MNIST-to-SVHN Domain Adaption

Paul Aristidou Olivier Lapabe-Goastat

April 15, 2024

1 Introduction

In this project, we tackle the challenge of domain adaptation between two significantly different digit image datasets: MNIST and SVHN. The source domain, MNIST, consists of labeled black-and-white images of handwritten digits. The target domain, SVHN, comprises unlabeled color images of house numbers in natural scenes, which introduces a complex, real-world scenario.

Necessity of Domain Adaptation: Effective domain adaptation is crucial due to the significant differences in visual appearance and background complexity between MNIST and SVHN. Direct application of a model trained on MNIST typically results in poor performance on SVHN due to these distribution discrepancies.

Domain Adaptation Techniques Employed: To bridge this domain gap, we explored several adaptation methods:

1. **Discrepancy-Based Methods:** These techniques aim to reduce the feature distribution disparities between the source and target domains.
2. **CyCADA:** This approach utilizes adversarial and cycle-consistency losses to adapt features effectively from MNIST for use on SVHN.
3. **Pseudo-Labeling:** We generate pseudo-labels for SVHN using predictions from a model trained on MNIST, enabling further training that enhances adaptation.

Our goal is to develop the best model, capable of recognizing MNIST digits and also effectively classify digits in SVHN images. By leveraging these domain adaptation strategies, we aim to manage the unlabeled nature of the target dataset and improve cross-domain applicability and performance.

2 Domain adaptation models and results

Let us start with the results:

Table 1: Accuracy Results of tested Domain Adaptation Models

Model	M-to-S	M-to-S with Data Aug.	S-to-M
Discrepancy	20.49%	18.04%	97.18%
CyCADA	42.54%	32.92%	75.79%
Pseudo-labeling	41.27%	49.68%	91.47%

This table presents the accuracy results for three domain adaptation models across different scenarios: MNIST to SVHN (M-to-S), MNIST to SVHN with Data Augmentation (M-to-S with Data Aug.), and SVHN to MNIST (S-to-M).

The results reveal that the Pseudo-labeling model, particularly with data augmentation, achieves the highest performance in the M-to-S adaptation, recording an accuracy of 49.68%. This is still below the current state of the art, which stands at 66% for similar tasks using advanced pseudo-labeling techniques. Despite this gap, the improvement highlights the effectiveness of incorporating realistic

variations through data augmentation, which likely helps the model better generalize from the digit-centric MNIST to the more diverse and complex SVHN. Pseudo-labeling's success with augmentation in the M-to-S scenario suggests that this method is particularly well-suited to bridging the gap between the structured, simple backgrounds of MNIST and the varied, often cluttered backgrounds of SVHN.

The CyCADA model experiences a decrease in performance with data augmentation in the M-to-S adaptation, from 42.54% to 32.92%. This decrease might suggest that the types of augmentations used could be misaligned with the features relevant to the SVHN dataset, potentially adding noise rather than aiding the model's learning process.

The Discrepancy-based model shows consistent performance degradation with data augmentation, further underscoring the importance of choosing appropriate augmentation strategies that reflect the target domain's characteristics.

Detailed descriptions of the data augmentation techniques used, along with their specific configurations, are provided in the annex.

2.1 Model 1: Discrepancy model

The discrepancy model is described in the article "Maximum Classifier Discrepancy for Unsupervised Domain Adaptation" by Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. [SWUH18]

The model structure comprises a generator and 2 classifiers. The generator will bring both domains closer, while the classifiers will ensure that the decision boundaries are accurate for both domains.

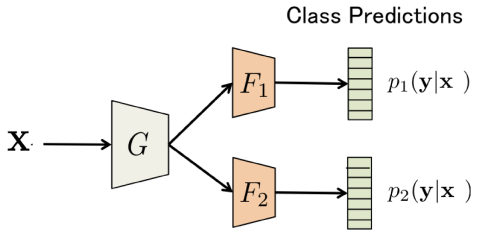

Figure 1: Discrepancy model - Structure

In order to adapt both domains MNIST and SVHN, the goal is to bring, for each class, the distribution of both domains SVHN and MNIST as close as possible (see figure below from article). The way to train the model is to follow these 3 steps for each iteration:

1. **Step 1:** We train the model on the Source dataset so that the decision boundaries are defined according to the Source. The loss that we want to minimize is the cross-entropy for both outputs :

$$\mathcal{L}(X_s, Y_s) = -E_{(x_s,y_s)\sim(X_s,Y_s)} \sum_{k=1}^{K} \mathbf{1}_{[k=y_s]} \log p(y|X_s) \tag{1}$$

2. **Step 2:** We fix the generator and train the classifiers to maximize the discrepancy between both outputs for Target data, while still minimizing the cross-entropy for both outputs for Source data. On this step, there is no domain adaptation per se, only the decision boundaries are adapated. Goal of this operation is to detect the Target samples and that the decision boundary avoids the Target data that are too far away from the Source data, while still keeping good fidelity to the Source data.

$$\min_{F_1,F_2} \mathcal{L}(X_s, Y_s) - \mathcal{L}_{\text{adv}}(X_t). \tag{2}$$

$$\mathcal{L}_{\text{adv}}(X_t) = E_{X_t\sim X_t}[d(p_1(y|X_t), p_2(y|X_t))] \tag{3}$$

3. **Step 3:** We fix the classifiers and train the generator to minimize the discrepancy between both outputs for Target data. On this step, the decision boundaries remain fix, we work purely on the domain adaptation. Goal of this step is to bring both domains as close as possible, by training the generators to minimize the discrepancy between both outputs.

$$\min_{G} \mathcal{L}_{\text{adv}}(X_t). \tag{4}$$

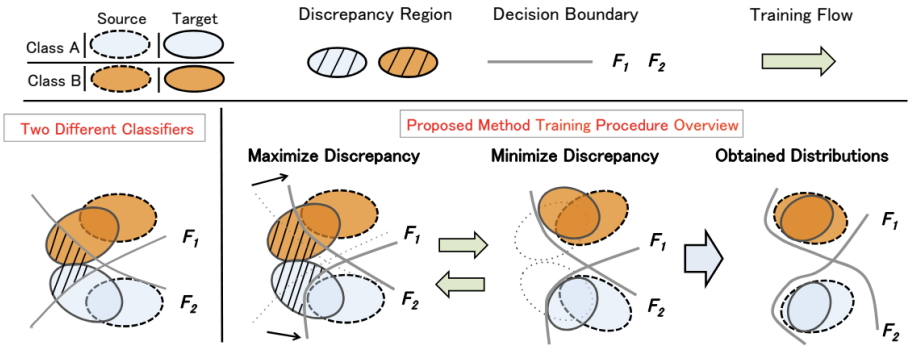

Figure 2: Discrepancy model - Training procedure and impact on distributions

2.2 Model 2: CyCADA

The CyCADA model is described in the article "CyCADA: Cycle-Consistent Adversarial Domain Adaptation" by Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, Trevor Darrell. [HTP+17]

CyCADA is a model designed for adapting data from a source domain to a target domain, particularly useful when direct labeled data in the target domain is scarce or unavailable. The core idea of CyCADA is to leverage both generative adversarial networks (GANs) and cycle-consistency to modify features or images from the source domain so that they appear as if they come from the target domain, while still preserving their original semantic content. This allows for effective domain adaptation without the need for labeled data in the target domain.

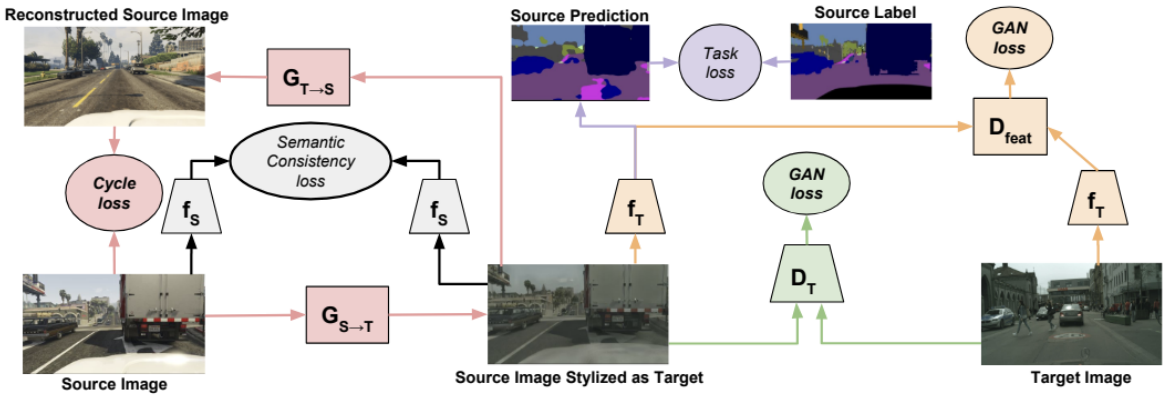

Figure 3: CyCADA model - Structure

First, we start by pre-training a task net f_S that can label MNIST.

Then, CyCADA (max-)minimizes five different types of losses to achieve this adaptation. Once this model is trained, only the task network f_T is used to label SVHN images.

1. **x2 Cycle-consistency loss (in red):** To ensure that the transformation process preserves the key attributes of the original data, a cycle-consistency loss is used. This loss ensures that data converted from the source to the target domain can be converted back to the original source data without significant loss of information. This loss is minimized twice, on the MNIST-to-SVHN and on the SVHN-to-MNIST directions.

$$\begin{aligned}\mathcal{L}_{cyc}(G_{S\to T}, G_{T\to S}, X_S, X_T) =& E_{x_s\sim X_S}\left[\|G_{T\to S}(G_{S\to T}(x_s)) - x_s\|_1\right] \\ &+ E_{x_t\sim X_T}\left[\|G_{S\to T}(G_{T\to S}(x_t)) - x_t\|_1\right]\end{aligned}$$

2. **x2 Semantic Consistency loss (in grey):** This loss ensures that the transformation process does not alter the semantic meaning of the data. For instance, in image adaptation, semantic

consistency loss would ensure that specific objects in the images remain identifiable and correctly labeled, regardless of the domain. This loss is minimized twice, on the MNIST-to-SVHN and on the SVHN-to-MNIST directions.

$$\begin{aligned}\mathcal{L}_{sem}(G_{S\to T}, G_{T\to S}, X_S, X_T, f_S) = & \, L_{task}(f_S, G_{T\to S}(X_T), p(f_S, X_T)) \\ & + L_{task}(f_S, G_{S\to T}(X_S), p(f_S, X_S))\end{aligned}$$

with :

$$\mathcal{L}_{task}(f_S, X_S, Y_S) = - E_{(x_s,y_s)\sim(X_S,Y_S)} \left[\sum_{k=1}^{K} \mathbf{1}_{[k=y_s]} \log \left(\sigma(f_S^{(k)}(x_s)) \right) \right]$$

3. **x2 Adversarial loss on image level (in green):** This loss ensures that the transformed source domain data is as close as possible from the target domain data, fooling a discriminator that is trained to distinguish between the two. This helps in aligning the data distributions of the two domains on the image level. This loss is max-minimized twice, on the MNIST-to-SVHN and on the SVHN-to-MNIST directions.

$$\mathcal{L}_{GAN}(G_{S\to T}, D_T, X_T, X_S) + \mathcal{L}_{GAN}(G_{T\to S}, D_S, X_S, X_T)$$

with:

$$\mathcal{L}_{GAN}(G_{S\to T}, D_T, X_T, X_S) = E_{x_t\sim X_T} \left[\log D_T(x_t)\right] + E_{x_s\sim X_S} \left[\log(1 - D_T(G_{S\to T}(x_s)))\right]$$

4. **Adversarial loss on feature level (in orange):** This loss ensures that the transformed source domain data is as close as possible from the target domain data, fooling a discriminator that is trained to distinguish between the two. This helps in aligning the data distributions of the two domains on the feature level.

$$\mathcal{L}_{GAN}(f_T, D_{feat}, f_S(G_{S\to T}(X_S)), X_T)$$

5. **Source task loss (in purple):** This loss directly ensures that the predicted label for the target is as close as possible to the label of the source. It is the loss for the model that will be used for the inferences (when willing to label SVHN images).

$$\mathcal{L}_{task}(f_T, G_{S\to T}(X_S), Y_S)$$

Through these five losses, CyCADA effectively bridges the gap between different domains, enabling models to perform well even when direct training on the target domain isn't feasible.

$$\begin{aligned}\mathcal{L}_{CyCADA}(f_T, X_S, X_T, Y_S, G_{S\to T}, G_{T\to S}, D_S, D_T) = & \, \mathcal{L}_{cyc}(G_{S\to T}, G_{T\to S}, X_S, X_T) \\ & + \mathcal{L}_{sem}(G_{S\to T}, G_{T\to S}, X_S, X_T, f_S) \\ & + \mathcal{L}_{GAN}(G_{S\to T}, D_T, X_T, X_S) + \mathcal{L}_{GAN}(G_{T\to S}, D_S, X_S, X_T) \\ & + \mathcal{L}_{GAN}(f_T, D_{feat}, f_S(G_{S\to T}(X_S)), X_T) \\ & + \mathcal{L}_{task}(f_T, G_{S\to T}(X_S), Y_S)\end{aligned}$$

Once the CyCADA model is trained, we use f_T^* to labelize SVHN images:

$$f_T^* = \arg\min_{f_T} \min_{G_{S\to T}} \max_{D_S, D_T} \mathcal{L}_{CyCADA}(f_T, X_S, X_T, Y_S, G_{S\to T}, G_{T\to S}, D_S, D_T)$$

Here are 2 examples of obtained MNIST-to-SVHN adpatation :

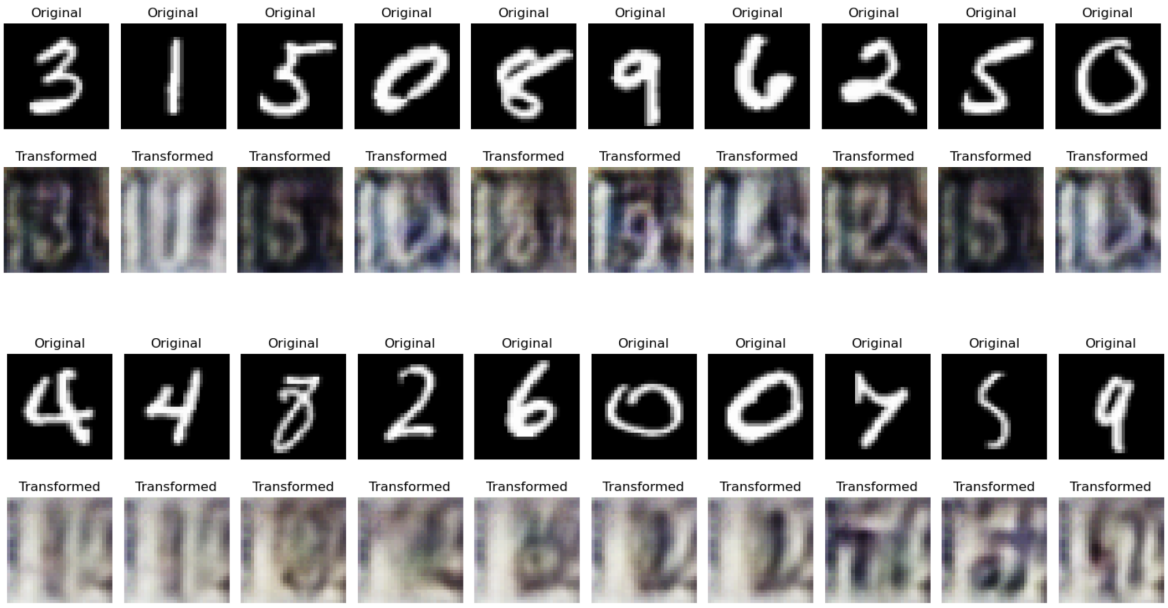

Figure 4: CyCADA model - MNIST-to-SVHN adaptation

We observe how the MNIST are adpated into a SVHN style. We observe that the adaptation works better into a specific SVHN style (first picture) and the adaptation works less into another SVHN style (second picture). It seems that some SVHN styles are better represented in the MIST-to-SVHN adaptation than others (probably the ones that are closer to MNIST).

2.3 Model 3: Pseudo-labeling

Pseudo-labeling leverages the model's own high-confidence predictions to label unlabeled data, which are then used as if they were true labels. This semi-supervised technique helps in utilizing unlabeled data effectively, particularly in domain adaptation scenarios where labeled data in the target domain is limited or unavailable.

We used the model described in the article "Domain Adaptation Using Pseudo Labels" by Sachin Chhabra, Hemanth Venkateswara, Baoxin Li.[CVL24]

Initially trained on the MNIST dataset, the model is subsequently used to generate predictions for the SVHN dataset. These predictions, when exceeding a certain confidence threshold, are deemed pseudo-labels. The process harnesses the predictive power of the model to facilitate training on the target domain, thereby bridging the gap between different domains.

Training Process: The training involves several key steps, outlined as follows:

1. **Step 1: Initial Training on Source Data (MNIST)**: The model is trained using the standard supervised learning approach. The loss function for this phase is the cross-entropy loss, defined as:
$$\mathcal{L}_{\text{CE}}(Y, \hat{Y}) = -\sum_i Y_i \log(\hat{Y}_i) \tag{5}$$
where Y represents the true labels and $\hat{Y}$ the predicted probabilities.

2. **Step 2: Generating Pseudo-Labels on Target Data (SVHN)**: Pseudo-labels are generated by applying the trained model to the unlabeled SVHN dataset and selecting predictions with high confidence:
$$\hat{Y}_{\text{pseudo}} = \arg\max(p(y|x)), \text{ for } p(y|x) > \text{threshold} \tag{6}$$

3. **Step 3: Retraining with Pseudo-Labels**: The model is then retrained on a combination of the original labeled data from MNIST and the pseudo-labeled data from SVHN. The combined training utilizes the following modified loss function:
$$\mathcal{L} = \mathcal{L}_{\text{CE}}(Y, \hat{Y}) + \lambda \mathcal{L}_{\text{CE}}(\hat{Y}_{\text{pseudo}}, \hat{Y}) \tag{7}$$

where λ is a weighting factor that controls the influence of the pseudo-labels in the loss function.

This approach iteratively refines the model's ability to adapt to the target domain, effectively utilizing the unlabeled data to enhance the model's performance and generalization.

The underlying theory of pseudo-labeling assumes that high-confidence predictions by a well-trained model are likely correct and can be used to teach the model about a new domain. This self-training mechanism enables the model to improve iteratively through exposure to both true labels and its own predictions.

The loss functions incorporated in the training process ensure that the model learns to not only fit the initial domain but also generalize to the new domain using the knowledge distilled from its own outputs.

3 Conclusion

Achievements:

1. **Pseudo-labeling with Data Augmentation:** This technique emerged as the most effective, especially when augmented with realistic transformations. It achieved a notable improvement in MNIST-to-SVHN adaptation, underscoring the value of simulating target domain characteristics in training data.

2. **Discrepancy-Based Models and CyCADA:** While these methods provided significant insights into domain adaptation, their performance varied, with CyCADA experiencing a decrease under certain augmentation scenarios, indicating a sensitivity to the types of augmentations applied.

Challenges and Insights:

- **Data Augmentation:** The efficacy of data augmentation highlighted its dual role in enhancing model generalization and potentially introducing noise. The findings emphasize the necessity for carefully tuned augmentation strategies that align closely with the target domain characteristics.

- **Model Sensitivity:** The varying performances of models under different augmentation setups suggest that domain adaptation strategies may need to be specifically tailored to accommodate the unique features and challenges of the target data.

Future Directions:

- **Exploration of New Models:** Continued experimentation with other domain adaptation models like SBADAGAN and DIRT-T could provide better reasults as the state of the art suggests.

- **Refinement of Augmentation Techniques:** Further refining augmentation techniques to better simulate the target domain's characteristics could enhance model robustness.

4 Annex

In the Annex are all details about the different models that were used.

4.1 Details on Model 1: Discrepancy model

For the MNIST-to-SVHN direction, we used more simple models for the generator and the classifiers that fits better, as MNIST pictures are "more simple" than SVHN, which allows us to avoid overfitting.

4.1.1 Generator

Operation	**Configuration**
Convolution + BN + ReLU	3, 4, 5x5, stride=1, pad=2
Max Pooling	3x3, stride=2, padding=1
Convolution + BN + ReLU	4, 8, 5x5, stride=1, pad=2
Max Pooling	3x3, stride=2, padding=1
Convolution + BN + ReLU	8, 16, 5x5, stride=1, pad=2
Flatten	-
Fully Connected + BN + ReLU + Dropout	1024 to 512

Table 2: Layer configuration of the generator - MNIST-to-SVHN

Operation	**Configuration**
Convolution + BN + ReLU	3, 64, 5x5, stride=1, pad=2
Max Pooling	3x3, stride=2, padding=1
Convolution + BN + ReLU	64, 64, 5x5, stride=1, pad=2
Max Pooling	3x3, stride=2, padding=1
Convolution + BN + ReLU	64, 128, 5x5, stride=1, pad=2
Flatten	-
Fully Connected + BN + ReLU + Dropout	8192 to 3072

Table 3: Layer configuration of the generator - SVHN-to-MNIST

4.1.2 Classifier

Operation	**Configuration**
Fully Connected + BN + ReLU	512 to 512
Fully Connected + BN + ReLU	512 to 256
Fully Connected	256 to 10

Table 4: Layer configuration of the classifiers - MNIST-to-SVHN

Operation	**Configuration**
Fully Connected + BN + ReLU	3072 to 3072
Fully Connected + BN + ReLU	3072 to 2048
Fully Connected	2048 to 10

Table 5: Layer configuration of the classifiers - SVHN-to-MNIST

4.1.3 Hyperparameters

Parameter	Value
Batch Size	128
Learning Rate	0.0002
# trainings of step C per iteration	4x (vs. 1x for step A and B)
Number of Epochs	50
Weight Decay for the Adam optimizer	0.0005

Table 6: Configuration of hyperparameters

4.2 Details on Model 2: CyCADA

4.2.1 Generator

Operation	Configuration
Convolution 1 + Instance Norm + ReLU	3, 64, 7x7, stride=1, pad=3
Convolution 2 + Instance Norm + ReLU	64, 128, 3x3, stride=2, pad=1
Convolution 3 + Instance Norm + ReLU	128, 256, 3x3, stride=2, pad=1
Convolution 4 + Instance Norm + ReLU	256, 256, 3x3, stride=1, pad=1
Convolution 5 + Instance Norm	256, 256, 3x3, stride=1, pad=1
Residual Addition from Convolution 3	-
Convolution 6 + Instance Norm + ReLU	256, 256, 3x3, stride=1, pad=1
Convolution 7 + Instance Norm	256, 256, 3x3, stride=1, pad=1
Residual Addition from Convolution 5	-
Transpose Convolution + Instance Norm + ReLU	256, 128, 3x3, stride=2, pad=1, output pad=1
Transpose Convolution + Instance Norm + ReLU	128, 64, 3x3, stride=2, pad=1, output pad=1
Transpose Convolution + Tanh	64, 3, 7x7, stride=1, pad=3

Table 7: Layer configuration of the generators

4.2.2 Task classifier

Operation	Configuration
Convolution	3, 20, 5x5, stride=1, pad=0
Max Pooling + ReLU	2x2, stride=2
Convolution + Dropout	20, 50, 5x5, stride=1, pad=0
Max Pooling + ReLU	2x2, stride=2
Fully Connected + ReLU + Dropout	50*5*5 to 500
Fully Connected	500 to 10

Table 8: Layer configuration of the task classifiers

4.2.3 Image discriminator

Operation	Configuration
Convolution + Leaky ReLU	3, 64, 4x4, stride=2, pad=2
Convolution + Instance Norm + Leaky ReLU (0.2)	64, 128, 4x4, stride=2, pad=2
Convolution + Instance Norm + Leaky ReLU (0.2)	128, 256, 4x4, stride=2, pad=2
Convolution + Instance Norm + Leaky ReLU (0.2)	256, 512, 4x4, stride=2, pad=2
Convolution + Instance Norm + Leaky ReLU (0.2)	512, 512, 4x4, stride=2, pad=2
Convolution + Sigmoid	512, 1, 2x2, stride=1, pad=0

Table 9: Layer configuration of the image discriminators

4.2.4 Feature discriminator

Operation	Configuration
Fully Connected + ReLU	10 to 500
Fully Connected + ReLU	500 to 500
Fully Connected + Softmax	500 to 2

Table 10: Layer configuration of the feature discriminator

4.2.5 Hyperparameters

Parameter	Value
Batch Size	128
Learning Rate	0.00001
Number of Epochs	100

Table 11: Configuration of hyperparameters

4.3 Details on Model 3: Pseudo-labeling

4.3.1 Model Architecture

The model used in the pseudo-labeling process is a convolutional neural network (CNN) with the following layers:

Layer Type	Configuration
Input Layer	3 color channels (RGB), 32x32 pixels
Convolution + ReLU	32 filters, 3x3 kernel, stride=1, padding=1
Batch Normalization	32 features
Max Pooling	2x2, stride=2
Convolution + ReLU	64 filters, 3x3 kernel, stride=1, padding=1
Batch Normalization	64 features
Max Pooling	2x2, stride=2
Convolution + ReLU	128 filters, 3x3 kernel, stride=1, padding=1
Batch Normalization	128 features
Max Pooling	2x2, stride=2
Fully Connected + ReLU	2048 to 512 units
Dropout	p=0.5
Fully Connected	512 to 10 classes (output)

Table 12: CNN architecture for pseudo-labeling

4.3.2 Hyperparameters and Training Details

Training was conducted using the following hyperparameters:

- **Optimizer**: Adam with an initial learning rate of 0.001.
- **Loss Function**: Cross-entropy loss for both initial training and training with pseudo-labels.
- **Batch Size**: 64 for both training and validation phases.
- **Learning Rate Scheduler**: StepLR, reducing the learning rate by a factor of 0.1 every 10 epochs.
- **Number of Epochs**: Initially 10 epochs on MNIST, followed by 5 iterative training phases with pseudo-labels on SVHN.

4.3.3 Experiments and Observations

A series of experiments were conducted to optimize the threshold for pseudo-label acceptance and the balance between the source and target domain data. The optimal threshold for pseudo-label confidence was found to be 0.95, ensuring high reliability of the pseudo-labels used in training. Modifications to the model's architecture, such as additional convolutional layers and adjustments to dropout rates, were tested to improve its robustness and accuracy.

4.4 Details on Data Augmentation

4.4.1 Augmentation Techniques Used

Two main data augmentation techniques were applied to the MNIST dataset: affine transformations and digit colorization. Notably, these augmentations were applied randomly at each epoch, introducing a diverse range of transformations that prevent the model from overfitting to specific orientations, scales, or colors, thus enhancing its robustness and generalization capabilities across domains.

- **Affine Transformations:** These transformations help to simulate the variability in positioning and orientation seen in the SVHN dataset. We applied:
  1. **Rotation:** Images were rotated by up to $\pm 30°$. This range was chosen to introduce variability while ensuring that digits remain recognizable.
  2. **Translation:** Images were translated horizontally and vertically by up to 10% of the image size to mimic the misalignment commonly found in natural scenes.

3. **Scaling:** Images were scaled by factors ranging from 0.8 to 1.2, allowing for slight zoom in and out effects that mimic the natural size variations found in SVHN.

- **Digit Colorization:** To address the color discrepancy between the grayscale MNIST images and the colorful SVHN images, a colorization process was implemented. This process randomly adds color tints to the digits, thus simulating the color diversity of SVHN.

Below are examples showing the effects of the affine transformations and colorization on MNIST images.

Figure 5: Data Augmentation - Example of random affine transformation and colorization

4.4.2 Lessons Learned from Augmentation Experiments

Through the process of implementing these augmentations, several key insights were gained:

- Increasing the rotation angle beyond 30° often led to unrecognizable digits, negatively impacting the model's ability to learn meaningful features.
- Excessive translation resulted in partial digit truncations, which similarly degraded model performance.
- The balance in colorization intensity was critical; overly vivid colors overshadowed the digit details, while too subtle tints made little difference in adapting to the color complexity of SVHN.

These augmentations, particularly when well-parameterized, proved effective in reducing the domain gap, as evidenced by improved performance in preliminary domain adaptation tasks.

References

[CVL24] Sachin Chhabra, Hemanth Venkateswara, and Baoxin Li. Domain adaptation using pseudo labels, 2024.

[HTP+17] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017.

[SWUH18] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation, 2018.