In the name of God

University of Tehran

Faculty of Electrical and Computer Engineering

Neural Networks and Deep Learning

Homework 6

Author	Mohammad Taha Majlesi
Student ID	810101504
Submission Date	June 20, 2025

Contents

List of Figures

List of Tables

1 Question 1: Unsupervised Learning and Domain Adaptation with GANs

1.1 Introduction

Large-scale labeled datasets like ImageNet and COCO have been pivotal for progress in computer vision. However, creating such datasets is expensive and time-consuming. An alternative is to use synthetic data from game engines or renderers, but models trained on synthetic data often perform poorly on real-world images due to the **domain gap**.

This project addresses this challenge through unsupervised domain adaptation. We aim to leverage knowledge from a labeled source domain to perform tasks on an unlabeled target domain. Specifically, we implement the approach from the paper "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks," which uses a GAN to translate images from the source domain to mimic the style of the target domain while preserving the original content. This pixel-level translation is performed without any labels from the target domain.

The key advantage of this method is its interpretability and the decoupling of the domain adaptation task from the main learning task. By exploring this paper's implementation, we gain practical experience with domain adaptation, adversarial learning, and the analysis of composite deep learning models.

1.2 Theoretical Section

1.2.1 GAN Instability

The training of Generative Adversarial Networks (GANs) is notoriously unstable. Achieving a stable training process where the generator and discriminator converge to a Nash equilibrium is challenging. Key issues include:

- **Vanishing Gradients:** If the discriminator becomes too powerful too quickly, its loss can saturate, providing near-zero gradients to the generator. This stalls the generator's learning process, as it receives no useful feedback.
- **Mode Collapse:** The generator learns to produce only a limited variety of samples that can fool the current discriminator. It fails to capture the full diversity of the true data distribution. A related issue is **mode dropping**, where the generator learns to produce samples from only a subset of the data classes.
- **Oscillation:** The generator and discriminator can get stuck in a cycle, where one undoes the progress of the other without converging to a stable point. This is due to their conflicting objectives.

Mechanisms to combat these instabilities include architectural changes (e.g., using different normalization layers), adding regularization terms to the loss functions, modifying the training procedure (e.g., two-timescale update rule), and carefully tuning hyperparameters. The paper we implement also uses specific techniques, such as noise vectors and a content similarity loss, to enhance stability.

1.2.2 Use of Both Noise and Image as Input

In a standard GAN, the generator maps a random noise vector z to an image. In this paper's model, the generator takes both a source image x^s and a noise vector z as input. This design has several advantages:

- **Content Preservation:** The source image provides the content that needs to be preserved during the style transfer. The model learns to change the style (e.g., from black-and-white MNIST to colorful MNIST-M) while keeping the digit's shape intact.
- **Stochasticity and Diversity:** The noise vector z allows the generator to produce diverse outputs for the same input image. By providing the same source image with different noise vectors, we can generate multiple versions of that image in the target domain's style, which is useful for data augmentation and robust training.
- **Control and Stability:** Using the source image as a strong prior makes the generator's task easier and more stable than generating an image from noise alone. It constrains the output space, which can help prevent mode collapse and improve convergence.

1.2.3 The Three Core Model Components

The model consists of three main components: a Generator (G), a Discriminator (D), and a Classifier (T).

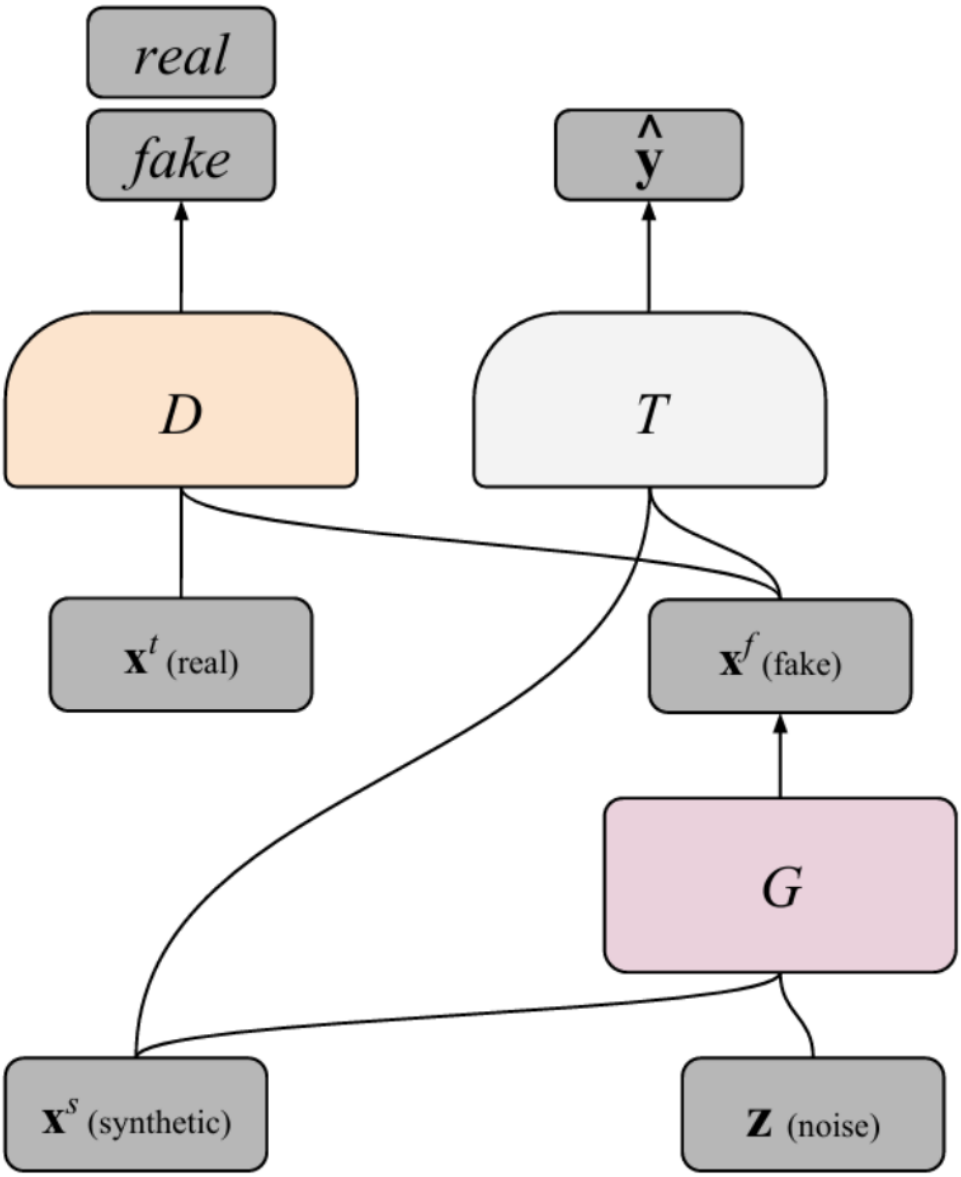


Figure 1: The overall architecture of the model, showing the interplay between the Generator, Discriminator, and Classifier (Task-specific model).

- **Generator (G):** Its goal is to learn the mapping from the source domain to the target domain. It takes a source image x^s and a noise vector z and produces a fake image $x^f = G(x^s, z)$ that should look like it was drawn from the target domain.
- **Discriminator (D):** This is a standard GAN discriminator. It is trained to distinguish between real images from the target domain (x^t) and fake images (x^f) produced by the generator. Its feedback forces the generator to create more realistic images.
- **Classifier (T):** This component is responsible for the final task, which in our case is digit classification. It is trained on both the original source images and the generated fake images. By forcing the generator to produce images that the classifier can correctly identify, we ensure that the content (the digit) is preserved

during the style translation. Without the classifier, the generator might produce realistic-looking target images that have lost the original digit's identity.

1.2.4 Content Similarity Loss

The content similarity loss, L_c, is a crucial feature for ensuring that the generator preserves the content of the source image. While the classifier's loss implicitly encourages content preservation, an explicit loss can make this constraint stronger and the training more stable. The paper mentions using a loss like Perceptual Similarity or, in a simpler case, a pixel-wise loss like Masked Pairwise Mean Squared Error (PMSE). This loss penalizes large deviations between the foreground pixels of the source image and the generated image.

Without this loss, the generator might find it easier to create realistic target images by drastically changing the content. For example, it might turn a "1" into a "7" if that makes the image look more like a typical sample from the target domain. The content similarity loss prevents this, forcing the generator to focus only on changing the style.

1.2.5 Training the Classifier on Both Real and Fake Images

Training the classifier on both real source images and the generated fake images is essential for stable learning. ítemize

Training on Real Source Images: This is the standard supervised training step. It teaches the classifier what the digits look like in the source domain.

Training on Fake Generated Images: This step creates a closed loop with the generator. It forces the generator to produce images that are not only realistic (to fool the discriminator) but also classifiable (to fool the classifier). If the generator produces a "2" that looks like a "7", the classifier's loss will be high, and this gradient will be passed back to the generator to correct its behavior.

If the classifier were trained only on real source images, there would be no direct gradient signal to ensure the generator preserves content. If it were trained only on fake images, it might learn to classify artifacts of the generator rather than the true content. Training on both ensures it learns a robust representation applicable to both domains.

1.2.6 Semantic vs. Stylistic Domain Gaps

The method in this paper is designed for domains like MNIST and MNIST-M, where the difference is primarily stylistic (background, color, texture) while the semantic content (the shape of the digits) is shared.

The approach might not work as well for domains with a large **semantic gap**. For example, adapting from a dataset of cats to a dataset of cars. In this case, there is no shared content to preserve. The generator would have to learn a much more complex, object-level transformation, which is beyond the scope of this pixel-level adaptation method. The core assumption of a shared label space and preserved content would be violated. However, for tasks with subtle semantic shifts (e.g., adapting from photos of dogs to paintings of dogs), the method could still be effective.

1.3 Practical Section

1.3.1 Data Preparation

We use the MNIST and MNIST-M datasets. MNIST contains grayscale images of handwritten digits, serving as our source domain. MNIST-M contains MNIST digits blended with random color patches from the BSDS500 dataset, serving as our target domain. The digit content and image order are preserved between the two datasets.

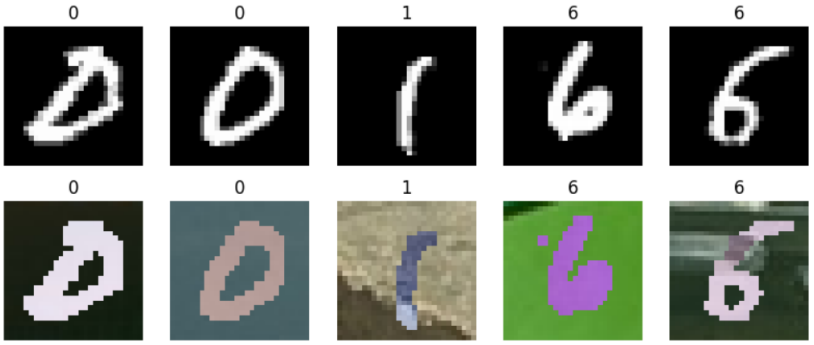


Figure 2: Corresponding samples from the MNIST (top row) and MNIST-M (bottom row) datasets.

The data is preprocessed according to the paper: images are resized to 32x32, MNIST is converted to 3-channel RGB, and pixel values are normalized to [-1, 1]. The datasets are split into 80% training and 20% testing sets using shared indices to maintain correspondence.

1.3.2 Baseline Model: Classifier Trained on Source Only

To quantify the domain gap, we first train a classifier only on the source (MNIST) data and then test its performance on both MNIST and MNIST-M. The classifier architecture is taken from the paper.

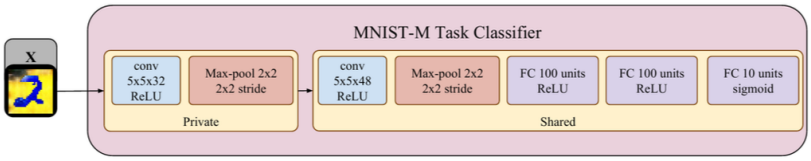


Figure 3: Architecture of the task classifier model, as described in the paper.

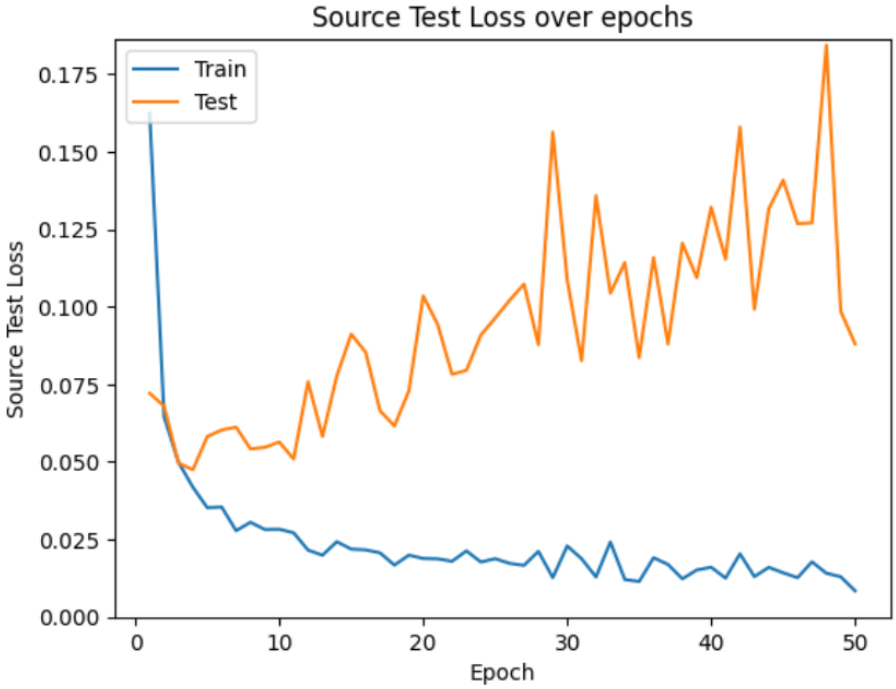


Figure 4: Baseline classifier test loss on the source (MNIST) domain.

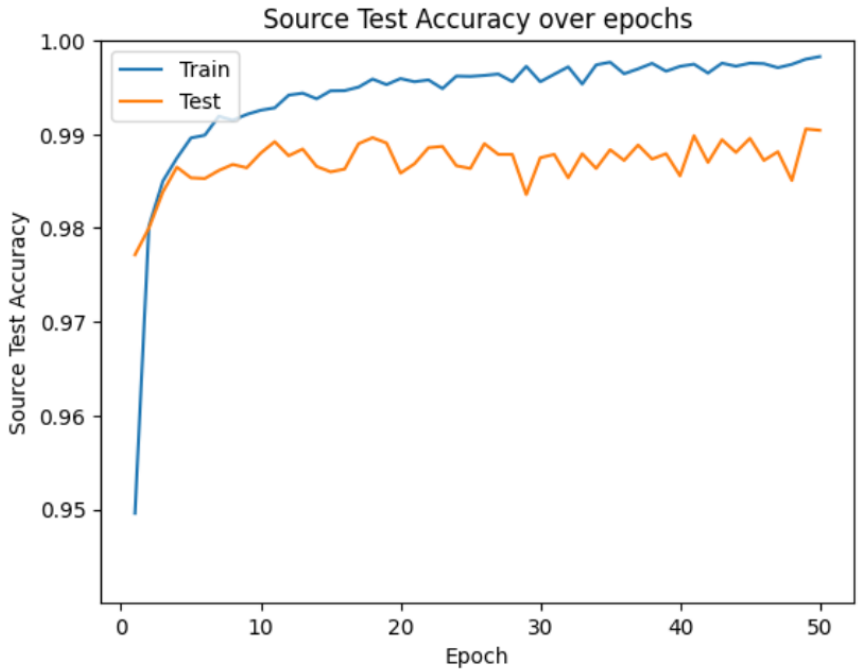
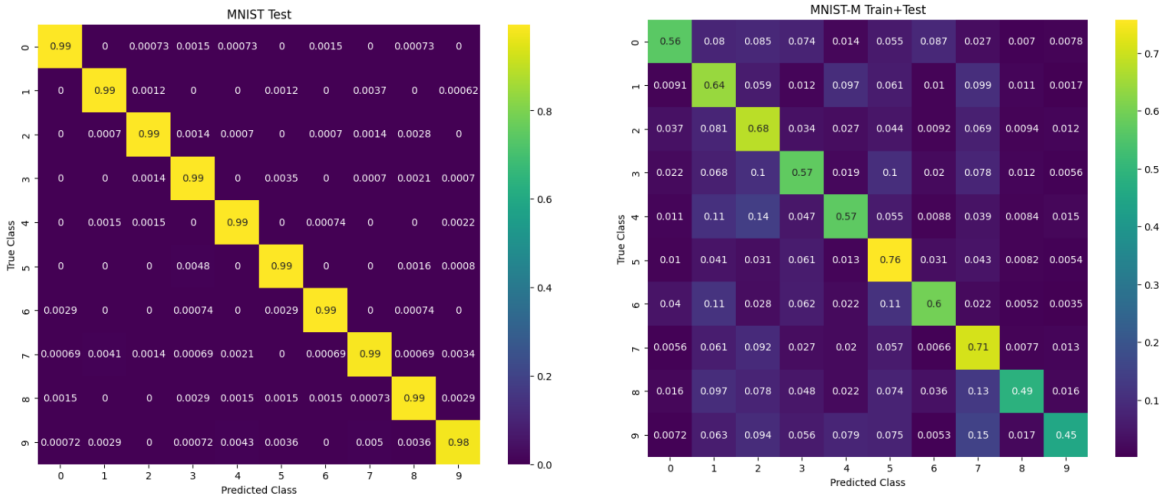


Figure 5: Baseline classifier test accuracy on the source (MNIST) domain.

Table 1: Baseline model evaluation metrics.

Metric	MNIST Test	MNIST-M Test
Accuracy	0.9904	0.6023
Precision	0.9904	0.6451
Recall	0.9904	0.6023
F1 score	0.9904	0.6044



(a) Confusion Matrix on MNIST Test Set. (b) Confusion Matrix on MNIST-M Test Set.

Figure 6: Baseline classifier confusion matrices.

Analysis: The baseline model achieves near-perfect accuracy (99%) on the MNIST test set but performs poorly on the MNIST-M test set (60% accuracy). This significant drop in performance is a clear demonstration of the **domain gap**. The model has learned features specific to the grayscale MNIST distribution and fails to generalize to the colorful, textured MNIST-M domain.

1.3.3 Implementation of the Full Domain Adaptation Model

We now implement the full model, including the generator and discriminator, following the architectures specified in the paper.

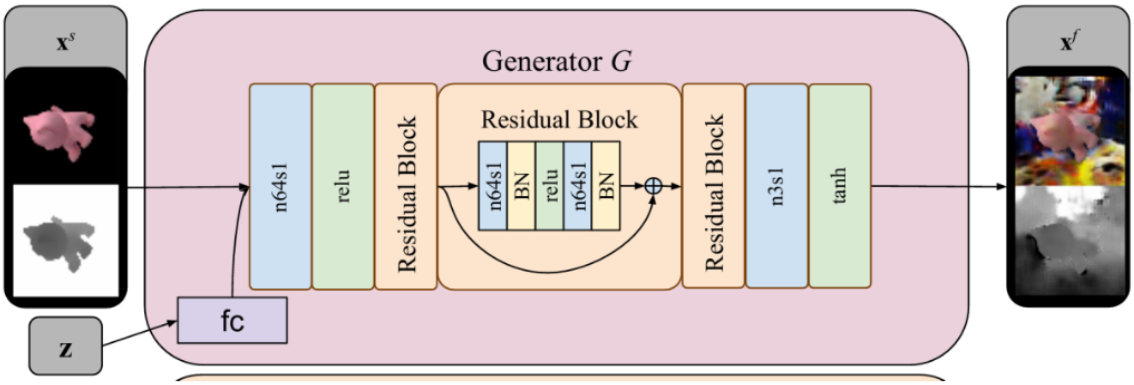


Figure 7: Architecture of the Generator model.

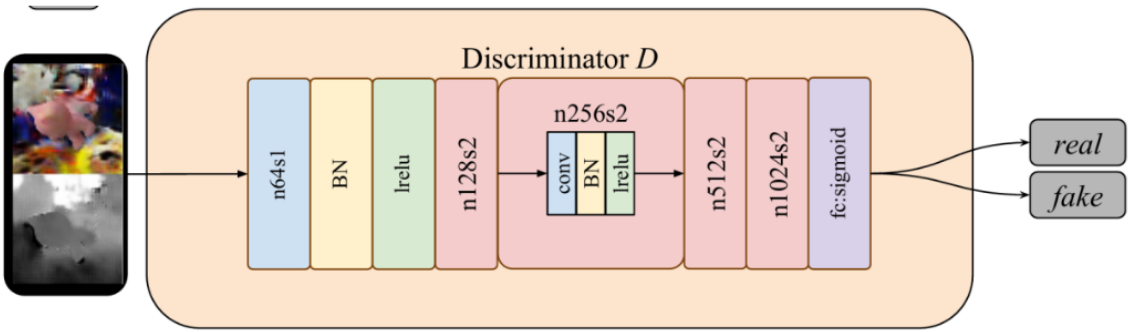
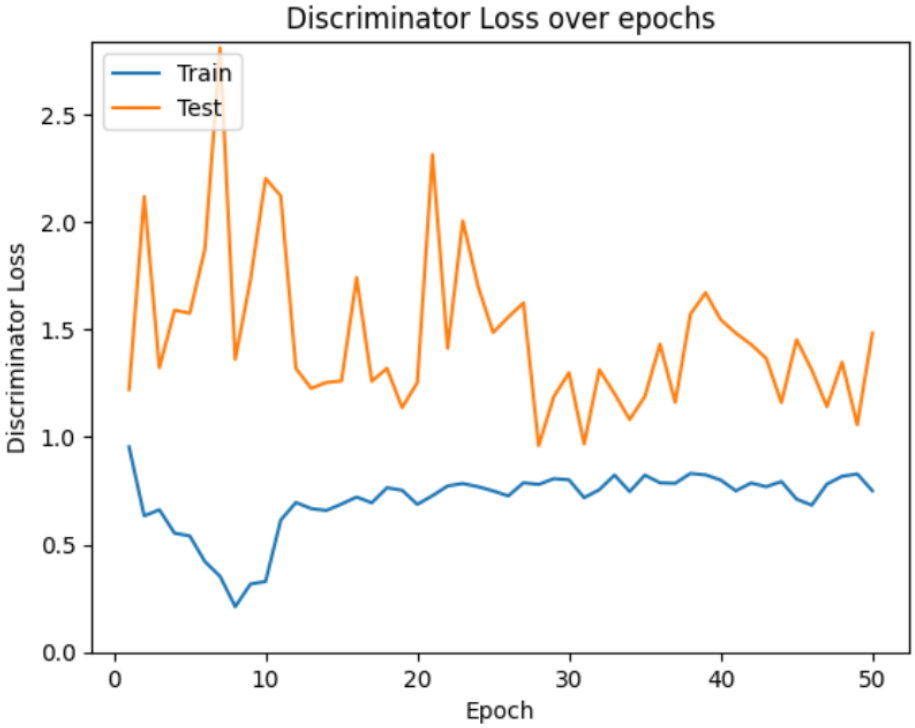


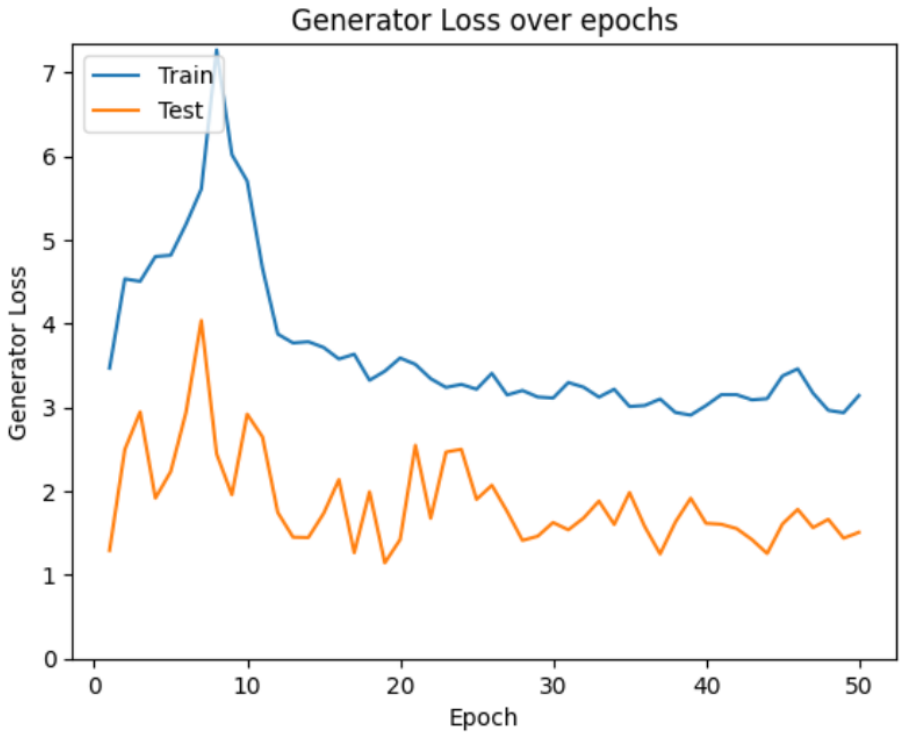
Figure 8: Architecture of the Discriminator model.

1.3.4 Model Training

The full model is trained for 50 epochs. The losses and accuracies for all components are tracked.



(a) Discriminator Loss



(b) Generator Loss

Figure 9: Adversarial training losses.

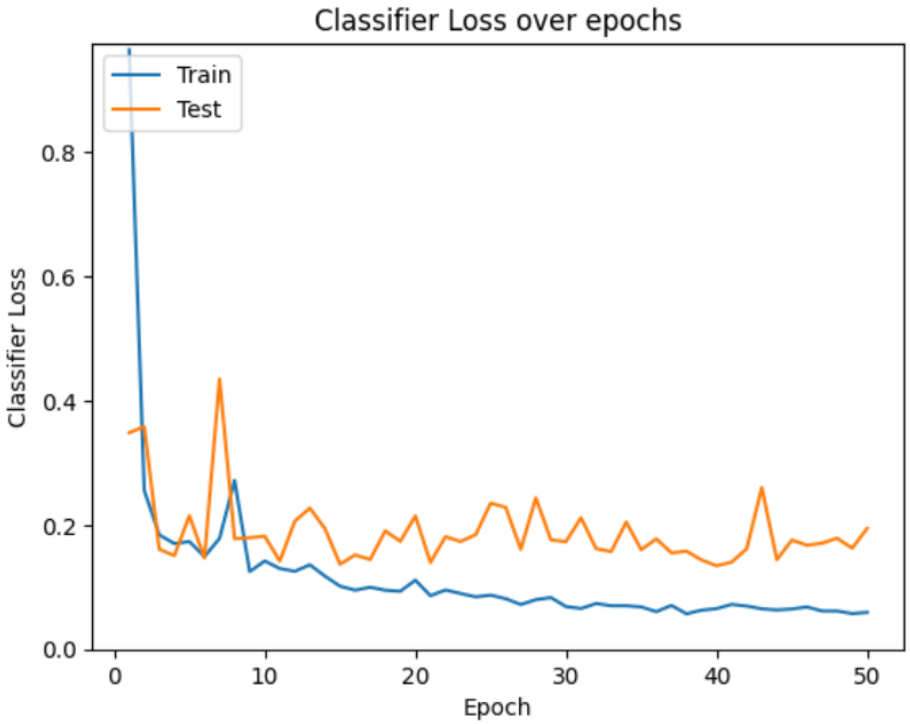
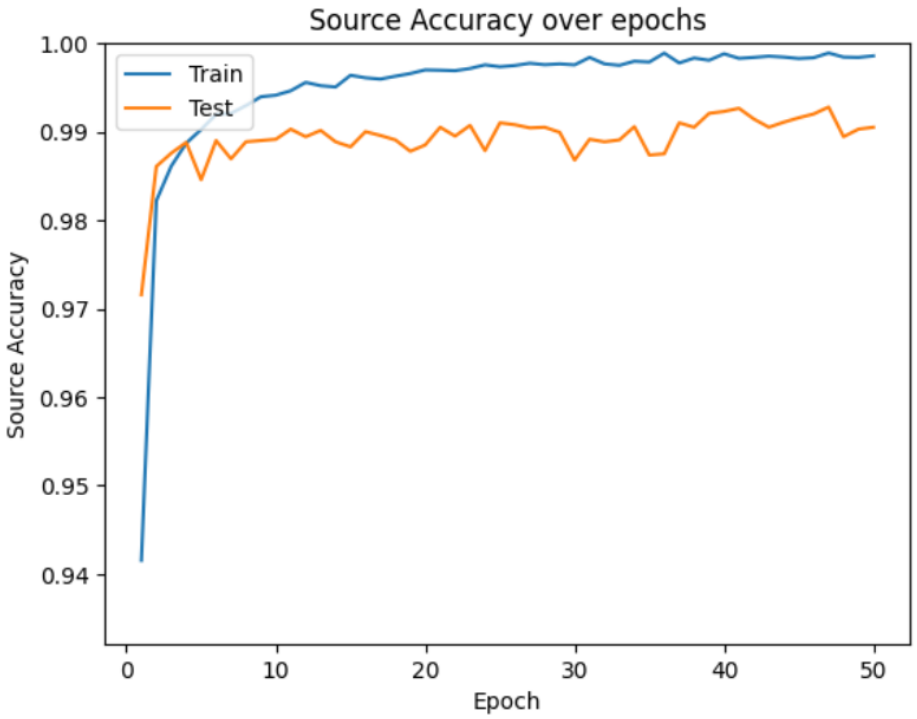
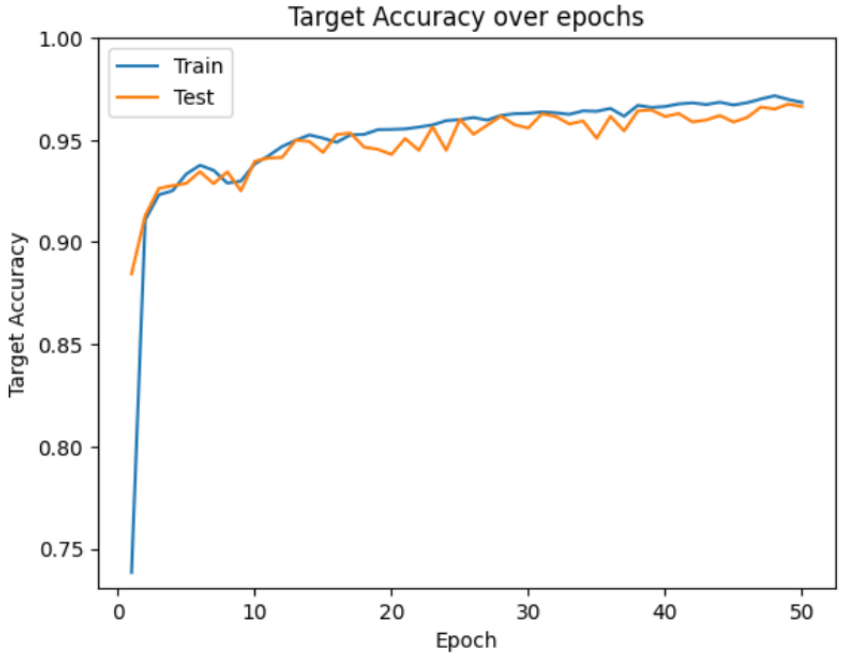


Figure 10: Classifier Loss during training.



(a) Source Domain (MNIST) Accuracy



(b) Target Domain (MNIST-M) Accuracy

Analysis of Training Dynamics: The loss curves for the generator and discriminator show the characteristic adversarial dynamic, with some oscillations but overall convergence. The classifier loss decreases steadily. The accuracy on the target (MNIST-M) domain shows a significant and steady improvement, which is the primary goal of this exercise.

1.3.5 Quantitative Evaluation

After training, we evaluate the final model's performance.

Table 2: Final model evaluation metrics.

Metric	MNIST Test	MNIST-M Test	Fake MNIST-M Test
Accuracy	0.9905	0.9664	0.9706
Precision	0.9905	0.9667	0.9706
Recall	0.9905	0.9664	0.9706
F1 score	0.9905	0.9664	0.9706

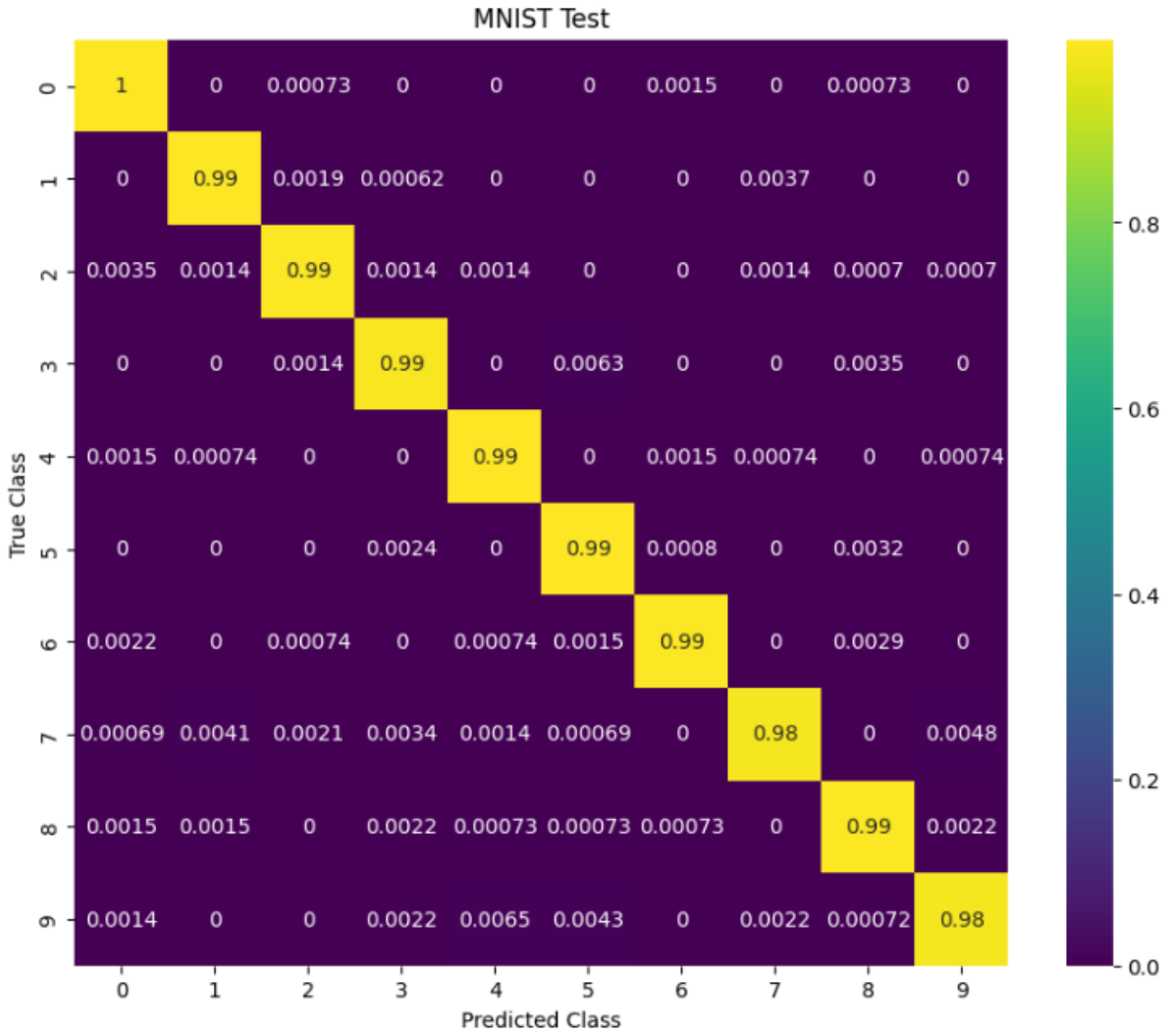


Figure 12: Confusion matrix on the source (MNIST) test set.

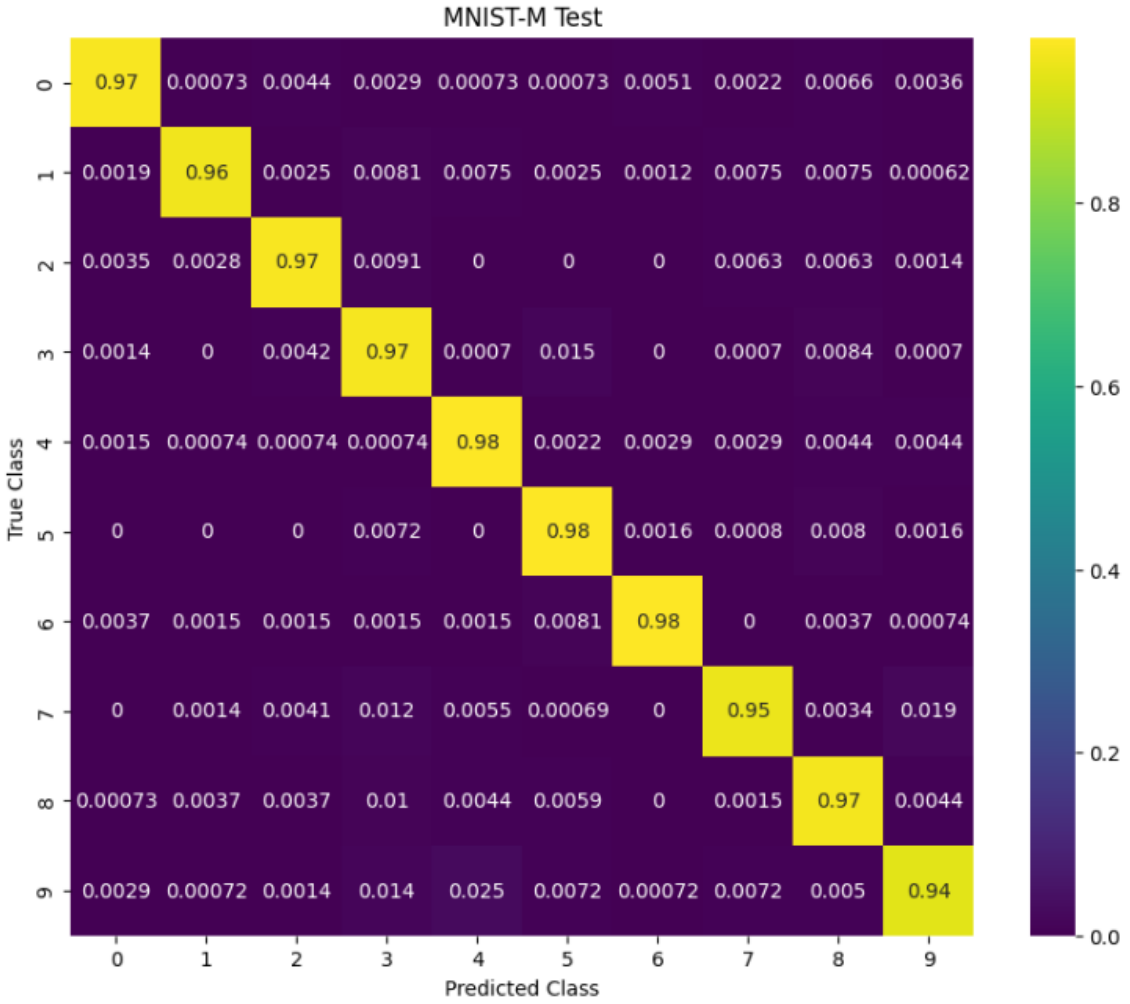


Figure 13: Confusion matrix on the target (MNIST-M) test set.

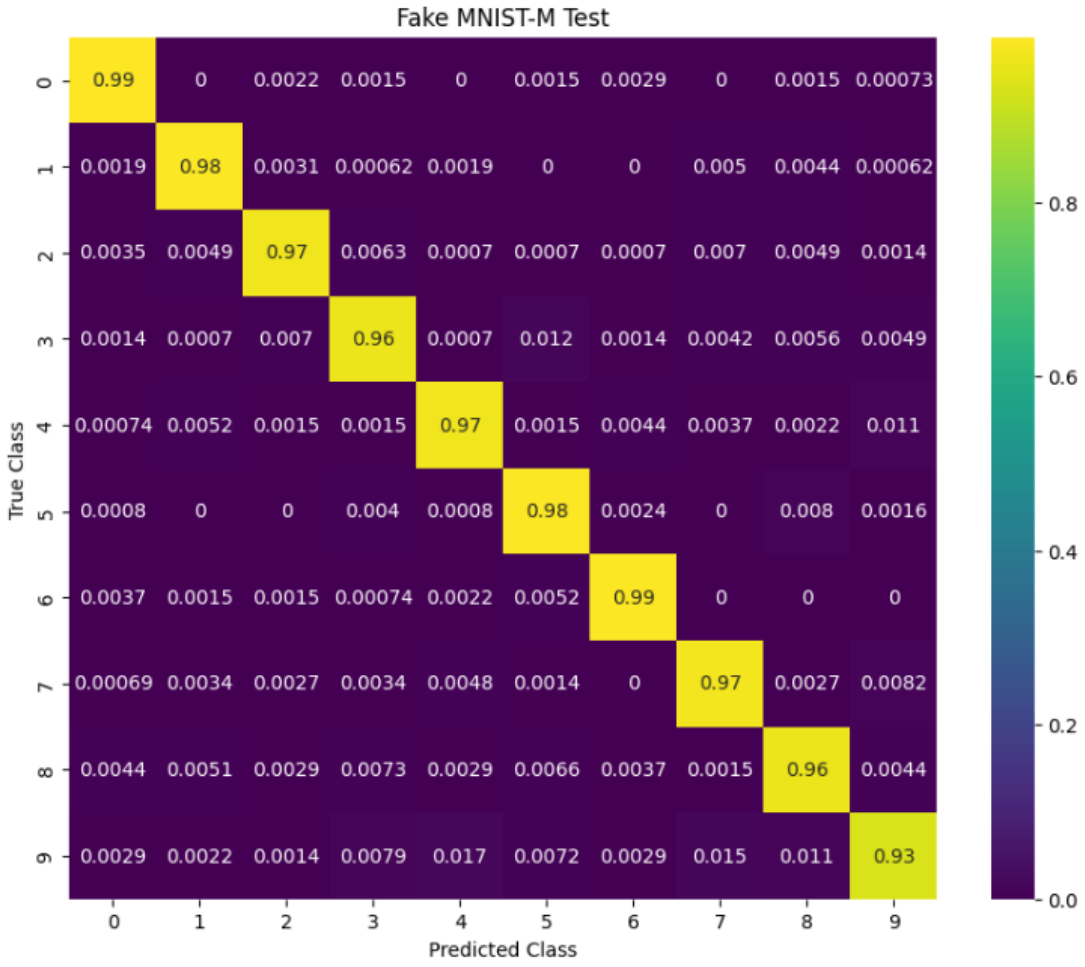


Figure 14: Confusion matrix on the generated (Fake) test set.

Analysis: The results are excellent. The accuracy on the MNIST-M test set has improved dramatically from **60.2%** (baseline) to **96.6%**. This demonstrates that the model has successfully bridged the domain gap. It can now classify images from the target domain almost as well as it classifies images from the source domain, without ever having seen a single label from the target domain. The high accuracy on the "Fake" images also shows that the generator is successfully preserving the digit content during translation.

1.3.6 Qualitative Evaluation

We visualize the generator's output to qualitatively assess the domain transfer.

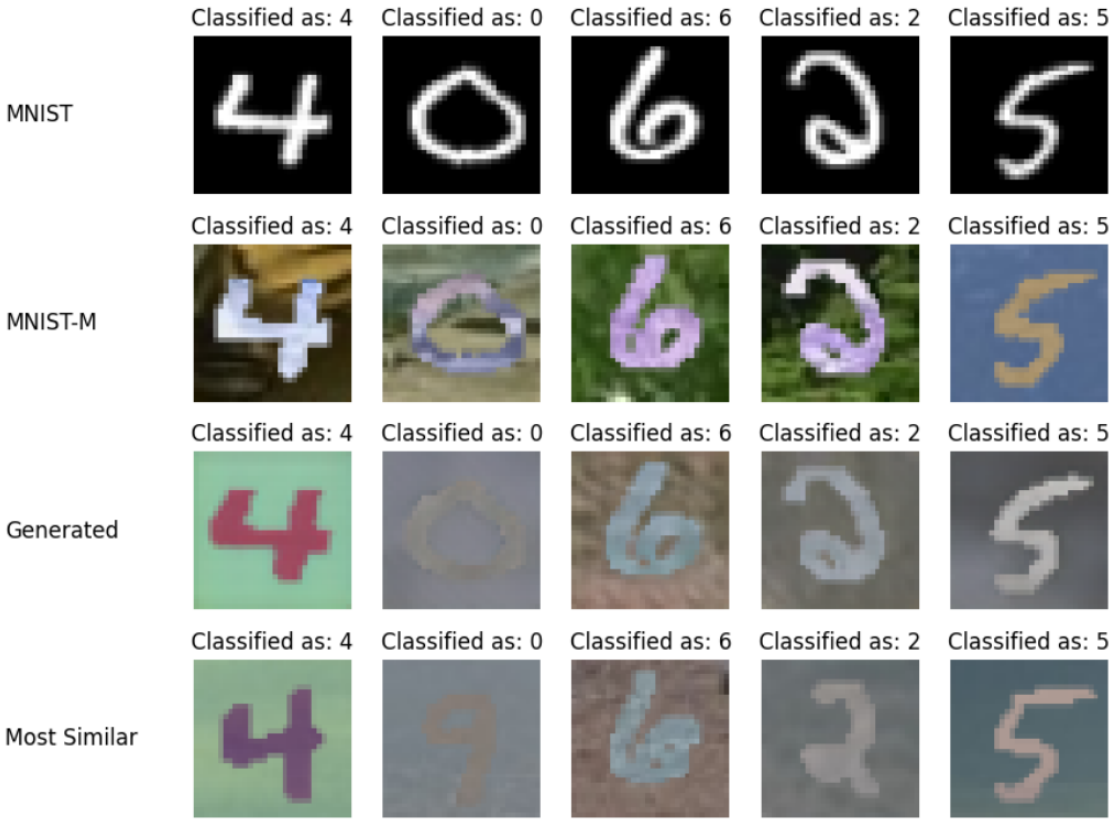


Figure 15: Qualitative results. From top to bottom: Original MNIST image, corresponding real MNIST-M image, generated image, and the most similar training image from the target domain.

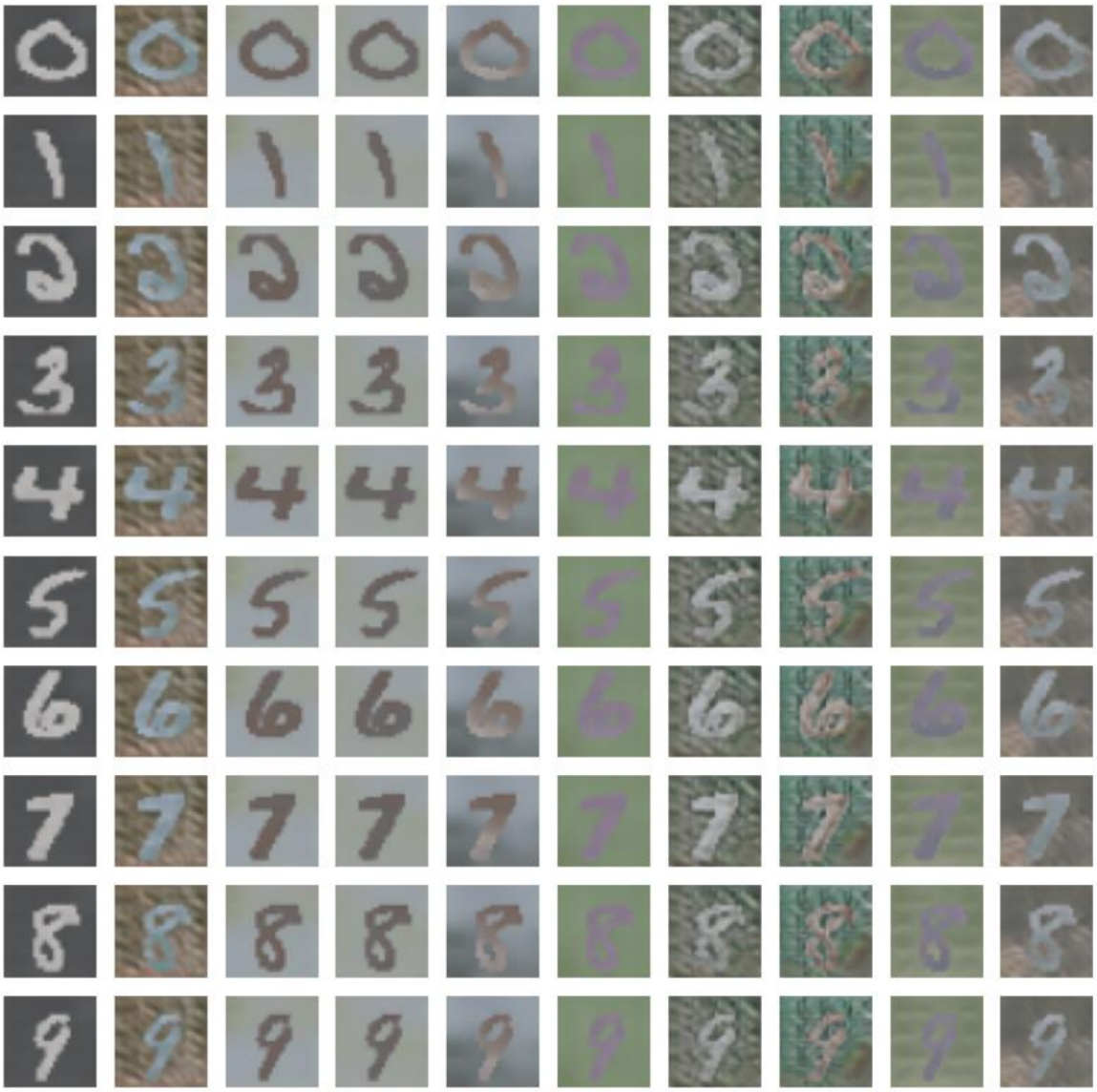


Figure 16: Generated samples for the same digit using 10 different noise vectors.

Analysis: The generated images successfully capture the style of the MNIST-M dataset (colorful, textured backgrounds) while perfectly preserving the digit from the MNIST source image. The classifier's predictions on these generated images are highly accurate. The second figure demonstrates the effect of the noise vector: for a single input digit, the model can generate a variety of backgrounds, showcasing the diversity of the learned mapping.

1.3.7 Ablation Study: Using a Single Unified Classifier

As an ablation study, we modified the architecture to use a single, fully shared classifier for both domains instead of the private-shared structure from the paper.

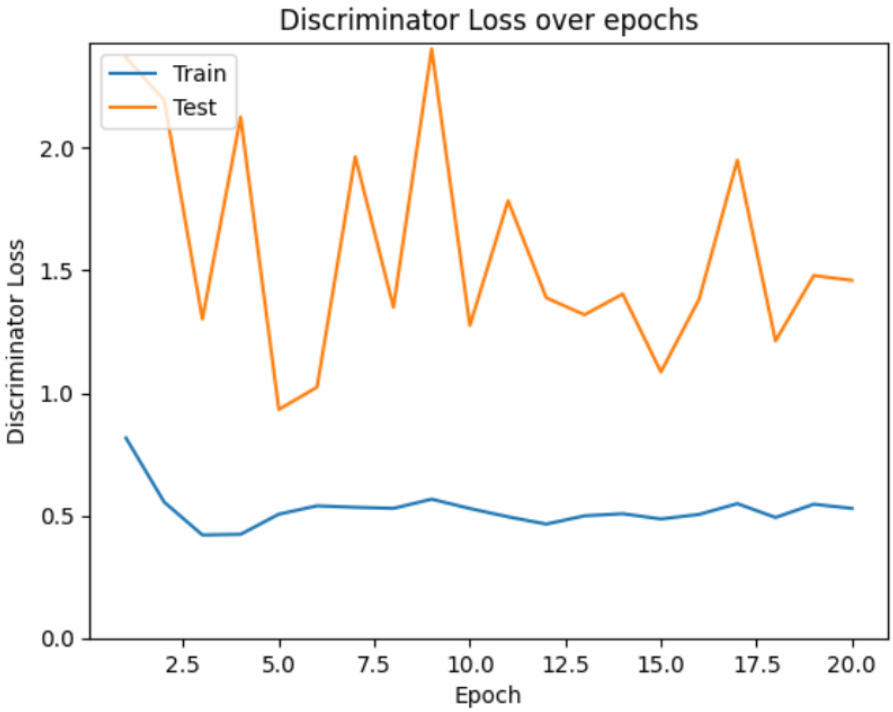


Figure 17: Discriminator Loss (Unified Classifier).

Figure 18: Generator Loss (Unified Classifier).

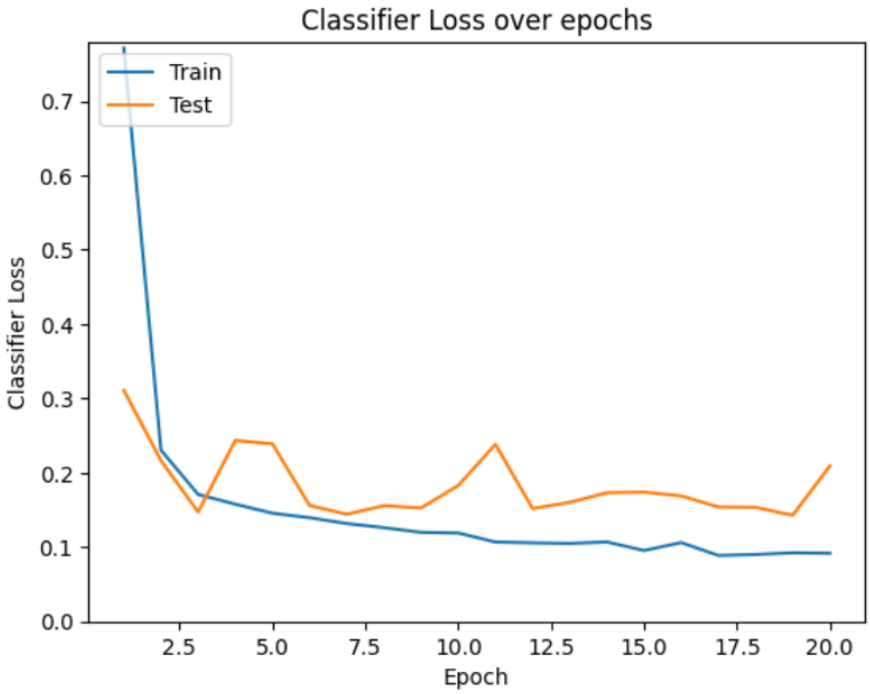


Figure 19: Classifier Loss (Unified Classifier).

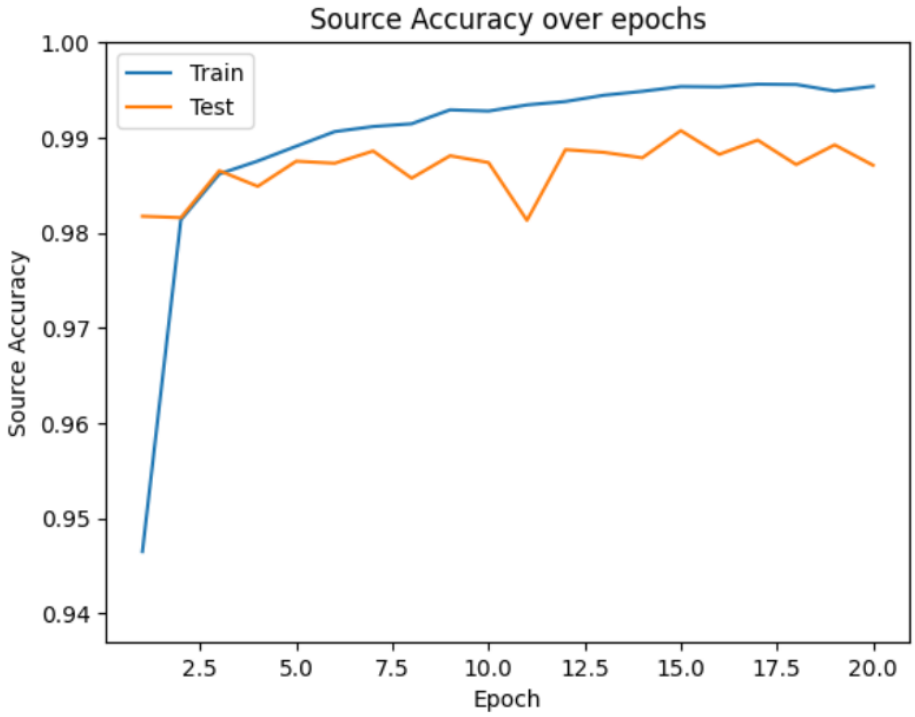


Figure 20: Source Accuracy (Unified Classifier).

Figure 21: Target Accuracy (Unified Classifier).

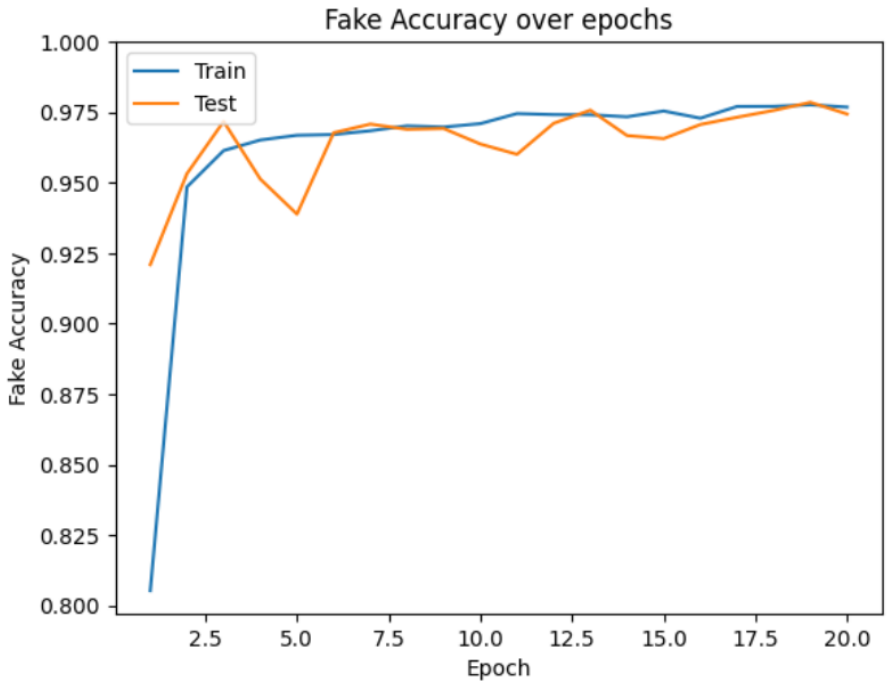


Figure 22: Fake Image Accuracy (Unified Classifier).

Table 3: Evaluation metrics for the model with a single unified classifier.

Metric	MNIST Test	MNIST-M Test	Fake MNIST-M Test
Accuracy	0.9871	0.9423	0.9754
Precision	0.9872	0.9440	0.9756
Recall	0.9871	0.9423	0.9754
F1 score	0.9871	0.9425	0.9754

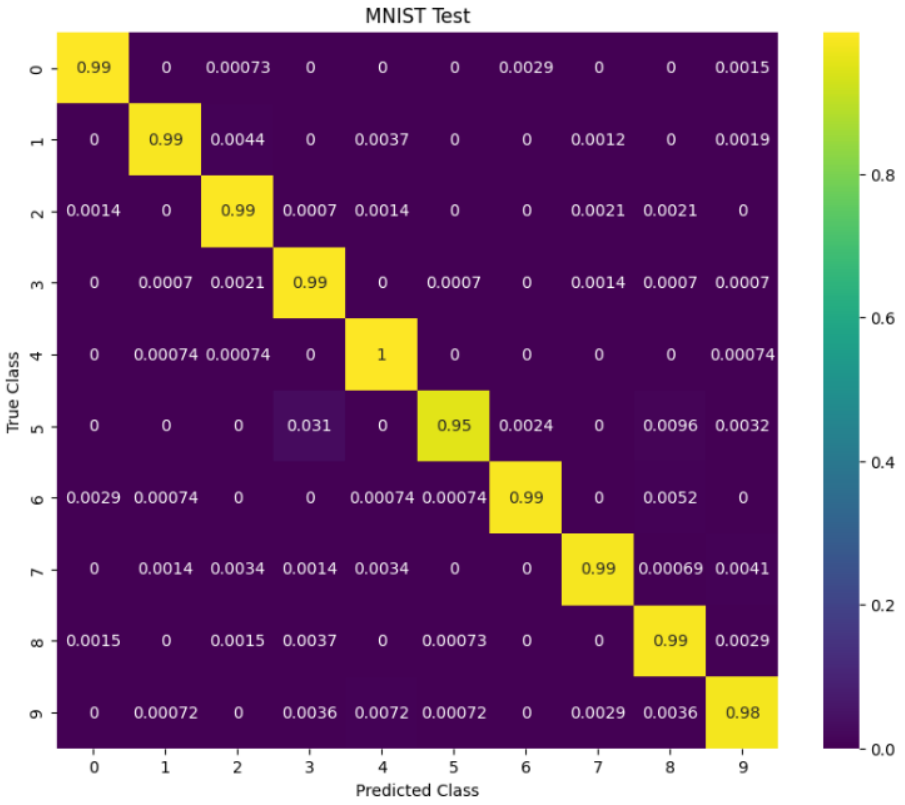


Figure 23: Confusion Matrix on MNIST Test (Unified Classifier).

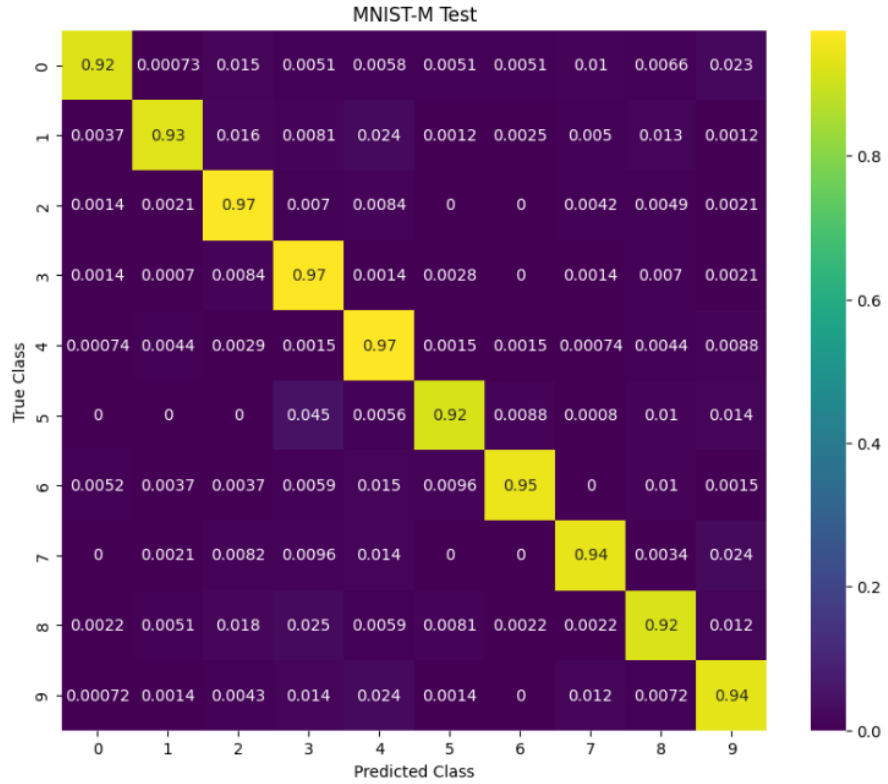


Figure 24: Confusion Matrix on MNIST-M Test (Unified Classifier).

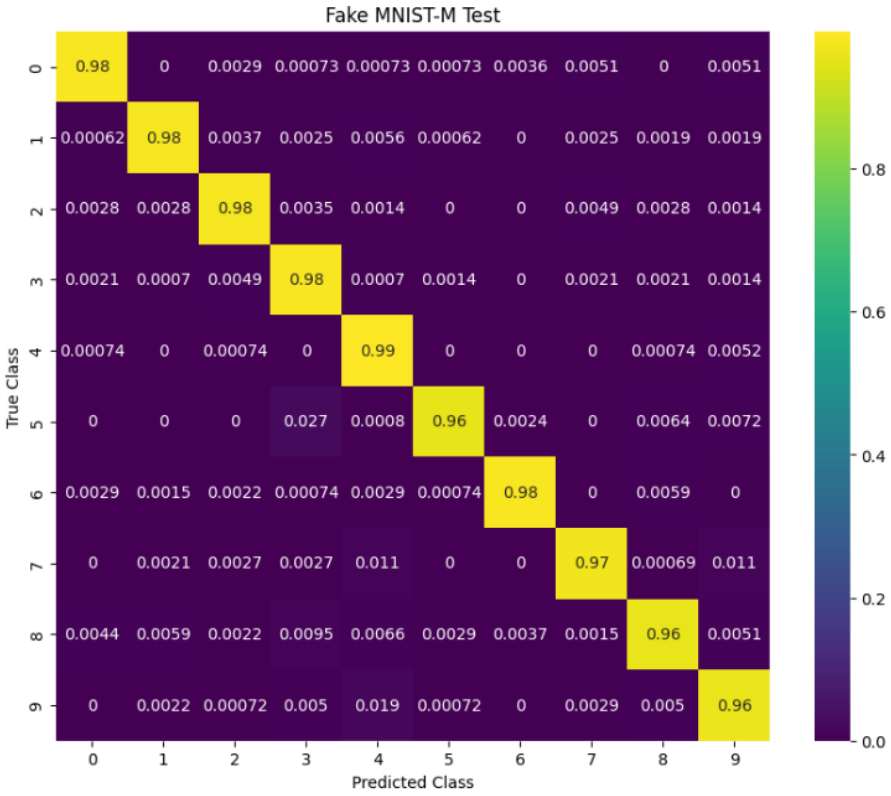


Figure 25: Confusion Matrix on Fake Test Set (Unified Classifier).

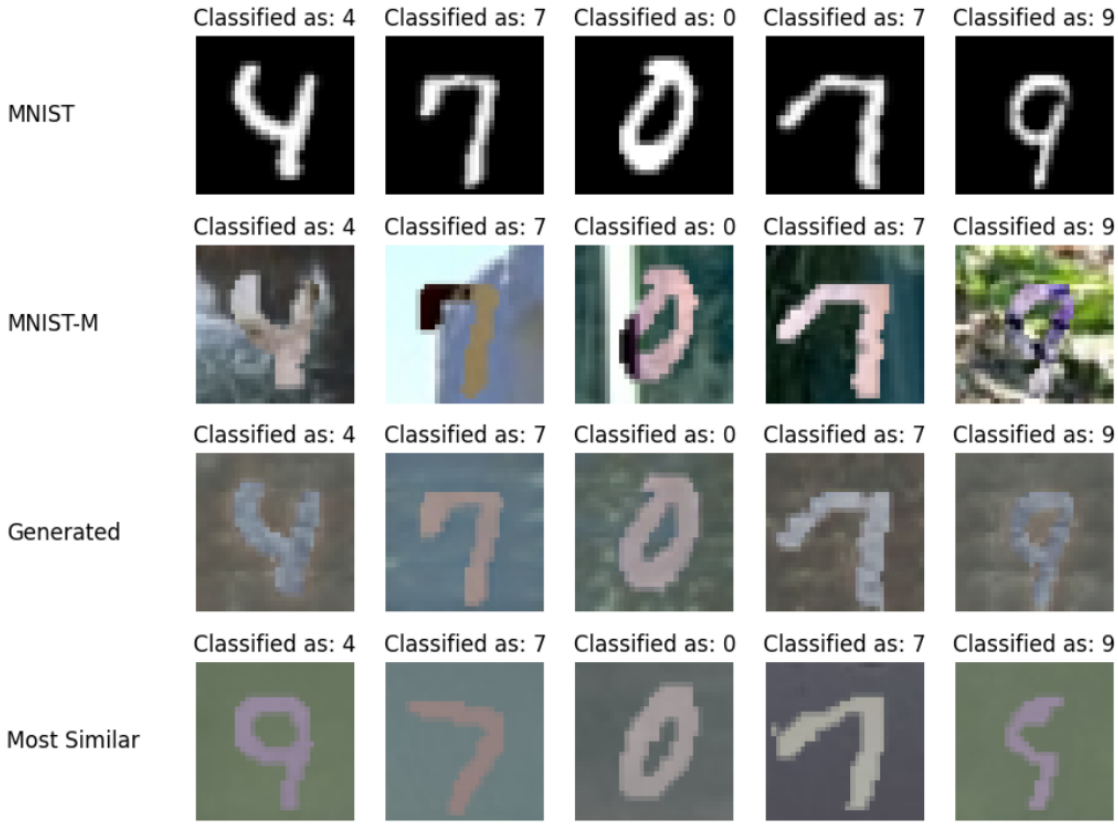


Figure 26: Qualitative results with the unified classifier.

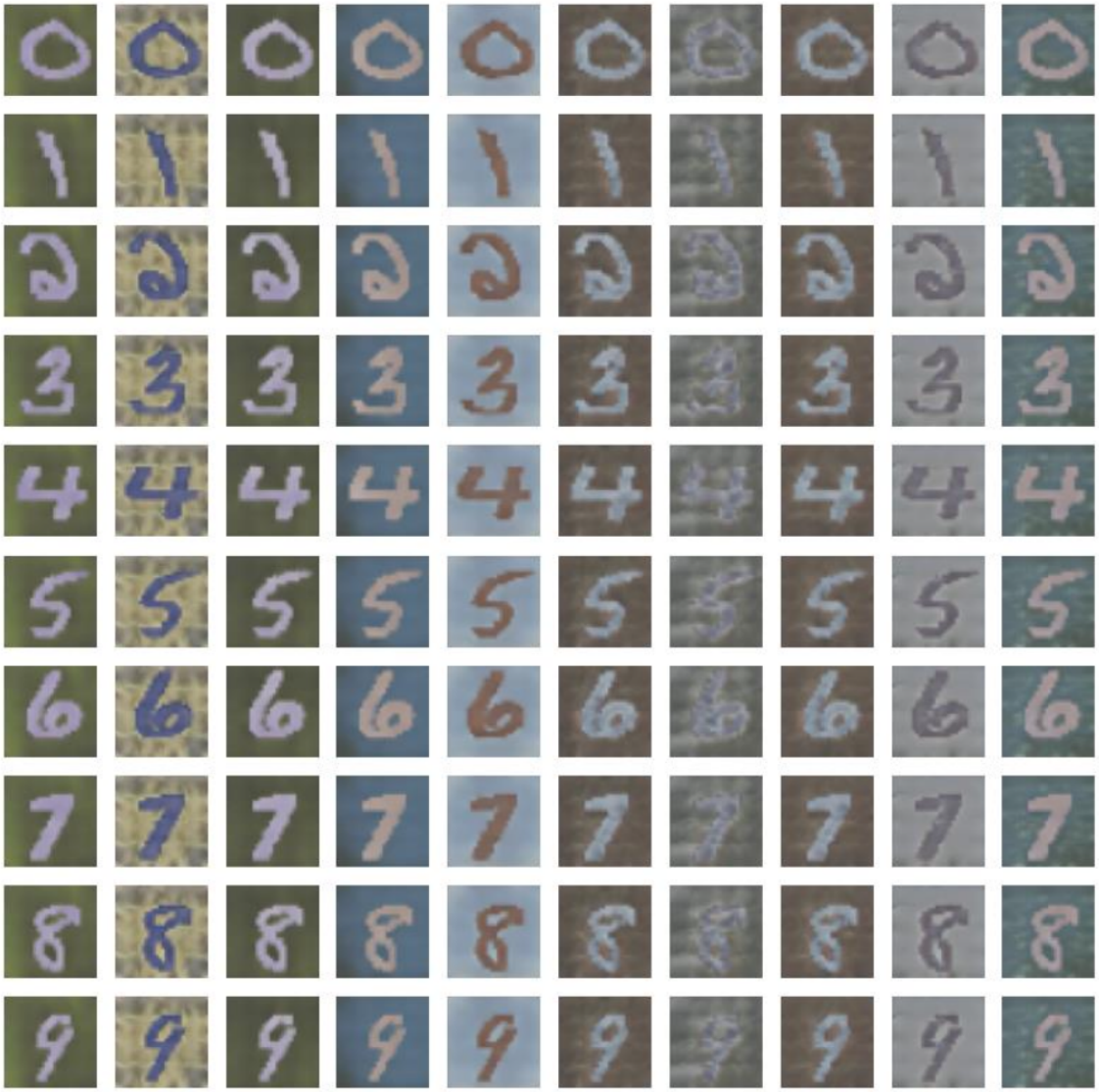


Figure 27: Generated samples for the same digit using 10 different noise vectors (Unified Classifier).

Analysis: The model with a single unified classifier also performs very well, achieving 94.2% accuracy on MNIST-M. This is only slightly lower than the 96.6% from the original architecture. This suggests that while having private initial layers is beneficial, a fully shared model can still effectively learn the domain-invariant features necessary for this task. The flexibility of the original design might be more critical for domains with a larger gap.

1.4 Conclusion

In this project, we successfully replicated the core findings of the paper on unsupervised domain adaptation. We demonstrated a significant domain gap between MNIST and MNIST-M and showed that a GAN-based pixel-level adaptation approach can effectively bridge this gap. The model learned to translate images from the source to the target

style while preserving semantic content, boosting the classifier's performance on the unlabeled target domain from 60% to over 96%. We analyzed the roles of the different model components, the importance of the training strategy, and the effect of architectural choices. This work highlights the power of adversarial learning for creating robust models that can generalize across different data distributions, a critical capability for real-world applications in fields like medical imaging and autonomous systems.

Part I

Reconstruction of Endoscopic Polyp Images with EndoVAE

2 Project Overview

This project focuses on reconstructing endoscopic polyp images using a Variational Autoencoder (VAE), based on the architecture proposed in the EndoVAE paper. The goal is to build a generative model capable of capturing the essential features of polyp images and reconstructing them. We use the Kvasir dataset instead of the original KID dataset.

3 Methodology

3.1 Data Preprocessing

1. **Data Selection:** We use "normal" images (from 'normal-z-line', 'normal-pylorus', 'normal-cecum') and "polyp" images from the Kvasir-SEG dataset.
2. **Preprocessing Steps:** All images are loaded, converted to RGB, and resized to 96x96 pixels using bilinear interpolation. Pixel values are normalized to the [0, 1] range. The processed images are saved into 'processed/normal' and 'processed/polyp' directories.

3.2 EndoVAE Architecture Design

The model is a VAE with an Encoder and a Decoder.

- **Encoder:** Consists of six convolutional layers (Conv2D + ReLU), followed by a flatten operation. The flattened output is passed through two separate fully connected (FC) layers to produce the mean (μ) and log-variance ($\log(\sigma^2)$) of the latent distribution. The latent dimension is set to 6.
- **Reparameterization Trick:** A latent vector z is sampled from $\mathcal{N}(\mu, \sigma^2)$ using the reparameterization trick: $z = \mu + \sigma \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$.
- **Decoder:** The latent vector z is first passed through an FC layer to expand it to a 256x6x6 tensor. This is followed by seven transposed convolutional layers (ConvTranspose2d + ReLU), with a final Sigmoid activation to produce the reconstructed 96x96x3 image.

3.3 Loss Function Definition

The total loss is a combination of reconstruction loss and the KL divergence:

$$\text{Total Loss} = \text{Reconstruction Loss} + 1.0 \times \text{KL Divergence}$$

- **Reconstruction Loss:** We use Binary Cross-Entropy (BCE) to measure the pixel-wise difference between the original and reconstructed images.
- **KL Divergence:** The standard KL divergence term for VAEs, which acts as a regularizer on the latent space, forcing it to be close to a standard normal distribution.

3.4 Training and Evaluation

3.4.1 Training

The model is trained on the "normal" images only, following the hyperparameters in section 4.A of the paper. The training loss is monitored over epochs.

3.4.2 Qualitative Evaluation

- **Generation:** We sample 9 random vectors from a standard normal distribution, pass them through the decoder, and visualize the 9 generated images.
- **Reconstruction:** We take 5 real polyp images (which the model has not seen during training), pass them through the full VAE, and display them side-by-side with their reconstructions.

3.4.3 Quantitative Evaluation

For 50 randomly selected polyp images, we calculate the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) between the original and reconstructed images. We report the mean and standard deviation of these metrics.

3.4.4 Result Analysis

- **Qualitative Analysis:** How realistic are the generated images? What details of the polyps are preserved or lost during reconstruction?
- **Quantitative Analysis:** What do the PSNR and SSIM values indicate about the reconstruction quality? If the values are low, what could be the cause (architecture, hyperparameters, preprocessing)?

3.4.5 Bonus Task

A simple CNN classifier is trained to distinguish between real and reconstructed polyp images. The accuracy and AUC are reported. This helps to analyze whether the reconstructed images are statistically similar to the real ones.