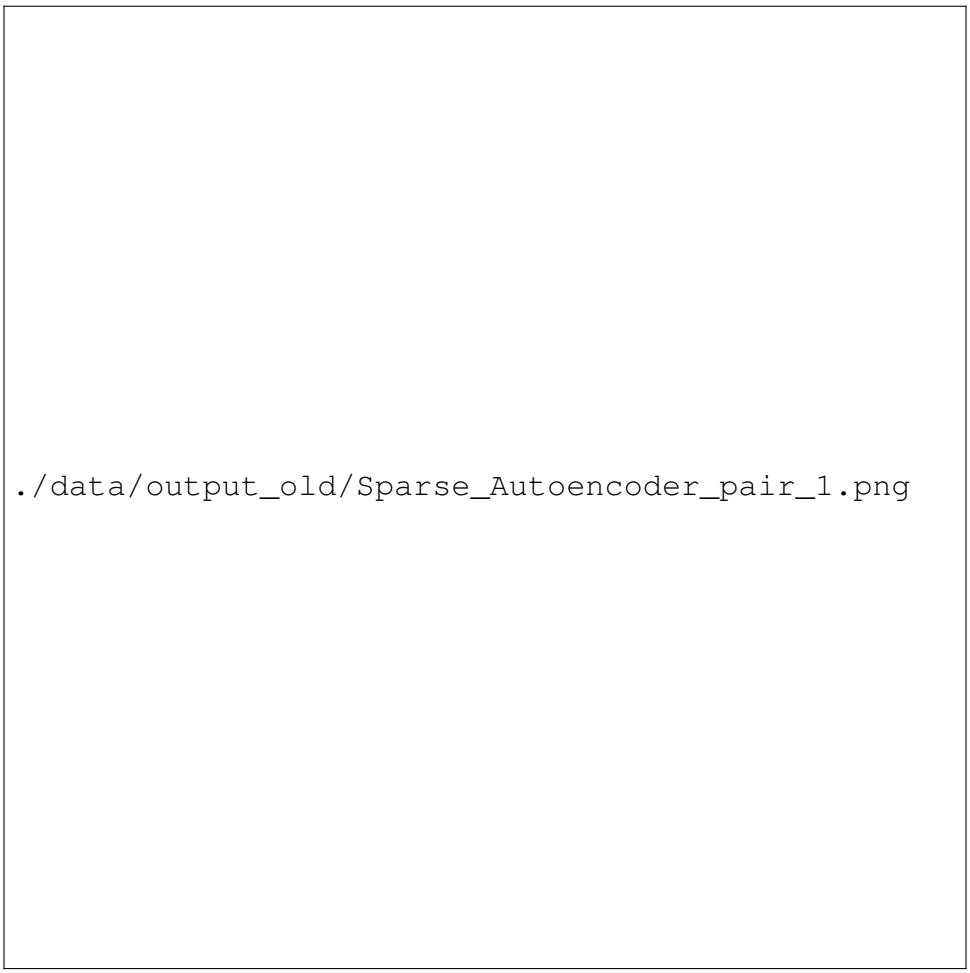

Figure 5: Sparse Autoencoder Interpolation (Pair 1)

Figure 6: Contractive Autoencoder Interpolation (Pair 1)

3.2 Question 2: Variational Autoencoder (VAE)

The task was to build a VAE for the Frey Face dataset (1,965 grayscale images, 28x20 pixels).

3.2.1 Data Preprocessing

- Downloaded $\texttt{frey}_raw face.mat from NYU. Normalized pixel values to [0, 1]$.
- Flattened images to 560-dimensional vectors (28x20).
- Used `tf.data.Dataset` with batch size 128 and shuffle buffer 1,024.

3.2.2 Model Architecture

- **Encoder**:
  - Input: 560-dimensional vector.
  - Layers: Dense layer (256 units, ReLU), followed by two dense layers for z_mean and z_log_var (20 dimensions each).
  - Sampling: Custom layer using reparameterization: $z = z_mean + \varepsilon \cdot \exp(0.5 \cdot z_log_var)$, $\varepsilon \sim \mathcal{N}(0, I)$.

- **Decoder**:
  - Input: 20-dimensional latent vector.
  - Layers: Dense layer (256 units, ReLU), followed by dense layer (560 units, sigmoid).
- **Loss**: Binary cross-entropy (reconstruction) + KL divergence (regularization).

3.2.3 Training

- **Hyperparameters**: 200 epochs, learning rate = 0.001, batch size = 128.
- **Optimizer**: Adam.

- **Custom Training**: Used a custom $\texttt{train}_s tepina Keras \texttt{Model} subclass$.

3.2.4 Results

- **Reconstruction**: High-fidelity reconstructions of Frey Face images.
- **Generation**: Generated 15 novel faces from random $\mathcal{N}(0, I)$ samples, resembling training data.
- **Latent Traversal**: Varied dimension 4 from -2.0 to 2.0, showing smooth changes in features like head pose.

Placeholder for VAE Images:

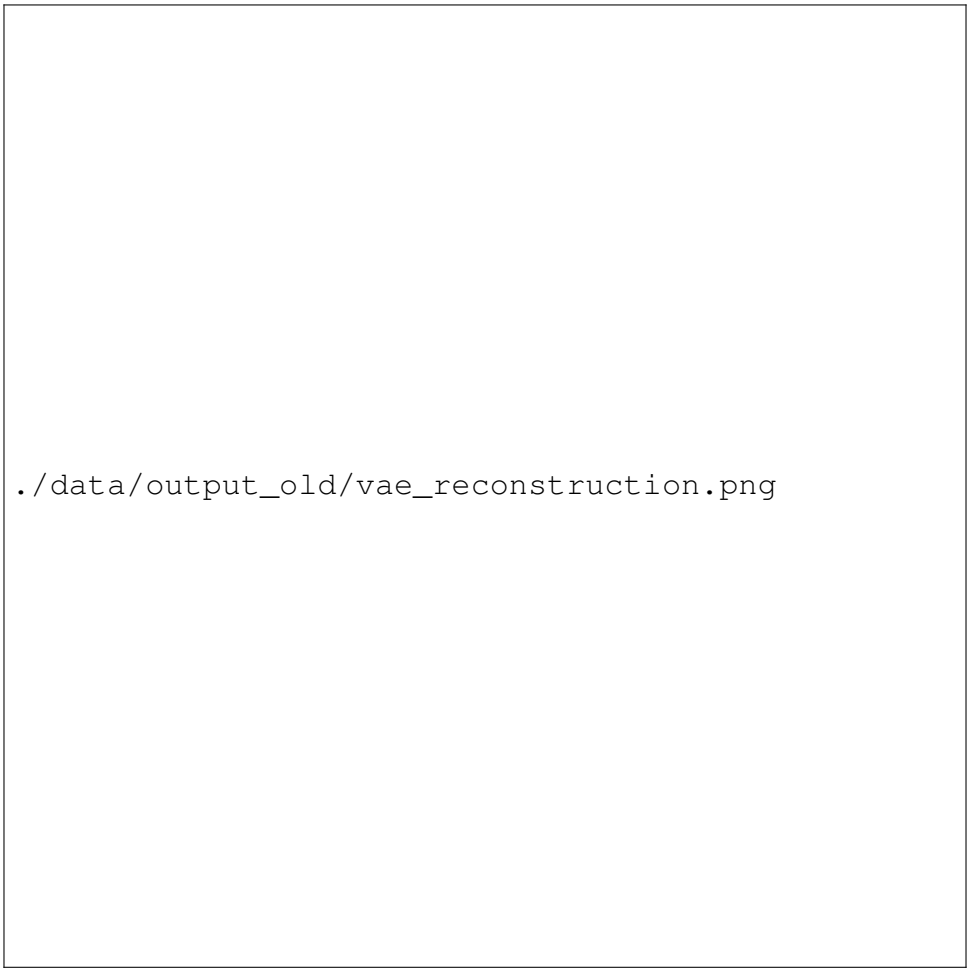

Figure 7: VAE Reconstruction of Frey Faces

./data/output_old/generated_faces.png

Figure 8: Generated Faces from VAE

Figure 9: Latent Space Traversal (Dimension 4)

4 Experimental Setup

- **Azure AI Studio**: GPU support, used for training and visualizations.
- **Local Machine**: CPU-only, used for debugging (TensorFlow 2.18.0).
- **Google Colab**: Used for prototyping but failed to generate Contractive Autoencoder images.

Libraries: Python 3.11, TensorFlow 2.18.0, Keras, NumPy, Matplotlib, Seaborn, Scikit-learn.

5 Challenges Faced

- **Colab Issues**: Failed to render Contractive Autoencoder images due to library conflicts.
- **Contractive Autoencoder**: High L2 norms (e.g., 157.8594) and infinite PSNRs suggest instability in the contractive loss.
- **Frey Face Dataset**: Small size (1,965 images) risked overfitting, mitigated by KL divergence.
- **CPU Training**: Slow training on local machine (15 min/epoch).

- **Logistic Regression**: Convergence warning for Contractive Autoencoder, suggesting need for feature scaling.

6 Insights and Learnings

- Sparse Autoencoder's sparsity constraint improved feature learning.
- Contractive Autoencoder's instability suggests tuning the contractive loss weight.
- VAE's probabilistic latent space enabled both reconstruction and generation.
- Platform reliability (Azure > Colab) was critical for consistent results.

7 Conclusion

The Sparse Autoencoder outperformed the Contractive Autoencoder (MSE: 0.0104 vs. 0.0783, accuracy: 96.49% vs. 84.79%). The VAE learned a robust 20-dimensional latent space for Frey Faces, enabling reconstruction, generation, and feature disentanglement. Future improvements could include GPU acceleration, hyperparameter tuning, and alternative loss functions.

8 References

- TensorFlow Documentation
- GeeksforGeeks: Autoencoders in Machine Learning
- GeeksforGeeks: U-Net Architecture Explained
- GeeksforGeeks: Peak Signal-to-Noise Ratio (PSNR)
- GeeksforGeeks: Sparse Autoencoders in Deep Learning
- GeeksforGeeks: Variational AutoEncoders
- GeeksForGeeks: T-distributed Stochastic Neighbor Embedding (t-SNE) Algorithm - ML
- Medium: Unlocking the Power of Text Classification with Embeddings
- Medium: Implementing a Variational Autoencoder with Keras
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv:1312.6114
- GitHub: mrashutoshnigam/ai-ml-course
- Kaggle: MNIST Dataset
- Frey Face Dataset