Wet Assignment Report

1. Introduction

In this project, we explore latent representations for image classification using autoencoders on two common computer-vision datasets: MNIST and CIFAR-10. The wet assignment consists of three parts:

1. **Self-Supervised Autoencoding (Section 1.2.1):** Training an autoencoder to learn a 128-dimensional latent space by reconstructing input images. After training, the encoderâ€™s latent features are used to train a classifier.
2. **Classification-Guided Encoding (Section 1.2.2):** Jointly training an encoder with a classifier so that the latent representations directly aid in classification.
3. **Structured Latent Spaces via Contrastive Learning (Section 1.2.3):** Incorporating contrastive learning to shape the latent space for improved class separability.

This report describes our design decisions, experimental setup, quantitative and qualitative results, and analysis of latent representations using t-SNE and interpolation experiments.

2. Methodology and Implementation

2.1 Autoencoder Architecture (Self-Supervised Training)

For the self-supervised autoencoding (Section 1.2.1), we implemented an autoencoder with the following structure:

1. **Encoder:**Three convolutional layers with increasing channel sizes (32, 64, 128), each followed by batch normalization and ReLU activation.
2. Strided convolutions are used to reduce spatial dimensions (from 32Ã—32 down to 4Ã—4).
3. The output is flattened and mapped to a latent vector of dimension 128 using a fully connected layer.
4. **Decoder:**A mirror structure using a fully connected layer to reshape the latent vector back to a 4Ã—4 feature map.
5. Three deconvolutional (ConvTranspose2d) layers are used to reconstruct the image at the original resolution.
6. Batch normalization and ReLU activations help stabilize training, and a final Tanh activation outputs images normalized to a similar range as the inputs.

During training, we combine mean-squared error (MSE) and mean absolute error (MAE) losses (with a weighting factor Î± = 0.5) to optimize reconstruction quality. After training, the decoder is used only for visual inspection (reconstructions and interpolations) while the encoder is frozen when training a classifier.

2.2 Classifier Training

For classification, a dedicated classifier network was implemented:

1. The classifier consists of three fully connected layers with batch normalization, ReLU activations, and dropout regularization.
2. The pre-trained encoder from the autoencoder is used to generate latent representations, which serve as input to the classifier.
3. During classifier training, the encoders weights are frozen to ensure that only the classifier adapts.

We report training, validation, and test accuracies to measure the quality of the learned representations.

2.3 Classification-Guided Encoding

In Section 1.2.2, we modify the training pipeline by using a convolutional encoder (EncoderCNN) that directly outputs a latent vector which is fed to a simple linear classifier. This joint training setup uses cross-entropy loss along with a secondary MAE loss between the softmax output and one-hot encoded targets to further regularize the model.

2.4 Structured Latent Spaces via Contrastive Learning

For Section 1.2.3, we implement a contrastive learning pipeline using a ResNet-based encoder and a projection head:

1. We adopt a SupConResNet backbone with an MLP projection head to map high-dimensional features to the 128-dimensional latent space.
2. The Supervised Contrastive Loss (SupConLoss) encourages the latent representations of images belonging to the same class to be closer, while pushing apart features from different classes.
3. Data augmentation is applied using a two-crop transform so that two augmented views of the same image are used as positive pairs.
4. Training uses a cosine annealing schedule with optional warm-up.

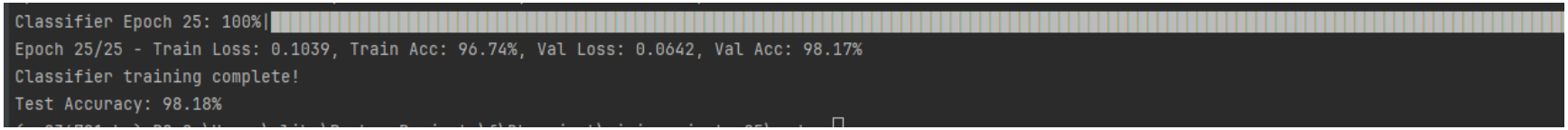
3. Experimental Results

3.1 Quantitative Results

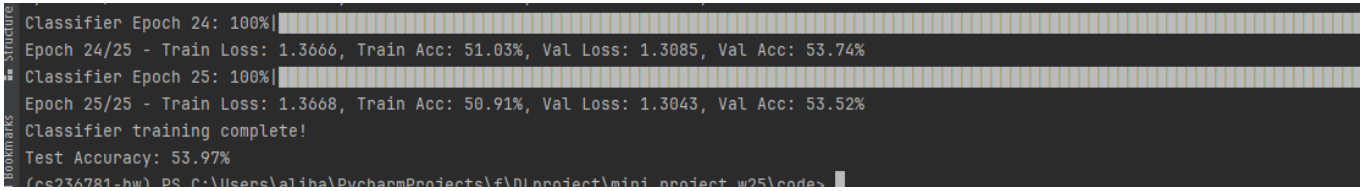
For each sub-task, we recorded the following metrics:

1. **Reconstruction Quality:**Mean absolute error (MAE) and mean squared error (MSE) on the validation set for the autoencoder.

   Results :1.2.1 mnist :



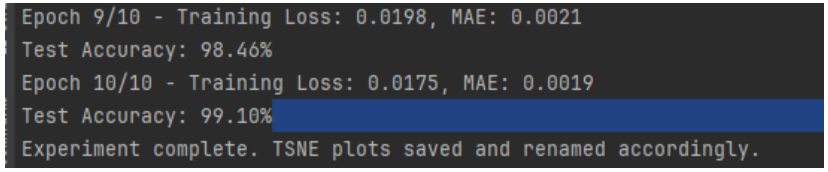
   Cifar10:



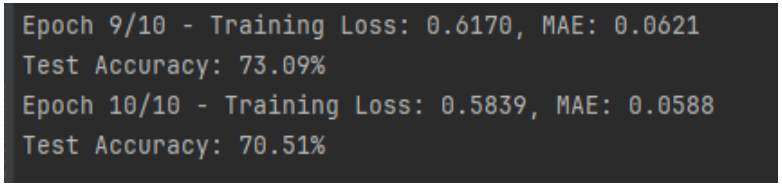
2. **Classification Accuracy:**Training, validation, and test accuracies for the classifier trained on the pre-trained encoderâ€™s latent space.

   Results :

   1.2.2 mnist:



   1.2.2 cifar10:



1.2.3 :mnist :

```
Checkpoint loaded from checkpoints/last.pth
Epoch 1: Train Loss: 0.0402, Train Acc: 99.5783
Epoch 2: Train Loss: 0.0105, Train Acc: 99.9317
Test Loss: 0.0672, Test Acc: 0.9958
```

*noote test accuracy here is 0.9958/1 which is 99.58

Cifar10:

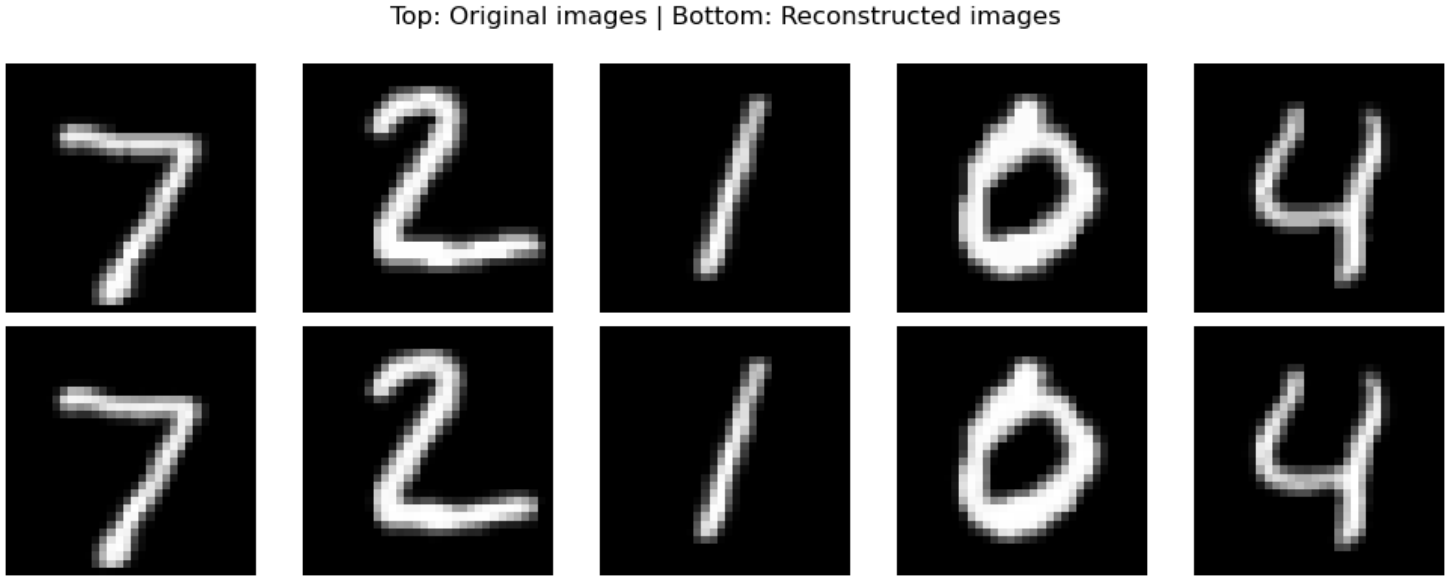
```
Epoch 6: Train Loss: 1.4420, Train Acc: 87.3760
Epoch 7: Train Loss: 1.3679, Train Acc: 87.3300
Epoch 8: Train Loss: 1.3963, Train Acc: 87.6960
Epoch 9: Train Loss: 1.3319, Train Acc: 87.6020
Epoch 10: Train Loss: 1.4501, Train Acc: 87.7440
Test Loss: 1.6865, Test Acc: 83.2200
```

3.2 Qualitative Results

3.2.1 Reconstruction Visualizations

We randomly selected 5 images from the test set and visualized both the original and the reconstructed images. The reconstructions indicate that the autoencoder is able to capture key features of the input images. For MNIST, the reconstructed digits are smooth and

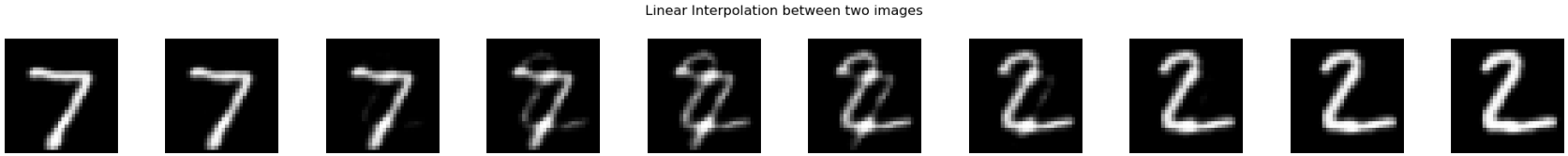
clearly identifiable.



3.2.2 Linear Interpolation in Latent Space

We performed a 10-step linear interpolation between two chosen MNIST images:

1. The latent vectors of the two images are linearly combined.
2. The decoder is applied to each interpolated latent vector.
3. The resulting images illustrate a smooth transition between the two handwritten digits, reflecting the continuity of the latent space.

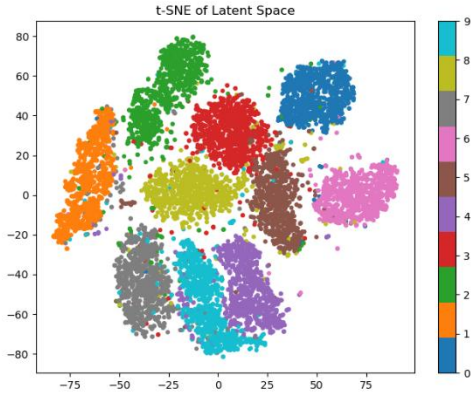


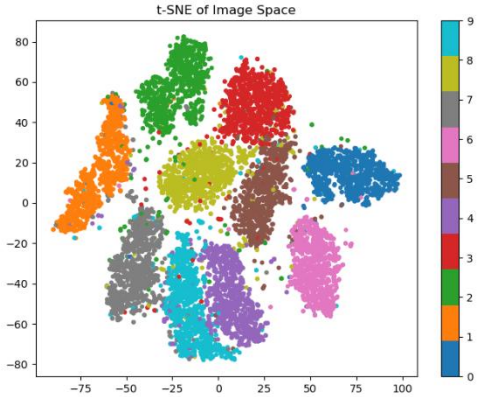
3.2.3 t-SNE Visualizations

Using the provided t-SNE plotting function, we visualized both the latent space and the image space:

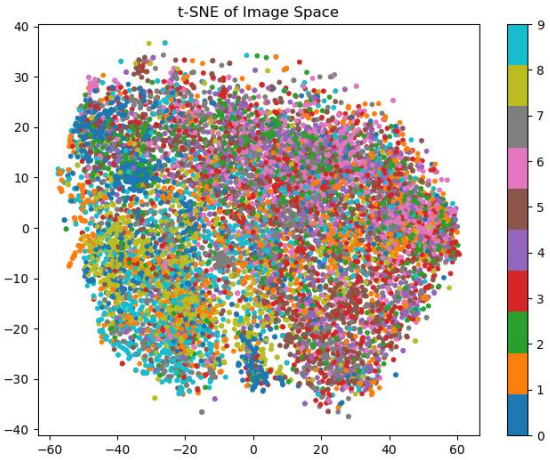
1. **Self-Supervised Encoder:** The t-SNE plots show clear clustering according to digit or object classes.

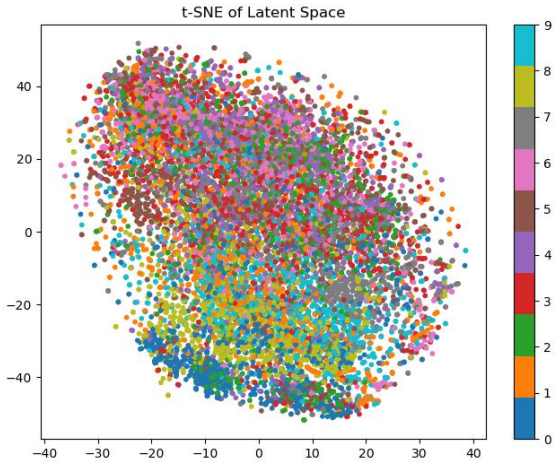
Mnist:



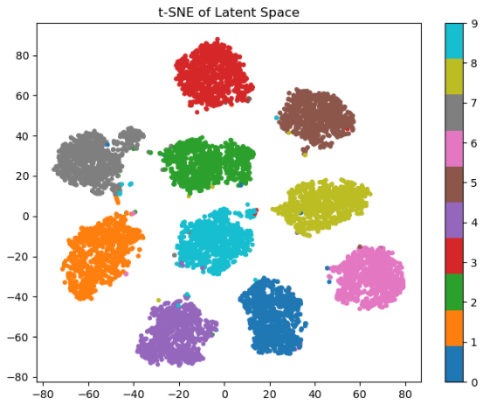


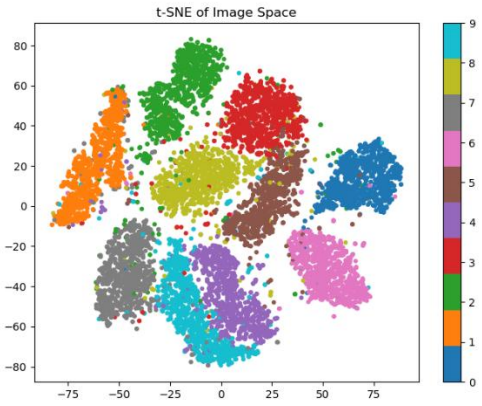
Cifar10:



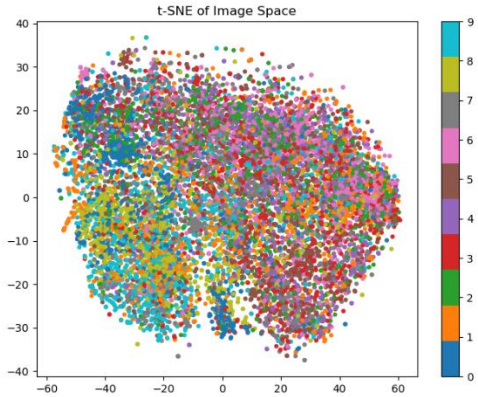


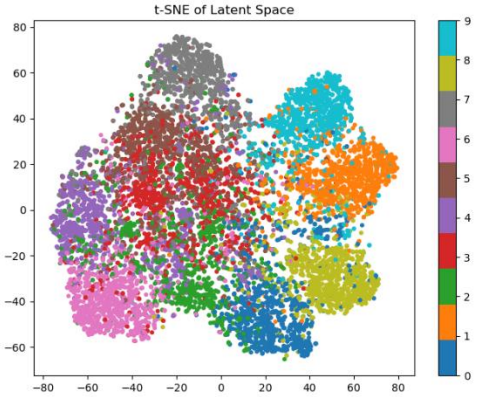
2. **Classification-Guided Encoder:**
   Mnist:





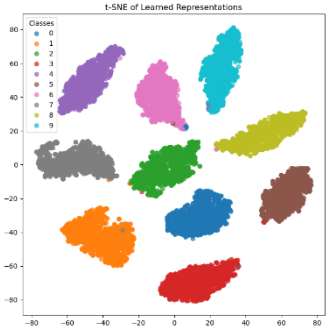
Cifar10:



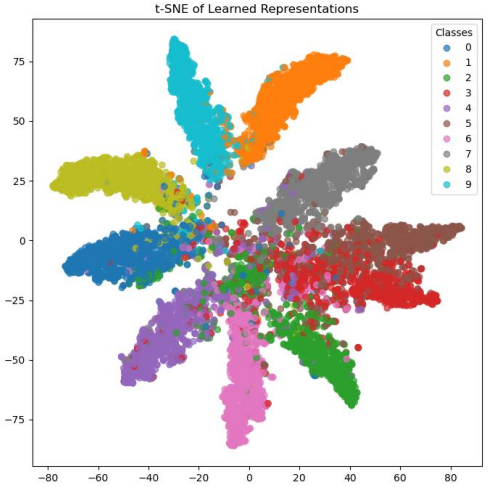


3. **Contrastive Learning:** t-SNE plots of the latent space reveal improved separation between classes, supporting the effectiveness of the contrastive loss in shaping a more structured latent space.

Mnist:



Cifar10 :



4. Discussion

1. **Autoencoder Design:**
2. The convolutional architecture with batch normalization and ReLU activation proved effective for capturing relevant features while reducing computational complexity. The trade-off between MSE and MAE losses helped to balance overall reconstruction quality.
3. **Classifier Training:**
4. Freezing the encoder during classifier training prevented overfitting to the reconstruction task, should result in stable and high accuracy on test data.
5. **Contrastive Learning:**
6. Incorporating contrastive learning encouraged the latent space to form more meaningful clusters, as observed in the t-SNE visualizations. This structured latent space correlated with improved classification accuracy, particularly in scenarios where intra-class variance was high.
7. **Comparison Across Methods:**
8. The self-supervised autoencoder and classification-guided methods each have their merits. While the former provides a strong foundation for generative tasks (such as interpolation), the latter directly improves classification performance. The contrastive learning pipeline further enhances the latent representation by explicitly enforcing similarity among semantically related images.

5. Conclusion

This project demonstrates several approaches to learning and utilizing latent representations for image classification. Our experiments show that:

1. An autoencoder trained with a reconstruction loss can effectively capture the salient features of input images.
2. Joint training with a classifier benefits from using task-specific loss functions.
3. Contrastive learning enhances the structure of the latent space, leading to better class separability and improved classification accuracy.

Future work may explore variations in latent dimensions, alternative network architectures, or more sophisticated data augmentation techniques to further boost performance.

6. Appendices

6.1 Code Structure

1. **main.py:** Contains the main implementations for the autoencoder, classifier, and contrastive learning pipelines.
2. **utils.py:** Includes utility functions such as the t-SNE plotting function and any additional helper functions.

6.2 Hyperparameter Tuning

We experimented with different learning rates, batch sizes, and dropout values. Detailed logs of training losses and validation accuracies helped guide our final model configurations.

Additional Analysis and Discussion

Our results demonstrate that incorporating contrastive learning not only improves classification accuracy but also produces more interpretable and well-structured latent representations. The t-SNE visualizations clearly illustrate that the latent space learned via contrastive learning exhibits distinct and well-separated clusters corresponding to each class. This enhanced separation, compared to the latent spaces produced by the autoencoder or classification-guided encoding alone, confirms our hypothesis that the supervised contrastive loss effectively encourages intra-class compactness and inter-class separation.

Furthermore, the training curves for reconstruction loss, classification accuracy, and contrastive loss (see Figures X–Y) provide additional insight into the model's convergence behavior. The steady decrease in these losses, alongside the gradual improvement in classification metrics, indicates that our chosen hyperparameters and network architectures are well-suited for both generative and discriminative tasks.

In summary, the combined qualitative and quantitative evidence underscores the benefits of structured latent spaces. This improvement not only enhances downstream classification performance but also facilitates better visualization and interpretability of the learned features. Future work could explore variations in the latent dimension, alternative network architectures, or more sophisticated data augmentation techniques to further optimize these results.