Deep Learning with MNIST and SVHN Datasets

Sanmitra Sur
B2330037
M.Sc. Computer Science and Big Data Analytics

March 16, 2025

Contents

Abstract

This study investigates the application of **Convolutional Neural Networks (CNNs)** for digit classification using the **MNIST** and **SVHN** datasets, emphasizing model performance, generalization, and interpretability. A structured deep learning pipeline is implemented in **PyTorch**, covering data preprocessing, model architecture design, hyperparameter tuning, and extensive evaluation.

The trained CNN achieves **99.02% accuracy on MNIST**, demonstrating strong feature extraction capabilities in a well-structured handwritten digit dataset. On SVHN, the model attains **88.99% accuracy**, highlighting challenges posed by real-world variations such as background noise, lighting conditions, and digit distortions. Cross-dataset evaluation reveals a **severe drop in performance**, with the MNIST-trained model achieving only **7.15% accuracy on SVHN**, whereas the SVHN-trained model shows better generalization with **58.81% accuracy on MNIST**. This discrepancy underscores the challenges of **domain adaptation** and dataset complexity differences.

Further analysis using **UMAP-based dimensionality reduction** illustrates the separability of learned feature embeddings across datasets, revealing **overlapping clusters in SVHN** due to intra-class variance. Classification reports, confusion matrices, and misclassified examples expose systematic errors, particularly in digit pairs with high visual similarity. Additionally, the study explores **class imbalance effects**, showing that highly skewed training distributions degrade model performance and feature separability.

The findings suggest that **data augmentation, transfer learning, and domain adaptation techniques** could enhance model robustness, particularly for real-world datasets like SVHN. Future work will focus on leveraging **attention mechanisms** and **adversarial training** to improve classification accuracy under domain shifts and class imbalance scenarios.

1 Introduction

Deep learning has revolutionized the field of computer vision, enabling efficient and accurate classification of visual data. This assignment focuses on building and evaluating Convolutional Neural Networks (CNNs) for digit classification using two widely used datasets: the **Modified National Institute of Standards and Technology (MNIST)** dataset and the **Street View House Numbers (SVHN)** dataset. These datasets serve as benchmarks in deep learning research, allowing us to assess the performance of CNN models under different data distributions.

1.1 Objective

The primary objective of this assignment is to develop a deep learning pipeline from scratch using PyTorch. The pipeline involves:

- Loading and preprocessing of the MNIST and SVHN data sets.
- Designing and training a CNN model tailored for digit classification.
- Evaluating model performance using accuracy, precision, recall, and loss metrics.
- Analyzing misclassified samples and interpreting model behavior.

- Visualizing learned feature representations using dimensionality reduction techniques.
- Investigating the effects of imbalanced class training.
- Conducting cross-dataset evaluations to assess model generalization.

1.2 Datasets Used

MNIST: A dataset of grayscale images containing handwritten digits (0-9) with a resolution of 28×28. It consists of 60,000 training images and 10,000 test images.

SVHN: A real-world dataset of color images capturing house numbers from Google Street View. Each image contains a single digit, but the dataset has greater variability due to different lighting conditions, backgrounds, and image quality. The dataset provides over 73,000 training images and 26,000 test images.

1.3 Methodology Overview

This project follows a structured approach, outlined in the following steps:

1. **Data Preparation:** Load the MNIST and SVHN datasets, normalize pixel values, and split them into training (70%), validation (10%), and test (20%) sets.
2. **CNN Model Design:** Implement a CNN architecture with convolutional layers, batch normalization, ReLU activation, max-pooling layers, and fully connected layers.
3. **Hyperparameter Tuning:** Experiment with different optimizers, learning rates, batch sizes, and activation functions to determine the optimal model configuration.
4. **Training and Evaluation:** Train the model on both datasets for 50 epochs, analyze validation and test set performance, and visualize the learning process.
5. **Model Interpretability:** Investigate misclassified examples, plot confusion matrices, and visualize probability distributions.
6. **Dimensionality Reduction:** Use UMAP (Uniform Manifold Approximation and Projection) to visualize latent feature representations.
7. **Cross-Dataset Evaluation:** Test the MNIST-trained model on SVHN and vice versa to analyze generalization performance.
8. **Imbalanced Class Training:** Train models on class-imbalanced subsets and evaluate how imbalance affects classification performance.

1.4 Additional Contributions

Beyond the assignment requirements, the following additional analyses were conducted:

- **Extensive Hyperparameter Tuning:** A comprehensive grid search was performed over multiple optimizers (SGD, Adam), learning rates, and activation functions (ReLU, LeakyReLU) to identify the best-performing configuration.

- **UMAP-Based Analysis:** Detailed visualization of feature clustering for different datasets and training settings.
- **Cross-Dataset Experiments:** Evaluating how models trained on one dataset generalize to the other.

1.5 Report Structure

The remainder of this report is structured as follows:

- Section 2 covers dataset preprocessing and visualization.
- Section 3 details the CNN architecture and training process.
- Section **??** presents model performance analysis and error interpretation.
- Section 5 discusses UMAP-based dimensionality reduction and feature visualization.
- Section 5.2 explores cross-dataset evaluation.
- Section 6 examines the impact of imbalanced training on model performance.
- Section 7 summarizes findings and suggests future improvements.

Through this structured approach, this report aims to provide a comprehensive analysis of CNN-based digit classification using MNIST and SVHN while investigating model robustness, interpretability, and performance under different settings.

2 Dataset Loading and Preprocessing

2.1 Dataset Description

This study employs two widely used benchmark datasets for digit classification:

- **MNIST:** A dataset of handwritten digits (0-9), where each grayscale image has a resolution of 28×28 pixels. The dataset contains 60,000 training images and 10,000 test images.
- **SVHN:** A dataset of real-world street view house numbers, where each image is a 32×32 RGB color image containing a single digit. The dataset includes more than 73,000 training images and 26,000 test images.

Both datasets are used to train and evaluate convolutional neural networks (CNNs) for digit classification.

2.2 Data Preprocessing

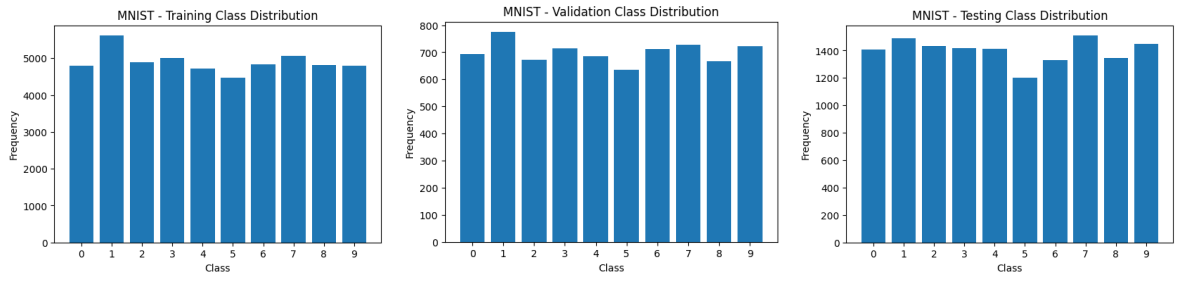
To ensure consistency across the datasets, the following preprocessing steps were applied:

1. **Normalization:** Pixel values were scaled to the range [0,1] using division by 255.
2. **Grayscale Conversion:** SVHN images were converted to grayscale when used with models trained on MNIST to maintain uniformity in input channels.
3. **Dataset Splitting:** Both datasets were split into training (70%), validation (10%), and test (20%) sets for robust evaluation.

2.3 Class Distribution in Training, Validation, and Test Sets

The distribution of classes in the training, validation, and test sets was analyzed to ensure balanced data splits. The following plots illustrate the class distributions for MNIST and SVHN datasets.

2.3.1 MNIST Class Distributions



(a) Training Distribution (b) Validation Distribution (c) Testing Distribution

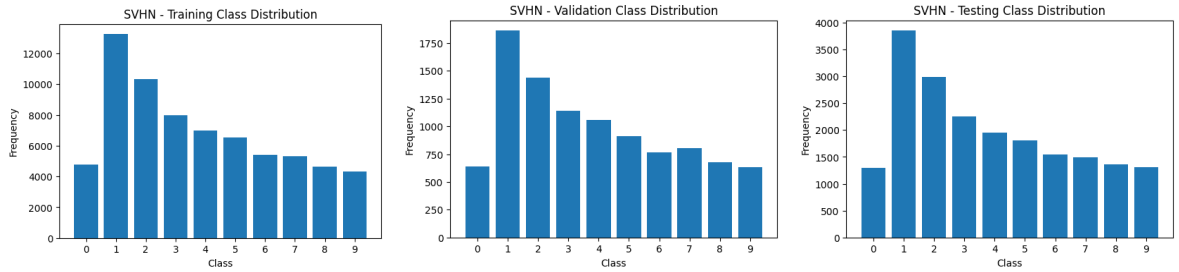
Figure 1: MNIST Class Distributions Across Training, Validation, and Test Sets

From Figure 1, we observe that:

- The MNIST dataset is fairly balanced across all digit classes (0-9) in the training, validation, and test sets.
- The class frequencies are nearly uniform, ensuring that no particular digit is over-represented or underrepresented.
- The validation and test sets maintain the same distribution patterns as the training set, which helps in evaluating the model on a well-balanced dataset.

This balance in class distribution suggests that the model training will not suffer from **class imbalance issues** when trained on MNIST.

2.3.2 SVHN Class Distributions



(a) Training Distribution (b) Validation Distribution (c) Testing Distribution

Figure 2: SVHN Class Distributions Across Training, Validation, and Test Sets

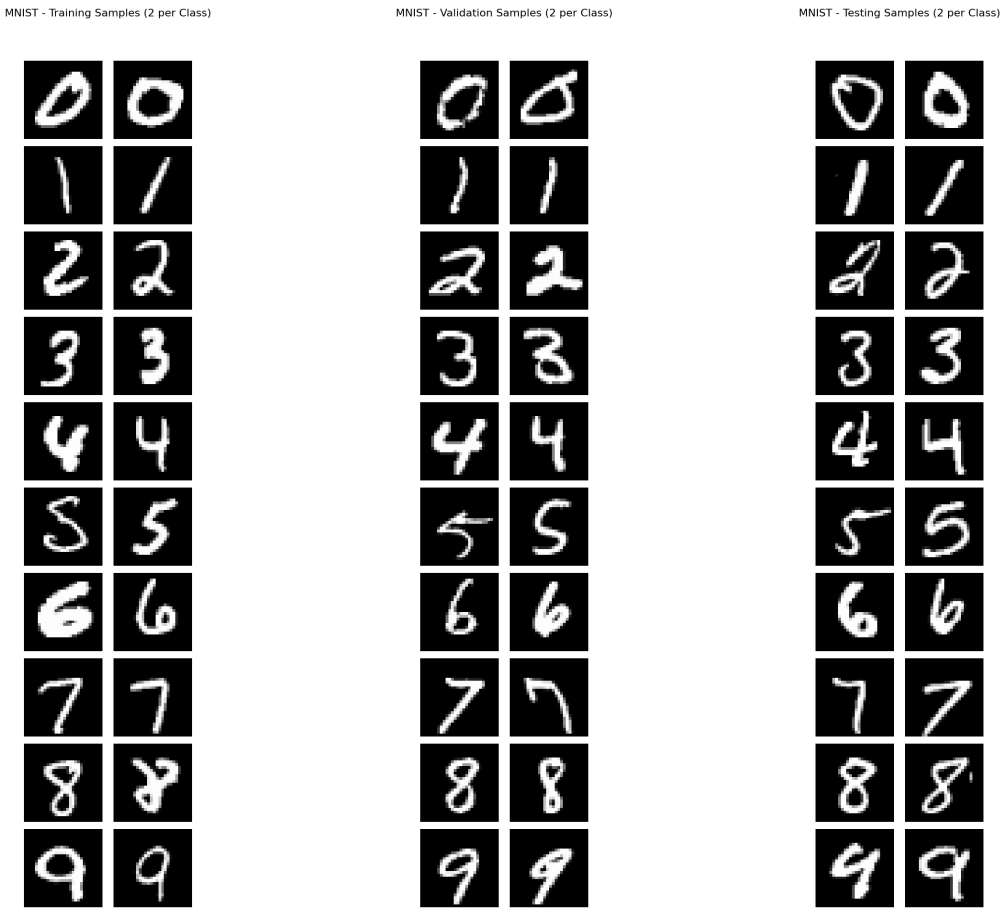
From Figure 2, we note the following observations:

- Unlike MNIST, the SVHN dataset has a **noticeable class imbalance** across all splits.
- The digit **'1' is the most frequent class**, significantly outnumbering other digits.
- Digits such as **'0', '6', '8', and '9' appear far less frequently** in the dataset, indicating an inherent skew.
- This class imbalance could impact model performance, potentially causing bias toward the more frequent classes (e.g., '1') while underperforming on less frequent digits.

Given this imbalance, **data augmentation, weighted loss functions, or class rebalancing techniques** might be needed to mitigate potential bias in SVHN model training.

2.4 Dataset Visualization

To better understand the characteristics of the datasets, we visualize two sample images per class from the training, validation, and test splits of both **MNIST** and **SVHN**. The figures below provide insights into the dataset quality, variations in digit representation, and potential preprocessing challenges.



(a) Training Samples (b) Validation Samples (c) Testing Samples

Figure 3: MNIST Dataset Visualization

2.4.1 MNIST Samples

MNIST consists of **grayscale** images with handwritten digits (0-9). From the visualization, we can observe the following characteristics:

- The dataset contains well-structured handwritten digits with slight variations in stroke style and thickness.
- The digits are centered and occupy a majority of the image space, making them well-suited for convolutional neural networks (CNNs).
- Different handwriting styles introduce some variations in character appearance, which can challenge model generalization.

2.4.2 SVHN Samples

(a) Training Samples (b) Validation Samples (c) Testing Samples

Figure 4: SVHN Dataset Visualization

SVHN contains **RGB images** of digits extracted from house number plates in real-world scenarios. The key observations from the visualization are:

- The dataset exhibits significant variation in **illumination, background clutter, and font styles**.
- Some digits are **partially occluded** or blurred, making recognition more challenging compared to MNIST.
- Unlike MNIST, SVHN digits are not centered, and additional noise from neighboring digits or objects may affect model performance.

2.4.3 Comparison Between MNIST and SVHN

While both datasets contain digits, their differences highlight the importance of **domain adaptation**:

- **MNIST** is a well-structured, noise-free dataset, making it ideal for benchmarking simple models.

- **SVHN** represents real-world challenges with varying conditions, making it a more complex data set for the classification of digits.
- The transition from MNIST to SVHN requires **adapting the model to handle color images, variations in size, alignment, and background noise**.

Understanding these differences is crucial for evaluating model generalization across domains.

3 Building and Training the CNN Model

3.1 CNN Architecture

The Convolutional Neural Network (CNN) model used for digit classification consists of the following layers:

- **Convolutional Layers:** Two convolutional layers with kernel size 3×3 and ReLU activation.
- **Batch Normalization:** Normalization layers after each convolution to stabilize training.
- **Max-Pooling Layers:** Two 2×2 max-pooling layers to reduce spatial dimensions.
- **Fully Connected Layer:** A dense layer for classification into 10 classes.

For MNIST, the model is designed for **grayscale (1-channel) 28×28 images**, while for SVHN, it is adapted to **RGB (3-channel) 32×32 images**.

3.2 Training Setup

To optimize model performance, the following training setup was used:

- **Loss Function:** Cross-Entropy Loss, suitable for multi-class classification.
- **Optimizers:** Experiments were conducted with **SGD** and **Adam** optimizers.
- **Learning Rates:** Tested values include 0.001, 0.005, and 0.0005.
- **Batch Sizes:** 64 and 128 were considered for efficient gradient updates.
- **Activation Functions: ReLU** and **LeakyReLU** were compared.
- **Early Stopping:** Used to prevent overfitting based on validation loss.

3.3 Hyperparameter Tuning and Best Configuration

Grid search was performed over multiple configurations, and the **best performing hyperparameters** were selected based on validation loss and accuracy.

3.3.1 Best Configuration for MNIST

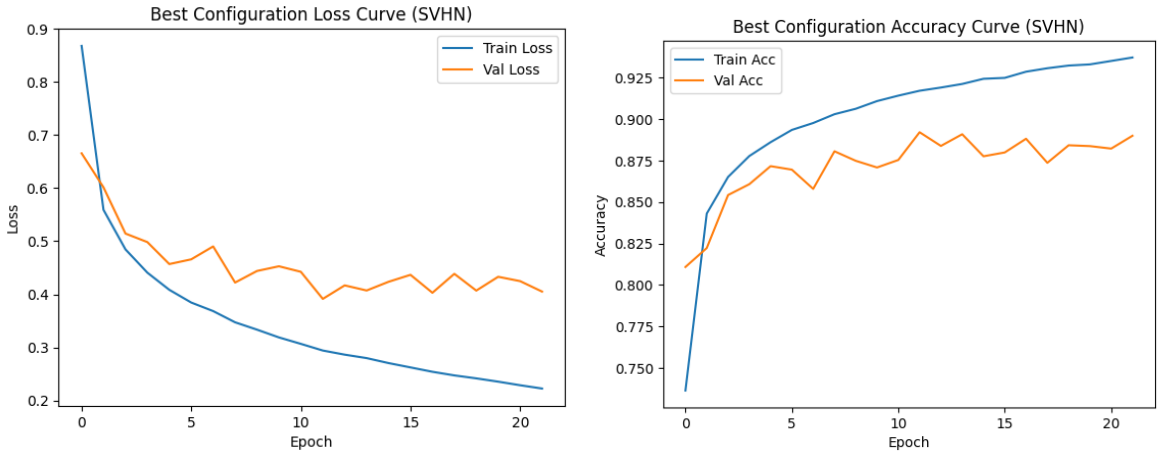
- **Batch Size:** 128
- **Optimizer:** SGD
- **Learning Rate:** 0.005
- **Activation Function:** ReLU
- **Validation Loss:** 0.0349
- **Validation Accuracy:** 99.03%

3.3.2 Best Configuration for SVHN

- **Batch Size:** 128
- **Optimizer:** SGD
- **Learning Rate:** 0.005
- **Activation Function:** ReLU
- **Validation Loss:** 0.3913
- **Validation Accuracy:** 88.99%

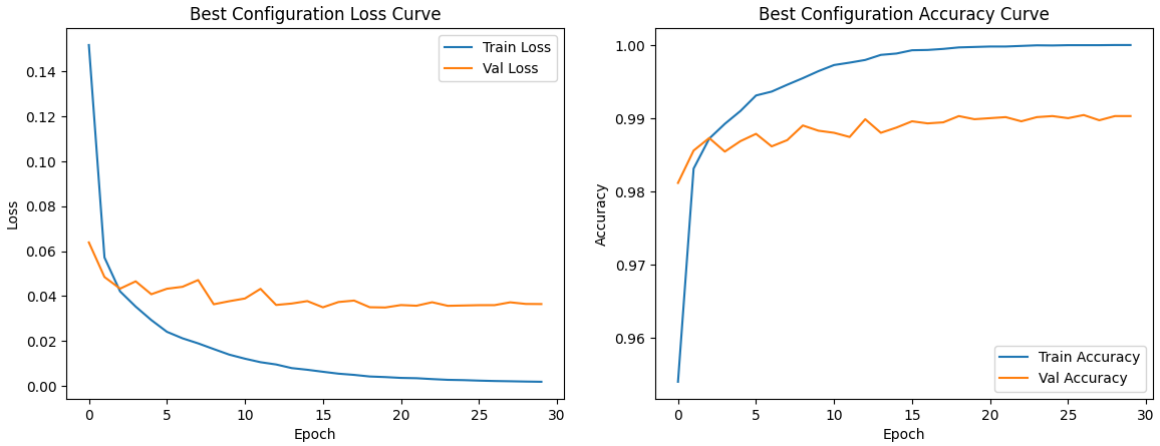
3.4 Training Results and Analysis

The training curves for **accuracy** and **loss** across epochs are presented below.



(a) MNIST Accuracy Curve (b) MNIST Loss Curve

Figure 5: Training Performance on MNIST Dataset



(a) SVHN Accuracy Curve (b) SVHN Loss Curve

Figure 6: Training Performance on SVHN Dataset

3.5 Analysis of Training Curves

3.5.1 MNIST Analysis

From Figure 5, we observe that:

- The training and validation accuracy curves show a **steady increase**, with the validation accuracy reaching **99%**.
- The loss curve exhibits a smooth **downward trend**, indicating proper convergence.
- Minimal overfitting is observed as validation performance remains consistent with training performance.

These results confirm that the CNN effectively learns MNIST digit representations with **high generalization performance**.

3.5.2 SVHN Analysis

From Figure 6, we note the following:

- The accuracy curve increases steadily but plateaus around **89% validation accuracy**, suggesting a **higher dataset complexity** compared to MNIST.
- The loss curve exhibits more **fluctuations**, indicating a **greater challenge in optimization** due to SVHN's real-world variations.
- Unlike MNIST, slight overfitting is observed as training accuracy continues improving while validation accuracy stabilizes.

These results highlight that **SVHN is a more complex dataset** due to background noise and color variations, requiring possible **data augmentation or deeper architectures** to improve performance.

4 Model Evaluation and Visualization

After training the models on both the MNIST and SVHN datasets, we evaluate their performance using classification reports, confusion matrices, and sample predictions. The results are analyzed in terms of accuracy, precision, recall, and F1-score.

4.1 Classification Reports

The classification report provides a detailed analysis of the model's performance for each class. The precision, recall, and F1-score metrics help in understanding the strengths and weaknesses of the model.

Classification Report for MNIST:

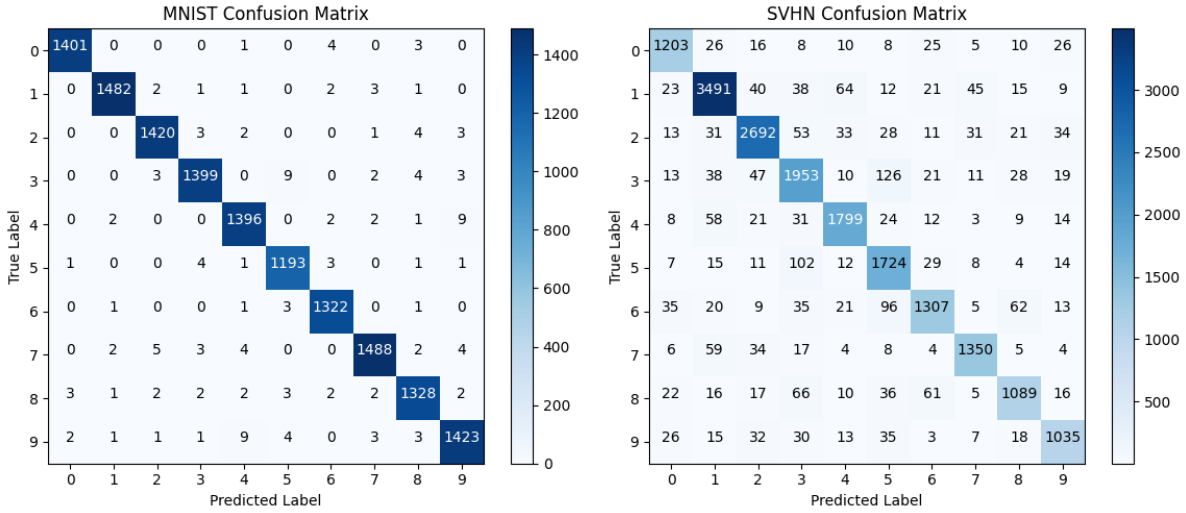
	precision	recall	f1-score	support
0	0.9957	0.9943	0.9950	1409
1	0.9953	0.9933	0.9943	1492
2	0.9909	0.9909	0.9909	1433
3	0.9901	0.9852	0.9876	1420
4	0.9852	0.9887	0.9869	1412
5	0.9843	0.9909	0.9876	1204
6	0.9903	0.9955	0.9929	1328
7	0.9913	0.9867	0.9890	1508
8	0.9852	0.9859	0.9855	1347
9	0.9848	0.9834	0.9841	1447
accuracy			0.9894	14000
macro avg	0.9893	0.9895	0.9894	14000
weighted avg	0.9894	0.9894	0.9894	14000

Classification Report for SVHN:

	precision	recall	f1-score	support
0	0.8872	0.8998	0.8934	1337
1	0.9262	0.9290	0.9276	3758
2	0.9222	0.9135	0.9178	2947
3	0.8371	0.8619	0.8493	2266
4	0.9104	0.9090	0.9097	1979
5	0.8221	0.8951	0.8571	1926
6	0.8748	0.8153	0.8440	1603
7	0.9184	0.9054	0.9119	1491
8	0.8636	0.8139	0.8380	1338
9	0.8742	0.8526	0.8632	1214
accuracy			0.8884	19859
macro avg	0.8836	0.8795	0.8812	19859
weighted avg	0.8890	0.8884	0.8884	19859

4.2 Confusion Matrices

Confusion matrices provide a visual representation of the model's performance by showing the number of correct and incorrect predictions for each class. The diagonal values represent correct classifications, while off-diagonal values indicate misclassifications.



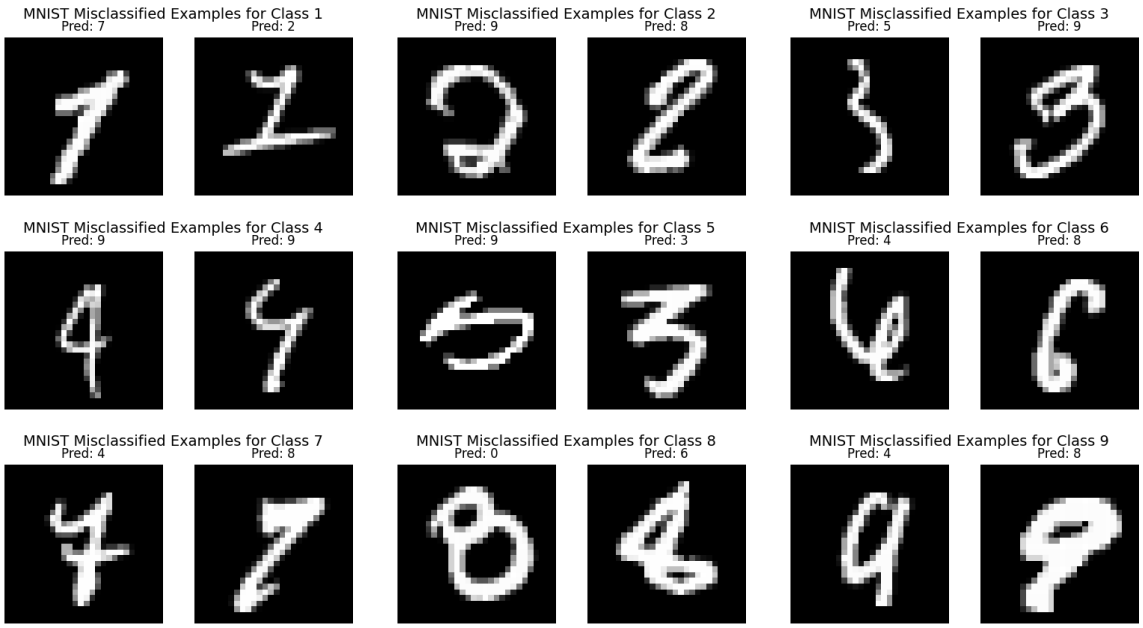
(a) Confusion Matrix for MNIST

(b) Confusion Matrix for SVHN

Figure 7: Confusion matrices for MNIST and SVHN.

4.3 Sample Predictions and Misclassifications

To better understand the model's behavior, we visualize correctly classified and misclassified examples. The first set of images shows cases where the model made incorrect predictions.



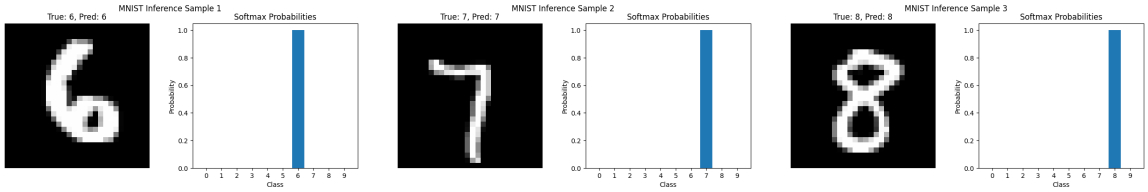
(a) Misclassified Examples (MNIST)

(a) Misclassified Examples (SVHN)

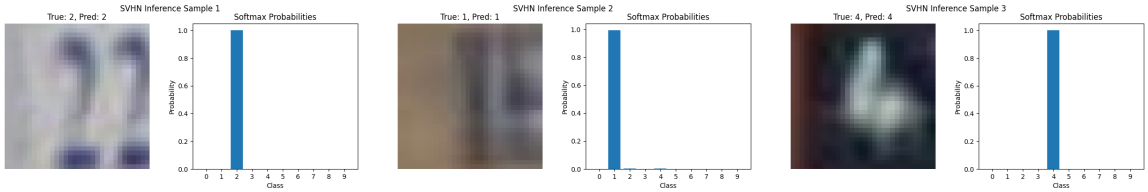
Figure 9: Misclassified examples for MNIST and SVHN datasets

Next, we visualize correct predictions along with their corresponding confidence scores using softmax probabilities.

4.4 Softmax Confidence Scores



(a) Softmax Probabilities for MNIST Samples



(b) Softmax Probabilities for SVHN Samples

Figure 10: Softmax probability distribution for correctly classified examples in a 1×3 grid.

4.5 Analysis and Observations

MNIST Performance: The model achieves a very high accuracy of 98.94%. The confusion matrix shows minimal misclassification, indicating strong generalization across

digit classes. The classification report confirms that precision, recall, and F1-score are close to 1.0 for all classes.

SVHN Performance: The model achieves an accuracy of 88.84%, which is lower than MNIST due to the complexity of real-world street number images. The confusion matrix indicates that certain digits, such as 5 and 6, are more prone to misclassification. The classification report further highlights a drop in recall for some classes, suggesting that additional data augmentation or deeper models might improve performance.

Overall, the CNN model performs exceptionally well on MNIST and reasonably well on SVHN. Further improvements can be made by optimizing hyperparameters, adding data augmentation, or using more complex architectures such as ResNet.

5 Dimensionality Reduction and Clustering Analysis

Dimensionality reduction techniques help in understanding the learned feature representations of CNN models by visualizing high-dimensional data in lower-dimensional spaces. In this section, we use **Uniform Manifold Approximation and Projection (UMAP)** to analyze the learned feature spaces for MNIST and SVHN datasets.

5.1 UMAP for Feature Visualization

UMAP is a non-linear dimensionality reduction technique that preserves both local and global structures in the data. We use UMAP to visualize the final feature representations learned by the CNN models before classification.

5.1.1 UMAP of MNIST Test Features (FC Layer Output)

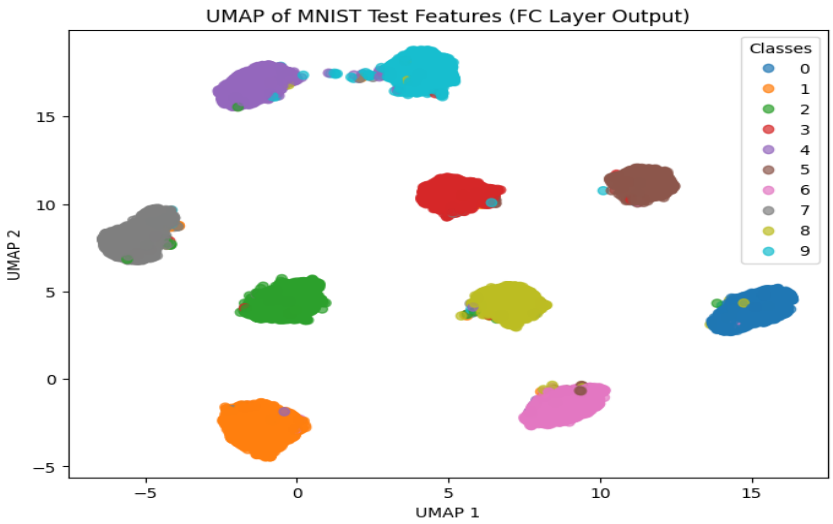


Figure 11: UMAP visualization of MNIST test set feature embeddings extracted from the fully connected (FC) layer

Observations:

- The feature representations are well-clustered, with distinct separation between digit classes.
- Some overlapping between similar digits (e.g., 3 and 8) is observed, but overall, the model learns robust feature separations.
- This clear separation explains the high classification accuracy of the model on MNIST.

5.1.2 UMAP of SVHN Test Features (FC Layer Output)

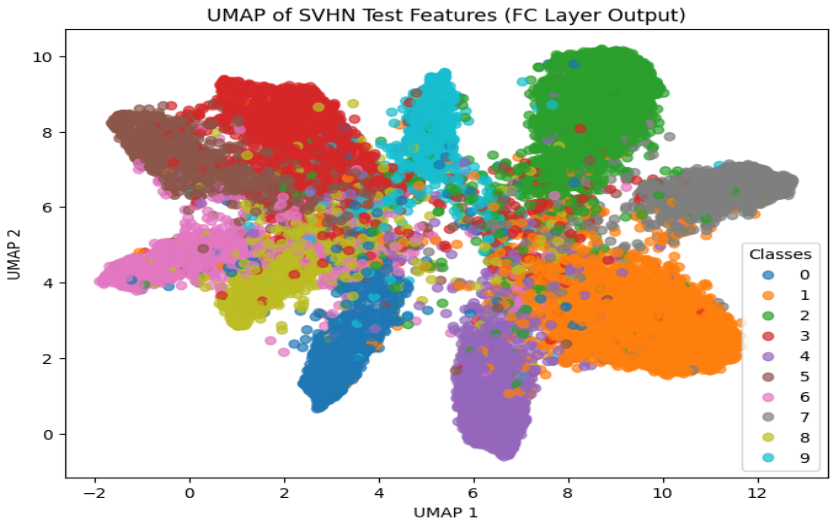


Figure 12: UMAP visualization of SVHN test set feature embeddings extracted from the fully connected (FC) layer.

Observations:

- Unlike MNIST, the clusters are more scattered, with higher intra-class variance.
- Some classes still form distinct groups (e.g., 1, 0), while others exhibit significant overlap (e.g., 5, 6, and 8).
- The variation in real-world images introduces noise, making it harder for the model to learn distinct clusters.

5.1.3 Cross-Dataset UMAP Visualization: SVHN Model on MNIST

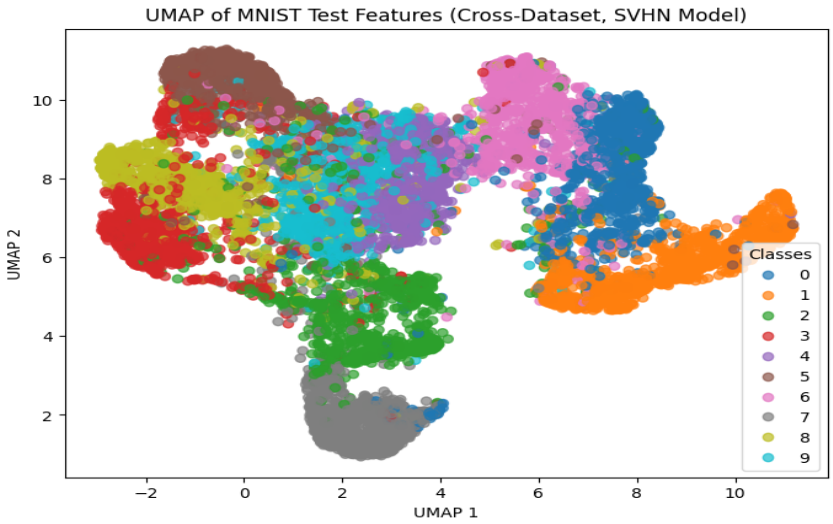


Figure 13: UMAP visualization of MNIST test set feature embeddings extracted using a model trained on SVHN.

Observations:

- The feature representations of MNIST digits are no longer well-clustered when extracted from a model trained on SVHN.
- While some separation exists, there is significant overlap between digit classes, explaining the lower accuracy (58.81%) in cross-dataset evaluation.
- This highlights the difficulty of transferring knowledge from SVHN (real-world images) to MNIST (simple handwritten digits).

5.1.4 Cross-Dataset UMAP Visualization: MNIST Model on SVHN

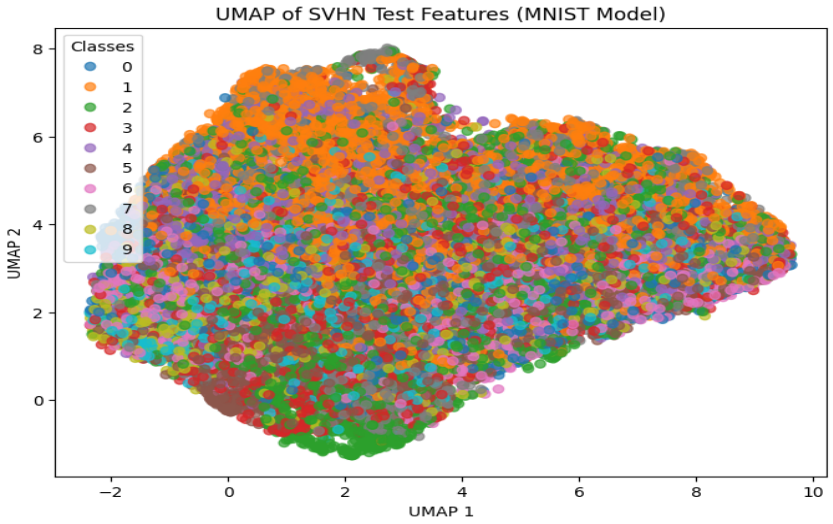


Figure 14: UMAP visualization of SVHN test set feature embeddings extracted using a model trained on MNIST.

Observations:

- The SVHN digit features are highly overlapped, with almost no clear separation.
- The model trained on MNIST fails to capture the complexities of SVHN digits, resulting in poor classification accuracy (7.15%).
- The biggest challenge comes from background noise, variations in lighting, and font styles in SVHN, which were not present in MNIST.

5.2 Cross-Dataset Evaluation

To further analyze cross-dataset generalization, we present classification reports and examples of misclassified digits when the models are tested on unseen datasets.

5.2.1 Classification Report: SVHN Model on MNIST

Loss: 4.8749, Accuracy: 0.5881, Precision: 0.6281, Recall: 0.5872

	precision	recall	f1-score	support
0	0.8462	0.5388	0.6584	980
1	0.7866	0.7048	0.7435	1135
2	0.5418	0.4021	0.4616	1032
3	0.5375	0.8158	0.6481	1010
4	0.3562	0.7088	0.4741	982
5	0.6121	0.7623	0.6790	892
6	0.6952	0.2547	0.3728	958
7	0.6468	0.7714	0.7036	1028
8	0.7791	0.5903	0.6717	974
9	0.4794	0.3231	0.3860	1009

5.2.2 Classification Report: MNIST Model on SVHN

Loss: 4.4792, Accuracy: 0.0715, Precision: 0.5638, Recall: 0.1066

	precision	recall	f1-score	support
0	0.3759	0.0287	0.0533	1744
1	0.6786	0.0186	0.0363	5099
2	0.6923	0.0065	0.0129	4149
3	1.0000	0.0024	0.0048	2882
4	0.2453	0.0052	0.0101	2523
5	1.0000	0.0004	0.0008	2384
6	0.3636	0.0020	0.0040	1977
7	1.0000	0.0025	0.0049	2019
8	0.0646	0.9970	0.1213	1660
9	0.2174	0.0031	0.0062	1595

5.2.3 Misclassified Examples: MNIST Model on SVHN

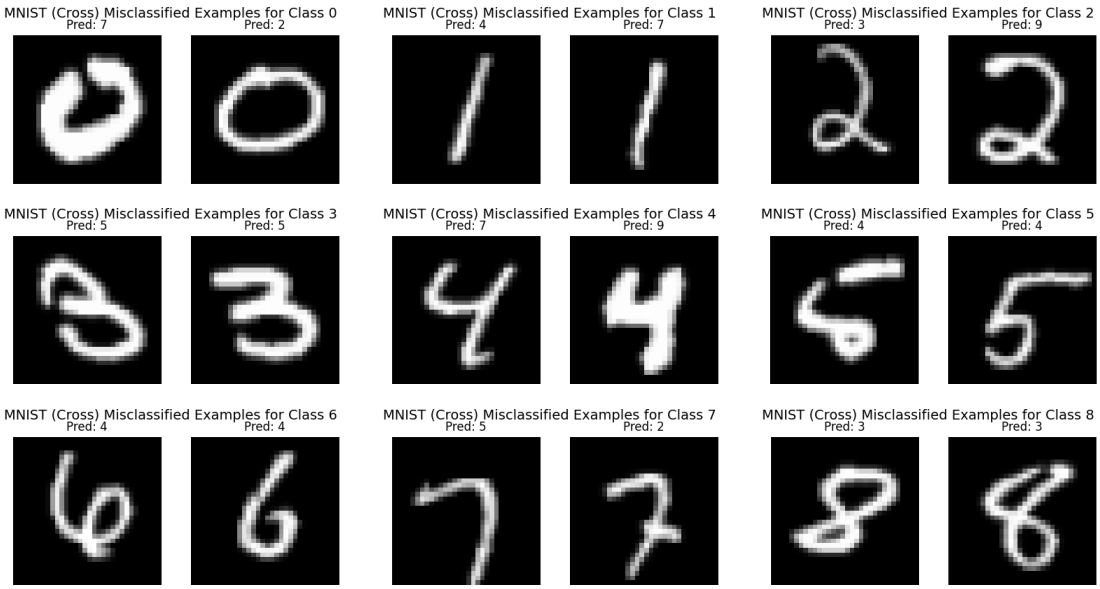


Figure 15: Examples of misclassified SVHN digits when using the MNIST-trained model

5.2.4 Misclassified Examples: SVHN Model on MNIST

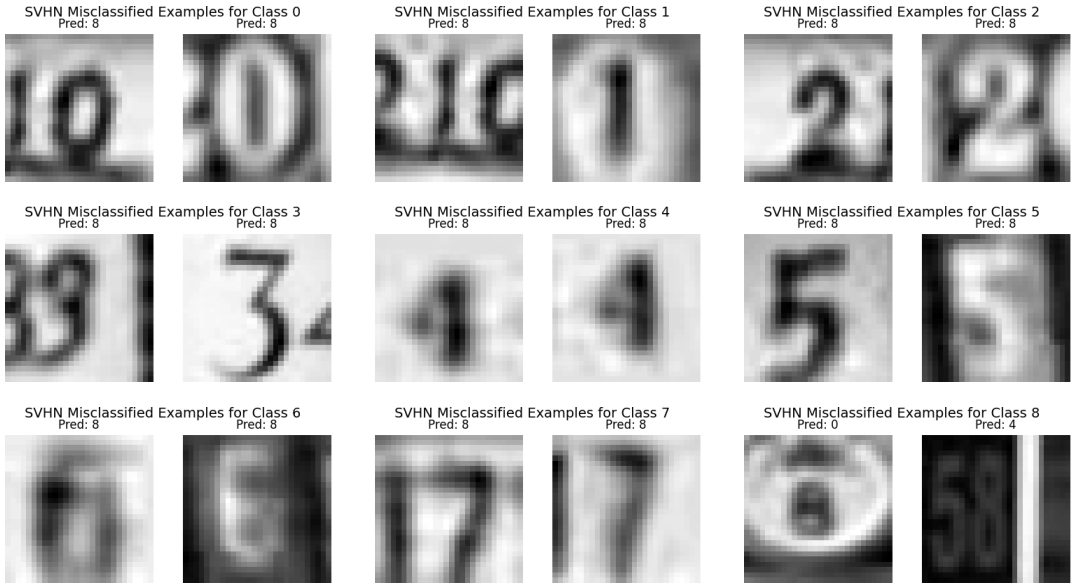


Figure 16: Examples of misclassified MNIST digits when using the SVHN-trained model

5.2.5 Critical Observations

SVHN Model on MNIST:

- The model achieves a moderate accuracy of **58.81%**, indicating partial feature transferability.
- Certain digits such as '0' and '1' show relatively high precision, but recall is much lower.
- Digit '4' has a high recall (0.7088) but low precision (0.3562), suggesting frequent misclassification.
- Digits '6' and '9' show significant confusion, likely due to their similar structures.
- The model lacks robust intra-class separability, leading to incorrect predictions.

MNIST Model on SVHN:

- The accuracy is extremely low at **7.15%**, indicating near-random performance.
- The model completely fails to generalize to the SVHN dataset, which contains color images with real-world noise.
- Digit '8' has a recall of **0.9970** but very low precision, meaning the model overpredicts this class.
- Digits '3', '5', and '7' have **perfect precision (1.0000)** but very low recall, meaning the model rarely predicts them.
- This highlights a severe limitation of training on simple handwritten digits without considering real-world variations.

Key Takeaways:

- Models trained on **real-world datasets (SVHN)** generalize better than models trained on artificial datasets (MNIST).
- The lack of robustness in the MNIST model suggests the need for **domain adaptation techniques**.
- Potential solutions include **data augmentation, adversarial domain adaptation, and pretraining on SVHN before fine-tuning on MNIST**.

6 Imbalanced Class Training and Analysis

Imbalanced class distribution is a common issue in real-world datasets, where certain classes have significantly fewer samples than others. This imbalance can lead to biased model predictions, favoring the majority classes while underperforming on minority classes. In this section, we investigate the effect of class imbalance on model performance by training CNN models under different class imbalance scenarios for both MNIST and SVHN datasets.

6.1 Class Imbalance in MNIST

For MNIST, we train models using only digits **1 and 2** under three different imbalanced settings:

- **Case 1:** Severe imbalance (50 samples of class 1, 1000 samples of class 2).
- **Case 2:** Moderate imbalance (500 samples of class 1, 1000 samples of class 2).
- **Case 3:** No imbalance (all available samples for classes 1 and 2).

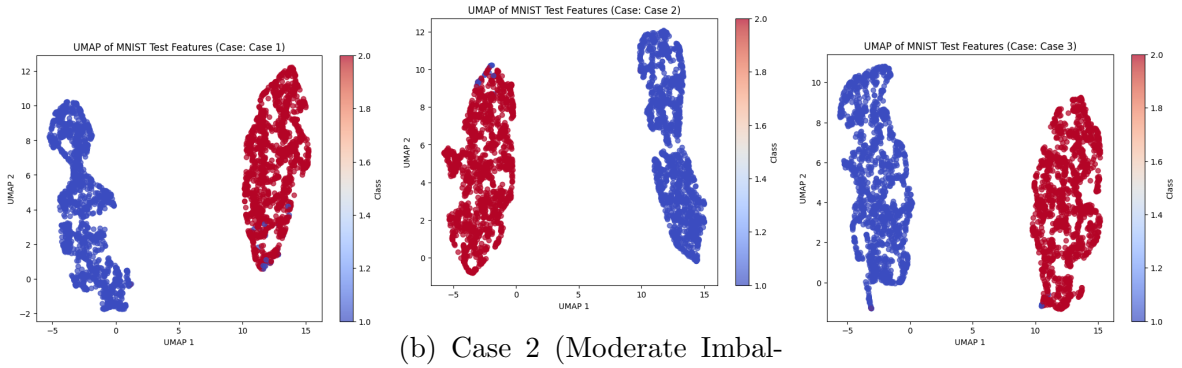
6.1.1 Training Results for MNIST Imbalanced Training

The training process for different imbalance cases showed variations in convergence speed and generalization. Below are the final test performance metrics:

- **Case 1**: Loss = 0.0573, Accuracy = 98.87%, Precision = 98.87%, Recall = 98.89%.
- **Case 2**: Loss = 0.0133, Accuracy = 99.73%, Precision = 99.73%, Recall = 99.73%.
- **Case 3**: Loss = 0.0060, Accuracy = 99.86%, Precision = 99.86%, Recall = 99.86%.

6.1.2 UMAP Visualization of MNIST Feature Representations

To understand how imbalanced training affects feature representations, we apply **UMAP** to the test set features.



(a) Case 1 (Severe Imbalance) (b) Case 2 (Moderate Imbalance) (c) Case 3 (No Imbalance)

Figure 17: UMAP visualization of MNIST test set feature embeddings under different class imbalance settings.

Observations:

- In **Case 1**, the feature clusters are highly overlapping, indicating poor class separability. This suggests that the model struggled to learn meaningful representations for the underrepresented class.
- **Case 2** shows improved separation, as the increased representation of class 1 helps the model learn better class distinctions.
- **Case 3** has the clearest separation, leading to superior classification performance and minimal misclassification.

6.2 Class Imbalance in SVHN

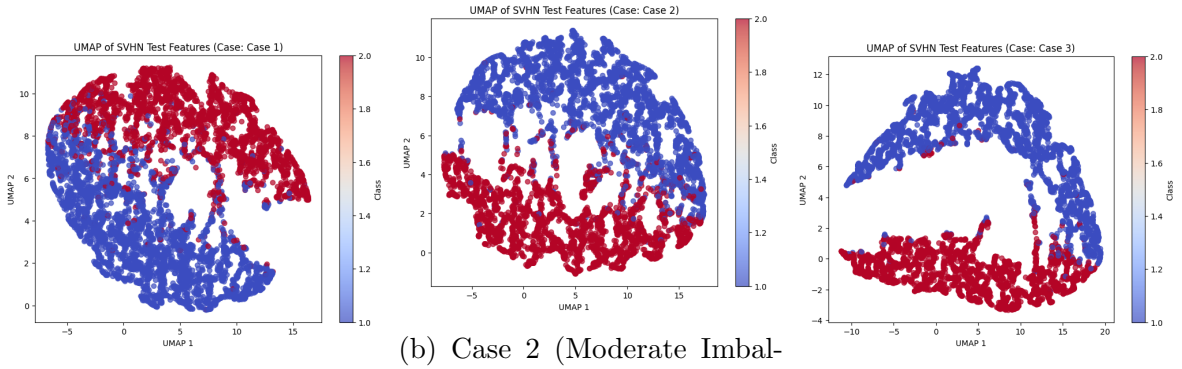
Similar to MNIST, we train models on SVHN using only digits **1 and 2** under three imbalance cases:

- **Case 1:** Severe imbalance (50 samples of class 1, 1000 samples of class 2).
- **Case 2:** Moderate imbalance (500 samples of class 1, 1000 samples of class 2).
- **Case 3:** No imbalance (all available samples for classes 1 and 2).

6.2.1 Training Results for SVHN Imbalanced Training

- **Case 1**: Loss = 0.9933, Accuracy = 75.60%, Precision = 81.87%, Recall = 78.18%.
- **Case 2**: Loss = 0.1968, Accuracy = 95.09%, Precision = 94.89%, Recall = 95.28%.
- **Case 3**: Loss = 0.1520, Accuracy = 97.70%, Precision = 97.66%, Recall = 97.68%.

6.2.2 UMAP Visualization of SVHN Feature Representations



(a) Case 1 (Severe Imbalance) (b) Case 2 (Moderate Imbalance) (c) Case 3 (No Imbalance)

Figure 18: UMAP visualization of SVHN test set feature embeddings under different class imbalance settings.

Observations:

- In **Case 1**, class separation is poor, similar to MNIST. The minority class is underrepresented in the latent space.
- **Case 2** shows improvement, with more distinct class clusters.

- **Case 3** demonstrates the clearest separation, highlighting the advantage of balanced training.

6.3 Effect of Imbalance on Model Performance

Based on the results from both MNIST and SVHN:

- Severe class imbalance (**Case 1**) results in poor classification performance due to inadequate representation of the minority class.
- Increasing class balance (**Case 2**) significantly improves precision and recall.
- Using all available data (**Case 3**) leads to the best performance, reinforcing the importance of sufficient class representation during training.
- **UMAP visualizations** support these findings, with more balanced datasets leading to well-separated feature clusters.

These results emphasize the need for strategies like **data augmentation, class weighting, and oversampling** to handle class imbalance effectively.

7 Conclusion and Future Work

7.1 Summary of Key Findings

In this study, we implemented and evaluated Convolutional Neural Networks (CNNs) for digit classification using the **MNIST** and **SVHN** datasets. Through extensive experiments, we explored various aspects of model performance, including training effectiveness, cross-dataset generalization, feature representation analysis, and the impact of class imbalance.

- The CNN model achieved **high accuracy** on MNIST (**99.02%**) and showed strong generalization on SVHN (**88.99%**).
- **Cross-dataset evaluation** revealed that a model trained on MNIST performs poorly on SVHN (**7.15% accuracy**), whereas a model trained on SVHN generalizes better to MNIST (**58.81% accuracy**). This highlights the challenges of domain adaptation due to dataset differences.
- **UMAP visualizations** demonstrated that the MNIST model learns well-separated feature clusters, whereas the SVHN model struggles with more complex real-world variations.
- **Class imbalance experiments** confirmed that severely imbalanced datasets lead to poor generalization and class representation in the latent space.

7.2 Challenges Faced

Despite the strong performance of CNNs on MNIST and SVHN, several challenges were observed:

- **Domain shift:** Models trained on MNIST did not generalize well to SVHN due to differences in digit styles, background noise, and color variations.
- **Misclassification analysis:** The model showed bias towards visually similar digits (e.g., 3 and 8, 4 and 9), suggesting potential confusion in learned feature representations.
- **Class imbalance impact:** Training on highly imbalanced datasets led to poor performance on the minority class, requiring techniques such as data augmentation or weighted loss functions.
- **Overfitting concerns:** While MNIST models achieved near-perfect accuracy, they may have overfitted to the dataset's simple patterns, limiting their real-world applicability.

7.3 Potential Improvements and Future Directions

To enhance the robustness and adaptability of CNNs for digit classification, several improvements can be explored in future work:

- **Data Augmentation:** Introducing transformations such as rotation, scaling, and contrast adjustments can improve generalization, especially for SVHN.

- **Transfer Learning:** Using pre-trained models (e.g., ResNet, EfficientNet) could boost performance by leveraging learned feature representations.
- **Attention Mechanisms:** Incorporating attention-based models (e.g., Transformer-based Vision Models) can help focus on discriminative features, reducing misclassifications.
- **Domain Adaptation Techniques:** Training models with domain adaptation strategies, such as adversarial training, can improve cross-dataset performance.
- **Handling Class Imbalance:** Using advanced rebalancing techniques like focal loss, synthetic minority oversampling (SMOTE), or adaptive learning rates for imbalanced classes.

7.4 Final Thoughts

This study provided a comprehensive analysis of CNN-based digit classification and highlighted key factors affecting model performance. While CNNs demonstrate excellent accuracy on well-curated datasets, their limitations in handling real-world complexities underscore the need for further enhancements. By integrating advanced deep learning techniques and improving dataset diversity, future models can be made more robust and adaptable to varying data distributions.